OPTICAL MUSIC RECOGNITION INFRASTRUCTURE FOR LARGE-SCALE MUSIC DOCUMENT ANALYSIS

Andrew Noah Hankinson

Music Technology Area
Department of Music Research
Schulich School of Music

McGill University, Montréal, Québec

December 2014

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Contents

List of Figures

List of Tables

Abstract

Optical music recognition (OMR) is a technology that can transform large quantities of music document page images, acquired through mass digitization efforts, into searchable and retrievable document entities. Current OMR systems, supporting tools, and best practices are not designed to support large-scale music digitization and recognition projects. The dissertation will present several novel techniques for large-scale music document recognition programs, with the goal of creating large, searchable symbolic music corpora derived from libraries' and archives' print and manuscript collections.

Optical character recognition (OCR) is used in large-scale text recognition projects to automatically transcribe the content of page images. These transcriptions maintain the relationship between the textual content and its spatial location on a page image, and are used to build systems capable of simultaneously delivering a visual representation consistent with the original source (i.e., the digital image), while also allowing page-level textual content search and retrieval. While this technique has been developed and deployed in several prominent text digitization efforts (such as Google Books and JSTOR), no similar techniques have been developed for music document recognition and retrieval. A discussion of the emergence of this technique in textual recognition will be the starting point of the dissertation as a precursor to discussing techniques for developing similar approaches for music document image retrieval.

Optical music recognition may be described as a process of discrete steps, involving computer-based tools developed by experts in image processing, machine learning and classification, music theory and palaeography. Scientific workflow systems are proposed as a model for building bespoke music recognition systems by co-ordinating the flow of images and data between different image processing and music recognition tools.

In addition to workflow-based OMR, the dissertation will describe several tools that have been built to support large-scale OMR systems development. Rodan, a prototype system for workflow-based OMR, is presented along with tools that support OMR, digital symbolic music representation, digital image navigation, and web-based crowd-sourced correction.

Finally, the dissertation will describe three prototype projects built to demonstrate the use of rich content-based data for image-based navigation and retrieval for music: The *Liber usualis* project, an on-line application containing automatically transcribed and searchable chant notation; and two projects involving the *Salzinnes Antiphonal* that demonstrate metadata and notation image navigation of a 16th-century manuscript source.

Creating large collections of transcribed music documents is the first step in developing globally-accessible and navigable document image collections drawn from libraries the world over, transforming how people interact with these collections and enabling new methods of content-based exploration and understanding.

Résumé

La reconnaissance optique de partitions musicales («optical music recognition» ou «OMR») est une technologie qui sert à transformer des images de partitions musicales – acquises par le biais de vastes projets de numérisation – en documents permettant la recherche de données. Cependant, les systèmes d'OMR et les outils de soutien existants, de même que les meilleures pratiques développées, ne sont pas conçus pour supporter les projets de numérisation et de reconnaissance musicale à grande échelle. La thèse présentera plusieurs techniques nouvelles destinées à des programmes de reconnaissance de documents musicaux d'envergure, techniques dont l'objectif est de permettre la recherche à travers un vaste corpus de symboles musicaux créé à partir de collections d'imprimés et de manuscrits en provenance de bibliothèques et d'archives.

La reconnaissance optique de caractères («optical character recognition» ou «OCR») est utilisée dans les projets à grande échelle de reconnaissance de texte pour transcrire automatiquement le contenu d'images de texte. Ces transcriptions maintiennent la relation entre le contenu textuel et sa position sur la page. Elles sont utilisées pour créer des systèmes capables de livrer une représentation visuelle conforme à la source originale (c'est-à-dire l'image numérique), tout en permettant la recherche du contenu textuel dans la page. Alors que cette technique a été développée et est employée dans plusieurs projets importants de numérisation de textes (tels que Google Books et JSTOR), aucune technique similaire n'a été développée pour des documents musicaux. Une présentation de l'émergence de cette technique de reconnaissance textuelle sera le point de départ de la thèse, ce qui permettra ensuite d'aborder des techniques visant le développement d'approches similaires pour les documents musicaux.

La reconnaissance optique de partitions musicales peut être décrite comme un processus d'étapes distinctes impliquant non seulement des outils informatiques développés par des experts en traitement de l'image, mais aussi l'apprentissage automatique («machine learning»), la classification, la théorie musicale et la paléographie. Des systèmes scientifiques de flux opérationnels («workflow») sont proposés comme modèles pour bâtir sur mesure des systèmes de reconnaissance de partitions musicales en coordonnant le flux d'images et de données entre les différents outils de traitement d'images et de reconnaissance de partitions.

En plus de l'OMR basé sur des flux opérationnels, la thèse décrira plusieurs outils qui ont été conçus pour supporter le développement de systèmes d'OMR à grande échelle. Le prototype Rodan – un système d'OMR basé sur le flux de travaux – sera présenté, de même que des outils qui prennent en charge l'OMR, la représentation de musique symbolique numérique, la navigation d'image numérique et la correction en externalisation ouverte (crowdsourcing).

Finalement, la thèse présentera trois prototypes conçus pour la récupération d'images et la navigation à travers des ouvrages musicaux au contenu imposant : le projet *Liber usualis* – une application en ligne permettant la recherche à travers un corpus de transcriptions automatiques de plain-chants – de même que deux projets autour du *Salzinnes Antiphonal*, qui démontrent la navigation par métadonnées et par images pour une source de notation musicale manuscrite du 16e siècle.

La création de vastes collections de transcriptions de partitions musicales est la première étape en vue du développement de collections d'images de documents navigables et accessibles à tous provenant de bibliothèques à travers le monde, transformant la façon dont les utilisateurs interagissent avec ces collections et permettant de nouvelles méthodes d'exploration et de compréhension de leurs contenus.

Acknowledgements

To say that this dissertation was a collaborative effort would be a gross understatement. My supervisor Ichiro Fujinaga has gone far above and beyond the call of duty in supporting this work. Julie Cumming, my co-supervisor, has likewise been constant in her support and enthusiasm. I am especially indebted to both.

My friends and colleagues, past and present, have made the Distributed Digital Music Archives and Libraries lab a wonderful and supportive environment. Jason Hockman, Johanna Devaney, John Ashley Burgoyne, Darryl Cameron, Andrew Horwitz, Cory McKay, Alastair Porter, Gabriel Vigliensoni, Greg Burlet, Jordan Smith, Beinan Li, Jessica Thompson, and Hannah Robertson have provided encouragement and support, both academic and personal, and I am deeply indebted to them. Laurent Pugin deserves special mention as a long-time colleague, collaborator, and sounding board.

I am especially grateful to the students employed in the course of the many projects and development efforts: Christopher Antila, Ryan Bannon, Laurier Baribeau, Ruth Berkow, Remi Chiu, Andrew Fogarty, Wei Gao, Mahtab Ghamsari, Peter Henderson, Anton Khelou, Jamie Klassen, Saining Li, Wendy Liu, Evan Magoni, Mikaela Miller, Lillio Mok, Laura Osterlund, Deepanjan Roy, Harry Simmonds, Caylin Smith, Brian Stern, and Timothy Wilfong. Their specific contributions will be described later in the acknowledgements, but I wish to express my gratitude here.

Perry Roland has been an inspiration and a friend. Much of my work would not have been possible without his long-time, constant, and unswerving dedication to the Music Encoding Initiative. Erin Mayhood, Daniel Pitti, and the folks at the University of Virginia Music Library graciously hosted me on two occasions, first in April of 2011, and then in May of 2012 as a visiting scholar, where I was able to work alongside Perry and pester him about a great many things.

Many others have provided invaluable support through these seven years: Eric Lewis and Wilson Blakely at the Improvisation, Community and Social Practice project; Cynthia Leive, Cathy Martin, Brian McMillan, and Andrew Senior at the Marvin Duchow Music Library; Jamil Ragep and Andrew Staples in the Islamic Studies department; Richard Freedman and Micah Walter at Haverford College. Alexis Risler provided the French translation of my abstract. Thank you all.

The Social Sciences and Humanities Research Council has been my primary source of funding. The Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT) has supported this work with several grants and travel funding. The Schulich School of Music has also provided several travel grants.

My parents, Bruce and Deborah, have been a constant source of love and support on this long road, and I could not have done any of this without them. Finally, Catherine Motuz has had the patience of a saint and a limitless amount of love to endure the day-to-day challenges in writing this dissertation.

Specific Contributions

This section will outline the contributions of the individuals involved in the tools and projects discussed in the dissertation. I was fully or partially responsible for planning and supervising each of these initiatives, and in many cases have contributed significantly to their implementation; I have noted where I have not been the primary developer. A detailed log of software contributions are maintained in the commit logs available on each project's GitHub page.

Software

Rodan

Project page: https://github.com/DDMAL/Rodan
Preliminary investigations into building a workflow-based OMR system were undertaken using the Taverna environment. Gabriel Vigli-

ensoni was responsible for the integration of Gamera and the IMPACT Interoperability Framework within Taverna.

Rodan has undergone two revisions. The first version was built in the summer of 2012. I proposed the design of the system, including the workflow and job system, to our development team. The principle software authors for this version were Alastair Porter and Wendy Liu, with contributions from Brian Stern, Anton Khelou, and Peter Henderson.

In the fall of 2012 and winter of 2013, I rewrote large portions of the Rodan server system, including the design and development of the REST API and a full re-write of the workflow running components. This also included the initial development of the browser-based client application, written using the Cappuccino toolkit. In the summer of 2013 my work was expanded by Ryan Bannon, Deepanjan Roy, and Laurier Baribeau.

Gabriel Vigliensoni was the principle author of the Rodan pitch-finding system, the core of the neume recognition process, under the supervision of John Ashley Burgoyne and myself. The first version of this was developed to support the *Liber usualis* project, with several refinements contributed in the years following. A set of supporting Gamera modules was developed by Anton Khelou and myself (`https://github.com/DDMAL/rodan_plugins`). The document pre-processing toolkit was developed by John Ashley Burgoyne and Yue Ouyang in 2009 as part of the *Gamut for Early Music on Microfilms* (GEMM) project. Laurier Baribeau was responsible for creating the first web version of the Gamera classifier interface under my supervision.

Tim Wilfong's tireless efforts at testing the Rodan interface are particularly noted. He has been responsible for constantly "kicking the tires" in the process of creating transcriptions of the Salzinnes and St. Gall manuscript sources.

Work on Rodan continues. In May 2014, Ryan Bannon, Ruth Berkow, and Harry Simmonds have been developing a new workflow execution system and user interface. Discussion of these components are not included in the dissertation.

Neume Notation Encoding

I have had two visiting research positions at the University of Virginia, the first in April of 2011, and the second in June of 2012. While there I worked closely with Perry Roland on several MEI-based tools. Perry provided advice and feedback during the creation of the Solesmes neume notation encoding schema in 2011, and the first versions of the SibMEI plugin in 2012.

LibMEI

Project page: https://github.com/DDMAL/libmei

I developed the initial version of LibMEI as a Python module to support the *Liber usualis* project (`https://github.com/ahankinson/pymei/`). Afterwards it was decided to port this module to C++ to make it more readily accessible in a wide range of software applications. Alastair Porter and I designed this library, with Alastair responsible for establishing most of the design patterns and testing framework. Mahtab Ghamsari-Esfahani provided the initial implementation of the library following these designs, and Alastair and I have continued this work. To make it possible to use LibMEI in a Python environment I wrote the Boost Python wrappers. Gregory Burlet contributed several significant bug-fixes to this code.

Jamie Klassen contributed the first version of the library auto-generation code, which I re-wrote and expanded to support the generation of C++, Python, and the Sibelius ManuScript language.

SibMEI

Project page: https://github.com/DuChemin/sibmei

I developed the initial version of the Sibelius plugin while on my visiting research position at the University of Virginia in 2012. Over the summer of 2013 this work was expanded by Micah Walter at Haverford College as part of the Digital Du Chemin project under the supervision of Richard Freedman. In the winter of 2014, Zoltan Komives contributed several bug fixes.

Diva

Project page: https://github.com/DDMAL/diva.js

I developed the initial version of Diva.js (then called "DocumentViewer") with Laurent Pugin in 2008 as part of a project sponsored by the Swiss Répertoire International des Sources Musicale (RISM) working group. This was demonstrated at two international conferences: IAML (Amsterdam) and ISMIR (Kobe) in 2009. The first version was developed using the ExtJS JavaScript library. This version contained the core tile and page layout algorithms.

To support the *Liber usualis* project we ported the viewer to jQuery. I contributed the initial project setup and design, however Wendy Liu developed the bulk of the current Diva.js system. Her contributions are particularly noteworthy since they include dynamic DOM element loading and unloading, browser-based image manipulation, the grid layout view, JPEG2000 support, and an optimized method for loading the page layout information from the server. Wendy was also responsible for the development of the image manipulation plug-ins. Since her departure from our lab, I have maintained and continued to develop the Diva project. Since May 2014 Evan Magoni has continued developing Diva.

Neon.js

Project page: https://github.com/DDMAL/Neon.js

The overall design of Neon.js, and the choice to create a web-based OMR correction editor, was the product of several conversations between John Ashley Burgoyne and myself. We assigned Gregory Burlet to build this component. He is responsible for the complete implementation of the editor, including both the client and server interactions, the user interface, and the "neumify" algorithms.

Interactive Correction Interfaces

Project page: https://github.com/DDMAL/js-image-suite

The interactive interfaces for correction were initially developed as a standalone library, the JS Image Suite, which I initiated. These components were implemented and developed primarily by Brian Stern and Wendy Liu. In the summer of 2013, Ryan Bannon and I rewrote significant portions of these components to integrate them into a newer version of Rodan.

Projects

Liber usualis

Project page: http://ddmal.music.mcgill.ca/liber/

The *Liber usualis* project involved several people over a number of months. The project was jointly managed by John Ashley Burgoyne and myself under the supervision of Ichiro Fujinaga. Remi Chiu developed the neume classification system for the Gamera shape classifier, and performed a large portion of the shape correction work. Mikaela Miller, Catherine Motuz, Laura Osterlund, and Caylin Smith were our long-suffering correctors for the project. Gabriel Vigliensoni and I wrote much of the custom code necessary for "gluing" the tools together. Jamie Klassen wrote the custom neume correction interface for Aruspix. Saining Li handled the OCR tools with the Ocropus toolkit. Wendy Liu, Saining Li, and Alastair Porter handled the development of the web interface. Jessica Thompson wrote the search indexing system, first using ElasticSearch and CouchDB, and then using Solr.

Salzinnes I

Project page: http://ddmal.music.mcgill.ca/salzinnes/

The first version of the Salzinnes web interface used personal photos from Judy Dietz, who also supplied the cataloguing data through the CANTUS project. Judy was also responsible for putting us in touch with the staff at the Canadian Conservation Institute (CCI), who sent us high-resolution versions of the manuscript for use in the second phase of the Salzinnes project.

Catherine Motuz and I worked on a system for "massaging" the CANTUS data into a human-readable format and indexing this data in Solr. Saining Li, Wendy Liu, and Alastair Porter developed the website interface.

Salzinnes II

Project page: http://cantus.simssa.ca

The first version of the Salzinnes II project was developed for a presentation at the International Association of Music Libraries conference in Vienna in July 2013. For this version, I re-wrote the website interface to incorporate the CCI images and provide an interface for searching both text and notation. Ryan Bannon developed the processing workflows in Rodan and managed the correction of twenty demonstration pages.

In May 2014 development of a new manuscript viewing interface was started. Andrew Horwitz and Andrew Fogarty have developed this system as a more formalized version of the previous one. The demonstration page data has been loaded into this system.

Publications

Several papers and presentations have emerged in the course of this dissertation, and are cited within. In each of the publications where I am listed as first author I have been the primary contributor to that paper or presentation.

1. Introduction

Large-scale digitization initiatives, including the well known Google Books project, have digitized over 20 million items—books, magazines, newspapers—held in libraries and archives worldwide. This mass digitization effort has created billions of digital page images. For textual materials, optical character recognition (OCR) is used to automatically transcribe these sources, enabling the search and retrieval of these page images. Users can retrieve a specific page from a book, held in a library thousands of kilometres away, using full-text search engines. The immediacy of access and comprehensive coverage of these collections are facilitating new ways of interacting with library collections, prompting people to develop new insights into the information contained in digitized print media the world over.

Optical music recognition (OMR) provides the same automated transcription possibilities as OCR, but existing music recognition systems are not designed to process the large quantities of page images produced by mass digitization efforts. Current OMR systems are designed to provide music document transcription for personal use, and scaling this process to cover large collections or corpora requires rethinking the design of OMR tools to provide high-throughput systems capable of digitizing and transcribing billions of pages.

The starting point, and central hypothesis, for the dissertation is that the development of systems which maintained alignment between images and transcribed text was a major catalyst in allowing large-scale recognition initiatives to move forward. The development of the alignment technique replaced a transcription-centred view of OCR, where the text was extracted from the image with no reference to the original, with one where the transcription was conceived of as an "invisible layer" on the image. This layer was used in retrieval systems to provide full-text search, while still allowing users to read the document from the original image. This technique had the effect of mitigating the impact of error-prone transcriptions on a user's ability to read and understand the text content of the original document.

The same transition beyond a transcription-centric recognition model has not occurred for OMR. In current OMR systems, and in the research literature of the field, the emphasis is placed on achieving complete accuracy in the recognition process, while a labour-intensive manual post-correction stage is seen as a necessary component to creating a useable transcription. While accuracy is an important consideration in any recognition system, relying on automated systems, of any sophistication, to create error-free transcriptions of millions of page images is an impossible task. OMR systems that preserve alignment between transcribed musical content and page images can allow "good-enough" automated transcriptions to be used in a document retrieval system without the need for extensive and costly human involvement in the process.

Creating a large corpus of digitally transcribed music documents is the first step in constructing a symbolic music retrieval system for the purposes of large-scale search and analysis. How users will interact with these systems, however, is not yet known. In what forms can users express musical queries? How are musical queries posed to a retrieval system? What kinds of responses will they expect when they search? These are important questions that must be addressed in future work, but are outside of the scope of this dissertation. For the present purposes, I will focus on methods of building digital corpora. In the conclusion of this dissertation, I will provide further thoughts on the issue of search and retrieval.

1.1 Background

This section will include a short introduction to recognition technologies and some of the common techniques and challenges shared by OMR and OCR. It will describe the emergence of mass digitization efforts for text and the techniques developed in these efforts that pertain to issues of accuracy and scale, introducing issues discussed further in Chapter 2. Applications of similar technologies to music document recognition will be provided as a prelude to longer discussions in Chapters 3 and 4.

1.1.1 Introduction to Recognition Technologies

OCR and OMR are similar technologies. Both are used to extract a symbolic representation from a visual representation of a page (i.e., a digital image), applying a combination of image processing, symbol matching, and heuristic knowledge to transcribe the content of the image. For OCR, the general technique is to match a collection of pixels to known character shapes to produce text suitable for searching and manipulating (e.g, editing, reformatting) on a computer. For OMR the goal is much the same: to extract music notation symbols in order to recreate a structural representation suitable for searching, editing, or auditory synthesis. Both technologies provide automated transcription functions, turning pixels into symbols—textual or musical—thereby circumventing manual transcription.

A symbolic representation is the most convenient format for manipulating the textual or musical content of an image. Symbolic representations of text and music may be edited, indexed, searched, retrieved, manipulated, and analyzed. Symbolic representations of music may also be heard with the application of audio synthesis techniques. Without optical recognition to extract the graphical content of digital images into a symbolic representation, a human is required to transcribe image contents; an extremely labour-intensive process. Automated recognition, using OMR and OCR, can be used to provide a representation that a computer can understand, and in significantly less time than a human could perform the same task.

While automated recognition is a relatively quick process to perform on an image, it is also an error-prone process. Each step required in processing an image for recognition, including rotation, de-skewing, de-warping, or binarization, can introduce errors. Damage to the physical item, including mould, insects, or simply wear and tear, can also have an impact on the readability of the image and cause problems with the recognition process. Document layout processing requires a computer to automatically segment an image into regions that contain different page elements, such as advertisements, tables, pictures, columns, or headings. Mis-identification of a region will likely lead to unusable recognition results.

Automated recognition of music documents is a significantly more complex task than textual recognition, as described in detail by Bainbridge and Bell (2001). In music documents, document structure and symbol are tightly bound. Staves, clefs, measures, systems are all page elements that do not immediately translate into a performable elements, yet they are crucial to providing a structure in which the notes and rests—the elements that most directly translate to the "content" of piece of music—can hold meaning. Translating a visual indication of structure into a well-defined and logical symbolic representation of that structure such that it may be understood and manipulated in a computer is at the heart of the challenge of optical recognition technologies.

Despite the complexities associated with recognition, OCR and OMR systems are both capable of transcribing their respective contents with high levels of accuracy. For OCR, transcription of content is frequently well above 90% accuracy for most documents, sometimes approaching 99% accuracy (for certain types of documents) (Rice et al. 1996). While there are no widely accepted standards for OMR systems comparison and evaluation (Bellini et al. 2007; Byrd et al. 2010), accuracy rates better than 90% are frequently reported for individual systems (Bainbridge and Wijaya 1999; Pugin et al. 2007a; Rebelo et al. 2010). Yet even with these high accuracy rates, the amount of work required to create completely accurate transcriptions can be significant.

> Despite high accuracy rates that rival other OMR systems, counted in terms of editing cost, the computer reconstructed music required on average 10 editing operations per 100 notes, which represents considerable effort in correcting the data generated. Accuracy rates quoted as percentages, therefore, can be deceptive in the amount of human effort they require to correct (Bainbridge and Wijaya 1999, 477).

To create error-free transcriptions, the results of recognition software may be proofread by a person, who can correct the errors to produce an accurate representation of the symbolic content of the page (Newby and Franks 2003). While this produces a faithful transcription of the content, the introduction of a person into the process comes at a significant expense, measured in both time and money. Compared to computers, people are slow, expensive, and easily bored with repetitive and menial tasks. Yet in a transcription-only context, there is a dependence on a person to provide a corrected version of the page. While people are the most effective means of producing accurate transcriptions, their involvement does not scale to large operations. One of the largest book digitization initiatives, the HathiTrust (Christenson 2011), has over three billion page images in its collection, representing over 11 million volumes. Even assuming an average (and very optimistic) correction time of just one minute per page, it would take over 74 *centuries* of constant effort for a single human to correct the errors on every page of the collection. Spreading this over several humans does little to negate the effects of scale in any practical sense. Employing 1,000 people in full-time jobs it would still take over 40 years to manually correct every page image. While the size of music collections is comparatively much smaller than text collections, the effects of scale and accuracy remain the same.

The enormity of the task presents the first challenge that must be addressed when discussing large-scale music document recognition. While optical recognition systems are accurate enough to be a useful tool in automatically transcribing the contents of images, the complexities involved in accurately transcribing a structured representation from these images all but guarantee that this process will *never* be error free. For text recognition projects, a solution to this challenge was developed in which recognition results were aligned with the original page images, offering opportunities for interacting with both representations simultaneously. This same approach has not yet been adopted by music recognition projects.

1.1.2 Symbolic and Visual Alignment for Text

The late 1980s and early 1990s were a period of tremendous growth in computer systems. Advances in technology and the decreasing cost of computing contributed to rapid growth in personal computer systems. Graphical user interfaces (GUIs), an alternative to the older command-line text interfaces, allowed users to view and manipulate information visually. For the first time since the development of the computer system, there were systems capable of displaying digital images on personal computers without exotic or expensive hardware.

Concurrent with these developments, organizations began to investigate, and invest in, the creation of large digital document collections motivated by the advantages of digitized media over physical media. A digital book could be used by many people at once, did not require library shelf space, did not deteriorate with intensive use, and could be distributed to remote locations over electronic networks very quickly. Some of the first efforts at building document image collections attempted to use sophisticated document analysis systems to provide complete and accurate translations of print collections into structured digital formats (Myka and Guntzer 1993; Farrow et al. 1994; Myka 1994). These systems demonstrated this was not a feasible approach without extensive human intervention.

Visual interfaces that could simultaneously display and manipulate both images and text were adopted by document digitization efforts (Story et al. 1992; Atkinson and Stackpole 1995). Systems began emerging that could display OCR-derived text, while simultaneously offering users the ability to view the original page image (Loughry 1993). The creation of this technique allowed digitization efforts to separate the visual representation from the automatically-transcribed OCR content, providing users with a familiar "interface" to the content (i.e., the page image) while still retaining the advantages of automatically-transcribed information—the indexing, searching, and retrieving of pages within digitized materials—by maintaining a relationship between the page image and its extracted text content (figure 1.1).

> While we do use [OCR] to obtain the text for searches, the OCR results are never visible to the user, but are spatially associated with the location of the text on each page image… The main reasons for displaying the image and not the ASCII is that most readers are already familiar with general graphical layout conventions, especially those used in journals they have read before, so they can rely on this familiarity when they scan the page images for content. A second practical reason is that OCR and page layout analysis results are not guaranteed to be flawless. Rather than display OCR errors to the users, the problem is sidestepped by showing only the image, and "hiding" the associated OCR text and layout planes (Hoffman et al. 1993, 447).

While complete layout alignment was not strictly necessary to allow retrieval of a page image, many systems adopted a symbolic representation format that preserved information about the locations of each word on the page. This allowed a search system to retrieve the exact page, and to highlight the occurrence of the word or search phrase on the image.

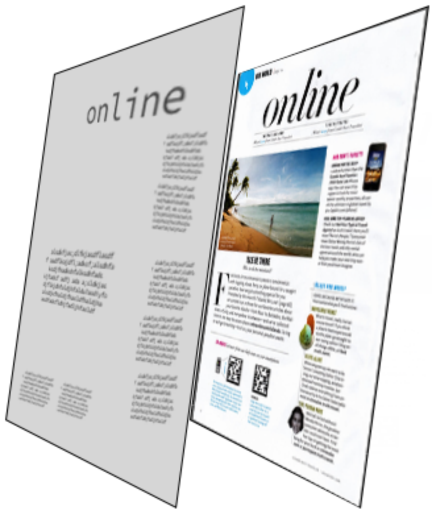


Figure 1.1: Text-image overlay, maintaining correspondence between OCR text and original page image.

This new technique marked a departure from the conventional use of OCR to transcribe content. The use of invisible OCR text offered an important navigation method, permitting users to retrieve and interact with a page image based on its textual content. It also mitigated the presence of inaccurate transcription by allowing humans to interact with both the symbolic and visual representations of a page simultaneously. This technique was originally developed and described in several prototype or small-scale digitization and recognition initiatives (Nagy et al. 1992; Hoffman et al. 1993; Lesk 1994), but was gradually adopted by large digitization initiatives such as JSTOR (Schonfeld 2003), the British Library newspapers digitization projects (Deegan et al. 2001), and eventually, the Google Book digitization project (Sherman 2003) and the HathiTrust (Christenson 2011).

The emergence of the image overlay technique is a largely un-remarked development in the history of large-scale recognition. Chapter 2 of the dissertation will provide an in-depth history of this development. By addressing, or more properly, side-stepping the issue of providing completely accurate transcriptions, this method has allowed for the creation of document recognition workflows with minimal human involvement necessary for correcting the texts. This creates a faster, more efficient, and practical solution to the complexities inherent in recognition techniques, allowing page images to be processed in a fraction of the time but still providing a way of searching, navigating, and viewing documents within a retrieval system.

1.1.3 Alignment and Music Document Recognition

Despite the widespread application of page image and symbolic alignment in textual document digitization efforts, it has not been widely adopted by music document recognition efforts. OMR systems, symbolic representation file formats, and software for viewing and manipulating this type of file have not been integrated in such a way as to make this possible. Adopting the alignment technique from text recognition systems for use with music documents may be the first step in allowing music recognition efforts to scale to millions of pages. It provides a way to grant access to page-level symbolic music content but mitigates the need for providing completely accurate transcriptions, removing the dependency on human correction from the process and, in doing so, provides a way to process the large quantities of page images produced by mass music document digitization programs.

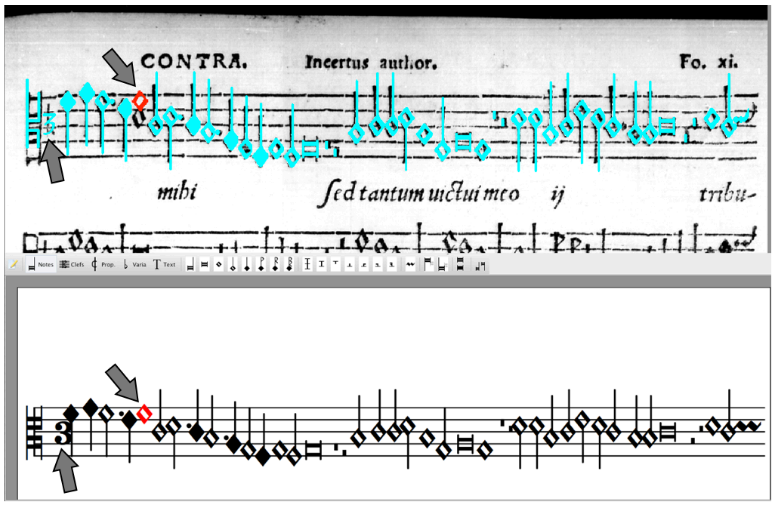


Figure 1.2: Image and music correspondence, shown in the Aruspix OMR notation editor

An example of music and image alignment is given in figure 1.2, where the OMR-derived results are shown in overlay on the original image (top), but the musical representation can be viewed and edited (bottom). Despite errors in the OMR (specifically, the key signature has been mis-recognized as a time signature and the fifth note has an incorrect pitch), the user may still view the notation on the original page image.

Addressing symbol and image correspondence is the first step towards building large-scale music document recognition systems. Several other design considerations will be proposed in this dissertation, addressing issues of workflow, throughput, accuracy, collaboration, and representation of OMR results in the context of large-scale music document recognition processes.

1.1.4 Scientific workflow systems

Scientific workflow software systems (Taylor et al. 2007) have been developed to manage a wide variety of software-based processes in a variety of disciplines, including astronomy, bioinformatics, and data mining. These workflow systems allow users to design a data processing pipeline by connecting processing tools together using a directed graph metaphor, where nodes represent tools for manipulating a particular data representation, and edges represent the flow of data between the tools. The output of one tool becomes the input to a subsequent process.

Prior to the development of scientific workflow systems, tools and processes were combined within a contained and static system, typically useful for a single purpose and only modifiable by a developer experienced with the software. These scripts were typically only useful for a single experiment, and are characterized by Gil (2007) as "unassisted workflow systems." A formalized workflow system, where tools and data flow can be designed, allows non-developers to assemble heterogeneous collections of software tools in processes that are reusable, re-purposable, and repeatable.

Workflow systems have been used in historical (pre-20th century) document recognition (Dogan et al. 2010; Neudecker et al. 2011) as part of the Improving Access to Text (IMPACT) project. They demonstrated that adopting a scientific workflow system approach for historical text recognition allowed users to design and execute customized document recognition processes using tools drawn from a number of different software systems, into a single bespoke recognition system that was tailored to the particular needs of those document images. Workflow systems change document recognition systems from a single piece of tightly-integrated software to a system where many different tools, from many different toolkits, can be assembled and re-organized to build bespoke recognition systems customized for a specific application. As a result, scientific workflow systems present new opportunities for creating and executing the OMR process using procedures drawn from a number of tools.

Designing and executing the OMR process in a workflow system can provide high-throughput systems to automatically process large quantities of page images in a consistent and repeatable manner.

New analysis methods may be introduced in the workflow to customize the OMR process for different repertoires, or to improve the recognition of existing workflows. Different techniques may be automatically compared and evaluated for their effectiveness on a given repertoire. A workflow approach can be used to build customized recognition systems that can address issues of complexity and accuracy by providing customizable processes to document images.

1.1.5 Distributed OMR

Conventional OMR systems are designed and built as desktop applications, operated by a single user on a single workstation. Naturally, this limits the number and location of the people using these applications (i.e., one person per application, sitting at one workstation), and also limits the amount of computing power available to process page images to that of the desktop computer being used.

Web applications have demonstrated that it is possible to create sophisticated software systems that behave like conventional desktop applications, but that are accessible over a network connection through a web browser (Garrett 2005). Users may be geographically distributed, yet may access and operate a common OMR system, opening up new avenues for collaboration.

The web browser interface creates a separation between the control surface (i.e., the interface) and the underlying systems that perform image manipulation and recognition tasks. These tasks can be handled by a remote server system, containing significantly more hardware resources (faster and larger storage systems, more RAM, faster CPUs) than desktop or laptop systems. Server systems can be networked together, creating a "cluster" of computers operating in concert using distributed and parallel computing techniques. OMR tasks may be allocated to many machines for simultaneous processing. OMR systems built for these clusters can utilize all available computing resources, rather than just the resources available in a single workstation, resulting in faster processing.

A distributed OMR system offers new ways of creating systems for large-scale OMR initiatives that go beyond the conventional design of a desktop OMR system. A workflow-based OMR system, operated through a web browser and with a theoretically limitless amount of computing power behind it opens up new possibilities for processing large numbers of page images, and making human interaction in the process more efficient.

1.1.6 Collaborative OMR

Aligning a visual representation with an automatically-transcribed symbolic representation of a page image does not imply accepting bad recognition results in perpetuity. Error correction may be addressed in other ways that do not impose a dependency on human involvement prior to making a document searchable and accessible.

Fully automated methods may be employed to re-process document page images as new and improved OMR techniques become available. Advances in automated image processing, new symbol classification techniques, or entirely new workflows may be applied to collections of music document images, offering opportunities for incrementally, and automatically, increasing automated transcription accuracy.

"Crowdsourced" correction is another method of dealing with inaccuracies. Crowdsourcing solicits members of a population to contribute small, easily-accomplished tasks. Uncorrected recognition results may be corrected by users interested in contributing their time and effort to improving recognized documents. Crowdsourced correction efforts have been employed, some with great success, by several large-scale text digitization efforts, giving the general public access to uncorrected OCR results and soliciting their corrections (von Ahn et al. 2008; Holley 2009b). The same approach has only started to emerge for music recognition efforts (Dalitz and Crawford 2013).

Combining human contributions with computational techniques may help address issues of building recognition systems that can process a wide variety of music documents. Symbol shapes can vary from one publisher to another, and some repertoires, including those written in older notation and in newer *avant-garde* notation, can introduce new symbols to a standardized library, or change how a symbol is to be interpreted within a musical structure. Adaptive optical music recognition (Fujinaga 1996a) uses human-supplied transcriptions as "ground-truth" data, employing corrected music recognition results to "learn" how to interpret new symbols. Human feedback can be also used to build a recognition system that improves in accuracy as more pages are corrected (Pugin et al. 2007a). In a networked environment, annotations and corrections may be collected from one user and employed in the recognition processes of other members of that network, allowing a collaborative approach to building constantly-improving adaptive recognition systems.

1.1.7 Representation of Results

A symbolic representation preserves musical structures in a way that can be read and manipulated in a software environment. There have been many efforts at encoding symbolic music notation in a computer-readable format beginning with systems using computers to analyze symbolic music (Kostka 1971; Selfridge-Field 1997a)

In general, a symbolic representation format strives to maintain the properties of, and relationships between, musical symbols. A clef symbol, for example, has a shape and position on a musical staff, but it also dictates the relationships between pitches on that staff by establishing the pitch of a reference staff line (e.g., a treble clef establishes the pitch of the second staff line, "G",and by extension, all of the other lines and spaces).

A common representation format provides a means of interchange between software systems that manipulate or process symbolic music notation. The output of OMR systems have typically been used for input into notation editing software, such as MusicXML (Good 2009) or for audio synthesis using MIDI (MIDI Manufacturers Association 1996). These representation formats are not capable of storing the relationships between a visual page representation (i.e., a digital image) and the symbolic content of that page, and are most often designed to encode conventional Western music notation (CWMN) music notation. The MIDI format, to take a familiar encoding example, can encode a pitch and duration of a note event; however, it cannot encode the differences between enharmonic equivalents—a C♯4 and a D♭4 are represented by the same numeric value (61), making it impossible to distinguish once it has been decoded. Some formats, such as the Lilypond format (Nienhuys and Nieuwenhuizen 2003) prioritize visual layout over preserving music structure, while others, such as the Kern format (Huron 1997) prioritize the musical structure without reference to a visual realization of the notation.

In a large-scale recognition initiative, a wide variety of music documents, demonstrating different styles of notation, are likely to be encountered. Music notation has evolved over time according to varying needs to communicate performance information (Selfridge-Field 1997b). As a result, accurately preserving musical structure in an encoding system across many different repertoires can be complex. A format that is designed to support CWMN exclusively cannot support the musical semantics present in mensural notation without requiring a translation from one system into another. Imposing modern notational conventions onto music that was conceived with a different understanding of rhythm and meter, for instance, is a "lossy" conversion, removing information present in the original and implicitly adding information (Bent 1994).

The need to "natively" represent the semantics of a system of notation must be balanced with the practical complexities of using many incompatible methods of encoding music notation. To encode the particular semantics of multiple systems of notation, the conventional approach has been to create dedicated encoding systems representing one type of notation at the exclusion of all others. Formats for encoding neume notation (Barton 2002), mensural notation (Dumitrescu and Berchum 2009), or lute tablature (Crawford 1991) have been developed to accommodate the particular needs of each type of notation, but these initiatives have resulted in "data silos," where a particular encoding system was developed for a specific purpose, but eventually abandoned once the project ended. These corpora cannot be easily decoded without access to the tools used to create them. When dealing with large and varied music document corpora, a balance must be found between accommodating variation in musical semantics between all sources of notation and the practical consideration of having a common representation for the purposes of compatibility between software components and an eye towards future maintenance of the schema.

The Music Encoding Initiative (MEI) (Roland 2009) is a symbolic encoding format that is especially suited for use in large-scale document recognition projects. It provides methods for maintaining correspondence between visual and symbolic representations, and its modular design provides mechanisms for encoding repertoire-specific notation features within a broader document encoding framework, allowing all MEI-encoded documents to share common encoding practices, while simultaneously providing a means of customizing the format to accommodate different repertoires. It is the flexibility, adaptability, and community-developed nature of the MEI system of notation encoding that makes this an especially suitable format for building large corpora of automatically-transcribed music notation.

1.2 Structure of the Dissertation

This dissertation is organized into eight chapters, including this introduction. Chapter 2 describes the emergence of OCR as a document navigation technology that allows document images to be automatically transcribed and made available for electronic retrieval.

Chapter 3 provides an overview of the OMR process, and describes the current state of this field. Issues of evaluation, previous efforts at large-scale processing, server-based OMR systems, and OMR techniques for medieval and renaissance music notation provide background to specific aspects of OMR that will be expanded in later chapters.

The first part of chapter 4 introduces several new applications of existing techniques and technologies to OMR, each offering possibilities for addressing aspects of scaling music recognition to accommodate the processing of large document image collections. Workflow systems and software present opportunities for managing and creating custom OMR systems to process a wide variety of music documents. Distributed OMR is introduced as a means of dividing the tasks associated with OMR processing among many different participants, human and computer. Collaborative OMR is introduced to describe methods of allowing several people to operate OMR systems collectively, using shared systems to build collectively accessible resources. Finally, a section on networked adaptive OMR describes systems that use distributed and collaborative OMR processes to build data sets suitable for constant re-integration into a machine-learning approach (i.e., adaptive) to OMR.

Chapter 4 continues with a discussion of structural music notation representation and its importance in the OMR process. The Music Encoding Initiative (MEI) is described, and two unique aspects of this encoding format are highlighted in detail, explaining why the MEI format is the most appropriate choice for encoding OMR-derived symbolic music.

Several tools and techniques have been developed in the course of this research program. Chapter 5 provides details on Rodan, a prototype OMR workflow system, and describes the underlying design and specific implementation details. Chapter 6 describes several tools built to support large-scale music document recognition. Chapter 7 presents details on three prototype projects, each dedicated to investigating aspects of music document image navigation: The *Liber usualis* project, and two prototype systems built around the *Salzinnes Antiphonal*. Finally, chapter 8 is the conclusion of this dissertation and includes a discussion of opportunities for future work.

1.3 Original Contributions

This work will contribute several novel techniques and tools useful for designing and implementing large-scale music document recognition systems. Chapter 2 is, as far as I know, the first comprehensive history of the emergence of image and symbol alignment in large-scale recognition projects. Given the pervasiveness of this technique in digitization projects, this represents an important contribution to the general literature for document image analysis and large digital document libraries, beyond the specific focus of this dissertation on music document digitization.

The application of scientific workflow software to the OMR process has not been previously explored. Likewise, descriptions of distributed, collaborative, and networked adaptive OMR represent unique contributions to guide and direct future OMR systems development and applications.

The tools presented in the latter half of this dissertation were developed in the course of this research program. Each tool was built to address a perceived need in the overall infrastructure required for large-scale OMR and, as such, each represent unique contributions. Furthermore, all systems developed as part of this research are available as open-source software, allowing them to be re-used and built upon in future work.

Finally, the projects presented in chapter 7 of this dissertation represent first attempts at building systems focused on large-scale image-based music document navigation. Although they have been developed as prototypes, they are publicly available on the Internet and our web server logs indicate that they are in use by the general public. While these systems were built through a collective effort of several teams of people, I had a significant role in their design and execution, managing or co-managing the teams and directing their development efforts. This dissertation represents the first comprehensive history and description of all three projects. Full disclosure about individual contributions are provided in the acknowledgements of the dissertation.

Creating large collections of transcribed music documents is the first step in developing globally-accessible and navigable document image collections drawn from libraries the world over, transforming how people interact with these collections and enabling new methods of content-based exploration and understanding.

2.
Large-Scale Text Digitization

This chapter will focus on the development of optical character recognition (OCR) systems for large-scale text document digitization programs. OCR is the method used to extract a symbolic representation (i.e., text) from digital images. This chapter will trace the growing importance of images in OCR projects from the time the subject was first conceived to the present. This discussion will serve to highlight how the alignment of image and text has been the underlying technology that allows digitization projects with text recognition components to scale and, in comparison, how the same techniques have not been adopted in OMR research and development.

For early computer systems, digital images were difficult and expensive to manipulate with a computer. They were extremely inefficient representations of textual content, so they were used as a method of translating a physical page to a digital format for the purposes of transcription, and then discarded. As OCR technology developed it became increasingly clear that a "perfect" OCR system capable of accurately capturing all document content was not feasible. Computers advanced to a point where working with and manipulating images was not as cumbersome, so new methods began emerging that used both the original image and the OCR-transcribed text in document retrieval systems. This mitigated the effects of imperfect OCR by presenting humans with an accurate representation of the page image, while still providing a means of searching and navigating physical document collections in an electronic context. This technique was adopted by several large document digitization initiatives. In this chapter, I will argue that employing the original image, rather than relying on a purely transcriptive approach, is an under-recognized but key development in the creation of large-scale document databases.

The first section of this chapter (§ 2.1) covers the beginning of the modern computing era from the 1950s through the 1980s, reviewing some of the earliest developments in digital text document retrieval, as well as the emergence of OCR. The next section (§ 2.2) looks at the emergence of document image retrieval systems, coinciding with the development of affordable consumer-level computer systems, the Internet, and a realization that a purely transcriptive approach to OCR was unlikely to produce satisfactory results. After these developments, a number of large initiatives emerged (§ 2.3) that represent the beginning of the industrial digitization era. These projects would set the stage for the true mass-digitization projects (§ 2.4) that have demonstrated that it is possible to search and access a large body of printed materials held in libraries distributed around the world. Finally, a brief discussion of document text encoding formats is provided (§ 2.6).

2.1 Early Systems (1950–1990)

The beginning of the modern computing era was marked by an optimism that complex human-computer interaction problems were solvable in the near term (Samuel 1964). This era was one where practical, generally-available solutions for speech recognition, natural language processing, artificial intelligence, and "reading" machines capable of perfectly extracting text from digital images was expected to be less than twenty years away.

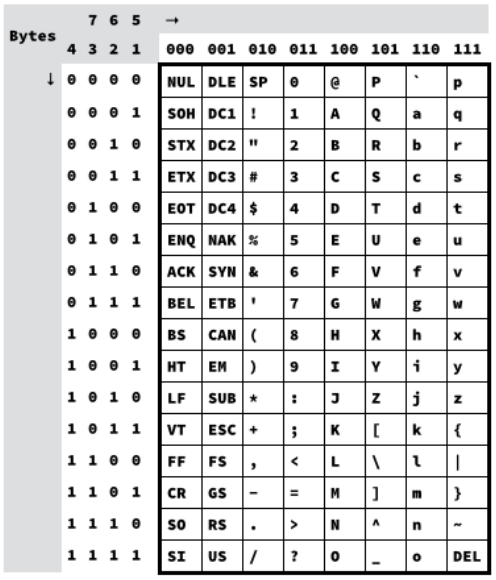
> At the beginning stage it was thought that it would be easy to develop an OCR, and the introduction of a very rigorous reading machine was expected in the 1950's. Roughly speaking, the 1950's and 1960's... were periods when researchers imagined an ideal OCR, even though they were aware of the great difficulty of the problem. Actually this is an instance of a common

> phenomenon which occurred in the research field of artificial intelligence in general (Mori et al. 1992, 1033).

The push for these reading machines, of which the technology became known as "optical character recognition," was seen as a practical solution to the post-war information explosion. Vannevar Bush's seminal article, "As we may think" (Bush 1945) imagines information tools in the office of the future and is an oft-cited article in this era, providing inspiration for many researchers in ways in which computers could help create this future. OCR was a means of converting large amounts of paper documents to an electronic format as a way of "taming" information creation and bringing about efficiencies in office work, saving time in data entry applications and co-ordinating efforts between remote locations.

> The need for fast, reliable character recognition has become more acute in recent years... Consider data storage and information retrieval alone. It is estimated that man's total accumulation of knowledge doubled during the 7 years from 1960 to 1967... This information merely augments what anyone who has recently made a literature search knows—information storage and retrieval is a very real problem with us today (Balm 1970, 152).

There were several developments in the early years that have been of lasting importance for document text encoding and retrieval. The American Standard Code for Information Interchange (ASCII) (Gorn et al. 1963), was introduced in 1963 as a solution to multiple incompatible methods of representing text across different computer systems. The ASCII standard specifies how text is stored and transmitted, representing Roman characters, punctuation, and specialized control codes with 7-bit binary integers (figure 2.1). Its adoption by large computer manufacturers made it the *lingua franca* of encoded text in computer systems, peripherals, and software. ASCII was instrumental in allowing digitized texts to be shared. Project Gutenberg, founded in 1971 (Hart 1992), was the first networked project dedicated to creating and sharing existing electronic texts among networked computers. A subsequent character encoding system, Unicode, features expanded cross-platform character encoding for over one million possible characters (The Unicode Consortium 2014).



Bytes 7 6 5	→							
4 3 2 1	000	001	010	011	100	101	110	111
↓ 0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	_	o	DEL

Figure 2.1: Table of ASCII codes

Meanwhile, other research was developing new methods of textual search and retrieval. The Cranfield experiments (Cleverdon and Keen 1966; Cleverdon et al. 1966), conducted between 1957 and 1966, were experiments that demonstrated the effectiveness of "single-term" indexing for document retrieval. Prior to these experiments, complex indexing schemes featuring controlled terms, abstract concepts (i.e., subject indexing), and hierarchical structures were presumed to provide superiour document retrieval methods, but had not been rigorously tested. In hindsight, the Cranfield experiments were a key factor in demonstrating the effectiveness of "full text" search for document retrieval.

Through the 1960s and 1970s, the types of data that could be easily manipulated by computer systems were limited. Encoded text was the easiest format to work with. It could be efficiently and easily stored, transmitted, and displayed on remote mainframe terminals. Graphics—pictures, diagrams, charts—were more cumbersome and needed to be stored and transmitted separately on analogue media such as videotape or microfilm (Haring and Roberge 1969; Knudson and Teicher 1969).

The OCR process on these machines required significant human intervention in the scanning and image manipulation process (Nagy 1968). Image processing for complete page images was slow and even the smallest images could only be processed on dedicated research machines. Using computer systems with video display terminals for any length of time was not a comfortable experience due to the poor legibility of text and flicker present on these displays (Muter et al. 1982; Mills and Weldon 1987).

2.1.1 Early OCR Systems

The origins of OCR technology extend back to the early twentieth century, with electro-mechanical character recognition methods used to convert typewritten texts into alternate representations, such as printed cards (Tauschek 1929; Handel 1931) (figure 2.2).

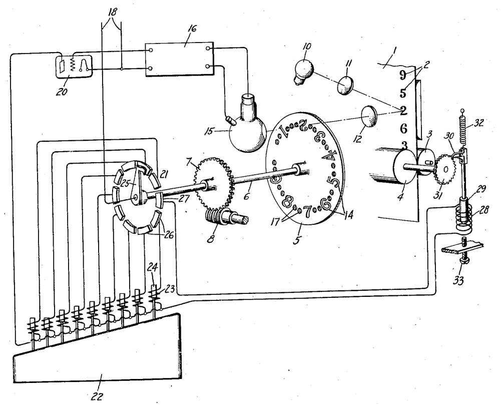


Figure 2.2: Early electro-mechanical OCR system (Handel 1931)

Computerized OCR systems were first introduced in the 1950s. A 1953 patent (Jones 1953; Shephard 1953) describes an OCR machine that was built by the Intelligent Machines Research Corporation. This machine was the first commercially-available OCR system, sold to corporations such as AT&T and Reader's Digest (Nagy et al. 1999; Martin 2007).

From the 1950s through the 1980s, computers evolved from large, warehouse-size systems to machines that could be easily moved by a single person. For OCR applications, this era began with systems with highly limited graphical capabilities, and ended with interactive visual display systems. The next three sections will describe several case studies of large-scale text digitization initiatives that exemplify these transition periods.

2.1.2 DARPA and the United States Military

For organizations, the possibility of automatically transcribing full-text versions of their paper documents showed potential for reducing costs and increasing co-ordination in large-scale efforts. Military procurement efforts were one such area, where large quantities of paperwork, including technical handbooks, engineering specifications, and regulations, could be digitized and distributed electronically to all parties involved, reducing the amount of co-ordination and duplication of effort between different sites.

The United States Military and the Defense Advanced Research Projects Agency (DARPA) began to explore the use of OCR technology in large-scale information retrieval applications in the mid-1960s and throughout the 1970s (Varley 1969; Agnew et al. 1974; McGeehan and Maddock 1975). Technical reports prepared for these organizations begin to make note of OCR as a document transcription technology that could save time and effort for co-ordinating large procurement projects. However, they also note several major challenges in making this a reality, most notably the ability to deal with multiple typefaces and graphical information such as diagrams, tables, or illustrations. Varley (1969) discusses data input techniques for converting print collections of military logistics data to computer-based information systems, focusing specifically on error detection. While the primary focus of this report is on manual transcription, the author makes a note in his "Future Trends" section:

> The future use of Optical Character Recognition (OCR) during the next decade will produce great cost saving for certain areas. The use of multi-font machines is just coming into being, and will be perfected within the next few years. (Varley 1969, 43).

Several years later in 1974, a quarterly report (Agnew et al. 1974) for the ARPANET (Advanced Research Projects Agency NETwork), a DARPA project and the precursor of the modern Internet, listed OCR as a suitable technology for managing and co-ordinating military procurement information: technical manuals, handbooks, regulations, and numerous other document types. The authors of this report identified the military equipment procurement process as an application of recently-networked computer systems, because it required references to large files of information, constant communication among many different partners, standardized forms, and many people requiring timely access to the same data.

This ARPANET report estimated that the total size of the procurement texts to be converted at around 1.7 billion characters, and calculated that manual re-typing of all materials to make these materials available for online indexing and storing would be prohibitively expensive, between 0.4¢ and 1¢ per character ($4–$10 per thousand), resulting in a total cost between $5–21 million for conversion (Agnew et al. 1974, p. C-7). By comparison, a theoretical multi-font OCR system, with a cost-per-character of 0.0046¢ ($0.04, or 4¢, per thousand), would lead to a significant cost savings. However, the authors of the report take care to note that the OCR technology required to achieve this was "just at the edge of the state of the art" and they estimate that a system capable of handling the variety of document types was still two years away. They advised the readers of their report to wait to implement their proposed system until the technology was ready. It is unknown whether their system was ever built.

The authors of the ARPANET report spent considerable time trying to devise a system that could retrieve graphical document elements—diagrams, figures, tables, and other illustrations. They imagined a hybrid system for physical retrieval of microfiche, where codes for a given graphic could be inserted in the OCR-derived text and, if the user wished to consult the graphic, they could send a request for this figure to a central operator. The request could be delivered, hours or days later, by a human operating a flying spot scanner transmitting an image of the microfiche to a user's television display, or with a facsimile machine that could queue image requests and transmit a paper version of the graphic back to the user.

2.1.3 Kurzweil Data Entry Machine

In 1976 Kurzweil Computer Products introduced an optical character recognition machine designed to help blind people read textbooks. This machine, the Kurzweil Reading Machine, was equipped with several technologies that would become important in later digitization and recognition programs: the CCD (charge coupled device) flatbed scanner, and the first omni-font character recognition software (Kleiner and Kurzweil 1977). The Reading Machine also featured the first commercially available text-to-speech system.

The technology pioneered in the Kurzweil Reading Machine was later repurposed for the Kurzweil Data Entry Machine (KDEM). The first customer of this system was the Mead Corporation for its databases, Lexis (a legal documents indexing service) and Nexis (a news indexing service) (Kurzweil 2013). The KDEM provided these organizations with the means to index a constant stream of paper documents (including newspapers and legal decisions), a task that other indexes were accomplishing by manual re-keying. The KDEM was used to transcribe these reports into an online database and make the contents available for full-text searching.

The KDEM was used in several other projects. Oxford University was the first academic institution to purchase and install a KDEM in 1980 (The cost of a KDEM was $80,000 after Kurzweil was acquired by Xerox in 1980; before that it was $100,000 (Galloway 1981)). The KDEM used an adaptive system to provide multi-font recognition functionality. An initial recognition phase allowed the system operator to train the system on the particular typeface in use (Hockey 1986). In practical use, up to 20 A4 pages could be processed in an hour.

2.1.4 National Libraries of the United States

Throughout the 1980s, several of the national libraries and archives of the United States identified OCR as a technology that had the potential to address their large-scale data entry problems. These institutions included the National Agricultural Library (NAL), National Archives and Records Administration (NARA), and the National Library of Medicine (NLM). Each of these institutions embarked on several projects that sought to use OCR to make the text of their large collections of documents searchable and retrievable.

In 1983 the NARA identified OCR as one of three potential conversion technologies that could have a significant impact on the future of archival work (the other two being speech pattern recognition and digital image scanning). Their initial study, published in their 1984 *Technology Assessment Report* (Dollar and Hooton 1984), investigated the process of converting typed index cards into electronic texts with "disappointing results." Nevertheless, work continued on automated conversion techniques and two subsequent projects in 1985 and 1986 (Allen 1987) examined OCR in greater detail specifically for two areas of interest: The conversion of typed archival finding aids into an electronic database using OCR, and the development of an OCR system for handwriting recognition. The transcription rate for the finding aids on index cards demonstrated that OCR held significant promise for the bulk conversion of these materials. The results showed that an OCR system could convert 2621 characters per minute, while manual keying of the same data converted a maximum of 223 characters per minute. The OCR system that was used could accurately read 40% of the documents with an error rate of less than 10%. The OCR data was automatically loaded into a database system to permit searching with Boolean operators and proximity searches on the full text of the transcribed finding aids.

In 1986 the NARA entered into a contract with a London (UK) company, Optiram Automation, to produce a pilot project for several of its handwritten document collections (Andre and Eaton 1988). An initial report of this initiative is filled with high expectations that the handwritten document recognition barrier was finally broken.

> In late 1984...there were reports that an English company [Optiram] had broken the handwriting barrier, with accuracy rates of 99.9 percent and fee schedules less than those for manual data entry (Allen 1987, 94).

The promise was that this technology would open up several valuable archival collections, including handwritten American Civil War records, to full-text search and retrieval. By 1988, however, this enthusiasm had been tempered by the realisation that there was a considerable amount of manual labour involved in preparing data for the OCR process, due to the wide range of variability among the documents in their collections:

> ...Optiram found numerous data anomalies and special cases not covered in the specifications. In the case of archival documents or finding aids comprising formatted document sets, some of which were generated before the age of automation, the authors

> could not have foreseen the need for consistency in data format going from one document to the next (Holmes 1988, 32).

Optiram was secretive about the processes used to produce its results, and its software was exclusively available in-house. They preferred to act as an OCR service provider rather than as a software vendor. Holmes notes that Optiram was reluctant to disclose its exact procedures, preferring to say that their technique involved a "large complement of mathematical algorithms, developed and refined over an 8-year period" (Holmes 1988 p. 29). However, Holmes deduces that they were in fact using an adaptive character recognition system (Nagy and Shelton 1966) that required a significant corpus of training data, likely supplied by individuals in-house.

By 1989 a post to the Humanist Discussion Group by Bob Kraft (1989) indicates that Optiram had hit a wall for the accuracy of their handwriting recognition technique. Kraft indicates that he had contacted Optiram for a quote to perform recognition on his great-grandfather's handwritten journal collection. He was quoted $3.25 per 1000 ASCII characters, or almost $5 per page. He then notes that he was currently paying a student $7 per hour, transcribing between five and seven pages per hour. Thus, the price of automatic transcription was almost three times as expensive, and would still need human correction to resolve ambiguous characters after recognition.

2.1.5 National Agricultural Library

During the 1980s, the National Agricultural library was involved in several image-based document digitization projects, including The Pork Industry Handbook (1984) and Laser II (ca. 1986); and The Image Transmission Project (1989). These projects are summarized by McCone (1992).

Figure 2.3 shows the retrieval system used to search and retrieve page images and text for the Laser Optical Picture Disk display system, used in early NAL digitization efforts (1984), starting with *The Pork Industry Handbook.* Of particular interest is the wide variety of peripherals needed to support image retrieval shown in figure 2.3. In this system Laserdisk was used to store analogue (not digital) copies of the graphics. A television monitor was controlled by the computer system and was used to display the document graphics when a page of text was called up on the computer display.

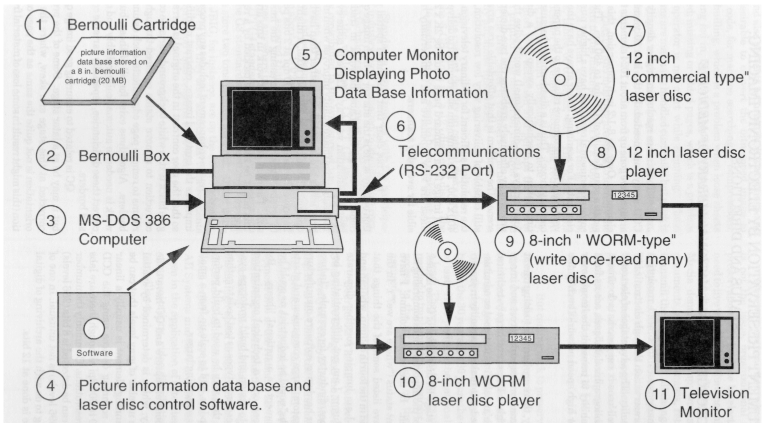

1 Bernoulli Cartridge
picture information data base stored on a 8 in. bernoulli cartridge (20 MB)
2 Bernoulli Box
3 MS-DOS 386 Computer
Software
4 Picture information data base and laser disc control software.
5 Computer Monitor Displaying Photo Data Base Information
6 Telecommunications (RS-232 Port)
7 12 inch "commercial type" laser disc
8 12 inch laser disc player
12345
9 8-inch " WORM-type" (write once-read many) laser disc
10 8-inch WORM laser disc player
11 Television Monitor

Figure 2.3: National Agricultural Library Laser Optical Picture Disc Display System (from McCone 1992)

The National Agricultural Text Digitization Project (NATDP) was founded in 1987 as a partnership between the NAL and 44 land-grant university libraries in the United States (Andre and Eaton 1988; Andre et al. 1988). This database was published on CD-ROM and distributed to the partner libraries. The initial disks used OCR to create full-text indexes of the journals, while a built-in retrieval system allowed the document page images to be retrieved from the CD-ROM. However, the recognition results (around 95% accuracy) were deemed unacceptable without labour-intensive manual correction, and as such the full text index was omitted for later disks, and the database contained just bibliographic data for each article (McCone 1992).

Figure 2.4 shows the workflow for the NATDP project (1987), and the process by which the scanned image, MARC (Machine Readable

Cataloging) cataloging record, and the OCR-extracted text were all combined on a single CD-ROM to allow the search and retrieval of document images.

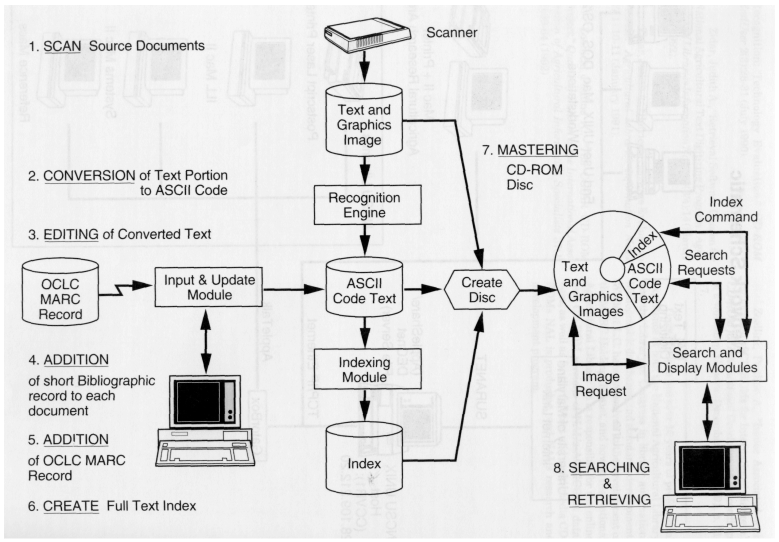


Figure 2.4: Text Digitizing at the National Agricultural Library (from McCone 1992)

2.1.6 National Library of Medicine

The National Library of Medicine (NLM) built a prototype system for scanning, storing, retrieving, and displaying biomedical documents (Henderson 1983; Cookson 1984; Thoma et al. 1985). It used bibliographic data to retrieve digitized images of documents, which were then displayed on a dedicated high-resolution CRT monitor (1728 × 2200 pixels) or printed using a laser printer. Figure 2.5 shows a block diagram of the various hardware systems involved in this initiative. The prototype system had a storage capacity of approximately 1,000 images. The user had the option of viewing the digitized images on the "soft copy" (screen) display, or sending it to a "hard copy" printer.

In an updated report on their document image retrieval system some years later, Walker and Thoma (1990) provide further details about the performance of this system in a real-world context. They noted that in their original prototype the architecture did not allow them to scale the system to add more terminals. They also had problems with error rates in transmitting compressed images over their networks. They proceeded to describe a new system that used both optical media (CD-ROM) as an image storage platform as well as a network-connected image server. When a user found a document they were interested in viewing, the search system would give the index number of the CD-ROM on which the document was available. The user could then either load the CD-ROM into a local workstation or, if their organization had a "jukebox" disk system, they could call up the images to their workstation over a local area network.

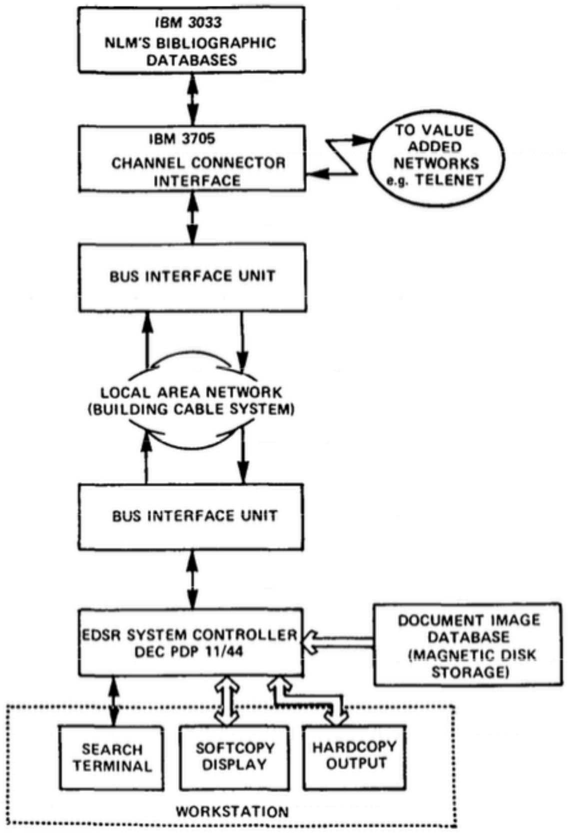

IBM 3033
NLM'S BIBLIOGRAPHIC DATABASES
IBM 3705
CHANNEL CONNECTOR INTERFACE
TO VALUE ADDED NETWORKS e.g. TELENET
BUS INTERFACE UNIT
LOCAL AREA NETWORK (BUILDING CABLE SYSTEM)
BUS INTERFACE UNIT
EDSR SYSTEM CONTROLLER DEC PDP 11/44
DOCUMENT IMAGE DATABASE (MAGNETIC DISK STORAGE)
SEARCH TERMINAL
SOFTCOPY DISPLAY
HARDCOPY OUTPUT
WORKSTATION

Figure 2.5: The National Library of Medicine Electronic Document Storage and Retrieval System (from Thoma et al. 1985)

2.1.7 Early OCR Summary

In early computer systems the textual content of page images was the easiest, most manipulable, and most useful representation of a physical document. It was the format that could be transcribed, encoded, searched, indexed, retrieved, displayed, manipulated, and shared without needing exotic hardware capabilities. By contrast, image data in early computer systems were expensive to store, transmit, and slow to manipulate. As a result, early OCR (and, by extension, early OMR) systems were not designed to preserve any relationship with the original page image. The textual content was of primary importance, and the expected arrival of the "perfect" OCR system came with the assumption that a recognition system would be capable of perfectly transcribing page text and layout faster and better than a human transcriber.

Towards the end of this era there was a Conference on Application of Scanning Methodologies for Libraries , held 17–18 November 1988 in Beltsville, Maryland (Blamberg et al. 1988). This was the same conference where the aforementioned NARA report concerning the performance of Optiram for handwritten OCR, along with similar reports on OCR from the NAL, the Library of Congress, and a number of other institutions were presented. The concluding address of this conference was provided by Robert Hayes who, among other things, expressed some doubt that the age of the "magic wand" of perfect OCR had arrived:

> Joe [Howard's][1] concept was that of a magical waving of a wand across text, from wherever it came, a journal published in India or some handwritten script, with the text automatically being converted into the desired abstracts. With all due respect to the claims of Optiram, I think that vision is not yet here.... I suspect,

1. Joe Howard was at the National Agricultural Library working on their digitization and OCR initiatives.

> though, that they have a room full of, not elves perhaps, but at least persons sitting before CRTs and keying when it is appropriate to do so. (Hayes 1988, 135)

Later, in the same address, Hayes brought forward one of the first mentions of the importance of the digital document image. He reported on several studies conducted at the University of California, Los Angeles (where he was the dean of the library and information studies faculty). These studies were aimed at trying to identify the information needs of the faculty at UCLA.

> The result of these studies has been the identification across the campus of some generic needs. It is on one of them that I want to focus, since it is directly related to this conference and is paramount among the needs that have been identified. I'm personally convinced that it will represent one of the most important components of research progress over the coming decades. It is the *digitized image* [emphasis original]. Perhaps it comes as no surprise to this audience that digitized images are important in faculty research, but I haven't seen that much attention paid to it in the library literature. (Hayes 1988, 137)

Hayes goes on to describe the full spectrum of digital imaging technologies, from the conversion of print collections to work with digital satellite images. He concludes, however, that:

> The crucial point in all of this is that you need the supercomputer to do the image processing, but you also must have the files of digitized images to be retrieved, processed, compared, and analyzed. You need to have the tools for organization and management of these files and for retrieval from them. (Hayes 1988, 140)

In this address, we see the beginnings of the emergence of the digital image in its own right as an important artifact in a text digitization initiative. Prior to this, the digital image may have been treated as a means to an end—an intermediary format required to move a text into a digital context, but was generally discarded once the text was extracted from the image. The decreasing costs associated with digital image storage, and the growing number of graphical interfaces on which images could be displayed may have played a part in seeing the image as a useful visual representation of the page, and not just data from which text could be extracted.

2.2 The Emergence of Image Navigation (1990–1995)

By the 1990s, computers available to consumers and researchers had advanced beyond storing and processing textual information. The introduction of Intel's 80486 CPU in 1989 represented a significant advance in the processing speed available in a consumer CPU (Lewis 1989). Progress in hard disk technology allowed these systems to affordably store increasingly large numbers of images (McCallum 2013). These developments coincided with the adoption of new internet-connected systems and, in businesses and universities especially, the availability of high-speed local-area networks.

Throughout the 1980s, graphical user interfaces (GUIs) had been developed that allowed computer users to interact with a computer using pictorial representations. The Xerox Alto (Thacker et al. 1982), first developed in 1973, was the first computer system that featured a bitmapped user interface and a mouse for interaction (English et al. 1967). The Apple Macintosh computer (Williams 1984), first introduced in 1984, was largely responsible for the creation of the desktop publishing industry (Gray 1986), in which documents were capable of being composed visually, rather than programmatically. By the beginning of the 1990s, operating systems featuring graphical user interfaces, such as Microsoft Windows, Apple's MacOS, NeXTSTEP, and X Windows, were widely available, and allowed users to work with scanned page images on the screen with commonly-available hardware.

In this era there were corresponding advances in document digitization and recognition technologies. At the beginning of the 1980s, even the most powerful computers were not capable of easily storing and processing large images, but by the early 1990s researchers had access to systems capable of decomposing entire page images and performing advanced layout analysis on them (Casey and Nagy 1991). CCD-based flatbed scanners were becoming increasingly affordable and available for digitization projects (Drummond and Bosma 1989).

Refinement of character recognition techniques continued, but were beginning to be integrated into systems designed to extract more than just textual information from a page image. Two fields began to emerge that dealt with *in situ* page details, rather than simple text extraction: Document image *analysis*, which was an attempt to automate the identification of document geometry, investigating methods of identifying and separating physical page components (e.g., columns, rows, lines); and document *understanding*, an attempt to automate the identification of logical document structures by investigating semantic structure within the document (e.g., identifying titles, authors, abstract regions on a page image) (Tang et al. 1991; Palowitch and Stewart 1995). Each of these technologies employed OCR as a component, but the focus of the field was increasingly moving towards methods of automatically understanding and reproducing complex document layouts.

How the original image could be used in an OCR workflow was not immediately understood. In a report to the Council on Library and Information Resources, Lesk (1990b) elaborates on the need for studying whether the page image or a "purely" structural representation composed of ASCII-encoded text with encoded layout information would better meet the needs of users of digitized texts:

There is a question as to whether even those who wish to read

> the texts will prefer images of pages to ASCII; more research is needed on this point. In general ASCII storage preserves the words in the text only, not their appearance, and some users express a need for the appearance… Some disciplines that rely highly on images and on the book as an artifact in their research will prefer image storage. In the long run, however, scholars are likely to prefer ASCII storage of text for many of their informational needs. ASCII storage permits searching, copying, and duplicating in much more powerful ways than any image storage. Online catalogs, for example, are replacing microfiche catalogs throughout the United Kingdom, and we see no libraries moving towards fiche for catalogs (unless perhaps they are moving from cards). At present, however, it's too expensive to get to full ASCII; and, for most of the relatively rarely used material considered for preservation, it is likely to remain too expensive to use ASCII until optical character recognition becomes feasible (Lesk 1990b).

The last sentence of this quotation provides evidence that the universal OCR system was still an expected outcome, and that the availability of the original image was a stop-gap measure until OCR became "feasible" or, more directly, capable of accurately transcribing and reproducing complex page layouts.

As the decade progressed, however, page images began to take an increasingly central role in document retrieval.

> Yet the scholar may prefer to work with the page images because, in general, it is the only digital representation that maintains the full information content of the original, including illustrations, layout, and uncommon markings (as in musical or mathematical notation). This is vitally important for historically significant documents such as hand-written manuscripts, illuminated books, and fine press materials, where the typography materially contributes to the value of the work. Even were the OCR to be perfect, translations into other digital representations inevitably lose information. And it is difficult to predict what needs preserving; at the extreme, it is possible that in the course of time that even the worm holes become

> significant, as a variant interpolation of the text eaten out may yield a different translation and, perhaps, a new interpretation. (Phelps and Wilensky 1996, 101)

Lesk mentions that the National Agricultural Library (NAL) had conceived of and worked on a system for image-based retrieval of text documents as early as the 1970s:

> Although OCR reaches an adequate level of performance on very clean modern printing or typing, it is not accurate enough on old print or deteriorated paper to be a replacement for the [page images]. What OCR can do, however, is provide a text to be used for indexing.... This approach has been suggested for at least two decades. It was first tried seriously at the National Agricultural Library in the mid-1970s.... (Lesk 1997, 64).

Unfortunately, no sources could be found that describe these systems, so this claim cannot be confirmed.[2]

This period of emerging image delivery and navigation systems featured several prototypes or short-lived projects as people began to explore options enabled by image storage and delivery systems. This section will trace the development of these projects, chosen because they demonstrate the unsure nature of electronic document delivery services and a period of trial, error, and experimentation.

2. In an e-mail conversation with Michael Lesk, I asked if there was a reference for this claim. While he could not remember a specific reference, he wrote: "My memory is that I was talking around the idea of 'if OCR has too many errors, use it only for indexing and only display the scanned images' and somebody told me in conversation '[The National Agricultural Library] did that' ... The person who told me was probably Jan Olsen, then librarian of the Mann Library at Cornell ...." (M. Lesk, e-mail communication with the author, 9 January 2014).

2.2.1 ADONIS: Document Image Delivery Service

ADONIS[3] was a project established by a consortium of biomedical publishers concerned with the loss of revenue and subscriptions due to the availability of photocopying devices (Compier and Campbell 1992; Grant 1994). It was one of the first large-scale document image delivery services. First established in 1980, it introduced a commercial service for libraries in 1991, after a decade of market studies and pilot projects. ADONIS delivered page images and article indexes on CD-ROM to subscribing libraries and covered 600 journals in the science, technical, and medical fields. The articles were retrievable using an index on a restricted number of fields (e.g., article title, author), but no full-text search was possible. It was hoped that providing journal subscriptions through an image-based document delivery service would encourage libraries to pay a per-item licensing fee when a user printed an article, thereby replacing the lost subscription revenue.

When ADONIS was introduced to the public the CD-ROM was the easiest and most cost-effective solution for storing and transmitting page images. However, for libraries there were fundamental problems with the delivery of document images in this format. Most notably, the storage capacity of CD-ROMS, the amount of material that required imaging, and the desire to limit the disks published per year to 50 (to reduce administrative overhead for the libraries), meant that a new image storage file format was needed to fit the required page images on a single disk. The working memory (Random Access Memory, or RAM) of computers was low (typically restricted to 640KB), presenting challenges to loading the large page images for viewing. "Multi-tasking," the ability for an operating system to run more than

3. The name ADONIS, as far as is indicated in the literature, is styled in all capitals but is not an acronym.

one program at a time, was not readily available in MS-DOS and so work-arounds needed to be devised (Compier and Campbell 1992).

Since the goal of the ADONIS project was to have users pay when they printed an article, the availability of high-speed laser printers was also a concern. Page images could take between 50 seconds and 3 minutes to print on many printers available at the time. This was deemed unacceptably slow. The introduction of larger networked printers was offered as a potential solution to this problem.

Although ADONIS was one of the first large-scale document image delivery services, it was never seen as a particularly cost-effective replacement for paper-based journals. To control licensing, it required dedicated workstations, a large "jukebox" for switching and loading CD-ROMs, and access to a networked printer. By 1996 the British Library had determined that it was not a viable service (Braid 2003). By this time, networked databases and journal access over the World Wide Web had largely replaced CD-ROM as the most promising media for document image distribution.

2.2.2 The CORE Project

The CORE (Chemical Online Retrieval Experiment) project (Lesk 1990a; Entlich et al. 1997) was undertaken at Cornell University in partnership with the American Chemical Society, Chemical Abstracts, Bell Research Laboratories, and OCLC (Online Computer Library Center). The CORE project is notable for its focus on usability and its study of how users in chemistry—a discipline noted for its equal reliance on both textual and graphical information in the form of visual chemical formulas—would interact with the contents of digitized journals.

The CORE project provided page image search of journal articles by using an existing corpus of machine-readable text, drawn from the desktop publishing files used to print the journals. Figure 2.6 illustrates the workflow in the CORE project to create a searchable representation of a digitized text from the text and microfilm images. Developed at the peak of CD-ROM acceptance, the CORE project was notable for instead delivering images and text over the network using dedicated client software for PC, Macintosh, and UNIX workstations.

In the initial project report, Lesk (1990a) identifies three main objectives for the CORE project:

1. Develop and test page analysis software to partition page images into textual, tabular, and pictorial regions.
2. Develop and test search software to search on the full-text records of the journals.
3. Test existing browsing and reading software for reading material journals on-line.

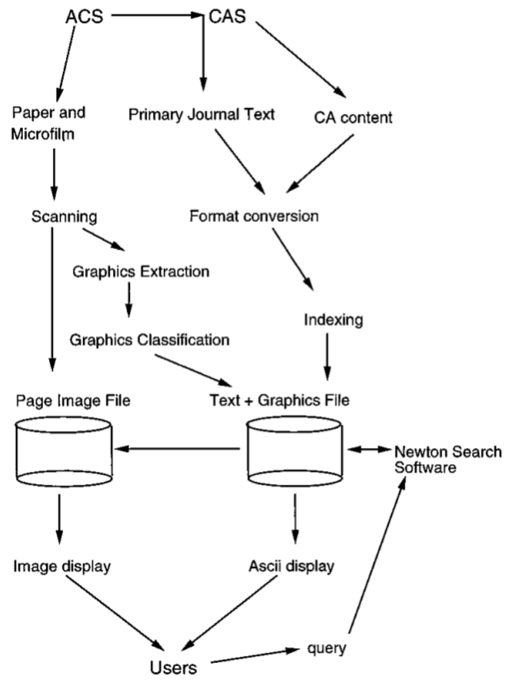


Figure 2.6: Data Flow in the CORE Project (from Entlich et al. 1997). ACS is the American Chemical Society; CAS is the Chemical Abstracts Service. These services provide textual content, which is then aligned with a digitized paper representation.

This initial project report provides the results of an evaluation of two software packages for displaying the data from their system. The "Superbook" system used ASCII page text. Graphical elements—diagrams, illustrations, chemical formulas—loaded only when a user requested it. Lesk notes that this display has the advantage of enabling on-the-fly page layout to optimally fit the display and the window size of the user's system. Since the text was provided using an ASCII representation, the Superbook display also made it possible to highlight a user's search terms in this display, allowing them to quickly identify occurrences of keywords.

The other display, "Pixlook," displayed the images of the original pages. Each article was first displayed at a resolution of 100 PPI (pixels-per-inch), 1-bit per pixel. At this resolution it was possible to make out larger page structures, but it was not suitable for reading smaller text or viewing fine details in illustrations. The user could zoom in on the page image up to 200 PPI, which Lesk notes is "normally adequate for reading." No keyword highlighting was available, but users could retrieve individual pages that contained their search terms.

Lesk offers some preliminary conclusions on the differences and effectiveness of use between the two display systems:

> Experiments with Superbook in the past have shown superior performance for searches aimed at specific target information. Not only is the searching efficient...but the display is effective at calling the user's attention to the material found. However, we have not evaluated Superbook formally in applications such as skimming, nor for the problem of known-item retrieval in a large document collection. Image-based displays may well be superior in these applications since they retain the format the users find familiar and which has been tailored over the years for effective use by chemists. (Lesk 1990a 5-3)

In the final report of the CORE project (Entlich et al. 1997) the authors provide a summary of the work and lessons derived from this project. They describe attempts at fully encoding the structural and textual representations, but conclude that data that involve more than flat ASCII are "tricky" and introduce problems in providing users with the ability to accurately read and understand complex documents, due to inadequate character set standardization for complex chemical formulae and mathematical equations. They also note the value of presenting users with the graphical components of page images such as diagrams, figures, and chemical formulae, and note that users preferred to flip through the interface looking at the graphics and diagrams as a means of navigating the documents before attempting to read the text.

2.2.3 TORPEDO

The TORPEDO project (The Optical Retrieval Project: Electronic Documents Online) was an experimental project at the US Naval Research Laboratory (NRL) to deliver digitized issues of two journals from the American Physical Society, *Physical Review Letters* and *Physical Review E* (Atkinson and Stackpole 1995). The journals were scanned and stored as TIFF files, and OCR was performed using Calera WordScan Plus software. Both the image and ASCII files were then imported into an image management database from Excalibur Technologies, EFS. This software was a server-based system with native clients for Windows, Macintosh, and UNIX systems running X Windows. Images and text were imported into the database and indexed for full-text search. The entire import process for a single journal issue took 24 hours from the time the library received the physical issue.

Users could search the journal articles from their workstations using a number of methods: browsing, full-text search, or "fuzzy" search that was apparently resilient to raw ("dirty") OCR-processed text. Field-based search of specific bibliographic data was also available.

In 1996 the decision was made to migrate the NRL's electronic document resources to a web-based delivery system (Stackpole and Atkinson 1998). By 1999, the TORPEDO system, by then called TORPEDO Ultra, was widely available within the Naval Research Laboratory as part of their larger digital library system (Stackpole and King 1999). The TORPEDO Ultra project contained over two million pages, drawn from technical reports, press releases, conference papers, and over 200,000 articles from more than 200 journals. The articles were distributed as PDF files, either created from scanned raster images or as native PDFs created by desktop publishing software as part of the issue production process.

2.2.4 RightPages

The RightPages project (O'Gorman 1992; Story et al. 1992; Hoffman et al. 1993) was an initiative by AT&T Bell Labs to develop a journal delivery system to their employees. This project was developed to deliver page images and text directly to users' workstations over a network connection. Along with Gobbledoc and Dienst (described in the next two sections) it is one of the earliest systems to feature spatially aligned OCR text with page images, rather than a non-aligned transcription of the ASCII text of the page. This spatial alignment was conceived of as an "invisible layer" of text over page images and was designed specifically to present the document in a format that was familiar to users, and to side-step the need for a "perfect" OCR system.

> While we do use [OCR] to obtain the text for searches, the OCR results are never visible to the user, but are spatially associated with the location of the text on each page image... The main reasons for displaying the image and not the ASCII is that most readers are already familiar with general graphical layout conventions, especially those used in journals they have read before, so they can rely on this familiarity when they scan the page images for content. A second practical reason is that OCR and page layout analysis results are not guaranteed to be flawless. Rather than display OCR errors to the users, the

> problem is sidestepped by showing only the image, and "hiding" the associated OCR text and layout planes. (Hoffman et al. 1993, 447).

In addition to performing OCR on the page images, the RightPages system incorporated a document layout analysis component that automatically segmented and labelled page regions. This allowed users to restrict their full-text searches to particular page elements without the need to manually enter that content into a field-based retrieval system. For example, users could restrict their searches to just article titles derived from automatic page region identification of page titles.

The results of the OCR process were saved in a plain text ASCII file, with additional data stored to identify spatial positions for each word. These encoded ASCII files were loaded into the system at runtime and parsed into C++ objects. This was the initial database system used for search and retrieval.

Users of the RightPages system interacted with the system using a X Windows-based graphical interface installed on their workstation. When a journal was digitized the document text was automatically recognized. The system would automatically alert users if the content of the article matched their interests based on pre-specified keywords in their profile. Users could open the RightPages client software to browse to the page images of the journal article where their keywords appeared. While RightPages was initially developed for use internally at AT&T, it was eventually adopted as the software underlying a journal delivery project at the University of California, the Red Sage project (Arnold et al. 1997).

The RightPages and Sage project user interface was a departure from how most people thought electronic journals would be delivered. Considerable effort was being placed on creating electronic journal delivery systems that would be driven by text-based displays using semantic markup technologies to create interactive publications (see, for example, Kirstein and Montasser-Kohsari 1995). The RightPages system placed an emphasis on delivering the journal to the scholar in a way that closely represented the "look and feel" of print by presenting the user with the original page image.

> While others in higher education are working to create new-fangled, interactive electronic journals that have little resemblance to their paper ancestors, the Red Sage Project is taking a much more conservative approach. "We're not trying to create on-line journals," [Richard] Lucier says. "We're just starting the whole process by putting journals on line."
>
> The strategy behind the project, he notes, is to make the computerized version of a journal as similar as possible to the paper edition so that faculty members are comfortable using it. (Loughry 1993)

The initial version of the RightPages user interface displayed only black and white images, but with plans to offer colour images as well as interactive image manipulation tools like zooming, rotating, and colour adjustments (Loughry 1993). A Macintosh version of the client was made available in 1995, but development of a Windows version was suspended as "access to the Red Sage Digital Journal Library through the World Wide Web is being actively investigated" (Lucier and Brantley 1995).

2.2.5 Gobbledoc

Gobbledoc was a prototype journal article delivery system under development at Rensselaer Polytechnic (Nagy et al. 1992). This system was developed to research electronic journal retrieval with an initial corpus of just 41 pages drawn from the *IBM Journal of Research and Development* and the *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Like the RightPages system, Gobbledoc was developed to deliver digitized journal article images to network-connected users. However, it featured several advanced document analysis systems in addition to OCR. A semantic document analysis process was performed on documents in the Gobbledocs system to automatically provide labels for page segments, such as titles, authors, section headers, and graphics. The character recognition stage was handled by the OmniPage OCR software. The Gobbledocs system used the output of the page analysis system to retrieve segmented portions of a full-page image containing a user's query term, and then a page layout browser provided the user with the ability to navigate images containing one segment of the page to another (e.g., from the title to the left column, or from one column to another).

2.2.6 Dienst

Dienst was a server and a protocol developed in a partnership between Xerox Corporation and Cornell University (Lagoze et al. 1995; Entlich et al. 1997). It was designed as a digital library system for computer science technical reports and supported a variety of media types, including HTML, plain text, and several types of image files. Retrospective document conversion was done by scanning paper documents at 600 PPI, but the system also supported importing Postscript files. It was one of the first digital library systems implemented to run on the World Wide Web, allowing users to interact with it using their web browsers.

Dealing with image files was a significant challenge in the implementation of a digital library with Dienst. In Lagoze et al. (1995) they note that their powerful (for the time) Sun SPARCStation 10 could require over 30 minutes to perform a simple 90° rotation on a page scanned at 600 PPI. Network transfer speeds were also a important performance consideration. The TIFF files could not be displayed using a standard web browser, and were so large that serving them to users would have had a significant effect on network performance. As such, the images were pre-processed to produce smaller image files for viewing.

Dienst featured a full-text search extension providing users with the ability to search the text of the documents and to retrieve page images. Although it was possible to extract text directly from Post-script files, the implementation team at Cornell chose to convert PostScript documents to TIFF images, and then use OCR to extract the text and document structure from these documents. Presumably, the extraction of paragraph and other document structure from the Post-script files was more complex and error-prone than extracting it with OCR software:

> At Cornell...we have chosen to use the results of OCR as the input for full-text indexing, because OCR not only extracts text but structure (e.g., paragraph) information about the documents.(Lagoze et al. 1995, 37)

Along with the text results, the OCR system extracted the physical positions of paragraphs on a given page image. When a user clicked on an image in the user interface, the system used the physical co-or-dinates information to retrieve the ASCII text, which was then used as an input to a search system to find articles similar to the one being viewed.

Another feature of the Dienst system was the "Page-Level Zoom" package, an optional utility that provided users with the ability to point and click on an image and have a portion of it shown to them enlarged. This image processing was performed on-demand from the original TIFF image, and was necessary since the display resolution of 72 PPI was not sufficient to read smaller text or equations.

2.2.7 Summary

The late 1980s and early 1990s was a period of experimentation with new forms of electronic publication. Structured document markup systems, the beginnings of hypertext, graphical user interfaces, and network communications were converging to produce new modes of publishing. The extensive work on creating structured document publishing platforms, and the creation of automated document image analysis systems that combined OCR, logical, and semantic page structure analysis, were seen to have a symbiotic relationship.

Yet for historic backfiles of documents, some saw an opportunity to display the original document images, providing the same benefits of fully structural documents for viewing the document, while OCR was used as a largely invisible process for searching these images. This effectively side-stepped the need to automatically extract structural representations of existing printed material, a process that, despite significant technological advances, was still a costly, error-prone, and labour-intensive process.

In the next section we will look at how the document image alignment techniques moved from the early small-scale initiatives to what is best described as industrial digitization processes, involving many people and producing millions of document images capable of being searched and retrieved.

2.3 Industrial Digitization (1995–2001)

At the end of the 20th century, a number of institutions initiated "industrial" digitization projects, moving digitization beyond small, focused projects into large-scale production initiatives. These projects were unprecedented in scope and scale, and were driven by hundreds of people across several institutions, producing databases containing millions of document images. These digitization projects typically focused on collections with large and extensive back-catalogues, primarily long-running newspapers and journals.

2.3.1 British Library Newspaper Digitization

The efforts of the British Library were among the first large-scale document digitization initiatives, designed to open archives of historical periodicals to the general public and to provide tools for navigating these publications in digital form. In a retrospective review of the availability of digitized British newspapers online, Holland writes:

> Modern technology advances have made possible the creation of this great corpus of historical newspapers. The falling cost of computer storage, the spread of broadband, software advances and fast communications with low-cost offshore conversion houses have all contributed. What makes so many millions of newspaper pages so useful today is our ability to search and find individual words and phrases within them, thanks to the advances made in optical character recognition (OCR) software.... (Holland 2008, 20)

The British Library engaged in microfilm digitization efforts once digital technology became cost-effective to implement over a multi-year period. Their efforts began in 1992 with an initiative to digitize microfilms of the Burney Collection of Newspapers, a collection of periodicals from the mid-17th to mid-18th centuries. By the time the project had concluded in 1996, they had collected 21GB of image data, digitizing approximately 6,000 frames of microfilm per month. However, experiments with applying OCR software to this collection was unsuccessful as it was felt that currently-available software did not produce acceptable results. Images were instead indexed by hand (Entlich 2002; King 2005; Holley 2009a).

A subsequent newspaper digitization program at the British Library, initiated in 2001, was a pilot project to provide online full-text searching of historic newspapers (Deegan et al. 2001; King 2005). This project was a collaboration between the British Library and Olive Software, a UK-based commercial software developer. Microfilms of newspapers were scanned at 300 PPI and sent to Olive Software's processing facility. The newspaper page images were automatically segmented into smaller semantic units (e.g., articles, pictures, advertisements) which were then run through a character recognition process. The output of the OCR was stored using Preservation Markup Language (PRML), which stored the co-ordinates for each word, as well as co-ordinates for page elements: paragraphs, columns, or titles. This was then ingested into the ActivePaper Archive system, where users could search the full-text archive and retrieve news articles and their original page images.

The reason for aligning the page image and the full-text transcription was to enhance users' ability to navigate and read the historical newspapers. In a 2001 project report for the British Library, Deegan et al. (2001) comment on the importance of separating "readability" and "searchability:"

> "Readability," defined as the user's capacity to view and comprehend historic text, and "searchability," defined as the user's capacity to reach relevant content through provision of search criteria, can be said to be the two components of "accessibility," or the user's capacity to retrieve and read relevant content....
>
> In the past, it was thought that text generated by OCR (Optical Character Recognition) could provide both readability and searchability. Due to the difficulty of extracting high-quality text from historic scans, this approach is now known to be impractical. ActivePaper Archive™ is among the first technologies based on, and enabling, separation of readability from accessibility. (Deegan et al. 2001, 6)

As previously discussed, the separation of "readability" and "searchability" had been developing for almost a decade before this report was written, so the novelty claims of ActivePaper are perhaps overstated. Nevertheless, there seems to have been a realization that OCR was unlikely to provide a tool for enabling perfect, universal transcription, while at the same time understanding that OCR could be repurposed to provide an enhanced form of accessibility for document images.

2.3.2 Making of America

The Making of America project was a collaboration initiated in the fall of 1995 between the University of Michigan and Cornell University (Shaw and Blumson 1997). The project focused on creating an online digital archive of printed materials documenting the United States social history from the years 1850–1877. The project proceeded in two phases. Phase I (1995–1996) involved the digitization of 1,600 books and 50,000 journal articles from the University of Michigan for a total of over 650,000 pages. The Cornell initiative at the same time totalled 907,750 page images from 967 monographs and 955 serial volumes (University of Michigan 2001). Phase II, also known as "Making of America IV" (MOA4) was a further expansion of the first phase but focused on collecting research data on conversion methods and processes involved in digitization for the principle funding body, the Andrew W. Mellon Foundation.[4] The MOA4 project added 2.5 million pages of monographs to the collection.

A notable feature of the MOA projects is their early adoption of the Standard Generalized Markup Language (SGML) Text Encoding Initiative (TEI). This was used to provide minimal markup regarding the logical structure of the document. Document navigation, such as tables of contents, were also encoded using SGML markup. This was then converted to HTML when a user requested a page allowing the TEI-encoded content to be viewed in a web browser. This markup did not include the full text of the document.

4. The Mellon Foundation at this point was heavily involved in the creation of JSTOR, also being developed at the University of Michigan. This will be discussed in the next section.

The full-text results of the OCR process were incorporated into the SGML markup but the word positions were not directly mapped to the image in the interface. Users could perform a full-text search over the entire corpus and retrieve the page images where their query result could be found. The first phase of the project used the Xerox ScanWorX OCR software, while the second phase used the PrimeOCR software from PrimeRecognition. While the word locations were not specified in the SGML markup, a relationship was maintained between the page image and the extracted text.

The project had a direct impact on facilitating access to previously forgotten materials. The MOA archive was freely available on the Internet with no restrictions. In the year 2000, materials that had previously been in inaccessible "cold" storage were searched an average of 120,000 times per month, and that users viewed over 500,000 digitized pages each month (University of Michigan 2001).

2.3.3 JSTOR

The early history of JSTOR is comprehensively covered in Roger Schonfeld's book, *JSTOR: A history* (Schonfeld 2003). This history is of particular importance in documenting the use of OCR in digitization initiatives, particularly with respect to documenting contemporary ideas surrounding image-to-text alignment around the time when the Internet was an emerging, and not yet fully-understood, distribution method.

The JSTOR project began in 1993 when William Bowen, the president of the Andrew W. Mellon Foundation, conceived of a project to digitize back catalogues of journals. Bowen had studied the impact of scholarly journal publication on libraries, especially with respect to the ever-increasing cost of storing and maintaining growing physical collections of these publications, including the cost of new buildings or renovations to existing buildings. By the early 1990s, no viable solution had emerged to help libraries deal with the space and management costs associated with housing scholarly journal collections.

Initially JSTOR was developed as a project to miniaturize journal collections by digitizing backfiles of journal microfilms and distributing these to libraries on CD-ROM. This would allow libraries to move the paper journal issues to cheaper off-site storage, or to deaccession these collections altogether while still providing their users with access to the complete collection. However, after seeing the growth of the Internet over the first few years of the decade, Bowen was convinced by Ira Fuchs, then vice-president of Computing and Information Technology at Princeton, to focus on distributing this collection over inter-institutional networks and, eventually, the World Wide Web,[5] even though network distribution in the mid 1990s was not without its own technical challenges.

> The pilot librarians worried that 'even a single user printing a full article composed of bitmapped images may seriously degrade performance' on the campus Internet gateway—slowing network traffic to a halt for all campus users. (Schonfeld 2003, 30)

Despite these concerns, the idea of distributing the journal back files on CD-ROM was scrapped and replaced with a decision to deliver the content over the World Wide Web.

Early in the planning and requirements phase there was some con-

5. For context, the World Wide Web was still in the early stages of adoption at this time and may have been viewed as a "fad" in some circles. The first widely-distributed web browser, *NCSA Mosaic,* was only released on 23 January 1993. Schonfeld relates in his book: *"In addition librarians, from the prospective test sites were adamant that, as a set of distributed CD-ROMs, JSTOR would 'provide little advantage over either microfilm or bound copies of the periodicals.' Fuchs recalls that their adamancy on this point was particularly valuable in pushing Mellon to deliver* JSTOR *over the Internet. There was as yet no specific decision as to how JSTOR could be brought online, wither via Gopher, the web, or some other application."* (Schonfeld 2003, 30).

cern over the format in which the digitized journals would be delivered. Contemporary experiments with electronic journal systems were favouring the use of text-based document markup languages, such as SGML or the Open Document Architecture (ODA) (Farrow et al. 1994), for online delivery since they offered a more efficient and adaptable means of delivery document content without the need to send large image files over slow networks. However, it was determined that the cost of re-creating page layouts in the journal backfiles to a document markup language would have been prohibitively expensive for the JSTOR project, increasing the cost from $0.08 to over $2 per page. As noted by Schonfeld:

> ...it was clear that JSTOR's electronic version would have to offer perfect fidelity to the original pagination and layout, as well as an accurate reproduction of the text. Scholars needed to know the page on which a sentence appeared, for their footnotes, and textual accuracy was of course imperative. A purely textual representation was ruled out early on because display would not be true to the original format and 100 percent accuracy would be laborious and exceedingly expensive to attain...had it been necessary to use text for display, it is likely that the economics would have been so prohibitive as to prevent the creation of JSTOR (Schonfeld 2003, 28).

However, it became clear that a system with only page images, without some means of navigating and retrieving them using text searches, was not desirable. Initial proposals for digitization did not include OCR as part of the process, but based on significant feedback on the initial proposal, an OCR phase in the digitization workflow was added.

> Many advisory committee members were convinced that fulltext searchability was a requirement of the electronic medium. Yet the page-image architecture seemed to preclude such searchability: some scholarly resources, such as experiments being undertaken at Cornell and Yale universities,

> used images without searchable text. But JSTOR's commitment to be responsive to user needs pushed it to add text files that would be searchable while remaining invisible behind the images. The layer of text could substantially enhance JSTOR's usefulness to scholars and students, who would be able to search the text of the journal for phrases.... With images in place for display, the fulltext's accuracy was of less concern—it could be, at least to some degree, "dirty." (Schonfeld 2003, 28–9)

The initial version of JSTOR was launched for the public in January 1997 with ten journals and approximately one million pages digitized from original documents (not microfilms, as was originally intended). Every page image was digitized at 600 PPI, processed with OCR software, and then manually corrected to 99.95% accuracy (Guthrie 1999). The results of each page were stored in a plain text file, which was not displayed to the users but used as an 'invisible' full-text search field for each page.

> For a retrospective collection like JSTOR, this database structure offers users the advantage of images (perfect fidelity to the original) without sacrificing the chief benefit of text files (allowing full-text searches). The resulting system is a powerful research and teaching tool. (Guthrie 1999, 293)

Although this system allowed users to search and retrieve page images based on the full-text of the journal, the text was not spatially aligned with the original page images. The data necessary to accomplish spatial alignment was collected and stored, however, and the present interface for JSTOR does provide alignment of text and page images.

While the JSTOR project was started with the goal of addressing the rising costs of collection infrastructure, libraries and users largely ignored this benefit, focusing primarily on the increased access to their journal back collections that this system provided. By 1997, JSTOR had become a web-based system for searching and navigating millions of journal page images using full-text search.

> JSTOR's dilemma has been that library demand for its collections seems to derive principally from access rather than space-saving, even at the many libraries that could benefit from both. Further expansion of JSTOR's collections will be most sensible if librarians perceive that real costs are saved.... Librarians have tended to view JSTOR more as an access tool to important journal backfiles than as a space-saver for titles of less interest. (Schonfeld 2003, 367)

The JSTOR project fundamentally changed how individuals used back catalogues of journal articles by creating a full-text image search system. Prior to JSTOR, historical journal issues were largely ignored parts of a library's collections since they presented a formidable and time-consuming navigation challenge, making them inaccessible for all but the most dedicated researcher. However, with JSTOR, opening up these backfiles with search has allowed researchers and students to instantly retrieve articles they likely would not have found otherwise (DeLoughry 1996; Bronner 1999; Chapman 2001). Today JSTOR is widely regarded as a resounding success and has set important precedents for subsequent digitization projects.

2.4 Mass Digitization (2001–present)

"What do you do with a million books?" was the question asked by Gregory Crane (2006) in an article discussing the emergence of mass library digitization programs. The digitization projects that Crane was referring to were different from any previous efforts by "orders of magnitude in at least five and probably six dimensions:"

- **Scale** The new digital libraries were at least 1,000 times larger than the largest previous initiatives, on the order of tens of millions of books and billions of pages;

- **Heterogeneity** Unlike previous projects that focused on a single collection or document type (newspapers, journals), these large-scale digitization programs were digitizing everything, in all languages;
- **Granularity** While previous projects focused on the physical object as the fundamental organizing unit of the digital library, new projects were focusing on extracting content and working with information inside the materials, providing methods of re-purposing and relating information in multiple sources beyond simple bibliographic search and retrieval;
- **Noise** Previous digitization projects tried to reduce the amount of "noise" in the automatically recognized texts by relying on manual correction or human re-keying of automatically-transcribed texts. The new generation of digitization projects emphasized input and throughput, rather than accuracy, with a corresponding increase in the amount of noise—textual errors—by orders of magnitude.
- **Audience** Previous attempts, like JSTOR, provided digitized text services to well-defined audiences like academics, librarians, or historians. The audiences of massive open-access digital libraries would be the general public, and might be two orders of magnitude larger than that of the largest subscription-based services.
- **Collections and Distributors** A unified digital library interface that provided global access to the entirety of the printed output in libraries would effectively replace the need to maintain the highly fragmented assortment of different library systems, each with their own unique interfaces and minimally-compatible back-end systems. The points of entry to the collective output of human cultural heritage would be reduced as the ease-of-use of a single point of entry gained popularity and ubiquity.

Collectively these projects have become known as "mass digitization" efforts (Coyle 2006; Hahn 2008). In the intervening years since Crane's article there have been a number of changes to the mass digitization project landscape. The HathiTrust has emerged as a partnership between more than 80 organizations engaged in large-scale book digitization efforts, including Google and the Internet Archive. There were approximately 7 million books digitized when Crane wrote his article. Today, estimates indicate that between 20 and 30 million books have been digitized.

With advances in digital imaging devices came the introduction of the digital camera, which, once sufficiently mature, came to replace the use of flatbed scanners in digitization initiatives (Taylor et al. 1999; Doermann et al. 2003). Digital cameras allowed organizations and institutions to process documents much faster because they could capture a page image in an instant (as opposed to the few seconds required by a flatbed scanner), reducing the the amount of physical effort in the digitization process.

Mass digitization projects feature OCR as a central navigation technology, enabling users to search and browse through their page images using highly-optimized search systems. Unlike previous programs, such as JSTOR, the emphasis of the mass digitization effort is on throughput, and not accuracy, in recognition. In many cases, no attempt is made to correct the recognition results of page images, resulting in systems with high recognition error rates. The errors in the recognition do not prevent these systems from being useful; Rather the OCR serves as an invisible search layer, preserving the spatial relationship between every word. This technique, discussed in the previous section, was a departure from extraction-only OCR and served to provide page-level access to the digital page images.

2.4.1 Carnegie Mellon Million Book project

The Million Book project, also known as the Universal Library Project (MBP/ULP), was started in 2000 as a collaboration between

Carnegie Mellon University (with a grant from the National Science Foundation) and several universities located in China and India (Reddy and StClair 2001). The goal of this project was to create a universal digital library, containing books drawn from many countries, in many languages. The initial goal for this project was the digitization and OCR of one million books. By 2007 the project had digitized and performed OCR on over 1.5 million books in over 20 languages (Universal Digital Library 2007). Although the project had aspirations of reaching 10 million books by 2010, this project seems to have finished digitization and been integrated into the Internet Archive's Book Digitization program (see Internet Archive 2014b).

Sankar (2006) provides details about the workflow and overall throughput of a single digitization centre at the Regional Mega Scanning Center (RMSC) in Hyderabad, India. This scanning centre achieved a throughput of 140,000 pages, or approximately 500 books, per day. There were 50 scanning stations operating 8 hours per day. For OCR, they determined that a single PC could process approximately 2000 images per day, including image editing. They had 125 OCR machines operating 8 hours per day, and achieved between 90% and 95% accuracy on texts in languages in Latin script. For texts in other scripts they noted that OCR was not possible.

The images were captured at a high resolution (600 PPI) but with a significant amount of loss due to their chosen image representation format. They chose TIFF with CCIT Group 4 Fax compression, a bitonal (black and white) image format suitable for low-bitrate transmission over standard telephone lines, for most of their documents. Only rare materials were digitized in full colour.

The Internet Archive currently hosts a subset of the books scanned in the MBP/ULP (Internet Archive 2014b). Today many of the links to texts seem to be broken on the Universal Digital Library main portal (http://ulib.org). It could not be determined if the collections of scanned books have been made available to other organizations participating in book digitization projects.

2.4.2 Amazon.com

While not as philanthropic or altruistic as many efforts at creating a digital library, the online retailer Amazon.com deserves a mention as one of the first efforts to create a large-scale online digital library from printed collections in 2003 (Kirkpatrick 2003; Price 2003). This project allows people to find books for purchase from the Amazon.com website by typing a search phrase and being taken to the page in a book where their search phrase occurred. It is unknown exactly how many books are available with the “Search Inside!” and “Look Inside!” programs.

2.4.3 Google Book Search

The Google Book Search project is an ambitious and unprecedented digitization initiative that has, to date, digitized over 20 million items held in major university and public libraries around the world. It is the primary contributor to the HathiTrust Consortium (§2.4.5) and has been a lightning rod for public engagement with digitized print collections. It has also faced significant legal challenges in the course of digitization, facing lawsuits from numerous publishing and authors groups that felt mass digitization initiatives were trampling legal rights. Today, Google integrates book search results into its main search interface, providing users with instant access to billions of pages of printed texts, applying the same technology that has made their search engine an overwhelmingly popular gateway to the Internet.

Google's book digitization efforts were separate from their opt-in program for publishers to submit books for scanning and search (the “Partners Program,” see Smith 2005). The Google Print project first launched experimentally in 2003 as a closed trial program that sought participation from major publishers and rights holders (Sherman 2003). In 2004 the project was publicly launched with an announcement that Google had partnered with the university libraries of Harvard, Stanford, Michigan, Oxford, and The New York Public Library to begin digitally scanning the books in their collections to make them available for searching (Google 2004). In 2005 the name of the project was changed to Google Book Search to avoid confusion with a service for printing documents (Grant 2005).

The book digitization program sought to digitize all books in partner libraries, making out-of-copyright works freely available and providing small previews of in-copyright works, with links to purchase or find a book in a library. Initially Google wanted to make books that were in-copyright but out-of-print (and thus difficult to consult) freely available. Almost immediately after announcing its library partnerships, Google was the focus of several lawsuits, most famously from the Association of American Publishers and the Authors Guild, for infringing author's copyrights. Several agreements and conclusions to legal proceedings have been reached among the various lawsuits. The first agreement in 2008 allowed the book digitization program to proceed (Drummond 2008). Most recently a second lawsuit brought by the Authors Guild was dismissed in November 2013 (Mullin 2013).

Google has since expanded its library digitization program to include partnerships with several large international libraries, such as the Austrian National Library (*Österreichische Nationalbibliothek*, Austria), the Bavarian State Library (*Bayerische Staatsbibliothek,* Germany), the Lyon Municipal Library (*Bibliothèque Municipale de Lyon*, France), and over 40 other libraries (Google 2014a).

As of 2012, Google had scanned more than 20 million books and has quietly started to scale back its operations, switching from scanning entire shelves to selecting individual items in its partner libraries that have not yet been digitized (Howard 2012).

Users of Google Books (the current name) see several variations of the book view, depending on the book's copyright status and the agreements Google has in place with the publisher. The "full view" provides users full access to all pages of the book, and users may search within the book, their search terms highlighted, *in situ*, as a coloured overlay to the page image. Limited Preview is used for works that are still in copyright, but for which the rights holders have granted Google permission to display a portion of the book. When a user searches the book they will be shown their search term *in situ*, but only a few pages before and after the search hits are shown. Users are directed to libraries or online book dealers to borrow or purchase the book, respectively. Snippet View will display just a few sentences of the book that match a user's search terms. Finally, there is a view for which no preview is available that just displays limited bibliographic data about the book, but no page image or content is shown (Google 2014b).

Access to Google's scanning facilities is tightly controlled since the company sees their workflow and tools as proprietary and offers a significant competitive advantage. As such, little is known about their equipment and workflow, or how the technologies developed by their research scientists are employed in a production context. A public FAQ on the scanning agreement between Google and the University of Michigan, one of its founding partners, offers some insight into the techniques used to digitize the volumes (University of Michigan 2005). Specifically:

- Google does not remove the binding from the materials that they are digitizing;
- Materials sent to a Google digitization facility to be digitized are unavailable for only a few days;
- Google provides each library with digital copies of the books from the library's collections, scanned at 600 PPI, with aligned OCR text provided;

- Google pays for all costs related to scanning, including pulling and re-shelving material selected for digitization.

The agreement that libraries would receive copies of all digital images created by the book scanning project would become an important point several years later when the HathiTrust was founded.

Google has funded the research and development of related technologies for OCR and text digitization, despite the fact that their production workflows are a closely-guarded corporate secret. The Tesseract OCR system, first developed by Hewlett Packard Labs in 1994 was open-sourced in 2005 after almost a decade of stalled development (Smith 2007). In 2006 Google re-released Tesseract, fixing a number of bugs (Vincent 2006). Since then it has been funding development of this software. Although it was initially developed for English-only texts, Tesseract has been adapted to support multilingual document recognition (Smith et al. 2009) and recognition based on adaptive language and image models (Lee and Smith 2012).

In addition to Tesseract, Google has also been funding the development of OCRopus, a character recognition, page layout and document analysis system (Breuel 2008; Breuel 2009). OCRopus is developed by the Image Understanding and Pattern Research group at the University of Kaiserslautern in Germany. It is designed as a modular OCR system featuring three primary components:

- **Physical layout analysis** identifies and segments text areas (e.g., columns, blocks, lines, and reading order);
- **Text line recognition** converts images to text, along with identifying possible alternate characters;
- **Statistical language modelling** integrates the text recognition with prior knowledge about the language, vocabulary, grammar, and domain of the document.

It is widely believed that Google's digitization program uses Tesseract and OCRopus, but to date there has been nothing published on how these tools are used in their digitization program.

2.4.4 The Internet Archive

In 2004 the Internet Archive began a pilot project at the University of Toronto, digitizing 2,000 out-of-copyright books (Carlson and Young 2005). The Open Content Alliance (Hafner 2005) was formed in 2005 through an alliance between The Internet Archive, Yahoo!, Adobe Systems, The European Archive, HP Labs, The UK National Archives, O'Reilly Media, Prelinger Archives, the University of California, and the University of Toronto (Advanced Technology Libraries 2005). The focus of the OCA was to digitize only out-of-copyright works, focusing on particular collections at partner organizations. Microsoft was also briefly involved with the book scanning projects, but has since backed out of the project (Nadella 2008).

Today the OCA partnership largely lives on under the auspices of The Internet Archive's book scanning and archiving initiatives (Internet Archive 2014a). The Internet Archive claims to have scanned over 600 million pages and placed 3.9 million books online, drawn from over 1,000 partnerships. They claim to have 33 scanning centres in 7 countries, actively scanning 1,500 books per day. Currently the digitization initiatives are funded on a project basis, with libraries paying the costs associated with digitization through grants or discretionary project funding.

Unlike the Google book scanning project, the Internet Archive publishes details about its scanning process and the tools they use (Miller 2012). At the core of their project is a specially-developed book digitization workstation called the "Scribe," (shown in figure 2.7) containing two digital cameras for simultaneously capturing the opening of a book (recto/verso) on a V-shaped cradle. A glass plate, operated by a foot pedal, is lowered onto the opening, and photography-grade lighting illuminates the pages.

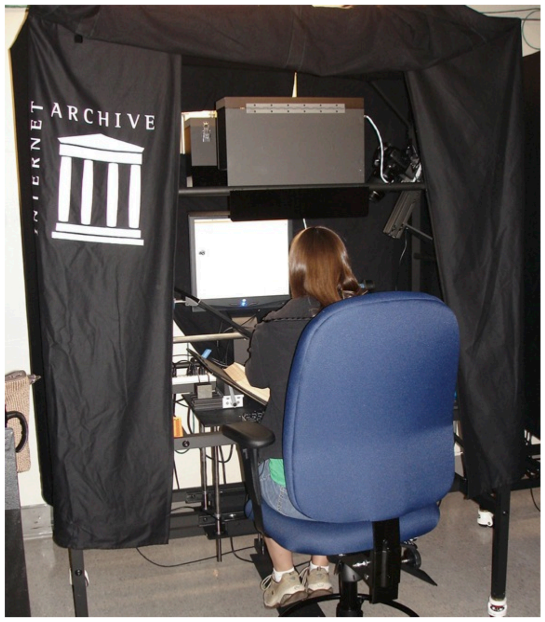


Figure 2.7: Internet Archive Scribe book scanning workstation (from Miller 2012)

The Internet Archive uses Abbyy OCR software to provide searchable representations of their texts. Each digitized book is run through their OCR software, and the resulting Abbyy XML file is stored and parsed by several tools to provide *in situ* search and image retrieval. The Internet Archive provides libraries with copies of the digitized images as well as perpetual free storage of the books on their servers. Currently the organization claims it stores 3.5 petabytes of digitized books and associated data and metadata.

2.4.5 HathiTrust

The HathiTrust emerged in 2008 as a partnership of thirteen universities in the United States, two of which (Wisconsin and Michigan) were involved in the Google Books digitization initiative. It is a clearinghouse of digitized texts held by the member organizations, including those digitized by third-party initiatives such as Google and the Open Content Alliance. Since 2008 the HathiTrust partnership has grown to include over 60 research libraries and consortia from the

USA, Canada, and Europe (York 2009; Conway 2010; Christenson 2011; HathiTrust 2012).

The size of the digital library collection as of January 2014 was:

- 10,885,715 total volumes
- 5,714,014 book titles
- 286,517 serial titles
- 3,810,000,250 page images
- 488 terabytes (TB)
- 3,531,165 works in the public domain

By far the largest contributors to the HathiTrust are the libraries participating in the Google Books digitization initiative, with 96.8% of the items in the HathiTrust contributed through this program. A further 2.9% is contributed through the Internet Archive Book digitization program, and local digitization efforts have contributed the remaining 0.3% (Downie 2012). The page images are encoded in JPEG2000 and TIFF formats and occupy approximately 98% of the 488TB, with textual data (OCR and metadata) occupying just 2% or approximately 13TB.

Storing word positions derived from OCR data is a significant technical challenge for providing fast search and retrieval. Co-ordinate data indicates where each OCR-transcribed word is located on every page image, and provides users with the ability to navigate the page images using a full-text search interface with their search word or phrase highlighted on a page image. An early HathiTrust blog post (Burton-West 2009) indicated that storing the word locations consumed approximately 85% of the total index size, meaning that the text itself likely occupied less than 15% of the search index. At the time, the index size was 4.5TB and 7 million documents. Extrapolating for the size of the collection today, the word position index would be approximately 11.5TB. The most commonly-occurring word in the English-language corpus was the word "the," occurring over 4.3

billion times, and the size of the word positions index for all occurrences of the word "the" was 4.3GB—it has probably grown to 10GB since then.

The HathiTrust Research Centre (HTRC) was established to provide access to the data generated by the project, and lead efforts for data analysis, text mining, and retrieval tools. The texts of digitized works in the public domain are available in two datasets that are freely downloadable for researchers: Works digitized by Google (approximately 2.8 million as of February 2013), and works digitized by partner institutions (approximately 350,000 as of February 2013) (HathiTrust 2014). The HathiTrust Data API provides another means of accessing the content of the HathiTrust using a web service.

2.5 OCR Systems Summary

Leonid Taycher, an engineer on the Google Books project, has estimated that there are approximately 130 million distinct titles of printed books, based on estimates of unique International Standard Book Numbers (ISBN) and other unique identifiers from library systems (Taycher 2010). With numbers of this magnitude it is obvious that the current pace of document digitization and recognition will be both time consuming and expensive.

Recently, attention has turned back to document recognition techniques for historical books and handwritten archival materials. The Improving Access to Text IMPACT project (Balk and Ploeger 2009) was an initiative looking at methods of improving OCR for historical (pre-1900) documents. Notably, aligned image and OCR-extracted text for the purposes of retrieval is listed as one of the primary uses of OCR in this project (Anderson 2010). Text and image alignment has also been proposed as a method to enable historical handwritten document retrieval (Toselli et al. 2007; Zinger et al. 2009; Toselli et al.

2011). The next section will examine the various formats used to store image and text alignment results.

2.6 Image and Text Alignment Formats

While ASCII makes it possible to share text between computer systems, it is only a character encoding format. Additional formats have been developed to preserve spatial positioning of text in relation to an image. The inclusion of this data has become essential for digitized text collections. In later chapters, encoding and storing spatial relationships for symbolic music will be discussed, so this section is given as a review of the historical and current text formats commonly used for the same purposes.

2.6.1 PDF

The Portable Document Format (International Standards Organization 2008) was first released by Adobe as an application-independent means of distributing digital documents for print publications. Over time it has become the *de facto* standard for distributing fixed-layout documents over the web, since it can maintain font definitions, layout specifications, and image placement in documents shared between users on different computer platforms.

In a PDF file page images are placed as a background layer, and the precise location of each word is specified through a co-ordinate system. This has allowed OCR results to be aligned with original page images, and has been widely adopted. Many digitization efforts that include an OCR step will typically, as a matter of course, embed the OCR results in a PDF representation. This allows users of these files to both download a scanned page image and search a document, with the PDF reader software highlighting their query terms on the page image.

2.6.2 DjVu

DjVu, first introduced in 1998, featured a new image format using wavelet-based compression to achieve higher compression ratios than existing image compression techniques (Haffner et al. 1998). It was targeted specifically at image-based digital libraries, as a means of providing users with access to documents containing scanned page images that could be distributed over the Internet at non-broadband speeds. Like PDF, the DjVu format also included an invisible text layer that could store the results of OCR, providing a mechanism for searching within a document. The developers of DjVu claimed that the advanced compression on DjVu make it a superior document image delivery format than PDF (Rile 2002), but it never achieved similar levels of adoption by document digitization initiatives and has largely fallen out of use.

2.6.3 XDOC

The XDOC format (Connelly et al. 1999) was the output of a number of related OCR systems. It was the output file format of ScanSoft ScanWorX (and related systems) but had its origins in the Kurzweil omni-font recognition system. In 1980 Kurzweil sold the OCR platform his company had developed to Xerox. In 1999, ScanSoft Inc. purchased the rights to the OCR software. As one of the more successful commercial OCR applications in the 1990s, it was a widely accepted format for processing the results of OCR in many image retrieval systems (Lagoze et al. 1995; Phelps and Wilensky 1996; Weir et al. 1997).

2.6.4 DAFS

The Document Attribute Format Specification (DAFS) (Dori et al. 1997) was an attempt at providing a container format for relating a document's logical structure with its physical layout. Figure 2.8 shows the representation of a "chapter," (reading from the bottom) and its physical structure (reading from the top). This format was expressed using SGML. One notable application of the DAFS format was its use as a container for character-level ground-truth data in an automated OCR evaluation system (Wang et al. 2001). DAFS is no longer supported or used by any OCR systems.

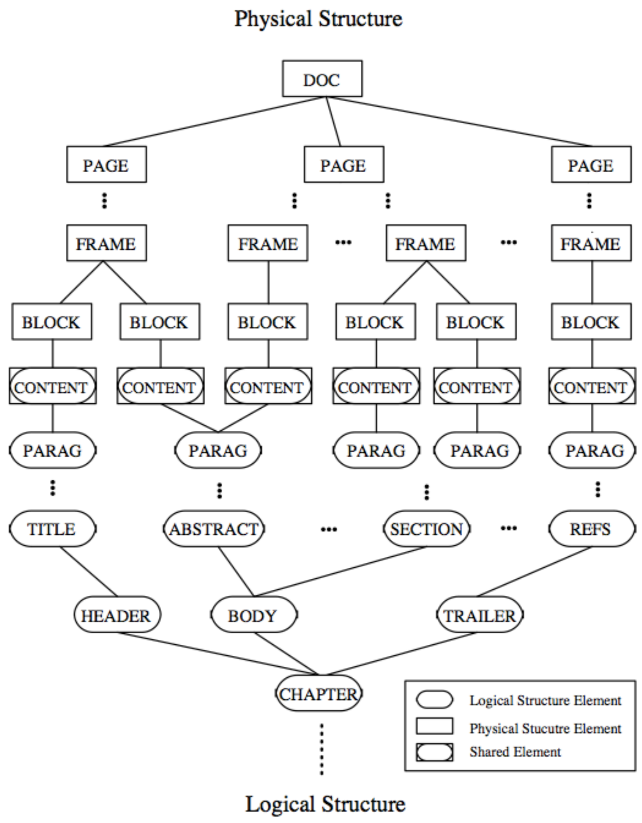


Figure 2.8: The DAFS expression of physical and logical structure of a chapter (from Dori et al., 1997)

2.6.5 METS/ALTO

The METS (Metadata Encoding and Transmission Standard) schema is an XML-based file format used to describe descriptive, administrative, and structural information of objects in digital libraries (Library of

Congress 2013). The ALTO (Analyzed Layout and Text Object) extension for METS is used to encode the physical content and layout of the document, including the precise pixel co-ordinate locations for each word on a digital image (Boddie 2009; Library of Congress 2014b). Using this format it is possible to encode logical structures (chapters, sections, individual news articles, advertisements) and to correlate these structures with their physical locations and contents on a page image. This can provide a rich source of data for advanced search and retrieval systems. METS/ALTO is used as the underlying format in several library digitization programs, notably the National Library of New Zealand's "Papers Past" newspaper digitization project (83 newspaper titles, 490,856 issues, 3,108,938 page images) (National Library of New Zealand 2013), the Australian Newspapers Digitization Program (650 newspaper titles, 12 million page images) (Holley 2009b; National Library of Australia 2013), and the US National Newspaper Digitization Program (Littman 2007; Library of Congress 2014a).

2.6.6 hOCR

The hOCR format was developed as part of the OCRopus document analysis and OCR system (Breuel 2007). Due to its relationship with OCRopus, it is likely that this is the file format being used by the Google book digitization project as the output from its scanning and recognition processes, but due to the secrecy surrounding Google's text digitization and recognition workflows this could not be confirmed.

hOCR extends the standard HTML tags with OCR-specific information, with the rationale that software clients that do not understand the special extended markup, such as web browsers, can still display the text of the document (Breuel 2010). Specially-designed client software, however, may use the extended positional markup to provide pixel-based alignment between the recognized text and the underlying image.

2.6.7 Summary

Spatial relationship encoding between OCR-transcribed text and its location on a page image is an important component of document image retrieval systems. This section has reviewed some of the historical and current formats that have enabled this functionality in a text-based context, in preparation for later discussions on enabling this in a symbolic music context.

2.7 Chapter Summary

This chapter has been written with the intent to provide a background against which to contrast the efforts of large-scale text recognition with that of large-scale music recognition. Through the history of large-scale OCR we have seen how OCR, a technology that was originally conceived as a system for moving texts from a paper-based medium into a transcribed textual medium, has been repurposed to facilitate navigating and retrieving digitized document images. This change was largely facilitated by increasingly sophisticated hardware capabilities, but was identified as a useful technology with the realization that a purely transcriptive approach to OCR resulted in documents containing too many errors to be useful. This approach is now a fundamental part of how the leading large-scale text recognition initiatives are providing access to digitized materials from libraries and archives the world over.

Image-based document retrieval systems are fundamentally changing the way we interact with texts and libraries. A search on the HathiTrust website today will scan the contents of over 3 billion pages in a matter of seconds. The same search, even just a decade ago, would have had to be performed manually and taken many lifetimes to complete.

For printed music materials, there are no large-scale digitization and transcription efforts currently underway. While OCR tools, best practices, search, retrieval, and display techniques have been steadily developed to provide text transcription for image navigation, the same cannot be said of OMR tools. The context in which OMR tools currently operate is that of transcription of musical content, and these tools are not designed to produce results compatible with any systems for image content search and retrieval. The next chapter will examine the current state of the art for OMR tools.

3. Music Document Recognition

This chapter will present a review of optical music recognition (OMR) systems. The review will focus on several themes to be touched on later in the dissertation. Comprehensive reviews of OMR systems and development have been published many times previously—approximately once per decade for the last forty years: (Kassler 1972; Carter et al. 1988; Selfridge-Field 1994; Bainbridge and Bell 2001; Bellini et al. 2008; Rebelo et al. 2012), as well as several recent graduate-level theses and dissertations with extensive literature reviews (McPherson 2006; Pugin 2006a; Johansen 2009; Rebelo 2012). As such, this review will provide a more thematic approach to the literature, identifying several areas of research and development so that it can be referred to in later chapters.

A brief history of the development of OMR will be presented (§3.2). Next, the OMR process will be discussed, along with the various tools and technologies used to convert an image into a symbolic music representation (§3.3). This will be followed by a discussion of the state of evaluation in OMR systems (§3.4). Previous attempts at large-scale processing of music document image collections will be described (§3.5), along with a short review of server-based OMR solutions (§3.6). Finally, a review of OMR for non-CWMN (common Western music notation) will be provided (§3.7).

3.1 Visual and Symbolic Representations of Music

Optical music recognition describes the process of converting a visual representation of music notation to a symbolic representation. Visual representations may come in two forms: raster images, where the symbolic music shapes are composed of tiny discrete picture elements (*pixels*); or vector images, where the image is composed of geometrical primitives (points, lines, curves). Symbolic representations encode music notation symbols using a system of tokens. These tokens are stored in a way that maintains the musical relationships, or "semantics" of, and among, the notation symbols. Symbolic representations require software for rendering and manipulating the encoded music symbols (e.g., editing, transposition), which may be further used to produce a visual representation using a graphical layout system suitable for printing, or a synthesized audio representation. Encoded symbolic music may also be used for symbolic computational analysis or score searching and retrieval. There have been many symbolic music formats developed; see Selfridge-Field (1997a) for a overview of most historical formats.

3.2 History of OMR

The first efforts at OMR were described in a dissertation by Pruslin (1966). The system described operated on a single measure of CWMN, and was capable of recognizing a limited set of musical symbols. Several years later, Prerau (1970) introduced the "DO-RE-MI" OMR system, capable of recognizing three measures of printed CWMN consisting of a single voice on two staves in a single font.

The goal of most early OMR development was a universal recognition system, capable of recognizing the entirety of music notation output, in much the same way that early OCR systems were envisioned as universal transcribers of textual content. Thus, despite the limitations of the early OMR systems, Prerau concludes that the software "should be able to be expanded to the recognition of all printed music" (Prerau 1971). Kassler echoes a similar sentiment in his 1972 review of these first two dissertations:

> Perhaps the greatest accomplishment of the authors is that, as a result of their work, the logic of a machine that 'reads' multiple

> parallel staffs bearing polylynear [sic] printed music in at least one 'fount' and size can be seen to be no further than another couple of M.I.T. dissertations away. Quite possibly such dissertations may get completed before much thought is directed toward deciding what wisely to do with the masses of musical data that an operational [music] OCR system could make available for computer processing. It is remarkable, nonetheless, that (of all things) this technology may cause return of musicologists' attention to the core concepts of their field which constitute musical theory. (Kassler 1972, 253–4).

However, by the 1990s most OMR researchers had abandoned the idea of building a single music recognition system that could reliably process the output of all music notation.

> ...in practice, composers and publishers often feel free to adapt old notation to new uses, and invent new notation, as they see fit. There are in fact national "dialects" of music notation, and musical works use many different levels of notational complexity. Thus it may not be possible to devise a single recognition system capable of recognizing all music notation. [Pruslin 1966] states that a complete solution to the music recognition problem is "the specification of: which notes are present, what order they are played in, their time values or durations, and volume, tempo, and interpretation." This level of recognition suffices for only some of the applications listed [later in this paper]. (Blostein and Baird 1992)

The realization that a single, universal recognition system was not a likely outcome of OMR systems development occurred at approximately the same time as similar realizations in OCR systems development. As discussed in the previous chapter, this realization roughly coincided with the simultaneous development of aligned image and transcribed textual content, thereby reducing the need for developing a completely accurate universal recognition system. However, while textual initiatives transitioned to building systems with images and aligned symbolic transcriptions, these technologies were not developed to support similar capabilities in music document recognition research. Instead, OMR systems have continued with a transcription-based design that produces output suitable for editing in notation software, with no attempt to maintain a relationship between the notation and the recognized image.

As research and development continued through the 1990s and 2000s, contemporary OMR systems were developed to specialize in transcribing specific repertoires or styles of notation. In addition to CWMN recognition systems, repertoire-specific recognition systems exist for many different music notation styles, including lute tablature (Dalitz and Karsten 2005; Wei et al. 2008; Dalitz and Pranzas 2009), Byzantine chant (Gezerlis and Theodoridis 2002; Dalitz et al. 2008b), mensural notation, both in print (Pugin 2006a) and manuscript sources (Tardón et al. 2010), and others.

Several OMR toolkits have also been developed to help assemble bespoke OMR systems. These systems have been used to build several of the aforementioned systems, and provide a generalized structure and toolset from which these customized OMR systems may be built. Examples of these frameworks include the CANTOR system (Bainbridge 1997) and the Gamera system (MacMillan et al. 2002). While these systems present a more flexible approach to OMR, they require significantly more development expertise to create and run OMR than "turnkey" systems and, as such, are generally only used in research contexts.

3.2.1 Current OMR Software Systems

At present there are a number of commercial and open-source OMR systems available. The most common application of these systems is on sources of printed CWMN. Of the commercial systems, the most established systems are PhotoScore (Neuratron 2014), SmartScore (Musitek 2014), and capella-scan (Capella Software 2014). Development on SharpEye (Jones 2008b), previously regarded as one of the best systems, seems to have stopped. There is at least one OMR application for mobile devices, iSeeNotes (Gear Up AB 2014). In the open-source community, Audiveris (Bitteur 2014) is the most mature and continually developed OMR application for CWMN.

There is very little published literature detailing how both commercial and open-source OMR systems work and what processes they use to perform their recognition. Open-source software has the advantage that developers can read the source code for details on the implementation, but this is not generally available to users of these applications. The user has no means of configuring or substituting different techniques within the process. As a result, these systems operate largely as "black boxes," where an image input is provided and a transcription is produced, but with no indication of the specific implementation of each step, or the combination of steps used to produce the result.

OMR systems developed for research purposes are better described in the literature, but are often highly customized to address a special need or research topic. As well, many OMR systems developed for research are never made publicly available. Since research systems are often built for a distinct environment, they require difficult and onerous setup outside of their development environment. They are also designed for a specific type of input, as a means of evaluating a particular component of the OMR process, such as staff-line removal (Dutta et al. 2010), a particular recognition technology (Rebelo et al. 2011), or different approaches to musical symbol re-construction (Raphael and Wang 2011). While these systems may serve as useful development platforms for a portion of the OMR process, they are not intended for use as complete systems.

3.2.2 Summary

OMR has been a topic of active development since the late 1960s. In the beginning it was thought to be a panacea for transcribing all possible musical documents, but this is no longer a widely-held belief. Instead, current OMR systems primarily focus on transcription of a particular type of music notation.

Since the beginning, OMR has been conceived of as a process through which an image is transformed into a transcribed symbolic representation. A number of tools and techniques have been developed that can be used in this process, but most commercial and open-source implementations of OMR software intended for use by the general public implement only one chain of these processes. The next section will focus on OMR as a process, and present a selection of different approaches to each step within the process.

3.3 OMR as a Process

OMR refers to a process involving a number of steps, where each step requires a different technique for manipulating an image or a symbolic representation of the notation. Some steps use image manipulation and transformation techniques, preparing an image for the recognition step where shapes are identified and classified into musically recognizable symbols. The results from this step are passed on to a representation stage where the identified symbols are related and assembled into data structures, and individual symbols are given a musical context—for example, a note symbol is determined to be on a specific staff, with its pitch and duration governed by contextually-appropriate clef and key signature symbols. Finally, this musical structure is encoded in a known symbolic representation and saved as a file, to enable sharing the symbolic musical results among other software programs.

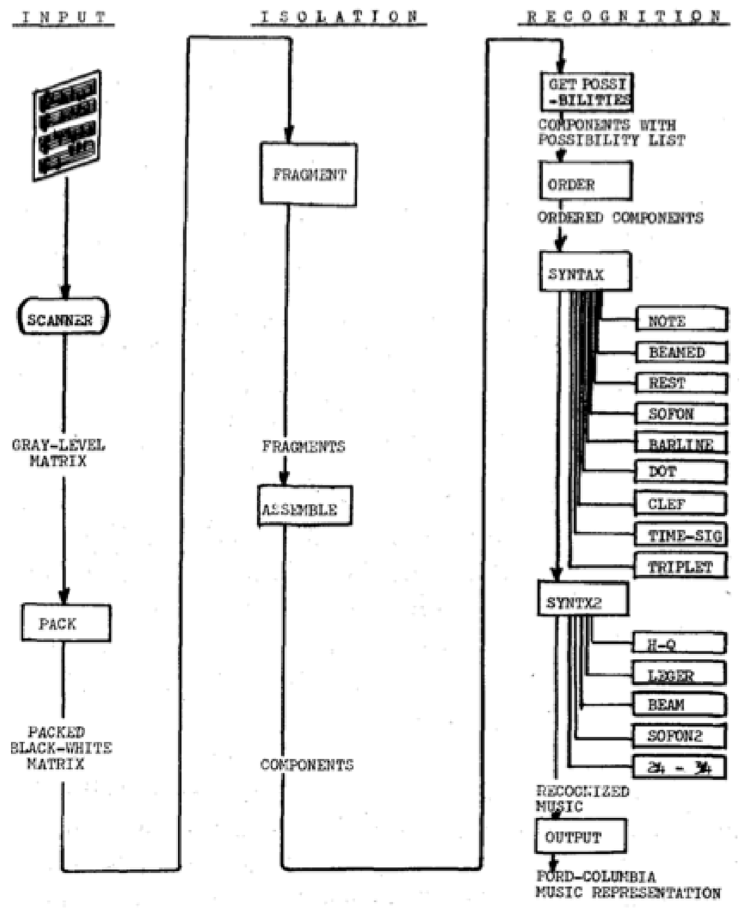


Figure 3.1: The OMR Process in the DO-RE-MI system (Prerau 1971)

While OMR is widely recognized as a process involving many different sub-systems, there is no universal agreement on the names given to each stage in this process, the order in which they must be executed, or on the specific methods used for processing. Many OMR authors have given different names to each component in this process. Prerau (1971) divides the process into *input, isolation, recognition,* and *output* (figure 3.1), producing symbolic music encoded in the Ford-Columbia Music Representation (a.k.a. DARMS) format (Erickson 1975). Rebelo et al. (2012) describe the steps in the process as *preprocessing, music symbol recognition, musical notation recognition,* and *final representation construction.* Others, such as Bainbridge and Bell

(2001) identify the process as consisting of *staff line identification, musical object location, musical feature classification,* and *musical semantics.* Bellini et al. (2008) describe the process as *segmentation, basic symbol recognition, music notation symbol reconstruction,* and *music notation model.* McPherson (2006) describes a system in which data may pass through several components, but the system is capable of automatically identifying ways of improving the recognition system with automated feedback, sending an image back to a previous stage to adjust the processing parameters to obtain better results in later stages. Depending on the requirements of a particular system the exact composition of the process changes to meet the specific needs of the recognition system.

For the present discussion, the stages of this process will be described as *image pre-processing, symbol recognition, notation recognition,* and *representation construction* (figure 3.2). This description of the process provides a high-level overview of the entire OMR process, and broadly groups the tools required to perform each task into logical divisions.

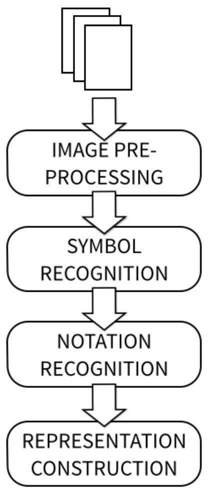


Figure 3.2: A simplified representation of the stages of the OMR process

Within each of these broad stages there are many tasks that may be performed (some examples are listed in table 3.1), but the specific techniques used to implement each stage varies from system to system. For example, an OMR system designed to accept and process images digitized from microfilm (Pugin 2006b; Burgoyne et al. 2008) may employ different tasks in the pre-processing stage than one designed to accept high-resolution, full-colour scans of an original source. Specialized processes for older documents may combine techniques for foreground and background separation to reduce ink bleed-through (Leedham et al. 2002; Rowley-Brooke and Kokaram 2012), or to identify and remove extraneous image information from borders (Ouyang et al. 2009). Similarly, a system designed for processing CWMN requires different symbol classification and musical reconstruction techniques than music set in neume notation (McGee and Merkley 1991; Helsen 2011) or mensural notation (Carter 1992; Pugin 2006a; Tardón et al. 2010).

Step	**Example Tasks**
Image Pre-Processing	Binarization, de-skewing, de-warping, rotation correction, foreground-background estimation, segmentation, staff-line region identification / removal, staff-height normalization
Symbol Recognition	Connected component analysis, classification, template matching, glyph feature extraction
Notation Recognition	Determine inter-symbol relationships (clefs to notes, etc.), re-introduce staff lines, encode note pitch, duration, chord voicing, structural identification (measures, systems, staves, etc.)
Representation Construction	Translate musical structure to encoding format (MIDI, MusicXML, MEI, etc.)

Table 3.1: OMR steps and some example tasks for each step

While "full," or complete transcription is the most common application of optical music recognition, some systems have been proposed to extract different representations from a music document image. Vigliensoni et al. (2013) propose a system for recognizing the location of bar lines and measures to align multiple sources of the same work. Several systems have been proposed that operate primarily on textual underlay (i.e., lyrics) of music scores (George 2004; Burgoyne et al. 2009). While these systems require many of the same operations at some stages of the process, they also require custom recognition systems and tools for later stages.

In the next sections the four stages of recognition will be reviewed, describing some of the newer or more notable techniques for each stage.

3.3.1 Image Pre-Processing

Image pre-processing is the stage in the OMR process where a digital image is manipulated to prepare it for processing by later stages. The sequence of images shown in figure 3.3 demonstrates one possible application of techniques used to prepare a particular image for symbol recognition. The original image (figure 3.3a) is first binarized to remove the colour information (figure 3.3b). The decorative border is then removed (figure 3.3c) and then certain page elements (staves, lyrics, ornate first letters) are also masked (figure 3.3d). The result is an image that contains only music notation. The staff line locations are identified, and the staff lines automatically removed to prepare the image for musical symbol classification (figure 3.3e).

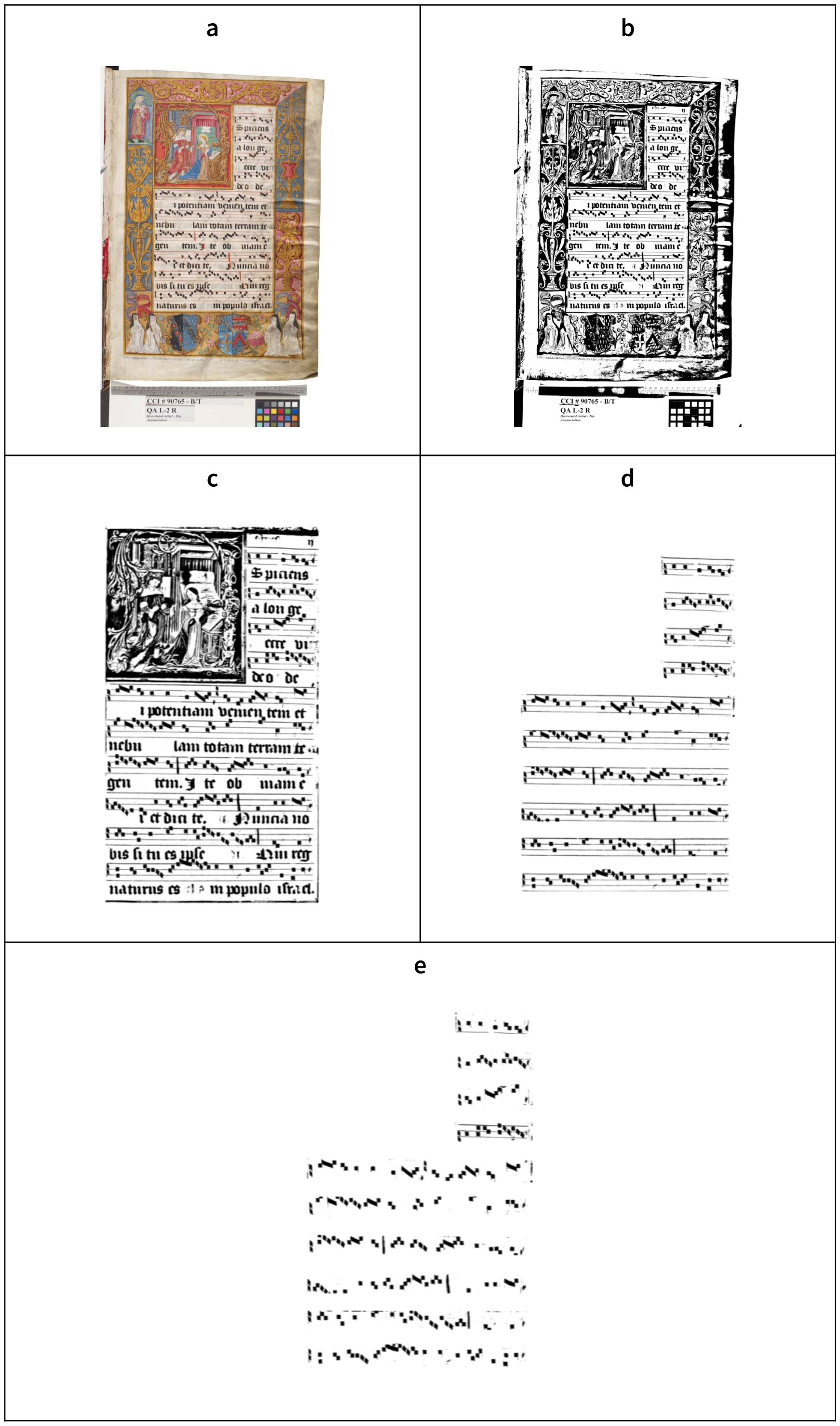


Figure 3.3: Example image pre-processing steps

3.3.1.1 Binarization

Binarization creates a binary image from the original colour image, and is typically one of the first steps in any document recognition process. The process of binarization simplifies the image by separating the foreground, or page content, from the background, or "noise" through the application of a threshold. Pixels that represent content are coloured black and referred to as the "foreground," while pixels that do not represent the content are coloured white, and represent the "background." Accurately separating the foreground from the background is crucial to minimizing recognition errors later in the process. Sezgin and Sankur (2004) present an overview of several types of image thresholding techniques. Leedham et al. (2002) present a survey of global thresholding techniques for working with degraded documents. They conclude that no single technique can perform adequately on all types of documents, and instead propose multi-stage thresholding, where an image may be subjected to multiple iterations of a thresholding technique.

For older documents, where the paper can exhibit a degraded physical state, such as bleedthrough, mildew, water stains, or foxing, accurate separation between foreground and background is difficult to achieve. Recto-verso registration (Burgoyne et al. 2008; Rowley-Brooke and Kokaram 2012) has been proposed as one method of dealing with bleedthrough (figure 3.4), where ink from the other side of the page has "bled" through and presents itself as a candidate for the foreground. The registration process aligns the reverse of the page with the front, and then subtracts the content of the reverse from the front, effectively removing bleedthrough.

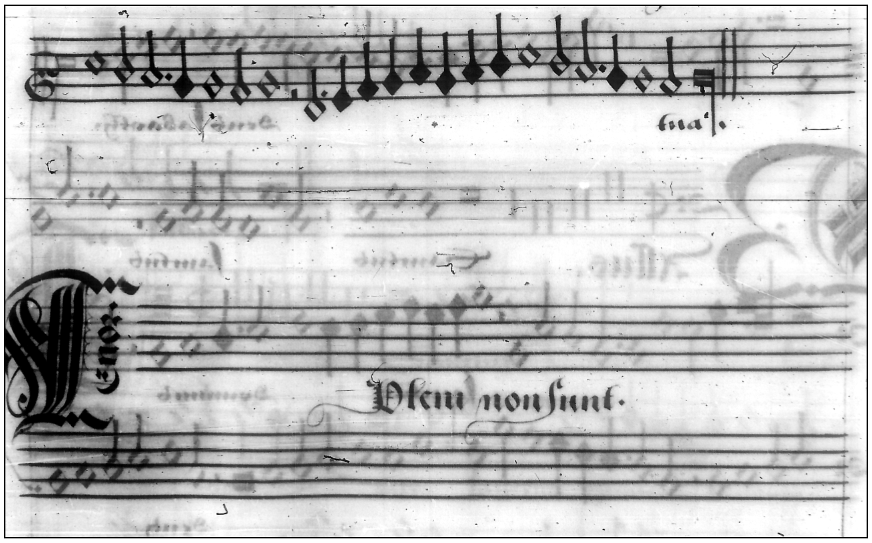


Figure 3.4: A page image demonstrating bleedthrough

Burgoyne et al. (2007) employed a goal-directed technique (Trier and Jain 1995) to evaluate the performance of 24 binarization methods. The goal-directed technique uses an evaluation metric based on the effects of the method on later recognition stages. They ranked the binarization algorithms according to their symbol classification performance using precision and recall. They present a ranking of the techniques used according to their performance, with the Brink and Pendock (1996) method emerging as the best performer. They contrast this to an earlier evaluation of binarization techniques (Sezgin and Sankur 2004) where the Brink and Pendock method did not perform particularly well. In light of their results they note that special attention should be given to developing music-specific binarization techniques.

Pinto et al. (2011) propose a domain-specific approach for binarization in music scores. They use knowledge about staff line thickness to maximize retention of information for musical symbols in a binarization process.

3.3.1.2 Image Segmentation

Image segmentation is a class of image analysis techniques to automatically classify document regions for the purposes of removal or applying specialized recognition techniques to specific types of content. For example, image border detection and removal (Ávila and Lins 2004; Ouyang et al. 2009) can remove extraneous information from the margins of page images, including illustrations or artefacts of the digitization process, such as black borders, colour bars, or rulers. Lyrics, titles, or other textual regions of the page may be automatically segmented from music notation, allowing an OCR process to be applied only to these regions. Different approaches are used for different types of segmentation tasks. For example, Carter (1992), working with printed mensural notation sources, describes a system that identifies and removes ornamental first letters simply by identifying the largest object in the upper-left hand of the document. Burgoyne et al. (2009) describe a technique for segmenting text underlay (i.e., lyrics) from music notation. They take advantage of the fact that lyric lines are largely uniform in their horizontal upper and lower boundaries (due to the near-uniform nature of letters), while music notation symbols undulate across a page. By identifying areas of uniform baselines, they can mask either the areas containing lyrics (i.e., lyric removal), or preserve only areas containing lyrics (i.e., lyric extraction).

Staff line region identification and removal is a music-specific form of page segmentation. This process identifies areas of musical content on page images by attempting to automatically determine staff regions. On a binarized image, staff lines connect most of the musical symbols, making it difficult to apply shape classification techniques to these symbols. Several approaches for staff line identification and removal have been proposed (Wijaya and Bainbridge 1999; Fujinaga

2004; Rebelo et al. 2007; Cui et al. 2010). Dalitz et al. (2008a) provide a comparative study of staff removal algorithms.

Recently a competition-style evaluation has been organized as part of the International Conference on Graphics Recognition (Fornés et al. 2011b; Fornés et al. 2013; Visani et al. 2013). The tasks for this competition have focused on score writer identification (i.e., identifying the handwriting of a particular person by the shape of the symbols they write) of and staff removal algorithms. These were tasks tailored to exploit the ground-truth data available in the CVC-MUSICMA dataset described in Fornés et al. (2011a).

After the pre-processing step, the image is now ready to proceed to a symbol recognition stage.

3.3.2 Symbol Recognition

The symbol recognition stage attempts to label shapes on the page with an identification as a musical symbol. There have been several approaches to symbol classification, many of which are discussed and compared in Rebelo et al. (2012). One area that is not covered extensively by this paper is the use of an adaptive recognition system to dynamically improve symbol classification with human feedback.

3.3.2.1 Adaptive OMR

Adaptive optical music recognition, first proposed by Fujinaga (1989, 1996a), is the re-integration of corrected and verified symbols back into a recognition system effectively creating a system that can "learn" new symbols or increase the recognition accuracy for known symbols (figure 3.5). With adaptive recognition, a human editor verifies and corrects the output of the recognition stage.

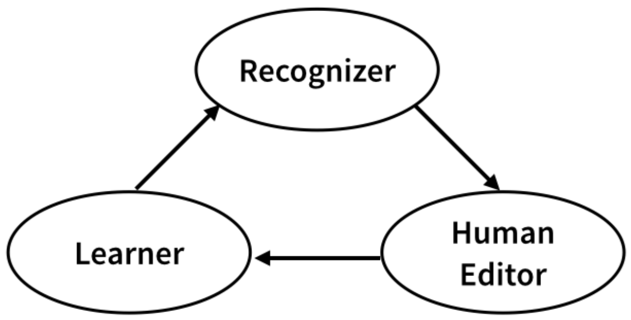


Figure 3.5: The Adaptive OMR process (Adapted from Fujinaga et al. 1991)

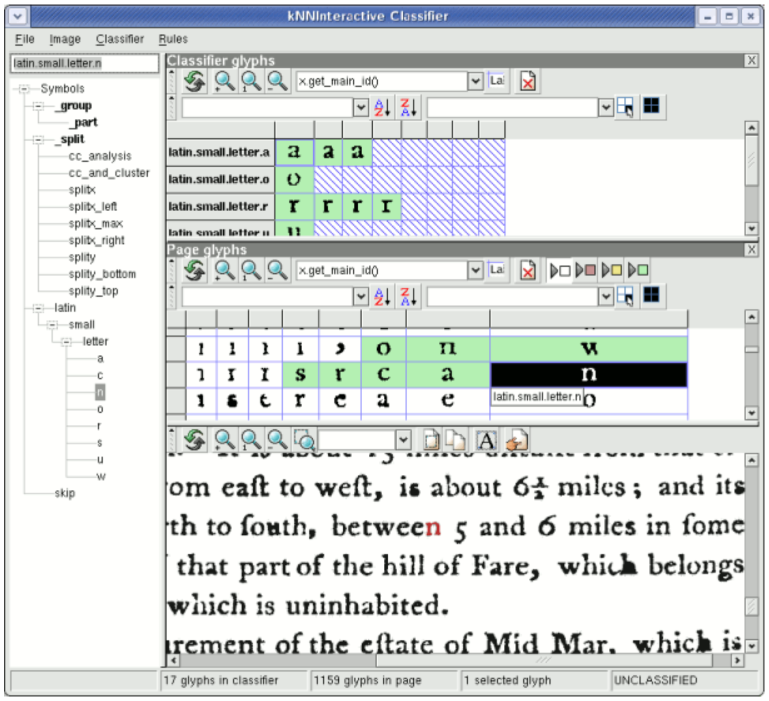
An adaptive OMR recognition stage, like non-adaptive OMR systems, begins by using a process to match a visual representation of a symbol with a label identifying that particular shape. The specific procedure can vary from system to system, but generally speaking, a database of known symbol instances are compared against unknown symbol instances (i.e., the symbols extracted from the page) and an attempt is made to match an unknown symbol with a known symbol.

With the addition of human feedback correcting mis-recognized symbols, the database of symbols is continuously expanded to include these new exemplars. With subsequent attempts to recognize that symbol there is a much greater likelihood that it will be correctly identified.

Later in this dissertation, an adaptive approach to OMR will be presented as having significant advantages over non-adaptive approaches. To provide background for these discussions, this section will describe and compare two adaptive OMR systems available, Gamera and Aruspix, and discuss the adaptive capabilities of the Audiveris OMR system.

Gamera

Gamera (MacMillan et al. 2001) is a framework for document analysis written in Python and C++. It is a "toolbox" for building custom recognition systems (Droettboom et al. 2003), which may include music documents as well as other document types. It contains a wide variety of image processing algorithms, as well as a symbol classifier based on the *k*-nearest neighbour (KNN) algorithm.



om eaſt to weſt, is about 6¼ miles; and its
th to ſouth, between 5 and 6 miles in ſome
that part of the hill of Fare, which belongs
which is uninhabited.
rement of the eſtate of Mid Mar, which is

Figure 3.6: The Gamera classifier interface

The classifier interface (figure 3.6) is a graphical user interface for correcting and maintaining the database of symbols. The "connected components" on the page—that is, the areas of contiguous black pixels extracted from a binarized image—are compared against a classifier containing labelled symbols, constructed using human feedback. New exemplars of known symbols can be added to provide a more robust characterization of that symbol. As subsequent pages are processed, the results of each page, and the additions to the classifier, allow the recognition system to constantly learn new symbols and improve results.

The KNN classifier (Cover and Hart 1967) uses feature sets extracted from each glyph. Features are essential descriptors used to quantify certain properties of a shape. Simple features might include the height or width of the glyph, while others, such as projections or central moments may be computed properties of the glyph (Fujinaga et al. 1991). While a single feature is unlikely to carry enough information to accurately distinguish between two symbols, features may be combined to create a unique "fingerprint" that serves to uniquely describe a glyph. The *k*-nearest neighbour classifier in Gamera compares the features of each unknown glyph with the features of the known glyphs in the database, and assigns the unknown glyph the label of the group to which it most closely resembles.

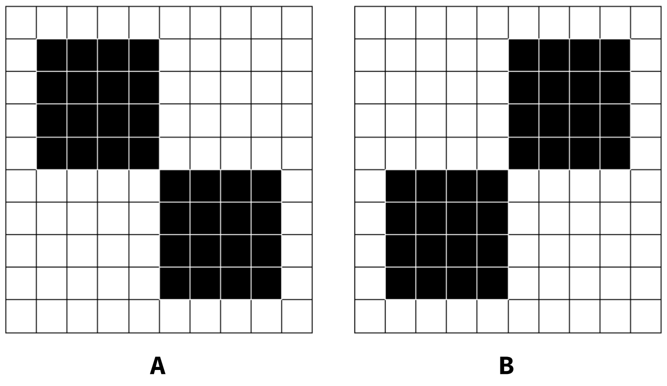


Figure 3.7: Two different glyphs with identical height, width, and black area.

To illustrate, two glyphs are shown in figure 3.7. They are represented as a grid of pixels of equal height, width, and area (i.e., equal numbers of black pixels to white pixels). Each of these measurements constitute a single feature of the glyph. Despite being obviously different symbols to a human, the height, width, and area features alone do not provide enough information to discriminate between the symbols. To accurately classify the two symbols another feature may be calculated, the diagonal projection (figure 3.8). This feature is extracted by rotating the glyph clockwise 45°, and then collapsing all the pixels in the glyph into a single dimension, forming a histogram of the glyph. This histogram may be added to the features set for the classifier, providing additional information to help discriminate between the two glyphs.

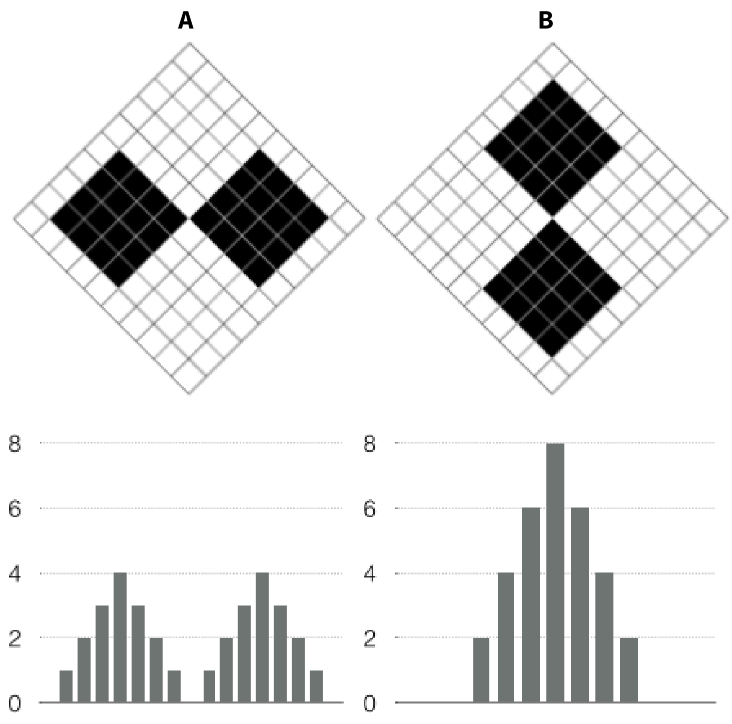


Figure 3.8: Diagonal projections of glyphs

In addition to the KNN classifier, Gamera has a feature weighting system that uses genetic algorithms (GA) to optimize and improve recognition (Fujinaga 1996b). Calculating the solution to the optimal weighting of all features over all symbols is a computationally difficult task, so a GA is used to discover the most optimal solution in a reasonable amount of time. Using the GA, a solution is derived that places more or less emphasis on features depending on their contribution to recognition accuracy on a given symbol set. Thus a feature that provides greater discriminatory effect within the classifier is given emphasis over a feature that has less of an effect.

The GAMUT system (Gamera-based Adaptive Music Understanding Tool) was an OMR system built using the tools available in Gamera (MacMillan et al. 2002; Droettboom and Fujinaga 2004). It used the Gamera tools for feature extraction and KNN classifier to provide symbol recognition, but required custom-built code for musical semantic assembly.

Aruspix

Pugin (2006b, 2006a) is an adaptive OMR system designed for recognition of 16th-century single-impression printed music sources. Unlike most other OMR systems, Aruspix does not need to remove stafflines to segment the musical symbols into separate glyphs. Instead it treats a musical symbol and its intersection with a staff as a complete glyph (this reflects how the source was originally printed, making Aruspix, in essence, a system for recognizing individual pieces of single-impression type). Aruspix uses a hidden Markov model (HMM) classifier to provide symbol recognition using the HTK software (Young et al. 2006).

The initial version of Aruspix was tested with 240 pages of music drawn from a number of different 16th-century print sources (Pugin 2006a). Staves in Aruspix are normalized to a constant 100 pixel height by analyzing and identifying staff regions on the page, and automatically segmenting them from non-staff regions. A non-overlapping window, 2 pixels wide and 100 pixels high, is passed over the staves, and six features are calculated for each window. Two recognition accuracy rates were reported: An "effective" recognition rate calculated considering all deviations from the evaluation data, and a "musical" recognition rate that ignored errors that were of no musical consequence (e.g., a mis-recognized clef would only count as a single error, despite causing all notes on the staff to have an incorrect pitch value). The system achieved an effective recognition rate of 96.82% and a musical recognition rate of 97.11% on a mixed corpus of prints.

The first version of Aruspix was a non-adaptive OMR system. HMM-based classifiers are computationally intensive and very slow to train, often taking several hours to re-integrate new data into its recognition system. Therefore, automatically re-training the recognition system as pages were corrected was not feasible. To address this, a faster adaptation stage was added (Pugin et al. 2007b) using *maximum a posteriori* (MAP) adaptation (Gauvain and Lee 1994). This technique provided online adaptation of the recognition system based on human feedback without requiring a full re-training step.

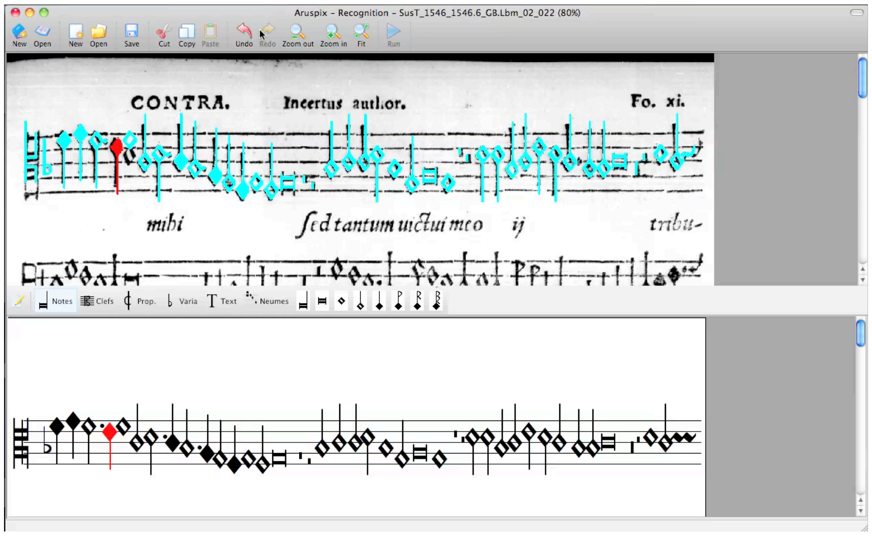


Figure 3.9: Editor interface in Aruspix

To provide correction data, Aruspix features a notation editor (figure 3.9). Users may correct the transcribed notation (shown on the bottom half of the screen) and may enable an overlay that shows image and symbol correspondence (shown on the top). This second interface allows users to very quickly identify mistakes in the recognition and correct these mistakes.

The adaptive recognition component allowed Pugin et al. (2007a) to introduce a book-adaptive and book-dependent approach to recognition. This was inspired by similar initiatives in speech recognition, where a “base” recognition model could be adapted to recognize words spoken by a particular speaker (Huang and Lee 1993). In the same way, a varied corpus of music notation data was used to create a generic recognition system for a baseline transcription. This generic recognition model could then be tailored to a specific book—more accurately, the typeface used in that book—as pages of that book were subject to the correction process, and the results fed back in to the recognition process. This approach follows similar techniques in OCR (Xu and Nagy 1999; Rawat et al. 2006; Kluzner et al. 2009; Lee and Smith 2012).

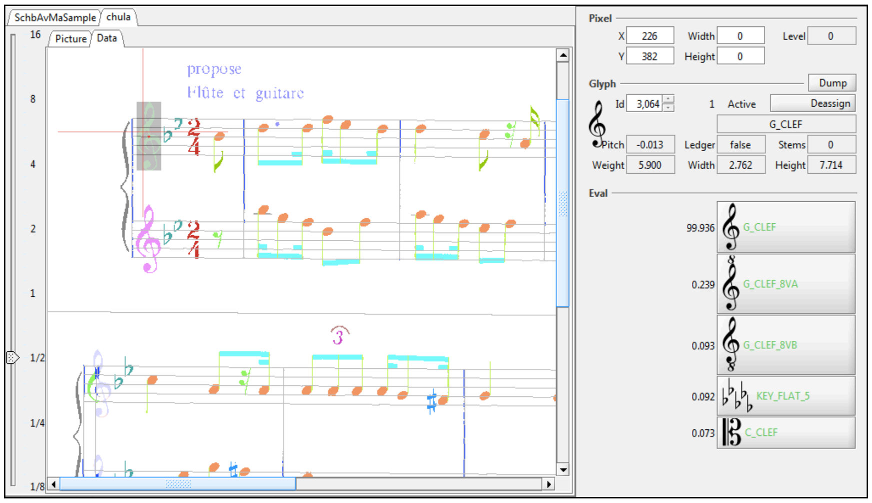
Gamera and Aruspix Compared

Pugin et al. (2008) compared the symbol classification techniques employed by Gamera/GAMUT (KNN) with those employed by Aruspix (HMM). They evaluated both systems on a set of four 16th-century part books printed using single-impression techniques. Their results show that the HMM-based approach was more accurate than the KNN approach in recognizing this particular set of documents. In addition they compared the optimization techniques available: Feature weighting using the built-in GA optimization in Gamera, and *n*-gram modelling of musical sequences for Aruspix. They concluded that the behaviour of both systems demonstrated the strengths of an adaptable system for learning and adapting to a diverse set of books and symbols, as would be encountered in a large-scale recognition effort.

Audiveris

The Audiveris OMR system (Bitteur 2014) features a trainable neural-network recognition system. Symbols can be interactively added to the classifier, and then the recognition system re-trained to accommodate new symbols.

To build the classifier, users select a symbol and are presented with the automatically transcribed properties of the symbol, as well as an ordered ranking of the most likely interpretations of the symbol (figure 3.10). If the system cannot recognize a symbol, the user can manually assign the interpretation.

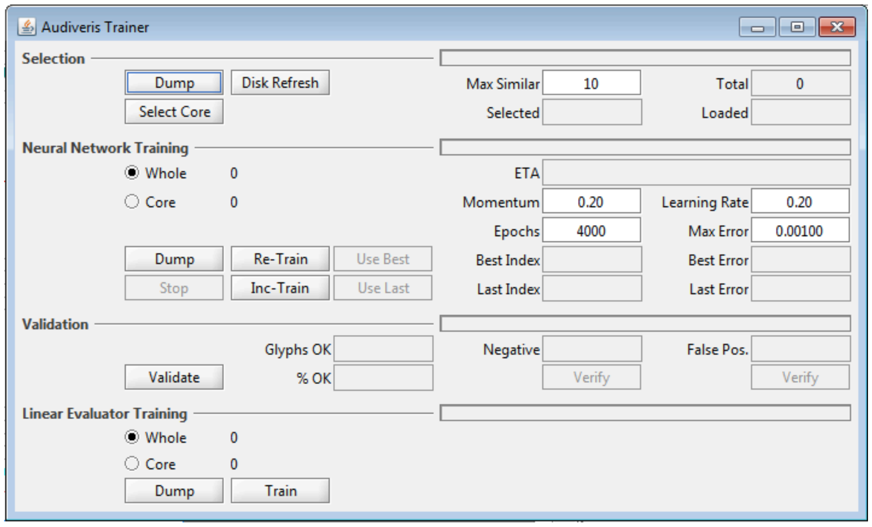


Pixel
X 226 Width 0 Level 0
Y 382 Height 0

Glyph [Dump]
Id 3,064 1 Active [Deassign]
G_CLEF
Pitch -0.013 Ledger false Stems 0
Weight 5.900 Width 2.762 Height 7.714

Eval
99.936 G_CLEF
0.239 G_CLEF_8VA
0.093 G_CLEF_8VB
0.092 KEY_FLAT_5
0.073 C_CLEF

Figure 3.10: The Audiveris recognition view



Audiveris Trainer

Selection
[Dump] [Disk Refresh] Max Similar 10 Total 0
[Select Core] Selected Loaded

Neural Network Training
[● Whole] 0 ETA
○ Core 0 Momentum 0.20 Learning Rate 0.20
Epochs 4000 Max Error 0.00100
[Dump] [Re-Train] [Use Best] Best Index Best Error
[Stop] [Inc-Train] [Use Last] Last Index Last Error

Validation
Glyphs OK Negative False Pos.
[Validate] % OK [Verify] [Verify]

Linear Evaluator Training
[● Whole] 0
○ Core 0
[Dump] [Train]

Figure 3.11: The Audiveris classifier training configuration

Once a customized classifier has been built with additional symbols, the system can be re-trained to accommodate the new symbols. The configuration panel for this system is shown in figure 3.11. Unlike both Gamera and Aruspix, there have been no published descriptions or evaluations of the adaptive components of this software.

3.3.2.2 Summary

This section has discussed adaptive OMR classification techniques, by discussing the implementations of this technique in the systems currently available. The end result of both adaptive and non-adaptive approaches to symbol classification is a set of labelled symbols. These symbols will now proceed to the next stage, where these musical symbols are placed in context with other symbols, thereby introducing the semantics of the symbolic representation.

3.3.3 Notation Reconstruction

The notation reconstruction stage attempts to reproduce the formalized rules of music notation by examining spatial relationships between the classified symbols. Spatial proximity between symbols, in both horizontal and vertical dimensions, is an important consideration for reconstructing the "semantics" of a particular notated piece of music. For example, in symbolic notation the pitch of a given note is determined by a clef. To re-construct this symbolic relationship using shapes, the pitch of a given shape identified as a note must be inferred by locating the nearest and most probable shape identified as a clef, and then determining how the clef, staff lines, and note head align to identify the pitch of that note.

Grammar-based approaches (Coüasnon and Camillerapp 1995; Ferrand et al. 1999; Bainbridge and Bell 2003) re-construct musical structures by assembling the notation primitives according to a formalized declaration of musical syntax, similar to those which govern natural language grammars (e.g., "a complete sentence must contain a subject, verb, and object"). This approach attempts to re-construct their function in a "top-down" approach by examining all primitives on a given scene and applying syntactic rules for which symbol might be found in a given context. In other words, a grammar-based approach attempts to pre-define all of the given contexts in which a particular symbol can appear, and its behaviour when encountered in a context. For example, a sharp symbol (♯) that appears directly following a clef symbol is understood to have a high likelihood that it is specifying a key signature, while the same symbol encountered immediately preceding a note symbol is understood to alter the pitch of that note, but only within the scope of the current measure. The advantage of the grammar-based approach is that several syntaxes may be used on the same set of musical primitives, which can be useful for identifying several hypotheses about the interpretation of the musical symbols, and then determine the most probable interpretation from these hypotheses.

Another approach is to apply heuristic knowledge of the nature of the symbols to reconstruct the musical structures based on supplied rules for each symbol (Droettboom et al. 2009; Vigliensoni et al. 2011). This is a deterministic process, providing a single set of known rules to which each symbol must adhere. To return to the previous example of identifying a sharp symbol, a heuristic approach to reconstructing the score might specify that, should a sharp symbol be encountered within a certain distance from the left of a stave, it is recognized as a component to a key signature. Identifying an accidental may be done as a sub-process of note identification, where the immediate area around every note is searched for a sharp or flat symbol, within a pre-specified distance. Conceptually this is a relatively straightforward approach, but it is inflexible to different possible interpretations of a symbol and may require complex logic to identify and handle all possible symbols. This means that most heuristic approaches to notation reconstruction are bound to a very narrow set of notation styles, and are difficult to modify to accommodate documents that fall outside of the style.

3.3.4 Encoding and Music Representation

After symbol classification and music reconstruction, the output of an OMR system is typically encoded in a symbolic music representation format, for interchange between the OMR system and other software such as notation editors or audio synthesis. This requires converting the derived notation representation into a standardized structure.

Structural representations of music notation in machine-readable formats have existed since the early days of computing. Selfridge-Field (Selfridge-Field 1997a) has compiled the definitive resource for both functional and historical overviews of many of them. Some formats, like Kern (Huron 1997) or MuseData (Hewlett 1997), use ASCII text based structures while others, like NIFF (Grande and Belkin 1996), are binary file formats. In recent years, XML has been the basis for several structural music encoding initiatives (Castan et al. 2001), including MusicXML (Good 2001), IEEE1599 (Baggi and Haus 2009), and the Music Encoding Initiative (MEI) (Roland 2002). Formats that were once popular in notation software, like DARMS (Erickson 1975), are not used in most current generation software systems. While MIDI is a commonly-accepted standard, the representation of symbolic music in this format is very poor when encoding anything other than performance information (Selfridge-Field 1997b). For example, since MIDI encodes a pitch value as an integer, it does not distinguish between enharmonic equivalents; C♯4 and D♭4 are both represented using the integer value 61. Similarly, MIDI does not encode indications of dynamics, but represents dynamics in a playback context by varying the volume at which a particular note is sounded.

Most commercial OMR systems offer export in multiple notation formats, typically for use in either sound synthesis applications (e.g., MIDI) or editing in a music notation editor (e.g., MusicXML and NIFF). Most OMR systems also have their own internal data storage format (Jones 2008a), but these are rarely understood by other software systems and, as such, are not suitable formats for interchange between systems.

3.3.5 Summary

OMR systems are composed of several distinct sub-processes that transform an image into a digital symbolic representation of the music notation. The specific operations used in this process differ from system to system, depending on the expected input or the desired output from the system, but they generally fall into the categories of image pre-processing, symbol recognition, notation reconstruction, and music representation. Each of these steps typically take an input, process it, and produce an output suitable for processing by another tool, or a final representation for use in other tools.

While there are many software tools for performing OMR tasks, evaluating their performance relative to each other is not a solved problem. Each task within the process may produce results that have an impact on the overall accuracy of the entire OMR process. To assess and compare the impact and efficacy of each tool, their performance must be evaluated using agreed-upon benchmarks. Automated methods of evaluating the process is a particularly difficult area of OMR systems development and will be discussed next.

3.4 Evaluation of OMR Results

Evaluation of OMR software allows researchers and developers to quantitatively understand improvements within their own system, and compare the performance of their system to others. Unfortunately, there are few commonly accepted benchmarks for performing these evaluations, or for expressing the results of these evaluations. Byrd and Simonsen (2013) provide an overview of some of the more significant challenges in constructing an effective evaluation protocol for OMR systems.

The most challenging aspect of OMR evaluation is creating an automated method of comparing the uncorrected results of an OMR system with a human-corrected version to determine the errors. Automated systems are necessary tools for comparison, as manually evaluating every musical symbol produced by several different systems is extremely labour intensive and time consuming. However, automated comparison systems have not been developed due to several challenges. The difficulty of comparing the results stems primarily from the amount of processing, interpretation, and contextual knowledge required to "read" musical symbols. Errors may be introduced at any stage of the OMR process, and there is a need to distinguish between the errors that may be introduced within each of the stages: image processing, symbol recognition, and errors in the score reconstruction or musical semantics phase.

To give an example: if a note is correctly recognized and placed on a staff but the clef governing its pitch was either mis-recognized, misinterpreted, or ignored, the system will assign the note the wrong pitch value. The mis-recognition of the clef may be due to over-zealous binarization or staff-removal operations. Moreover, since the clef governs the pitch on the staff, the pitch of every note on that staff would likely be incorrectly identified. A naïve automated evaluation system that simply checks against a known-correct representation for the proper pitch would add an error for every note when, in fact, it is just a single recognition error with the clef. However, the recognition error would not indicate a problem with clef shape recognition, since the error was likely introduced in the image processing task, producing an illegible clef symbol.

Another challenge for automated evaluation is that OMR systems, as they are presently conceived for end-users, are essentially "black boxes," where an input is provided and an output observed, but with no opportunity to evaluate the impact each of the steps in the process has on the output. Should a new technique come up that shows promise for a given task, it cannot be "swapped" with a component in a commercial OMR system. Some methods of addressing this have been developed, for example, the cost- and goal-directed evaluation techniques (§3.4.6) but this does not completely address a need to quantify the impact of each step in the overall process.

In this section, several approaches to evaluation will be presented. None of these systems have, at the time of this writing, developed into a commonly-accepted approach to OMR systems evaluation.

3.4.1 Two-level evaluation

Bellini et al. (2007) propose an assessment criteria for judging OMR systems performance. They note that one of the challenges of comparing OMR systems is the variety of music notation encoding schemes. The same musical sequence may be modelled in several different, but correct, ways within a single representation scheme. Without a means of objectively and consistently assessing different OMR systems, indications of relative performance and identification of techniques that improve recognition are difficult to discover. Their assessment criteria are based on two levels of recognition. The first is the correct identification of basic symbol primitives, designed to evaluate the symbol classification phase, while the second evaluates the correct identification of composite symbols, designed to evaluate the musical reconstruction phase.

3.4.2 Symbol-level Evaluation

Droettboom and Fujinaga (2004) describe a system for evaluating small unambiguous units by relying on a purely graphical evaluation of the system's symbol interpretation, rather than relying on an underlying representation language. They provide an example of a system for identifying dots of augmentation where the presence of a dot of augmentation is identified as a binary condition. This creates a very simple and straightforward means of evaluating the accuracy of the system. They claimed that this evaluation system was particularly useful in their adaptive recognition system for improving the results of correctly pitched note-head symbol recognition.

3.4.3 Semantic-level Evaluation

Szwoch (2008) proposed an evaluation system that operates directly on the underlying symbolic representation—in this case, MusicXML was chosen as the symbolic representation. They note the difficulty in comparing a complex symbolic representation across different systems, where it is unlikely that two OMR systems will generate the exact same encoding for the same musical content. As such they propose a method where the OMR system is used to create the initial encoding. This encoding is then hand-corrected, and the original and hand-corrected encodings are compared for consistency. While this approach was successful, it does little to reduce the labour-intensive nature of OMR evaluation.

3.4.4 Multiple-recognizer OMR

Multiple-recognizer OMR (MROMR) was introduced by Byrd and Schindele (2006; Byrd et al. 2010) and further developed by others (Bugge et al. 2011). In this system, the output of multiple "black box" recognition systems are combined, on the theory that each system will have different failure modes. One system may recognize accidentals better than another, but may perform worse on clef recognition.

The heart of an MROMR system is an evaluation method for comparing and "voting" on particular musical output features. If a symbol is transcribed the same way across all OMR systems it is likely to be correct. Disagreement between systems would indicate a problem. By identifying and understanding the failure modes, a weighting may be assigned to a particular recognition system for that specific mode. They give the example of note beams, where the SharpEye system was the most accurate system for beam recognition. As such, the output of SharpEye with respect to beams could be considered higher, even if all other recognition systems (incorrectly) identified beams in that location.

3.4.5 SmartScore, SharpEye, and O3MR

Jones et al. (2008) proposed two different types of evaluation. The first is an evaluation based on an export to MIDI to provide a common, standardized encoding scheme across several OMR systems. Unfortunately, as they note, an export to MIDI reduces the amount of information contained in the score. MIDI would discard several important musical features such as staccato markings, dynamics, and other performance indicators.

Their second evaluation focused on the symbolic reconstruction phase of OMR. Their evaluation focused on a system's ability to correctly relate symbol primitives with higher-level complete musical symbols. For example, they propose a metric for identifying pitch and duration of a note, which is dependent on accurately recognizing and relating the note head, stem, and flags (i.e., eighth- or sixteenth-note stems). They report the results of this evaluation against three different OMR systems, SharpEye, SmartScore, and O^{3}MR. Their evaluation criteria allowed them to identify which relationships a particular OMR system had difficulties processing. For example, they demonstrated that the SharpEye system could correctly detect a clef or clef change with 66.21% accuracy, while the O^{3}MR system could correctly identify it with 96.55% accuracy. They concluded that the SharpEye system had the best average performance of the three systems across all of their criteria.

3.4.6 Cost-directed Evaluation

In Pugin et al. (2007a) they extend the goal-directed evaluation technique to estimate the real-world "cost" of errors based on an estimate of the amount of human involvement needed to correct errors. For example, deleting an extra note is low cost as it involves a single operation (hitting a delete key), while adding a missing symbol is the most expensive operation involving several selection and insertion operations. They evaluated an adaptive system that improved its recognition rate as more symbols were corrected and re-integrated into the adaptive recognition system. They reported a decrease in human editing cost by a factor of three as the accuracy of the system improved.

3.4.7 Summary

Evaluating the performance and accuracy of different OMR systems is one of the largest unresolved issues in OMR systems development. Systems for automatically evaluating and comparing different approaches to OMR are needed to measure improvements to the tools in the field, but there are currently no available "out-of-the-box" tools for evaluating OMR systems.

3.5 Large-scale processing

> [OMR] has been an active area of research since its inception in 1966, and even though there has been the development of many systems with impressively high accuracy rates...it is surprising to note that there is little evidence of large collections being processed with the technology... (Bainbridge and Wijaya 1999, p. 474)

In this section we will look at several OMR systems capable of processing large amounts of music documents. These systems are in varying stages of implementation (including existing as just a proposal), but none are currently available for public use (as far as I know).

3.5.1 CANTOR

CANTOR was designed as a "general framework" for OMR (Bainbridge and Bell 1996; Bainbridge 1997), where jobs within the OMR process are controlled by a co-ordinating process. A single CANTOR instance may incorporate many different techniques to perform the same task, and the co-ordinator process can automatically "judge" the outcome and use the result of the task that produces the fewest errors later in the pipeline (McPherson and Bainbridge 2001). The CANTOR system is designed to run "headless"—that is, without a graphical user interface—which makes it suitable for automated processing.

Bainbridge and Wijaya (1999) report the results of processing a 672 page "Fake Book" of popular songs, and *The Sacred Harp,* a 417 page book of music for four-part choirs. To evaluate the accuracy of their system they use the number of edit operations needed to correct the score to be a "faithful reproduction" of the original. Despite their system achieving high accuracy rates (above 90% in most cases), they note that this translated to approximately seventeen edits per hundred notes for the Fake Book, and three per hundred for *The Sacred Harp* to recreate the original source. This translates into "considerable effort" on the part of a human editor. The CANTOR system was used as the OMR component in the Melody Index (MELDEX) project, part of The New Zealand Digital Library project (Bainbridge et al. 1999; Bainbridge 2000).

3.5.2 Lester S. Levy Sheet Music Collection

The Lester S. Levy Collection of Sheet Music at Johns Hopkins University was the first project to propose the development of a workflow system for large-scale optical music recognition retrieval systems (Choudhury et al. 2000a; Choudhury et al. 2000b; Choudhury et al.

2001). While these workflow solutions were proposed it appears, however, that only a few components of this project were actually built.

3.5.3 IMSLP and Peachnote

The International Music Score Library Project (IMSLP) (Project Petrucci 2014), also known as the Petrucci Music Library, is a crowd-sourced wiki of public domain scores. As of this writing it contains 79,962 works by 11,106 composers. Users digitize and upload scanned scores to the website, of varying quality and resolution, and these scores are then curated to check that the submitted files are in order and do not infringe copyright. The scores that users submit may be digitized from printed or manuscript sources, or they may be editions that were digitally typeset by that user.

Viro (2011) developed a system, Peachnote, for performing OMR on the entire IMSLP corpus with the goal of making the music searchable. An automated system was built that would automatically download a PDF document from the IMSLP database and perform OMR on the page images using commercially-available OMR systems. The uncorrected recognition output was indexed as melodic *n*-grams (Downie 1999) to enable search and retrieval of these scores. Users can provide a query for pitch and rhythm, and the results are visualized using the Google *n*-gram viewer (Michel et al. 2011), with links provided to download the PDF from the IMSLP collection. No details have been provided on the accuracy of commercial OMR systems on such a diverse set of documents, or on the effectiveness of retrieval using the extracted data.

3.5.4 PROBADO

The PROBADO project is an initiative for developing content-based analysis for non-textual materials, including music documents. One of the goals of this project is to create automated methods of aligning multiple representations of music, particularly audio recordings and their scores (Damm et al. 2012). They apply OMR to scanned sheet-music images, extracting a mid-level chroma representation of the scanned score from an uncorrected MIDI representation. A similar representation is extracted from the audio recording. These mid-level representations are automatically aligned and synchronized using dynamic time warping (figure 3.12) (Müller et al. 2006; Kurth et al. 2008). This approach is robust to noise caused by errors in both the OMR and the polyphonic audio transcription. This technique was reported to correctly align scores and audio 87% of the time (Fremerey et al. 2008).

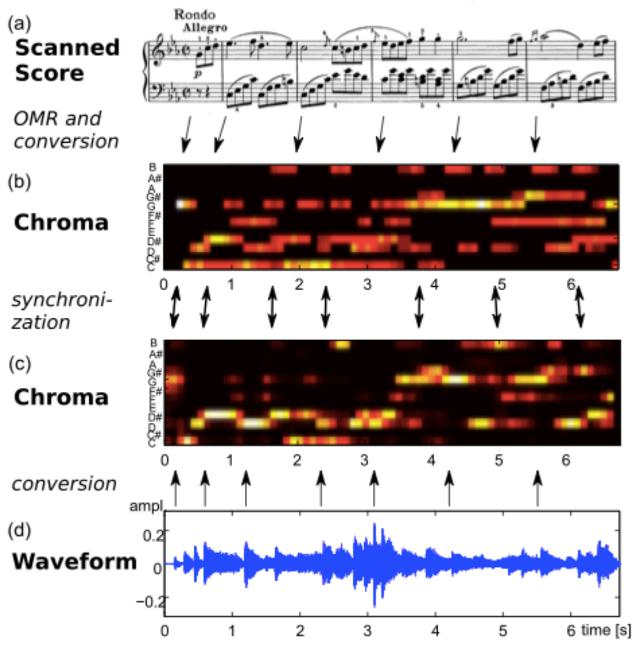


Figure 3.12: Score-audio synchronization in PROBADO (from Kurth et al. 2008)

Users of the PROBADO Music interface may search and navigate a multimodal collection of score images, audio, and lyrical information. The synchronization data is used in their SyncPlayer application where the interface displays the original image with a highlighted region indicating the location as the score plays (figure 3.13) (Thomas et al. 2012b).

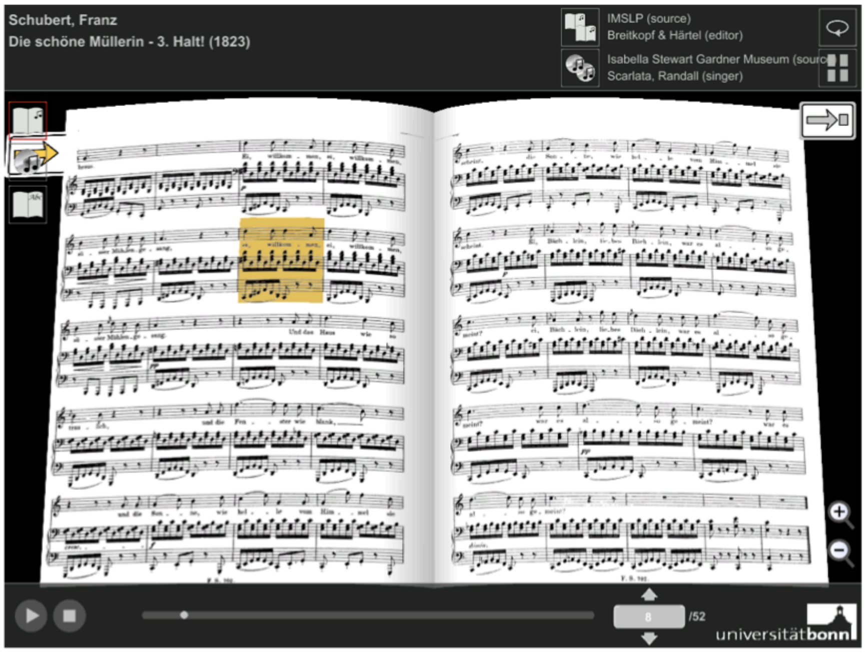

Schubert, Franz
Die schöne Müllerin - 3. Halt! (1823)
IMSLP (source)
Breitkopf & Härtel (editor)
Isabella Stewart Gardner Museum (source)
Scarlata, Randall (singer)
8
/52
universitätbonn

Figure 3.13: SyncPlayer interface for the PROBADO project (from Thomas et al. 2012)

A custom-built workflow system, MACAO, is used to maintain a consistent chain of tools and processes for each piece of material (Thomas et al. 2009; Thomas et al. 2012a). This system was used to process approximately 72,000 score page images and 800 commercial CDs held by the Bavarian State Library. (Due to copyright concerns these materials are not available outside of the library.) A proof-of-concept system[6] was built using public domain materials, containing

6. The address for this proof-of-concept site is provided at http://www-mmdb.iai.uni-bonn.de/probado/ (accessed 8 July 2014). However, despite trying to access it several times in the course of writing this chapter, the actual demo site (http://probado.iai.uni-bonn.de:8080) has been inaccessible. A video (in German) showing the system is available at https://www.youtube.com/watch?v=ubp9QGVmxrQ, beginning at 2:00.

a further 1,900 score pages and 31 hours of audio recordings (Thomas et al. 2012a).

3.5.5 Summary

Although OMR was conceived from the beginning as a technology that could operate on large collections of document images, in practice this application of the technology has not been widely deployed. In this section we have examined some of the most notable projects in large-scale music document processing.

One promising avenue for enabling large-scale processing is server-based OMR systems. Remotely-accessible servers can allow multiple simultaneous users of a shared OMR application to take advantage of enhanced hardware resources for processing larger quantities of document images than desktop or laptop systems. The next section will review existing server-based OMR systems.

3.6 Server-based OMR

Most OMR software packages, especially those intended for use by the general public, are locally-installed desktop applications. This includes popular commercial packages such as PhotoScore (Neuratron 2014) and SmartScore (Musitek 2014), as well as open-source applications such as Audiveris (Bitteur 2014). These applications present users with a graphical user interface (GUI) featuring tools for performing OMR on page images or PDF files that are locally available on a user's computer.

Server-based OMR systems differ from locally-installed applications. They are designed to run on remote computers and can be used by multiple people interacting with the same software over a network connection. This has several advantages. Updates to the software are immediately available to all users of the system, without requiring individuals to download and install patches or updates. A server-based OMR system can take advantage of the higher performance components and networks available in these systems when compared to desktop and laptop systems: faster processors, larger and faster hard disk systems, greater amounts of RAM, and faster network connections. As well, multiple server systems may be networked together to provide a cluster-based solution for OMR, where the recognition workload may be distributed over multiple computers.

Some server-based OMR systems are designed to run "headless" or without a graphical interface for user input. This is suitable for automatically processing large numbers of image files. Other server-based systems provide a remotely-accessible interface, such as one available in a web browser, to control and interact with the system over a network connection.

While server-based OMR system design has been proposed several times before (Bainbridge and Wijaya 1999; Choudhury et al. 2000a; Choudhury et al. 2000b), there are currently no publicly-available server-based OMR systems available. However, two are known to have been implemented as prototype systems.

3.6.1 OMRSYS

The OMRSYS system described by Capela et al. (2008) was implemented as a Ruby on Rails application that ran on a web server. Users could upload images to a web server to be processed. To perform OMR it used the OpenOMR library (Desaedeleer 2006), although there were provisions for integrating multiple OMR recognition stages. The uploaded images were submitted to the OMR process, and a MusicXML file was produced of the score. A text-based notation editor (i.e., an XML editor) allowed users to edit the MusicXML file directly.

3.6.2 Audiveris

Audiveris is an open-source desktop-based OMR application for recognizing CWMN. At the 2013 Music Hack Day in Vienna a prototype system for using Audiveris on a server system was demonstrated (Bitteur 2013; Bonte et al. 2013). To perform OMR on a page image, a user uploaded an image using a web browser. This image was then processed by a remote installation of Audiveris running on an Amazon EC2 instance (Amazon.com 2013). There was no graphical user interface for controlling the Audiveris process. The results of the OMR process were sent back to the user as a MusicXML file.

3.6.3 Summary

Server-based OMR is a promising application for scaling OMR systems to accommodate multiple users on computer systems that typically have more hardware and software resources than personal desktop or laptop computers. This section has reviewed the few known instances of server-based OMR systems.

3.7 OMR for Older Music Notation

The final section of this chapter will review applications of OMR for older music notations. Later chapters of this dissertation will examining systems built for document analysis of non-CWMN music documents; in particular, chant and mensural notation. This section is presented as background.

The first system for performing OMR on older notation systems was proposed by McGee and Merkley (1991). Their system was used to recognize handwritten manuscripts of chant notation. Although they were operating on notation with distinct staff lines and absolute pitches (given by a clef) their paper makes no mention of how the pitches were identified. Rather, they claim that their system can operate by identifying bounding rectangles and the neume shapes could be classified based on this information. Staff lines were removed at an early stage in the process, but they provide no details on how these staff line positions were re-integrated with the neume symbols.

Carter (1992) describes a system for recognizing madrigals written in white mensural notation. His paper focuses on the image processing techniques necessary to prepare an image for OMR, but offers few details on how the symbol recognition is actually performed.

The CANTOR OMR system proposed by Bainbridge (1997) featured a purpose-built programming language, PRIMELA, for programming an extensible OMR system. They demonstrated the effectiveness of their approach by performing OMR on many different styles of music notation, including one example of square-note chant notation.

Caldas Pinto et al. (2000) performed several experiments on recognition of ancient music documents, including evaluations of binarization algorithms for foreground and background separation (Castro and Caldas Pinto 2007), and recto-verso registration for removing bleed-through (Castro et al. 2008).

The previously-mentioned Aruspix system (Pugin 2006a) was designed to work exclusively on Renaissance sources using single-impression printing (figure 3.14). This system does not remove staff-lines to segment the musical shapes from the connected components. Instead, it treats each musical symbol and its intersection with staff-lines as the complete symbol.

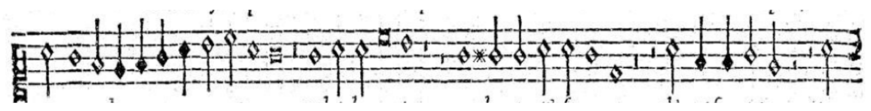


Figure 3.14: Example of Renaissance single-impression music printing

Dalitz et al. (2008b) proposed a complete system for OMR of psaltic Byzantine chant notation built on the Gamera framework. Their system used Gamera's built-in *k*-nearest neighbour classification system using a set of 14 features. They tested their recognition system on 65 randomly-selected pages from six printed books of Byzantine chant and report an accuracy rate of between 95–98%.

Ramirez and Ohya (2010, 2011) have demonstrated a system for classification of square note notation based on handwritten manuscripts of the 14th-17th century. Recently the “Optical Neume Recognition Project” has been attempting to automate transcription of neume notation written in the staffless St. Gallen style (figure 3.15) (Helsen 2011).

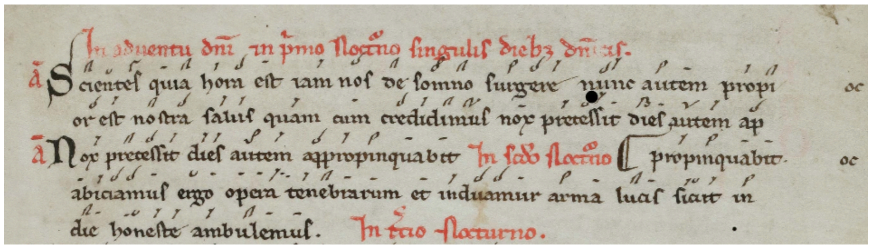


Figure 3.15: Example of St. Gallen neume notation

3.8 Chapter Summary

This chapter has reviewed several specific areas in music document recognition. In particular, it has attempted to frame OMR as a process composed of several stages. There is a wide range of tools available for performing each step in the OMR process, with each tool or technique demonstrating different strengths and weaknesses when applied to specific types of document images. Many of these tools are readily available in open-source toolkits and frameworks and have been used to implement several OMR systems suitable for transcribing a wide range of music document types, featuring many different notation styles.

Yet most OMR systems available make it difficult or impossible to integrate these different tools. Commercial systems are provided as opaque “black boxes” and must be evaluated based on the output of the entire process. Open-source systems have the advantage of having the source code readily available, but integrating different techniques means the user must become a developer. Research systems are not generally available and are often highly customized to address a specific need or research topic.

Evaluating OMR systems is one of the largest open issues that must be addressed. Evaluation allows comparisons of different approaches but with the black-box approach it is difficult to identify the particular stages in any given OMR system that have the greatest (or least) impact in the process and, more importantly, it does not provide a means of quantifying the differences between systems. Several approaches have been developed to address this problem, but no techniques have been widely adopted by the community.

There is currently no OMR software available for use in large-scale recognition projects. Despite the similarities between OMR and OCR, music recognition research is still focused on a transcription-only approach. OMR systems capable of high-throughput page image processing, enough flexibility to deal with the wide variety of notation symbols and document types, and symbolic notation formats capable of capturing and communicating OMR results between systems are all under-developed areas in OMR research.

Very few large-scale OMR projects have been proposed, and even fewer successfully implemented. This is likely due, at least in part, to the amount of work required to correct the results such that transcribed output of the OMR process is as error-free as possible. In a transcription-only OMR process, the correction process is crucial to creating an accurate representation of the underlying source material. However, error correction requires significant human intervention in the process, and is labour-intensive and costly. As such the ability to scale recognition projects within time and financial budgets is directly related to a project's requirements for accuracy in the system.

Despite the challenges in evaluating OMR systems, many of the systems that have been developed report accuracy rates approaching or greater than 90%. While this represents a significant number of errors to correct, it may be that these OMR systems can still provide usable results in an uncorrected context. In textual recognition, usable systems for OCR-based document retrieval were created by employing the technique of aligning transcriptions and images. This same technique is needed for music recognition, and may represent the first step to enabling music document retrieval at a similar scope and scale.

The remaining four chapters of this dissertation will describe several novel approaches and tools for OMR that may be applied to build music document image collections.

4. Towards Large-scale Music Document Recognition

Large-scale music document recognition is the application of optical music recognition techniques and technologies to mass digitization efforts, transcribing music notation content from these images to make them searchable and retrievable. As discussed in chapter 2, mass digitization projects are notable for their scale and scope, applying character recognition to diverse collections of texts regardless of content, language, or historical importance. In the same way, systems for mass digitization of music documents need to accommodate a wide variety of music document types, including different notation systems and printing methods.

Existing OMR systems are designed for small-scale recognition. Very few OMR systems have been built to process large quantities of music document images, and no purpose-built systems for this task have been developed since the beginning of the mass digitization era. OCR has become the core tool for enabling text-search navigation through document image collections. No OMR systems exist that can provide the same document image navigation as those available for textual materials. This chapter will discuss some of the reasons behind this, and suggest ways to build OMR systems designed for large-scale recognition tasks.

To accomplish large-scale music document recognition, new systems should be built to decentralize, distribute, and co-ordinate OMR processing tasks in such a way that many people, with varying skill sets and in many different places, can work together towards optical music recognition of millions of digitized music documents—a substantial undertaking. While most OMR systems to date have focused on small-scale initiatives, there is almost no literature that discusses the implications of scaling OMR systems to millions of documents.

This chapter will begin with a brief discussion of cost and scale in digitization workflows (§4.1). It will then look at scientific workflow management software (§4.2) and discuss how it can serve as a useful model for designing large-scale music document recognition systems. Three novel techniques for OMR will be described: Distributed OMR (§4.3), Collaborative OMR (§4.4), including opportunities for "crowd-sourced" participation, and Networked Adaptive OMR (§4.5). Finally this chapter will introduce the MEI format for symbolic encoding of music notation (§4.6), suitable for use in a large-scale OMR system.

4.1 Costs and Scale in OMR

One of the primary challenges to address for music recognition is how to design OMR systems that are capable of high throughput (i.e., processing large quantities of page images) while minimizing human involvement. Current OMR systems are designed for small-scale recognition tasks, and have a built-in dependency on human involvement to operate the process on a single workstation. This chapter will describe new ways of designing the OMR process so that a single human can control the recognition of thousands, or even millions, of page images. Similarly, methods of distributing OMR tasks across many different computers and involving many different people are also proposed as a means of optimizing human involvement. These are all designed to ultimately help reduce the costs of large-scale music recognition so that libraries and other institutions can have a practical means of creating electronically searchable representations of their digitized print collections.

Large-scale digitization and recognition initiatives are a costly venture. There have been several detailed studies about large-scale text document digitisation and recognition, focusing especially on costs associated with these projects (the JSTOR and Making of America sections discussed in Chapter 2, for example). The processes associated with digitizing and transforming digital images into searchable, retrievable entities using OCR frequently run in the millions of dollars for large collections. Brewster Kahle (2009), founder of the Internet Archive and the Open Content Alliance, gives estimates of the costs of several book digitization programs. Notably, he estimates that the cost of the Google Book project is between \$5–\$10 per book, which gives a total cost for 20 million books at between \$100 and \$200 million. (Google has not released any official numbers providing these details.) Of these costs, the cost of human involvement, including salaries, tools, and working environments, is the single biggest expense in a digitization initiative (Arms 2000).

There have been no similar studies on the cost of large-scale music document digitization and recognition. However, it is safe to assume that such initiatives would incur similar, if not higher costs due to the greater complexity of music notation and the challenges of reconstructing a symbolic music representation from a page image.

Optimizing OMR workflows for large-scale digitization and recognition requires re-evaluating existing tools to identify ways in which human and computer actors may co-operatively accomplish tasks. Computers do not get bored or require breaks, and as such can be tasked with applying the same operation to many images, but a fully-automated recognition system is likely to produce unacceptable recognition results. Humans, on the other hand, can manage the process and provide the computer with direction, feedback, and discernment. Humans can also work together to co-operate and co-ordinate efforts to produce accurately transcribed materials.

The following design principles for large-scale OMR systems are proposed as ways of producing music document transcriptions from large quantities of document images through the use of both human and machine contributions:

1. **Scientific workflow systems** for consistent, repeatable, and scalable processes to collections of music document page images.
2. **Distributed OMR** to decentralize the OMR process, and distribute the tasks in the OMR process across multiple people and multiple computer systems.
3. **Collaborative OMR**, including crowdsourcing, where multiple individuals are provided with web-based tools to provide correction, verification, or quality control to an OMR process.
4. **Networked Adaptive OMR**, where adaptive OMR techniques can be expanded in the context of networked OMR systems, collecting correction data to improve the accuracy of the recognition process across the entire network.
5. Output format encoding, using the **Music Encoding Initiative** supports storage and retrieval of music notation in relationship to the digitized document images, for use in search and analysis systems.

The remainder of this chapter will examine each of these design principles, and describe how they may be used to create OMR systems that scale to large numbers of page images, while at the same time addressing issues of cost and workload on human participants in the process.

4.2 Workflow Software

Workflows are at the heart of many industrial processes. In their most basic form they are chains of individual tasks that, when joined together form a "pipeline" through which an object passes, with each step producing an output for the subsequent task. Workflows are present in factory assembly lines, business processes, and many other areas where a consistent and reproducible result is desired.

Generally stated, workflows are directed graphs. Nodes in a workflow represent operations or tasks, while directed edges (arrows) represent the flow of data or materials between nodes. A visual representation of a workflow can provide a high-level representation of the overall process by showing these nodes and edges. A simple visual representation of a workflow is given in figure 4.1.

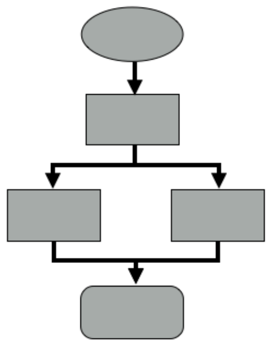


Figure 4.1: Simplified visual representation of a workflow

Several software systems have been developed to design workflows for various disciplines. Business procedures, accounting systems, customer support services, industrial assembly lines, and other process-oriented tasks often have specialized software to model the flow of data or goods through a pipeline of tasks.

Scientific workflow software is a class of applications that is used to construct and execute experimental processes by co-ordinating the flow of data between different data processing tools and data sources. These systems allow scientists to use heterogeneous collections of tools (i.e., software drawn from several different toolkits) to share, co-ordinate, and co-operatively develop experimental processes. These are used in a variety of disciplines to, for example, process data gathered by astronomy telescopes, genetic sequences, or textual data mining. They have also been used for document recognition where scientific workflow systems have been used to define the process for extracting text or document image structure from large collections of page images (Neudecker et al. 2011; Blanke et al. 2012). The key feature of these systems is their ability to chain together different tools, drawn from different toolkits, in an environment where the document recognition process can be designed visually and, when executed, can run these processes on several computer systems with minimal human oversight.

This section will discuss scientific workflow systems, including an in-depth look at how they were used for historical document recognition in the Improving Access to Text (IMPACT) project. The advantages of employing scientific workflow systems for OMR will then be described.

4.2.1 Scientific workflow systems

Scientific workflow software systems were developed as an alternative to the conventional method of experimental data processing using custom-built "glue" software to bind together tools in a one-off system for a single experiment. Gil (2007) refers to the conventional method as "unassisted workflow composition" and notes that this approach has several limitations in terms of usability and scale. These limitations include: the re-usability of the workflow by people other than the original developers of the system, application of the tools to different data sets, tight coupling of the processing steps such that adding or removing a task in a workflow requires intimate knowledge of the underlying implementations of previous and subsequent tasks, and the error-prone nature of manually managed processes. As an alternative, Gil suggests the use of scientific workflow software to manage complex scientific workflows, where custom glue code is replaced with a formalized environment for defining the tools in the process and the flow of data between these tools.

While workflow systems have not been proposed for OMR applications, they have been proposed for other aspects of music information retrieval (MIR). Page et al. (2013) notes that the formalized workflow approach differs significantly from traditional approaches to data processing in MIR. In their paper they examined how researchers applied several popular audio processing tools to research in MIR. In most instances, these researchers chose a single, integrated environment for processing their data, rather than choosing to integrate a wide variety of tools drawn from a heterogeneous collection of processing toolkits. Page et al. posit that the "monolithic," integrated approach was favoured due to the lower "overhead," or effort required to integrate tools drawn from several different frameworks.

> While our study has shown that no single MIR system provides comprehensive coverage across all notions of reuse, it also raises plentiful opportunities for systems that share common concepts to use these as a basis for abstraction and interoperability. Yet ISMIR proceedings indicate little cross-fertilization of most systems beyond the "home" lab and close collaborators. An explanation for this may be the difference between the *potential* for reuse and the overhead of actual implementation. (Page et al. 2013, 450)

They conclude that this overhead, as Gil also suggests, is caused by the lack of formalized methods of interoperability between heterogeneous tools—or, to be more specific, the lack of systems to provide "reusable, repurposable, and repeatable" methods of performing MIR-related research by leveraging heterogeneous collections of processing tools and environments. They conclude that scientific workflow systems can provide this type of environment.

Repeatability and re-usability of a particular workflow is one of the key advantages of using a workflow system for building a data processing environment (Curcin and Ghanem 2008). Conclusions drawn from experimental data can be verified by running the same workflow with different data and checking if the initial conclusions can be verified and generalized. A workflow may be shared with others in order to re-create a process that someone has developed for performing a specific task. Shared workflows may be collaboratively built, creating a mechanism for iteratively refining a process to accommodate different data or investigate new hypotheses (Goble and De Roure 2007).

Several scientific workflow management software applications have been created. These include Taverna (Hull et al. 2006), Kepler (Michener et al. 2007), and Meandre (Llorà et al. 2008) each developed as workflow systems to manage analysis of large amounts of scientific data for use in a wide range of fields, including climate studies, genetics, or astronomy.

All of these workflow systems operate along similar lines. A graphical user interface allows users to visually compose a workflow by arranging (and re-arranging) a graph representation of the workflow. Nodes in the graph represent tools that can be used to perform a given task. Tools may be provided by locally-installed software, or may be provided by a remote system over a network connection. The flow of data between tools is represented by lines that connect the nodes (i.e., the edges of the graph). The user creating the workflow can configure each tool with settings to customize the behaviour of the tool. An example of a graphical representation of a workflow developed in Taverna Workbench is shown in figure 4.2, taken from a figure provided in Hull et al. (2006). A second example showing a workflow in Meandre is shown in figure 4.3 reproduced from Page et al. (2013).

After it is designed a workflow is executed. The user provides one or more input sources (e.g., one or more datasets, depending on the task), and the data progresses through each task, taking input from a previous task and producing an output for the next task. These tasks may be provided by different software tool sets. The same workflow application may be used to assemble workflows for processing genomic data (Curcin and Ghanem 2008), astronomy (Walton and Gonzalez-Solares 2009), or text mining (Llorà et al. 2008). Tools originally designed for searching genomic data may be integrated with generalized interfaces for downloading specific data sets, or provided by a tool for performing a specific statistical analysis.

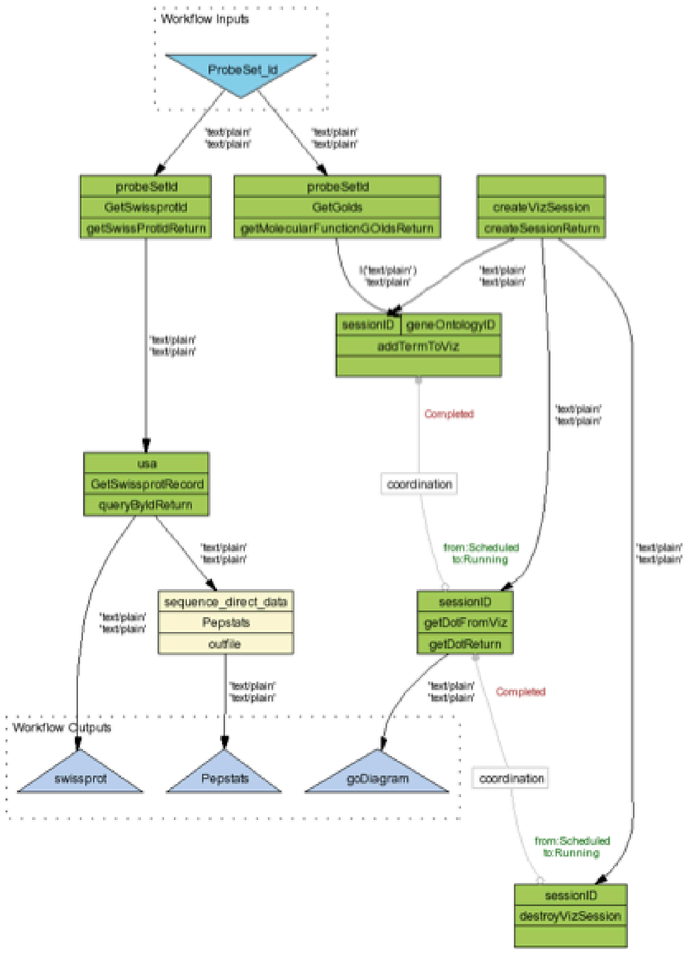


Figure 4.2: Graphical workflow example composed in Taverna. This workflow was extracted from a Grave's Disease case study example (from Oinn et al. 2006).

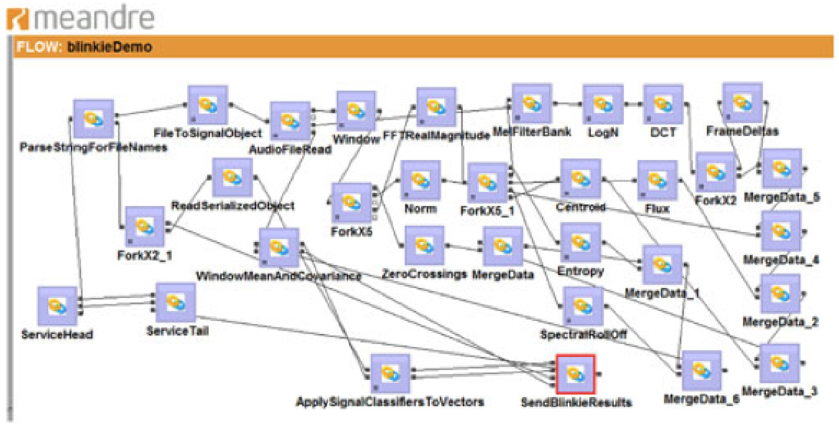


Figure 4.3: Graphical representation of a workflow in Meandre. This workflow was built to provide genre analysis of audio data.

Scientific workflow systems are also used for document image analysis and OCR, bringing different tools together to operate on digital page images. This will be examined further in the next section.

4.2.2 Workflows in Document Image Recognition

A flexible approach to building document workflow systems has been proposed as part of the Improving Access to Text (IMPACT) project (Dogan et al. 2010). The focus of the IMPACT project was to develop textual recognition systems for historical texts that standard commercial OCR software could not accurately recognize. As part of this process, they identified a need for a highly customizable OCR system where they could integrate multiple independent tools for dealing with variations in font faces, page layouts, or page images taken from damaged sources.

4.2.2.1 IMPACT and Taverna

As part of their investigations, the IMPACT project adopted the Taverna scientific workflow system to provide processing "pipelines" for performing document analysis. Using Taverna, they were able to integrate existing tools into custom document image processing pipelines. These tools were installed on remote systems, allowing users to use them without needing to install and maintain these systems independently. Interfaces to these tools were created following a service-oriented architecture (SOA) approach (Perrey and Lycett 2003) so that individual components could be accessed by sending network requests to the host system for that tool. This is shown in figure 4.4.

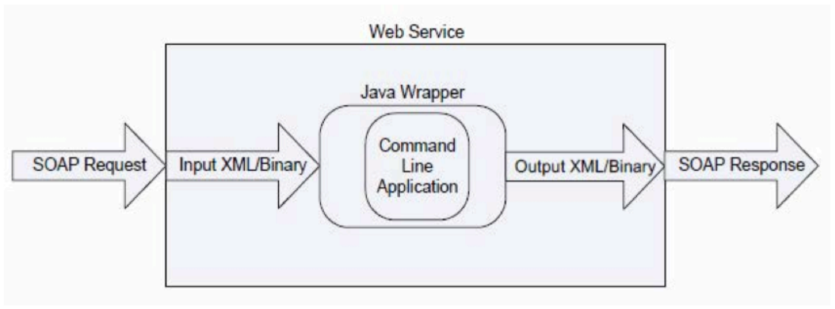


Figure 4.4: Network interface for a command-line application in Taverna (from Neudecker et al. 2011). This is a generic representation of one tool (i.e., node) in a workflow.

In Neudecker et al. (2011) they identified several benefits of this approach over traditional approaches to document image analysis. In these scenarios, a developer is a person or group of people developing new software tools for performing document image recognition tasks, while a user is an individual who uses these tools to design and execute workflows composed of these tasks.

Distributed development

Remotely-hosted services (i.e., server-based software installations) do not require users to install and maintain document image analysis systems on their local machines. Users have access to a large library of document analysis tools, as well as computing resources, with no systems administration overhead for installing or maintaining updated versions. Developers of these systems can make new recognition techniques available to these users by installing them on a single server, making a single installation available to all users. Developers may focus on developing a single tool that can be integrated into users' workflows, rather than developing entire applications.

Demonstration design

The method of building a document recognition system by dragging and dropping objects onto a workspace provides a straightforward visual metaphor for integrating a wide variety of remotely-hosted tools. Users may build their own workflow, or modify an existing workflow, to incorporate new tools as they become available. Using a visual method of building the workflow allows the user to visually "demonstrate" and communicate the design. In non-workflow systems this is typically accomplished by editing the underlying code for the application to add new functionality.

New Service Mashups

In Taverna, workflows themselves may be exposed as complete services. This creates the opportunity for users to incorporate existing workflows into their own, creating a "mashup" composed of many workflows. The tasks, and the software used to perform these tasks, is hosted on a remote machine but network access means that users can integrate services drawn from several sources into a single workflow.

Evaluation

Evaluation services, to measure recognition accuracy or other performance measurements, may be incorporated into a workflow. The IMPACT project developed several tools to automatically compare workflow results with pre-transcribed "ground truth," giving researchers the ability to quickly iterate tool designs to increase accuracy.

An example is shown in figure 4.5 where a simple workflow has been developed to compare the results of an image de-warping process on the resulting OCR results. (This resembles the goal-directed approaches to evaluation discussed in the previous chapter; see also Trier and Jain 1995). Using an automated evaluation task, researchers can incorporate new processes very easily and immediately see a quantitative analysis of the results of these modifications in the recognition output.

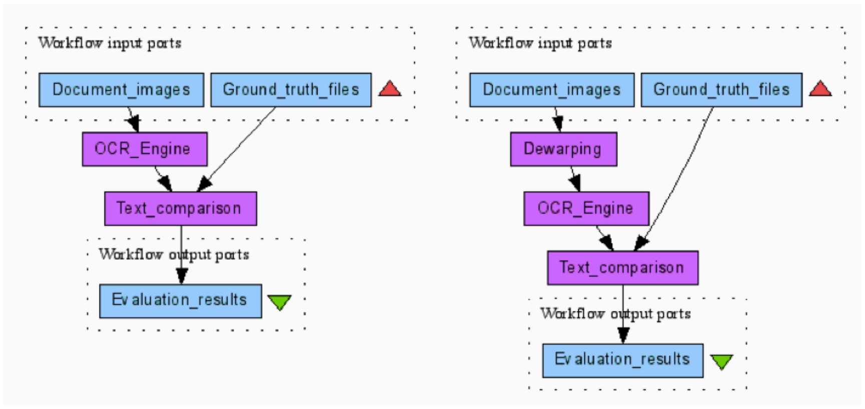


Figure 4.5: Evaluating the effects of image de-warping on OCR recognition results (from Neudecker et al. 2011)

Scalability

Taverna was designed to handle and process very large data sets in disciplines such as bioinformatics and astronomy, and can take advantage of parallel and distributed computing approaches to scale and distribute workloads over several processors, or even several computers. For document recognition workloads, this means that several computers may be employed in executing a single workflow, with a central Taverna instance co-ordinating the distribution of images and collecting results from several systems.

4.2.2.2 *Taverna in Use*

As of this writing, the IMPACT project has made 484 tools available to users of its platform for performing text digitisation (IMPACT Project 2014). Users of their digitisation system can access this library of tools and incorporate them into their own workflows for processing their collections. As of 2011, they reported that over 100,000 page images had been processed by the Taverna workflow system (Neudecker et al. 2011).

While the IMPACT project has demonstrated that workflow software is a promising approach to developing document recognition systems for textual materials, there have been no investigations of their use for music document recognition systems. In the next section we will look at conventional approaches to building OMR software, and propose a novel approach using workflow systems to build customizable OMR processes.

4.2.3 Workflows in OMR

While textual and musical document recognition processes share many similarities, they are sufficiently different in the tools and processes they use to warrant explicit investigation and discussion of a music-specific approach to document recognition workflows. To date, there have been no investigations of the use of scientific workflow systems for OMR.

The conventional approach to OMR systems can be described as a vertically-integrated system, created using the "unassisted workflow composition" techniques discussed previously. As discussed in Chapter 3, there are a wide variety of tools and approaches that have been developed for each stage of the OMR process. However, all existing OMR systems incorporate just a small selection of these approaches, incorporating them into a single, purpose-built OMR application. Users may have the ability to change a few settings to adjust the execution parameters of a given task, but they have no ability to change the order, or composition, of the overall process without editing the underlying code. In most OMR systems designed for general use, modification of the OMR process to use different techniques is difficult or, in the case of closed-source systems, impossible.

In a workflow-based OMR system, the recognition system is the aggregate of the tasks assembled in the workflow, and not a single piece of software. Using workflow management software, custom OMR systems may be assembled from a pool of available processing techniques, with tools available to be "mixed and matched" according to their performance on a given set of document images. The "glue" that binds each task together is the workflow execution environment, and connecting these tasks, from the output of one to the input of another, allows the user to direct data from task input to task output using a heterogeneous collection of tools.

4.2.4 Criticisms

While workflow management systems provide a way of chaining together disparate systems, the tools and interfaces required to represent the process to the user can pose several problems.

One problem is the use of visual programming interfaces for displaying and manipulating the workflow process. Whitley (1997) provides a thorough discussion of the areas and tasks where a visual programming approach may lead to problems with understanding the design of a software application. While these interfaces have been shown to be effective in some cases for learning and understanding logic flow in a programming environment, the "leaky abstractions" (Spolsky 2004) that must be made to translate software design to this visual representation can obscure many of the underlying implementation details and hamper the debugging process.

A second problem is that a workflow system can provide a user with *less* flexibility despite being designed to provide greater interoperability between disparate components. Reijers (2006) presents an anecdotal case for this by observing that workflow management systems in traditional business applications were difficult to set up. Once set up, individuals were reticent to disturb a "precious balance," even at the expense of developing *ad-hoc* software to fill un-met needs. Rijers notes, however, that once established, a workflow system helped to standardize procedures, practices, and information sources among participants in the workflow.

> [Workflow management systems] are widely applied and have become popular because of their positive effects on logistic parameters such as flow time, service time, and resource utilization. At the same time, the promise of bringing flexibility to the work floor has not yet been fulfilled (Reijers 2006, 272).

4.2.5 Summary

This section examined scientific workflow systems and their uses in processing a wide variety of data. A workflow system allows users to design and manage the process of transforming data from one representation to another, for the purposes of extracting specific information from a diverse range of possible data input. Workflow systems were contrasted with the more conventional approach of unassisted workflow composition, where manually-created workflows built using custom, single-purpose "glue" code were critiqued for their limited scalability and re-usability.

The IMPACT project has demonstrated the viability of a workflow approach for building document recognition systems for historical document recognition. This inspired us to create workflow-based approaches to OMR, Rodan (Chapter 5). In an OMR workflow, tasks may be assembled into customized recognition workflows that are tailored to processing a given input and producing a specific output. Tasks that specialize in processing certain types of data, such as degraded images or specialized notation symbol classification, may be combined with other tasks, providing users with the ability to build a processing system customized for the specific needs of a project, corpus, or document collection.

The next three sections will propose several novel principles for designing OMR systems. These techniques are an expansion of the benefits of workflow-centric systems proposed by the IMPACT project, and are proposed as a way to further develop OMR systems that can be used to process large numbers of music document images.

4.3 Distributed OMR

Distributed OMR refers to the distribution of tasks in an OMR workflow across multiple actors. These actors may be human or computer, and all actors may be in geographically different locations from each other, communicating across a network. Unlike desktop-oriented OMR systems, where a human operator controls the software installed on their personal computer, a distributed OMR system allows multiple users to interact with remotely-hosted OMR tools, running on one or more remote server systems. The distribution of OMR tasks across several individuals and across several machines has not been explored as an area of development for OMR systems. This section will review two technologies that enables distributed OMR systems: Web applications, for distributing control and interaction across several users, and parallel and distributed computing for distributing task execution and processing across several computers.

4.3.1 Web Applications

One of the first requirements of a distributed OMR system is the separation of a control surface, or user interface, from the system that performs the OMR. This requirement allows several individuals at different workstations to connect with, observe, and control a remote OMR system. This section will provide a brief background of how web applications function, and a discussion of how they can be used to provide a new kind of OMR system.

The World Wide Web (WWW) was first conceived as a system for delivering remotely-hosted documents, or "pages," to a user's computer (Berners-Lee et al. 1992). These documents are structured using the HyperText Markup Language (HTML), a minimal programming language that provides a method of "marking up" text documents (e.g., delineating paragraphs or headings) as well as the ability to create links to other HTML pages on other remote systems. Web clients, of which the most common is the "web browser," use this markup to render the page in a visually suitable way for a user.

Every document on the WWW is identified by a unique Uniform Resource Locator (URL), or, more colloquially, an "address." These addresses contain enough information to allow the underlying network system to resolve the address to a specific page hosted on a specific server. Users type in an address, or follow a link in a page, and retrieve the desired page hosted by that server.

The protocol governing the transmission of HTML between the client and server is the HyperText Transport Protocol (HTTP) and is designed to be asynchronous and stateless (Fielding et al. 1999). Stateless communication requires that both the client and the server have to assume no prior knowledge ("shared state") of each other. Any communication between a client and a server requires the client to first send a full record identifying it, and the type of information it is requesting. In other words, it does not matter if a client has requested a page one time, or a thousand times—the complete information to identify the client and its request must be sent each time. In the first web browsers, this translated to a need to re-load a page should new information become available on the server.

In 1995, Brendan Eich, working at Netscape, created the JavaScript programming language (W3 Consortium 2012). JavaScript allows a developer to write and execute small programs in the browser ("scripts"), creating a programmatic way to manipulate the structure, appearance, and function of a web page. With the introduction of JavaScript, the static HTML document became a dynamic and interactive interface on which increasingly sophisticated web pages were being built. However, the reliance on the underlying HTTP protocol still dictated that retrieving new information from a server still required a complete re-load of the page.

In 2000, Microsoft added a new application programming interface (API) to the Internet Explorer browser, XMLHTTP. This API was created to allow their web browser-based e-mail client, "Outlook Web Access," to emulate the responsive and dynamic nature of a desktop application (Hopmann 2007). The XMLHTTP API could be triggered by a JavaScript program, and allowed the browser to initiate communication with the server without requiring a complete page re-load. The results retrieved from the server could be dynamically re-integrated into the HTML of the page, using JavaScript, without requiring a complete refresh. This change in communication paradigm did not change the underlying HTTP protocol—all communication was still asynchronous and stateless—but it provided a means of interacting with a remote web server, and dynamically updating a page with new information thereby giving the illusion of synchronous, shared-state communication.

In time other web browser developers, such as Mozilla Firefox and Apple's Safari, adopted the same API within their own browsers. Eventually this mode of communication, and the associated technologies that enabled it, were given the term "AJAX", or "Asychronous JavaScript and XML" (Garrett 2005). Increasingly sophisticated browser-based applications were introduced that emulated locally-installed software, but required no installation and were available through any web browser. The technologies under the umbrella term AJAX were used to emulate a synchronous communication protocol, giving the appearance of a constant communication between a web client (e.g., a browser) and a web server. This, in turn, was used to develop applications that ran in a web browser, but which emulated native software applications. These were called web applications.

Web applications provide a way to separate a control surface (i.e., a user interface in a web browser) from software running on a remote server machine, through the use of APIs that emulate synchronous communication. The next section will discuss this specifically in the context of OMR systems.

4.3.1.1 Web Applications and OMR

In conventional OMR systems, the user interface and the processes that actually perform the music recognition tasks are "tightly coupled" and cannot be separated. This tight coupling of interface and processing system dictates that these OMR applications must be operated and controlled by only one user, on one workstation, at a time (i.e., a user sitting at a single workstation where the software is installed).

In contrast, a distributed OMR system built as a web application, separates the control surface from the processing system. In such a system, the control surface is operated through a web browser, but the image processing and recognition systems are located on remote systems. The separation of control surface from the underlying processing software provides a way for multiple users to interact with the processing software through multiple control interfaces in many users' browsers. The AJAX APIs allow a distributed OMR web application to exhibit behaviours of a traditional application—transparent updates to the state of the application without needing a page reload—while still operating in a networked context.

There are several further advantages to this approach. Software updates to the user interface or to the underlying OMR system can be made available to users immediately, without requiring them to download and install any new software. In a workflow-based system, new tools may be made available to all users of the system. Web browsers are available on all modern computing devices, including tablets and smartphones, as well as traditional desktop and laptop systems. Users are familiar with how browsers operate, and despite earlier problems with web applications that do not work the same way in every browser, most browser developers have been producing standards-compliant systems ensuring consistent behaviour across browsers and computing platforms.

Distributed OMR describes the distribution of tasks among both human and computer-based actors in the OMR process. This section on web applications looked at how web applications work, and how the introduction of AJAX techniques can provide a native desktop application-like interface in a user's web browser to provide a means of distributing access to multiple human actors in the overall process. The next section will discuss how parallel and distributed computing can enable multiple computers to interact and distribute workloads across multiple server systems.

4.3.2 Parallel and Distributed Computing

Parallel processing is the concurrent execution of tasks in a computing system. A parallel processing workload can be distributed among several processing units, either within a single computer or across several physical systems. A distributed computing system is a network of discrete physical computer systems co-ordinated to operate on shared tasks in parallel. The main advantage of parallel and distributed computing is the reduction in processing time required to complete any given task.

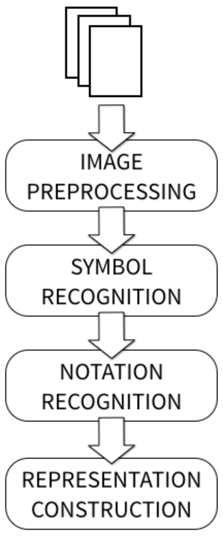


Figure 4.6: A multi-page OMR workflow

Executing a single OMR workflow on multiple pages is an example of an "embarassingly parallel" problem. Each page in the workflow will undergo the same process without a dependence on the previous or next page and so the workflow for each page may be executed simultaneously.

To illustrate, a *sequentially* processed representation of a multi-page workflow is shown in figure 4.6. Multiple pages (shown at the top) are processed in turn by a number of tasks in a workflow. This workflow would require a total time of $w \times p$ seconds, where w is the total time taken to process a workflow of one page, and p is the number of pages to be processed.

Figure 4.7 shows this same workflow, re-factored as a *parallelized* OMR process. Assuming all parallel workflows are processed in a constant w seconds, the total time to process all pages is dependent on the number of parallel processes, n, available to compute the workflow. The time to completely process a workflow approaches a theoretical maximum of $((w \times p) / n)$ seconds. (This is a theoretical maximum, since $n < p$ requires a queue system to hold waiting tasks and assign them to processors as they become available, which requires some overhead in task management and execution.) As $p \longrightarrow n$, the time to compute the results of p workflows approach w, or the time required to compute a single workflow. In other words, the example given in figure 4.7 illustrates that given an equal number of pages and available computers to process each page, the amount of time to execute all workflows on all pages is equal to the amount of time to compute one workflow on one page.

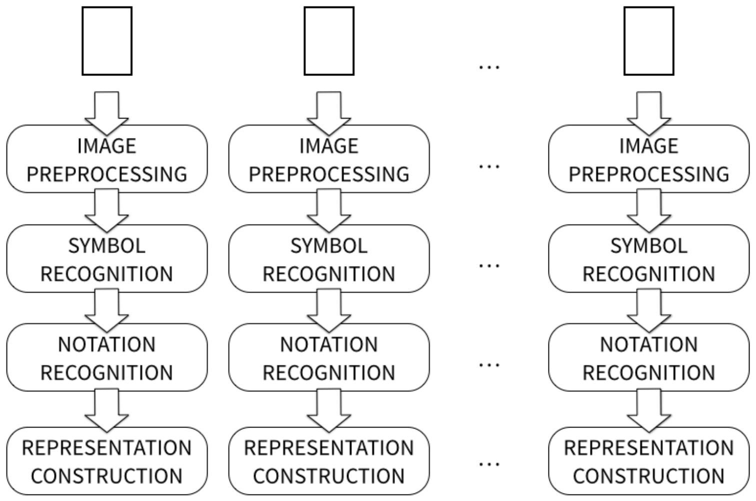


Figure 4.7: A parallelized representation of a multi-page OMR workflow

Parallel processing of tasks within a single physical system is limited by the number of instruction "cores" contained within the system's CPU. However, distributed computing allows a virtually unlimited number of processors to be connected by providing a communication and co-ordination layer over a network-connected group of physical systems. Parallelizable tasks may be distributed across these systems, each capable of running multiple concurrent processes. A group of computers capable of acting in concert on distributed tasks may be referred to as a "cluster," while individual systems within a cluster may be referred to as a "node."

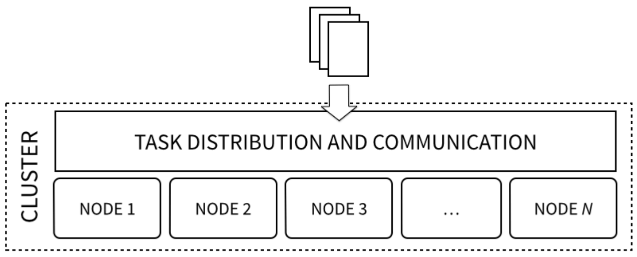


Figure 4.8: Distribution of pages among several nodes in a cluster

A task distribution and communication system monitors the progress of each task (figure 4.8). As tasks are completed, new tasks are automatically assigned from a queue of available tasks waiting to be processed. A machine that has completed a task is automatically given another available task, regardless of which task, in which workflow, it had previously executed. An example of workflow task distribution across a number of nodes is shown in figure 4.9.

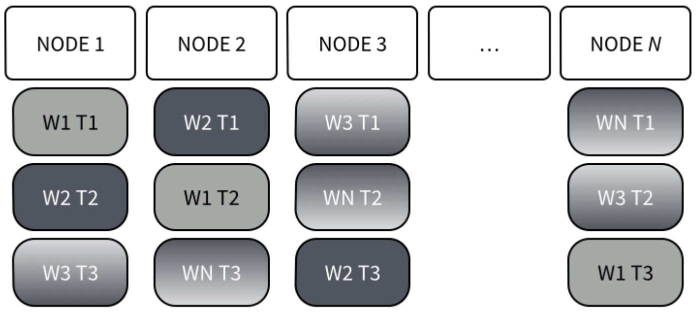


Figure 4.9: 1…N Workflows and three Tasks distributed across N nodes. Variations in shading indicate the workflow (W) to which a task (T) belongs.

Distribution of tasks across several discrete server systems for parallel processing is not a new concept. However, the relatively recent development of "cloud" computing services has created several opportunities for using distributed virtualized computers available in remote data centres over the public internet. These systems are distinguished from traditional distributed cluster computing by providing users with the ability to create *ad hoc* clusters running on remotely-accessible machines.

4.3.2.1 Cloud Computing

The term "cloud computing" was coined as a reference to remotely-hosted computer services available over the public internet. The term seems to have originated from the common practice of representing the Internet as a "cloud," or amorphous shape, in network or process diagrams. The phrase was first coined in a 1996 internal presentation at Compaq Computer Corporation (Compaq Computer Corporation 1996; Regalado 2011).

In 2006 Amazon.com launched a new service, "Amazon Web Services" (AWS) (Amazon.com 2013). This was one of the first instances of a commercially-available cloud platform for purchasing time on systems hosted in remote data centres. The AWS service utilizes the computing power of idle server systems by allowing users to provision *ad hoc* "virtual" servers on demand. This is also known as the "infrastructure as a service," or "IaaS" model. A virtual server is a self-contained instance of computer system that utilizes a portion of the resources available on the underlying physical hardware. A single physical computer system may support many virtualized computer systems. These virtual servers can be created and deleted on-demand, providing a method of quickly commissioning computing nodes to act on a distributed computing task. Costs for these virtual computers are determined according to the amount of hardware resources allocated (CPU, memory, and disk space) and the length of time it is in use.

As a result of cloud services, computing resources have been commoditized as "utility computing." Users interested in creating applications that require distributed and parallel computing over large data sets do not need to maintain physical infrastructure (hardware, cooling, backup power). Additional processing power, storage, or memory may be added or removed as needed up to the maximum available on the host system.

The previously mentioned scientific workflow systems (Taverna, Kepler, and Meandre) are all designed to operate on cloud-based systems. They are capable of co-ordinating and distributing tasks across a network, permitting distributed and parallel computing of large workflows that require large amounts of computing power to process (common in fields such as genomics and astronomy). The next section will discuss the application of parallel and distributed computing for OMR.

4.3.2.2 Distributed computing and OMR

Distributed computing platforms for OMR have not been developed. A distributed approach to task execution in an OMR system would allow potentially large clusters of networked computers to work in concert on an OMR process. When compared to the current system of a single, dedicated workstation with an OMR software installation, this is an obvious enhancement for large-scale music document image recognition initiatives. Cloud computing offers a way to scale an OMR system to accommodate large workloads without needing large numbers of dedicated hardware systems.

The OMR process on multiple page images is an easily parallelized problem. Page images may be treated separately from each other and their recognition results determined in parallel, with no dependencies across page images (i.e., the results of one page are not determined by the results of the previous page). Tasks that may require inter-page dependency, such as determining continuation of musical lines across pages, may be handled in a separate step after the music recognition process has completed.

4.3.3 Distributed OMR Summary

Distributed OMR is a novel approach to OMR that has not been extensively explored in the literature. The distribution of human actors in the OMR process can be enabled by separating the control surface from remotely-hosted OMR systems. This control surface is provided as a web application, running in a user's web browser. Users interact with a remote OMR system capable of executing recognition workflows.

For large-scale OMR, distributed human and computer actors are necessary components for processing large numbers of music document page images. With distributed humans, many individuals may work together to accomplish a single task, regardless of their geographic location. With distributed computing, recognition tasks may be executed in parallel across several networked systems.

Distributed computing has been commoditized with the introduction and availability of cloud computing and "infrastructure-as-a-service" (IaaS) models. An OMR system designed to process large numbers of page images should have the ability to dynamically scale the number of computers available to execute the OMR process on these pages.

4.4 Collaborative OMR

Chapter 2 reviewed several initiatives that used collaboration in text digitization projects to enable groups of people to work together to improve digitized texts. These have become known as "crowdsourced" correction initiatives. The distribution of tasks across many different human actors has provided a level of human oversight and quality control in large text recognition initiatives that was not previously possible due to time and expense. This section will propose collaborative OMR for the same reasons. It will discuss how OMR systems using workflow-based operations provide an opportunity for distributing tasks among several actors in the process, and how crowdsourcing, either to the general public or to targeted specialist groups, can enable correction and quality control beyond the capabilities of any single organization.

4.4.1 Task Distribution

A distributed OMR system allows for shared access to a remote server capable of executing OMR workflows. Many users may log in to the same system from their personal computers and participate in building or executing an OMR workflow.

In large-scale digitization projects, different parties are involved in converting physical page images to a digitized representation. Digitization activities may be spread among many different people, each responsible for one component of the whole process. A library may have a department dedicated to document imaging, or they may enlist the services of a third-party digitization service. Several staff members may be in charge of image or metadata quality control. Recognition and correction activities may be performed by separate groups of people, including the general public. Systems staff may be responsible for importing the results of the recognition process into software for search and display. All of these tasks may be overseen and directed by project managers.

A collaborative approach to OMR can be enabled by distributing tasks to groups of actors participating the overall process, and by providing a shared virtual space and tools that run in a web browser for them to meet and participate in the process. Allowing individuals to focus, or even specialize, in accomplishing a given task provides a division of labour similar to that of an assembly line.

Collaborative activities can be divided into two categories. It may either be explicit, where individuals knowingly participate with each other to accomplish a single task, or implicit, where contributors simply execute a given task without being aware of their collaborators. In the latter case, collecting and utilizing the efforts of many individuals can enable massively collaborative OMR systems that are beyond the means of any single organization. There are many examples of these systems in text digitization and recognition workflows, but to date there are very few that seek corrected data from automatically-recognized symbolic music.

The next two sections will discuss how enabling interactivity in a workflow can leverage the abilities of humans and computers for collaborating on the same tasks in a workflow.

4.4.2 Interactivity in workflow systems

Scientific workflow systems typically separate workflow creation and execution. A user will define a workflow by creating a chain of tools, and connecting them together to indicate how the data is to flow from one tool to another. Once a workflow has been created, a user provides data input and then "runs" the workflow. In most workflow systems, once a workflow has started executing, it will continue until it finishes or meets an error condition. Humans cannot interact with a workflow that is currently executing, and so cannot supply decision-making or discernment information to a task to help the computer complete the task accurately. This has an impact on later tasks in the chain. Should a computer provide a sub-optimal solution to a given problem, it may have an effect on how well later tools in the workflow perform. An interactive workflow system would allow a human to provide input into a currently-executing workflow, and in doing so potentially increase the accuracy of later workflow tasks.

There have been some attempts at introducing interactivity in scientific workflow systems. Sonntag et al. (2010) have proposed merging scientific and business workflow systems for creating interactive workflows that incorporate human interaction with in-process workflows. Cao et al. (2011) have proposed an expansion of the Kepler workflow system to accommodate human interaction in the workflow process. Neither of these approaches have been incorporated as a core component of any scientific workflow system.

Incorporating humans in a workflow execution provides a mechanism to support runtime human involvement in the execution of a task. Automated methods of certain image processing tasks, such as segmenting between musical and lyrical content, or discriminating between foreground and background components, are not perfectly reliable due to variations in image contents and conditions. A human is capable of applying decision making and discernment to assist the process in arriving at the best possible solution.

The next chapter of this dissertation will describe a new workflow system, Rodan, modelled on scientific workflow systems but with the ability to incorporate human feedback in the execution of a workflow. In the proposed interactive workflow system there are two types of workflow tasks: A non-interactive task, which can be computed reliably without the need for human intervention; and an interactive task, which requires the input of a human, either in a verification or a processing role, before the task can be completed and the data sent to the next process in the chain. A hybrid semi-interactive task may pre-compute a possible solution and simply ask the human to verify and, if needed, adjust the computed solution.

An example of a non-interactive task might be cropping or resizing an image to a given size. This requires applying settings prior to workflow execution, but once the workflow is started it may be applied to each image in turn without requiring further input.

An example of a semi-interactive task would be correcting automatically-determined page layout analysis. Layout analysis attempts to differentiate between page elements: the location of staves and systems, lyrics, titles, and other elements. Automated methods for determining these elements exist, but are not always completely accurate. A human can use the result of the automated analysis as a starting point for correcting and verifying that the layout analysis can be used for the next process.

Finally, a fully interactive task might be the creation of a database of labelled symbols, suitable for pattern recognition and machine learning purposes. Humans may supply labels to unlabelled shapes, building a database of known symbol and label groups.

4.4.3 Crowdsourcing and OMR

Initiatives such as RECAPTCHA (von Ahn et al. 2008), The Gutenberg Project's Distributed Proofreader platform (Newby and Franks 2003), or the Australian Libraries' Trove Newspaper Digitization (Holley 2009b) have all provided different approaches to collecting OCR corrections from the general public. These initiatives have provided voluntarily-supplied human-generated data beyond the capabilities of their own staff.

The only known crowdsourced initiative for symbolic music correction is proposed by Dalitz and Crawford (2013). They report on a system for collecting lute tablature correction data from an optical tablature recognition project of 16th-century lute tablature using a web-based correction tool. They report attracting over 50 participants to their project.

There are many possibilities for integrating crowdsourcing into an OMR workflow depending on the audience and the nature of the task. Initiatives like RECAPTCHA may be developed with simple discrimination tasks that can be accomplished by a large audience of non-expert users. These tasks may include shape, position, or colour discrimination which requires no special training and which may be solved in a few seconds. On the other end of the spectrum may be a dedicated platform for collecting large amounts of user-supplied musical corrections, following the model of the Trove Newspaper Digitization project. This platform would present OMR-recognized music and solicit musical corrections from musically-trained participants (i.e., those who can read musical notation).

Since few large-scale OMR initiatives exist, the tools for collecting crowdsourced corrections have not been developed. Chapters 5 and 6 of the dissertation will contain proposals for new tools to enable crowdsourced OMR correction similar to those that have been implemented for text recognition.

4.4.4 Collaborative OMR Summary

A distributed OMR workflow system provides a platform that can be used to enable collaborative efforts in many areas of the OMR process. Collaborations may either be explicit, where participants have a set role in the overall process, or implicit, where participants simply supply answers to a given problem. Both of these types of collaborations are enabled through operating in a shared space in which the results of one individual's work can be used and built on by others.

The next section will discuss how collaboratively-collected data can be used to augment the traditional approach to adaptive OMR by enabling many users to participate in the human feedback component of this technique. This will be referred to as "Networked Adaptive OMR."

4.5 Networked Adaptive OMR

Networked adaptive OMR refers to the use of adaptive OMR techniques in a distributed, collaborative environment. Adaptive OMR (§3.3.2.1) is a technique for incorporating human feedback in the recognition process. With adaptive OMR new symbols may be added to the recognition system, or symbol discrimination can be improved with the addition of more shape exemplars. The introduction of collaborative OMR techniques will allow individuals to co-operatively build these exemplar collections. With networked adaptive OMR, the contributions of a single individual may be used in the recognition tasks of all participants in the network. In a computationally distributed context, virtual machines may be tasked with monitoring incoming training data and applying this data to various machine learning techniques, creating a continuously adaptive OMR system. This creates a system where the results of a single individual's corrections may be applied to the recognition processes of all members of the network. This section will explore some of the potential applications of adaptive OMR in a networked context.

4.5.1 Networked Adaptation

As demonstrated by Pugin et al. (2007a, 2007b) adaptive approaches that specialize in a single source or set of symbols (i.e., a particular font used by a single printer) can result in significant improvements in recognition accuracy by creating a recognition engine specialized in that particular set of symbols. They demonstrated that an adaptive OMR system, Aruspix, can be adapted to the typeface used in a single book—or, more correctly, a specific musical font—yielding a recognition system that is specialized for symbol recognition of that particular source. A "book adaptive" or "book dependent" approach was shown to yield a significant improvement in recognition accuracy over a "book independent" (i.e., generic) model trained on data gathered from a more varied data set. In a networked adaptive OMR system this same technique may be applied at a larger scale, enabling the creation and storage of "book-adaptive" models for previously recognized and corrected musical materials.

In a distributed, collaborative OMR system, the incoming correction data, possibly supplied by a crowdsourcing interface, can be used to develop libraries of specialized recognition models. As users submit corrections drawn from a variety of sources, these data may be used to create ground-truth libraries that enable recognition of a given typeface or symbol set. A library of these specialized models may be maintained such that individuals who submit images containing notation for which an trained classifier already exists do not have to go through the additional step of submitting corrections for the purposes of increasing recognition accuracy.

In addition to specialized typeface or symbol recognition models, continuously-adaptive systems can leverage collectively-created data sets to apply pattern recognition and classification to new contexts. Pugin (2006a, 2006b) notes that in addition to typeface recognition, the recognition system used in Aruspix can model musical sequences and predict likely (and unlikely) transitions between notes as an additional error-checking and validation step in producing accurate recognition output. Models that specialize in particular genres or instrumentation can detect unlikely output and either automatically correct it, or alert a human user to a particular area where it has detected a likely error.

4.5.2 Networked Adaptive OMR Summary

Adaptive OMR systems have been shown to significantly increase recognition accuracy on heterogeneous collections of music notation images but, to date, these systems have developed as conventional software installations on discrete workstations. Bringing adaptive OMR techniques into a networked workflow-based OMR context offers new opportunities for collecting and utilizing ground truth data supplied by others to simultaneously improve the recognition results of all members of the network.

Continuously-adaptive OMR in a distributed system is a novel approach to document recognition, even among text recognition initatives. It offers new opportunities for integrating multiple classifier systems to provide competing hypotheses for symbol classification. It also serves to reduce the amount of effort required to build a trained classifier in adaptive OMR—a very labour-intensive process—by allowing training data to be shared across the entire network.

4.6 The Music Encoding Initiative

The goal of OMR is to convert a visual representation of music to an encoded symbolic music file. This chapter and the previous chapter have discussed the mechanics behind producing these symbolic encodings, but the specific output of the OMR process as encoded symbolic notation has not been considered yet. This is a critical component of all OMR systems, as the information stored in the encoding is crucial for enabling functionality in subsequent software systems for searching, analysing, displaying, and manipulating the results of OMR.

This section will present the Music Encoding Initiative (MEI) as an especially suitable format for OMR output encoding. The MEI is an open-source effort to define a system for encoding musical documents in a machine-readable structure. The MEI closely mirrors work done by text scholars in the Text Encoding Initiative (TEI) (Text Encoding Initiative Consortium 2014) and, while the two encoding initiatives are not formally related, they share many common characteristics and development practices (Roland 2002). The MEI, like the TEI, is an umbrella term to simultaneously describe an organization, a research community, and a markup language. It brings together specialists from various music research communities, including technologists, librarians, historians, and theorists in a common effort to discuss and define best practices for representing a broad range of musical documents and structures. The results of these discussions are formalized into the MEI schema, a core set of rules, expressed as an eXtensible Markup Language (XML) schema, for recording physical and intellectual characteristics of music notation documents.

The MEI offers a flexible and extensible format that may be used to describe a wide variety of notation types in a way that maintains both the structure and semantics of a particular notation type, as well as the spatial relationships between the notation symbols and the page image from which it was originally derived. Few standard symbolic music encoding formats provide formalized methods of encoding and synchronizing visual and symbolic representations. As such, MEI is the most promising symbolic encoding scheme on which to build systems for searching and navigating document image collections.

This section will provide a short overview of the principles of notation encoding, and then discuss the MEI format providing further details on how the MEI can be used in an OMR system.

4.6.1 Notation Encoding

The wide variety of encoding formats and approaches to music representation may be attributed to the complexity of music notation itself. Music notation conveys meaning in multiple dimensions (Carter et al. 1988; Dannenberg 1993). This complexity creates significant challenges for creating a comprehensive standard that can represent all dimensions accurately. As a result, most notation formats focus on a particular application or musical style. Some formats focus on visual representation (i.e., object placement for printing), while others focus on representing the logical structure of the notation itself, with no visual information. Still others, such as Csound (Vercoe 1991), contain data that represent the performed sonic properties of the score, with little information to connect it to either the visual or a recognized conventional symbolic notation.

Maxwell (1981), for use in the Mockingbird music notation editor, outlined a model which described three separate domains: *physical, logical,* and *graphical.* This model was later extended by the Standard Music Description Language (SMDL) (International Standards Organization 1995), an encoding system that was expressed using the Standard General Markup Language (SGML) (Goldfarb and Rubinsky 1990) (SGML was a precursor to later representations using XML (Bray et al. 2006b).) SMDL built upon the Mockingbird model and extended it to define a representation model consisting of four domains for music notation representation: *logical, gestural, visual,* and *analytical.* Although SMDL was not widely adopted, these domains remain useful for separating the functions of symbols within a music representation system.

In this model, the *logical* domain includes the musical content or structure including pitches, time values, articulations, dynamics, and all other elements—defined as the symbols that communicate the composer's intentions. The *gestural* domain relates to a performed interpretation of the logical domain (i.e., it encodes information that may be added by a performer such as explicit realizations of "swing" or *rubato*). The *visual* domain describes the contributions of an editor, engraver, or typesetter, and encodes information about the physical appearance of the score, such as symbol locations, page layout, or font. Finally, the *analytical* domain covers commentary and analysis of the music document in any of the three previous domains.

The MEI maintains these distinctions in its design. It is possible to, for example, encode the structural function of a note separate from its visual appearance on the page, or encode a performance realization separate from the written notation. Each of these domains may be encoded using elements and attributes inside a hierarchical text structure: the "encoding." More importantly, a given encoding may be validated against a pre-defined schema to ensure that it conforms to these rules. This process is explained in the next section.

4.6.2 XML Representation

XML is a hierarchical encoding system. An example of a very simple MEI XML hierarchy is shown in figure 4.10, showing both the XML representation and the same structure using a "tree" diagram. Elements are the core objects in an XML representation, and are represented using "tags." These tags are a name enclosed in angle brackets (e.g., a note object is represented using the `<note>` tag). These tags may be nested within each other such that every tag (except for the "root," or top-most, element) has one and only one "parent." Each element may contain zero or more "children." The child elements of a single parent are all "siblings" (or "peers") of each other. Extending the family metaphor further, a parent of a parent (and so on "up" the tree) are known as "ancestors," while children of children are called "descendants." For the sake of brevity and clarity, this section will use the term "structure" to refer to a collection of elements that are all descended from a single parent element, and form a sub-tree of a more complete encoding.

```
<mei meiversion="2013">
    <meiHead>
        <fileDesc>…</fileDesc>
    </meiHead>
    <music>
        <body>
            …
        </body>
    </music>
</mei>
```

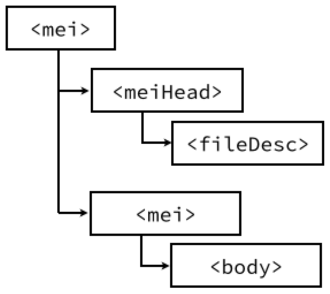


Figure 4.10: Example MEI as XML (top) and as a tree representation (bottom)

Attributes of an element are used to define properties of a particular object and are represented as key-value pairs, taking the form `key="value"`. A `<note>` object that represents "middle C" (C4) on a keyboard would be encoded in MEI as `<note pname="c" oct="4" />`. (In prose, attribute and element names are distinguished by an additional "`@`" character on the attribute name, (e.g., `@pname`), even though this symbol does not appear in the XML encoding.)

At a minimum, a valid MEI-encoded file contains two structures within the parent `<mei>` element: the `<meiHead>` and `<music>` elements. The `<meiHead>` structure contains elements that describe the work (i.e., metadata), including information about authorship, encoding standards, and provenance. The `<music>` structure contains information regarding the encoded music itself. The music notation is represented using XML tags, arranged in a hierarchical relationship. One possible encoding of a small fragment of music notation is shown in figure 4.11.

```
<beam>
  <note xml:id="d1e129"
      stem.dir="up"
      pname="f"
      dur="8"
      oct="4" />
  <note xml:id="d1e130"
      stem.dir="up"
      pname="c"
      dur="8"
      oct="4" />
</beam>
```

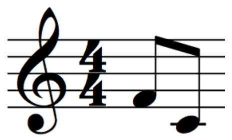


Figure 4.11: Left: XML Encoding; Right: Possible graphical representation (NB: Clef and Time Signature are given for reference and are not present in the encoding)

The MEI provides support for musical encoding by grouping the symbol definitions into modules. These modules define the elements and the rules on how these elements should interact to support a wide range of music notation encoding features. Several notation systems are supported, including common Western music notation (CWMN), mensural notation, neume notation, and guitar and lute tablature. Other modules provide support for analytical markup, editorial processes, and extended CWMN elements. A full list of the MEI 2013 modules is available in the Music Encoding Initiative Guidelines (Music Encoding Initiative Council 2013).

The structure of an MEI file can be validated to ensure that the XML used to express an encoded representation of the notation follows the rules set out by the MEI guidelines. This process, which is common across all XML representation formats and not just MEI, uses a schema that encodes the rules and behaviours of the elements and attributes, governing where and how they may be used. There are several XML schema languages, such as DTDS, W3C XML Schema, or RelaxNG (Quin 2010). They are used in conjunction with validation software (e.g., `xmllint`, Veillard 2014) to validate whether a particular document conforms to the rules of a given schema. This can be used to determine whether an encoded document will cause problems for interoperability between different systems.

For example, an MEI file that contains lyric information as a subset of a `<rest>` element would not pass the validation process, since this does not conform to the rules that the MEI schema defines for the behaviour of both the rest and lyric elements. This behaviour is encoded in the schema to formalize the way that the music itself is known to function; in most repertoires, a lyric item attached to a rest is a musical error.

Yet in some cases, including *avant-garde* notation, composer-specific repertoire, or ancient notation, it may be desirable to customize the behaviour of the MEI schema to include rules for validating the behaviour of uncommon practices for music notation. The MEI accomplishes this using a process called "schema customization," where the behaviour of the encoding can be altered to produce new schemas.

4.6.3 Schema Customization

MEI is expressed in a meta-schema language developed by the Text Encoding Initiative (Hankinson et al. 2011b). The "One Document Does-it-all" (ODD) format (Burnard and Rahtz 2004) is designed to encode both the behaviours of the document encoding, and the human-readable documentation following a "literate programming" technique (Knuth 1984). From a single ODD file, a formalized schema in one of the previously-mentioned XML schema languages, as well as human-readable documentation for that schema, may be derived. The MEI schemas and guideline documentation are created using the Roma software developed to support the TEI project (Burnard and Rahtz 2004).

To generate a schema and documentation, two files must be provided to the Roma processor. The first is the MEI "core" ODD file. This contains the rules and definitions of the behaviours of all elements supported by MEI, in all modules. The second is a "customization" file. This file, also written using ODD, is used to modify the encoding features supported in the generated schema, either by altering the behaviour of the core MEI elements or by defining new ones. A customization file may also specify that entire modules in the MEI core are not necessary to include in the resulting schema. This provides a mechanism for generating dedicated schemas for validating only CWMN notation documents, and rejecting documents that encode, for example, neume or mensural notation, or *vice-versa*. These customization files may be shared with other users, allowing co-operative development of customized encoding systems for different repertoires. The customization process is shown in figure 4.12.

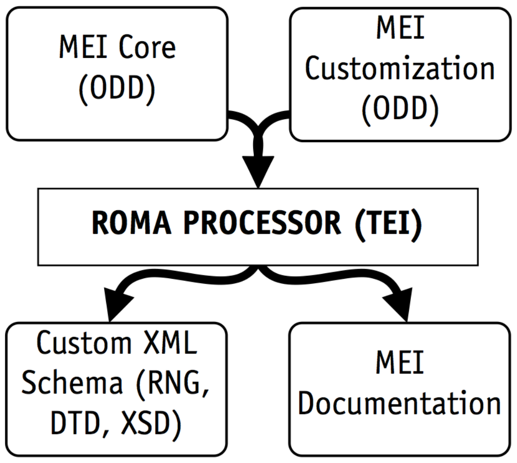
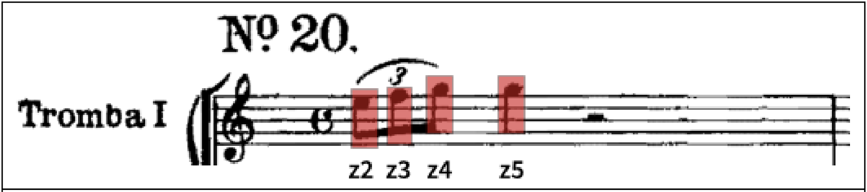


Figure 4.12: The ODD customization process

This customization method is unique among music encoding systems. Customization and validation provide a flexible but testable means of producing MEI encodings that conform to a particular set of musical rules, and a formalized method of customizing and extending MEI make it especially suitable for encoding heterogeneous music document collections. Should new features be needed to encode a particular type of notation, it is not necessary to develop an entirely new document encoding system to support this notation. Instead, users may add or modify the behaviour of the MEI core to adapt the encoding system to support new musical features.

4.6.4 Synchronizing Media

OCR encoding systems that maintained correspondence between the transcribed symbol (i.e., text) and the location on the image where that text occurs have been previously discussed (§2.6). To enable a similar approach for music, MEI can be used to capture the correspondence between a transcribed music symbol and its location on an image. A highly simplified example of the MEI encoding for this is given in figure 4.13, where note elements are related to zone elements through the use of the `@facs` attribute.



```
<zone id="z2" ulx="100" uly="50" lrx="120" lry="80" />
<zone id="z3" ulx="130" uly="45" lrx="150" lry="75" />
<zone id="z4" ulx="160" uly="40" lrx="180" lry="70" />
<zone id="z5" ulx="210" uly="40" lrx="230" lry="70" />

<note pname="e" oct="5" facs="z2"/>
<note pname="f" oct="5" facs="z3"/>
<note pname="g" oct="5" facs="z4"/>
<note pname="g" oct="5" facs="z5"/>
```

Figure 4.13: Simplified example of music and image correspondence in MEI

Before examining the mechanics of image and notation correspondence, however, it may be useful to examine the generic mechanism used by the MEI to synchronize many media types with symbolic music, both spatially expressed (i.e., images) as well as temporally expressed (i.e., sound, video). For OMR we are primarily interested in spatial alignment but this section will discuss both spatial and temporal alignment strategies in MEI since they use similar methods.

Although MEI is expressed in XML, which is an inherently hierarchical format, there are occasions where music notation is not best served by adhering to a strictly hierarchical design. This mechanism is used throughout MEI, and is the mechanism used to synchronize media, but it may be easily understood using the example of a slur over several notes. Since the slur involves multiple notes, the correct relationship between slur and note is not hierarchical. The slur is not a child of any single note, yet it is also not really correct to suggest that the slur is the "parent" of each note. This may be further complicated when considering multiple groupings under the slur, such as eighth-note beams. Is the slur a child of a beamed group, or is it the parent? If the encoding required adherence to this purely hierarchical system, the encoding would create implicit relationships that simply are not musically correct.

The MEI approach circumvents this hierarchy by providing a system of "pointers" that can reference elements outside of a given hierarchy. This system uses attributes that may refer to another element's unique ID. In the example of a slur, a separate `<slur>` element outside of the note hierarchy can "point" to the member notes in a way that makes the relationship unambiguous but non-hierarchical. A simplified example of this is shown in figure 4.14.

```
<beam>
    <note xml:id="n1" />
    <note xml:id="n2" />
    <note xml:id="n3" />
    <note xml:id="n4" />
</beam>
...
<slur startid="n1" endid="n4" />
```

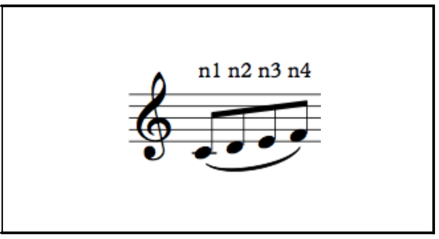


Figure 4.14: A simplified non-hierarchical method of encoding a slur

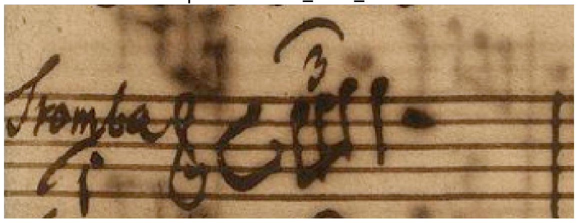
For synchronising multiple media types, MEI uses a similar method. Separate hierarchical structures contain descriptions of the media, and attributes are used to point between the unique identifiers of the structures to relate one element to another. An element that describes a symbolic notation feature—for example, a `<measure>` element—may contain attributes that point to other elements present in a hierarchy that describes that element's position in space or time. The `@facs` attribute, for example, may be used to point from the `<measure>` element to a `<zone>` element that defines pixel-based coordinates in relation to an image. Similarly, the `<measure>` element may also employ the `@when` attribute to point to a `<when>` element within a separate `<timeline>` structure that contains references to points in time within the media file. An example of both spatial and temporal alignment is shown in figure 4.15. Combining both spatial and temporal alignment, an encoded symbolic representation can supply the necessary synchronization points for measure-by-measure, or even note-by-note, highlighting on a page image as an audio or video performance of the work is played.

```
<facsimile>
 <surface>
   <graphic xlink:href="bwv_232_print.tiff"/>
   <zone xml:id="z1" uly="58" ulx="21" lry="64" lrx="23"/>
   <zone xml:id="z2" uly="63" ulx="35" lry="65" lrx="36"/>
 </surface>
</facsimile>
...
<timeline avref="bwv_232.mp3" origin="w1">
 <when xml:id="w1" absolute="00:00:01.00"/>
 <when xml:id="w2" absolute="00:00:04.05"/>
 <when xml:id="w3" absolute="00:01:00.00"/>
</timeline>
...
<measure n="1" when="w1" facs="z1"/>
<note when="w2" facs="z2"/>
<chord when="w3" />
```

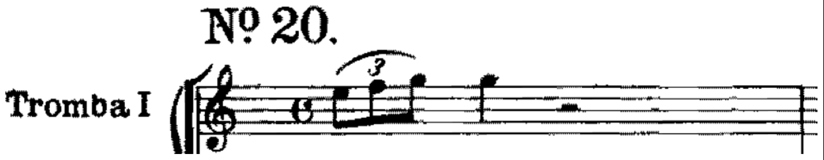
Figure 4.15: Spatial and temporal alignment in MEI. The emphasized attributes illustrate the linking and relationship mechanism between the elements in different parts of the MEI encoding.

This method, however, can only be used to relate a single type of media (i.e., an image or a recording) with a musical structure. If multiple media of the same type are present this method does not allow synchronization between them. The reason is that the `@facs` attribute can only make reference to a single image, and the `@when` attribute to only one timepoint. To address this an alternate method is provided that changes the *direction* in which the attributes point. In the first example, the attributes point from the symbolic representation to the alignment information. By reversing the direction from the alignment element (the `<zone>` or `<when>` elements) to the symbolic element with the `@data` attribute, the encoding can accommodate multiple media representations pointing to the same musical element. The `@data` attribute on the media definition holds a reference to the `@xml:id` of the notated object. An example of multiple page image alignment is given in figure 4.16. In this example, two images—one a manuscript and one a printed version—are related to the same musical structure.

Excerpt of bwv_232_ms.tiff



Excerpt of bwv_232_print.tiff



```
<facsimile>
 <surface>
   <graphic xlink:href="bwv_232_ms.tiff"/>
   <zone data="m1" uly="58" ulx="21" lry="64" lrx="23"/>
   <zone data="n1" uly="63" ulx="35" lry="65" lrx="36"/>
 </surface>
</facsimile>
<facsimile>
 <surface>
  <graphic xlink:href="bwv_232_print.tiff"/>
   <zone data="m1" uly="58" ulx="21" lry="64" lrx="23"/>
   <zone data="n1" uly="63" ulx="35" lry="65" lrx="36"/>
 </surface>
</facsimile>
...
<measure xml:id="m1" n="1" />
<note xml:id="n1" />
```

Figure 4.16: Multiple page images aligned with a single symbolic encoding

Aligning multiple music representations, incorporating symbolic, visual, and sonic media, has the potential to foster new means of interacting with music source materials in a digital context. The ability to spatially align symbolically-encoded notation and images is a necessary component of building music document image retrieval systems using the results of an OMR process, but the ability to align multiple temporal representations may also be used to provide new ways of interacting with transcribed symbolic scores.

4.7 Chapter Summary

This chapter has discussed several new techniques for OMR in an effort to identify ways to design new systems to handle large recognition workloads. Large-scale recognition initiatives are costly, and the most significant costs are centred around areas of human involvement. As a result, one of the first considerations of large-scale initiatives should be to examine the overall process and look for areas where human involvement can be optimized. Current OMR systems are designed for a single human operator at a single workstation, processing a single page image at a time. To optimize human involvement there are several areas where computers can play a larger role in managing and applying processes to a greater number of images, and where a human, or groups of humans acting together, can observe and manage the parts of the OMR process with much greater effect on the overall throughput.

Scientific workflow systems have been designed to apply a consistent and repeatable result on large data sets. These systems were examined for their usefulness and application to document recognition processes. These workflow systems are notable for their flexible approach to visually building processing pipelines, and for providing opportunities to share and build on workflows created by others.

Three new techniques for OMR were presented. Distributed OMR was presented as a way of decentralizing OMR systems and allowing many actors, both human and computer, to take part in the process. Collaborative OMR, including crowdsourced participation, is one result of the distribution of tasks, allowing many people to participate in the same process regardless of geographical location or institutional affiliation. Finally, Networked Adaptive OMR builds on established techniques in adaptive OMR. It proposes using distributed and collaborative systems to build shared datasets capable of being used as training data for adaptive OMR systems. These data can be used by machine learning algorithms to build constantly-improving systems for optical music recognition, allowing all participants in the network to take advantage of work done by others for improving their own recognition tasks.

The MEI notation format was discussed as the most promising candidate on which to construct systems for searching and navigating document image collections. Features for aligning multiple representations, both spatial and temporal, were examined as a way to enable image and notation correspondence, similar to previously discussed formats developed for text recognition initiatives. Additionally, MEI may be used to build custom notation encoding formats, a requirement for dealing with the variety of music notation systems that will be encountered in any large-scale recognition initiative.

The next chapter will present Rodan, a web-based OMR workflow system that was built to provide an implementation of these approaches to OMR.

5. Rodan

This chapter introduces Rodan, a workflow-based prototype OMR system designed using the principles of distributed, collaborative, and networked adaptive OMR. This chapter begins with a history of the development of Rodan (§5.1), design patterns, data types, and communication protocols (§5.2) and then continues with a look at the specific implementations of the Rodan server (§5.3) and the Rodan client (§5.4).

5.1 Development History

The *Liber usualis* project, described in full in Chapter 7, was a project to perform OMR and OCR on the complete *Liber usualis,* a 19th Century Roman Catholic service book containing chants and texts used in the Roman liturgy. This project required a number of software tools and involved several individuals over a period of approximately six months.[7] After completing the project, we formalized the workflow that had emerged; this is shown in figure 5.1. The lessons drawn from the *Liber usualis* project, especially in co-ordinating tasks, people, and computing resources to complete the chain of steps for each page image, were what inspired the creation of a formalized OMR workflow system composed of interchangeable and interoperable tools.

In the *Liber usualis* project, the co-ordination between tasks and people was an example of "unassisted workflow composition" (Gil 2007; discussed previously in §4.2.1). Several customized OMR and OCR tools were created and assembled in an *ad hoc* fashion. For ex-

7. These individuals are credited in the acknowledgements of the dissertation.

ample, the automatic page layout analysis step was performed by a custom-built version of the Aruspix software, adapted to accommodate musical source material written using four-line staves. A custom-built Python script passed each page image to this version of Aruspix, which automatically attempted to segment the regions of the page, producing a file for each page containing regions highlighted in colours according to their function—text, lyrics, musical content, or other page elements (figure 5.2). The output files were stored on a shared network drive. A research assistant then manually corrected the automatic layout segmentation for every page image.

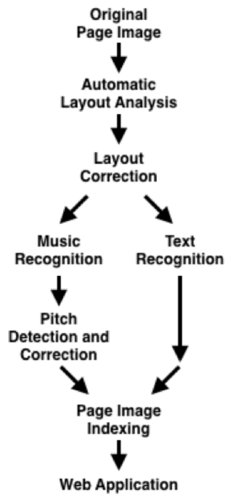


Figure 5.1: Workflow for the *Liber usualis* project

There were several challenges to the manually-managed workflow system. The first challenge was defining and developing the workflow itself. Many tasks in the workflow involved some form of computational manipulation of data, but the tools and techniques necessary for performing these tasks were developed or introduced on an "as-needed" basis. As such, many tools were custom-built for the *Liber usualis*, and had little use outside of the specific workflow in which they were developed.

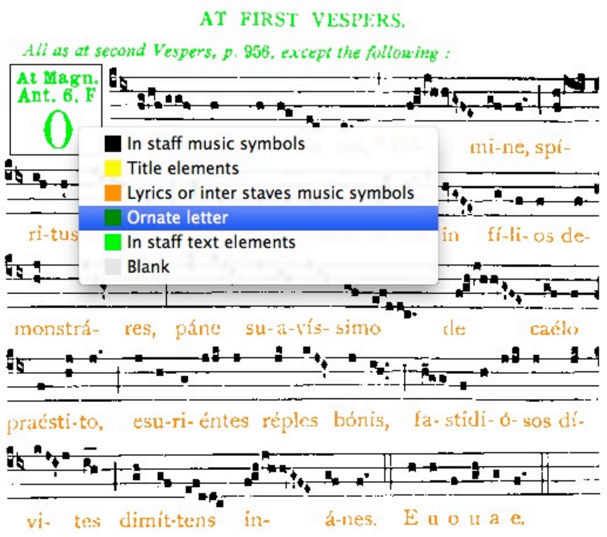

AT FIRST VESPERS.
All as at second Vespers, p. 956, except the following :
At Magn. Ant. 6. F
O
In staff music symbols
Title elements
Lyrics or inter staves music symbols
Ornate letter
In staff text elements
Blank
mi-ne, spí-
ri-tus
in fí-li- os de-
monstrá- res, páne su-a-vís- simo de caélo
praésti-to, esu-ri- éntes réples bónis, fa- stidi- ó- sos dí-
vi- tes dimít-tens in- á-nes. E u o u a e.

Figure 5.2: A page of the *Liber usualis* segmented using Aruspix

Another challenge was managing data transformations among team members, co-ordinating file versions and determining what work was completed or waiting to be completed. This included ensuring that each person knew which files they could work on, what needed to be done to them, and where the results of their task should be placed so that they could be used in subsequent tasks in the workflow.

A third challenge was the inevitable software failures and bugs due to the custom nature of the tools used to perform the processing. Since many of the tools we were using were being custom-developed for a given step in our workflow, we would only become aware of problems in the software once a number of pages had been processed, sometimes even after hundreds of pages had been manually corrected. One particular example stands out to illustrate this. There was a bug in one of our tools where the image dimensions, after cropping and de-skewing, were improperly calculated and stored. The result of this error was that the symbolic OMR results could not be aligned with the original images since we had lost the data to re-align the images after these image transformations. Unfortunately we only noticed this after processing and manually correcting several hundred pages, as the error did not present itself until we attempted to align symbol and image data. To fix this without re-doing the manual corrections (several person-weeks worth of work) we had to re-calculate the page dimensions for all files, and then merge the already-corrected files with the files that still needed to be corrected to produce a single corpus from which our human annotators could work.

Ultimately, each of these challenges had an impact on the reproducability of the overall workflow. After completing the *Liber usualis* project, we were asked to expand this technique to perform OMR on other sources. Unfortunately, the custom nature of the tools, and the "glue" scripts built outside of a formalized environment for constructing a workflow resulted in an over-specified system for performing OMR on a single source. After the *Liber usualis* experience, it was realized that a new approach was needed where workflows, and tasks within these workflows, could be centrally defined and managed, and performed within a shared context.

5.1.1 Experimentation with Taverna

In November 2011 I attended the IMPACT project's final conference at the British Library in London. There I saw a presentation on the IMPACT Interoperability Framework (IIF) (Neudecker 2011; Neudecker et al. 2011), a tool that used the Taverna workflow system to create and run document recognition and OCR workflows. This was the inspiration for building a similar system for OMR. We attempted to build a similar system for OMR by implementing a very simple Gamera-based image manipulation workflow in Taverna, using the IIF. A screenshot of our workflow in the Taverna Workbench application is shown in figure 5.3.

The Taverna environment was the first implementation of our workflow-based OMR system, but we determined that one desired feature was missing. In our *Liber usualis* workflow we were able to incorporate human feedback within the workflow, providing the ability for a human to correct the results of automated processing before passing the images on to the next step in the workflow. Additionally, experiments within our lab on adaptive OMR had shown the value of incorporating human feedback in the OMR process, increasing recognition accuracy while simultaneously reducing overall costs (Pugin et al. 2007a; Pugin et al. 2007b). With a Taverna workflow, it was not possible to pause an executing workflow to wait for human input. Workflows, once started, would continue executing until they had finished or encountered an error condition. These environments did not support the introduction of a human actor in the workflow to react to previously-computed tasks in the workflow. Moreover, a cursory study of the existing workflow software systems at the time did not yield any systems that permitted interactivity in the workflow. We began investigating alternative workflow solutions, and eventually decided to build our own.

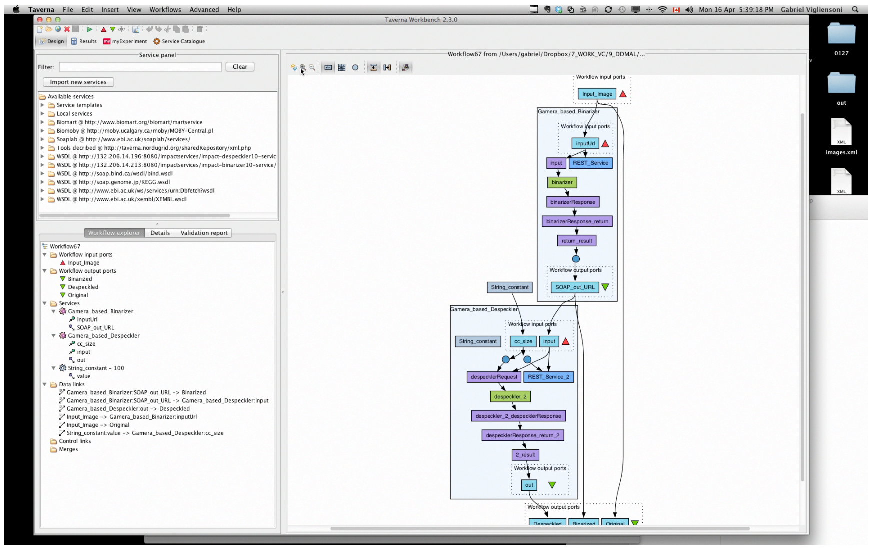


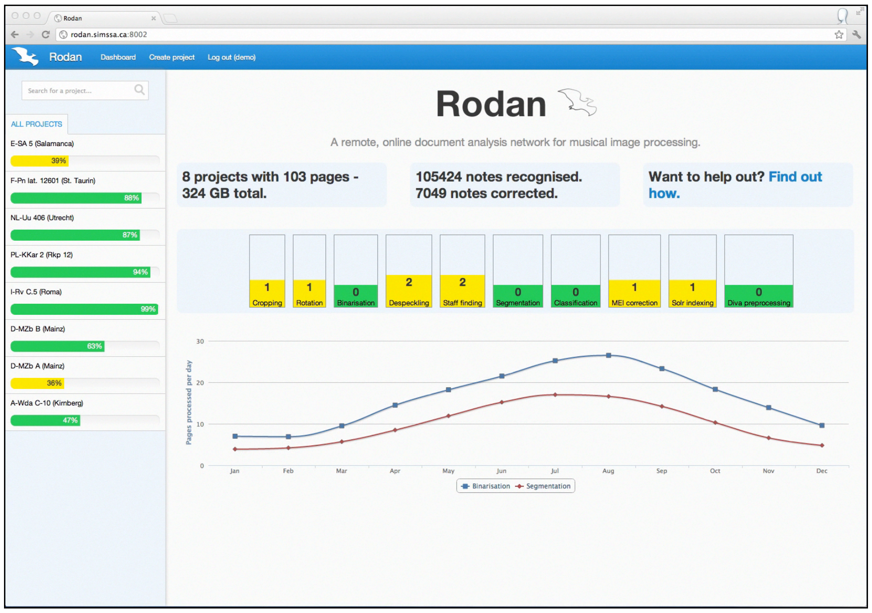
Figure 5.3: Simple Gamera image manipulation workflow in Taverna

5.1.2 Initial Version of Rodan

The initial version of Rodan[8] was developed during the summer of 2012. The front page of the web application is shown in figure 5.4. The concept of the workflow was the central feature retained from the initial work with the Taverna system. In Rodan, users could define workflows by chaining together a number of available processing jobs (figure 5.5). These jobs were then executed, in order, on each page image. Users of the Rodan system would be alerted if there were interactive jobs that required human involvement before a workflow could continue executing (shown in figure 5.6). Users could click on the indication of a waiting job, perform a task, and then submit their answers. The system would then continue executing the workflow until another interactive task was encountered.

In the autumn of 2012, Rodan was re-written to integrate several enhancements to the application. This included a clean separation between the client and server components with a well-documented API connecting these components, a simpler task management and execution subsystem, and greater flexibility in workflow definitions.

8. The name "Rodan" is taken from a fictional Japanese kaiju monster, similar to the origins of the name Gamera. Rodan, a pterosaur, could be said to be a "cloud-based" creature, much like a web-based distributed OMR system ("OMR in the cloud").



Rodan Dashboard Create project Log out (demo)

ALL PROJECTS

E-SA 5 (Salamanca)	39%
F-Pn lat. 12601 (St. Taurin)	88%
NL-Uu 406 (Utrecht)	87%
PL-KKar 2 (Rkp 12)	94%
I-Rv C.5 (Roma)	99%
D-MZb B (Mainz)	63%
D-MZb A (Mainz)	36%
A-Wda C-10 (Kirnberg)	47%

Rodan

A remote, online document analysis network for musical image processing.

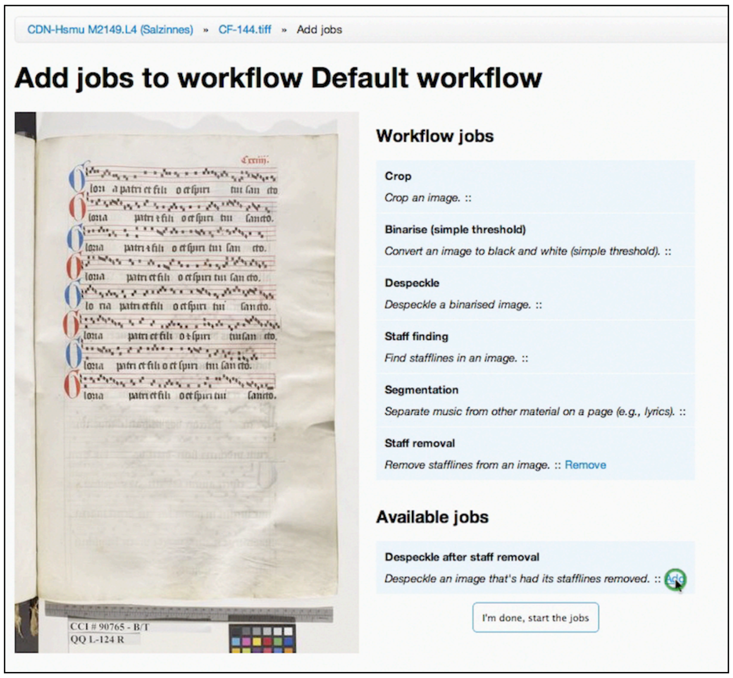
8 projects with 103 pages - 324 GB total.

105424 notes recognised. 7049 notes corrected.

Want to help out? Find out how.

Cropping	Rotation	Binarisation	Despeckling	Staff finding	Segmentation	Classification	MEI correction	Solr indexing	Diva preprocessing
1	1	0	2	2	0	0	1	1	0

Figure 5.4: The first Rodan web application front page



CDN-Hsmu M2149.L4 (Salzinnes) » CF-144.tiff » Add jobs

Add jobs to workflow Default workflow

Workflow jobs

Crop
Crop an image. ::

Binarise (simple threshold)
Convert an image to black and white (simple threshold). ::

Despeckle
Despeckle a binarised image. ::

Staff finding
Find stafflines in an image. ::

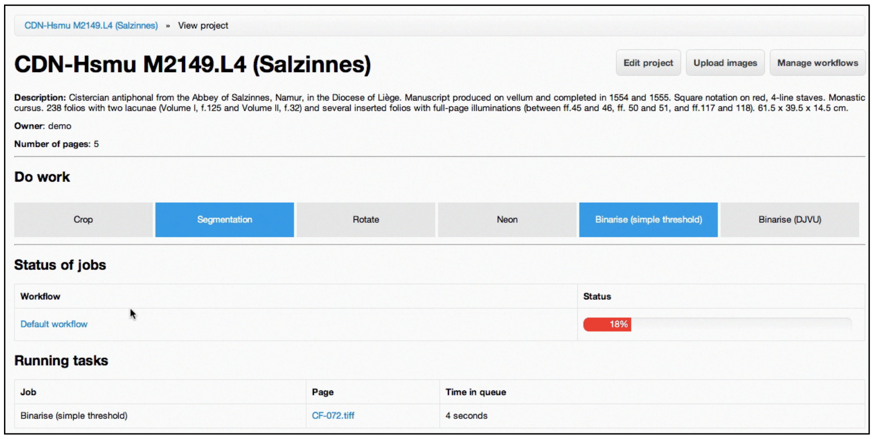
Segmentation
Separate music from other material on a page (e.g., lyrics). ::

Staff removal
Remove stafflines from an image. :: Remove

Available jobs

Despeckle after staff removal
Despeckle an image that's had its stafflines removed. ::

Figure 5.5: Workflow definition screen of the first version of Rodan



CDN-Hsmu M2149.L4 (Salzinnes)

Description: Cistercian antiphonal from the Abbey of Salzinnes, Namur, in the Diocese of Liège. Manuscript produced on vellum and completed in 1554 and 1555. Square notation on red, 4-line staves. Monastic cursus. 238 folios with two lacunae (Volume I, f.125 and Volume II, f.32) and several inserted folios with full-page illuminations (between ff.45 and 46, ff. 50 and 51, and ff.117 and 118). 61.5 x 39.5 x 14.5 cm.

Owner: demo

Number of pages: 5

Do work

Status of jobs

Workflow	Status
Default workflow	18%

Running tasks

Job	Page	Time in queue
Binarise (simple threshold)	CF-072.tiff	4 seconds

Figure 5.6: Two jobs waiting for human intervention (highlighted in blue; the jobs are “segmentation” and “binarise”).

5.2 The Design of Rodan

The current version of Rodan is developed as two components. The Rodan server component is a web application that runs on a remote server system and manages the underlying image processing and music recognition tools. The Rodan client is a graphical user interface built to run in a web browser. It acts as a control surface for a user to interact with and control a remote Rodan server instance.

An overview of the Rodan client and server architecture is provided in figure 5.7. Working from left to right in the diagram:

- People (“users”) interact with clients. This may be the “official” Rodan client web application, or other clients (e.g., a smartphone or tablet application) designed to interact with a Rodan server.
- Interaction between the Rodan clients and a server takes place over the public internet (the “cloud”) using the HyperText Transfer Protocol (HTTP)
- Clients interact with the Rodan server using a well-defined Application Programming Interface (API). The API manages bi-direc-

tional (send and receive) communication with the client applications.

- The Rodan server has three components. The API sends status information and retrieves control messages from clients to control the server, the Object-Relational Mapper (ORM) system manages a persistent storage layer in the form of a relational database, and the task queue system co-ordinates workflow processing among nodes in a cluster, distributing OMR tasks to be processed with assorted image processing and OMR toolkits.

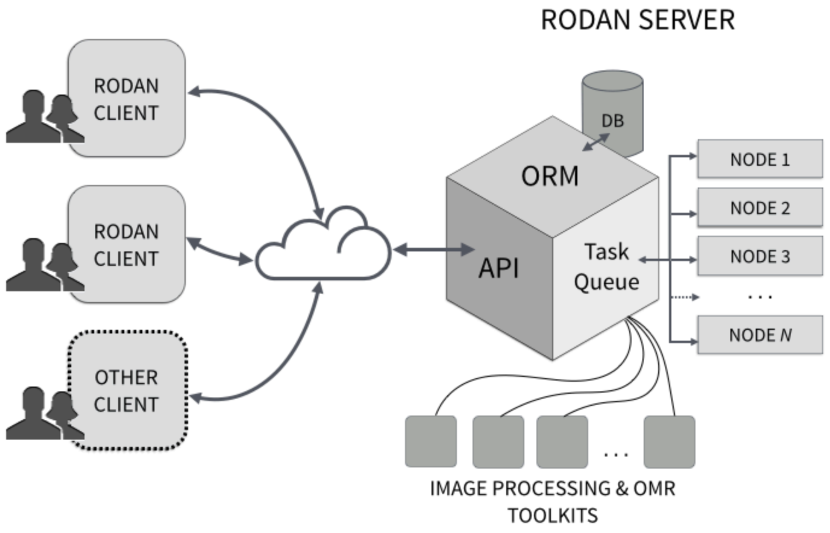


Figure 5.7: Overview of the Rodan Client and Server architecture

This section will describe the design patterns employed in the construction of both the client and the server applications, and give an introduction to the communications protocols used in communication between the Rodan server and client applications.

5.2.1 Model-View-Controller

The Model-View-Controller (MVC) design pattern governs the structure, representation, and manipulation of data in an application. This pattern was introduced by Trygve Reenskaug during a period where he was a visiting scientist at Xerox PARC in 1978–9.

> The essential purpose of MVC is to bridge the gap between the human user's mental model and the digital model that exists in the computer. The ideal MVC solution supports the user illusion of seeing and manipulating the domain information directly. The structure is useful if the user needs to see the same model element simultaneously in different contexts and/or from different viewpoints. (Reenskaug 2010)

MVC was first popularized in the Smalltalk-80 programming language (Krasner and Pope 1988). It has since become one of the most important design patterns for structuring and modelling data and user interactions within applications.

MVC makes a distinction between three types of objects in an application. Model objects define the data that are stored and manipulated in the application, and therefore represent a large part of the underlying structure and purpose of the application. View objects are responsible for displaying the data, and presenting controls (e.g., buttons) for interacting with the system. Controller objects control interactions between models and views, taking actions initiated by the user and routing it to the appropriate actions for managing and manipulating the data. A representation of this is given in figure 5.8 (adapted from the Apple Cocoa Design Patterns documentation, Apple Computer 2012). The Rodan server uses the Django application framework (Django Project 2014) to manage the MVC interactions in a web-based environment. In this environment, models represent persistent data stored in a database.

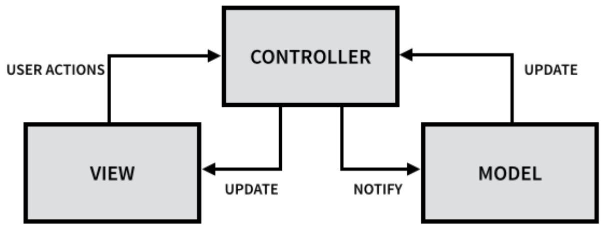


Figure 5.8: Model-View-Controller Representation

Since the models describe the application state and the data contained in the application, understanding the model layer is fundamental to understanding the structure of an application. The next section will present the Rodan models and describe how they interact with each other.

5.2.2 Entity-Relationship Diagram

The relationship between models in Rodan is presented as an Entity Relationship Diagram (ERD) shown in figure 5.9. This graphical representation illustrates the properties of, and relationships between, model objects in Rodan. Lines that connect the models are terminated with indications of how the models are related to each other, as indicated in the legend.

5.2.3 Rodan Models

Each model object defines the data stored in instances of that model. An instance of a model is an occurrence of a particular record. For example, the Project model defines how data for individual projects are stored, while an instance of the Project model would structure the data describing a particular project. In a Django application, model instances are mapped to the underlying database storage system through an Object-Relational Mapper (ORM). When a new object is created (e.g., a new project), a reference to this object and its properties (name, creator), is created and stored in the database.

All model objects are uniquely identified by a Universally Unique

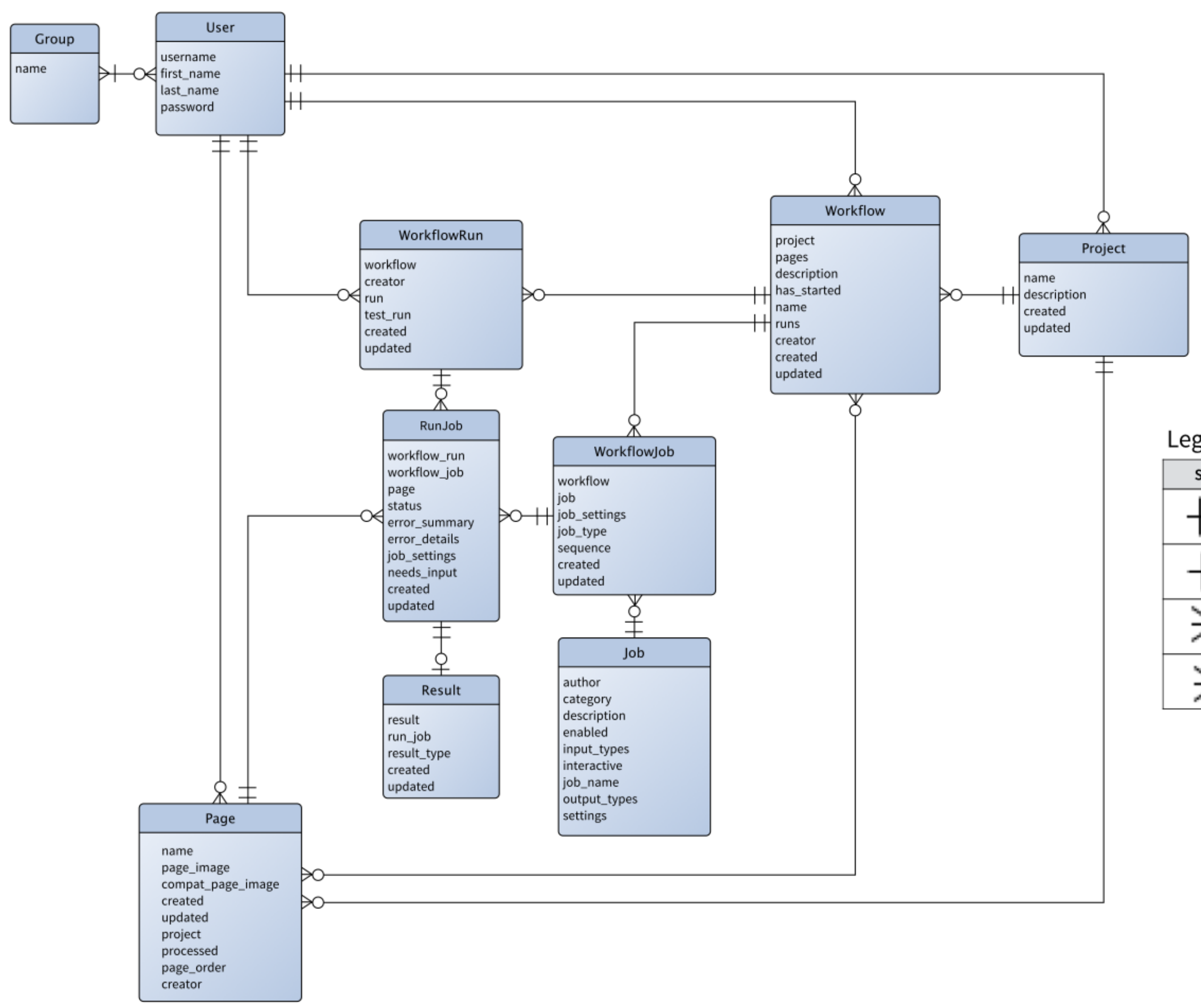
ID (UUID) to ensure that entities within a single Rodan installation can be uniquely identified in a global context, and across distributed systems, without needing significant central co-ordination. The UUID is a 128-bit randomly-generated alphanumeric string that has a very low probability of introducing identifier collisions (i.e., it is highly unlikely that two randomly-generated UUIDs will be the same). An example of a UUID would be "`38a65ed6-931c-428b-b837-4a34e62dd52b`". The UUID for all objects is used as the primary key identifier in the database.

5.2.3.1 Users

User objects represent a human actor in the system. Users are identified by a user name and password, and can authenticate and "log in" to the system. Permissions and group assignment, using the built-in Django authorization system, are used to govern the activities in which a User may participate.

5.2.3.2 Projects

A Project is the basic organizational unit in Rodan. A Project must be created to organize resources and workflows into logical groups. For example, a project would be used to organize all recognition activities around a particular book, or a particular collection. Users may be restricted to accessing only certain projects in the system, providing a way of restricting access to projects for which they do not have permissions.



Legend

Symbol	Description	Example
┼○	Zero or One	A Run Job may have zero or one Results
┼┼	One and Only One	A Workflow Run must have one and only one creator
>○	Zero or Many	A Workflow may have zero or many Workflow Runs
>┼	One or Many	A user must belong to one or many groups

Figure 5.9: The Rodan Entity Relationship Diagram

5.2.3.3 Pages and Images

A Page represents an image, the basic object for analysis in a document recognition system. A Page has a one-to-one relationship with an image, but it also contains extra information about the image in both a recognition context and in its relationship with other Page images. Multiple Pages may belong to a single project. Pages are given a sequence number as a way of ordering the images, often corresponding to the order of pages in a book.

Pages are created when a user uploads an image to the Rodan server. In this process, a "compatible image" is automatically created. This is a derivative image that has been translated to a standard, lossless image format. The compatible page image format in Rodan is the Portable Network Graphics (PNG), (Roelofs 1999). The compatible image simplifies processing the image data by providing a known image format derived from the wide variety of formats that users may upload (e.g., JPEG, TIFF, GIF). Another automatic process creates several thumbnail images from the compatible image, allowing smaller image representations to be displayed in the interface.

5.2.3.4 Jobs

Jobs represent abstractions of individual manipulation tasks that may be applied to pages. A simplified example of a Job definition is provided in figure 5.10. Job objects use a dot-delimited name as an indication of the tool that provides the processing task. In the example below, "`gamera.toolkits.rodan_plugins.rdn_rotate.rdn_rotate`" indicates that this job is provided by the Gamera toolkit, but is part of a third-party toolkit ("`rodan_plugins`") and provides the "`rdn_rotate`" function (the last component of the name is always the function provided by that particular job).

```
URL: http://example.com/job/f6769a3c9a394a4998f8acb9e5fc4f0b/
JOB NAME: "gamera.toolkits.rodan_plugins.rdn_rotate.rdn_rotate"
SETTINGS: [
   {
     name: “angle”
     type: “int”
     default value: 0
   }
],
INPUT TYPES: {
    pixel types: [0, 1, 2, 3, 4, 5]
},
OUTPUT TYPES: {
    pixel types: [0, 1, 2, 3, 4, 5]
}
CATEGORY: “Rotation”,
INTERACTIVE: True
```

Figure 5.10: A sample Rodan job definition for interactive rotation

“Input Types” and “Output Types” specify the types of data that this Job can accept and produce. The “pixel types” are used to identify the particular types of data the job may accept, process, and produce. This is a misnomer, since Rodan jobs may accept and produce non-image data, but is presently named for historical and compatibility reasons. A more accurate name would be “data types.” A list of data types currently accepted by Rodan jobs is provided in figure 5.11.

```
ONEBIT = 0
GREYSCALE = 1
GREY16 = 2
RGB = 3
FLOAT = 4
COMPLEX = 5
MEI = 6
JPEG2000 = 7
# (a `package` is used to
produce multiple outputs from
a single job.)
PACKAGE = 8
GAMERA_XML = 9
```

Figure 5.11: Rodan job data input and output types

The “Settings” field stores an array of settings for executing a particular job. Each setting defines a name, a default value, and an expected data type for the setting value. In the example, the job takes one parameter, identified as "angle," with a default value of "0" (no rotation) and expects a signed integer. This particular job definition is set to be interactive (identified by the "interactive" parameter). Once this Job is placed in a Workflow (described in the next section) this parameter identifies that a particular job definition is designed to pause the workflow and wait for human input. The human input will modify the value of the settings to provide, in this case, a custom rotation angle. For organizational purposes a job is assigned to a category to group similar functions. In this case, the rotate job is, unsurprisingly, assigned to the "Rotation" jobs category.

5.2.3.5 Workflows and WorkflowJobs

In Rodan, a Workflow can be thought of as an organizational object that relates a collection of processing tasks and a collection of page images to be processed by these tasks.

A Project may contain multiple Workflows. Each Workflow may operate on one or many Pages. Several Workflows may be defined to process the Page images in a project. Users may organize their Workflows and Pages to provide custom processing tasks for features unique to a subset of images. To give an example, a user may wish to create two Workflows for processing a single book, designed to accommodate differences in camera frame orientations or image size introduced in the digitization process. Thus one Workflow may be tailored to processing the *recto* page images, and the other tailored to processing the *verso* page images.

In Rodan, a distinction is made between the *definition* of a Job (described above), and its existence in a particular workflow. When a Job is assigned to a workflow, an instance of another model is created, the "WorkflowJob." When a user adds a Job to a Workflow, the information in the Job object is copied to a new instance of a WorkflowJob and added to the workflow. The default settings of a Job are copied to the WorkflowJob.

A distinction is made between objects that represent the definition of the Workflow, and objects that are created in the process of running a Workflow (see figure 5.13). Workflows, Jobs, WorkflowJobs, and Pages are the objects that are used to define the function of a Workflow prior to execution, while WorkflowRuns, RunJobs, and Result objects are created when executing a workflow.

5.2.3.6 WorkflowRuns and RunJobs

Once a Workflow has been created by defining a sequence of WorkflowJobs, the user can execute the Workflow, where each Job processes each Page attached to the Workflow. When a Workflow is executed, the system creates a new instance of a WorkflowRun, which represents a running workflow. This object is created to store the status of a running Workflow. A Workflow may be executed several times, each time generating a new WorkflowRun object. Since the Job settings, and indeed the entire Workflow definition, may change between successive runs, each WorkflowJob definition is used to derive a new instance of a RunJob object, containing the settings used to process an image.

A RunJob represents the application of a processing task to an image, and as such represents the core of the processing system. A RunJob can have several states, shown in figure 5.12. Many of these states are self-explanatory. States that may require further explanation are `WAITING_FOR_INPUT` and `RUN_ONCE_WAITING`. A RunJob that is waiting for input is an interactive job that is waiting for a human to complete a task so that it may finish executing. A `RUN_ONCE_WAITING` RunJob is a particular feature of the co-operative nature of task execution in Rodan. In this case, a preliminary operation has been applied to a particular image, and the RunJob is waiting for a human to correct or verify the solution before continuing.

```
NOT_RUNNING = 0
RUNNING = 1
WAITING_FOR_INPUT = 2
RUN_ONCE_WAITING = 3
HAS_FINISHED = 4
FAILED = -1
CANCELLED = 9
```

Figure 5.12: Run Job states

An example of a RunJob in a `RUN_ONCE_WAITING` state is a process for staff segmentation. A particular staff-line recognition algorithm has automatically identified the locations of each staff on the page for the purposes of segmenting the musical content from other page elements (e.g., textual, lyrical). (A picture of this is shown in figure 5.15 along with a description of the interactive task.) The automated pass is designed to minimize human involvement by providing a preliminary solution to the segmentation task. The system is then set to hold the task and wait for the human to verify, and possibly correct the automated solution.

Separating Workflows and WorkflowJob from the WorkflowRuns and the RunJobs objects provides a mechanism for accessing the results of all previous workflow runs, as well as the settings used to compute every result in past runs. This allows the results of different settings to be compared, to determine the settings that produce the most desirable output in a workflow.

5.2.3.7 Results

The Results object holds an image or data that represents the output of a RunJob instance. If a RunJob is expected to produce output for the next job in the workflow, it is the Result object that is used as input for the next job.

5.2.3.8 Summary

An overview of the process of defining and running a workflow is shown in figure 5.13. Entities involved with defining a workflow are shown above the “start” line, while the entities created when a workflow has started execution are represented below this line.

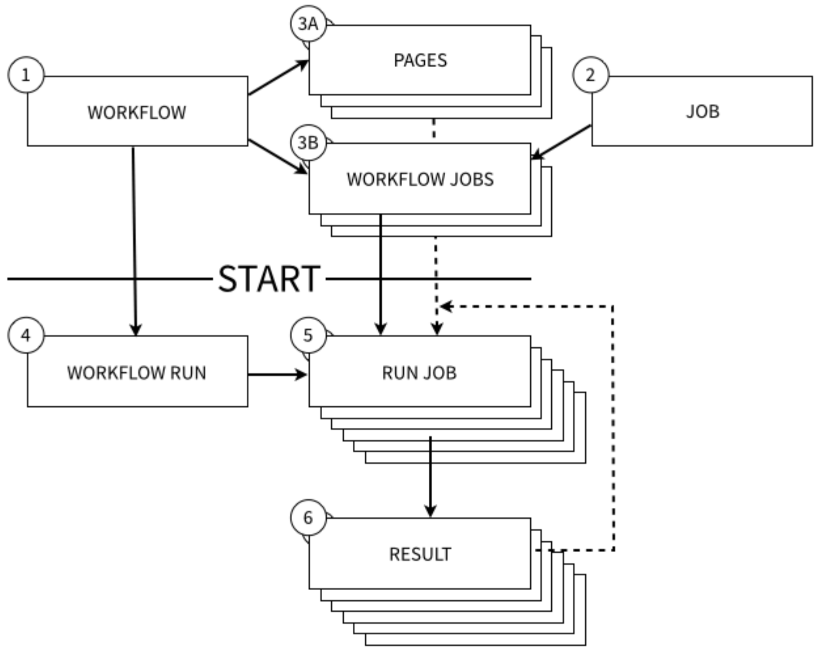


Figure 5.13: Rodan workflow system overview

A Workflow (1) is composed of Pages (3A) and WorkflowJobs (3B). Jobs (2) provide a representation and definition of every processing task available in the system. Thus Workflows are combinations of Pages and Jobs that are applied to a workflow, or WorkflowJobs. After a Workflow is executed (“start”), a new WorkflowRun (4) is created. WorkflowRuns are ordered collections of RunJobs (5), with each Run-Job instance representing an instance of a WorkflowJob and a Page (for the first task in the Workflow) or Result image (i.e., the output of the previous task). A RunJob produces a Result (6), and this Result may be used as the input to the next RunJob in the sequence (represented by the dotted line connecting Results to RunJobs). The initial RunJob in a workflow is paired with a Page image to provide the initial input into the workflow (represented by the dotted line connecting Pages to RunJobs).

The workflow system is the central component of Rodan, and provides a way for different, independent image processing and music recognition tasks to be chained together to form bespoke OMR systems with processes tailored for individual recognition projects. The next section will describe how the workflow system is managed and controlled through a HyperText Transport Protocol (HTTP) API.

5.2.4 Application Programming Interface and REST

An application programming interface (API) is a specification for interaction between software components. Through an API, software can provide input for a certain operation, control a running process, or inquire about the status of a process.

The Rodan server component features a web API that defines how other systems can interact with the server over the HTTP protocol. Rodan is designed to separate the user interface from the underlying OMR system implementation. All functions, including authentication, uploading pages, and creating and running workflows, are operated through the HTTP API.

5.2.4.1 Representational State Transfer (REST)

The Rodan API follows the Representational State Transfer (REST) design (Fielding 2001) to govern the structure of the API. REST was designed to take advantage of the underlying mechanisms of HTTP, and in doing so, simplify and formalize the mechanisms with which two pieces of software can interact over the World Wide Web (WWW).

REST is not, in itself, a protocol. Instead, it is a description of how

HTTP can be used to design a remote service API. The author of REST, Roy Fielding, was one of the original authors of HTTP.

> REST'S client-server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the effectiveness of performance tuning, and increases the scalability of pure server components. Layered system constraints allow intermediaries—proxies, gateways, and firewalls—to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance via large-scale, shared caching. REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cacheability (Fielding 2001, 98–9).

Adhering to a RESTful design allows a developer to maximize the underlying mechanisms of HTTP for addressing and interacting with remote resources. It also leverages the existing infrastructure that makes up HTTP-based communication, simplifying the mechanism of sending and receiving messages.

There are several alternatives to REST for web service protocols. One of the most widely used alternative is the Simple Object Access Protocol (SOAP) (Gudgin et al. 2007). The SOAP protocol defines a platform for exchanging information between two remote systems using a defined message format, implemented as an XML-structured file. Messages may be transferred over a network connection to request information from a remote system, or trigger an action of some sort. Message encapsulation allows SOAP to be transport-agnostic. It may be used over a wide variety of communication protocols including HTTP, the Simple Mail Transport Protocol (SMTP, used for e-mail), or even the Transport Control Protocol (TCP), the "raw" communication protocol of the Internet.

However, when designing a system that is designed to be operated exclusively over HTTP, employing a SOAP-based protocol leads to a large amount of duplication of effort. The HTTP protocol contains several pre-built mechanisms to define how systems may interact. SOAP-based APIs largely ignore these mechanisms. SOAP-based systems need dedicated software to manage these interactions on both the client and the server, while REST, since it is based directly on the HTTP specification, will work with a wide range of standard web browsers and web servers. For Rodan, we chose REST to avoid incompatibilities and reduce the amount of extra software needed to manage the communication process between the Rodan server and its client applications.

5.2.4.2 Resources and Identifiers

In Rodan, models (§5.2.3) are exposed as resources in the API. In a RESTful design, the resource is the central organizing principle. A resource is an object that can be identified by a unique, global identifier (typically a Universal Resource Identifier, or URI)[9]. These resources are addressed using Element URIs, pointing to a single instance of a resource, or Collection URIs, a list of all available resources of a particular type. To give an example, the URI for all jobs available in a given Rodan installation may be accessed at:

```
http://example.com/jobs/
```

When this URI is requested the server will respond with a list of all job objects in the system. To access one particular job instance, a client may use the URI of the job object—a globally unique identifier (truncated for space):

9. A URI is a globally unique identifier for a particular object (a unique name). A Universal Resource Locator (URL) is a location where that resource may be found (a unique address). In most systems, URIs and URLs are the same thing, since a globally unique name, and a globally unique address, can be one and the same.

```
http://example.com/job/f6769a3c9a...e5fc4f0b/
```

This will return a representation of a single job object, for further use (display or manipulation) by the client (these representations will be discussed later in §5.2.4.4).

5.2.4.3 HTTP Methods

To tell the server to take an action, the Rodan API makes use of the pre-defined HTTP methods, or "verbs." These verbs identify actions that the client wishes to invoke on the server, and include GET, POST, PUT (or PATCH) and DELETE. Every HTTP request includes one of these methods. GET is the most frequently used method, as it is used to request information from a server. In our previous example we used the following URI:

```
GET http://example.com/job/f6769a3c9a...e5fc4f0b/
```

In practice this would be used by a client, such as a web browser, to indicate that we were requesting a particular resource from the server. Other HTTP methods may be used to manipulate resources available at a given URI. To delete a particular job, the server would require a different verb:

```
DELETE http://example.com/job/f6769a3c9a...e5fc4f0b/
```

Request methods have expected actions. The designer and the user of a REST API can therefore rely on convention, rather than documentation, to understand the expected actions and responses to a particular method. These responses include:

- **GET** When sent to a Collection URI (a URI that is expected to provide a list of objects) returns the list. When sent to an Element URI (a single object) returns the object.
- **DELETE** Will delete all objects in a collection, or a single element.
- **PUT** or **PATCH** Typically used to modify a single resource. The

PUT method specifies that a complete record must be sent to an Element URI, replacing the record available at that address. The PATCH method supports partial updates, so a client may simply send the modified fields to update the resource. (The PATCH method is a proposed extension of HTTP and is not a part of the standard to date. It is, however, supported by many newer software systems and is expected to be standardized in the next version of the HTTP protocol)

- **POST** Creates a new resource. A POST request sent to a Collection URI is expected to return a newly-created resource with the URI to address it. A POST request to an Element URI is rarely used, since the expected behaviour (create a new resource) is not compatible with addressing a single, existing resource.

Utilizing existing methods and conforming to expected actions given a method is designed to make an API easily understandable, and support knowledge transfer across different APIs.

5.2.4.4 Serialization

The Rodan API currently supports resource serialization using the JSON format. All requests to, and responses from the server must be formatted as JSON-serialized messages. A JSON object is a method of serializing and exchanging structured data between a client and a server, and is so named because it corresponds to the format in which JavaScript objects can be represented. In Rodan, model objects stored on the server side (Workflow, Job, Project, etc.) are serialized as JSON and sent between client and server. Clients that request a list of projects from a Rodan server send a GET request to the URI identifying the project collection (`http://example.com/projects/`) with `Accept: application/json` as a header in the request object. The server responds with the list of projects serialized as a JSON object, which may be parsed in a JavaScript environment (i.e., a browser).

5.2.4.5 Authentication and Authorization

Two types of authentication are available in Rodan. Session authentication allows a user to provide log-in credentials to the server. If the user credentials match an existing user account (i.e., a user successfully logs in), the server responds with a shared secret in the form of a "cookie," a small randomly generated and encrypted identifier. This cookie is automatically included with each request by a user's browser. Before fulfilling a request, the server verifies that the cookie value matches the value it has stored. If it matches, it permits the request to proceed; if it does not match the server assumes the client is not authorized and refuses the request, returning an error to the client.

Token authentication is an alternative to session authentication. In this method, every user is assigned a randomly-generated token, stored on the server. A user can request their token from the server and, if the username and password match, the server will respond with their unique token. This token may then be stored on the client side, negating any need for future authentication requests or cookie management system. The client can then include this token in each HTTP request in the `Authorization` header, e.g.,

```
"Authorization: Token 655aff7dc865866fc9bd9e7fafb32bfeb"
```

This token uniquely identifies the user operating a client, and provides an indication that this user has been duly authenticated. The server uses this token to manage the actions permitted to that user. This is provided as an alternative to Session authentication since it does not rely on a cookie-based session identification. Cookies can be cumbersome to maintain in a non-browser environment, so Token authentication may be used in clients that are not browsers, such as third-party software interacting with a remote Rodan instance.

5.2.4.6 Status and Error Codes

Status and error notifications are an important component in communicating the state of the server system, and provide indications of error conditions within the system as well as available means of correcting these errors. The HTTP specification defines a number of status codes which can be used by a server to communicate the status of a request. The full list of HTTP status codes is given in the HTTP protocol (Fielding et al. 1999). Status code categories are identified by their leading digit. The categories of status codes are given in table 5.1.

`1XX`	`Informational`
`2XX`	`Successful`
`3XX`	`Redirection`
`4XX`	`Client Error`
`5XX`	`Server Error`

Table 5.1: Categories of HTTP status codes

Every HTTP response contains a status code to indicate the status of the request. The most common status code is `200 OK`, which identifies that the request has been successfully fulfilled by the server; however, this is typically not seen by most users. The `404 Not Found` status code is perhaps the most visible in everyday use. This error is sent by a server in response to a request for a resource that is not present, either because the request was malformed (a mistyped URL, for example) or because the resource has been moved to a different location.

In Rodan, the standard HTTP status codes are used, rather than a custom status system. Among other things, this provides a level of compatibility with existing HTTP clients such as web browsers. A table listing some of the status codes used in Rodan is given in table 5.2.

200 OK	A request for a resource was successful
201 Created	A POST request resulted in the creation of a new resource
400 Bad Request	A client has sent a malformed request (i.e., a request that is missing required information)
401 Unauthorized	A client has attempted to perform an action without authorization
403 Forbidden	A client has attempted to perform an action for which it does not have permission
404 Not Found	A client has requested a resource that does not exist
405 Method Not Allowed	A client has used an HTTP Request Method that it is not allowed to use (e.g., a POST request by a non-authenticated user)
500 Server Error	The server has encountered an error and cannot continue

Table 5.2: Supported Rodan status codes

A client that accepts and understands these codes should be able to recover from them, or pass them along to the user for further diagnosis. For example, a `401 Unauthorized` error can prompt a software client to present an opportunity for the user to authenticate and authorize ("log in") by redirecting them to a log-in page.

5.2.4.7 API Summary

The Rodan API provides a mechanism for external clients to interact with a remote Rodan server for all activities related to creating and managing the data held on this server. This section provided an overview of the design patterns governing this API, primarily expressed through a RESTful design and utilizing the underlying mechanisms of the HTTP protocol. The complete Rodan API documentation is provided in Appendix B.

5.2.5 Interactive Jobs and Interactive Interfaces

Interactivity within a running workflow allows humans to engage with tasks, providing decisions that may improve the accuracy of the recognition process in a way that cannot be achieved by a fully automated process. Rodan implements interactivity in the workflow by specifying that a job can pause and wait for a human to supply it with further input before it continues to execute subsequent jobs in a workflow.

In Rodan, several JavaScript-based user interfaces have been developed to aid the user in accomplishing these tasks using their web browser.[10] These interactive jobs use the HTML `canvas` element to draw and manipulate the images at the pixel level within the browser. For each of these tasks the image in the browser serves as a proxy for the image on the server. The interface is used to determine the values (e.g., a binarization threshold or a rotation angle) which are then sent to an identical processing algorithm on the server. This reduces the amount of image data required for processing in the web browser, and allows the values, as determined by the user, to be applied to the full quality images.

10. The tools discussed in this section are integrated as part of the Rodan interactive workflow system, but are also developed as standalone components in the JS Image Suite package. Source code for these implementations are available at https://github.com/DDMAL/js-image-suite.

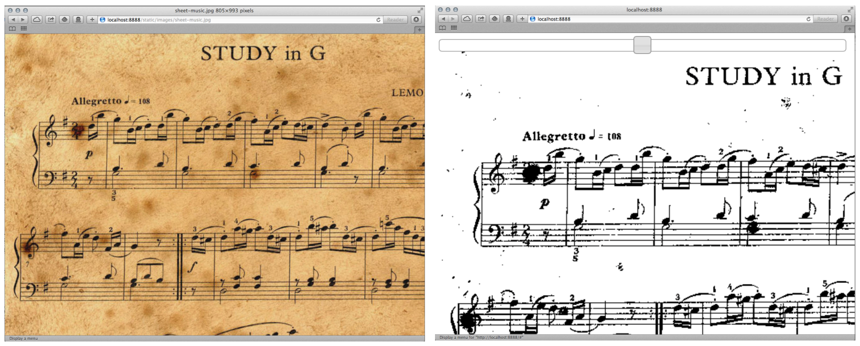


Figure 5.14: Interactive JavaScript Binarization interface. Original image on the left; binarised image on the right. The user can adjust the level of binarization interactively by moving the slider seen above the image on the right.

For example, in the binarization interface we have implemented a simple thresholding algorithm in JavaScript, where the value from the slider is used as the threshold value (figure 5.14). The goal of this task is to interactively determine the optimal global threshold of the image in response to the movement of the slider, rather than assuming one value for all pages in a workflow. The value is sent back to the server, which then applies the user-determined threshold value to an image.

A different interface provides an interactive way of separating musical staves from other page content, including lyrics, text, and noise (figure 5.15). A server-side process for determining the staff locations is executed on an image on the server side. This results in a polygon representing the locations of the musical staves as determined by a staff finding algorithm. These polygons are then used by the correction interface to draw an interactive polygon shape over those regions of the image. The user may use the mouse to adjust the size and shape of these polygons, with the goal of masking all the areas of musical content, while leaving areas of non-musical content (e.g., lyrics, ornate letters) un-masked. When the user submits a solution, the interface sends a corresponding list of polygon shapes back to the server, which then applies these polygons as a mask to remove areas of the image, leaving only the musical content.

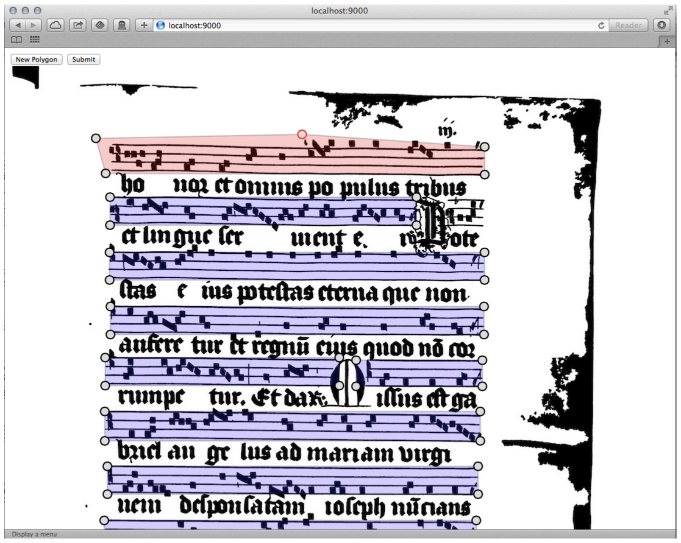


Figure 5.15: Interactive JavaScript staff segmentation interface. The blue boxes represent automatically-detected staves. The user is asked to correct this to ensure all the musical content can be segmented from other content (e.g., lyrics, decorations, noise, etc.).

Other interactive interfaces, such as image rotation, cropping, despeckling, and lyric region identification, have been implemented or are in the process of being implemented.

5.2.6 Design Summary

This section examined some of the design considerations underlying the development of the Rodan server. The Model-View-Controller pattern represents a structural separation of functionality among different components in the application. Clients interact with the Rodan Server over an HTTP API designed according to the principles of Representational State Transfer (REST). Tasks in Rodan workflows may include human interaction, and JavaScript correction interfaces allow users to participate in the recognition process by providing feedback within a running workflow.

The next sections will examine several implementation details of the Rodan Server and Rodan Client, including development platforms and core dependencies.

5.3 The Rodan Server

The Rodan Server is implemented using the Python programming language, and built using the open-source Django web application framework (Django Project 2014). Django is implemented using Python, and contains several tools to manage interactions between HTTP clients, database systems, and distributed processing queues.

5.3.1 The Rodan Job System

The core function of Rodan's Job system is to allow heterogeneous document processing and OMR tools to interact within a single workflow. These tools may take the form of a command-line application, or a Python module, or a remote web service.

For each tool, a task definition defines how the job should be executed, what sorts of input the tool accepts and the type of output the tool produces. This task definition is written in Python, and forms the bridge between the tool and the Rodan server.

In Rodan, the execution environment for tools is provided by the "Celery" module, a Python implementation of the RabbitMQ Message system, which will be discussed next.

5.3.2 RabbitMQ and Celery

Rodan uses the RabbitMQ (Videla and Williams 2012) message broker system for managing task execution. Celery (Lunacek et al. 2013) is a Python module that provides the language-specific tools for interacting with RabbitMQ. These systems provide Rodan with a distributed task queue, allowing image processing tasks to be "queued up" and executed in turn. These systems are designed to operate on a single computer system, or to distribute tasks across a number of "workers" on networked computer systems.

Rodan is designed to handle large, high-quality document images, and perform computationally-intensive tasks on them. As such, a distributed task queue is necessary to separate the components users interact with from the task execution system. When a message is received by the server to perform a task, this task is queued up for processing, and a status message is immediately sent back to the user notifying them that their job has been queued, with a URL that can be periodically queried to retrieve the status of the job.

An interactive job is treated as a special case in the workflow execution process. The presence of an interactive job effectively pauses the execution of a workflow until a human can provide the data needed to continue that task. In the queue system, an interactive task will be assigned to a worker for execution. If the system has not received human input (i.e., it has the status `WAITING_FOR_INPUT` or `RUN_ONCE_WAITING`) it will be marked for later execution and placed back into the queue. Should input be provided between execution attempts, the data is saved in the RunJob model, and the status of the RunJob is changed to `NOT_RUNNING`. The next time this job comes up for execution it will be executed with the provided input and the workflow will continue until it finishes or the next interactive job is encountered.

5.3.3 Server Summary

The Rodan server provides the ability to create, execute, and store workflows. With the Rodan server, many different toolkits can be integrated to form workflows.

However, the Rodan server does not have a human-friendly interface. All aspects of the Rodan server are controlled through its HTTP API, which is designed for computer-mediated interaction. The next section will describe the Rodan client, a browser-based user interface for the Rodan server.

5.4 The Rodan Client

The Rodan client is the reference implementation of functionality available from the Rodan server. While it is dependent on an instance of the Rodan server to operate (i.e., it has no workflow management or image processing systems itself), it is developed as a separate, standalone application. The Rodan client runs in most modern web browsers. A single instance of a Rodan server may interact with many client instances operating in many users' browsers.

5.4.1 Objective-J and Cappuccino

The Rodan client application is written using a dialect of the JavaScript language called Objective-J (Cappuccino Project 2014a). This dialect draws its inspiration from Objective-C, a programming language most commonly used in development for Apple devices (Macintosh and iOS). Objective-C adds object-oriented programming patterns to the C language, while also being a strict superset of the C programming language, meaning that any valid C code is also valid Objective-C code. In the same way, Objective-J is a strict superset of JavaScript, but adds several object-oriented features to the language. Since web browsers do not understand the Objective-J dialect a specialized compiler is used to translate code written in Objective-J to standard JavaScript.

The Cappuccino Framework (Cappuccino Project 2014b) utilizes the Objective-J language to provide a number of tools and "widgets"—buttons, form controls, windows—for building browser-based applications. Cappuccino mimics the Cocoa Framework (Hillegass and Preble 2011), Apple's API for developing on Mac OS X. The Cappuccino framework includes methods for creating and interacting with controls (buttons, sliders), information display (tables, progress bars) and AJAX communication.

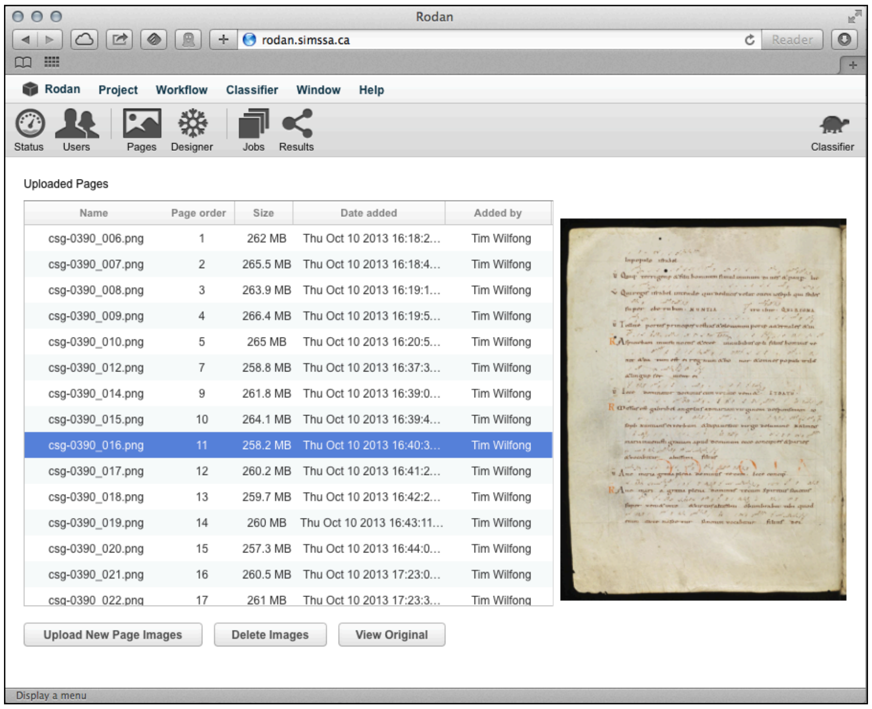
One of the advantages of writing the client application using Cappuccino is that this framework abstracts most of the underlying web technologies into a single programming environment. Most web applications require developers to write the interface using a combination of HTML and Cascading Style Sheets (CSS) for the visual elements, and JavaScript to define the behaviour of the application. With Cappuccino, however, these browser technologies are abstracted from the developer, and a single language environment, Objective-J, is used for all aspects of application development, including rendering a button, managing application state, or interacting with a remote server.

5.4.2 Application User Interface

Although the Rodan client application runs in a web browser, its visual design mimics that of a desktop application (figure 5.16). This is a feature of the Cappuccino framework, designed to emulate a desktop application built using the Cocoa frameworks. It was specifically chosen for Rodan to provide a visual bridge between traditional desktop applications while still operating in a user's browser.

5.4.2.1 Project Managment

The project management view (figure 5.17) is the first screen users see after they have successfully authenticated in the client against a Rodan server. This screen allows the user to choose or delete an existing project, or create a new project. Once they have chosen a project, they can click the "Open" button to proceed to the main Rodan application view (figure 5.16).



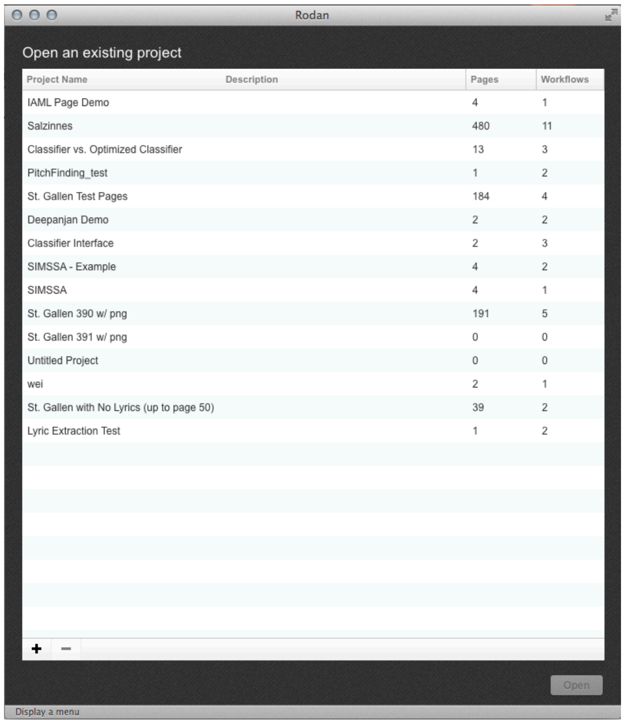
Uploaded Pages

Name	Page order	Size	Date added	Added by
csg-0390_006.png	1	262 MB	Thu Oct 10 2013 16:18:2...	Tim Wilfong
csg-0390_007.png	2	265.5 MB	Thu Oct 10 2013 16:18:4...	Tim Wilfong
csg-0390_008.png	3	263.9 MB	Thu Oct 10 2013 16:19:1...	Tim Wilfong
csg-0390_009.png	4	266.4 MB	Thu Oct 10 2013 16:19:5...	Tim Wilfong
csg-0390_010.png	5	265 MB	Thu Oct 10 2013 16:20:5...	Tim Wilfong
csg-0390_012.png	7	258.8 MB	Thu Oct 10 2013 16:37:3...	Tim Wilfong
csg-0390_014.png	9	261.8 MB	Thu Oct 10 2013 16:39:0...	Tim Wilfong
csg-0390_015.png	10	264.1 MB	Thu Oct 10 2013 16:39:4...	Tim Wilfong
csg-0390_016.png	11	258.2 MB	Thu Oct 10 2013 16:40:3...	Tim Wilfong
csg-0390_017.png	12	260.2 MB	Thu Oct 10 2013 16:41:2...	Tim Wilfong
csg-0390_018.png	13	259.7 MB	Thu Oct 10 2013 16:42:2...	Tim Wilfong
csg-0390_019.png	14	260 MB	Thu Oct 10 2013 16:43:11...	Tim Wilfong
csg-0390_020.png	15	257.3 MB	Thu Oct 10 2013 16:44:0...	Tim Wilfong
csg-0390_021.png	16	260.5 MB	Thu Oct 10 2013 17:23:0...	Tim Wilfong
csg-0390_022.png	17	261 MB	Thu Oct 10 2013 17:23:3...	Tim Wilfong

Figure 5.16: The Rodan Client interface

5.4.2.2 Main Application View

The main application view is the primary view of the Rodan client (shown with the "Pages" sub-view in figure 5.16). It is divided into separate screens that segment information views and interfaces in the client. The menu bar and toolbar sections at the top of the application (figure 5.18) are always available and allow the user to navigate between the different screens of the application. These screens will be discussed in turn.



Open an existing project

Project Name	Description	Pages	Workflows
IAML Page Demo		4	1
Salzinnes		480	11
Classifier vs. Optimized Classifier		13	3
PitchFinding_test		1	2
St. Gallen Test Pages		184	4
Deepanjan Demo		2	2
Classifier Interface		2	3
SIMSSA - Example		4	2
SIMSSA		4	1
St. Gallen 390 w/ png		191	5
St. Gallen 391 w/ png		0	0
Untitled Project		0	0
wei		2	1
St. Gallen with No Lyrics (up to page 50)		39	2
Lyric Extraction Test		1	2

Figure 5.17: Project Management View

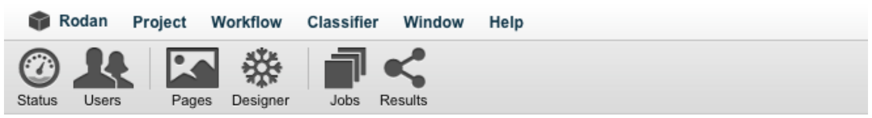


Figure 5.18: The Rodan Application Toolbar

5.4.2.3 Status

The status section, currently a placeholder and not implemented, offers users a visual overview of the Rodan server, the availability of the queues, and other status-related information relating to the state of the remote server.

5.4.2.4 Users

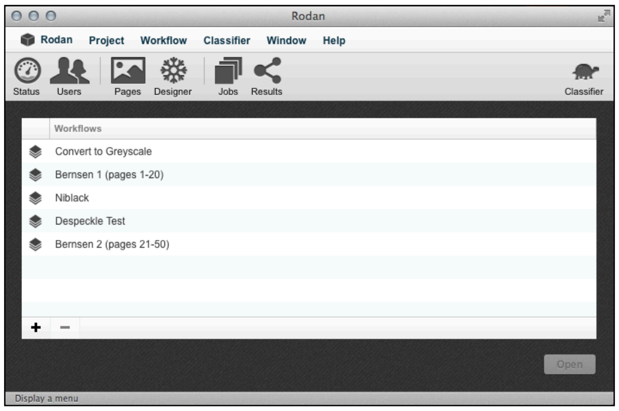
The user management pane, also not currently implemented, offers administrators the ability to assign and manage users and their roles in a project.

5.4.2.5 Pages

The pages view (shown in figure 5.16) provides the user with an interface for uploading and viewing all pages, and page images, associated with a project. Users click the "Upload new page images" button and are shown a file browser, allowing them to choose multiple files on their local machine to upload to the server. The list of uploaded page images are displayed in the central table, and a thumbnail representation of a selected page is displayed to the right of the table.

5.4.2.6 Designer

The designer view displays management controls for working with workflows associated with a project. When clicking on the designer toolbar icon, they are taken to a workflow management view (figure 5.19) where they can create a new workflow, or edit or delete existing workflows.

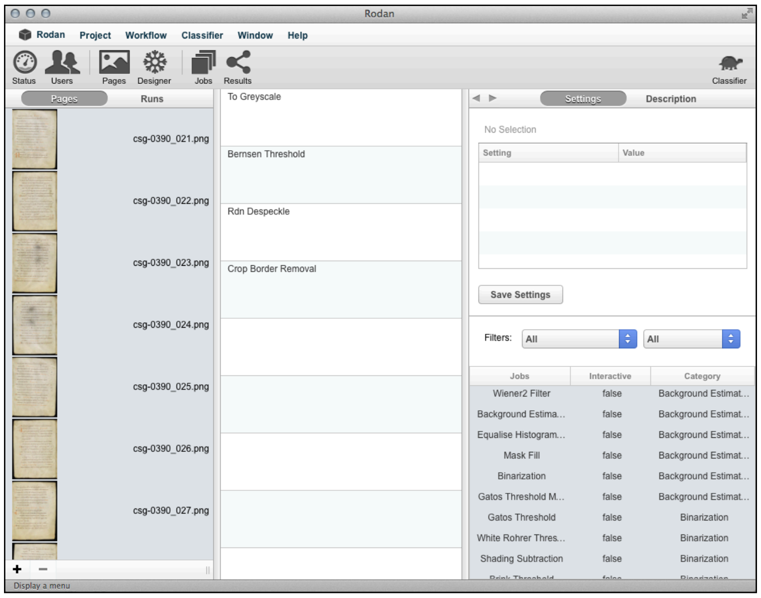


Workflows
Convert to Greyscale
Bernsen 1 (pages 1-20)
Niblack
Despeckle Test
Bernsen 2 (pages 21-50)

Figure 5.19: Workflow management view

To open a workflow, users select a workflow from the list and click the "Open" button. The user then sees the main workflow designer view (figure 5.20). This view allows the user to manage the pages (left) and runs (left, tab not shown), tasks (centre), and job settings (right, above), and view the library of jobs available (right, below).

Users add previously-uploaded pages to the workflow by clicking the "+" button below the page area, or choosing the "Workflow" menu and selecting "Add Pages…". They may drag and drop jobs from the job library into the central workflow designer area. Users can filter the available jobs list by category or alphabetically. Each job indicates whether it is an interactive or non-interactive job in the list of available jobs. Jobs placed in the central workflow area are executed in order, from top to bottom. If a user attempts to drag and drop a job that cannot follow another, the system will not allow it. Clicking on a job that is placed on the workflow area shows the settings available for that job in the upper-right region.



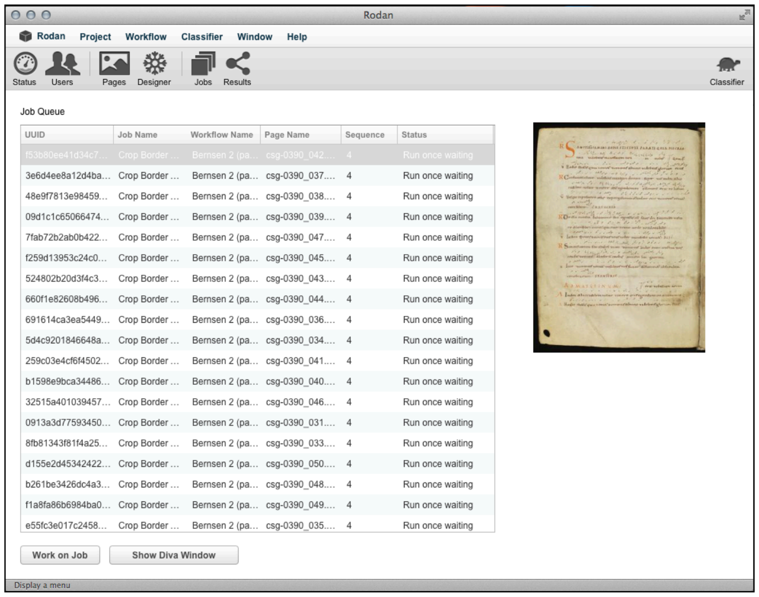
csg-0390_021.png
csg-0390_022.png
csg-0390_023.png
csg-0390_024.png
csg-0390_025.png
csg-0390_026.png
csg-0390_027.png

To Greyscale
Bernsen Threshold
Rdn Despeckle
Crop Border Removal

Setting	Value

Jobs	Interactive	Category
Wiener2 Filter	false	Background Estimat…
Background Estima…	false	Background Estimat…
Equalise Histogram…	false	Background Estimat…
Mask Fill	false	Background Estimat…
Binarization	false	Background Estimat…
Gatos Threshold M…	false	Background Estimat…
Gatos Threshold	false	Binarization
White Rohrer Thres…	false	Binarization
Shading Subtraction	false	Binarization
Brink Threshold	false	Binarization

Figure 5.20: Workflow designer view



UUID	Job Name	Workflow Name	Page Name	Sequence	Status
f53b80ee41d34c7...	Crop Border ...	Bernsen 2 (pa...	csg-0390_042....	4	Run once waiting
3e6d4ee8a12d4ba...	Crop Border ...	Bernsen 2 (pa...	csg-0390_037....	4	Run once waiting
48e9f7813e98459...	Crop Border ...	Bernsen 2 (pa...	csg-0390_038....	4	Run once waiting
09d1c1c65066474...	Crop Border ...	Bernsen 2 (pa...	csg-0390_039....	4	Run once waiting
7fab72b2ab0b422...	Crop Border ...	Bernsen 2 (pa...	csg-0390_047....	4	Run once waiting
f259d13953c24c0...	Crop Border ...	Bernsen 2 (pa...	csg-0390_045....	4	Run once waiting
524802b20d3f4c3...	Crop Border ...	Bernsen 2 (pa...	csg-0390_043....	4	Run once waiting
660f1e82608b496...	Crop Border ...	Bernsen 2 (pa...	csg-0390_044....	4	Run once waiting
691614ca3ea5449...	Crop Border ...	Bernsen 2 (pa...	csg-0390_036....	4	Run once waiting
5d4c9201846648a...	Crop Border ...	Bernsen 2 (pa...	csg-0390_034....	4	Run once waiting
259c03e4cf6f4502...	Crop Border ...	Bernsen 2 (pa...	csg-0390_041....	4	Run once waiting
b1598e9bca34486...	Crop Border ...	Bernsen 2 (pa...	csg-0390_040....	4	Run once waiting
32515a401039457...	Crop Border ...	Bernsen 2 (pa...	csg-0390_046....	4	Run once waiting
0913a3d77593450...	Crop Border ...	Bernsen 2 (pa...	csg-0390_031....	4	Run once waiting
8fb81343f81f4a25...	Crop Border ...	Bernsen 2 (pa...	csg-0390_033....	4	Run once waiting
d155e2d45342422...	Crop Border ...	Bernsen 2 (pa...	csg-0390_050....	4	Run once waiting
b261be3426dc4a3...	Crop Border ...	Bernsen 2 (pa...	csg-0390_048....	4	Run once waiting
f1a8fa86b6984ba0...	Crop Border ...	Bernsen 2 (pa...	csg-0390_049....	4	Run once waiting
e55fc3e017c2458...	Crop Border ...	Bernsen 2 (pa...	csg-0390_035....	4	Run once waiting

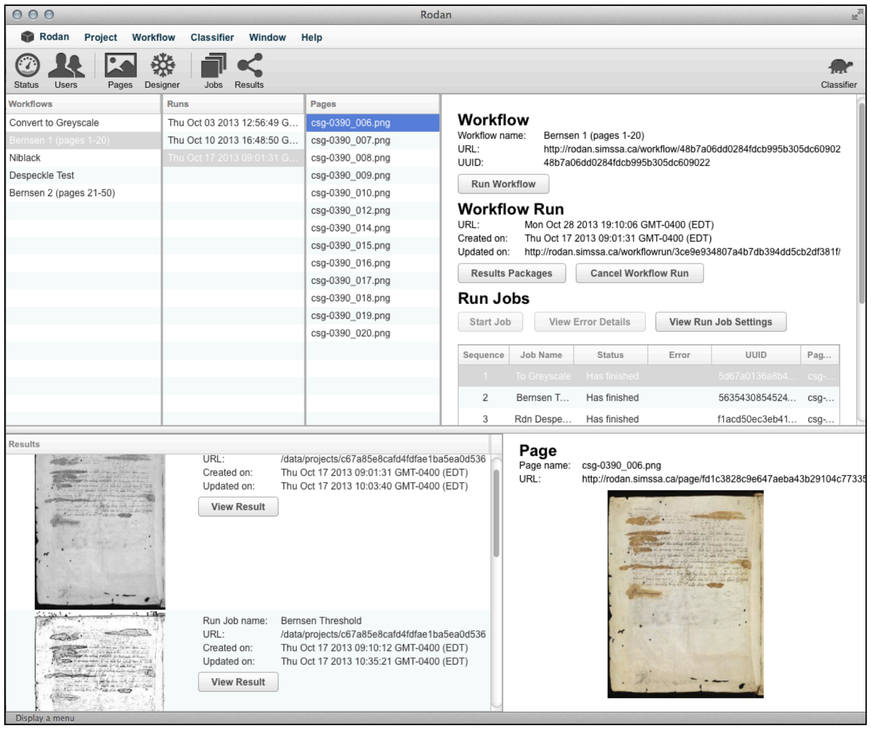
Figure 5.21: Jobs view

5.4.2.7 Jobs

The jobs view (figure 5.21) displays the status of every job in a workflow that has been executed or is being executed. The interface also provides a central place to view all interactive jobs that are waiting for human intervention. Users can select a job and click the "Work on Job" button, which will then display an interface (similar to those shown in figures 5.14 and ?) for performing this task.

5.4.2.8 Results

The results view (figure 5.22) is the central control panel for viewing and managing workflows that have started, are currently running, or have finished.



Workflows	Runs	Pages
Convert to Greyscale	Thu Oct 03 2013 12:56:49 G…	csg-0390_006.png
Bernsen 1 (pages 1-20)	Thu Oct 10 2013 16:48:50 G…	csg-0390_007.png
Niblack	Thu Oct 17 2013 09:01:31 G…	csg-0390_008.png
Despeckle Test		csg-0390_009.png
Bernsen 2 (pages 21-50)		csg-0390_010.png
		csg-0390_012.png
		csg-0390_014.png
		csg-0390_015.png
		csg-0390_016.png
		csg-0390_017.png
		csg-0390_018.png
		csg-0390_019.png
		csg-0390_020.png

Workflow

Workflow name:	Bernsen 1 (pages 1-20)
URL:	http://rodan.simssa.ca/workflow/48b7a06dd0284fdcb995b305dc60902
UUID:	48b7a06dd0284fdcb995b305dc609022

Workflow Run

URL:	Mon Oct 28 2013 19:10:06 GMT-0400 (EDT)
Created on:	Thu Oct 17 2013 09:01:31 GMT-0400 (EDT)
Updated on:	http://rodan.simssa.ca/workflowrun/3ce9e934807a4b7db394dd5cb2df381f/

Run Jobs

Sequence	Job Name	Status	Error	UUID	Pag…
1	To Greyscale	Has finished		5d67a0136a8b4…	csg-…
2	Bernsen T…	Has finished		5635430854524…	csg-…
3	Rdn Despe…	Has finished		f1acd50ec3eb41…	csg-…

Results

URL:	/data/projects/c67a85e8cafd4fdfae1ba5ea0d536
Created on:	Thu Oct 17 2013 09:01:31 GMT-0400 (EDT)
Updated on:	Thu Oct 17 2013 10:03:40 GMT-0400 (EDT)

Run Job name:	Bernsen Threshold
URL:	/data/projects/c67a85e8cafd4fdfae1ba5ea0d536
Created on:	Thu Oct 17 2013 09:10:12 GMT-0400 (EDT)
Updated on:	Thu Oct 17 2013 10:35:21 GMT-0400 (EDT)

Page

Page name:	csg-0390_006.png
URL:	http://rodan.simssa.ca/page/fd1c3828c9e647aeba43b29104c77335

Figure 5.22: Results view

Project workflows are shown in the upper-left corner. Selecting a workflow will display a list of workflow runs in the upper-left centre column. Selecting a run will display the pages associated with that run in the right column. Selecting a page from the list will display further information about that page in the bottom half of the view. The results of all jobs in the workflow are shown in the bottom left, while the bottom right displays a thumbnail image of the original page image.

The upper-right quadrant displays information and controls for the currently selected workflow and workflow run. Choosing a page will display a list of jobs, and their status, associated with that page. Users also have controls for performing any interactive tasks associated with a given page image.

5.4.3 Client Summary

The current user interface for Rodan is a preliminary attempt at building a control surface for a remote Rodan server instance Using the Rodan client application, many users can log in and interact with a remote Rodan server instance, working collaboratively to build and run workflows, interact with running workflow tasks, and administer the remote Rodan server instance all through a standard web browser.

5.5 Chapter Summary

Rodan is a novel prototype application for building workflow-based OMR systems, utilizing aspects of web application programming and distributed computing. The core function of Rodan is to serve as a bridge between large, mixed collections of independent image processing, machine learning, and symbolic music processing toolkits, creating an environment where they can interoperate in bespoke OMR processes, managed and executed in a web-based environment.

The design of Rodan was inspired by the manual process and procedures encountered in the OMR workflows for the *Liber usualis* project. This led to preliminary attempts at building an OMR system using Taverna, a scientific workflow system. In these investigations we discovered a need to integrate human feedback into the overall process, a feature that Taverna did not support at the time. This led to the creation of Rodan, a workflow system that can support interactive workflows and execution of tasks based on human input.

Rodan is developed as two separate projects, a server and a client. The Rodan server system uses the Django web application framework, an MVC-based environment for developing customized applications using the Python programming language. Interaction with the Rodan server uses a well-defined API based on the principles of REST, using built-in methods and notification systems of the HTTP protocol. Rodan is designed to bring together existing image processing, machine learning, and symbolic music processing tools, providing an environment where these tools can interoperate as a components in an OMR workflow.

The Rodan client application is a graphical user interface that provides a way for people to interact with the Rodan server API in a web browser. The Rodan client is built using the Objective-J language and the Cappuccino Framework, abstracting many of the technologies underlying web application development (HTML, CSS, the DOM) and integrating them into a single development environment.

With the Rodan client application users can organize recognition projects, upload images, and build and manage custom document recognition workflows. These workflows can then be executed on a remote server, with built-in mechanisms for utilizing distributed and parallel computing. The ability to run several workflows in parallel across many computer systems creates the opportunity for scaling the capacity of an OMR system to accommodate larger workloads than may be executed on a single personal computer. While the networked adaptive OMR functionality has not been developed yet, Rodan will provide an ideal environment for building and experimenting with this technique.

While Rodan is the central prototype developed as part of this dissertation, several other tools for supporting large-scale, web-based recognition, search, and retrieval of music documents have been developed in the course of this research program. The next chapter will examine these tools.

6. Technologies supporting large-scale recognition

The previous chapter presented Rodan, a platform for creating distributed OMR (Optical Music Recognition) workflows. This chapter will present several tools and technologies developed to support large-scale OMR projects.

Each section of the chapter describes a different tool or process developed in the course of the dissertation. The first section (§6.1) describes a software library, LibMEI, to read, write, and manipulate MEI files. Diva (§6.2) is a document image viewer that uses several novel techniques for displaying large, high-resolution images in a web browser. The Solesmes MEI neume notation customization (§6.3) was developed as way to encode the notation used in the *Liber usualis* project (described in chapter 7). The OMR Interchange Package (OIP) is presented as a package file format for sharing and using the image and notation results of an OMR system (§6.4). Finally, two prototypes for web-based crowdsourced correction tools are presented (§6.5).

Several of the projects mentioned in this chapter were the result of collaboration with others, who will be mentioned in this chapter where appropriate. The acknowledgements of the dissertation provides a detailed description of individual contributions.

6.1 LibMEI

LibMEI is an object-oriented, open-source C++ library for reading and writing MEI-encoded files. It was built to provide an interface between "raw" MEI XML and applications that support reading and writing MEI files. It has been tested on Linux and Macintosh operating systems. The source code and documentation are publicly available on GitHub (Hankinson and Porter 2014). Alastair Porter was co-designer of this library, and helped formulate many of the core functions of this software.

An initial version of LibMEI was built using the Python language. This version was used to prototype features and behaviours of the library. The Python prototype was then ported to C++ to help improve performance and to provide greater compatibility with other software applications. Python compatibility with the C++ version of LibMEI is handled using the Boost Python library (Abrahams and Grosse-Kunstleve 2003) to create "bindings" between the two language environments.

While MEI is most often expressed as XML, it may also be expressed in other structured representation formats; for example, JavaScript Object Notation (JSON) (ECMA International 2013). Similarly, LibMEI is designed to be "expression-agnostic." LibMEI translates an XML encoding into its own internal object hierarchy. This abstraction allows the user to work with the MEI structure using methods and data structures native to the language (e.g., C++, Python, or JavaScript), rather than relying on XML-centric parsing libraries. This makes LibMEI suitable for use in, for example, a web application where a server reads in an XML file but produces a JSON representation that can be sent to a browser and manipulated by JavaScript methods.

6.1.1 LibMEI Core

The core of LibMEI provides basic functionality for reading, parsing, writing, and traversing MEI structure. Since LibMEI is expression-agnostic, XML-specific functionality is limited to reading and writing; all other functions for traversing and manipulating the underlying MEI operate on an internal data structure. Expression of MEI in any other document serialization format (e.g., JSON) would operate in a similar way, where the form is parsed into the internal format.

There are four "core" object classes in LibMEI:

- `MeiElement`
- `MeiDocument`
- `MeiAttribute`
- `MeiNamespace`

In addition, there are two object classes for reading and writing XML:

- `XmlImport`
- `XmlExport`

Figure 6.1 illustrates the relationships between these objects. The import and export functionality (left of the dotted line) is separate from the MEI object structure. A hypothetical JSON import and export is shown here to indicate that LibMEI can support multiple representations. Importing and exporting are handled via the `MeiDocument` class, which acts as a container object and provides some global document functionality, such as searching and traversing all objects in the document.

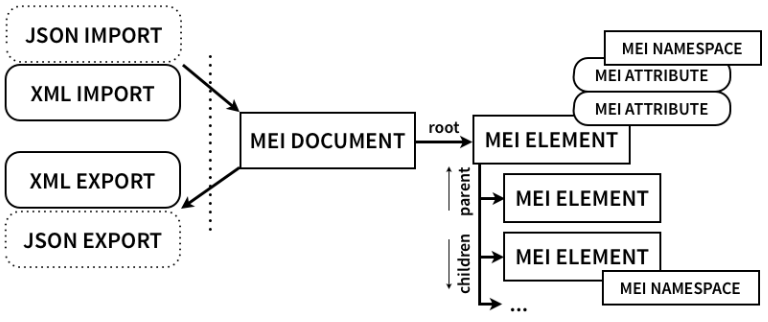


Figure 6.1: LibMEI Object Structure

In LibMEI, relationships between objects are maintained by pointers. A pointer is a programming language construct that allows a single object to be referenced in many places without requiring multiple copies of the object. (Technically, a pointer references the location in a computer's memory storage system where an instance of the object is stored). When an MEI file is parsed, `MeiElement` objects are created in memory for each element. These objects are related to each other by maintaining pointers between elements, allowing a single object to be referenced in two places (e.g., as the parent object of one element, and the child of another).

The `MeiDocument` object holds a pointer reference to a single `MeiElement` object as the root of the document. Each `MeiElement` contains an ordered array of pointers to child objects, as well as a pointer to its parent `MeiElement` object. Each `MeiElement` also holds an array of attributes that act as simple containers for storing keys and values for each attribute of an object. A pointer to a `MeiNamespace` object provides functionality for handling objects from different XML namespaces (Bray et al. 2006a), and applies to both `MeiAttribute` and `MeiElement` objects.

Finally, the `XmlImport` and `XmlExport` objects are classes built to convert XML into `MeiDocument` and `MeiElement` objects, and vice-versa. XML importing is handled via `libxml2`, a third-party library for handling XML files (Veillard 2014).

When an MEI file is loaded into LibMEI, several different representations of the structure are generated to assist in parsing and traversing the document. A tree representation corresponding to the hierarchical structure is generated through parent-child relationships on each element. An ordered 1-dimensional representation, created as a simple array of pointers, is also created. This array is a flattened list containing pointers to all the elements in the MEI file. This is an optimization for navigating the XML tree structure, transforming search operations from a recursive operation to a linear scanning operation. Since every element is a pointer containing other pointers to its parent and its children, the tree hierarchy may be entered and traversed using any element without any unnecessary duplication of structure (e.g., two copies of objects that represent the same element).

6.1.2 Library Generation and LibMEI Extended

LibMEI is distributed with a tool, `parseschema2.py`, to parse an ODD ("One Document Does-it-all") schema file and automatically generate a customized library for reading and writing MEI files. While this was initially conceived as a means of keeping LibMEI synchronized with developments in the MEI schema, it also has the benefit of providing a strongly-typed system for supporting customized versions of MEI. For example, it is possible to generate a C++ library specifically for use in mensural notation software which would fail to parse a MEI file that uses elements drawn from the Common Music Notation (CMN) module. This failure is designed as an explicit safety feature to ensure that a software system (e.g., a notation editor) does not try to parse an MEI file containing data that it is not designed to handle. The schema parsing script generates source code for classes corresponding to MEI elements defined in the ODD file.

Other systems, known as "bindings," exist for automatically parsing an XML schema to a native representation in a programming language. These bindings allow programmers to interact with objects defined in an XML schema as objects available in their programming language. For Java, JAXB (Project JAXB 2014) and Castor (Exolab Group 2013) provide automatic generation of bindings, while CodeSynthesis XSD (Code Synthesis Tools 2014) provides similar functionality for C++. Before writing our own, we examined these libraries. We chose to implement the core of our library in C++ to provide a wider range of supported platforms, but bindings generated by the CodeSynthesis XSD tool produced an overly complex system, so it was decided to build our own tool to generate the bindings directly from the ODD representation.

LibMEI is distributed with a library generated from the "MEI All" customization, a schema that includes all modules, elements, and attributes specified in MEI. This allows LibMEI to read and write MEI-encoded notation files that would pass validation by the MEI All RelaxNG schema.

The code generation functionality for LibMEI has been used through three major versions of the MEI schema: 2011-05, 2012, and 2013. In addition, it has been used to generate libraries for reading and writing customized schemas for the *Liber usualis* (§7.1) and *Salzinnes* projects (§7.2, §7.3). The next sections will describe how the elements, attributes, and auxiliary functionality are accommodated with the `parseschema2.py` script.

6.1.2.1 Typed Objects

The customization system automatically generates C++ code that defines classes for every element defined in MEI. This code can be compiled by a C++ compiler. These generated classes wrap and extend the functionality available in the LibMEI core by providing a "type system" for every MEI element; that is, the `<note />` element is automatically converted to the C++ code that defines a `Note` object. Every typed element inherits from the base `MeiElement` class.

To illustrate the differences between the "core" LibMEI and the extended, type-specific functionality, figure 6.2 shows two procedures for creating an MEI "note" element. The example on the top uses the LibMEI "core," creating a `MeiElement` object with the name "note." The example on the bottom uses objects auto-generated from the ODD file. This example shows that a `Note` object can be constructed simply by instantiating the corresponding class that was auto-generated from the MEI ODD file.

```
MeiElement *p = new MeiElement("note");
```

```
Note *p = new Note();
```

Figure 6.2: Comparison of LibMEI note object instantiation in LibMEI core (top) and extended (bottom)

LibMEI uses the extended, typed version of the library when reading in an XML file. This is accomplished using a mapping system that maps the name of an element with the particular class used to instantiate an object for that element. For each automatically-identified class definition, derived from the MEI ODD file, there are two pre-processing macros defined, `REGISTER_DECLARATION` and `REGISTER_DEFINITION`. At compile-time these macros register a string corresponding to the element name (e.g., "`note`" or "`meiHead`") and map it to the symbol of the compiled class definition, creating a class registry. When an XML file is imported at run-time, the import layer reads the element name, finds the corresponding class symbol in the class registry, and then instantiates that class to represent that element. If the tag name is not found, either because of an error in the MEI file or the particular customization of MEI does not contain support for that element, the library will raise an exception and will fail to import the file.

When the `parseschema2.py` script generates the library, it divides the source code into separate source files corresponding to the MEI module from which the code was derived. This includes creating both header and implementation files. The `Note` class, for example, can be found in the `shared.h` (header) and `shared.cpp` (implementation) files, since the `<note>` element is a member of the `MEI.shared` module.

6.1.2.2 Attribute Groups

Element definitions in MEI may belong to many attribute groups. Attribute groups are used to bring together similar attribute definitions so that they are defined in one place but may also be shared between elements. For example, both `<note>` and `<rest>` elements belong to the `att.duration` attribute class, but only the `<note>` element would belong to the classes that define attributes for pitch and octave value (i.e., `@pname` and `@oct`, respectively). Attribute groups are used to ensure consistent behaviour of attributes across all elements in the schema. The code generation script uses this information to consolidate attribute classes into similar classes that define methods for manipulating those attributes on that object. When the MEI schema is parsed each attribute class is added to the elements that belong to that class by using a "mix-in" system following the composition design pattern.

The composition design pattern (Freeman et al. 2004) is a method of achieving polymorphic behaviour in an object-oriented system without using multiple inheritance. The distinction between traditional multiple-inheritance polymorphism and the composition pattern may be expressed as the difference between *is-a* and *has-a*. A class that inherits from two superclasses in the traditional inheritance model may be described as having the attributes of both classes; thus, `ClassX` *is-a* composite of both `ClassA` and `ClassB`. The *has-a* pattern, used in composition, allows methods to be defined as members of multiple classes without needing to explicitly inherit from this class. So `ClassY`, implemented using composition, *has* the methods defined in `ClassA` and `ClassB` without inheriting from them.

The purpose of employing the composition pattern in LibMEI is to reduce the amount of repeated code for managing attributes on the element class objects. This is accomplished by *mix-ins*, or attribute class definitions that can be added to elements ("mixed in") as needed. As of this writing, the `Note` class has 51 separate mix-in classes, each containing definitions of one or more attribute classes. The mix-in pattern significantly reduces the amount of generated code by grouping and re-using these method definitions on multiple elements. In addition, pre-defined accessors for each attribute can reduce the amount of "boilerplate" code for working with attributes (figure 6.3).

```
MeiAttribute *a = new MeiAttribute("pname", "c");
p->addAttribute(a);
std::string *v = p->getAttribute("pname").getValue();
```

```
p->m_Pitch->setPname("c");
std::string *v = p->m_Pitch->getPname();
```

Figure 6.3: Generic (top) and mix-in (bottom) methods of accessing the @pname attribute.

A fragment of the implementation for the LibMEI `Note` class is given in figure 6.4. The mix-in class, `PitchMixIn`, (in `sharedmixins.h`) contains methods for working with an attribute defined in the schema for this class, `pname`. Four methods are defined in this mix-in for this attribute: `getPname`, `setPname`, `hasPname`, and `removePname`. In the `Note` class definition, the `PitchMixIn` class is set as member variable, `m_Pitch`. The class initialization list, shown in `shared.cpp` (also in figure 6.4), sets the `m_Pitch` member variable when the object is created. This places the instantiation of each mix-in class as part of the instantiation process of the `Note` object. For each mix-in, a pointer to the specific instance of the class ("`this`") is passed to the mix-in constructor. In the implementation of each mix-in function, `this` provides a way to access the member variables defined in the `Note` object, and is available in the mix-in class via the member variable `b` (for "base"). The mix-in function implementation for the `pname` attribute is shown in figure 6.5. The script examines an element's membership in attribute groups, and automatically builds a dedicated class used to hold accessors ("get," "set," "has," and "delete" methods) for supported attribute classes on that element.

```
// sharedmixins.h

class PitchMixIn {
    public:
        explicit PitchMixIn(MeiElement *b);
        virtual ~PitchMixIn();
        MeiAttribute* getPname();
        void setPname(std::string _pname);
        bool hasPname();
        void removePname();
    private:
        MeiElement *b;
};

// shared.h

class MEI_EXPORT Note : public MeiElement {
    public:
        Note();
        Note(const Note& other);
        virtual ~Note();
        CommonMixIn    m_Common;
        PitchMixIn    m_Pitch;
        ...
    private:
        REGISTER_DECLARATION(Note);
};

// shared.cpp

mei::Note::Note() :
    MeiElement("note"),
    m_Common(this),
    m_Pitch(this),
    ...;
{}
```

Figure 6.4: Sample Mix-In implementation in LibMEI

```
// sharedmixins.h

MeiAttribute* mei::PitchMixIn::getPname() {
    if (!b->hasAttribute("pname")) {
        throw AttributeNotFoundException("pname");
    }
    return b->getAttribute("pname");
};

void mei::PitchMixIn::setPname(std::string _pname) {
    MeiAttribute *a = new MeiAttribute("pname", _pname);
    b->addAttribute(a);
};

bool mei::PitchMixIn::hasPname() {
    return b->hasAttribute("pname");
};

void mei::PitchMixIn::removePname() {
    b->removeAttribute("pname");
};
```

Figure 6.5: Sample mix-in function implementation for pname

6.1.2.3 Custom methods

While auto-generation of library functions allows for easier code maintenance with newer revisions of the MEI schema, this comes with a significant drawback. Custom methods for performing tasks beyond simple accessor functions are difficult to integrate into the library, since all of the source code for each class definition is auto-generated and is therefore lost when the library is re-generated. In other words, anything not defined in the ODD schema or derived from getting or setting attribute values on objects is lost when the library is re-generated. To address this drawback, a method for injecting custom methods into the library generation process is provided by the `parseschema2.py` script. Custom methods are defined in an includes file, and are automatically added to the source code when it is re-generated. When the include function is run in the library generation process, the methods to be included are written into the appropriate place in the source code, which may then be compiled in to LibMEI as an available library function.

This process will be illustrated by way of an example of extending the auto-generated class definition of the `Tie` element. Suppose we want to write a custom method, `getMembers`, available on the `Tie` object that, when called, could return all `Note` objects ("members") attached to that `Tie`. Since this is custom functionality for the `Tie` object, we need to devise a way of injecting this method *after* the class code has been automatically generated from the ODD schema.

The `Tie` object is defined in the `cmn.h` and `cmn.cpp` files. Custom methods are stored in files within an `includes` directory, with each file named in a way that allows the parsing system to automatically discover them. For this example, the files we create would be `includes/cpp/cmn.cpp.inc` and `includes/cpp/cmn.h.inc`. Our custom code for our `getMembers` method would be placed within a specially-formatted comment block for both the header and implementation files. An example definition for injecting the appropriate code into the header file is shown in figure 6.6.

```
// cmn.h.inc

/* <tie> */
MeiElement* getSystem();
std::vector<mei::MeiElement*> getMembers();
/* </tie> */
```

Figure 6.6: Example include definition for the Tie element

When the MEI schema is parsed and the class definitions are created from the ODD file, a "bookmark," in the form of a specially-formatted comment, is injected into the appropriate place in the class definition. In this case, the comment would be `/* include <tie> */` (figure 6.7), injected into the file close to the `Tie` object definition. The includes-parsing stage of the `parseschema2.py` script examines the includes directory for any definitions to inject. If it finds a includes file, it examines this file for any custom methods. It then uses the specially-formatted comment blocks to inject these methods back into the source code by replacing the comment in the object definition with the custom method defined in the includes file.

```
class MEI_EXPORT Tie : public MeiElement {
    public:
        Tie();
        Tie(const Tie& other);
        virtual ~Tie();

/* include <tie> */

        CommonMixIn    m_Common;
```

Figure 6.7: Example definition for the Tie element, showing the comment where the custom methods will be injected.

The resulting `Tie` object, once compiled, would then contain the `getMembers()` method, allowing the developer to call `Tie->getMembers()` on the object. This method would then parse the information available on the `Tie` object, and, depending on the implementation, likely return an array of pointers to the `Note` objects that are members of the `Tie`.

6.1.3 SibMEI

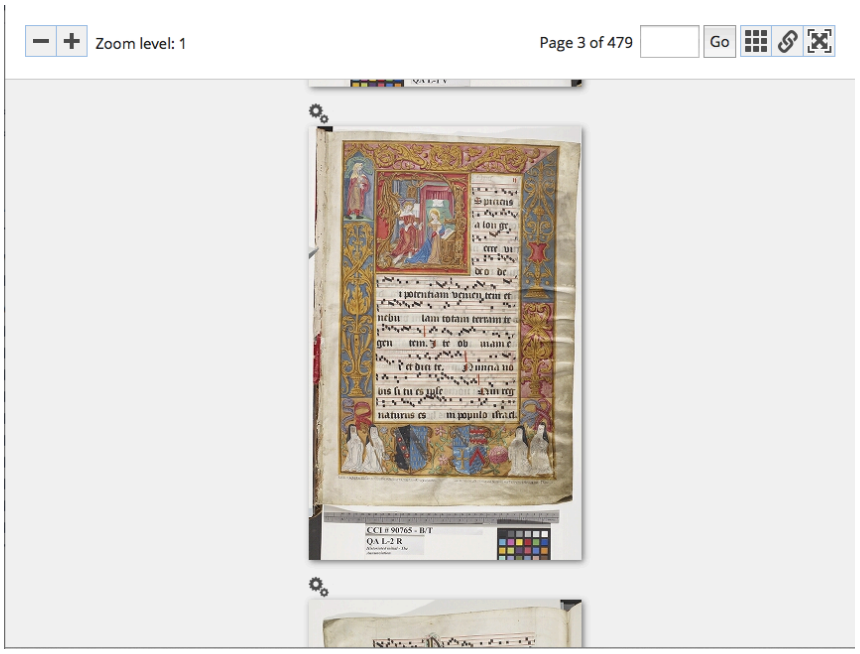
The SibMEI plug-in (Hankinson and Walter 2014) is open-source software that uses LibMEI idioms to bring MEI import and export functionality to the Sibelius notation editor. It is a separate project from LibMEI and does not use the C++ core of LibMEI, but it does use the schema parsing system to automatically generate a Sibelius plug-in that contains functions for reading and writing MEI files.

One of the first challenges when bringing MEI functionality to Sibelius was the lack of an XML library for reading and writing MEI within the Sibelius development environment. Developers write Sibelius plug-ins in a dedicated language called ManuScript (Finn et al. 2011). This environment supports reading and writing plain-text files (including Unicode) but has no functionality for parsing or writing XML files. To allow SibMEI to import MEI files, an XML parser was ported to ManuScript from an existing Java implementation (Brandt 2002). For export, an original XML exporter was written.

At the time of writing, MEI export from Sibelius is fully functional. For import, an MEI file can be opened and parsed into an internal representation, but actually drawing the notation in Sibelius is not currently implemented and is slated for future work.

6.2 Diva: Document Image Viewer

Diva (Document Image Viewer with Ajax) is a book image viewer suitable for displaying large, high-quality book images in a web browser without needing to install special plug-ins (Hankinson et al. 2009; Hankinson et al. 2011a; Hankinson et al. 2012). With Diva it is possible to view smaller, low-quality images (suitable for fast browsing and document navigation) or larger, high-resolution pages (suitable for close study) in the same interface, while maintaining the ability to scroll through all the images in the document, similar to scrolling through a multi-page word processing or PDF document (figure 6.8). The central feature of the Diva viewer, and one which is unique among all known document image viewers, is that it combines both of these features in a single interface.



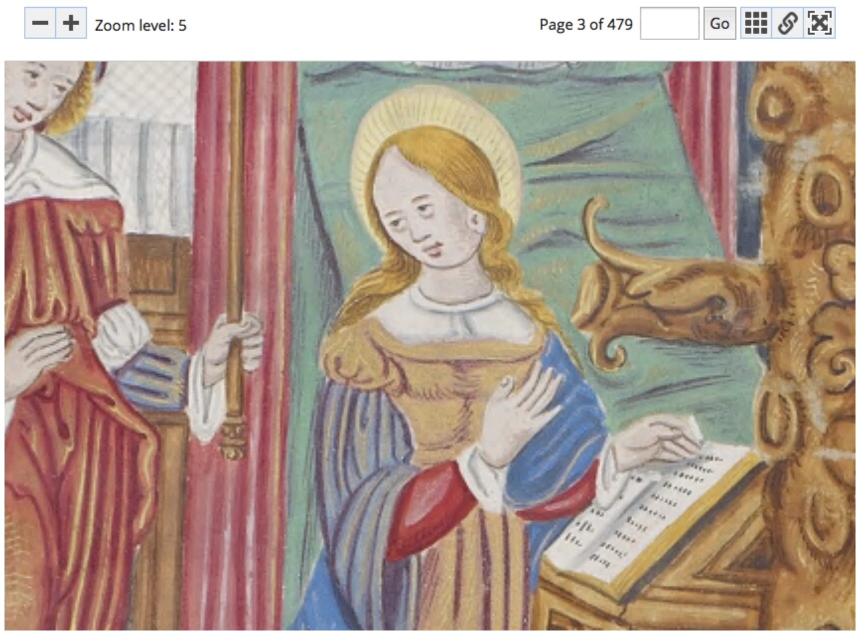


Figure 6.8: The Diva interface, showing multi-page scrolling (top) and zoom functionality (bottom).

6.2.1 Development History

The Diva project began as a pilot project with the Swiss working group of the Répertoire International des Sources Musicale (RISM) project. RISM is an international initiative, founded in 1952, whose purpose is to identify and catalogue musical sources held in libraries and archives around the world. In 2008, the Swiss RISM working group received funding for an exploratory project in digitizing musical sources (prints and manuscripts) by Swiss composers held in libraries and monasteries across Switzerland. The goal of this project was to build a system capable of publishing high-quality images of a source online, along with the extensive, research-quality metadata gathered by RISM.

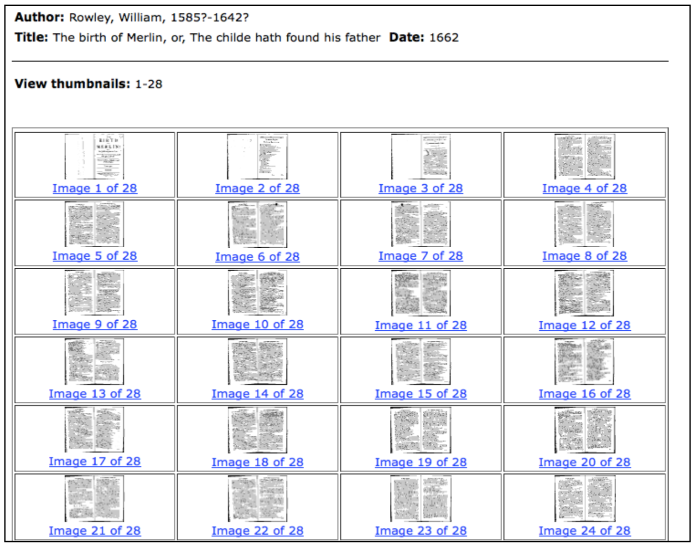
6.2.1.1 Environmental Scan of Digital Image Viewers

Several web-based document image viewers have been developed in recent years, primarily in response to extensive document digitization efforts. Presented here are the results of an environmental scan conducted prior to the development of Diva. The goal of this review was to examine tools and current practices for displaying high-resolution document images online.

Document Image Gallery

The most common method of displaying document images in digitized book systems uses the "document image gallery" format, where page images are displayed as individual images and users navigate through document images by clicking on small thumbnail representations arranged on a grid or in a list. An example of this type of interface is illustrated in figure 6.9, taken from the Early English Books Online (EEBO) interface. We found that interfaces designed in this style make it difficult to navigate the pages of the book as a cohesive unit, requiring users to click through several separate web pages as they move from one page to another.

To illustrate: All pages of a book are presented as a grid view, with each cell containing a small page image. To examine any single page, the user must click on an image from the grid view. This brings up a second view of the page optimized for viewing in the browser, but may not be usable for close examination if the text on the page is too small. Should a user wish to examine any part of a page in particular detail, the site may provide an option to download a larger, high-resolution image. The user must wait for this large image to download to their browser, which may take several minutes depending on the speed of the network connection and the size of the image. If the user is waiting for the full-quality image to download but wishes to continue browsing the document on the next page they must open a new window, navigate back to where they were, then traverse through the smaller thumbnails and start the process again.



Author: Rowley, William, 1585?-1642?
Title: The birth of Merlin, or, The childe hath found his father **Date:** 1662

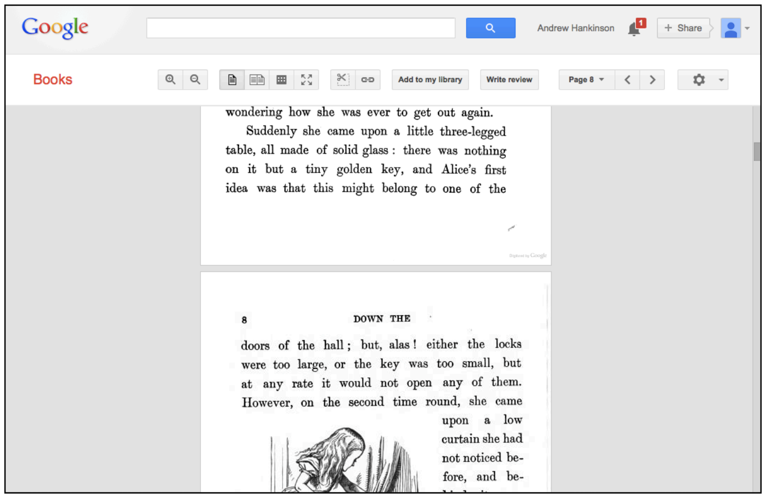
View thumbnails: 1-28

Image 1 of 28	Image 2 of 28	Image 3 of 28	Image 4 of 28
Image 5 of 28	Image 6 of 28	Image 7 of 28	Image 8 of 28
Image 9 of 28	Image 10 of 28	Image 11 of 28	Image 12 of 28
Image 13 of 28	Image 14 of 28	Image 15 of 28	Image 16 of 28
Image 17 of 28	Image 18 of 28	Image 19 of 28	Image 20 of 28
Image 21 of 28	Image 22 of 28	Image 23 of 28	Image 24 of 28

Figure 6.9: Book browsing interface from the Early English Books Online database

Google Books

The Google Books project presents items as a single scrollable entity embedded within the webpage (Google 2014b). This allows users to quickly scroll and navigate through the entire work. Their viewer software is integrated with their own book image delivery system and is not available as a separate component for libraries and archives to integrate into their own collections.



wondering how she was ever to get out again.
Suddenly she came upon a little three-legged table, all made of solid glass: there was nothing on it but a tiny golden key, and Alice's first idea was that this might belong to one of the

8 DOWN THE

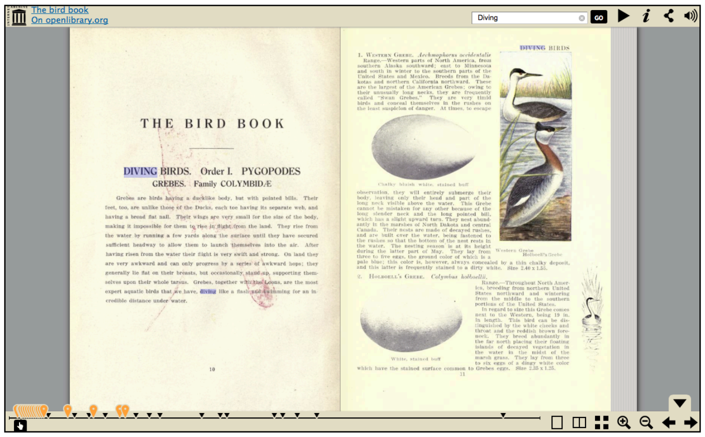
doors of the hall; but, alas! either the locks were too large, or the key was too small, but at any rate it would not open any of them. However, on the second time round, she came upon a low curtain she had not noticed before, and be-

Figure 6.10: Current Google Book Viewer

When the Google Book project launched in 2005 it used the image gallery format for viewing book images. Users were required to click "next" and "back" links to navigate between book page images. However, in 2006 the company launched a new interface that used AJAX techniques to automatically download page images as a user scrolled through the book (Chitu 2006) (figure 6.10). This mimicked the functionality of the popular Adobe Acrobat PDF browser plug-in but ran "natively" in the users browser without requiring installation of an extra plug-in. The Google Book viewer could load page images on demand as the user scrolled, and the user did not have to wait for the entire document to download to their computer before displaying the first page. The interface provides limited zoom functionality. Each page is delivered to the browser as a single image.

Internet Archive

The Internet Archive BookReader, developed for the OpenLibrary project, displays document page images in either a scrolling frame or as a physical book metaphor, allowing users to navigate the book by "turning" the pages (figure 6.11). Both the Google Books viewer and the Internet Archive BookReader serve full page images to the browser, but the entire image must be downloaded to view it. For high-resolution pages this can be slow as the user must wait for the complete image to download before the user can view the page.



THE BIRD BOOK

DIVING BIRDS. Order I. PYGOPODES
GREBES. Family COLYMBIDÆ

Grebes are birds having a ducklike body, but with pointed bills. Their feet, too, are unlike those of the Ducks, each toe having its separate web, and having a broad flat nail. Their wings are very small for the size of the body, making it impossible for them to rise in flight from the land. They rise from the water by running a few yards along the surface until they have secured sufficient headway to allow them to launch themselves into the air. After having risen from the water their flight is very swift and strong. On land they are very awkward and can only progress by a series of awkward hops; they generally lie flat on their breasts, but occasionally stand up, supporting themselves upon their whole tarsus. Grebes, together with the Loons, are the most expert aquatic birds that we have, diving like a flash and swimming for an incredible distance under water.

10

DIVING BIRDS

1. Western Grebe. *Aechmophorus occidentalis*

Range.—Western parts of North America, from southern Alaska southward; east to Minnesota and south in winter to the southern parts of the United States and Mexico. Breeds from the Dakotas and northern California northward. These are the largest of the American Grebes; owing to their unusually long necks, they are frequently called "Swan Grebes." They are very timid birds and conceal themselves in the rushes on the least suspicion of danger. At times, to escape

Chalky bluish white, stained buff

observation, they will entirely submerge their body, leaving only their head and part of the long neck visible above the water. This Grebe cannot be mistaken for any other because of the long slender neck and the long pointed bill, which has a slight upward turn. They nest abundantly in the marshes of North Dakota and central Canada. Their nests are made of decayed rushes, and are built over the water, being fastened to the rushes so that the bottom of the nest rests in the water. The nesting season is at its height during the latter part of May. They lay from three to five eggs, the ground color of which is a pale blue; this color is, however, always concealed by a thin chalky deposit, and this latter is frequently stained to a dirty white. Size 2.40 x 1.55.

Western Grebe
Holboell's Grebe

2. Holboell's Grebe. *Colymbus holboellii.*

Range.—Throughout North America, breeding from northern United States northward and wintering from the middle to the southern portions of the United States.

In regard to size this Grebe comes next to the Western, being 19 in. in length. This bird can be distinguished by the white cheeks and throat and the reddish brown foreneck. They breed abundantly in the far north placing their floating islands of decayed vegetation in the water in the midst of the marsh grass. They lay from three to six eggs of a dingy white color which have the stained surface common to Grebes eggs. Size 2.35 x 1.25.

White, stained buff

11

Figure 6.11: The Internet Archive BookReader interface

6.2.1.2 Design

Based on our initial environmental scan we developed five design criteria that drove the development of the first versions of this document viewer, and which have largely persisted throughout its development:

1. **Preserve document integrity**. Documents presented online should not be presented as disconnected images in an "image gallery" format. A user should have the ability to browse all

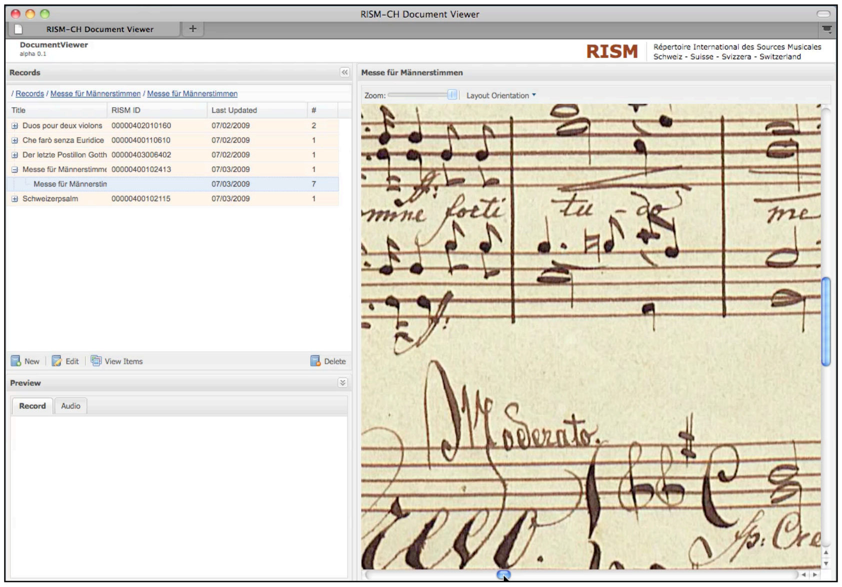
document images in a single web page, without needing to navigate between separate web pages for variants of the image.

2. **Allow side-by-side comparison of items.** For early music documents, especially, it may be highly desirable to display the images from multiple physical items in the same interface. The most common uses for this may be to display a score and parts, or images taken from multiple part books.
3. **Provide multiple page resolutions.** For some books it may be desirable to zoom in and out on a page, viewing greater or lesser image detail. For page images that have been digitized at very high resolutions this allows users to view small details that may otherwise be lost in versions of the image that have been resized or compressed to accommodate reasonable download times.
4. **Optimize page loading.** While it is desirable to view high-resolution images, these images can be slow to download and consume a significant amount of bandwidth. Using gigapixel image viewer technology, it is possible to break large images into smaller chunks, thereby optimizing the loading time by only downloading the portion of the images that the user is viewing at any time. (These gigapixel image viewers are typically used to view large images of the universe or detailed panoramic cityscapes.)
5. **Present item and metadata simultaneously.** Many libraries present their images separate from the metadata that describes the item they are viewing. If users are consulting the images, they typically have to flip back and forth between the images and the item record on different pages. This causes a disconnect between the two representations of the document.

Furthermore, we wanted to ensure that this image viewer ran natively in the browser without needing a plug-in, which places an extra barrier for usability of the digital collection, as users would have to download and install a plug-in before viewing the collection. We also felt a plug-in would impose a significant amount of development overhead, as it would need to function in many different browsers. We decided to implement the image viewer using only the technologies available in all web browsers: HTML, CSS, and JavaScript.

6.2.1.3 The DocumentViewer

The initial version of Diva, then simply called "DocumentViewer," was developed in 2008 using the Ext JS JavaScript framework (Orchard et al. 2009). It was developed as part of a prototype interface for viewing score images in the RISM database. Figure 6.12 shows the initial version of the interface "zoomed" in on a score, while figure 6.13 demonstrates the use of this interface to view two physical books simultaneously.



/ Records / Messe für Männerstimmen / Messe für Männerstimmen

Title	RISM ID	Last Updated	#
Duos pour deux violons	00000402010160	07/02/2009	2
Che farò senza Euridice	00000400110610	07/02/2009	1
Der letzte Postillon Gotth	00000403006402	07/02/2009	1
Messe für Männerstimme	00000400102413	07/03/2009	1
Messe für Männerstin		07/03/2009	7
Schweizerpsalm	00000400102115	07/03/2009	1

Figure 6.12: Initial version of the DocumentViewer

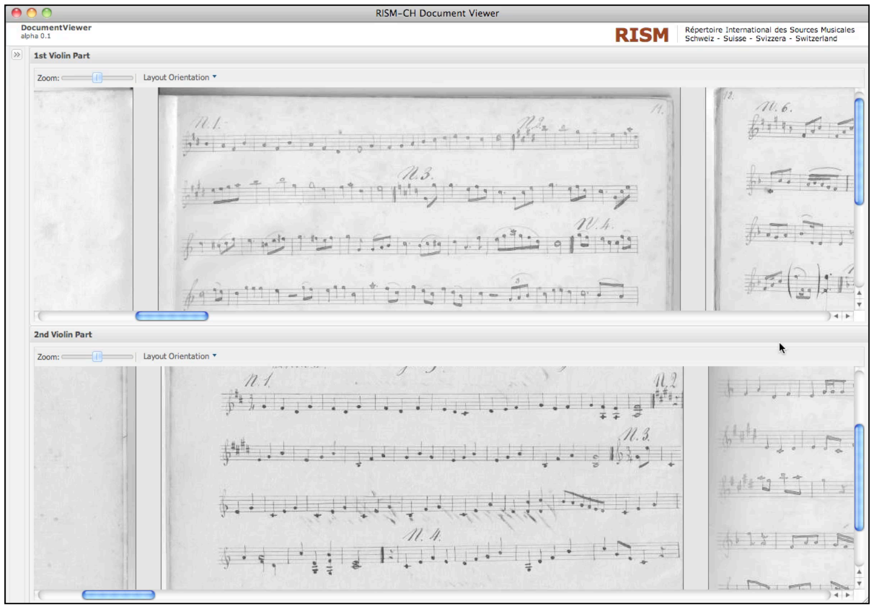
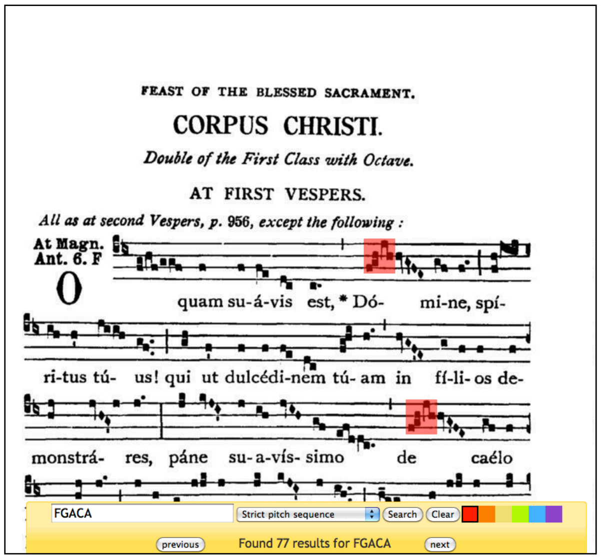


Figure 6.13: Viewing two books (Violin I and II) in the DocumentViewer interface

6.2.1.4 *Current Version*

When developing the *Liber usualis* project a viewer was needed to present images and search results in a web browser. We decided to re-implement the DocumentViewer core as a jQuery plug-in (jQuery Foundation 2014). This provided a more "lightweight" implementation of the page layout and viewing components, and made it easier to distribute the document viewer as a component that could be integrated into existing web applications. The DocumentViewer was renamed "Diva" and made publicly available as a standalone component. The first deployment of this new implementation was in the *Liber usualis* project, where it provided the interface for navigating through this book (figure 6.14).

Wendy Liu contributed several significant components for the current Diva viewer, developing the system through two releases of the software. She designed and implemented several of the components that will be discussed in the next section, including the dynamic DOM manipulation, the grid view, and the image manipulation functions.


FEAST OF THE BLESSED SACRAMENT.

CORPUS CHRISTI.

Double of the First Class with Octave.

AT FIRST VESPERS.

All as at second Vespers, p. 956, except the following :

At Magn. Ant. 6. F

O quam su-á-vis est, * Dó- mi-ne, spí- ri-tus tú- us! qui ut dulcédi-nem tú- am in fí-li- os de- monstrá- res, páne su-a-vís- simo de caélo

FGACA | Strict pitch sequence | Search | Clear

previous | Found 77 results for FGACA | next

Figure 6.14: *Liber usualis* interface showing region highlighting of search results

6.2.2 Components

A Diva installation has two primary software components, the IIP Image Server and a front end written in JavaScript (see figure 6.15). Measurement data encoded in JSON is served by a web server and used by the front-end to control the dimensions of the HTML elements used to display the page images in the interface.

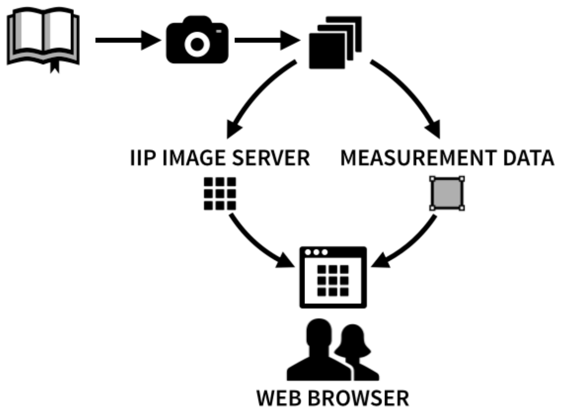


Figure 6.15: A high-level overview of Diva. Page images are digitally captured, and then served by the IIP Image Server as tiles to the user's browser, which uses pre-computed measurement data about the document stored and transferred in JSON to establish document and page layout.

6.2.2.1 Measurement data

An image pre-processing step generates measurement data for each document, and each image in the document. The data encoded in this file is listed below. Field names in the measurement data are intentionally succinct to reduce file size and optimize network transfer times. An example of the JSON measurement data stored by the process is given in figure 6.16.

The fields are as follows:

- `item_title`: The title of the document
- `dims`: A set of global dimensions for the entire document, including:
  - `a_wid`: Average page width at each zoom level
  - `a_hei`: Average page height at each zoom level
  - `max_w`: Width of the widest page at each zoom level
  - `max_h`: Height of the tallest page at each zoom level
  - `max_ratio`: The largest width:height ratio
  - `min_ratio`: The smallest width:height ratio
  - `t_hei`: Total height of all pages placed end-to-end

    - `t_wid`: Total width of all pages placed edge-to-edge
- `max_zoom`: The maximum number of zoom levels a page has in the document
- `pgs`: An ordered array of individual page measurements. Each page object contains the following fields:
    - `d`: An ordered array of dimension objects describing the page at each zoom level. The fields present in this object are:
        - `c`: Number of tile columns
        - `r`: Number of tile rows
        - `h`: Page height at that zoom level
        - `w`: Page width at that zoom level
    - `m`: The max number of zoom levels for that page
    - `f`: The file name for that page

```
{"item_title":"Salzinnes-cci",
"dims": {
 "a_wid":[137.93,275.87,551.75,1103.5,2207,4414],
 "a_hei":[216.52,433.05,866.11,1732.22,3464.45,6928.91],
 "max_w":[137.9375,275.875,551.75,1103.5,2207,4414],
 "max_h":[218.53125,437.0625,874.125,1748.25,3496.5,6993],
 "max_ratio":1.5842772995016,
 "min_ratio":1.4961486180335,
 "t_hei":[103717,207434,414869,829738,1659476,3318952],
 "t_wid":[66072,132144,264288,528576,1057153,2114306]
},
"max_zoom":5,
"pgs": [
   {"d":[{"c":1,"r":1,"h":218.53125,"w":137.9375},
         {"c":2,"r":2,"h":437.0625,"w":275.875},
         {"c":3,"r":4,"h":874.125,"w":551.75},
         {"c":5,"r":7,"h":1748.25,"w":1103.5},
         {"c":9,"r":14,"h":3496.5,"w":2207},
         {"c":18,"r":28,"h":6993,"w":4414}
        ],
"m":5,
"f":"salz-001r.tif"},
{...other pages...}]
}
```

Figure 6.16: JSON-encoded Diva document dimensions

6.2.2.2 IIP Image Server

The IIP Image Server (Pitzalis et al. 2006) is an open-source image server designed to serve high-resolution images over HTTP. It accomplishes this by allowing a client to request just a portion of the full image. Images are stored as formats that support a multi-resolution version of the image. To serve the image, IIP can address sections of each image as a tile. Each layer, and each tile within that layer, may be individually retrieved from the server using a URL.

The IIP Image Server is typically used to present extremely large images in a browser. A typical use for it might be to serve a high-resolution image of the entire earth, of a portion of the universe taken from satellite photographs, or detailed images of paintings and artwork (Pillay 2012). We use the IIP Image Server to provide the Diva viewer with the ability to view large, high-resolution images. Prior to serving the document images they must be converted to a format that supports high-resolution document viewing. This will be discussed next.

6.2.2.3 Image Processing

The high-resolution images served by the IIP server must be encoded in an image format that supports multiple resolution tiled image delivery. The IIP Image Server supports two such formats: Multi-resolution ("Pyramid") TIFF and JPEG2000.

The number of image resolutions available for a given image is determined by the maximum resolution of the original file and the size of the individual tiles. The formula for calculating the number of resolutions (n) available for a given image size is given in equation 6.1.

$$n = \left\lceil \log_2\left(\frac{D+1}{Ts+1} \right) + 1 \right\rceil$$

Equation 6.1: Number of resolutions based on image and tile sizes

The largest dimension, *D*, of the image (width or height) and the tile size, *Ts*, are both given in pixels. For example, an image with the largest dimension of 4120 pixels and a tile size of 256 pixels would result in an image with six discrete zoom levels. Each of these zoom levels are then decomposed into a representative number of 256 × 256 pixel tiles.

JPEG2000 (der Knijff 2011; JPEG Group 2014) is a newer and more advanced image format that supports tiling and multiple-resolution representations. Rather than store copies of the same image at varying resolutions, like multi-resolution TIFF, JPEG2000 uses wavelet transforms to create an image which may be dynamically represented at multiple resolutions by extracting lower or higher-resolution representations of the file (Li 2001).

6.2.3 Functionality

This section will present a few of the features of the Diva interface and provide some implementation details.

6.2.3.1 Preserve document integrity

To preserve document integrity, Diva presents the user with an entire document (i.e., a collection of page images) on a single web page. The user can scroll through this collection of page images in the document without needing to visit multiple web pages. Users also have the ability to "zoom" in and out to view multiple image sizes (i.e., high and low resolution representations of the images) within same scrolling interface, eliminating the need to visit multiple independent pages to view the images at larger or smaller sizes.

6.2.3.2 Multi-resolution image display

Whether Diva is displaying the complete page, or a small portion of a much larger page image, the number of tiles needed to represent the page images remains largely constant (figure 6.8). Should a user zoom in on an image to a point where the full image is larger than the visible area in the browser (figure 6.17), Diva optimizes the page image loading by requesting just the tile images that are visible to the user. Tiles that are immediately outside of the viewing area are pre-loaded to reduce waiting times for the user as they move the image. Moving the image within the visible area to expose previously-unloaded areas will automatically trigger the software to fetch and place the image tiles in their appropriate location.

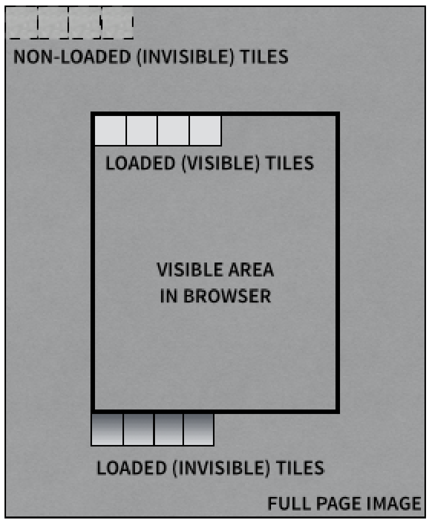


Figure 6.17: Diva loads only the tiles that are displayed in the visible area of the user's browser

6.2.3.3 Optimized page loading

In Diva, the decomposition of the page into tiles creates a highly optimized viewing system that can present high-resolution page images to users without requiring that they first download the entire document, or even the entire page image—a significant advantage over methods that distribute entire images or, in the case of PDFs, an entire document in a single file.

The JavaScript component manages the HTML elements for just three pages at a time (figure 6.18). This is done to reduce the memory usage of Diva, especially on devices with restricted processing and memory resources. Creating thousands of non-visible `<div>` elements was found to consume large amounts of memory in documents with many pages, resulting in reduced performance on all browsers, especially those with reduced memory resources such as tablets and smartphones. As the user scrolls, the JavaScript component adds and removes the page elements from the underlying HTML elements via the Document Object Model (DOM) interface. Page elements, and their associated tile elements, are dynamically added and deleted from the DOM as the user scrolls through the document images. A flowchart of the page loading and unloading process is shown in figure 6.19.

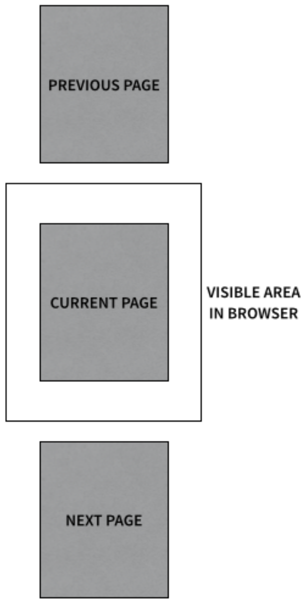


Figure 6.18: Illustration of the page loading optimization. Diva maintains the HTML elements for just three pages (previous, current, next) regardless of the total number of pages

6.2.3.4 Grid view

The Diva grid view was introduced to utilize available space in the browser to display multiple document page images. The grid view displays all the pages in a row and column arrangement (pages are arranged from the left to right, and then top to bottom) (figure 6.20). The number of pages per row can be dynamically adjusted. If a user sees a page they would like to view in detail, double-clicking on the page image will bring the image up in the page-view interface, where they can zoom in on the image further. Users may switch back and forth between grid and page view while maintaining their position in the whole document. This provides users with an overview display of the document without leaving the Diva viewing interface.

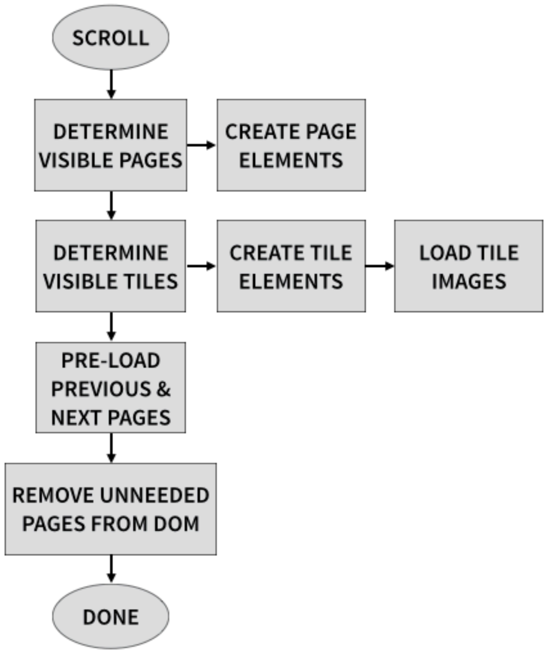


Figure 6.19: Diva scroll-loading Flowchart

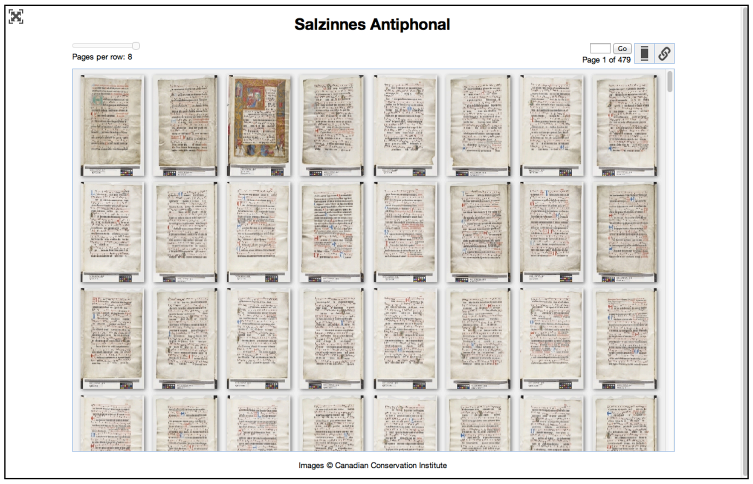


Figure 6.20: Diva grid view at eight pages per row

6.2.3.5 Permanent linking

Diva preserves the state of a user's viewer (zoom level, page position) in a URL. This URL may be bookmarked and shared, and used to re-create the original view in a later browsing session, or in another user's browser.

6.2.3.6 Synchronized viewer instances

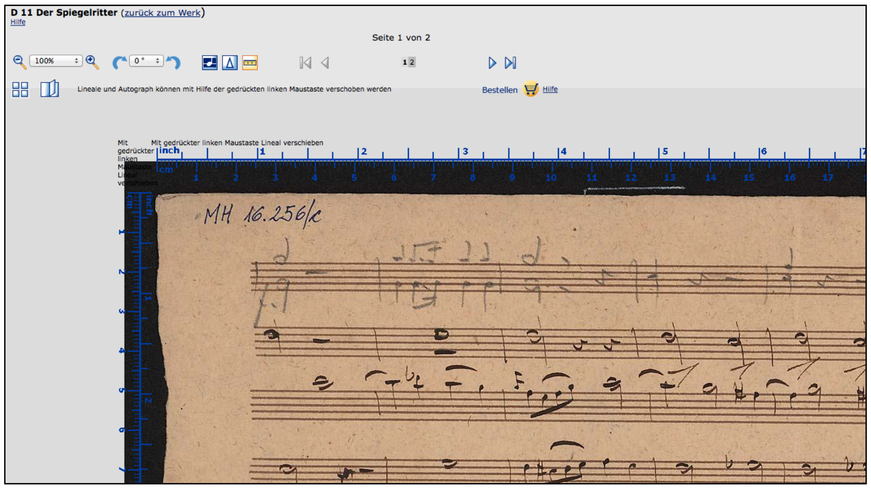
Multiple Diva viewers may be instantiated on a single web page, allowing multiple documents to be viewed simultaneously. Viewers may be scrolled independently, or they may be "locked" together to synchronize scrolling between two documents on the same web page.

6.2.4 Browser-based image manipulation

A unique feature in Diva that is not available in other digital document viewers is the ability to perform basic image manipulation tasks on the page images. These image manipulations include rotation, brightness, contrast, and RGB colour channel manipulation. This feature was designed especially to help users enhance particular aspects of a document image to make it more readable.

For example, image rotation can be useful to view marginalia that run perpendicular to the page orientation, or perpendicular fly-outs that have been folded over a page. Brightness, contrast, and colour channel manipulations may be used to enhance faded inks on a manuscript page, making them more legible.

Previous attempts at image manipulation in a document viewer have used tools installed on the server. In systems of this type, rotating a page image sends a request to the server, which performs the image manipulation operation and then returns the resulting image. This is a high-latency operation, as the user has to wait for the server to perform the manipulation and the browser to re-download the image. This type of interface is used, for example, in the Schubert Manuskripte project (Wienbibliothek im Rathaus 2010) (figure 6.21).



Lineale und Autograph können mit Hilfe der gedrückten linken Maustaste verschoben werden

Mit gedrückter linken Maustaste Lineal verschieben

Figure 6.21: The Schubert Manuskripte Image Viewer Interface. The image manipulation toolbar sends commands to the server, which then re-processes the image and sends the image back to the browser.

Instead of employing a server-side approach, Diva uses the HTML `canvas` element for pixel-level manipulation of images. Browser-based image manipulation is still a relatively new technology, and browsers cannot manipulate images with the same speed expected of a modern desktop application. On larger images this poses a user interaction problem, since an image transformation cannot be applied to the page image in a real-time reaction to user interactions. We mitigate this problem by providing users with a smaller preview image that can provide real-time feedback on how their manipulation might look when applied to the larger image. When the user is satisfied with their manipulations, they can then apply the manipulation to the larger image (figure 6.22).

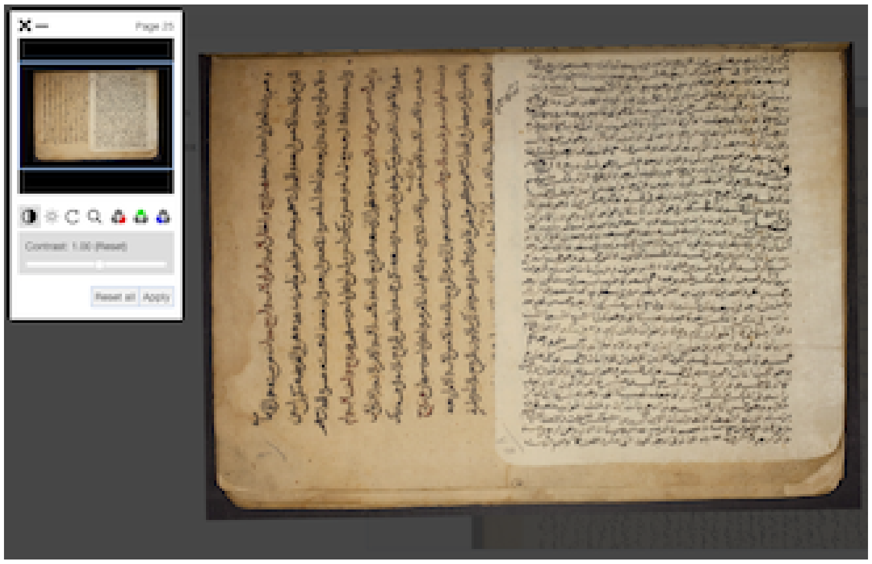


Figure 6.22: Browser-based image manipulation in Diva. A page image has been rotated 90° to view text written on a leaf that runs perpendicular to the orientation of the document image. Note the smaller preview image above for the manipulation controls.

6.2.5 Extending Diva

There are two ways for programmatic interaction with Diva. One is an API that allow other software to "hook" into its processes, receiving notifications of actions performed in the viewer, or changing the state of the viewer. Developers may define methods ("callbacks") to react to events that are triggered via interactions such as scrolling, zooming, switching modes (fullscreen or contained), or views (page view or grid view). This can be useful for loading extra information as the user scrolls through pages, such as loading and displaying metadata to the user about the currently visible page to the user.

A formalized method of extending Diva is the plug-in system. The plug-in capability was provided as a means of extending Diva without needing to modify the "core" functionality. Developers may write plug-ins to extend the functionality of Diva to accommodate new sources of information, or display new interfaces that can use the page images. A default installation of Diva comes with two plug-ins. One is browser-based image manipulation tools (§6.2.4), and the other is a simple plug-in that provides a link to download a given page image.

6.2.6 Discussion

While Diva is not the first, or the only, web-based document image viewer, it has several features that set it apart from other document viewers. The unique tile-based document image delivery system, along with the ability to scroll through all pages in a document, was designed to preserve document integrity and give users the feeling that they are browsing a complete document rather than a series of separate images. Diva does not require any third-party browser plug-ins. Integrated scrolling and zooming is a novel application of tile-based image viewers, incorporating document navigation and multi-resolution image views. Optimized page loading presents the user with a nearly "instant-on" view of the document, as only the visible portions of the document are loaded at any given time. Basic client-side image manipulation in a document viewer is also an important new feature, providing users with tools to manipulate document images through simple transformations and adjustments such as rotation, brightness, and contrast controls. Finally, a callback system allows developers to "hook" into the Diva viewer to interact with other page elements, while a plug-in system provides a method to extend the document image and delivery system to incorporate new functions without needing to "hack" the core system.

Several new features are in the planning stage for Diva. A document structure system will provide the viewer with the ability to present document images using metadata that describes physical layout (i.e., page openings), as well as structural data (i.e., chapter or section locations). A new plug-in is under development that allows for page-region highlighting. This may be used to integrate Diva with image-based document navigation, where page regions can be highlighted when they match a user's search query.

6.3 Neume Notation Encoding

This section will describe the creation of a customized MEI schema for encoding music from the *Liber usualis* (Benedictines of Solesmes 1961). This OMR project is described in further detail in chapter 7 of this dissertation. A description of the customization process, including definitions of the terminology and typographical conventions used here, is provided in chapter 4 (§4.6).

6.3.1 About the Liber usualis

The abbey at Solesmes, France was responsible for reviving the tradition of Gregorian chant in the late 19th century (Bergeron 1998). They produced a number of liturgical service books for the Catholic Church including missals, graduals, and, perhaps most famously, the *Liber usualis*, a book containing chants for Mass and offices for the most important feasts of the church year. These books were notated using square-note neume notation on a four-line staff (an example is shown in figure 6.23).

6.3.2 MEI Customization

Although MEI has included provisions for encoding neume notation since 2007, it lacks several elements needed to accurately capture Solesmes-style neume notation. A customized version of MEI was developed to add support for certain features of the notation that was not already supported in the encoding schema, such as *divisions* (similar, but not exactly equivalent to breath marks, graphically represented by several types of vertical lines across the staff), *episema* (a sign indicating note stress) and neume names and forms that were not present in the existing MEI encoding system.

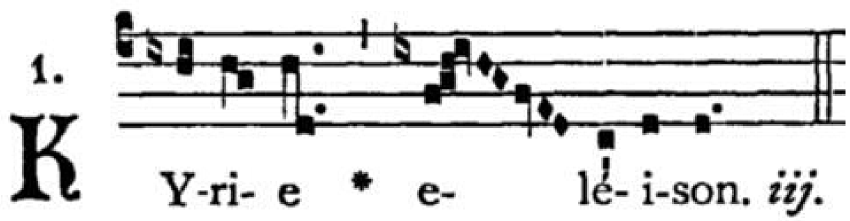


Figure 6.23: An example of the Solesmes neume notation showing a four-line staff, neumes, and divisions (vertical lines).

The ODD (One Document Does-it-all) modification file created for the Solesmes customization defines four new elements for MEI, as well as accompanying attribute definitions. A RelaxNG XML schema (§4.6.2) was derived from these definitions to validate the MEI-encoded output of our OMR system. In this section, the Solesmes customization will be illustrated by examining the definition of just the `<division>` element. Figure 6.24 shows how this element could be used in valid and non-valid contexts.

```
<division form=”comma” />
Valid, @form can take comma as a value.

<division form=”bell” />
Invalid, the value of @form must be one of the specified
values: comma, major, minor, small, final.

<division name=”long” />
Invalid, @name is not allowed on this element.

<clef>
   <division />
</clef>
Invalid, division cannot be a child of the clef element.
```

Figure 6.24: Valid and invalid use of the <division> element defined in the Solesmes module.

6.3.2.1 Element Definitions

The `<elementSpec>` definition (figure 6.25) is used to define a new element, `<division>`, with the name specified using the `@ident` attribute. The `@module` attribute specifies the MEI module to which this element belongs (`MEI.solesmes`), and the `@mode` attribute specifies the mode the Roma processor should use for this element. The `@mode` attribute may be one of “`add`,” for adding a new element, “`delete`,” for removing an existing element from the resulting schema, or “`replace`,” for re-defining an existing element (the “`delete`” and “`replace`” attribute values are not shown).

The `<desc>` tags enclose the documentation for this element. The Roma processor uses this information to derive the documentation for the resulting schema customization. The `<classes>` element specifies the classes, or groups of like-functioning elements, to which this element will belong. In this case, the `<division>` element belongs to, and automatically inherits, the XML attributes specified in the `att.common`, `att.facsimile`, and `att.solesemes.division` classes. Of these three attribute declaration classes the first two are defined in the MEI core, while the third is declared in the Solesmes customization.

```
<elementSpec ident="division" module="MEI.solesmes" mode="add">
  <desc>Encodes the presence of a division on a staff.</desc>
  <classes>
    <memberOf key="att.common"/>
    <memberOf key="att.facsimile"/>
    <memberOf key="att.solesmes.division" />
  </classes>
</elementSpec>
```

Figure 6.25: Declaration of the <division> element in ODD

6.3.2.2 Attribute Definitions

The `<classSpec>` declaration (figure 6.26) creates a new class of attributes, `att.solesmes.division`. This class defines a new group of attributes on any element that is a member of this class; in this case, only the `<division>` element is a member of this class. More general classes of attributes may be defined that apply to multiple XML elements (like the `att.common` class). The `@form` attribute is declared by the `<attDef>` element. Additional attributes may be declared by creating more `<attDef>` children of the `<attList>` element. The `@usage` attribute on `<attDef>` declares this attribute to be optional, meaning that it is acceptable if a `<division>` element does not possess a `@form` attribute. Required attributes may be specified by setting the value of this attribute to "`req`".

The `<valList>` element defines the possible values that the `@form` attribute may have; in this case the only valid values for the `@form` attribute are given by the `<valItem>` elements. Since the value list here is a closed set, specified by the value of `@type`, any values supplied in the `@form` attribute that is not one of those specified will not pass validation (figure 6.24).

The full Solesmes module contains definitions for four new elements, `<division>`, `<episema>`, `<neume>`, and `<nc>` (neume component) and eight new attributes to accompany these elements. When this customization is processed with the Roma processor against the

2013 MEI core, a schema is produced that can be used to validate Solesmes-style neume notation MEI files.

```
<classSpec ident="att.solesmes.division"
   type="atts" mode="add">
  <desc>Divisions are breath and
       phrasing indicators.</desc>
  <attList>
    <attDef ident="form" usage="opt">
      <desc>Types of divisions.</desc>
      <valList type="closed">
        <valItem ident="comma" />
        <valItem ident="major" />
        <valItem ident="minor" />
        <valItem ident="small" />
        <valItem ident="final" />
      </valList>
    </attDef>
  </attList>
</classSpec>
```

Figure 6.26: Declaration of the att.solesmes.division class to describe a common attribute group.

6.4 OMR Interchange Package

The purpose of the OMR Interchange Package (OIP) (Hankinson et al. 2010) is to define a file structure that combines image files and the OMR-derived MEI files into a single container suitable for transporting between software systems. The OIP format may be used by both OMR applications and applications that use the results of the OMR process.

For text documents there are similar formats that can be used to maintain correspondence between text and image (reviewed in Chapter 2, §2.6). For music notation applications, there are no file formats available that maintain both images and notation data as a single file, similar to PDF or DjVu. Likewise, for OMR systems there are no equivalent formats for encapsulating and transmitting the output of the OMR process in a way that preserves correspondence between the symbolic and the visual representation, like hOCR, METS/ALTO, or

XDOC. There is a need for a file format that can preserve both symbolic music notation data and image files in a single file, suitable for interchange between different software components, including search systems and display systems. The OIP format was developed to meet this need.

An OIP file is a collection of files and folders serialized as a single file, (i.e., a ZIP file). A minimal standard for organizing the content of these files, the "BagIt" specification, was chosen to impose a standardized file contents structure, rather than simply defining an *ad hoc* method of bundling these files and folders together.

6.4.0.1 BagIt

The BagIt format is a lightweight file bundling specification maintained by the Library of Congress and the California Digital Library. It is currently an Internet Engineering Task Force (IETF) draft standard (Boyko et al. 2014). The name, "BagIt," refers to a colloquial rendering of the Enclose and Deposit method (Tabata et al. 2005), also known as the "bag it and tag it" method.

This format defines a simple hierarchy of files and folders, known collectively as a "bag." These can be represented plainly on any computer system as standard files and folders, and may also be a single file compressed using standard ZIP or TAR packaging systems.

Minimally, one directory and two files must be present in every bag in order to be considered compliant to the standard. One of the required files is a `bagit.txt` file that stores the version of the BagIt specification to which that bag conforms and the character encoding used for the metadata files. The second required file is a manifest file listing checksums for each file within the data directory, helping to ensure the integrity and identity of each of the files in the bag. A `data` directory must be present, but this is unstructured and may contain any arrangement of files or folders. This is the bag's "payload." Other optional files are outlined in the BagIt specification.

6.4.0.2 OMR-specific folder layout

For the OIP format, we specify a file hierarchy within the `data` directory of a bag. A folder is created in the `data` directory for each page in a multi-page document, allowing the format to accommodate documents of any number of pages. In each page folder, we store files relating to this page, including any image files (originals or derivatives generated from the OMR process), and notation files (i.e., MEI) containing the transcribed symbolic music notation. Other files may be stored in each page folder, as desired. A generalized example of the OIP structure is given in figure 6.27.

```
<bag-directory>
     |- bagit.txt
     |- manifest-md5.txt
     |- [other optional bagit files]
     |- data
         |- [page 1]
         |     |- [image files]
         |     |- [notation files]
         |     |- [metadata files]
         |- [page 2]
         |     |- [image files]
         |     |- [notation files]
         |     |- [metadata files]
         |- [etc.]
```

Figure 6.27: A generalized OIP structure

The BagIt specification allows for a free-form structure within the `data` directory. Software capable of processing BagIt files can therefore guarantee the integrity and identity of each file in the bag without needing to understand the OIP format. Since the format specifies a standard compression technique, and the structure is implemented using a standard directory structure, there are countless tools that can uncompress and display the contents of the BagIt file. However, further processing of the contents, to perform tasks such as validating file contents against their stored checksums, or ensuring the structure conforms to the BagIt specification, requires software that understands and implements that specification.

6.4.1 PyBagIt

PyBagIt (Hankinson 2013) is a Python software library for creating, reading, validating, and writing BagIt files. Currently it can work with BagIt files that conform to version 0.96 of the specification. It was developed as a tool for creating OIP packages, but it may be used in other systems as a generic implementation of the BagIt specification—it contains no OIP-specific functionality. As a Python module it can be incorporated into many of the tools used for OMR as a tool for packaging and delivering the results of an OMR process.

6.4.2 OIP Summary and Future Work

The OMR interchange format was developed to specify a consistent file structure between different OMR clients, as well as any software that may be used to further process or display these results. Several uses for OIP files have been implemented or envisioned as part of future work on building OMR-derived notation search systems:

1. As the input for a notation and image search system.
2. As a format for interchange of OMR “ground truth” information between adaptive OMR systems.
3. As a format, similar to PDF, for allowing users to browse musical images in either a web-based or desktop-based application. These applications might also supply *in situ* search and highlighting within the document, similar to searching within a PDF file.

6.5 Crowdsourced Correction Tools

In chapter 4, we looked at how a collaborative approach to OMR could enable distributed participation in the OMR process. This section will describe two browser-based tools developed to collect human-supplied notation correction data. These tools are envisioned as part of a larger effort to enable distributed music correction interfaces for large-scale OMR, and also as a core component of the vision for networked adaptive OMR. By focusing on developing browser-based interfaces for these tools, we forego the need to install any extra software on a user's computer, providing a large potential user base for performing crowdsourced tasks.

One area where crowdsourced OMR collaboration will be most needed is in the area of automated transcription correction. Beyond the obvious need for correcting mistakes in automated recognition, an adaptive recognition system can use human-corrected results to adapt and refine its recognition models to improve its recognition accuracy on subsequent documents.

The first system presented is Neon.js, a web-based neume notation editor that supports correction and image alignment of automated recognition results. The second, a web-based classifier interface, is an initial prototype that mimics the Gamera symbol classification interface in a web context.

6.5.1 Neon

Neon (Burlet et al. 2012) is a web-based neume notation editor. It was developed as both a standalone editor and an interactive tool in the Rodan OMR system. It was designed primarily as a correction interface for OMR, preserving the direct relationship between the symbolic notation and the original image (figure 6.28). Neon was developed by Greg Burlet, working under my direction. Greg is responsible for developing all of the internal layout, notation interaction system, and the development of the "neumify" algorithm that attempts to predict a neume shape based on the configuration of individual pitches. For example, if two pitches are placed in ascending order and then grouped, the neumify algorithm will produce a *podatus;* If they are placed in descending order and grouped, it will produce a *clivis.*

In Neon the music notation is encoded in MEI, along with the positional information, storing where, on the original image, a particular musical symbol (notes, clefs, staves, etc.) is located. As users edit, add, or delete symbols from the score displayed on-screen, the underlying encoding is updated to maintain their correspondence with the page image.

The Neon interface features several controls to help the user in their correction task. The opacities of the original image and the notation layer may be adjusted, allowing users to configure their view to align the notation and the image.

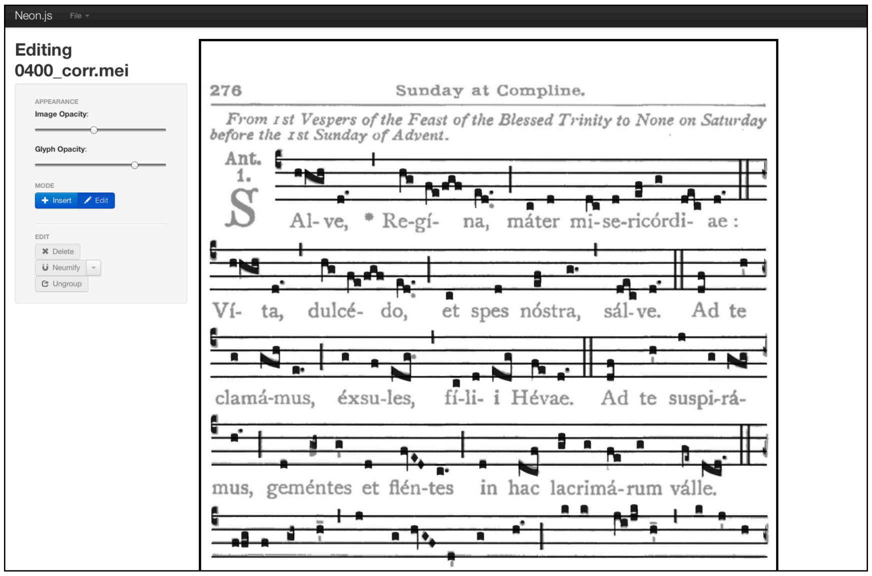

Neon.js
File
Editing
0400_corr.mei
APPEARANCE
Image Opacity:
Glyph Opacity:
MODE
Insert
Edit
EDIT
Delete
Neumify
Ungroup
276
Sunday at Compline.
From 1st Vespers of the Feast of the Blessed Trinity to None on Saturday before the 1st Sunday of Advent.
Ant. 1.
S
Al- ve, * Re-gí- na, máter mi-se-ricórdi- ae :
Ví- ta, dulcé- do, et spes nóstra, sál-ve. Ad te
clamá-mus, éxsu-les, fí-li- i Hévae. Ad te suspi-rá-
mus, geméntes et flén-tes in hac lacrimá-rum válle.

Figure 6.28: Neon.js editing interface. The transcribed notation (darker) can be seen overlaid with the original image (lighter).

The initial version of the Neon editor was designed to work with notation from the *Liber usualis.* It has since been used to correct OMR results from pages of the *Salzinnes Antiphonal* (see §7.2, §7.3).

6.5.2 Web-based Gamera Classifier

The Gamera classifier interface (figure 6.29) was developed by Michael Droettboom and Karl MacMillian at The Johns Hopkins University. This interface is designed for labelling connected components—glyphs formed by contiguous black pixels—for use in the Gamera shape classifier. Users see a list of previously-used symbol classes (the sidebar on the left), the classified glyph shapes (right, top), the unclassified page glyphs (right, centre), and the original context of the glyph on the page image (right, bottom). Selecting a glyph on the page image or the page glyphs section allows the user to assign a label to the glyph, adding it to the available glyphs in the classifier.

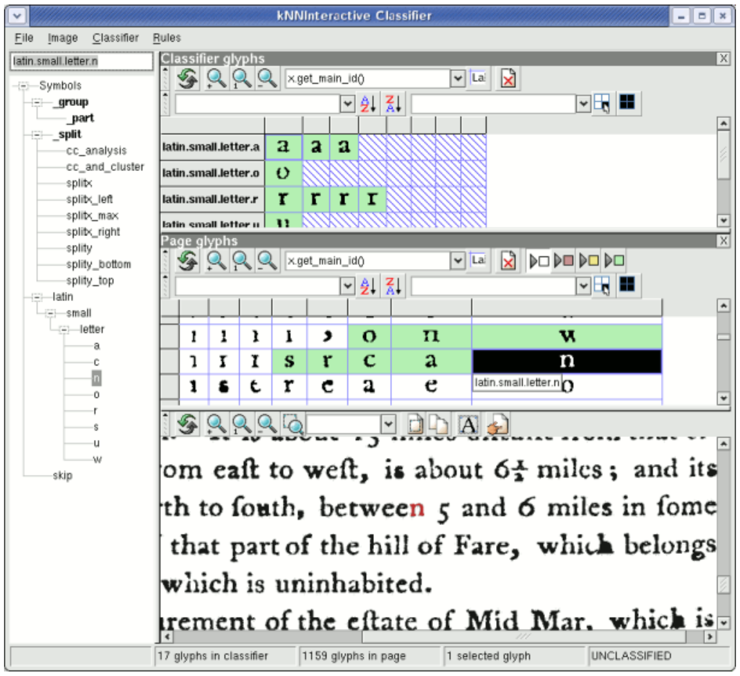


Figure 6.29: The Gamera classifier interface (from Gamera 2012)

While the Gamera interface works well for its designed purpose, it requires the installation the Gamera software, a task typically performed by software developers and not one suited for most computer users. Users also need to manually manage the XML files generated by the classifier interface.

For Rodan we developed a browser-based version of the Gamera classifier interface (figure 6.30) as part of an interactive workflow task for collecting labels for page glyphs. Laurier Baribeau was responsible for implementing this interface using the Cappuccino framework (§5.4.1).

Using this interface, users of Rodan can assign labels to shapes. This interface automatically reads from, and writes to, a server-side XML file, negating the need to manually manage classifier data. As users supply label data, these become accessible to all Gamera-based classifier tasks in a Rodan workflow. This allows a classifier used for one page to be extended using symbols from other pages, and allows pre-built classifiers to be used by other workflows without needing to re-build and re-train the classifier database for additional documents.

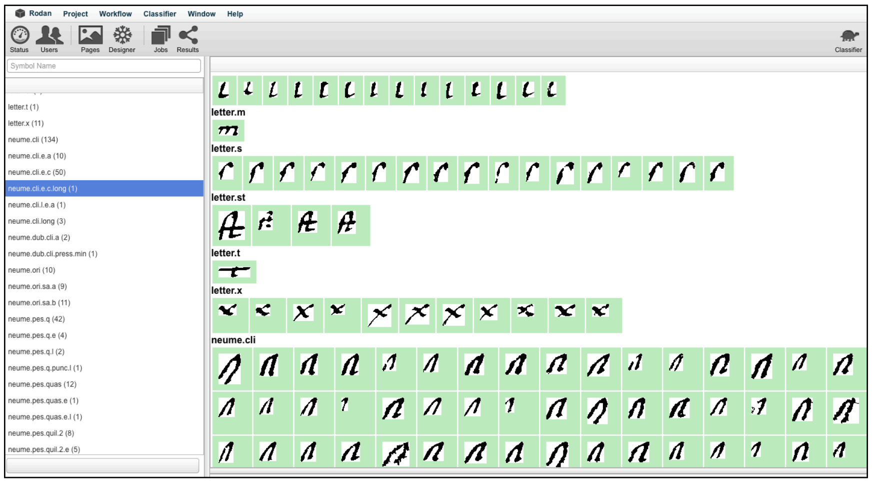


Figure 6.30: Web-based Gamera Classifier Interface used for classifying St. Gall neume shapes

6.6 Chapter Summary

This chapter presented several tools created to support the vision of large-scale OMR, although several of these tools have also been used in contexts outside of OMR.

LibMEI was created as an expression-agnostic tool for working with MEI files. With LibMEI it is possible to express MEI files in several hierarchical constructions, which demonstrates the applicability of the MEI rules of encoding beyond any specific expression format. Using the tools developed for LibMEI, one can auto-generate an extended version of the library that conforms to either the core, or a customized, version of the MEI schema. In addition to the C++ and Python versions of LibMEI, a derivative version of the library was created to enable reading and writing MEI files in the Sibelius notation editor.

The Diva document image viewer is a web-based, multi-page, high-resolution document image viewer. The client-side system and interface is implemented entirely in JavaScript, HTML, and CSS, while the server-side system depends on the IIP Image Server and a standard web server for delivering image measurement data. Diva has several unique features among existing book image viewers, including highly optimized page loading techniques, browser-based image manipulation, and the ability to write "hooks" and "plug-ins" to further integrate the Diva viewer into existing interfaces, and extend the functionality of the core viewer.

For the *Liber usualis* project we developed a customized MEI encoding schema to encode the neume notation used by the monks of Solesmes for their publication of the *Liber usualis.* The process of developing a customized MEI schema was illustrated by way of discussing the specific implementation details of our custom neume notation encoding scheme.

The OMR Interchange Package (OIP) is a file format designed to store the results of an OMR process. This format uses the BagIt specification to control the structure of the file, but the OIP describes a hierarchy of files and folders for page images into a single serialized file (i.e., a ZIP file) suitable for sharing between software clients. A review of similar file formats for text recognition revealed that there are several methods for delivering aligned text and images, but there are currently no formats available for delivering the same functionality for music notation.

Finally, distributed correction tools were developed for performing browser-based OMR correction. Neon.js is a JavaScript neume notation editor, designed to show both images and notation simultaneously to aid users in correcting the output of OMR results. The Gamera classifier interface was ported to run as a component of Rodan, offering users the ability to perform connected-component classification in a way similar to that offered by the Gamera classifier interface.

The next chapter will present three prototype projects developed in the course of this research. These projects provided the impetus for the development of these tools, and have served as a technology demonstration platform to inform and direct their design and implementation.

7. Applications

The previous chapters have discussed processes, tools, and applications for large-scale OMR. These were developed in the context of several proof-of-concept projects, each designed to investigate page images as the central means of navigating a digitized music document. This chapter will describe each of these projects, beginning with the *Liber usualis* (§7.1), and will then describe two projects centred on the *Salzinnes Antiphonal* (§ 7.2, § 7.3).

7.1 Liber usualis

The *Liber usualis* was chosen as the basis for our first large-scale production efforts because it features a number of characteristics useful for prototyping large-scale OMR, as well as providing a useful resource for scholars and students. It is a moderately large book (2,340 page images in the digitized version), which made it suitable for observing characteristics of a large-scale OMR effort that might not be noticeable with a project involving fewer page images. It is mechanically produced (i.e., printed), so the layout and symbols are uniform across the whole book (unlike hand-produced sources, i.e., manuscripts). It contains a mixture of text (lyrics, liturgical texts, performance instructions, etc.) and music notation, which provides a real-world example of a mixed-content source where the software must identify and separate the musical content from non-musical content. The music is monophonic (a single line of music) and as such does not require rhythmic alignment or separation of multiple voices (e.g., chords or multiple instruments on a single staff). The tools that provide symbol recognition and notation reconstruction are therefore relatively straightforward to build, providing opportunities to observe the throughput of the entire system without having to deal with some of the more complex aspects of CWMN.

The *Liber usualis* is an important reference book for musicologists and librarians attempting to locate chants and information about them. The *Liber usualis* contains codified versions of chants introduced as early as the 8th century. This book is still used as a service book in the liturgies of the Roman Catholic church. As well, it serves as a reference for identifying chants used as the *cantus firmus* in a polyphonic work. In the *Liber usualis* project a web-based search interface is provided to help users search and retrieve the chants from the page images. This web interface is currently available at:
`http://ddmal.music.mcgill.ca/liber/`

7.1.1 Music notation in the *Liber usualis*

Neume notation is a broad term used to describe several styles of music notation. It is widely believed that it emerged as a mnemonic device to provide a singer, or a group of singers, with a way of recording a melody for singing a liturgical text (Helsen 2013). The music in the *Liber usualis* is notated using square-note neume notation (figure 7.1), a system dating back to the 12th and 13th centuries (Helsen 2013), but modified in the printed Solesmes chant books.

The neume notation of the *Liber usualis* is placed on a four-line staff. A short example of neume notation from the *Liber usualis* is shown in figure 7.1. Pitches are grouped into graphical symbols called "neumes" which are used as guides for singing syllables of text, where a single syllable may get two, three, or more pitches; in melismatic passages many neumes may be present on a single syllable. There are a number of standard neumes shapes, each with its own name, but custom groupings of neumes may also be formed and fall under the general "compound" neume type. Some examples of these are given in figure 7.2.

In figure 7.1, the first symbol on the staff is a "C" clef on the top line. The syllable "Ky" is sung to a two-note *podatus*. The first syllable of "e-le-i-son" is notated using nine pitches arranged in several neumes.

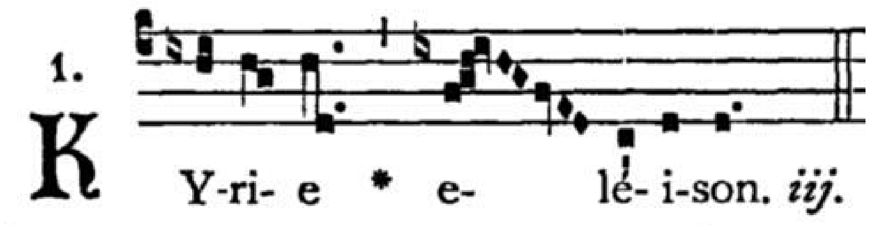


Figure 7.1: A sample of neume notation from the *Liber usualis*

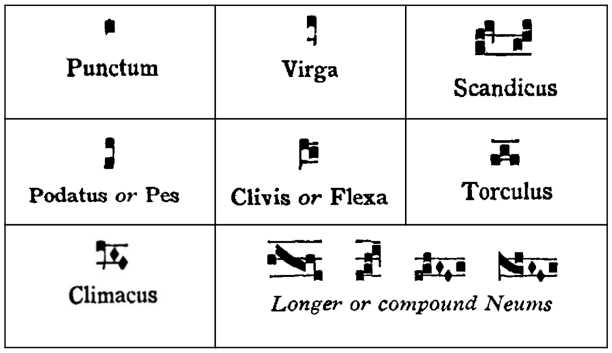


Figure 7.2: Neume shapes and their names

7.1.2 Workflow

A description of the history of the workflow for processing the *Liber usualis* can be found in Chapter 5 (§5.1). This was provided as background for the development of a generic workflow system for OMR. Here the specific steps of the *Liber usualis* project workflow, including the lessons learned from this project, will be discussed. A diagram of the workflow is given in figure 7.3.

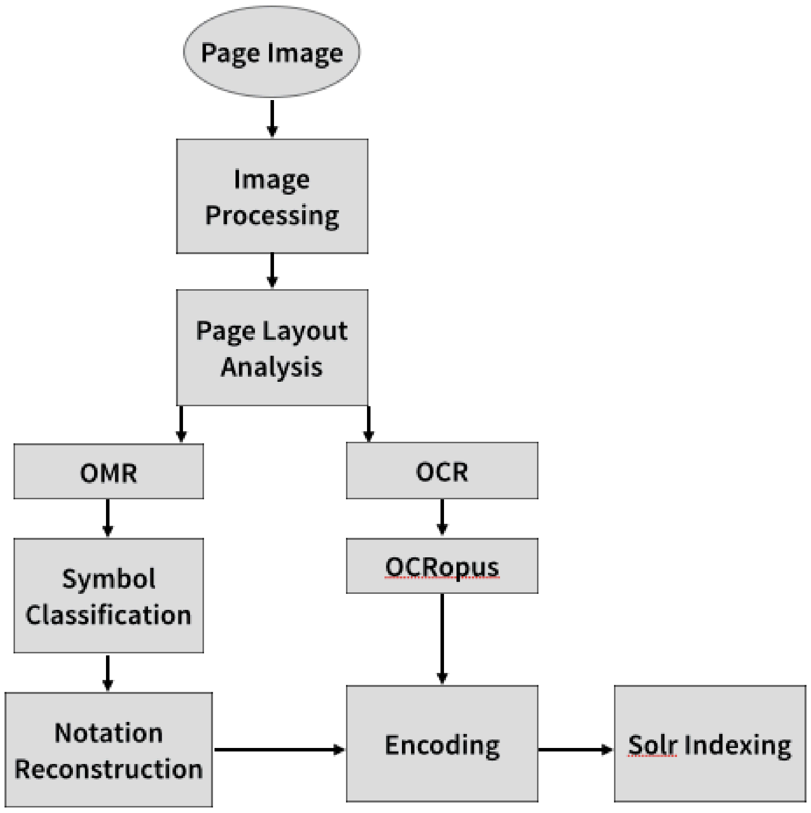


Figure 7.3: Workflow used for creating the searchable *Liber usualis*

7.1.2.1 Image Processing

The page images for the *Liber usualis* were extracted from a PDF file made available by the Canons Regular of St. John Cantius (Canons Regular of Saint John Cantius 2010). This PDF contains images taken from the 1961 edition of the *Liber usualis.* The page images were exported from PDF using Adobe Acrobat Professional as 500 PPI TIFFS, resulting in 2,340 image files. The source file had been previously processed for OCR using ABBYY FineReader (Abbyy 2014), and so the resulting image files had been previously binarized using the built-in ABBYY binarization tools.

7.1.2.2 Layout Analysis

Once the images were exported, we performed automated page layout analysis on them using a modified version of Aruspix, an OMR system designed for early music (Chapter 3, §3.3.2.1). This layout analysis automatically detected five different types of page elements: music, titles, lyrics, ornate letters, and other text elements (figure 7.4). A sixth option, "blank," is generally used for image artifacts like borders and creases that may appear as black pixels on the image but are not part of the content of the page. A Python script was written to automatically run Aruspix on every image without human intervention. After all the pages were analyzed we had a human confirm and correct any errors. The median layout correction time per page was 77 seconds, with the majority of pages taking between 30 and 130 seconds. The result was an image that could be segmented into separate layers containing textual or musical content exclusively. The layers containing musical notation were sent to OMR software, while the text layers were sent to OCR software.

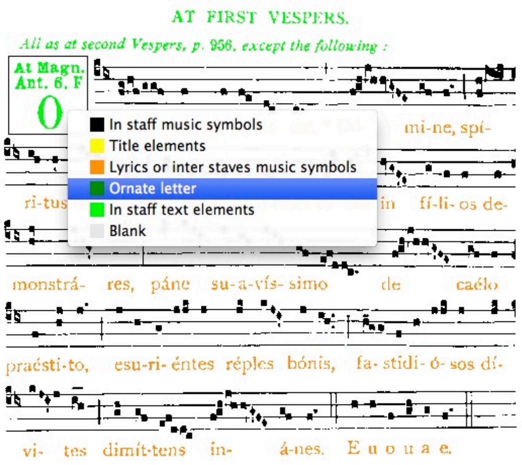

AT FIRST VESPERS.
All as at second Vespers, p. 956, except the following :
At Magn. Ant. 6. F
O
In staff music symbols
Title elements
Lyrics or inter staves music symbols
Ornate letter
In staff text elements
Blank
mi-ne, spí-
ri-tus
in fí-li-os de-
monstrá- res, páne su-a-ví-simo de caélo
praésti-to, esu-ri-éntes réples bónis, fa-stidi-ó-sos dí-
vi- tes dimít-tens in- á-nes. E u o u a e.

Figure 7.4: A page of the *Liber usualis* segmented using Aruspix

7.1.2.3 Music Segmentation and Symbol Classification

Gamera was used for performing symbol classification. Images containing the music notation layer were processed to remove the staff lines using the Musicstaves toolkit (Dalitz et al. 2008a). The result was an image that had only neume shapes on it (figure 7.5). These shapes were classified using a Gamera classifier containing labelled neume shapes drawn from a number of pages of the *Liber usualis*. The automated classification for each page was verified and corrected by musicians familiar with square-note notation. The median correction time per page was 11 minutes, with the majority of pages taking between 7 and 16 minutes.

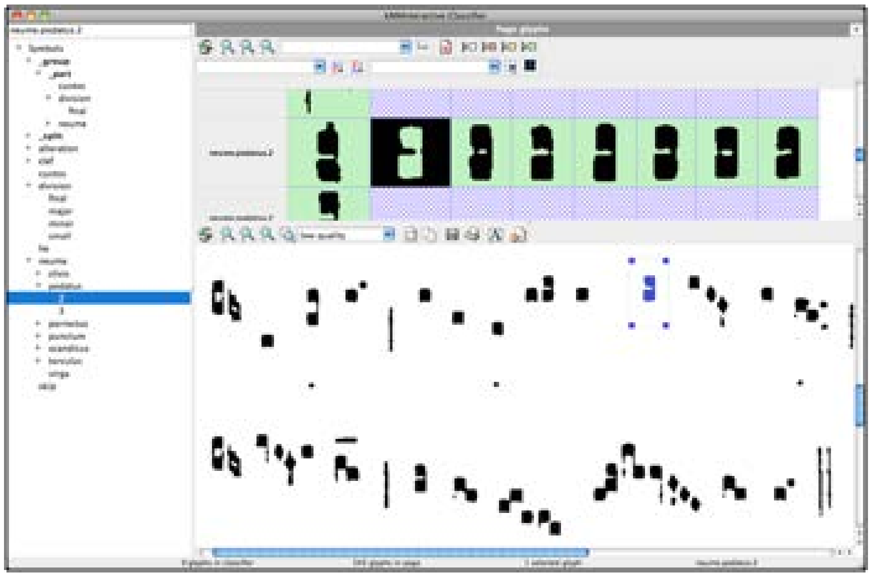


Figure 7.5: A *Liber usualis* page loaded into the Gamera classifier interface

The class names were designed to assist in recognizing the pitch content of a neume group (table 7.1). Class names for the shapes were constructed using the neume name along with the pitch contour of the neume. Each named neume shape has a specific contour which was "hard-coded" into our notation reconstruction software, so interval direction (up, down, same) was not explicitly encoded. The size of the intervals in the neume was provided using dot-separated notation (see table 7.1 for examples). For complex compound neumes where the pitch contour could not be inferred from the name, direction information was encoded in the class name. Thus, the specific pitch content of a particular neume shape could be reconstructed from knowing only its starting pitch and the width of each interval. For example, a three-note torculus (which goes up, then down) belonging to the `torculus.3.2` class and starting on a G would outline the notes G, B, and A. Additional features of the neume shape could also be encoded in the class name, including horizontal and vertical episemas and dots. Full details of this work, including performance evaluations of this technique, can be found in Vigliensoni et al. (2011).

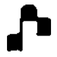
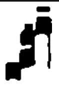
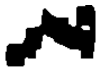
Neume Class Name	**Shape**
`neume.torculus.3.2`	
`neume.scandicus.2.2.2.he`	
`neume.compound.u2.u2.d2.u2`	

Table 7.1: Neume shapes and their Gamera class names

7.1.2.4 Encoding and Correction

After the music recognition was complete the staff lines were re-introduced on the image. The clef shape (F or C) and position for each staff was identified, and correlated with a staff line. The initial pitch for each neume was identified based on the lowest-left corner of the shape, and correlated relative to the staff line and the clef. The neume class name was used to identify each of the pitches in that neume.

The Music Encoding Initiative (MEI) format (Roland 2009) was used to encode the results of the music recognition stage with the customized neume encoding schema presented in Chapter 6 (§6.3).

For a final pitch verification and correction step the MEI and image files were opened in Aruspix using a purpose-built graphical neume editor (figure 7.6). The human corrector could see the original page image and the automatically recognized musical output rendered as editable notation. The notation was corrected and roughly aligned with the original image, and the output saved as MEI-encoded notation.

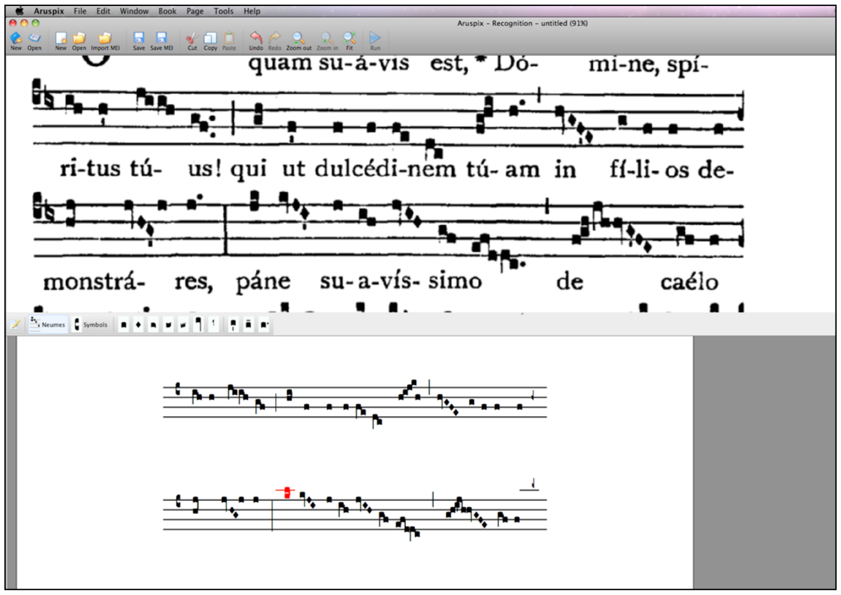

Aruspix File Edit Window Book Page Tools Help
Aruspix – Recognition – untitled (91%)
quam su-á-vis est, * Dó- mi-ne, spí-
ri-tus tú- us! qui ut dulcédi-nem tú- am in fí-li- os de-
monstrá- res, páne su-a-vís- simo de caélo
Neumes
Symbols

Figure 7.6: Custom neume editor in Aruspix developed for the *Liber usualis* project.

7.1.2.5 Optical Character Recognition

Page images with only textual content were sent to a separate recognition process using the OCRopus OCR software (Breuel 2008). The "raw" recognized text was processed through an automatic spelling corrector (Norvig 2014) trained on a dictionary of Latin and English words.[11] For lyrics, syllables with dashes between them were automatically joined to form a single word for the purposes of searching the text underlay. The results of the OCR process were not corrected or verified by a human due to time constraints, and are therefore suitable as a proof-of-concept but not entirely reliable. The OCR output for each line of text was correlated with a region on the image in order to allow us to highlight search results *in situ.* The text lines with position information were stored in the MEI file for each page, but with no encoded relationship between the musical content and the text (i.e., the text underlay was not encoded as a component of the music notation).

7.1.2.6 Indexing and Search System

To provide a very basic pitch search, the neume pitches in each MEI document were indexed using simple *n*-grams, ranging from 2- to 10-grams. We used a variation of the technique presented by Downie (Downie 1999), storing the explicit pitch value of each *n*–gram, contour, interval, and component neume names (figure 7.7). In addition, the co-ordinates of each *n*-gram on the page image were calculated and stored. Pre-computing these indexes into text allowed us to store the indexed content in readily-available tools for textual search.

```
contour: 'dduurr'
intervals: 'd2_d2_u2_u2_r_r'
location: [{'width': 407, 'ulx': 257, 'uly': 1459,
'height': 65}]
neumes: 'punctum_clivis_podatus_punctum_punctum'
pagen: 157
pnames: 'edcdeee'
semitones: '-2_-2_2_2_0_0'
```

Figure 7.7: Sample search index entry

11. The complete list of words used for training the corrector is available at: https://github.com/DDMAL/liberusualis/blob/master/ocr/latin-english.txt.

The initial system used ElasticSearch (Kuć and Rogoziński 2013) as our search software (Thompson et al. 2011). This software responded to updates to our CouchDB (Anderson et al. 2010) databases and automatically updated its index as new content was added, facilitating very fast lookups over large indexes. While this setup provided the desired functionality, ElasticSearch crashed regularly and needed constant monitoring. To remedy these problems we replaced both the CouchDB and ElasticSearch components with a Solr (Smiley and Pugh 2009) installation, which is still in use.

Our Solr instance contains 3,006,964 unique *n*-gram documents in an indexing structure identical to the one shown for our CouchDB system. Indexing the *Liber usualis* corpus was performed with a custom Python script, and took one hour to complete. Response times for searching these *n*-grams is nearly immediate, resulting in very fast look-up times for users of our web application. The text content of the pages was also indexed, with just the OCR-transcribed text and the location of the line available in the index.

7.1.2.7 Web Application

The recognized and encoded *Liber usualis* is made available on the web using a custom-built web application to provide an interface for viewing and searching the OMR and OCR results of the document (figure 7.8). The web application serves HTML and JavaScript to the users, and manages the server-side components for the image viewer, including a system for communicating with the Solr index.

A modified version of the Diva document image viewer was used to provide an image-based search system. To provide search result highlighting on the image, we used the co-ordinates stored during the *n*-gram indexing to highlight the results of a pitch sequence search on the pages. Highlighting the search result on the image uses an HTML `<div/>` element overlaid over the region of the image. This element is positioned using the *n*-gram pixel-coordinate data stored in the Solr index, and is drawn using a slightly transparent colour as a way of highlighting the location on the image.

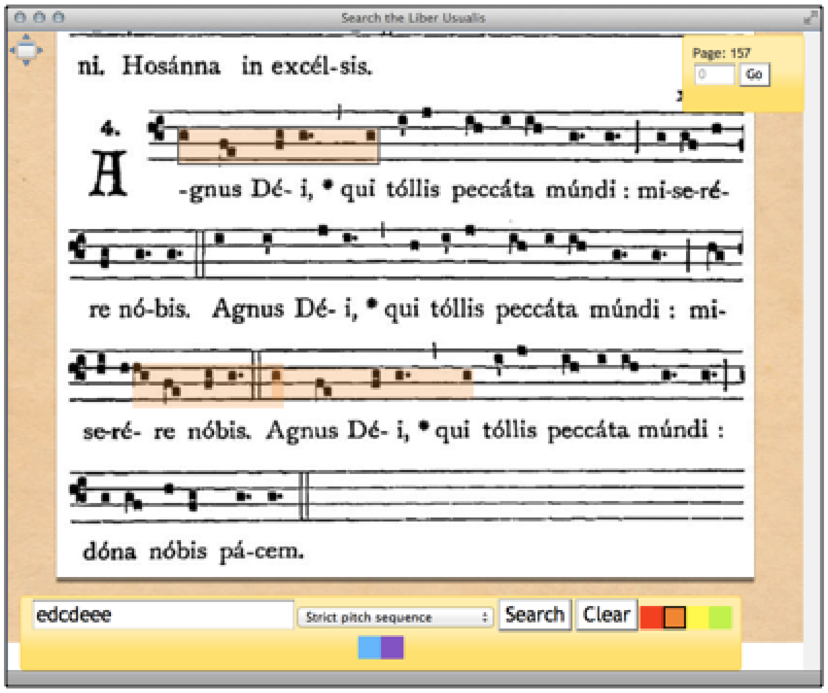

Search the Liber Usualis
Page: 157
Go
edcdeee
Strict pitch sequence
Search
Clear
ni. Hosánna in excél-sis.
4.
A -gnus Dé- i, * qui tóllis peccáta múndi : mi-se-ré-
re nó-bis. Agnus Dé- i, * qui tóllis peccáta múndi : mi-
se-ré- re nóbis. Agnus Dé- i, * qui tóllis peccáta múndi :
dóna nóbis pá-cem.

Figure 7.8: *Liber usualis* web application interface with a highlighted search query (“edcdeee”)

7.1.3 Discussion and Lessons Learned

The *Liber usualis* project demonstrated a need for a workflow system to manage and co-ordinate tools and people in a large-scale OMR project. The workflow, and the tools developed to support it, helped shape the direction of future research and development for later projects. Specifically, in this project we developed the techniques for MEI-based neume encoding and location capture, techniques for indexing and storing encoded information for search and retrieval, and the initial versions of our Diva, LibMEI, and pitch-finding tools.

The next two sections will discuss two phases of the “Salzinnes” project. These projects were created to further investigate image-based search and retrieval. The first project, Salzinnes I used metadata to provide a search and navigation interface; the second, Salzinnes II, used the Rodan system to provide a proof-of-concept symbolic notation search interface.

7.2 Salzinnes I

The first instance of the Salzinnes project was developed in the fall of 2011. The goal of this project was to synchronize the extensive data available for the Salzinnes Antiphonal in the CANTUS project (Koláček and Lacoste 2014a) with high-quality digital images of the original manuscript. The Salzinnes I project did not include notation search. The web application developed for this project is available at:

`http://ddmal.music.mcgill.ca/salzinnes/`

7.2.1 About the Manuscript

The *Salzinnes Antiphonal* (CDN-Hsmu M2149.L4) is a liturgical service book produced at the Abbey of Salzinnes in Belgium in 1554. It is now housed at St. Mary's University in Halifax, Nova Scotia, Canada. The antiphonal was described in detail and catalogued by Judith Dietz as part of her Masters in Arts degree (Dietz 2006). The cataloguing metadata was contributed to the CANTUS project, a database of ecclesiastical chant manuscripts and early printed sources.

7.2.2 Search System

The format of entries in the CANTUS database abbreviates the data stored in the fields, both to maintain consistency between entries as well as to efficiently utilize limited computer resources (the CANTUS database was first designed in 1987). This format is described by Koláček and Lacoste (2014b). In the Salzinnes I project these abbreviations were expanded to a more human readable form. These expansions were then indexed into a Solr search system. A sample record of one chant is shown in figure 7.9. In this example, the original value for the "genre" field, "A" has been expanded to a more readable value of "Antiphon."

```
<str name="cantusidnumber_t">2085</str>
<str name="caonumber_t">cao2085</str>
<arr name="concordances_strm">
  <str>I-IV 106 (Ivrea: Biblioteca Capitolare)</str>
  <str>CH-SGs 390-391 (Sankt Gallen: Stiftsbibliothek)</
str>
  <str>GB-Lbl add. 30850 (London: The British Library)</
str>
</arr>
<str name="feastcode_t">1011000</str>
<str name="feastname_t">Dom. 1 Adventus</str>
<str name="feastnameeng_s">1st Sunday of Advent</str>
<str name="feastnameeng_t">1st Sunday of Advent</str>
<str name="folio_t">001r</str>
<str name="fullmanuscripttext_t">Custodit dominus</str>
<str name="fullstandardtext_t">Custodit dominus omnes
diligentes se</str>
<str name="genre_t">Antiphon</str>
<str name="id">0001</str>
<str name="incipit_t">Custodit dominus*</str>
<str name="mode_t">No music</str>
<str name="office_t">First Vespers</str>
<str name="sequence_t">1</str>
<str name="siglum_t">CDN-Hsmu M2149.L4</str>
```

Figure 7.9: Sample Solr record for the Salzinnes chant interface

7.2.3 Image Processing

The images for the Salzinnes I interface were photographed by Judith Dietz. The images were delivered as JPEG files, and then converted to pyramid TIFF files for use by the IIP image server in a Diva viewer instance.

7.2.4 Web Application

A dedicated web application was written for this project using the Tornado web server and following the model developed in the *Liber usualis* project (figure 7.10). The browser interface uses a customized version of Diva. As the user scrolls the page images, this triggers an

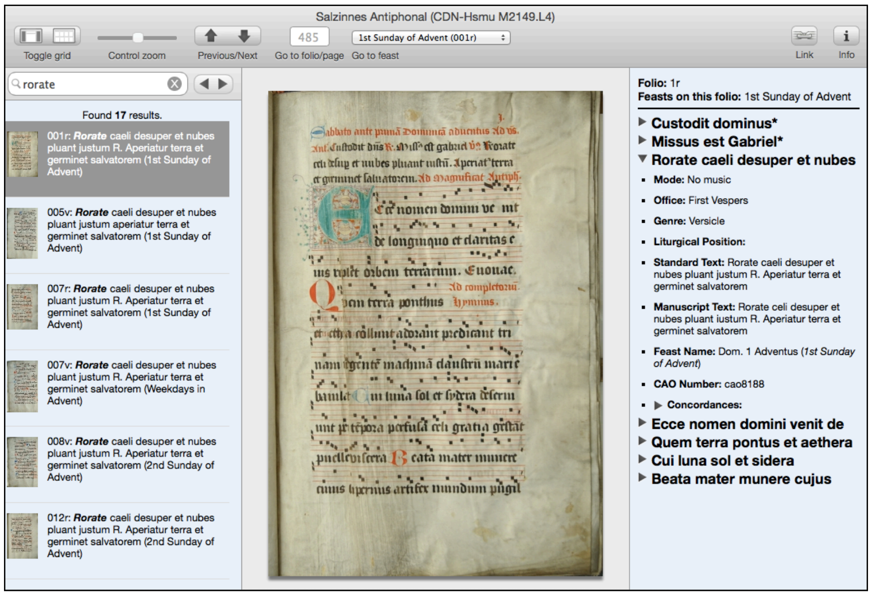
AJAX call to the Solr search system. The folio number of the visible page is used to query the Solr system for all chants on that page. The results of this are rendered in the interface and displayed to the user on the right-hand side of the page. The left-hand side of the page provides the user with a page-level metadata search system. The items in the list of search results are linked to the specific page and chant containing their query. Clicking on a result displays the corresponding page image in the centre, and the full record for the chant on the right. The fields available for full-text search are:

- Melodic Mode
- Liturgical Office
- Genre (e.g., Antiphon, Responsory)
- Liturgical Position (e.g., Antiphon for the Magnificat)
- Textual incipit
- Full text of the chant (standardized)
- Full text of the chant (as written in the source)
- Feast name
- CAO (*Corpus Antiphonalium Officii*) Number
- For illuminations, a short description or title

This manuscript is organized by chants for use on a specific liturgical feast day. Users could also choose a particular feast from the drop-down list of feasts catalogued in the manuscript. Selecting a particular feast displays the page image in the centre where the chants for that feast begin.

7.2.5 Discussion and Lessons Learned

The Salzinnes I project helped to further refine techniques for metadata-based page image retrieval. Although the CANTUS database contains extensive information for manuscripts at the folio and page level, this project was the first to experiment with synchronizing a page image representation with that information. This has allowed users to search and browse the manuscript images, seeing the results of their textual query in its original context on the document images.



Found **17** results.

001r: ***Rorate*** caeli desuper et nubes pluant justum R. Aperiatur terra et germinet salvatorem (1st Sunday of Advent)

005v: ***Rorate*** caeli desuper et nubes pluant justum aperiatur terra et germinet salvatorem (1st Sunday of Advent)

007r: ***Rorate*** caeli desuper et nubes pluant justum R. Aperiatur terra et germinet salvatorem (1st Sunday of Advent)

007v: ***Rorate*** caeli desuper et nubes pluant justum R. Aperiatur terra et germinet salvatorem (Weekdays in Advent)

008v: ***Rorate*** caeli desuper et nubes pluant justum R. Aperiatur terra et germinet salvatorem (2nd Sunday of Advent)

012r: ***Rorate*** caeli desuper et nubes pluant justum R. Aperiatur terra et germinet salvatorem (2nd Sunday of Advent)

Folio: 1r
Feasts on this folio: 1st Sunday of Advent

▶ **Custodit dominus***
▶ **Missus est Gabriel***
▼ **Rorate caeli desuper et nubes**

- **Mode:** No music
- **Office:** First Vespers
- **Genre:** Versicle
- **Liturgical Position:**
- **Standard Text:** Rorate caeli desuper et nubes pluant justum R. Aperiatur terra et germinet salvatorem
- **Manuscript Text:** Rorate celi desuper et nubes pluant iustum R. Aperiatur terra et germinet salvatorem
- **Feast Name:** Dom. 1 Adventus (*1st Sunday of Advent*)
- **CAO Number:** cao8188
- ▶ **Concordances:**

▶ **Ecce nomen domini venit de**
▶ **Quem terra pontus et aethera**
▶ **Cui luna sol et sidera**
▶ **Beata mater munere cujus**

Figure 7.10: Search and browse interface for the Salzinnes I project

7.3 Salzinnes II

The Salzinnes II project was a further refinement of the methods developed in the Salzinnes I project and in the *Liber usualis* project. This project was the first to use the Rodan workflow system to provide OMR-based notation search results on the manuscript. The results of the Salzinnes II project are available at:

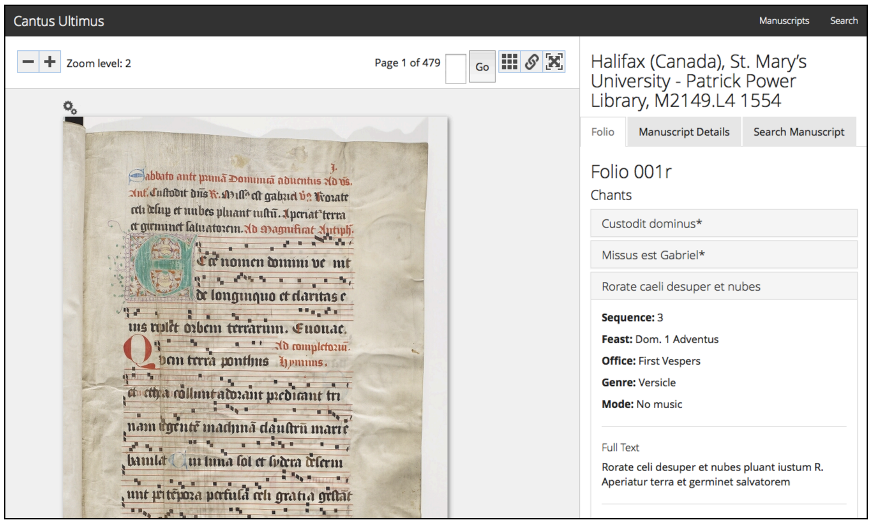
`http://cantus.simssa.ca/manuscript/133/`

As part of conservation and restoration efforts on this manuscript it was brought to the Canadian Conservation Institute (CCI) in Ottawa, Ontario for high-resolution digitization. They provided copies of the TIFF files produced in this imaging process. There are a total of 480 page images each approximately the same file size for a total document size of 82GB (table 7.2).

Like the Salzinnes I project, the image files were integrated into a web application that featured an instance of the Diva image viewer (figure 7.11). The CANTUS data from the previous Salzinnes project was indexed into a new Solr instance. As users scroll the images in the Diva interface, the document metadata updates to display detailed descriptions of the page content. Users can also search and retrieve pages using the same field indexes.

Size (pixels)	6993 x 4414
Resolution	~300 PPI
Bit Depth	16
Color Model	RGB
Size (MB)	~175MB

Table 7.2: Salzinnes image characteristics as digitized by the Canadian Conservation Institute



Halifax (Canada), St. Mary's University - Patrick Power Library, M2149.L4 1554

Folio 001r
Chants
Custodit dominus*
Missus est Gabriel*
Rorate caeli desuper et nubes

Sequence: 3
Feast: Dom. 1 Adventus
Office: First Vespers
Genre: Versicle
Mode: No music

Full Text
Rorate celi desuper et nubes pluant iustum R. Aperiatur terra et germinet salvatorem

Figure 7.11: The Salzinnes II interface

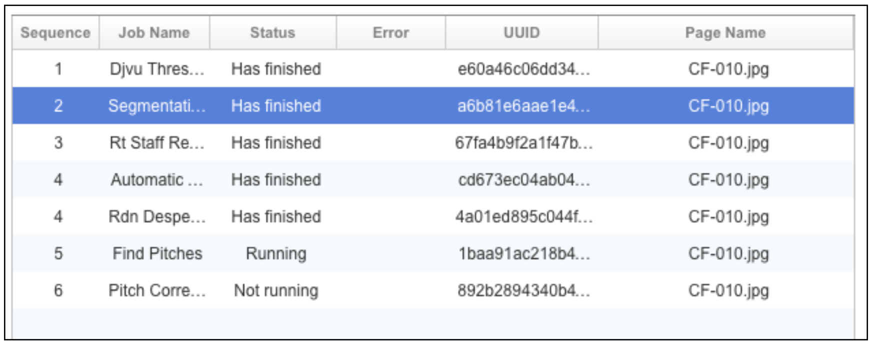
However, this instance of the Salzinnes viewer also features music notation data from an OMR process. This brought the same mode of interaction for searching and retrieving music notation demonstrated in the *Liber usualis* project into the Salzinnes interface. The process of performing recognition is described in the next section.

7.3.1 Integration of OMR Data

The Rodan workflow system was used to develop a custom OMR workflow for analyzing the Salzinnes document images (figure 7.12). The following Rodan jobs were used to perform this recognition:

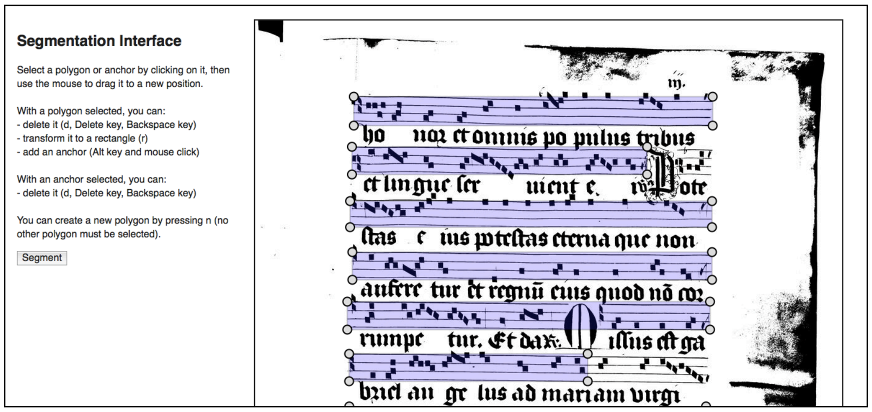
1. **DjVu Threshold** (Non-interactive job). Binarization of the colour images. This was drawn directly from the Gamera toolkit.
2. **Segmentation** (Interactive job). Separate the staff regions from the non-staff (e.g., text, illustrations) regions. An initial solution was calculated automatically, and users were asked to correct this using an interactive JavaScript interface (figure 7.13). This job used the Musicstaves toolkit to determine the staff region locations, and then presented the detected staff regions to the user to correct using a JavaScript interface.
3. **RT (Roach & Tatum) Staff Removal** (Non-interactive job). Uses the Roach and Tatem (1988) approach to remove the staff lines from the image. This was drawn directly from the Gamera Musicstaves toolkit.
4. **Rdn ("Rodan") Despeckle** (Non-interactive job). Removes small "noise" particles from the binarized image to clean the image for the connected-component classifier. This used the Gamera despeckle functionality, but modified for use in the Rodan workflow system.
5. **Automatic classification** (Non-interactive job). A pre-built Gamera classifier was provided to the system, built using the neume from several pages of the Salzinnes.
6. **Find Pitches** (Non-interactive job). Uses the approach described by Vigliensoni et al. (2011) to re-introduce musical semantics to the classified shapes. This used elements of Gamera, but was primarily developed as a custom job for this particular purpose.

7. **Pitch Correction** (Interactive job). Used the Neon neume editor to allow users to correct the results of the automatic classification (figure 7.14).



Sequence	Job Name	Status	Error	UUID	Page Name
1	Djvu Thres...	Has finished		e60a46c06dd34...	CF-010.jpg
2	Segmentati...	Has finished		a6b81e6aae1e4...	CF-010.jpg
3	Rt Staff Re...	Has finished		67fa4b9f2a1f47b...	CF-010.jpg
4	Automatic ...	Has finished		cd673ec04ab04...	CF-010.jpg
4	Rdn Despe...	Has finished		4a01ed895c044f...	CF-010.jpg
5	Find Pitches	Running		1baa91ac218b4...	CF-010.jpg
6	Pitch Corre...	Not running		892b2894340b4...	CF-010.jpg

Figure 7.12: A workflow in Rodan processing a Salzinnes page image



Segmentation Interface

Select a polygon or anchor by clicking on it, then use the mouse to drag it to a new position.

With a polygon selected, you can:
- delete it (d, Delete key, Backspace key)
- transform it to a rectangle (r)
- add an anchor (Alt key and mouse click)

With an anchor selected, you can:
- delete it (d, Delete key, Backspace key)

You can create a new polygon by pressing n (no other polygon must be selected).

Figure 7.13: Interactive Segmentation Interface in Rodan

In total, the OMR results for 20 pages were obtained using this process. The MEI files were indexed in Solr using the same methods as those used in the *Liber usualis* project. As of this writing the remaining pages have not been processed due to time and human resource constraints; however, as they become available they will be loaded into the search system.

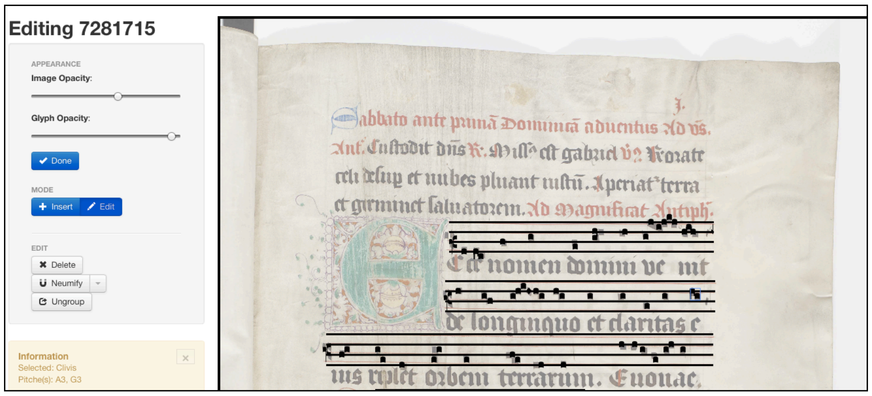


Figure 7.14: A Salzinnes page corrected in the Neon interface

7.3.2 Discussion and Lessons Learned

The Salzinnes II project was the culmination of the tools, techniques, and processes developed in the course of this dissertation. The tools and techniques developed in the previous two projects were further refined and generalized to support different document types. Specifically, the Rodan workflow system was demonstrated to be an effective approach to building a distributed and collaborative OMR system.

In the course of building the infrastructure for the Salzinnes II project, several of the tools developed for the earlier projects were enhanced to improve performance, reliability, or useability. LibMEI library (§6.1) was re-written from the original Python prototype to C++ to improve parsing speed, the Neon.js correction interface (§6.5.1) was developed to fill the need for a visual method of correcting OMR results, and several interactive JavaScript-based image manipulation tools (mentioned in §5.2.5) were developed. In addition, enhancements were made to the Diva document viewer software to allow others to take advantage of the image highlighting capabilities without requiring a customized version of the software.

The Salzinnes II project is the first publicly-available effort to provide document image navigation for music *manuscripts* using spatially aligned search results. Using the systems designed in the course of this project we feel it is now possible to generalize our approach to cover recognition for a wide range of document types, from the earliest manuscripts to modern-day printed sources.

7.4 Chapter Summary

This chapter has discussed three projects developed to demonstrated document image-based search and retrieval. The *Liber usualis* and Salzinnes II projects have demonstrated that it is possible to create systems similar to those available for textual document retrieval that align queries with their spatial location on digital page images. The Salzinnes I project was an important stepping stone between these two projects, and provided a means to refine metadata-based techniques for page image search and retrieval. The combination of both content and page-level metadata search provides users with a comprehensive way of accessing the data contained within a manuscript by searching and navigating large collections of music document images.

8. Conclusions, Contributions, and Future Work

The dissertation discusses ways to design optical music recognition systems for use in large-scale music document digitization initiatives. Chapter 2 discussed the emergence of image-based search and retrieval of textual document images using spatially aligned optical character recognition (OCR). This was used as a point of reference throughout the dissertation to illustrate several key points: that the emergence of this technique was a crucial development for enabling large-scale text recognition by mitigating the effects of imperfect OCR systems; that spatially aligned transcriptions provide users of these documents with a means of navigating large image collections; and, most importantly, to underline the point that a common and pervasive technique in large-scale OCR initiatives has not been adopted by optical music recognition (OMR) systems. OMR initiatives still rely on a purely transcriptive approach to document image recognition, separating their transcriptions from the original page image. This has the effect of requiring a "perfect" transcription of the document—an impossible requirement given the scope and scale of the task.

Chapter 3 provided an overview of OMR systems and techniques. OMR was framed as a process (§3.3) composed of several steps. While there is no universal agreement on the order of the steps, or on the tools and techniques to be used in each step, every OMR system incorporates aspects of image processing, machine learning and classification, music palaeography, music theory, symbolic notation encoding, and several other techniques, chained together to form a process. Other aspects of OMR research were presented, including a discussion of evaluation techniques for OMR (§3.4), previous attempts at large-scale music document recognition initiatives (§3.5), server-based OMR systems (§3.6), and OMR systems for historical, non-CWMN music notation (§3.7).

Chapter 4 presented the proposals central to the dissertation, introducing techniques for scaling OMR systems to process large numbers of music document images. A discussion of scientific workflow systems (§4.2) was proposed as an alternative approach to creating OMR systems capable of large-scale data processing, enabling the creation of bespoke OMR systems for specific repertoires, projects, document images, or automated evaluation tasks. The chapter then presented three new design patterns for OMR systems to meet the challenges presented by large-scale digitization initiatives: Distributed OMR (§4.3), which uses geographically and computationally distributed individuals and computers to accomplish OMR tasks; Collaborative OMR (§4.4), which enables distributed and heterogeneous groups to work together on OMR tasks; and Networked Adaptive OMR (§4.5), which uses distributed, collaborative OMR systems as a constant source of training and evaluation data to develop and deploy adaptive OMR systems shared by all users of a network. The Music Encoding Initiative (MEI) was introduced (§4.6) as a symbolic music encoding system with several unique characteristics making it the most suitable format for use in a large-scale music document recognition context.

Chapter 5 introduced Rodan, a web-based workflow system designed following the principles set out in Chapter 4. Rodan is developed following the model of the scientific workflow system. Unlike scientific workflow software packages, however, Rodan features methods for interacting with a running workflow, allowing humans to participate in the overall recognition process. Rodan is designed to execute OMR processes on remote server systems, accessible through a web browser.

Chapter 6 introduced several tools developed in the course of the research program to support large-scale OMR. A C++ and Python library, LibMEI, was presented (§6.1) as a software tool for reading and writing MEI files. The Diva document image viewer was described (§6.2) as a way to present and navigate high-resolution document image collections in a web browser. The neume encoding system used for the *Liber usualis* project was presented (§6.3), which also served as a discussion of how to customize MEI schema for different notation repertoires. The OMR Interchange Package was described (§6.4) as a technique for capturing and sharing OMR results between software systems. Finally, several interfaces for web-based document correction were presented (§6.5), demonstrating first steps towards a crowdsourcing approach to distributed and collaborative OMR.

Chapter 7 described three prototype projects developed as technology demonstrations. The *Liber usualis* project (§7.1) represented the first attempt at image-based OMR navigation over a large collection of document images. The Salzinnes I project (§7.2) focused on creating a system for navigating a 16th century chant manuscript using extensive metadata provided by the CANTUS project. Finally, the Salzinnes II project (§7.3) incorporated the CANTUS metadata and several pages of OMR-derived chant notation into a single interface for music score searching.

8.1 Summary of Contributions

The goal of the dissertation is to provide new ways of designing, building, and interacting with OMR software systems as means of building searchable digital collections. In the course of the dissertation, several novel contributions were developed to meet this goal.

8.1.1 Techniques for aligning images and OMR results

The dissertation takes as its starting point the hypothesis that aligning recognized text and images was a catalyst for the creation of large-scale OCR initiatives, starting with small prototype projects (§2.2), and resulting in application in global mass digitization efforts such as Google Books (§2.4). The most significant contribution of the dissertation, therefore, may be the proposal that a similar technique is required to enable OMR on the same scale. Previous attempts at large-scale OMR have not used the alignment technique and required complete and accurate human-supplied corrections to arrive at a useable representation of a music document—a labour-intensive and costly process. Aligning images with automatically transcribed notation gives the user of a hypothetical search and retrieval system a way to navigate the collection using the OMR-transcribed document contents, but presented using the original page images thereby presenting the user with a readable representation of the document despite any underlying recognition errors. Alignment of transcribed music notation and page images may prove to be a similar catalyst for enabling large-scale music document recognition by presenting an interface where the documents can be readable without requiring completely accurate transcriptions.

8.1.2 History of Alignment in Text Recognition

As far as could be determined, no comprehensive history of text and image alignment in OCR initiatives, such as was provided in chapter 2 of the dissertation, has been written. Given the pervasiveness of this technique in modern text digitization and recognition initiatives, this dissertation contributes a significant amount of historical research, tracing its development from the earliest recognition initiatives through to the present day. This may be of interest to researchers in fields beyond music document recognition.

8.1.3 Music document recognition workflows

Scientific workflow software has been tested and deployed for text-based documents, most notably in the IMPACT project (§4.2.2.1).

Chapter 4 of the dissertation presents a novel proposal for interactive, workflow-based OMR systems, which allow users to create bespoke OMR systems customized to a specific document type and composed of heterogeneous collections of remotely-hosted and executed document recognition tools. The distinction between interactive and non-interactive workflow jobs is also unique among all workflow systems, as are the web-based image manipulation interfaces used to perform interactive jobs in Rodan.

8.1.4 Distributed, collaborative, and networked adaptive OMR

The formalization of three design patterns of distributed, collaborative, and networked adaptive OMR is a unique contribution of this dissertation, and will serve to guide how new OMR systems are designed and implemented. First, distributed OMR (§4.3) proposes that new OMR systems can be designed as web applications, which have several advantages over traditional desktop-based OMR systems. Distributed OMR systems provide separation between the user interface and the underlying implementation of the OMR system, allowing multiple users to interact with a single, remote OMR installation. Distributed and parallel computing techniques, operating on clusters of remote computers, can provide greater computing resources, allowing the OMR process to accommodate larger workloads. Secondly, collaborative OMR (§4.4) proposes that geographically distributed users can work together on a central task. This can take place either as part of an explicit collaboration, in which individuals know and understand their role in the overall process, or as an implicit collaboration where individuals may only be given a small and well-defined task. The latter collaboration allows crowdsourcing, where globally distributed networks of people provide many small contributions that, in aggregate, contribute significantly to a project. Finally, networked adaptive OMR (§4.5) is proposed as a technique that uses the data provided by distributed and collaborative OMR systems to build large sets of "ground-truth" data suitable for training and improving adaptive OMR systems. Corrections and improvements to recognition can be distributed back to a network, allowing the contributions of one individual to have an impact on the recognition accuracy of all other members of the network.

8.1.5 Web-based OMR

While several web-based OMR systems have previously been proposed (§3.6), the system described in this dissertation, Rodan (Chapter 5), represents the first comprehensive approach to a fully interactive web-based OMR system.

8.1.6 Supporting technologies

Several of the tools developed to support the development of large-scale OMR and described in Chapter 6 are novel contributions in their own right.

8.1.6.1 Contributions to the Music Encoding Initiative (MEI)

Novel contributions to the MEI community include LibMEI software (§6.1), specifically its techniques for automatically deriving a strongly-typed library from the One Document Does-it-all (ODD) definition of MEI (§6.1.2). In the future this approach could be expanded to work with the Text Encoding Initiative (TEI). In addition, the SibMEI plugin (§6.1.3) uses the same design principles to enable MEI output from the Sibelius notation software. The OMR Interchange Package format (§6.4), based on MEI and the proposed BagIt packaging standard, represent first attempts at creating a common interchange format for OMR systems and for software that uses the output of OMR systems.

8.1.6.2 Diva

The Diva document viewer (§6.2) employs several novel techniques for presenting large document images in a web browser. The most distinguishing characteristic is the interface which combines scrolling through pages and zooming to see the highest-quality version of an image all in the same interface—no other document viewer integrates both scrolling and zooming to the same extent. This is accomplished by assembling document images using tiles, allowing Diva to request and display only the portion of a page image that the user is currently viewing. The integration of client-side image manipulation tools is also unique among document image viewers.

8.1.6.3 Crowdsourcing and OMR

While crowdsourcing has successfully been used by text recognition initiatives to allow large numbers of people to contribute time and energy to data correction and quality control, it is only starting to be used for music recognition. This dissertation attempts to introduce the use of crowdsourcing for OMR, with preliminary attempts to determine how crowdsourcing may be used in a music recognition context (§4.4). Several browser-based prototype interfaces (§5.2.5, §6.5) for collecting music correction contributions from individuals using web-based tools are also presented.

8.2 Future work

The most obvious avenue for future work is to continue developing new techniques for OMR that can be integrated into customizable OMR workflows. This will allow users, rather than developers, to create and deploy OMR systems tailored to a particular use. While this dissertation has focused primarily on the use of OMR for complete document image transcription, the scientific workflow system model for constructing bespoke OMR systems offers several opportunities for extracting new types of information from music document page images. For example, OMR developers can build workflows to automatically evaluate and compare tools and techniques, fulfilling one of the most pressing needs in OMR systems development. Already, Rodan has been used to extract and synchronize multiple images of a single piece by automatically identifying and aligning measures from different editions of the same piece (Vigliensoni et al. 2013). Specialized workflows can be used to automatically identify and classify the use of different hands in a manuscript source. In a document recognition context, workflow systems hold possibilities for working with and manipulating document images in unique and exciting ways.

8.2.1 SIMSSA

The Single Interface for Music Score Searching and Analysis (SIMSSA) project is a seven-year funded project to develop the tools and techniques described in this dissertation. The SIMSSA project is structured along two principle axes, *Content* and *Analysis*, designed to address issues relating to both music document recognition (Content) as well as music document search, retrieval and analysis (Analysis). Under SIMSSA, Rodan and many of the other tools mentioned in this dissertation will continue to be developed.

8.2.2 Crowdsourced OMR

Crowdsourcing is, by its nature, dependent on large-scale deployment and adoption. At the time of writing, crowdsourcing interfaces for symbolic music are in very early stages of development. Future work in this area should study how to attract and engage masses of users to contribute corrections to large-scale music document recognition initiatives and include studies on user interfaces for these tasks.

8.2.3 Continuous machine learning

The networked adaptive OMR proposed in the dissertation may provide new ways to design adaptive recognition systems. A networked OMR system capable of collecting a constant stream of correction data from a distributed user base, and of using this data to continuously train and improve multiple recognition systems, is a new concept that deserves attention from software developers and computer scientists in many different fields.

8.2.4 Towards Search and Retrieval

This dissertation has not discussed methods for search and retrieval, but has instead focused on technologies for *enabling* large-scale search and retrieval. In this section however, I will attempt speculate on how we can move forward with creating search and retrieval interfaces based on the output of large-scale OMR initiatives.

At present, it is not obvious how anyone will interact with a system that contains millions of transcribed musical documents. For music, there is only so much structure a person can be reasonably expected to provide a search system by using ASCII characters in a text field. Instead, I see a future where people interact with a large musical document corpus in ways that differ from text-based search. Models of style, genre, geography, composer, and countless other descriptions may be derived from the notation itself, allowing a search system to match user-provided descriptions with the underlying symbolic data.

It is my experience that people expect a notation-based retrieval system to provide access to a nearly unlimited number of possible musical dimensions. Pitch and rhythm are currently the most explicit dimensions for retrieval and, as such, are the dimensions used by most search interfaces. Beyond pitch and rhythm, however, additional dimensions such as timbre, form, harmonic function, counterpoint, texture, or *tessitura* may also provide valuable dimensions by which musical works may be retrieved, and which may be automatically derived from the notation content itself. Systems that can automatically derive searchable representations based on a limitless number of possible dimensions will be required, and will pose challenges to our current methods of abstracting and pre-indexing music documents.

8.3 Conclusion

Large-scale music recognition is the only viable means of converting the vast back-catalogues of printed music into a form that is immediately usable in a search context. This dissertation imagines a future where physical collections will have digital counterparts that do not replace the originals but complement them, offering exciting new modes of interacting with our global music collections.

Appendix A: Open Source Projects

This appendix provides a reference to the open source projects and source code written for this dissertation. Each project contains a complete contribution and revision history. A listing of contributions is provided in the acknowledgements of the dissertation.

Full Projects

These projects are full software systems.

Rodan

`https://github.com/DDMAL/Rodan`

Rodan Client

`https://github.com/DDMAL/RodanClient`

Rodan Gamera Plugins

`https://github.com/DDMAL/rodan_plugins`

Diva.js

`https://github.com/DDMAL/diva.js`

LibMEI

`https://github.com/DDMAL/libmei`

```
NB: An earlier Python-only version is archived at
https://github.com/ahankinson/pymei
```

SibMEI

`https://github.com/DuChemin/sibmei`

PyBagIt

`https://github.com/ahankinson/pybagit`

Prototypes

These projects are prototypes or experimental software systems, but may provide implementations of particular methods or processes which others may find useful.

JS Image Suite

`https://github.com/DDMAL/js-image-suite`

Solesmes MEI Customization

`https://github.com/music-encoding/mei-incubator/tree/master/solesmes`

Neon.js

`https://github.com/DDMAL/Neon.js`

The Gamera AOMR Toolkit

`https://github.com/DDMAL/aOMR`

Web Applications

These web applications have been developed for specific projects. They require a database and/or a Solr back-end containing the data for the application and cannot be redeployed by others, but may still provide useful information for specific technology implementations.

The Liber usualis Web Application

`https://github.com/DDMAL/liberusualis`

The Salzinnes (I) Web Application

`https://github.com/DDMAL/salzinnes-original`

The Salzinnes (II) Web Application

`https://github.com/DDMAL/cantus`

Project Contributions

In addition to the projects initiated as part of this dissertation, several existing open source projects were contributed to in the course of this work.

Gamera

`http://gamera.informatik.hsnr.de/index.html`

Cappuccino

`http://www.cappuccino-project.org`

Aruspix

`http://www.aruspix.net`

The Music Encoding Initiative

`http://music-encoding.org/home`

The Gamera Document Preprocessing Toolkit

`https://github.com/DDMAL/document-preprocessing-toolkit`

Appendix B: Rodan REST API

Characteristics

All resources are identified using plurals for collections, and singular for elements. For example, `http://example.com/projects/` will return a list of all projects, while `http://example.com/project/:id` will return a single project that matches the given ID.

All resources are identified using a UUID. This is to reduce the possibility of collisions, as well as make it easier to uniquely identify resources without relying on serial database keys.

All resources may be browsed at `http://example.com/browse/`

Four HTTP verbs are used by Rodan: GET, POST, PATCH, DELETE. The PUT method is not supported in favour of the PATCH method, since PUT requires a complete (changed) record while PATCH simply requires the fields that have been updated.

Errors are returned to the user in the form of HTTP Status Codes. *You must not* return an error message with a non-error code. (i.e., do not return an error with a status code of `200 OK`.)

At present, the serial form of the data must be JSON-encoded. Content type and Return Type are negotiated via HTTP Headers, not via query parameters. (i.e., you must send a message with an `Accepts: application/json` header).

Django models often map API resources, but they do not have a 1-to-1 mapping.

Some query parameter values will be processed if they are set, regardless of their value. For example, `by_page=true` will have the same effect as `by_page=false`, since the check is only whether the `by_page` parameter is set or not. Check the documentation for indications of where this may apply.

Authentication

Rodan implements two types of server authentication: Session-based and Token-based. Session-based authentication is most appropriate for browsers, where the session information is stored as a cookie. Token-based is useful for scripted and non-browser-based interaction.

Token Authentication

A token is created when your user account is created. To retrieve your token, you can send a request to the Rodan server with your username and password, e.g.,:

```
$> curl -v -XPOST -d username=$USERNAME -d
password=$PASSWORD http://example.com/auth/token/
```

This will respond with:

```
{"token": "655aff7dc865866fc9bd9e7fafb32bfeb484365a"}
```

This may then be used for requests to the Rodan server via the `Authorization` header:

```
curl -XGET -H "Authorization: Token
655aff7dc865866fc9bd9e7fafb32bfeb484365a" http:/
/localhost:8000/
```

Session Authentication

Currently the Rodan client web application uses Session authentication. If you are not authenticated, you will be asked to log in to the Rodan web application when you first visit. This will set two cookies, `crsftoken` and `sessionid`. These are persistent in your browsing session. "Logging out" will effectively delete these cookies, and require re-authentication on your next visit.

If you are browsing the JSON API in your browser, your session authentication credentials will be used for this. If you attempt to visit a URL and are not authenticated, you will be denied access.

Resources

All endpoints support the OPTIONS and HEAD methods. Although the server currently replies indicating support for the PUT method, this should not be used.

Projects

Collection

URI: `/projects/`

Returns a list of projects that the user has permissions to view. Accepts a POST request with a data body to create a new project. POST requests will return the newly-created project object.

- Methods Supported: GET, POST
- Permissions: Authenticated

Element

URI: `/project/$ID/`

Performs operations on a single project instance.

- Methods Supported: GET, PATCH, DELETE
- Permissions: Authenticated

Workflows

Collection

URI: `/workflows/`

Returns a list of all workflows. Accepts a POST request with a data body to create a new workflow. POST requests will return the newly-created workflow object.

- Supported Query Parameters:
  - `project=$ID`: Retrieve workflows belonging to project `$ID`.
- Methods supported: GET, POST
- Permissions: Authenticated

Example:

This returns the workflows associated with the supplied project ID.

```
$> curl -XGET http://localhost:8000/workflows/
?project=201e13914eb14e1ba08c7e55b3b
```

Element

URI: `/workflow/$ID/`

Performs operations on a single workflow instance.

- Methods Supported: GET, PATCH, DELETE
- Permissions: Authenticated

Jobs

Collection

URI: `/jobs/`

Returns a list of all workflows. Does not accept `POST` requests, since Jobs should be defined and loaded server-side.

- Supported Query Parameters:
    - `enabled=true`: Filter the list of available jobs by their enabled/disabled status.
- Methods Supported: GET
- Permissions: Read-only (public)

Element

URI: `/job/$ID/`

Returns a single instance of a Job. Read-only.

- Methods Supported: GET
- Permissions: Read-only (public)

Pages

Collection

URI: `/pages/`

Returns a list of all pages. Accepts a POST request with a data body to create a new page. POST requests will return the newly-created workflow object. Page image data should be sent with the `Content-type: multipart/form-data` header.

- Supported Query Parameters:
  - `project=$ID`: Retrieve pages belonging to project `$ID`
- Methods Supported: GET, POST
- Permissions: Authenticated

Element

URI: `/page/$ID/`

Performs operations on a single page instance.

- Methods Supported: GET, PATCH, DELETE
- Permissions: Authenticated

Workflow Jobs

Collection

URI: `/workflowjobs/`

Returns a list of all workflows jobs. Accepts a POST request with a data body to create a new workflow job. POST requests will return the newly-created workflow job object.

- Supported Query Parameters:
  - `workflow=$ID`: Retrieve workflow jobs belonging to workflow `$ID`.
- * Methods supported: GET, POST
- * Permissions: Authenticated

Element

URI: `/workflowjob/$ID/`

Performs operations on a single workflow job instance.

- Methods Supported: GET, PATCH, DELETE
- Permissions: Authenticated

Workflow Runs

Collection

URI: `/workflowruns/`

Returns a list of all workflows runs. Accepts a POST request with a data body to create a new workflow run. POST requests will return the newly-created workflow run object.

Creating a new `WorkflowRun` resource starts the workflow. When a WorkflowRun instance is created, Rodan will cycle through all the pages attached to the workflow, and apply all of the WorkflowJobs assigned to this workflow to each page. Each WorkflowJob is converted to a RunJob instance and the run is stored in the database.

This method also supports running a workflow in Test mode (see Supported Query Parameters below). Running a workflow in test mode requires a single page ID to operate on. Each non-test run of a workflow will increase the `runs` value on the Workflow model.

- Supported Query Parameters:
  - `workflow=$ID`: Retrieves all runs for workflow `$ID`. (GET only)
  - `run=int`: Retrieves a single run for a workflow. Most useful if combined with the `workflow` parameter. (GET only)
  - `test=true`: Sets whether this is a test run or not. (POST only)
  - `page_id=$ID`: If this is a test run, you **must** supply a page ID to test the workflow on. (POST only)
- Methods supported: GET, POST
- Permissions: Authenticated

Element

URI: `/workflowrun/$ID/`

Performs operations on a single workflow run instance.

- Supported Query Parameters:

    - `by_page=true`: If true, re-formats the returned workflow run object by returning it based on page-and-results, rather than runjob-and-page.
    - `include_results=true|false`: Sets whether to include the results (per page) for the workflow run (GET only; only checked if `by_page` is present).
- Methods Supported: GET, PATCH, DELETE
- Permissions: Authenticated

Run Jobs

Collection

URI: `/runjobs/`

Returns a list of all run jobs. Run Jobs are created by the server when a workflow is executed, so this does not accept POST requests.

- Supported Query Parameters:
    - `requires_interaction=true`: Sets whether only those Run Jobs that currently require interaction will be returned or not (GET only).
    - `project=$ID`: Retrieve runjobs belonging to project $ID (GET only).
    - `workflowrun=$ID`: Retrieve runjobs belonging to workflow run $ID (GET only).
    - `page=$ID`: Retrieve runjobs belonging to page $ID (GET only).
- Methods Supported: GET
- Permissions: Authenticated

Element

URI: `/runjob/$ID/`

Performs operations on a single Run Job instance. Does not support the DELETE method.

- Methods Supported: GET, PATCH
- Permissions: Authenticated

Users

Collection

URI: /users/

Returns a list of users. Accepts POST requests to create new users. A successful POST request will return the newly created user object.

- Methods Supported: GET, POST
- Permissions: Administrator Only

Element

URI: /user/$ID/

Performs operations on a single user record.

- Methods Supported: GET, PATCH, DELETE
- Permissions: Administrator Only

Bibliography

Abbyy. 2014. ABBYY Finereader. http://finereader.abbyy.com (accessed 24 March 2014).

Abrahams, D., and R. Grosse-Kunstleve. 2003. Building hybrid systems with boost.python. *C/C++ Users Journal* 21 (7).

Advanced Technology Libraries. 2005. Consortium forms OCA to bring additional content online. *Advanced Technology Libraries* 34 (11): 1, 9–10.

Agnew, C., P. Baran, D. Caulkins, V. Cerf, R. Crane, P. Goldstein, and E. Parker. 1974. ARPANET management study: New application areas. Advanced Research Projects Agency Second Quarterly Technical Report. CableData Associates Inc.

Allen, M. 1987. Optical character recognition: Technology with new relevance for archival automation projects. *The American Archivist* 50 (1): 88–99.

Amazon.com. 2013. Amazon web services: About us. http://aws.amazon.com/about-aws/ (accessed 7 March 2014).

Anderson, J., J. Lehnardt, and N. Slater. 2010. *CouchDB: The definitive guide.* Sebastapol, CA: O'Reilly.

Anderson, N. 2010. Optical character recognition: IMPACT briefing paper. IMPACT Project. https://www.digitisation.eu/fileadmin/user_upload/240/docbook/media/fd9fd1c8-cd6a-5324-65fb-54d836b9ed84.pdf (accessed 24 June 2014).

Andre, P., and N. Eaton. 1988. National agricultural text digitizing project. *Library Hi Tech News* 6 (23): 61–6.

Andre, P., N. Eaton, and J. Zidar. 1988. Scanning and digitizing technology employed in the national agricultural text digitizing project. In *Proceedings of the Conference on Application of Scanning Methodologies for Libraries.* Beltsville, MD, 17–18 November, 61–73.

Apple Computer. 2012. Model-view-controller. *Concepts in Objective-C Programming*. https://developer.apple.com/library/mac/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html#//apple_ref/doc/uid/TP40010810-CH14 (accessed 15 March 2014).

Arms, W. 2000. Automated digital libraries: How effectively can computers be used for the skilled tasks of professional librarianship? *D-Lib Magazine* 6 (7/8). http://www.dlib.org/dlib/july00/arms/07arms.html (accessed 10 March 2014).

Arnold, J., R. Badger, and R. Lucier. 1997. Red Sage final report. http://www.librarytechnology.org/ltg-displaytext.pl?RC=9762 (accessed 13 January 2014).

Atkinson, R., and L. Stackpole. 1995. TORPEDO: Networked access to full-text and page-image representations of physics journals and technical reports. *The Public-Access Computer Systems Review* 6 (3): 6–15.

Ávila, B., and R. Lins. 2004. Efficient removal of noisy borders from monochromatic documents. In *Proceedings of the International Conference on Image Analysis and Recognition.* Porto, Portugal, 29 September–1 October, 249–56.

Baggi, D., and G. Haus. 2009. IEEE 1599: Music encoding and interaction. *Computer* 42 (3): 84–7.

Bainbridge, D. 1997. Extensible optical music recognition. Ph.D diss., University of Canterbury.

———. 2000. The role of music IR in the New Zealand digital library project. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Plymouth, MA, 23–25 October,

Bainbridge, D., and T. Bell. 1996. An extensible optical music recognition system. *Australian Computer Science Communications* 18 (1): 308–17.

———. 2001. The challenge of optical music recognition. *Computers and the Humanities* 35 (2): 95–121.

———. 2003. A music notation construction engine for optical music recognition. *Software: Practice and Experience* 33 (2): 173–200.

Bainbridge, D., C. Nevill-Manning, I. Witten, L. Smith, and R. McNab. 1999. Towards a digital library of popular music. In *Proceedings of the The Fourth ACM Conference on Digital Libraries.* Berkeley, CA, 11–14 August, 161–9.

Bainbridge, D., and K. Wijaya. 1999. Bulk processing of optically scanned music. In *Proceedings of the International Conference on Image Processing and its Applications.* Manchester, UK, 13–15 July, 474–8.

Balk, H., and L. Ploeger. 2009. Impact: Working together to address the challenges involving mass digitization of historical printed text. *OCLC Systems and Services* 25 (4): 233–48.

Balm, G. 1970. An introduction to optical character reader considerations. *Pattern Recognition* 2: 151–66.

Barton, L. 2002. The NEUMES project: Digital transcription of Medieval chant manuscripts. In *Proceedings of the Web Delivering of Music.* Darmstadt, Germany, 9–11 December, 211–8.

Bellini, P., I. Bruno, and P. Nesi. 2007. Assessing optical music recognition tools. *Computer Music Journal* 31 (1): 68–93.

———. 2008. Optical music recognition: Architecture and algorithms. In *Interactive Multimedia Music Technologies,* edited by K. Ng, and P. Nesi, 80–110. Hershey, PA: Information Science Reference.

Benedictines of Solesmes. 1961. *The Liber usualis, with introduction and rubrics in english.* Tournai, Belgium: Desclée.

Bent, M. 1994. Editing early music: The dilemma of translation. *Early Music* 22 (3): 373–92.

Bergeron, K. 1998. *Decadent enchantments: The revival of Gregorian chant at Solesmes.* Berkeley, CA: University of California Press.

Berners-Lee, T., R. Cailliau, J.-F. Groff, and B. Pollermann. 1992. World-wide web: The information universe. *Internet Research* 2 (1): 52–8.

Bitteur, H. 2013. Audiveris: Optical music recognition. Presentation given at the Music Hack Day, Vienna, Austria, http://www.mdw.ac.at/mdwMediathek/ClassicalMusicHackDay/AudiverisMusicHackDayVienna2013.v1.pdf (accessed 14 February 2014).

———. 2014. Audiveris: Open music scanner. https://audiveris.kenai.com (accessed 11 February 2014).

Blamberg, D., C. Dowling, and C. Weston, eds. 1988. *Proceedings of the Conference on Application of Scanning Methodologies for Libraries.* Washington, DC: National Agricultural Library.

Blanke, T., M. Bryant, and M. Hedges. 2012. Open source optical character recognition for historical research. *Journal of Documentation* 68 (5): 659–83.

Blostein, D., and H. S. Baird. 1992. A critical survey of music image analysis. In *Structured Document Image Analysis,* edited by H. Baird, H. Bunke, and K. Yamamoto, 405–34. Berlin: Springer.

Boddie, S. 2009. What's METS/ALTO and should you care? http://www.dlconsulting.com/metadata/whats-mets-alto-and-should-you-care/ (accessed 12 January 2014).

Bonte, T., N. Froment, and H. Bitteur. 2013. MuseScore and Audiveris as a service. Presentation given at the Music Hack Day, Vienna, Austria, http://wwwyoutube.com/watch?v=a-OdfNsov_U (accessed 12 February 2014).

Boyko, A., J. Kunze, J. Littman, L. Madden, and B. Vargas. 2014. The BagIt file packaging format (v0.97). https://tools.ietf.org/html/draft-kunze-bagit-10. (accessed 23 March 2014).

Braid, A. 2003. The use of electronic journals in a document delivery service. *Serials* 16 (1): 37–40.

Brandt, S. 2002. Java tip 128: Create a quick-and-dirty XML parser. http://www.javaworld.com/article/2077493/mobile-java/java-tip-128--create-a-quick-and-dirty-xml-parser.html (accessed 26 July 2014).

Bray, T., D. Hollander, A. Layman, R. Tobin. 2006a. Namespaces in XML 1.1. 2nd ed. http://www.w3.org/TR/xml-names11 (accessed 9 December 2014).

Bray, T., J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan. 2006b. Extensible markup language. http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf (accessed 4 April 2014).

Breuel, T. 2007. The hOCR microformat for OCR workflow and results. In *Proceedings of the International Conference on Document Analysis and Recognition.* Curitiba, Brazil, 23–26 September, 1063–7.

———. 2008. The OCRopus open source OCR system. *Proceedings of the SPIE: Document Recognition and Retrieval* 6815.

———. 2009. Recent progress on the OCRopus OCR system. In *Proceedings of the International Workshop on Multilingual OCR.* Barcelona, Spain, 25 July, 1–10.

———. 2010. The hOCR embedded OCR workflow and output format. https://docs.google.com/document/d/1QQnIQtvdAC_8n92-LhwPcjtAUFwBlzE8EWnKAxlgVf0/preview (accessed 9 January 2014).

Brink, A., and N. Pendock. 1996. Minimum cross-entropy threshold selection. *Pattern Recognition* 29 (1): 179–88.

Bronner, E. 1999. You can look it up, hopefully. *The New York Times.* http://www.nytimes.com/1999/01/10/weekinreview/ideas-trends-you-can-look-it-up-hopefully.html (accessed 24 January 2014).

Bugge, E., K. Juncher, B. Mathiasen, and J. Simonsen. 2011. Using sequence alignment and voting to improve optical music recognition from multiple recognizers. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Miami, FL, 24–28 October, 405–10.

Burgoyne, J. A., J. Devaney, L. Pugin, and I. Fujinaga. 2008. Enhanced bleedthrough correction for early music documents with recto-verso registration. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Philadelphia, PA, 14–18 September, 407–12.

Burgoyne, J. A., Y. Ouyang, T. Himmelman, J. Devaney, L. Pugin, and I. Fujinaga. 2009. Lyric extraction and recognition on digital images of early music sources. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Kobe, Japan, 26–30 October, 723–7.

Burgoyne, J. A., L. Pugin, G. Eustace, and I. Fujinaga. 2007. A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Vienna, Austria, 23–27 September, 509–12.

Burlet, G., A. Porter, A. Hankinson, and I Fujinaga. 2012. Neon.js: Neume editor online. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Porto, Portugal, 8–12 October, 121–6.

Burnard, L., and S. Rahtz. 2004. RelaxNG with Son of ODD. In *Proceedings of the Extreme Markup Languages.* Montréal, QC, 2–6 August, Burton-West, T. 2009. Slow queries and common words (part 1). *HathiTrust: Large-scale Search Blog*. http://www.hathitrust.org/blogs/large-scale-search/slow-queries-and-common-words-part-1 (accessed 11 January 2014).

Bush, V. 1945. As we may think. *The Atlantic* 176 (1): 101–8.

Byrd, D., W. Guerin, M. Schindele, and I. Knopke. 2010. OMR evaluation and prospects for improved OMR via multiple recognizers. http://www.informatics.indiana.edu/donbyrd/MROMR2010Pap/OMREvaluation+Prospects4MROMR.doc (accessed 22 February 2014).

Byrd, D., and M. Schindele. 2006. Prospects for improving OMR with multiple recognizers. In *Proceedings of the Conference of the International Soceity for Music Information Retrieval*. Victoria, BC, 8–12 October, 41–6.

Byrd, D., and J. Simonsen. 2013. Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. Indiana University. Working Paper. http://www.informatics.indiana.edu/donbyrd/OMRTestbed/OMRStandardTestbed1Mar2013.pdf (accessed 13 June 2014).

Caldas Pinto, J., P. Vieira, M. Ramalho, M. Mengucci, P. Pina, and F. Muge. 2000. Ancient music recovery for digital libraries. In *Proceedings of the European Conference on Digital Libraries*. Lisbon, Portugal, 18–20 September, 24–34.

Canons Regular of Saint John Cantius. 2010. Liber usualis. http://www.sanctamissa.org/en/music/gregorian-chant/choir/liber-usualis-1961.html (accessed 13 July 2014).

Cao, G., L. Dou, Q. Hart, and B. Ludaescher. 2011. Kepler/g-pack: A kepler package using the Google cloud for interactive scientific workflows. In *Proceedings of the Ninth Biennial Ptolemy Miniconference*. Berkeley, CA, 16 February,

Capela, A., J. Cardoso, A. Rebelo, and C. Guedes. 2008. Integrated recognition system for music scores. In *Proceedings of the International Computer Music Conference*. Belfast, Northern Ireland, 24–29 August,

Capella Software. 2014. Info capella-scan. http://www.capella.de/us/index.cfm/products/capella-scan/info-capella-scan/ (accessed 8 July 2014).

Cappuccino Project. 2014a. Learning Objective-J. http://www.cappuccino-project.org/learn/objective-j.html (accessed 27 March 2014).

———. 2014b. What is Cappuccino? http://www.cappuccino-project.org/learn/ (accessed 27 March 2014).

Carlson, S., and J. Young. 2005. Yahoo works with academic libraries on a new project to digitize books. *The Chronicle of Higher Education* 52 (8): A34. https://chronicle.com/article/Yahoo-Works-With-Academic/28449 (accessed 16 January 2014).

Carter, N., R. Bacon, and T. Messenger. 1988. The acquisition, representation and reconstruction of printed music by computer: A review. *Computers and the Humanities* 22 (2): 117–36.

Carter, N. P. 1992. Segmentation and preliminary recognition of madrigals notated in white mensural notation. *Machine Vision and Applications* 5 (3): 223–30.

Casey, R., and G. Nagy. 1991. Document analysis: A broader view. In *Proceedings of the International Conference on Document Analysis and Recognition.* Saint-Malo, France, 30 September–2 October, 839–49.

Castan, G., M. Good, and P. Roland. 2001. Extensible markup language (XML) for music applications: An introduction. *Computing in Musicology* 12: 95–102.

Castro, P., R. Almeida, and J. Caldas Pinto. 2008. Restoration of double-sided ancient music documents with bleed-through. In *Proceedings of the Iberoamerican Congress on Pattern Recognition.* Valparaiso, Chile, 13–16 November, 940–9.

Castro, P., and J. Caldas Pinto. 2007. Methods for written ancient music restoration. In *Proceedings of the Image Analysis and Recognition.* Montreal, Canada, 22–24 August, 1194–205.

Chapman, K. 2001. An examination of the usefulness of JSTOR to researchers in finance. *Behavioral & Social Sciences Librarian* 19 (2): 39–47.

Chitu, A. 2006. The new Google Book Search. http://googlesystem.blogspot.ca/2006/11/new-google-book-search.html (accessed 14 January 2014).

Choudhury, G., T. DiLauro, M. Droettboom, I. Fujinaga, B. Harrington, and K. MacMillan. 2000a. Optical music recognition system within a large-scale digitization project. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Plymouth, MA, 23–25 October,

Choudhury, G., T. Dilauro, M. Droettboom, I. Fujinaga, and K. Macmillan. 2001. Strike up the score: Deriving searchable and playable digital formats from sheet music. *D-Lib Magazine* 7. http://dlib.org/dlib/february01/choudhury/02choudhury.html (accessed 30 January 2014).

Choudhury, G., C. Requardt, I. Fujinaga, T. DiLauro, E. W. Brown, J. W. Warner, and B. Harrington. 2000b. Digital workflow management: The Lester s. Levy digitized collection of sheet music. *First Monday* 5 (6). http://firstmonday.org/ojs/index.php/fm/article/view/756 (accessed 30 January 2014).

Christenson, H. 2011. HathiTrust: A research library at web scale. *Library Resources & Technical Services* 55 (2): 93–102.

Cleverdon, C., and M. Keen. 1966. Factors determining the performance of indexing systems. Aslib Cranfield Research Project. 2.

Cleverdon, C., J. Mills, and M. Keen. 1966. Factors determining the performance of indexing systems. Aslib Cranfield Research Project. 1.

Code Synthesis Tools. 2014. Code Synthesis XSD. http://www.codesynthesis.com/projects/xsd/ (accessed 18 July 2014).

Compaq Computer Corporation. 1996. Internet solutions division strategy for cloud computing. http://www.technologyreview.com/sites/default/files/legacy/compaq_cst_1996_0.pdf (accessed 7 March 2014).

Compier, H., and R. Campbell. 1992. ADONIS—latest developments and its role in the changing publisher/library relationship. *Health Libraries Review* 9: 3–13.

Connelly, D., B. Paddock, and R. Harvey. 1999. XDOC data format: Technical specification. ScanSoft Inc. Version 4.0.

Conway, P. 2010. Measuring content quality in a preservation repository: HathiTrust and large-scale book digitization. In *Proceedings of the International Conference on Preservation of Digital Objects.* Vienna, Austria, 19–24 September, 95–102.

Cookson, J. 1984. A demonstration database for document images. *Proceedings of the Society of Photo-Optical Instrumentation Engineers* 0515: 30–41.

Coüasnon, B., and J. Camillerapp. 1995. A way to separate knowledge from program in structured document analysis: Application to optical music recognition. *International Conference on Document Analysis and Recognition* : 1092–7.

Cover, T., and P. Hart. 1967. Nearest neighbour classification. *IEEE Transactions of Information Theory* 13 (1): 21–7.

Coyle, K. 2006. Mass digitization of books. *The Journal of Academic Librarianship* 32 (6): 641–5.

Crane, G. 2006. What do you do with a million books? *D-Lib Magazine* 12 (3). http://www.dlib.org/dlib/march06/crane/03crane.html (accessed 15 January 2014).

Crawford, T. 1991. Applications involving tablatures. *Computing in Musicology* 7: 57–9.

Cui, J., H. He, and Y. Wang. 2010. An adaptive staff line removal in music score images. In *Proceedings of the International Conference on Signal Processing.* Beijing, China, 24–28 October, 964–7.

Curcin, V., and M. Ghanem. 2008. Scientific workflow systems-can one size fit all? In *Proceedings of the Biomedical Engineering Conference.* Innsbruck, Austria, 13–14 February, 1–9.

Dalitz, C., and T. Crawford. 2013. From facsimile to content based retrieval: The electronic corpus of lute music. *Phoibos - Zeitschrift für Zupfmusik* 2: 167–85.

Dalitz, C., M. Droettboom, B. Pranzas, and I. Fujinaga. 2008a. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30 (5): 753–66.

Dalitz, C., and T. Karsten. 2005. Using the Gamera framework for building a lute tablature recognition system. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* London, UK, 11–15 September, 478–81.

Dalitz, C., G. Michalakis, and C. Pranzas. 2008b. Optical recognition of psaltic Byzantine chant notation. *International Journal on Document Analysis and Recognition* 11 (3): 143–58.

Dalitz, C., and B. Pranzas. 2009. German lute tablature recognition. In *Proceedings of the 10th International Conference on Document Analysis and Recognition.* Barcelona, Spain, 371–5.

Damm, D., C. Fremerey, V. Thomas, M. Clausen, F. Kurth, and M. Müller. 2012. A digital library framework for heterogeneous music collections: From document acquisition to cross-modal interaction. *International Journal on Digital Libraries* 12 (2-3): 53–71.

Dannenberg, R. 1993. Music representation issues, techniques, and systems. *Computer Music Journal* 17 (3): 20–30.

Deegan, M., E. King, and E. Steinvil. 2001. Project report: British Library microfilmed newspapers and Oxford grey literature online. The British Library, London.

DeLoughry, T. 1996. Journal articles dating back 100 years are being put on line. *The Chronicle of Higher Education* 43 (15): A30, A32.

der Knijff, J. 2011. JPEG 2000 for long-term preservation: JP2 as a preservation format. *D-Lib Magazine* 17 (5). http://dlib.org/dlib/may11/vanderknijff/05vanderknijff.html (accessed 22 July 2014).

Desaedeleer, A. 2006. Reading sheet music. MSc diss., Imperial College, University of London.

Dietz, J. 2006. Centuries of silence: The discovery of the Salzinnes Antiphonal. MA diss., St. Mary's University.

Django Project. 2014. Django. https://www.djangoproject.com (accessed 27 March 2014).

Doermann, D., J. Liang, and H. Li. 2003. Progress in camera-based document image analysis. In *Proceedings of the International Conference on Document Analysis and Recognition.* Edinburgh, UK, 3–6 August, 606–16.

Dogan, M., C. Neudecker, S. Schlarb, and G. Zechmeister. 2010. Experimental workflow development in digitisation. In *Proceedings of the Second International Conference on Qualitative and Quantitative Methods in Libraries.* Chania, Crete, Greece, 25–28 May, 377–84.

Dollar, C., and W. Hooton. 1984. Technology assessment report: Speech pattern recognition, optical character recognition, digital raster scanning. National Archives and Records Administration Service. PB125217/AS.

Dori, D., D. Doermann, C. Shin, R. Haralick, I. Phillips, M. Buchman, and D. Ross. 1997. The representation of document structure: A generic object-process analysis. In *Handbook on Optical Character Recognition and Document Image Analysis*, 421–56. Singapore: World Scientific.

Downie, J. S. 1999. Evaluating a simple approach to music information retrieval: Conceiving melodic *n*-grams as text. PhD diss., University of Western Ontario.

———. 2012. Introduction to the HathiTrust research center: A briefing. Presentation given at the The Faculty of Information & Media Studies, University of Western Ontario, http://www.hathitrust.org/documents/HTRC-UWO-201212.pptx (accessed 24 June 2014).

Droettboom, M., and I. Fujinaga. 2004. Symbol-level groundtruthing environment for OMR. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Barcelona, Spain, 10–14 October, 497–500.

Droettboom, M., I. Fujinaga, and K. MacMillan. 2009. Optical music interpretation. In *Proceedings of the IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition.* Windsor, Ontario, 6–9 August, 378–87.

Droettboom, M., K. MacMillan, and I. Fujinaga. 2003. The Gamera framework for building custom recognition systems. In *Proceedings of the Symposium on Document Image Understanding Technologies.* Greenbelt, MD, 9-11 April, 275–86.

Drummond, D. 2008. New chapter for Google Book Search. http://googleblog.blogspot.ca/2008/10/new-chapter-for-google-book-search.html (accessed 16 January 2014).

Drummond, J., and M. Bosma. 1989. A review of low-cost scanners. *International Journal of Geographical Information Systems* 3 (1): 83–95.

Dumitrescu, T., and M. Berchum. 2009. The cmme occo codex edition: Variants and versions in encoding and interface. In *Digitale edition zwischen experiment und standardisierung*, edited by P. Stadler, and J. Veit, 129–46.

Dutta, A., Pal. U., A. Fornés, and J. Lladós. 2010. An efficient staff removal approach from printed musical documents. In *Proceedings of the Twentieth IAPR International Conference on Pattern Recognition.* Istanbul, Turkey, 23–26 August, 1965–8.

ECMA International. 2013. The JSON data interchange format. ECMA-404. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf. (accessed 27 March 2014).

English, W., D. Engelbart, and M. Berman. 1967. Display-selection techniques for text manipulation. *IEEE Transactions on Human Factors in Electronics* 8 (1): 5–15.

Entlich, R. 2002. Where are they now? Digitizing microfilmed newspapers. *RLG Diginews* 6 (3).

Entlich, R., J. Olsen, L. Garson, M. Lesk, L. Normore, and S. Weibel. 1997. Making a digital library: The contents of the CORE project. *ACM Transactions on Information Systems* 15 (2): 103–23.

Erickson, R. 1975. The DARMS project: A status report. *Computers and the Humanities* 9 (6): 291–8.

Exolab Group. 2013. The Castor project. http://castor.codehaus.org (accessed 18 July 2014).

Farrow, G., C. Xydeas, and J. Oakley. 1994. Conversion of scanned documents to the open document architecture. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*. Adelaide, Australia, 19–22 April, 109–12.

Ferrand, M., J. A. Leite, and A. Cardoso. 1999. Hypothetical reasoning: An application to optical music recognition. In *Proceedings of the Joint Conference on Declarative Programming*. Aquila, Italy, 6–9 September, 367–81.

Fielding, R. 2001. Architectural styles and the design of network-based software architectures. PhD diss., University of California, Irvine.

Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. 1999. Hypertext transfer protocol: HTTP/1.1. RFC 2616. http://www.w3.org/Protocols/rfc2616/rfc2616.html. (accessed 6 March 2014).

Finn, J., J. Larcombe, Y. Assam, S. Whiteside, M. Copperwhite, P. Walmsley, G. Westlake, and M. Eastwood. 2011. Sibelius 7: Using the ManuScript language. Avid Technology. Burlington, MA. http://www.sibelius.com/download/documentation/pdfs/sibelius710-manuscript-en.pdf (accessed 18 July 2014).

Fornés, A., A. Dutta, A. Gordo, and J. Lladós. 2011a. CVC-MUSCIMA: A ground truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition* 15 (3): 243–51.

Fornés, A., A. Dutta, A. Gordo, and J. Llados. 2011b. The ICDAR 2011 music scores competition: Staff removal and writer identification. In *Proceedings of the International Conference on Document Analysis and Recognition*. Beijing, China, 18–21 September, 1511–5.

Fornés, A., A. Dutta, A. Gordo, and J. Lladós. 2013. The 2012 music scores competitions: Staff removal and writer identification. In *Proceedings of the International Conference on Graphics Recognition*. Seoul, Korea, 15–16 September, 173–86.

Freeman, E., K. Sierra, and B. Bates. 2004. *Head first design patterns*. Sebastopol, CA: O'Reilly.

Fremerey, C., M. Müller, F. Kurth, and M. Clausen. 2008. Automatic mapping of scanned sheet music to audio recordings. In *Proceedings of the Conference of the International Society for Music Information Retrieval*. Philadelphia, PA, 14–18 September, 413–8.

Fujinaga, I. 1996a. Adaptive optical music recognition. PhD diss., McGill University.

———. 1996b. Exemplar-based learning in adaptive optical music recognition system. In *Proceedings of the International Computer Music Conference*. Hong Kong, 55–6.

———. 2004. Staff detection and removal. In *Visual Perception of Music Notation: Online and Offline Recognition*, edited by S. George, 1–39. Hershey, PA: IRM Press.

Fujinaga, I., B. Alphonce, and B. Pennycook. 1989. Issues in the design of an optical music recognition system. In *Proceedings of the International Computer Music Conference*. Columbus, OH, 2–5 November, 113–6.

Fujinaga, I., B. Alphonce, B. Pennycook, and K. Hogan. 1991. Optical music recognition: Progress report. In *Proceedings of the International Computer Music Conference*. Montreal, QC, 66–73.

Galloway, P. 1981. Hardware review: KDEM. *Computers and the Humanities* 15 (3): 183–5.

Gamera. 2012. Training tutorial. http://gamera.sourceforge.net/doc/html/training_tutorial.html (accessed 25 March 2014).

Garrett, J. 2005. AJAX: A new approach to web applications. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/ (accessed 6 March 2014).

Gauvain, J.-L., and C.-H. Lee. 1994. Maximum a posteriori estimation for multivariate gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing* 2 (2): 291–8.

Gear Up AB. 2014. Iseenotes. http://www.iseenotes.com).

George, S. 2004. Lyric recognition and Christian music. In *Visual Perception of Music Notation: Online and Offline Recognition*, edited by S. George, 198–226. Hershey, PA: IRM Press.

Gezerlis, V., and S. Theodoridis. 2002. Optical character recognition of the orthodox Hellenic Byzantine music notation. *Pattern Recognition* 35 (4): 895–914.

Gil, Y. 2007. Workflow composition: Semantic representations for flexible automation. In *Workflows for e-Science,* edited by I. Taylor, E. Deelman, D. Gannon, and M. Shields, 244–57. London: Springer.

Goble, C., and D. De Roure. 2007. MyExperiment: Social networking for workflow-using e-scientists. In *Proceedings of the Workshop on Workflows in Support of Large-scale Science.* Monterey Bay, CA, 25 June, 1–2.

Goldfarb, C., and Y. Rubinsky. 1990. *The sgml handbook.* Oxford, UK: Oxford University Press.

Good, M. 2001. MusicXML for notation and analysis. *Computing in Musicology* 12: 113–24.

———. 2009. Using MusicXML 2.0 for music editorial applications. In *Digitale edition zwischen experiment und standardisierung,* edited by P. Stadler, and J. Veit, 157–74. Tübingen: Max Niemeyer.

Google. 2004. Google checks out library books. http://googlepress.blogspot.ca/2004/12/google-checks-out-library-books.html (accessed 15 January 2014).

———. 2014a. Library partners. http://books.google.com/googlebooks/library/partners.html (accessed 15 January 2014).

———. 2014b. What you'll see when you search on Google Books. http://books.google.com/googlebooks/library/screenshots.html (accessed 15 January 2014).

Gorn, S., R. Bemer, and J. Green. 1963. American standard code for information interchange. *Communications of the ACM* 6 (8): 422–6.

Grande, C., and A. Belkin. 1996. The development of the notation interchange file format. *Computer Music Journal* 20 (4): 33–46.

Grant, J. 2005. Judging book search by its cover. *Google Official Blog.* http://googleblog.blogspot.ca/2005/11/judging-book-search-by-its-cover.html (accessed 16 January 2014).

Grant, S. 1994. ADONIS: For developing countries? In *Proceedings of the International Federation of Library Associations Conference.* Havana, Cuba, 21–27 August, 217–25.

Gray, P. 1986. Desktop publishing. *Evaluation Practice* 7 (3): 40–9.

Gudgin, M., M. Hadley, N. Mendelsohn, J. Moreau, H. Nielsen, A. Karmakar, and Y. Lafon. 2007. Soap version 1.2 part 1: Messaging framework. http://www.w3.org/TR/soap12-part1/ (accessed 31 March 2014).

Guthrie, K. 1999. JSTOR: Large scale digitization of journals in the United States. *LIBER Quarterly* 9 (3): 291–7.

Haffner, P., L. Bottou, P. Howard, P. Simard, Y. Bengio, and Y. Le Cun. 1998. Browsing through high quality document images with DjVu. In *Proceedings of the International Forum on Research and Technology Advances in Digital Libraries*. Santa Barbara, CA, 22–24 April, 309–18.

Hafner, K. 2005. In challenge to Google, Yahoo will scan books. *New York Times*, 3 October. http://www.nytimes.com/2005/10/03/business/03yahoo.html?_r=2&scp=2&sq=open%20content%20alliance&st=cse&oref=slogin& (accessed 16 June 2014).

Hahn, T. 2008. Mass digitization: Implications for preserving the scholarly record. *Library Resources & Technical Services* 52 (1): 18–26.

Handel, P. 1931. Statistical machine. US Patent 1,915,993, filed 27 April 1931, and issued 27 June 1933.

Hankinson, A. 2013. Pybagit. https://github.com/ahankinson/pybagit (accessed 23 March 2014).

Hankinson, A., W. Liu, L. Pugin, and I. Fujinaga. 2011a. Diva.js: A continuous document image viewing interface. *Code4lib Journal* 14. http://journal.code4lib.org/articles/5418 (accessed 17 June 2014).

———. 2012. Diva: A web-based high-resolution document image viewer. In *Proceedings of the Theory and Practice of Digital Libraries*. Paphos, Cyprus, 23–27 September, 455–60.

Hankinson, A., and A. Porter. 2014. LibMEI. http://github.com/DDMAL/libmei (accessed 27 March 2014).

Hankinson, A., L. Pugin, and I. Fujinaga. 2009. Interfaces for document representation in digital music libraries. In *Proceedings of the Conference of the International Society for Music Information Retrieval*. Kobe, Japan, 26–30 October, 39–44.

———. 2010. An interchange format for optical music recognition applications. In *Proceedings of the Conference of the International Society for Music Information Retrieval*. Utrecht, The Netherlands, 9–13 August, 51–6.

Hankinson, A., P. Roland, and I. Fujinaga. 2011b. The Music Encoding Initiative as a document encoding framework. In *Proceedings of the Conference of the International Society for Music Information Retrieval*. Miami, FL, 24–28 October, 293–8.

Hankinson, A., and M. Walter. 2014. SibMEI. https://github.com/DuChemin/SibMEI (accessed 27 March 2014).

Haring, D., and J. Roberge. 1969. A combined display for computer generated data and scanned photographic images. In *Proceedings of the Spring Joint Conference of the American Federation of Information Processing Societies*. Boston, MA, 14–16 May, 483–90.

Hart, M. 1992. The history and philosophy of Project Gutenberg. http:/ /www.gutenberg.org/wiki/ Gutenberg:The_History_and_Philosophy_of_Project_Gutenberg_by_Michael _Hart (accessed 14 January 2014).

HathiTrust. 2012. The HathiTrust digital library. http://www.hathitrust.org/ (accessed 9 January 2014).

———. 2014. Datasets. (accessed 19 January 2014).

Hayes, R. 1988. Concluding address. In *Proceedings of the Conference on Application of Scanning Methodologies for Libraries*. Beltsville, MD, 17–18 November, 135–40.

Helsen, K. 2011. 'venite et vidite': First results in the optical neume recognition project. In *Proceedings of the Cantus Planus*. Vienna, Austria, 26 September,

———. 2013. The evolution of neumes into square notation in chant manuscripts. *Journal of the Alamire Foundation* 5 (2): 14374.

Henderson, B. E. 1983. Prototype for an electronic document storage and retrieval program. *Proceedings of the Society of Photo-Optical Instrumentation Engineers* 0418: 112–5.

Hewlett, W. 1997. MuseData: Multipurpose representation. In *Beyond MIDI: The Handbook of Musical Codes*, 402–47. Cambridge, MA: The MIT Press.

Hillegass, A., and A. Preble. 2011. *Cocoa programming for Mac OS X*. Boston: Addison-Wesley.

Hockey, S. 1986. OCR: The Kurzweil data entry machine. *Literary and Linguistic Computing* 1 (2): 63–7.

Hoffman, M., L. O'Gorman, G. Story, J. Arnold, and N. Macdonald. 1993. The RightPages™ service: An image-based electronic library. *Journal of the American Society for Information Science* 44 (8): 446–52.

Holland, M. 2008. Historical British newspapers online. *Library Hi Tech News* 7.

Holley, R. 2009a. How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine* . http:/ /www.dlib.org/dlib/march09/holley/03holley.html (accessed 16 June 2014).

———. 2009b. Many hands make light work: Public collaborative OCR text correction in Australian historic newspapers. National Library of Australia Staff Papers. http://www-prod.nla.gov.au/openpublish/index.php/nlasp/ article/viewArticle/1406 (accessed 7 January 2014).

Holmes, W. 1988. Comparison of scanning methodologies for conversion of typed, printed, handwritten, and microfilmed materials. In *Proceedings of the Conference on Application of Scanning Methodologies for Libraries*. Beltsville, MD, 17–18 November, 25–33.

Hopmann, A. 2007. The story of XMLHTTP. http://www.alexhopmann.com/xmlhttp.htm (accessed 6 March 2014).

Howard, J. 2012. Google begins to scale back its scanning of books from University libraries. *The Chronicle of Higher Education.* http://chronicle.com/article/Google-Begins-to-Scale-Back/131109/ (accessed 16 January 2014).

Huang, X., and K. Lee. 1993. On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition. *IEEE Transactions on Speech and Audio Processing* 1 (2): 150–7.

Hull, D., K. Wolstencroft, R. Stevens, C Goble, M. Pocock, P. Li, and T. Oinn. 2006. Taverna: A tool for building and running workflows of services. *Nucleic Acids Research* 34: 729–32.

Huron, D. 1997. Humdrum and kern: Selective feature encoding. In *Beyond MIDI: The Handbook of Musical Codes,* 375–401. Cambridge, MA: The MIT Press.

IMPACT Project. 2014. Overview on available tools for text digitisation. http://www.digitisation.eu/tools-survey/index-coc (accessed 7 April 2014).

International Standards Organization. 1995. Standard music description language. ISO 10743.

———. 2008. Document management — portable document format — part 1: PDF 1.7. 32000-1. http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf. (accessed 28 March 2014).

Internet Archive. 2014a. Digitizing print collections with The Internet Archive. http://archive.org/scanning (accessed 16 January 2014).

———. 2014b. Universal library. https://archive.org/details/universallibrary (accessed 18 January 2014).

Johansen, L. 2009. Optical music recognition. MSc diss., University of Oslo.

Jones, G. 2008a. OMR engine output file format. http://www.visiv.co.uk/tech-mro.htm (accessed 4 April 2014).

———. 2008b. Sharpeye music scanning. http://www.visiv.co.uk (accessed 23 February 2014).

Jones, G., B. Ong, I. Bruno, and K. Ng. 2008. Optical music imaging: Music document digitisation, recognition, evaluation and restoration. In *Interactive Multimedia Music Technologies,* edited by K. Ng, and P. Nesi, 50–79. Hershey, PA: Information Science Reference.

Jones, S. 1953. Machine that can read type face described as an office time-saver. *New York Times,* 26 December. 18.

JPEG Group. 2014. Jpeg 2000: Our new standard! http://www.jpeg.org/jpeg2000/ (accessed 21 July 2014).

jQuery Foundation. 2014. JQuery. http://jquery.com (accessed 25 March 2014).

Kahle, B. 2009. Economics of book digitization. http://www.opencontentalliance.org/2009/03/22/economics-of-book-digitization/ (accessed 10 March 2014).

Kassler, M. 1972. Optical character recognition of printed music: A review of two dissertations. *Perspectives of New Music* 11: 250–4.

King, E. 2005. Digitisation of newspapers at the British Library. *The Serials Librarian* 49 (1–2): 165–81.

Kirkpatrick, D. 2003. Amazon plan would allow searching texts of many books. *The New York Times*. http://www.nytimes.com/2003/07/21/business/amazon-plan-would-allow-searching-texts-of-many-books.html (accessed 9 January 2014).

Kirstein, P., and G. Montasser-Kohsari. 1995. The c-oda project: Experiences and tools. *The Computer Journal* 38 (8): 670–80.

Kleiner, A., and R. Kurzweil. 1977. A description of the Kurzweil reading machine and a status report on its testing and dissemination. *Bulletin of Prosthesis Research* 10 (27): 72–81.

Kluzner, V., A. Tzadok, Y. Shimony, E. Walach, and A. Antonacopoulos. 2009. Word-based adaptive OCR for historical books. In *Proceedings of the 10th International Conference on Document Analysis and Recognition*. Barcelona, Spain, 26–29 July, 501–5.

Knudson, D., and S. Teicher. 1969. Remote text access in a computerized library information retrieval system. In *Proceedings of the Spring Joint Conference of the American Federation of Information Processing Societies*. Boston, MA, 14–16 May, 475–81.

Knuth, D. 1984. Literate programming. *The Computer Journal* 27 (2): 97–111.

Koláček, J., and D. Lacoste. 2014a. CANTUS: A database for Latin ecclesiastical chant. http://cantusdatabase.org (accessed 18 July 2014).

———. 2014b. CANTUS: Field contents. http://cantusdatabase.org/description (accessed 1 April 2014).

Kostka, S. M. 1971. Recent developments in computer-assisted musical scholarship. *Computers and the Humanities* 6 (1): 15–21.

Kraft, B. 1989. Pricing of Optiram's OCR service. *Humanist Discussion Group*. http://dhhumanist.org/Archives/Virginia/v03/0367.html (accessed 13 January 2014).

Krasner, G., and S. Pope. 1988. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming* 1 (3): 26–49.

Kuć, R., and M. Rogoziński. 2013. *Elasticsearch server*. Birmingham, UK: Packt Publishing.

Kurth, F., D. Damm, C. Fremerey, M. Müller, and M. Clausen. 2008. A framework for managing multimodal digitized music collections. In *Proceedings of the European Conference on Digital Libraries*. Aarhus, Denmark, 14–19 September, 334–45.

Kurzweil, R. 2013. Keynote speech: Lawtech futures conference. http://wwwyoutube.com/watch?v=zcUxmRvjxM8 (accessed 7 January 2013).

Lagoze, C., E. Shaw, J. Davis, and D. Krafft. 1995. Dienst: Implementation reference manual. Computer Science Department, Cornell University, Ithaca, NY.

Lee, D., and R. Smith. 2012. Improving book OCR by adaptive language and image models. In *Proceedings of the 10th IAPR International Workshop on Document Analysis Systems*. Gold Coast, Australia, 27–29 March, 115–9.

Leedham, G., S. Varma, A. Patankar, V. Govindarayu, and D. De Roure. 2002. Separating text and background in degraded document images – a comparison of global threshholding techniques for multi-stage thresholding. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*. Buffalo, NY, 6–8 August, 244–9.

Lesk, M. 1990a. Full text retrieval with graphics. In *Proceedings of the Bridging the Gap: Technical Information Panel Specialists' Meeting*. Trondheim, Norway, 5–6 September, 5-1–13.

———. 1990b. Image formats for preservation and access. A report of the Technology Assessment Advisory Committee to the Commission on Preservation and Access. Council on Library and Information Resources. http://www.clir.org/pubs/reports/pub5/lesk.html/lesk.html#digpaper (accessed 16 June 2014).

———. 1994. Experiments on access to digital libraries: How can images and text be used together? In *Proceedings of the 20th Conference on Very Large Databases*. Santiago, Chile, 655–67.

———. 1997. *Practical digital libraries: Books, bytes, and bucks*. San Francisco: Morgan Kaufmann.

Lewis, P. 1989. The executive computer: The race to market a 486 machine. *The New York Times*. http://www.nytimes.com/1989/10/22/business/the-executive-computer-the-race-to-market-a-486-machine.html (accessed 22 March 2014).

Li, J. 2001. Image compression: The mechanics of the jpeg 2000. Microsoft Research. http://research.microsoft.com/en-us/um/people/jinl/paper_2001/msri_jpeg2000.pdf (accessed 21 July 2014).

Library of Congress. 2013. METS: Metadata encoding & transmission standard. http://www.loc.gov/standards/mets/ (accessed 9 January 2014).

———. 2014a. Chronicling America: Historic American newspapers. http:/ /chroniclingamerica.loc.gov (accessed 12 December 2014).

———. 2014b. Using ALTO with METS. http://www.loc.gov/standards/alto/ techcenter/use-with-mets.php (accessed 9 January 2014).

Littman, J. 2007. Actualized preservation threats: Practical lessons from Chronicling America. *D-Lib Magazine*. http://www.dlib.org/dlib/july07/littman/ 07littman.html (accessed 12 December 2014).

Llorà, X., B. Ács, L. Auvil, B. Capitanu, M. Welge, and D. Goldberg. 2008. Meandre: Semantic-driven data-intensive flows in the clouds. In *Proceedings of the Fourth IEEE International Conference on eScience*. Indianapolis, IN, 7–12 December, 238–45.

Loughry, T. 1993. Putting scholarly publications on line. *The Chronicle of Higher Education* . http://chronicle.com/article/Putting-Scholarly-Publications/ 71440/ (accessed 13 January 2014).

Lucier, R., and P. Brantley. 1995. The Red Sage project: An experimental digital journal library for the health sciences. *D-Lib Magazine* . http://www.dlib.org/ dlib/august95/lucier/08lucier.html (accessed 13 January 2014).

Lunacek, M., J. Braden, and T. Hauser. 2013. The scaling of many-task computing approaches in Python on cluster supercomputers. In *Proceedings of the International Conference on Cluster Computing*. Indianapolis, IN, 23–27 September, 1–8.

MacMillan, K, M Droettboom, and I Fujinaga. 2002. Gamera: Optical music recognition in a new shell. In *Proceedings of the International Computer Music Conference*. Gothenburg, Sweden, 482–5.

MacMillan, K., M. Droettboom, and I. Fujinaga. 2001. Gamera: A structured document recognition application development environment. In *Proceedings of the Conference of the International Society for Music Information Retrieval*. Bloomington, IN, 15–17 October, 15–6.

Martin, D. 2007. David H. Shephard, 84, dies; Optical reader inventor. *The New York Times*, 11 December. http://www.nytimes.com/2007/12/11/us/11shepard.html (accessed 16 April 2014).

Maxwell, J. 1981. Mockingbird: An interactive composer's aid. M.S. diss., MIT.

McCallum, J. 2013. Disk drive prices (1955-2013). http://jcmit.com/diskprice.htm (accessed 26 November 2013).

McCone, G. 1992. Preservation R&D activities at the National Agricultural Library. In *Proceedings of the Round table on Preservation Research and Development*. Washington, DC, 28–29 September, 62–9.

McGee, W., and P. Merkley. 1991. The optical scanning of medieval music. *Computers and the Humanities* 25 (1): 47–53.

McGeehan, T., and J. Maddock. 1975. DDC 10 year requirements and planning study. Interagency survey report. Defense Documentation Center.

McPherson, J. 2006. Coordinating knowledge to improve optical music recognition. PhD diss., University of Waikato.

McPherson, J., and D. Bainbridge. 2001. Coordinating knowledge within an optical music recognition system. In *Proceedings of the The 4th New Zealand Computer Science Research Students' Conference*. Christchurch, New Zealand, 50–8.

Michel, J., Y. Shen, A. Aiden, A. Veres, M. Gray, The Google Books Team, J. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. Nowak, and E. Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science* 331 (6014): 176–82.

Michener, W., J. Beach, M. Jones, B. Ludäscher, D. Pennington, R. Pereira, A. Rajasekar, and M. Schildhauer. 2007. A knowledge environment for the biodiversity and ecological sciences. *Journal of Intelligent Information Systems* 29 (1): 111–26.

MIDI Manufacturers Association. 1996. The complete MIDI 1.0 detailed specification. http://www.midi.org/techspecs/midispec.php (accessed 27 March 2014).

Miller, R. 2012. The Internet Archive book digitization process. The Internet Archive. https://archive.org/details/ProcessDocument (accessed 10 January 2014).

Mills, C., and L. Weldon. 1987. Reading text from computer screens. *ACM Computing Surveys* 19 (4): 329–57.

Mori, S., C. Suen, and K. Yamamoto. 1992. Historical review of OCR research and development. *Proceedings of the IEEE* 80 (7): 1029–58.

Müller, M., H. Mattes, and F. Kurth. 2006. An efficient multiscale approach to audio synchronization. In *Proceedings of the Conference of the International Society for Music Information Retrieval*. Victoria, BC, 8–12 October, 192–7.

Mullin, J. 2013. Google Books ruled legal in massive win for fair use. *Ars Technica*. http://arstechnica.com/tech-policy/2013/11/google-books-ruled-legal-in-massive-win-for-fair-use/ (accessed 15 January 2014).

Music Encoding Initiative Council. 2013. *The Music Encoding Initiative guidelines, release 2013*. Charlottesville, VA: Music Encoding Initiative Council.

Musitek. 2014. Smartscore music scanning software. https://www.musitek.com (accessed 11 February 2014).

Muter, P., S. Latrémouille, W. Treurniet, and P. Beam. 1982. Extended reading of continous text on television screens. *Human Factors* 24 (5): 501–8.

Myka, A. 1994. Putting paper documents in the world-wide web. In *Proceedings of the Second International WWW Conference*. Chicago, IL, 17–20 October, 199–208.

Myka, A., and J. Guntzer. 1993. Using electronic facsimiles of documents for automatic reconstruction of underlying hypertext structures. In *Proceedings of the Second International Conference on Document Analysis and Recognition*. Tsukuba City, Japan, 20–22 October, 528–32.

Nadella, S. 2008. Book search winding down. *Microsoft Search Blog*. http://www.bing.com/blogs/site_blogs/b/search/archive/2008/05/23/book-search-winding-down.aspx (accessed 11 January 2014).

Nagy, G. 1968. Preliminary investigation of techniques for automated reading of unformatted text. *Communications of the ACM* 11 (7): 480–7.

Nagy, G., T. Nartker, and S. Rice. 1999. Optical character recognition: An illustrated guide to the frontier. *Proceedings of the SPIE: Document Recognition and Retrieval* 3967: 58–69.

Nagy, G., S. Seth, and M. Viswanathan. 1992. A prototype document image analysis system for technical journals. *Computer* 25 (7): 10–22.

Nagy, G., and G. Shelton. 1966. Self-corrective character recognition system. *IEEE Transactions on Information Theory* 12 (2): 215–22.

National Library of Australia. 2013. Australian newspaper digitisation program. http://www.nla.gov.au/content/newspaper-digitisation-program (accessed 9 January 2014).

National Library of New Zealand. 2013. Papers past. http://paperspast.natlib.govt.nz/cgi-bin/paperspast (accessed 9 January 2014).

Neudecker, C. 2011. The IMPACT interoperability framework: Workflows for OCR and beyond. Presentation given at the IMPACT Final Conference, London, UK, http://vimeo.com/32027994 (accessed 24 June 2014).

Neudecker, C., S. Schlarb, Z. Dogan, P. Missier, S. Sufi, A. Williams, and K. Wolstencroft. 2011. An experimental workflow development platform for historical document digitisation and analysis. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*. Singapore, 16–17 September, 161–8.

Neuratron. 2014. Photoscore music scanning software. http://www.neuratron.com/photoscore.htm (accessed 11 February 2014).

Newby, G., and C. Franks. 2003. Distributed proofreading. In *Proceedings of the Joint Conference on Digital Libraries*. Houston, TX, 27–31 May, 361–3.

Nienhuys, H., and J. Nieuwenhuizen. 2003. Lilypond, a system for automated music engraving. In *Proceedings of the 14th Colloquium on Musical Informatics*. Firenze, Italy, 8–10 May, CIM-1–6.

Norvig, P. 2014. How to write a spelling corrector. http://norvig.com/spell-correct.html (accessed 6 August 2014).

O'Gorman, L. 1992. Image and document processing techniques for the RightPages electronic library system. In *Proceedings of the International Conference on Pattern Recognition.* The Hague, Netherlands, 30 August–3 September, 260–3.

Orchard, L., P. Pehlivanian, S. Koon, and H. Jones. 2009. Part III: Ext JS. In *Professional javascript frameworks prototype, YUI, Ext JS, Dojo and Mootools,* 335–450. Indianapolis, IN: Wiley.

Ouyang, Y., J. A. Burgoyne, L. Pugin, and I. Fujinaga. 2009. A robust border detection algorithm with application to medieval music manuscripts. In *Proceedings of the International Computer Music Conference.* Montréal, QC, 16–21 August, 101–4.

Page, K., B. Fields, D. de Roure, T. Crawford, and J. S. Downie. 2013. Capturing the workflows of music information retrieval for repeatability and reuse. *Journal of Intelligent Information Systems* 41: 435–59.

Palowitch, C., and D. Stewart. 1995. Automating the structural markup process in the conversion of print documents to electronic texts. In *Proceedings of the Theory and Practice of Digital Libraries.* Austin, TX, 11–13 August,

Perrey, R., and M. Lycett. 2003. Service-oriented architecture. In *Proceedings of the Symposium on Applications and the Internet.* Orlando, FL, 27–31 January, 116–9.

Phelps, T., and R. Wilensky. 1996. Toward active, extensible, networked documents: Multivalent architecture and applications. In *Proceedings of the First ACM International Conference on Digital Libraries.* Bethesda, MD, 20–23 March, 100–8.

Pillay, R. 2012. IIP image server: Demos. http://iipimage.sourceforge.net/demo/ (accessed 25 March 2014).

Pinto, T., A. Rebelo, G. Giraldi, and J. Cardoso. 2011. Music score binarization based on domain knowledge. In *Proceedings of the Pattern Recognition and Image Analysis.* Palmas de Gran Canaria, Spain, 8–10 June, 700–8.

Pitzalis, D., R. Pillay, and C. Lahanier. 2006. A new concept in high resolution internet image browsing. In *Proceedings of the International Conference on Electronic Publishing.* Bansko, Bulgaria, 14–16 June, 291–8.

Prerau, D. 1970. Computer pattern recognition of standard engraved music notation. PhD diss., Massachusetts Institute of Technology.

———. 1971. Computer pattern recognition of printed music. *AFIP Joint Computer Conferences* 39: 153–62.

Price, G. 2003. Amazon debuts new book search tool. *SearchEngineWatch.com.* http://searchenginewatch.com/article/2064555/Amazon-Debuts-New-Book-Search-Tool).

Project JAXB. 2014. Project jaxb. https://jaxb.java.net (accessed 18 July 2014).

Project Petrucci. 2014. International music score library project. http://imslp.org/wiki/Main_Page (accessed 13 June 2014).

Pruslin, D. 1966. Automatic recognition of sheet music. Sc. D. diss., Massachusetts Institute of Technology.

Pugin, L. 2006a. Lecture et traitement informatique de typographies musicales anciennes: Un logiciel de reconnaissance de partitions par modèles de Markov cachés. PhD diss., Geneva University.

———. 2006b. Optical music recognition of early typographic prints using hidden Markov models. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Victoria, BC, 8–12 October, 53–6.

Pugin, L., J. A. Burgoyne, D. Eck, and I. Fujinaga. 2007a. Book-adaptive and book-dependent models to accelerate digitization of early music. In *Proceedings of the NIPS Workshop on Music, Brain, and Cognition.* Whistler, BC, 7–8 December, 1–8.

Pugin, L., J. A. Burgoyne, and I. Fujinaga. 2007b. MAP adaptation to improve optical music recognition of early music documents using hidden Markov models. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Vienna, Austria, 23–27 September, 513–6.

Pugin, L., J. Hockman, J. A. Burgoyne, and I. Fujinaga. 2008. Gamera versus Aruspix: Two optical music recognition approaches. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Philadelphia, PA, 14–18 September, 419–24.

Quin, L. 2010. XML technology: Schema. http://www.w3.org/schema (accessed 21 March 2014).

Ramirez, C., and J. Ohya. 2010. Symbol classification approach for OMR of square notation manuscripts. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Utrecht, The Netherlands, 9–13 August, 549–53.

———. 2011. OMR of early plainchant manuscripts in square notation: A two-stage system. *Proceedings of the SPIE* 7874: 1–10.

Raphael, C., and J. Wang. 2011. New approaches to optical music recognition. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Miami, FL, 24–28 October, 305–10.

Rawat, S., K. Kumar, M. Meshesha, I. Sikdar, A. Balasubramanian, and C. Jawahar. 2006. A semi-automatic adaptive OCR for digital libraries. In *Proceedings of the International Workshop on Document Analysis Systems.* Nelson, New Zealand, 13–15 February, 13–24.

Rebelo, A. 2012. Robust optical recognition of handwritten musical scores based on domain knowledge. PhD diss., University of Porto.

Rebelo, A., A. Capela, J. da Costa, C. Guedes, E. Carrapatoso, and J. Cardoso. 2007. A shortest path approach for staff line detection. In *Proceedings of the Third International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS)*. 33–44.

Rebelo, A., G. Capela, and J. Cardoso. 2010. Optical recognition of music symbols: A comparative study. *International Journal on Document Analysis and Recognition* 13 (1): 19–31.

Rebelo, A., I. Fujinaga, F. Paszkiewicz, A. Marcal, C. Guedes, and J. Cardoso. 2012. Optical music recognition: State-of-the-art and open issues. *International Journal of Multimedia Information Retrieval* 1 (3): 173–90.

Rebelo, A., J. Tkaczuk, R Sousa, and A. Cardoso. 2011. Metric learning for music symbol recognition. In *Proceedings of the International Conference on Machine Learning and Applications*. Honolulu, HI, 18–21 December, 106–11.

Reddy, R., and G. StClair. 2001. The million book digital library project. http:/ /www.rr.cs.cmu.edu/mbdl.htm (accessed 19 December 2013).

Reenskaug, T. 2010. MVC: Xerox PARC 1978–79. http://heim.ifi.uio.no/~trygver/ themes/mvc/mvc-index.html (accessed 15 March 2014).

Regalado, A. 2011. Who coined 'cloud computing'? *MIT Technology Review* . http:/ /www.technologyreview.com/news/425970/who-coined-cloud-computing/ (accessed 7 March 2014).

Reijers, H. 2006. Workflow flexibility: The forlorn promise. In *Proceedings of the International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.* Manchester, UK, 26–28 June, 271–2.

Rice, S., F. Jenkins, and T. Nartker. 1996. The fifth annual test of OCR accuracy. Information Science Research Institute. TR-96-01. University of Nevada, Las Vegas, Las Vegas, NV.

Rile, J. 2002. DjVu-digital vs. "super hero" PDF. http://djvu.org/resources/ djvu_digital_vs_super_hero_pdf.php (accessed 24 June 2014).

Roach, J. W., and J. E. Tatem. 1988. Using domain knowledge in low-level visual processing to interpret handwritten music: An experiment. *Pattern Recognition* 21 (1): 33–44.

Roelofs, G. 1999. *PNG: The definitive guide*. O'Reilly: Sebastapol, CA.

Roland, P. 2002. The Music Encoding Initiative (MEI). In *Proceedings of the First International Conference on Musical Applications Using XML*. Milan, Italy, 19–20 September, 55–9.

———. 2009. MEI as an editorial music standard. In *Digitale edition zwischen experiment und standardisierung*, edited by P. Stadler, and J. Veit, 175–94. Tübingen: Max Niemeyer.

Rowley-Brooke, R., and A. Kokaram. 2012. Bleed-through removal in degraded documents. *Proceedings of the SPIE: Document Recognition and Retrieval* 8297.

Samuel, A. 1964. The banishment of paper-work. *New Scientist* 21 (380): 529–30.

Sankar, K., V. Ambati, L. Pratha, and C. Jawahar. 2006. Digitizing a million books: Challenges for document analysis. In *Proceedings of the International Workshop on Document Analysis Systems*. Nelson, New Zealand, 13–15 February, 425–36.

Schonfeld, R. 2003. *JSTOR: A history*. Princeton, NJ: Princeton University Press.

Selfridge-Field, E. 1994. Optical recognition of musical notation: A survey of current work. *Computing in Musicology* 9: 109–45.

Selfridge-Field, E. 1997a. *Beyond MIDI: The handbook of musical codes*. Cambridge, MA: The MIT Press.

———. 1997b. Introduction: Describing musical information. In *Beyond MIDI: The Handbook of Musical Codes*, edited by E. Selfridge-Field, 3–38. Cambridge, MA: The MIT Press.

Sezgin, M., and B. Sankur. 2004. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging* 13 (1): 146–65.

Shaw, E., and S. Blumson. 1997. Making of America: Online searching and page presentation at the University of Michigan. *D-Lib Magazine* 3 (7/8). http://www.dlib.org/dlib/july97/america/07shaw.html (accessed 21 March 2014).

Shephard, D. 1953. Apparatus for reading. US Patent 2,663,758, filed 27 April 1931, and issued 27 June 1933.

Sherman, C. 2003. Google introduces book searches. *SearchEngineWatch.com*. http://searchenginewatch.com/article/2065619/Google-Introduces-Book-Searches (accessed 14 January 2014).

Smiley, D., and E. Pugh. 2009. *Solr 1.4 enterprise search server*. Birmingham, UK: Packt Publishing.

Smith, A. 2005. Making books easier to find. http://googleblog.blogspot.ca/2005/08/making-books-easier-to-find.html (accessed 14 January 2014).

Smith, R. 2007. An overview of the Tesseract OCR engine. In *Proceedings of the International Conference on Document Analysis and Recognition*. Curitiba, Brazil, 23–26 September, 629–33.

Smith, R., D. Antonova, and D. Lee. 2009. Adapting the Tesseract open source OCR engine for multilingual OCR. In *Proceedings of the International Workshop on Multilingual OCR*. Barcelona, Spain, 25 July, 1–8.

Sonntag, M., D. Karastoyanova, and E. Deelman. 2010. Bridging the gap between business and scientific workflows: Humans in the loop of scientific workflows. In *Proceedings of the IEEE International Conference on e-Science*. Brisbane, Australia, 7–10 December, 206–13.

Spolsky, J. 2004. The law of leaky abstractions. In *Joel on software,* 197–202. Berkeley, CA: Apress.

Stackpole, L., and R. Atkinson. 1998. The national research library alliance: A federal consortium formed to provide inter-agency access to scientific information. *Issues in Science and Technology Librarianship* 18. http:/ /www.istl.org/98-spring/article6.html (accessed 16 June 2014).

Stackpole, L., and R. King. 1999. Electronic journals as a component of the digital library. *Issues in Science and Technology Librarianship* 22. http://www.istl.org/ 99-spring/article1.html (accessed 16 June 2014).

Story, G., L. O'Gorman, D. Fox, L. Schaper, and H. Jagadish. 1992. The RightPages image-based electronic library for alerting and browsing. *Computer* 25 (9): 17–26.

Szwoch, M. 2008. Using MusicXML to evaluate accuracy of OMR systems. In *Proceedings of the Fifth International Conference on Diagrammatic Representation and Inference*. Herrsching, Germany, 19–21 September, 419–22.

Tabata, K., T. Okada, M. Nagamori, T. Sakaguchi, and S. Sugimoto. 2005. A collaboration model between archival systems to enhance the reliability of preservation by an enclose-and-deposit method. In *Proceedings of the 5th International Web Archiving Workshop*. Vienna, Austria, 22–23 September,

Tang, Y., C. Suen, C. Yan, and M. Cheriet. 1991. Document analysis and understanding: A brief survey. In *Proceedings of the International Conference on Document Analysis and Recognition*. Saint-Malo, France, 30 September–2 October, 17–31.

Tardón, L., S. Sammartino, I. Barbancho, V. Gómez, and A. Oliver. 2010. Optical music recognition for scores written in white mensural notation. *EURASIP Journal on Image and Video Processing* 2009.

Tauschek, G. 1929. Reading machine. US Patent 2,026,329, filed 27 May 1929, and issued 31 December 1935.

Taycher, L. 2010. Books of the world, stand up and be counted! All 129,864,880 of you. *Google Book Search Blog*. http://booksearch.blogspot.ca/2010/08/books-of-world-stand-up-and-be-counted.html (accessed 24 June 2014).

Taylor, I., E. Deelman, D. Gannon, and M. Shields, eds. 2007. *Workflows for e-Science*. London: Springer.

Taylor, M., A. Zappala, W. Newman, and C. Dance. 1999. Documents through cameras. *Image and Vision Computing* 17 (11): 831–44.

Text Encoding Initiative Consortium. 2014. TEI: Text Encoding Initiative. http:/ /www.tei-c.org/ (accessed 5 August 2014).

Thacker, C., E. McCreight, B. Lampson, R. Sproull, and D. Boggs. 1982. ALTO: A personal computer. In *Computer structures: Principles and examples,* edited by D. Siewiorek, C. Bell, and A. Newell, 549–72. New York: McGraw-Hill.

The Unicode Consortium. 2014. *The Unicode standard, version 7.0.0.* Mountain View, CA: The Unicode Consortium.

Thoma, G., S. Suthasinekul, F. Walker, J. Cookson, and M. Rashidian. 1985. A prototype system for the electronic storage and retrieval of document images. *ACM Transactions on Information Systems* 3 (3): 279–91.

Thomas, V., D. Damm, C. Fremerey, M. Clausen, F. Kurth, and M. Müller. 2012a. PROBADO music: A multimodal online music library. In *Proceedings of the International Computer Music Conference*. Ljuljana, Slovenia, 9–14 September, 289–92.

Thomas, V., C. Fremerey, D. Damm, and M. Clausen. 2009. Slave: A score-lyrics-audio-video-explorer. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Kobe, Japan, 26–30 October, 717–22.

Thomas, V., C. Fremerey, M. Müller, and M. Clausen. 2012b. Linking sheet music and audio: Challenges and new approaches. In *Multimodal music processing*, edited by M. Müller, M. Goto, and M. Schedl, 1–22. Leibniz, Germany: Dagstuhl Publishing.

Thompson, J., A. Hankinson, and I. Fujinaga. 2011. Searching the Liber usualis: Using CouchDB and ElasticSearch to query graphical music documents. Poster presented at the Conference of the International Society for Music Information Retrieval, Miami, FL, 24–28 October.

Toselli, A., V. Romero, and E. Vidal. 2007. Viterbi based alignment between text images and their transcripts. In *Proceedings of the Workshop on Language Technology for Cultural Heritage Data*. Prague, Poland, 28 June, 9–16.

———. 2011. Alignment between text images and their transcripts for handwritten documents. In *Language technology for cultural heritage,* edited by C. Sporleder, A. Bosch, and K. Zervanou, 23–37. Berlin: Springer.

Trier, Ø., and A. Jain. 1995. Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (12): 1191–201.

Universal Digital Library. 2007. Current status. http://www.ulib.org/ ULIBProgressReport.htm (accessed 9 January 2014).

University of Michigan. 2001. Assessing the costs of conversion: Making of America IV: The American voice 1850–1876. The Andrew W. Mellon Foundation. The University of Michigan.

———. 2005. [University of Michigan] library/Google digitization partnership FAQ. http://www.lib.umich.edu/files/services/mdp/faq.pdf (accessed 15 January 2014).

Varley, T. 1969. Data input error detection and correction procedures. George Washington University Logistics Research Project. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0689365 (accessed 16 June 2014).

Veillard, D. 2014. The XML C parser and toolkit of Gnome. http://xmlsoft.org (accessed 27 March 2014).

Vercoe, B. 1991. *Csound.* Cambridge, MA: Massachusetts Institute of Technology.

Videla, A., and J. Williams. 2012. *RabbitMQ in action: Distributed messaging for everyone.* Shelter Island, NY: Manning Publications.

Vigliensoni, G., J. A. Burgoyne, A. Hankinson, and I. Fujinaga. 2011. Automatic pitch detection in printed square notation. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Miami, FL, 24–28 October, 423–8.

Vigliensoni, G., G. Burlet, and I. Fujinaga. 2013. Optical measure recognition in common music notation. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Curitiba, Brazil, 4–8 November, 125–30.

Vincent, L. 2006. Announcing Tesseract OCR. *Google Code Blog.* http://googlecode.blogspot.ca/2006/08/announcing-tesseract-ocr.html (accessed 15 January 2014).

Viro, V. 2011. Peachnote: Music score search and analysis platform. In *Proceedings of the Conference of the International Society for Music Information Retrieval.* Miami, FL, 24–28 October, 359–62.

Visani, M., V. Kieu, A. Fornés, and N. Journet. 2013. The ICDAR 2013 music scores competition: Staff removal. In *Proceedings of the International Conference on Document Analysis and Recognition.* Washington, DC, 25–28 August, 1407–11.

von Ahn, L., B. Maurer, C. McMillen, D. Abraham, and M. Blum. 2008. ReCAPTCHA: Human-based character recognition via web security measures. *Science* 321 (5895): 1465–8.

W3 Consortium. 2012. A short history of javascript. https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript (accessed 25 July 2014).

Walker, F., and G.R. Thoma. 1990. Access techniques for document image databases. *Library Trends* 38 (4): 751–86.

Walton, N., and E. Gonzalez-Solares. 2009. Astrogrid and the virtual observatory. In *Jets from young stars v*, edited by J. Gracia, F. Colle, and T. Downes, 81–113. Berlin: Springer.

Wang, Y., I. Phillips, and R. Haralick. 2001. Automatic table ground truth generation and a background-analysis-based table structure extraction method. In *Proceedings of the International Conference on Document Analysis and Recognition*. Seattle, WA, 10–13 September, 528–32.

Wei, L., Q. Salih, and H. Hock. 2008. Optical tablature recognition (OTR) system: Using fourier descriptors as a recognition tool. In *Proceedings of the International Conference on Audio, Language and Image Processing*. Shanghai, China, 1532–9.

Weir, C., S. Liebowitz Taylor, S. Harding, and B. Croft. 1997. The skeleton document image retrieval system. In *Proceedings of the Symposium on Document Image Understanding Technologies*. Annapolis, MD, 30 April–2 May, 8–15.

Whitley, K. 1997. Visual programming languages and the empirical evidence for and against. *Journal of Visual Languages and Computing* 8: 109–42.

Wienbibliothek im Rathaus. 2010. Schubert-autographe. http://www.schubert-online.at (accessed 14 January 2014).

Wijaya, K., and D. Bainbridge. 1999. Staff line restoration. In *Proceedings of the Seventh International Conference on Image Processing and Its Applications*. Manchester, UK, 13–15 July, 760–4.

Williams, G. 1984. The Apple Macintosh computer. *BYTE Magazine* February: 30–54.

Xu, Y., and G. Nagy. 1999. Prototype extraction and adaptive OCR. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (12): 1280–96.

York, J. 2009. This library never forgets: Preservation, cooperation, and the making of HathiTrust digital library. In *Proceedings of the Archiving Conference*. Arlington, VA, 4–7 May, 5–10.

Young, S., G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. 2006. *The HTK book*. Cambridge, UK: Cambridge University Engineering Department.

Zinger, S., J. Nerbonne, and L. Schomaker. 2009. Text-image alignment for historical handwritten documents. *Proceedings of the SPIE: Document Recognition and Retrieval* 7247.

Colophon

This dissertation was written with Mellel, a highly regarded word processor by a small software company, RedleX. I have been a user of Mellel for almost 10 years, and have found it to provide a more-than-adequate replacement for Microsoft Word without needing to understand the intricacies of LaTeX.

The body of this dissertation is set in 12 point Huronia Latin Pro, a font designed by Canadian typographer Ross Mills at Tiro Typeworks in Vancouver, British Columbia. Titles and monospace fonts are from the Adobe Source Sans Pro and Source Code Pro families, respectively. These are designed by Paul D. Hunt, and notable for being Adobe's first efforts at producing an open-source font.

MONTRÉAL, QUEBEC

DECEMBER 2014

E
0
T