The Design of an Integrated System For Rapid Prototyping With Ice

by

Eric Barnett

Doctor of Philosophy

Department of Mechanical Engineering

McGill University

Montreal, Quebec, Canada

May 2012

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

ABSTRACT

Ice construction has been used for practical and artistic purposes for thousands of years. In the past few decades, computer numerical control (CNC) ice sculpting has become a well-established industry, with many companies producing ice sculptures for corporate events. This thesis presents a novel form of ice construction, namely, 3D ice printing, also known as rapid freeze prototyping. Rapid prototyping is an interesting alternative technique for ice construction, since it can be used to produce many objects that would be difficult or impossible to build with manual or CNC techniques. A robot-assisted rapid prototyping system has been produced by retrofitting an Adept Cobra 600 robot. This thesis first provides an overview of the system; then, several of the innovative control algorithms that have been implemented are reported.

The Cobra 600 system is capable of constructing any 3D model, up to a size of approximately $300 \times 300 \times 200$ mm; the system accuracy is approximately 0.5 mm, with the rate of construction about one litre per day. Most of the retrofitting work has been on the software side; several innovative algorithms have been produced, which represent the main contributions to knowledge of this thesis. A part-slicing algorithm produces deposition paths from CAD parts; compared to the established algorithms on which it is based, computational efficiency has been greatly improved. A trajectory control algorithm produces data for controlling the Cobra 600 system during part construction. This algorithm is unique because of its simple, robust approach to the time-optimal trajectory planning problem, and its independence of a dynamic model of the robotic system. A geometric feedback system has been developed that detects and corrects geometry errors during part construction. An integral component of this system is a three-dimensional spatial PID controller, produced by transforming the traditional PID controller, which is one-dimensional in time. Time-varying flow control has also been developed and implemented, as both a necessary feature to

complement variable-speed deposition paths and a means to correct part errors detected with the geometric feedback algorithm. All of the algorithms described above have been integrated to produce the Cobra 600 software implementation. Many of the innovative techniques developed are applicable to other rapid prototyping systems; additionally, some are applicable to other fields of research, including mapping, manufacturing, and signal processing.

RÉSUMÉ

La construction en glace est utilisée à des fins pratiques et artistiques depuis des milliers d'années. Au cours des dernières décennies, l'usinage de glace par CNO (commande numérique par ordinateur) est devenu un secteur bien établi, où également diverses sociétés produisent des sculptures pour des événements corporatifs. Cette thèse porte sur une nouvelle technique de construction en glace par impression 3 D. Cette technique, aussi appelée le prototypage rapide, peut être utilisé pour produire des objets qu'il serait difficile ou même impossible de produire avec les techniques traditionnelles d'enlèvement de matière. Un système de prototypage rapide en glace a été conçu et mis au point au moyen d'un robot Cobra 600 d'Adept Technology. Cette thèse donne un aperçu du système d'impression 3 D et des divers algorithmes novateurs mis au point.

Le système Cobra 600 est capable de construire n'importe quel objet, de dimensions maximales de $300 \times 300 \times 200$ mm, avec une précision d'environ 0,5 mm, et une cadence de construction d'environ un litre par jour. La plupart des contributions à l'avancement des connaissances porte sur les logiciels. Un algorithme de tranchage de l'objet produit les trajectoires de déposition ; comparé aux algorithmes bien établis, sur lesquels il est basé, l'efficacité de calcul est nettement meilleur. Un deuxième algorithme génère l'information de commande nécessaire pour construire un objet avec le système Cobra 600. Son approche simple et robuste minimise le temps requis pour suivre une trajectoire, et le rend indépendant du modèle dynamique du robot. Un système d'asservissement géométrique détecte et corrige les erreurs géométriques durant la construction de l'objet. Un régulateur OIT (odométrique, intégral, tachymétrique) spatiale 3 D remplace le régulateur OIT traditionel, unidimensionnel et temporel. La commande variable de l'écoulement du liquide complète les trajectoires de déposition à vitesse variable, et corrige les erreurs détectées au

moyen du système d'asservissement géométrique. Tous les algorithmes décrits ci-dessus sont intégrés afin de permettre l'utilisation du logiciel du système Cobra 600. De nombreuses techniques novatrices mises au point dans le cadre de cette recherche sont susceptibles d'application aux autres systèmes de prototypage rapide ; en outre, certaines peuvent être appliquées à d'autres domaines, comme la cartographie, le traitement de signaux, et la fabrication.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Jorge Angeles, for his valued advice and support over the course of my thesis work. My co-supervisor, Professor Damiano Pasini, has also provided valuable input and guidance. Professors Angeles and Pasini both made valuable recommendations and editorial corrections during the preparation of my thesis. Professor Pieter Sijpkes, the director of the Ice Laboratory, has provided many helpful suggestions and ideas for my thesis work. I also thank the staff of the Centre for Intelligent Machines and the Department of Mechanical Engineering, including Irène Cartier, Jan Binder, Patrick McLean, Cynthia Davidson, Marlene Gray, Joyce Nault, and Nozomi Kanekatsu, for their support and assistance.

I would like to thank the Social Sciences and Humanities Research Council of Canada (SSHRC), which supported this research under Strategic Research Grant 848-2006-14. I acknowledge the scholarships I received from Natural Sciences and Engineering Research Council of Canada (NSERC), Quebec's Fonds québecois de la recherche sur la nature et les technologies (FQRNT), the Fondation Universitaire Pierre Arbour, and McGill University's Faculty of Engineering. The generous rebate received from Adept Technology is dutifully acknowledged.

I had many productive discussions with my colleagues within the Robotic Mechanical Systems Laboratory, including Toufic Azar and Xiaoqing Ma. I would also like to express my appreciation for the assistance provided by all of the undergraduate and exchange students who worked in the Ice Laboratory during the course of my thesis, including Tristan Pashley, Robert Pontarelli, and Eric McDonald. Alessandro Ossino is thanked for his contribution to the development of the Matlab slicing code. Finally, I would like to thank my parents for their continued support and encouragement over the years.

CLAIM OF ORIGINALITY

The author claims the originality of the research reported in this thesis. Specific claims are included below:

- The geometric feedback algorithm, *surface mapping feedback* (SMF), is original in both the rapid prototyping and control domains. SMF is the first true geometric feedback system designed for continuous-flow rapid prototyping. SMF is also a three-dimensional spatial PID control implementation, where the typical PID control loop is one-dimensional in time.
- The trajectory-planning algorithm, *minimum-time trajectory shaping* (MTTS), consists only of simple, robust steps; typically, a nonlinear optimization solution procedure is used for this problem. Additionally, a mathematical model of the system dynamics is not needed to implement MTTS.
- The unique frequency-modulation technique used for variable-flow control simultaneously synchronizes variable flow with variable end-effector speed and implements geometry correction with the geometric feedback system.
- The part-slicing algorithm, implemented in Matlab, has several unique capabilities. Firstly, the algorithm can successfully and efficiently slice CAD parts composed of tens of millions of triangles. Secondly, a CAD model of the scaffolding is not needed, scaffolding deposition paths being produced directly from the CAD model of the part. Finally, concentric fill paths are generated automatically, without error, regardless of the part geometry complexity.
- The discrete stability analysis for SMF establishes clear restrictions on the PID controller gains, within which the system is stable.
- A robust method for optimizing the SMF PID controller gains is developed, which places no restriction on the system disturbance model.

- Rapid prototyping is a novel application for a SCARA system, in this case the Cobra 600 robot. Normally, 3-axis Cartesian systems are used for positioning in rapid prototyping applications.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

$\mathbf{b}$:	vector whose entries are the value of the variable b, evaluated at each point along a path; b represents the following variables: a, c, δ, p, ρ, Q, s, t, θ, ν, ζ, x, y, z, ϕ, γ, ψ
$\Delta\mathbf{b}$:	vector whose entries are differences between adjacent entries of $\mathbf{b}$
$\mathbf{B}$:	matrix whose entries are the value of the surface B, evaluated at each point of a grid, where B represents the following variables: A, E, U
a:	acceleration along a path
A:	control action surface
CAD:	computer-aided design
CCD:	charge-coupled device
CMOS:	complementary metaloxidesemiconductor
CNC:	computer numerical control
c:	speed along a path
d:	disturbance for the SMF PID controller
e:	part error
E:	part error surface
EE:	end effector
$\mathcal{F}_0$:	the inertial base frame of a robot
$\mathcal{F}_i$:	coordinate frame attached to robot link $i-1$
h:	part height
IP:	interior-point
ISE:	integral of the squared error
IAE:	integral of the absolute value of the error
ITAE:	integral of the time multiplied by the absolute error

k: iteration number of the SMF PID controller

K_p: proportional gain

K_i: integral gain

K_d: derivative gain

l_i: the length of robot link i

ME: methyl ester

MS: medium-scale

MTTS: minimum-time trajectory shaping

n_d: number of deposition layers per measurement interval

O_i: The origin of coordinate frame i

p_i: position of joint i

P: The end effector operation point

PID: proportional-integral-derivative

PLY: Stanford polygon file format

Q: material flow rate

r: SMF PID controller input

RFP: rapid freeze prototyping

RMS: root-mean-squared

RP: rapid prototyping

s: the distance travelled along a path

S&H: spike and hold

SCARA: Selective Compliance Assembly Robot Arm

SFF: solid free-form fabrication

SME: shortening methyl ester

SMF: surface mapping feedback

STL: stereolithography file format

SSE: sum of the squared error

t:	time
T:	time of travel for a path
TOTP:	time-optimal trajectory planning
TTL:	transistor-transistor logic
u:	SMF PID controller output
U:	SMF PID controller output surface
U':	SMF PID controller output surface, saturated
UART:	universal asynchronous receiver-transmitter
V':	material volume deposited per distance travelled
x, y, z:	Cartesian positioning variables
w:	deposition path width
ZN:	Ziegler-Nichols

$\boldsymbol{\alpha}$:	control action surface, evaluated at each point along a path
δ:	point-to-point duration on a path
Δh_m:	measurement interval
Δh_d:	deposition layer thickness
ϕ, γ, ψ:	EE orientation variables
$\mathbf{\Gamma}$:	matrix of ones
ν:	microsolenoid valve frequency
θ_i:	position of revolute joint i
ρ:	process parameter, or SMF PID controller system output
σ_l^*:	RMS deviation from the nominal part height for one layer
σ_p^*:	Average value of σ_l^* all layers of the part
τ:	normalized time
ζ:	EE orientation in $\mathcal{F}_0$

CHAPTER 1
Introduction

Ice construction has fascinated people for thousands of years. On the practical side, ice roads and shelters have enabled access to and settlement of remote areas. Artistic ice construction has a long tradition of its own; in recent decades, it has become more popular than ever. As new technologies have become available, the manufacture of ice structures has become increasingly automated. Computer numerical control (CNC) ice construction has become quite well established, with several companies offering the capability to build certain parts on demand[1].

In traditional CNC machining, a part is formed by removing material, using operations such as milling and drilling. An obvious alternative to *material removal* is *material addition*, also known as solid free-form fabrication or rapid prototyping (RP). Compared to CNC, RP is a relatively new technology, first achieving widespread use in the 1980s. However, extremely complex parts can be built using RP that would either be impossible or require specific, expensive tooling using traditional CNC techniques. Also, the process from design to fabrication is much simpler using RP; it can be as simple as "printing" a 3D part much as one would print a 2D document with a regular printer. Of course, RP also has many drawbacks, the most obvious being fabrication time: a part that measures roughly 100 mm in each dimension could easily

[1]Ice Sculptures Ltd., Grand Rapids, MI (www.iceculture.com)
Ice Culture Inc., Hensall, ON (www.iceguru.com)

take 50 hours to build. Also, RP machines typically cost several hundred thousand dollars, and most construction materials cost at least $50/kg.

Rapid prototyping represents an attractive option for ice construction. In addition to the advantages of RP listed above, RP with ice is inexpensive: water costs almost nothing when compared to typically used RP materials. Also, fume control infrastructure and other expensive subsystems are unnecessary, since ice poses no health hazards. RP with ice also has almost no environmental impact, other than the power consumed by the system. Of course, these advantages are all characteristic of *water*; a scaffolding material is also needed, which might not have all of these advantages.

This thesis introduces a novel ice construction platform, dubbed the Cobra 600 rapid freeze prototyping (RFP) system, shown in Fig. 1–1. Emphasis is placed on the novel hardware and software systems developed and the unique contributions to knowledge. The material in the thesis follows the order of items listed below in the Objectives. In Ch. 2, the various hardware subsystems are introduced. Chapter 3 describes the part-slicing algorithm, which can produce deposition paths from any part in the STL or PLY format. In Ch. 4, the trajectory-control algorithm is introduced, which minimizes the time of travel for each path, subject to system and user-defined constraints. Chapter 5 describes the geometric feedback algorithm, which is used to periodically measure the top surface of the part, with a laser displacement system, and adjust subsequent deposition control data to correct detected errors. In Ch. 6, variable-flow control is proposed and applied to trajectory control and geometric feedback. In Ch. 7, the completed research is summarized and recommendations for future work are made.

The work described in this thesis is part of a larger ice construction research program [1] at McGill University, led by Professor Pieter Sijpkes, School of Architecture. Over the years, Prof. Sijpkes has led many manual ice construction projects, including hyperbolic paraboloid structures, a catenary arch, and a 1/5-scale model of

Figure 1–1: The Cobra 600 rapid freeze prototyping system

the Pantheon. In 2007, he formed a partnership with Professors Jorge Angeles and Damiano Pasini, Department of Mechanical Engineering, to develop automated ice construction techniques. The present work is the first example of this new initiative.

1.1 Objectives

The main objective of this research is to develop a rapid freeze prototyping system that produces ice parts as quickly and as accurately as possible. This broad objective is divided into five parts:

1. To produce a rapid freeze prototyping system by integrating several hardware subsystems with an industrial robot
2. To produce a part-slicing algorithm that can produce deposition path data for building *any* CAD part
3. To produce a trajectory-control algorithm that takes deposition paths and produces the data needed to control robot positioning and valve flow along the paths

4. To produce a geometric feedback algorithm that detects and corrects part geometric errors
5. To produce a variable-flow control system to complement the trajectory control and geometric feedback algorithms

The reason for the first objective is self-evident, since it concerns the creation of the ice prototyping system. The other objectives are all motivated by a desire to maximize the following system performance parameters: versatility, speed, accuracy, reliability, and automation. The part-slicing algorithm improves the system versatility and automation; it enables the the automated construction of any part, based only on the CAD data. The trajectory control algorithm produces variable-speed deposition paths, greatly reducing part-construction time, when compared to constant-speed paths. The geometric feedback algorithm increases system accuracy by a factor of about 10 and absolves the user of fine-tuning parameters that affect flow control. The variable-flow control system is needed for implementing Objective 3 and also provides a natural means of implementing error correction for Objective 4.

1.2 Literature Review

Professor Ming C. Leu et al., from the University of Missouri-Rolla, developed a rapid freeze prototyping (RFP) system consisting of a valve/nozzle liquid delivery system positioned by stepper-motor driven axes [2–4]. A sugar-water solution is used as a scaffolding material [4, 5]. One application of their system is the production of moulds for investment casting [6, 7].

Leu et al. have demonstrated that their system is a successful 3D ice printer, though several key features can be improved. Firstly, a stepper-motor driven Cartesian positioning system is inexpensive but cannot be used to implement advanced kinematic control techniques; a system with closed-loop positioning control would be preferred. Advanced kinematic control techniques can be used to optimize path speed and carefully synchronize it with material flow rate. Additionally, based on published

literature, it is unclear whether their system is capable of printing *any* 3D geometry. This feature is very important, since the ability to print highly complex geometry is frequently the main motivating factor for using a 3D printer rather than other manufacturing techniques. The 3D ice printer developed by Leu et al. also appears to be limited to producing small parts, up to approximately 10 cm length in any dimension; the ability to build much larger parts is desirable. Finally, in developing their scaffolding material, Leu et al. limit their investigation to water-based solutions, which diffuse into the ice part during deposition. Ideally, a scaffolding material should exhibit no diffusion.

The history of ice construction research at McGill University, led by Prof. Pieter Sijpkes, is described in [1]. In [8], the initial development and overview of two small-scale RFP systems, the Cobra 600 RFP system and the Fab@Home RFP system, are described. Elements of Ch. 2–5 correspond to material published in [9], [10], [11], and [12], respectively. The relevant literature for each of these chapters is reviewed in the subsections below.

1.2.1 Part Slicing

Part slicing refers to the generation of deposition path data for rapid prototyping, based on part geometry stored in a CAD file. The first step involves the intersection of horizontal planes with 3D geometry to produce boundary contours for a series of deposition layers. This process is similar to the intersection of planes with volumetric data during post-processing for computational fluid dynamics (CFD) [13, 14] and magnetic resonance imaging (MRI) [15]. In fact, Matlab includes functions such as SLICE and CONTOURSLICE, which compute planar intersections with volumetric data [16, p. 6-12]. An important distinction, however, is that RP input data are not volumetric, but contain *only* a set of triangles in 3D, which represent the surface enclosing a 3D volume. Therefore, volumetric data slicing techniques are not readily adaptable to part slicing for RP.

Other steps of the part slicing include the generation of scaffolding boundaries, boundary offsetting to account for deposition path width, and fill path generation. Literature on the subject typically focuses on only one of these steps [17–20].

The part-slicing algorithm developed for the Cobra 600 RFP system, called RPSLICE [10], is an improved version of a previously developed Matlab code [21]. The Matlab mapping toolbox provides many functions that are suitable for implementing parts of the part-slicing process. However, many of these functions have not been optimized for computational efficiency and are prohibitively slow to be used for part slicing. Consequently, decreasing the computational cost of these functions would make them feasible options for part slicing and also improve their overall usefulness. The performance of specific steps of RPSLICE is compared to the relevant literature as the steps are introduced in Ch. 3.

1.2.2 Minimum-Time Trajectory Shaping

The output of part slicing is a series of deposition paths, divided into many layers, that represent a 1D approximation of part and scaffolding geometry. However, no information is included pertaining to how a positioning system is meant to *follow* the paths. To minimize the total part-construction time, it desirable to minimize the time of travel along each deposition path. This is an example of the time-optimal trajectory planning (TOTP) problem, which has received extensive treatment in the literature.

Given that the Cobra 600 RFP system can be classified as a dispensing system, the trajectory control used for other such systems could be relevant. For example, revolute manipulators have been used in welding applications for decades. However, a common requirement for trajectory planning in welding is the use of constant, or near-constant, end-effector speed, to facilitate *uniform* dispensing [22, 23]. Highly-variable speed is undesirable because it makes it difficult to produce uniform welds. Trajectory control research for robot welding tends to focus on optimizing seam tracking [24],

weaving [25], and weld orientation [26]; achieving time-optimality by maximizing the end effector speed at all path locations is not typically an objective. Additionally, trajectory control for welding is normally performed *online*, while an *offline* method is preferable for RFP, as justified below.

Nearly all early techniques proposed in the literature for TOTP rely on a detailed dynamic model of the robotic manipulator and the ability to implement low-level, i.e., dynamic, feedback control. In [27], an algorithm is reported for minimizing the time of travel for a prescribed path by imposing actuator torque constraints and determining acceleration switching points on the trajectory, using forward and backward numerical integration in the position-velocity state space. A procedure that accepts a wider variety of problem constraints, including jerk limits, is used in [28]. A geometric approach to the problem is proposed in [29].

More recently [30], the time-optimal path tracking problem is transformed into a convex optimal control problem, and a technique for trading off time-optimality and energy-optimality is developed. Obradović et al. use the Miele method to solve the problem, which directly gives the absolute minimum, but has limited applicability to problems with kinematic constraints [31].

All dynamic techniques are designed with a positioning system *developer* in mind, who has access to a detailed mathematical model of the system and can implement low-level dynamic control. These techniques are inaccessible to the general *user* of a positioning system, since the low-level dynamic control is not normally possible with an off-the-shelf controller, and manufacturers do not typically provide detailed dynamic performance characteristics for their products. Even if dynamic control is possible, many users will prefer to work at the higher, kinematic level.

Typically, as is the case with the Adept Cobra 600 robot [32, p. 192], specifications are provided in terms of maximum joint rates and payloads. When developing a kinematic trajectory control technique, one can assume that the manufacturer's

controller will be able to perform the low-level dynamic control, as long as these specifications are respected.

Several kinematics approaches have also been proposed for solving the TOTP problem. In [33] and [34], robot paths are represented using cubic splines and quintic polynomials, respectively. For both methods, velocity, acceleration, and jerk constraints can be imposed, though the techniques cannot properly deal with path discontinuities. Dong et al. [35] use a procedural approach that can properly handle path discontinuities by limiting the jerk and imposing deceleration to zero speed. In [36, 37], the TOTP problem is solved with the rapid prototyping application in mind. Only the technique used by Dong et al. can properly handle path discontinuities, however. Additionally, some applications require thousands or even millions of paths to be followed, and would render many of these techniques too computationally expensive.

Based on the advantages and drawbacks of the kinematics and dynamics techniques listed above for solving the time-optimal trajectory planning problem, we can list the characteristics of an ideal technique, designed for a general user of a positioning system. Firstly, implementation at the kinematic level should be available, to make it accessible to the widest range of potential users. Secondly, time-consuming solution techniques should be avoided, to make the technique suitable for applications where thousands, or even millions of paths will be followed. Lastly, the technique should be versatile: it should place no restrictions on path geometry and accept a wide range of constraints.

1.2.3 Geometric Feedback for Rapid Prototyping

Traditional material removal machining techniques, such as milling or drilling, normally produce parts with a much higher dimensional accuracy than do rapid prototyping (RP) systems, which are more difficult to control because they typically use

formable materials that accumulate in *free* space. This discrepancy is especially apparent for RP systems that construct a part layer-by-layer: any error in predicting the material layer thickness is amplified with each deposited layer, thereby producing a dimensional error in the vertical direction that is proportional to part height. As a result, all deposition-based RP systems will have an upper limit on the size of an object that can be built before unacceptable height error occurs. Geometric feedback for rapid prototyping can be defined as the detection and correction of geometric error during the construction of a part. Typically, height error will dominate any geometric error in the horizontal plane, due to the error amplification problem.

The correction of geometric error in rapid prototyping has been primarily addressed at the *process* level, where parameters such as temperature, pressure, material flow rate, and tool position have been stabilized. However, regulation of process level parameters only indirectly affects the part geometry, and does not address the error amplification problem. Doumanidis [38] developed a thermal feedback control model and implemented it on an RP system, where an infrared camera is used to modulate the power and motion of a plasma-arc torch in laminated object manufacturing. Mazumder et al. [39] limited the maximum height of deposited material in a direct metal deposition system by adjusting the melt pool volume based on CCD camera measurements of part height. Muscato et al. [40] developed geometric feedback control for shaped metal deposition, where wire feed rate is adjusted based on CCD camera measurement of part geometry. However, their method does not account for *local* part height variations: feed rate is varied once per layer, based on the average height error measured.

While implementation of closed-loop control at the process level helps to reduce the part error induced *per layer*, it neither maintains the nominal height over the whole surface of the part being built nor addresses the error amplification problem. Recently, Cohen and Lipson developed a geometric feedback control model for solid

free-form fabrication (SFF) systems that does correct local errors in part geometry [41]. However, their technique is designed for drop-on-demand systems, which deposit discrete droplets of material at specific locations. A claim is made that their model can be adapted to continuous-flow systems, which continuously deposit material through a tool that is usually in motion. Their technique is also limited to the flow control of the deposition material.

A feedback system is therefore needed, designed for continuous-flow systems, which can measure geometry errors and correct them through adjustment of any suitable parameter. Additionally, since the envisioned system is rather unique in the field of control systems, a rigorous analysis should be conducted to establish conditions under which it is stable, and a suitable optimization of system parameters should be conducted.

Many standard techniques exist for the stability analysis of control systems. For a single-input, single-output (SISO), linear time invariant (LTI), discrete control system, the Routh-Hurwitz stability criterion [42, 43] can be applied. For a discrete control system, the Jury stability criterion [44, p. 80] can be used, with the same restrictions. Unfortunately, a geometric feedback system is expected to have many unique features that make it unsuitable for direct application of either of these criteria. Careful analysis will thus be needed to cast the control equations in the appropriate form, and several important assumptions must be made and justified to regard such a system as SISO.

To optimize the geometric feedback system, traditional manual techniques such as the Ziegler-Nichols [45] method can be used. However, it is likely more appropriate to formulate the optimization as a minimization problem, where the objective is to minimize the total error during the construction of a part. Again, standard optimization techniques such as those introduced by Lopez et al. [46] can be used, though they will need to be adapted to geometric feedback.

1.2.4 Variable-Flow Control

Variable-flow control is necessary to synchronize deposition with variable-speed paths; it is also a potential means for correcting part geometry errors with surface mapping feedback (SMF), which is introduced in Ch. 5. Many variables affect liquid flow in the system, though few are suitable for implementing flow variation that is fast enough to be synchronized with rapid changes in the end-effector speed. For example, system pressure and nozzle diameter both greatly impact the flow rate but cannot be varied *precisely* and *quickly* enough. Other system characteristics, such as temperature, elevation change, and flow control devices such as ball valves, filters, and reducers, also minimally affect flow rate and are unsuitable. The most logical method for varying flow, both precisely and quickly, is active flow control with a valve.

Two main classes valve flow-control devices exist: analog or proportional, and digital or on/off [47, p. 211]. For analog control, the actuator varies the valve position with respect to the valve seat to vary the fluid resistance, and consequently, the flow rate. For digital control, rapid on/off switching of the valve affects the flow rate; this is analogous to the limitation of power supplied to an electrical load via rapid on/off switching. Since digital valve control offers the greatest control flexibility, and many off-the-shelf products offer suitable flow rate ranges, it is the best flow-control choice for rapid prototyping applications.

Digital control is implemented through manipulation of the valve duty cycle, the fraction of time that the valve is on. The most common technique used for varying the duty cycle is pulse-width modulation (PWM) [47, p. 211]. With PWM, the signal frequency is held constant while the pulse width is varied to produce the desired signal. Dobson and Kendall used PWM to control gas flow rates [48]. Yi et al. used PWM to control fuel flow rates and increase combustion efficiency in gas turbine

engines [49]. Additionally, manufacturers such as Spraying Systems Limited[2] and AutoJet Technologies[3] sell liquid flow control units that use PWM. For PWM flow control in rapid prototyping, however, the signal frequency used should be sufficiently high to prevent any *discrete* effects from becoming apparent.

A second option for controlling the duty cycle is frequency modulation (FM), most commonly-used to produce FM radio signals [50, p. 112]. For FM radio applications, a carrier wave at a fixed frequency is used to transmit an analog signal. However, a carrier wave is not needed for FM flow control. Instead, a constant pulse width can be used, with the instantaneous valve frequency being manipulated to achieve the desired flow control. FM can produce a larger flow rate range than PWM but off-the-shelf implementations of this technique are not available.

A final consideration for implementing variable flow is *online* vs. *offline* control. With online control, the instantaneous valve signal is generated while the the deposition path is followed by the positioning system. With offline control, the valve signal for an entire deposition path is produced at some time before the path is followed. The signal is stored in the valve control device memory prior to the start of a deposition path and then sent to the valve while the path is followed.

Nearly all flow-control strategies used in industry are online; a wealth of products are available for implementing this type of control. The advantage of online control is a simplified software implementation; most off-the-shelf products can be easily configured to output the desired variable-flow control signal. In practice, this leads to reduced application development time.

[2]http://uk.spray.com

[3]http://www.autojet.com

Valve control for dispensing applications, including rapid prototyping, is complicated by the need to synchronize flow rate with tool speed. Industrial robot manufacturers often provide an optional dispensing module for their controllers, specifically for this purpose. Adept Technology, Inc. offers a *Dispensing Module* for its AIM menu-driven interface [51]. With such a module, an analog signal, suitable for proportional flow control, can typically be generated that is proportional to the Cartesian speed at the dispensing tool tip. In some cases, a PWM signal suitable for digital valve control can also be generated.

The hardware and software needed for implementing this type of control is normally provided by a robot manufacturer as an add-on, which typically costs several thousand dollars. A second disadvantage to using this type of add-on is limited flexibility: normally, constant flow or flow proportional to tool speed are the only options. For applications such as rapid prototyping, it is desirable to have the ability to manipulate the instantaneous flow rate along a deposition path, as desired.

Given this consideration, *offline* control is the most suitable technique for a rapid prototyping system. The main drawback of this option is increased development time, since both custom hardware and software are needed. The greatest development challenge is the need to synchronize valve flow with tool positioning speed.

CHAPTER 2

Test Platform: The Cobra 600 Rapid Freeze Prototyping System

There are several important factors to consider when designing a novel platform such as a rapid prototyping system for ice. In general, when selecting the subcomponents to make up the system, one has to consider the conflicting objectives of minimizing cost and development time, and maximizing performance and versatility. The most important decision to be made is whether to retrofit a system *designed* for rapid prototyping, or to modify a more general-purpose positioning system for the rapid prototyping application. The former option will obviously take less development time, but will also offer less versatility. Both options will have a cost vs. performance tradeoff.

Based on these considerations, we initially retrofitted an inexpensive 3D printer, with the intention of developing a higher-performance, customized RFP system, based on the insight gained from the first system [8]. The Fab@Home desktop 3D printer, developed by researchers at Cornell University [52], was chosen for developing the initial system because it was inexpensive, at approximately $3000, and completely open-source, making it adaptable to extensive modifications.

The most significant modification needed to convert the FAH was the replacement of the fluid delivery system. The screw-driven syringe delivery system that was included with the FAH was suitable for *extrudable* materials but not for liquids. Additionally, this system used 10 cc syringes, which had to be *manually* changed. Accordingly, a valve-nozzle deposition system was developed, which is suitable for

precise deposition control of liquids, and for which the material volume deliverable is only limited by the reservoir size.

Based on our experience with the FAH RFP system, the drawbacks below were identified, to be addressed during the development of a second-generation, higher-performance RFP system:

- small workspace ($200 \times 200 \times 135$ mm, or 5.4L)
- low end effector speed (15 mm/s)
- open-loop positioning system using stepper motors
- prone to hardware and software failure
- slicing code limitations
  - slicing code fails for many parts
  - scaffolding paths are not generated *automatically*
  - graphical interface, which makes the software unacceptably slow for complex geometries

In addressing these issues, the major consideration, again, was whether to retrofit an existing RP system or to use a more general-purpose positioning system. For the first option, many potential candidates exist. In general, these can be divided into three categories: low cost (<\$5000), medium-cost (\$5000–\$50 000), and high-cost (>\$50 000). High-cost systems were immediately ruled out because they were beyond the budgetary constraints of the research program. Medium-cost systems, such as the ProJet 1500 by 3D Systems, Inc., or the Monolith by Acme Design Co., offer reliable, accurate, high-performance systems. The disadvantage of these systems is that they are mostly *closed-source*, which significantly limits the hardware and software modifications that can be performed. This might not be a problem if a partnership were formed with a manufacturer and access to the closed-source material was allowed. From an academic perspective, however, this is undesirable because it would likely restrict the dissemination of research results. Low-cost RP systems such

as the RepRap and the MakerBot are also unsuitable, because they exhibit many of the same drawbacks as the FAH system.

Given the considerations listed above, the use of a general-purpose positioning system as the development platform for RFP becomes an attractive option, despite the longer development time needed. Since nearly all rapid prototyping systems use a Cartesian or gantry three-axis architecture, this would be the logical choice. However, another important consideration is the desirability to use a regular chest freezer as the deposition environment. Most standard gantry configurations would require placing the entire positioning system inside the freezer, which is undesirable because it limits the available workspace and exposes the moving components of the system to the harsh freezing environment. These problems could be averted by using a gantry system with the two horizontal axes mounted immediately above the freezer opening, and the vertical axis extending into the freezer. However, this would be a highly-customized positioning system, and would require extra development time.

Another option is to use a revolute manipulator for positioning. No examples were found in the literature of a revolute manipulator used for layer-based rapid prototyping, but Hartmann et al. used a five-revolute manipulator for shape deposition, which was one step of an integrated manufacturing process [53]. For rapid prototyping, a minimimum of three axes are needed for positioning. However, common three-revolute (3R) spatial manipulators do not maintain a constant, vertical end-effector orientation, which is desirable for rapid prototyping. An RRP manipulator, where P stands for prismatic, would provide the minimum positioning and orientation control, but such a manipulator is not commonly available. However, a closely-related architecture, the RRPR serial robot arm, also referred to as a serial Selective Compliance Assembly Robot Arm (SCARA), is one of the most common manipulators on the market. A SCARA system provides three-dimensional positioning, while maintaining

a vertical end effector orientation. Additionally, the last revolute joint provides orientation in the horizontal plane, which could be used for many purposes, including extending the robot reach to expand the workspace.

The main advantage of using an RRPR manipulator for RFP is the ability to install the robot outside the freezer, without customization of the positioning system. The horizontal positioning is accomplished by the first two revolute joints, above the freezer opening, while vertical positioning is accomplished with the prismatic joint, which extends into the freezer. The choice of a SCARA system for positioning is also noteworthy because it represents the novel application of a robot commonly-used for pick-and-place tasks.

The Adept Cobra 600 RRPR manipulator, along with the C40 compact controller, were selected as the positioning system for RFP. With a 600 mm reach and a 210 mm vertical stroke, the Cobra 600 provides a 51 L usable workspace for RFP, as shown in Fig. 2–1. The Cobra 600 can reach speeds in excess of 1 m/s, using closed-loop, servo control for positioning. One disadvantage of using the C40 compact controller is that access to full dynamic control is not made available by the manufacturer. A customized robot controller could have been used to achieve this control, but this advantage was judged to be less important than the drawback of adding yet another custom component to the RFP system, with the associated extra development time. Even with the C40 compact controller, sophisticated kinematic trajectory control can be accomplished, as explained in Ch. 4.

The Cobra 600 offers versatility in addition to high performance. Standard communication interfaces, such as RS-232 and Ethernet, are available, and digital signals can be used for low-level control of peripheral devices. As mentioned above, the main drawback of using the Cobra 600 is the considerable development time needed to retrofit it for RFP. This drawback can be viewed as an advantage, however, since

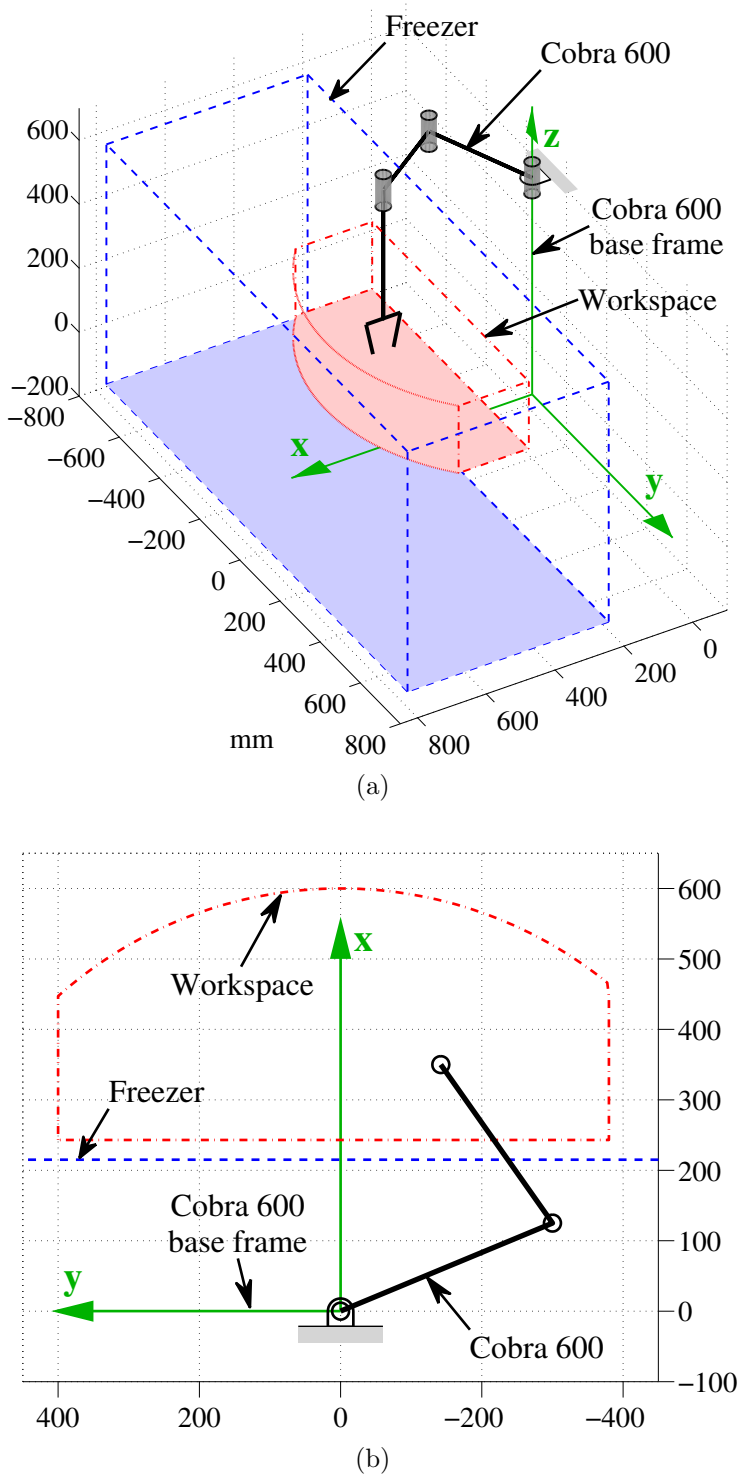


Figure 2–1: The Cobra 600 RFP system workspace, with dimensions in mm: (a) 3D view; (b) horizontal projection of the workspace, as generated by the first two R joints

custom solutions can be pursued for slicing, trajectory control, and valve control, all tailored specifically for RFP.

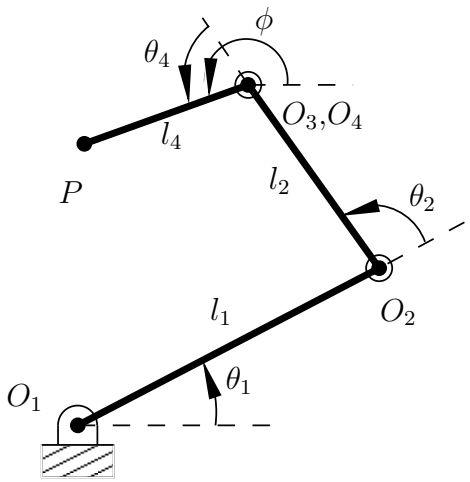


Figure 2–2: The horizontal projection of a RRPR manipulator, with the P joint vertical, physically colocated in the vertical axis of O_3, point O_4 travelling fixed to the EE

Some relevant specifications for the Cobra 600, obtained from the Adept instruction handbook [32, p. 192], are included in Table 2–1. A diagram of the manipulator geometry is shown in Fig. 2–2; the variables shown are defined in the Abbreviations list.

The Cobra 600 RFP system comprises several subsystems: fluid delivery, substrate levelling, and electrical and electronic subsystems are introduced in the first three sections of this chapter. Then, the scaffolding material is described and an overview of the part construction procedure is provided. In addition to the subsystems described here, a detailed bill of materials, along with a software configuration for the Control PC, can be found in the Cobra 600 RFP system User's Manual [9].

A customized end-effector (EE) was produced for housing the fluid lines, heating coils, electronics, and laser displacement sensor. The EE is a hollow aluminum tube, 300 mm long and 38 mm in diameter; it attaches to the Cobra 600 user flange. Manufacturing drawings for the EE are included in the User's Manual [9].

Table 2–1: Adept Cobra 600 Robot Specifications

Parameter	Value
Maximum reach	600 mm
Minimum reach	163 mm
Vertical stroke	210 mm
Joint rotation	
Joint 1	$\pm 105^\circ$
Joint 2	$\pm 150^\circ$
Joint 4	$\pm 360^\circ$
Maximum payload	5.5 kg
Maximum inertia about Joint 4	450 kg·cm^2
Resolution per encoder count	
Joint 1	0.00045°
Joint 2	0.00072°
Joint 3	0.0015 mm
Joint 4	0.03125°
Repeatability	
X,Y plane	0.02 mm
Joint 3	0.01 mm
Joint 4	0.03°
Maximum joint speed with 2 kg payload	
Joint 1	360°/s
Joint 2	672°/s
Joint 3	1100 mm/s
Joint 4	1200°/s

2.1 Fluid Delivery Subsystem

The fluid delivery subsystem, shown in Fig. 2–3, supplies water and shortening methyl ester (SME) scaffolding to be deposited. Relevant specifications for this system are shown in Table 2–2. SME is a type of biodiesel fuel and is incompatible with many commonly-used plumbing materials. Therefore, brass components, such as valves, filters, and other connectors, are replaced by stainless steel; and polyurethane tubing is replaced by tygothane tubing; natural rubber (buna-N) seals are replaced by viton or fluroelastomer seals. The pressure of each dispensing tank is individually regulated, though both are pressurized using the same air tank.

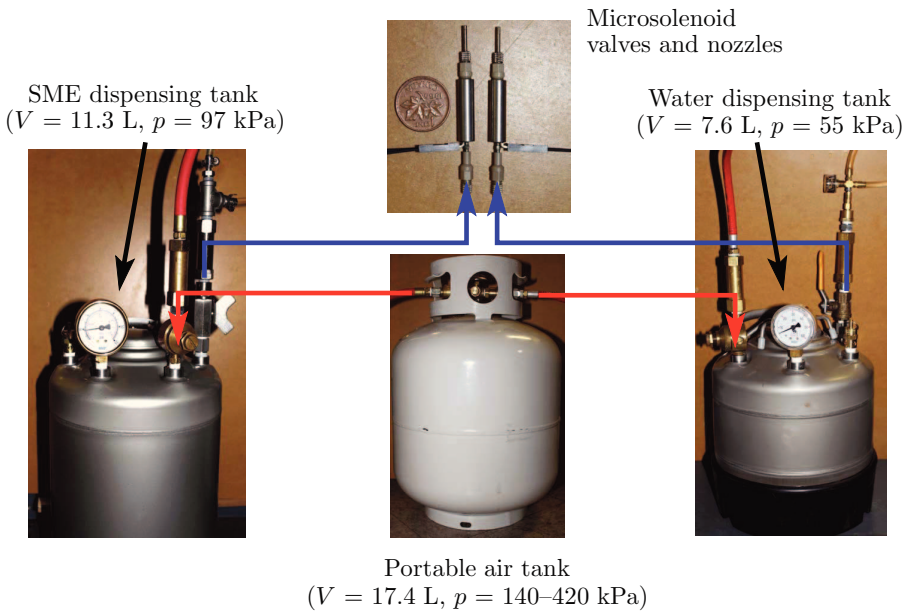


Figure 2–3: The fluid delivery subsystem

Three flow variables have a significant impact on the flow rate through the valves: line pressure, nozzle diameter, and valve control signal. Other variables, such as tubing diameter, filters, and elevation change, have minimal impact. The flow variables are configured to achieve a compromise among several desirable features, including maximum flow rate, flow rate resolution, and flow geometry. If the pressure is increased much beyond the values indicated in Table 2–2, the jets of fluid will splash upon hitting the deposition surface. On the other hand, if the pressure is too low, the fluid will not flow out of the nozzle as a uniform jet. A similar flow rate range is achieved with the two systems by compensating for higher SME fluid viscosity with higher tank pressure and increased nozzle diameter. Kinematic viscosity values in Table 2–2 were obtained from [54] and [55, p. 13]. No viscosity value was found specifically for SME, though values for Soy ME, Canola ME, Lard ME, and Tallow ME all lie within the range 4.5–4.9 mm^2/s @ 40°C. These MEs contain many of the same constituent fatty acids as SME.

Table 2–2: Properties of the water and SME dispensing systems

Parameter	Water	SME
Tank pressure (kPa)	55	97
Nozzle diameter (mm)	0.191	0.254
Kinematic viscosity @ 40°C (mm^2/s)	0.685	4.7
Maximum flow rate (μL/sec)	200	200

Figure 2–4: Levelling table

2.2 Levelling Table

The Cobra 600 RFP system accuracy is approximately 0.5 mm when the geometric feedback system, described in Ch. 5, is applied. The deposition surface or substrate flatness tolerance should be smaller than this value, such that it does not contribute noticeably to the part construction error. In practice, we have found that a flatness tolerance of approximately 0.25 mm can be achieved using the calibration procedure described below.

A levelling table, shown in Fig. 2–4, was designed and built for the Cobra 600 RFP system, measuring approximately 900×400 mm, which covers the entire RFP system workspace shown in Fig. 2–1. A substrate clamped to the table has a flatness tolerance of approximately 0.2 mm. Since the table is permanently located in a chest freezer at -20°C, it was built using only aluminum or stainless steel components, to avoid rust formation.

Table 2–3: Specifications of the laser displacement system

Parameter	Value
Reference distance[a]	30 mm
Measurement range	±5 mm
Spot diameter	30 μm
Sampling period	20–1000 μs
Operating temp.	0-50°C
RS-232C com.	115200 bit/s
Wavelength	650 nm
Output	0.95 mW
Linearity	± 0.05% of measurement range
Repeatability	0.05 μm
Controller Memory	65536 measurements

[a]Distance from the lower face of the laser to the center of the measurement range

Table flatness calibration is accomplished as follows. The Keyence laser LK-G32 displacement sensor, which is mounted on the end effector, is used to provide table height measurements. This sensor is also used as the measurement device for the geometric feedback system described in Ch. 5; detailed specifications are listed in Table 2–3. The Keyence LK-Navigator software, which accesses the USB connection of the laser controller, is used to observe the distance measured with the laser head.

Coarse calibration is achieved by positioning the robot arm at approximately nine equally-spaced locations in the robot workspace and turning the four swivelling feet fastened to the base of the table until all observed heights lie within 0.5 mm of each other. Four more levelling feet are then used to secure the table to the freezer walls. Fine calibration is accomplished by measuring at the same nine locations, approximately, and adjusting the nine spring-loaded screws that attach the floating aluminum plate to a fixed plate below, until all observed heights lie within 0.2 mm of each other.

2.3 Electrical and Electronic Subsystems

The main signal traffic during part construction for the Cobra 600 RFP system is shown in Fig. 2–5, with most of the electrical and electronic subsystems shown in

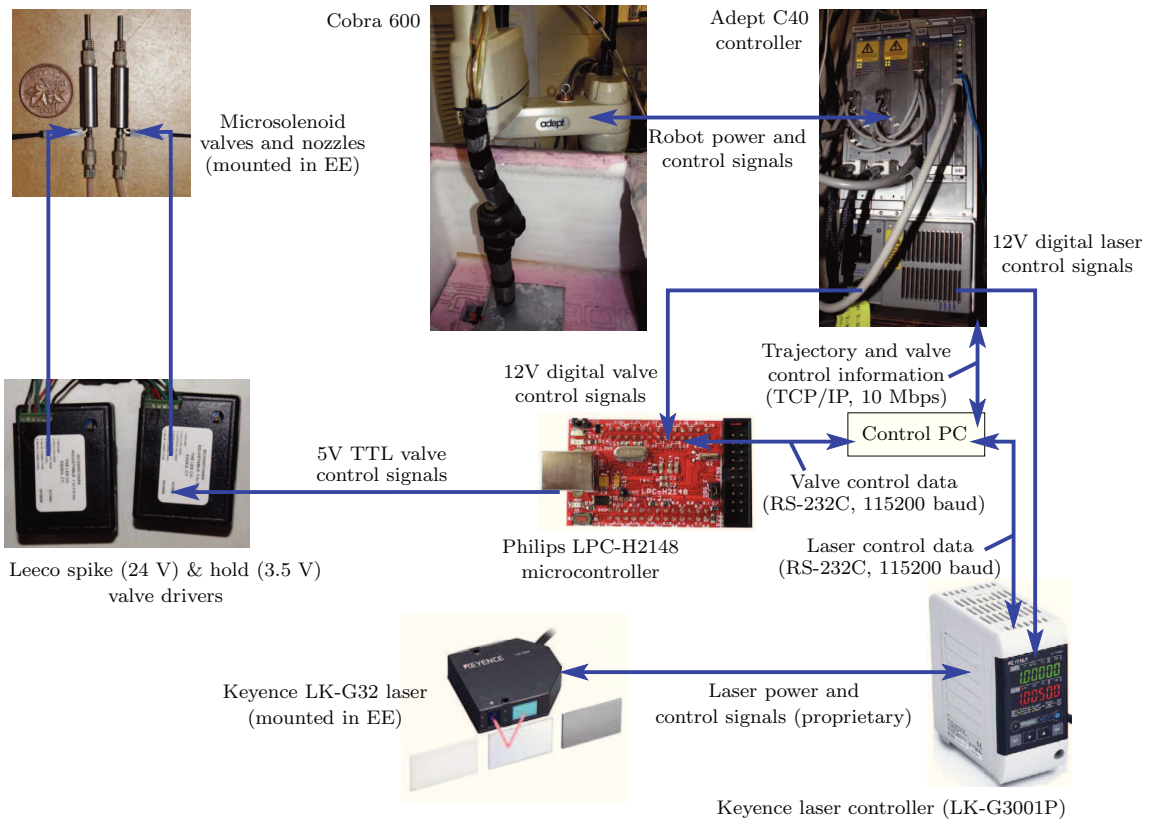


Figure 2–5: Signal traffic during part construction

Fig. 2–6. The system is configured to exploit the computing capabilities of the Control PC, the Adept C40 controller, and the Philips LPC-H2148 microcontroller. Most importantly, real-time control signals are all output from the Adept C40 controller, and not from the Control PC; this ensures precise synchronization between positioning and peripheral devices. Conversely, the C40 controller is ill-equipped to handle the communication, storage, and computational requirements of rapid prototyping; therefore, data-intensive operations are performed on the Control PC, whenever possible. As shown in Fig. 2–5, the laser controller and the valve microcontroller both receive control signals from C40 controller, but they perform two-way data communication with the Control PC.

Figure 2–6: Electrical and electronic subsystems

2.3.1 Microsolenoid Valve Control

To implement the *offline* variable-flow control described in Subsec. 1.2.4, a valve microcontroller is needed that has the features listed below, where the specific feature for the chosen microcontroller, the Philips LPC-H2148, is in included in parentheses:

- versatile, yet common firmware programming language (C)
- sufficient memory to store control data for one deposition path (32 KB)
- standard communication interface (UART0)
- digital input/output signals (3.3V CMOS)

The hardware solution for valve control is introduced here; the software solution, used to generate the digital waveform for each deposition path, is discussed in Ch. 6.

The first requirement is a very important consideration. Many programmable microcontrollers sacrifice versatility for ease of use, using a custom programming language that has a small set of simple commands, which enables novice users to master device operation quickly. Such a system, the BasicStamp BSP2PX microcontroller, was initially used for the Cobra 600 RFP system, but was abandoned because it was too restrictive for variable-flow valve control implementation.

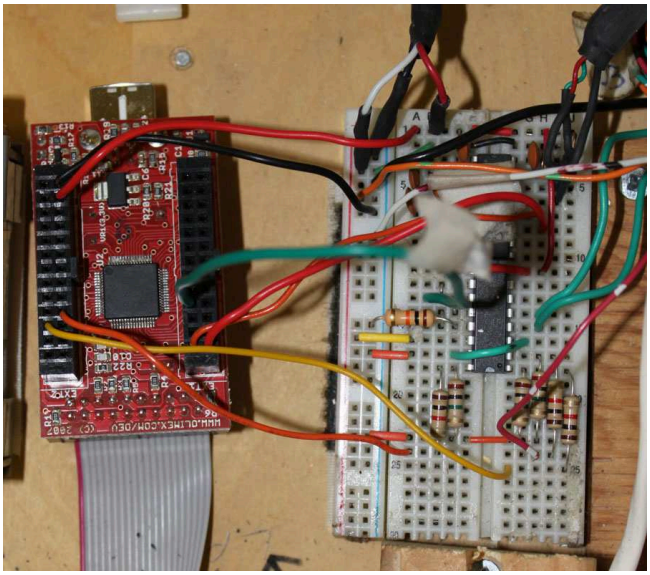


Figure 2–7: Valve microcontroller

The LPC-H2148, shown in Fig. 2–7, was chosen mainly for programming versatility: with C, the microcontroller can be programmed precisely for the desired purpose; additionally, since C is one of the most common programming languages available, a wealth of information is available on programming techniques. The downside of using the LPC-H2148 is its high demands on programming knowledge. Any user-developed firmware must contain explicit configuration of the communication interface, the timers, and any other features to be used. For example, simple operations such as reading or writing *any* data type using the UART0 interface must be explicitly programmed as subroutines in the firmware. Therefore, using the LPC-H2148 requires considerably more development time compared to more user-friendly options.

Custom firmware was written for the LPC-H2148 to configure it for use as the Cobra 600 RFP system valve microcontroller. Firmware execution begins with system initialization; then the code follows an infinite loop, where each loop iteration corresponds to one deposition path followed by the Cobra 600 RFP system.

The main system configuration for the LPC-H2148 is identical to the example firmware BLINKY, provided by Philips, except two pins are configured for output

signals. UART0 is then configured to operate at 115200 baud by writing to several registers. Interrupts are initialized in the same way as in BLINKY. Then, a 10-second watchdog timer is configured and initialized; the microcontroller reboots unless the watchdog timer register is written to at least once every 10 seconds. This is a fail-safe feature that prevents the code execution from stalling, for whatever reason. It can also be used as a passive means to reboot the microcontroller. As a last step of initialization, a system timer is configured for valve pulse timing.

After initialization, the main code loop repeats infinitely, until power is lost or the watchdog timer expires. For each loop iteration, the firmware continually checks the LPC-H2148 UART0 input buffer for a specific byte sequence that indicates path data are ready to be transferred. Then, the data are sent from the Control PC to the LPC-H2148 and the two devices communicate to ensure the correct data has been transferred. Next, the code pauses until a trigger signal is received from the C40 controller, which ensures that valve control is synchronized with positioning.

After this trigger signal is received, material dispensing and robot positioning proceed *independently* for the entire deposition path, which can last up to 131 seconds. Time synchronization is maintained to within approximately 5 ms. Robot position timing is controlled by specifying the point-to-point duration for all points along a deposition path, using minimum-time trajectory shaping (MTTS), as explained in Ch. 4. Valve control timing is accomplished by using available timers on the LPC-H2148, which can be programmed to smaller than μs precision. For each deposition path, valve signal control data are stored as an array of unsigned 16-bit integers, where each entry of the array represents the elapsed time from the start of the path, at which a single TTL pulse is to be output. The chosen data precision is a compromise between storage efficiency and time resolution. With the current configuration, the time-resolution is 2 ms, the maximum time of travel for an entire path is 131.072 seconds, and the maximum array size is 16 384 16-bit entries. If valve signal control data exceed

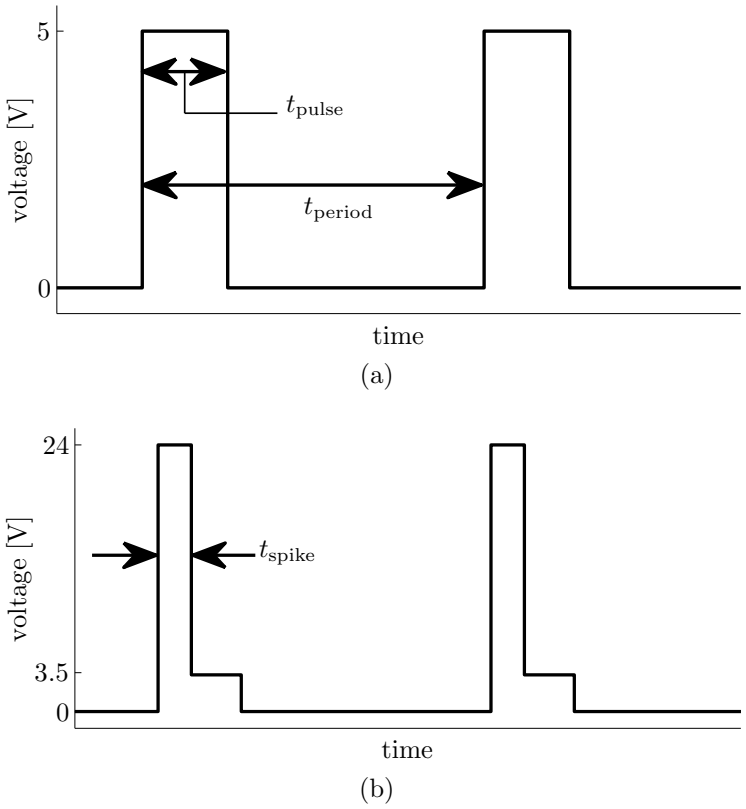


Figure 2–8: Microsolenoid valve control signal: (a) TTL control signal; (b) Associated valve-driving spike-and-hold signal

either of these limits, paths are automatically split into pieces by MTTS, though this rarely occurs.

Spike-and-hold drivers receive the valve control signal from the LPC-H2148 and convert it into a signal suitable for driving the valves; the TTL and spike-and-hold waveform shapes are shown in Fig. 2–8. More information about these signals is provided in Sec. 6.1, where the procedure used to generate the digital waveform for each deposition path is discussed.

Logic level conversion is an important consideration when configuring the communication and control signals passing among the devices shown in Fig. 2–5, where four different logic levels are present. When a device receives an incompatible logic signal, it might not respond properly and could be damaged.

Table 2–4: Logic levels for Cobra 600 RFP system components

Device	Logic Family	Output High (V)	Output Low (V)	Input Range High (V)	Input Range Low (V)
PC-serial port	RS-232	−10.0	+10.0	−15 to −3	+3 to +15
LPC-H2148	CMOS[a]	+3.3	0.0	+2 to +5	0 to +0.8
C40 controller	Digital	+12.0	+0.8	+10 to +24	0 to +3
S&H drivers	TTL	+5.0	0.0	+2 to +5	0 to +0.8

[a]The LPC-H2148 accepts TTL and CMOS logic levels for input signals.

Figure 2–9 shows the electronic configuration used to accomplish the desired logic level conversion. Two integrated circuit (IC) chips are needed to perform conversion among the LPC-H2148, the Adept C40 controller, the control PC, and the valve spike and hold drivers. The MAX232 is used to convert between PC RS232 logic levels and UART0 logic levels used by the LPC-H2148. The ADG509A is used to convert logic levels among the LPC-H2148, the Adept C40 controller, and the valve spike-and-hold drivers. The specific logic levels used by each device are listed in Table 2–4.

2.3.2 Laser Displacement Sensor Control

The laser displacement sensor system, manufactured by Keyence, is used primarily as the measurement device in the geometric feedback system, described in Ch. 5. Detailed specifications of the device are listed in Table 2–3. The system consists of LK-G32 sensor, connected by the LK-GC5 cable to the LK-G3001P laser controller. During measurement, a 24-V trigger signal from the Adept C40 controller is used to synchronize laser measurements with positioning. After each path is followed, laser height measurements are sent from the laser controller to the Control PC via the RS-232 (serial) interface.

The laser controller can also be controlled through its USB port using the Keyence LK-Navigator software. This software can be useful for laser calibration and testing, since it provides a graphical user interface and several diagnostic features that are not available through RS-232. Currently, this software is used for fine calibration of the levelling table, as described in Sec. 2.2. It would also be desirable to use USB control

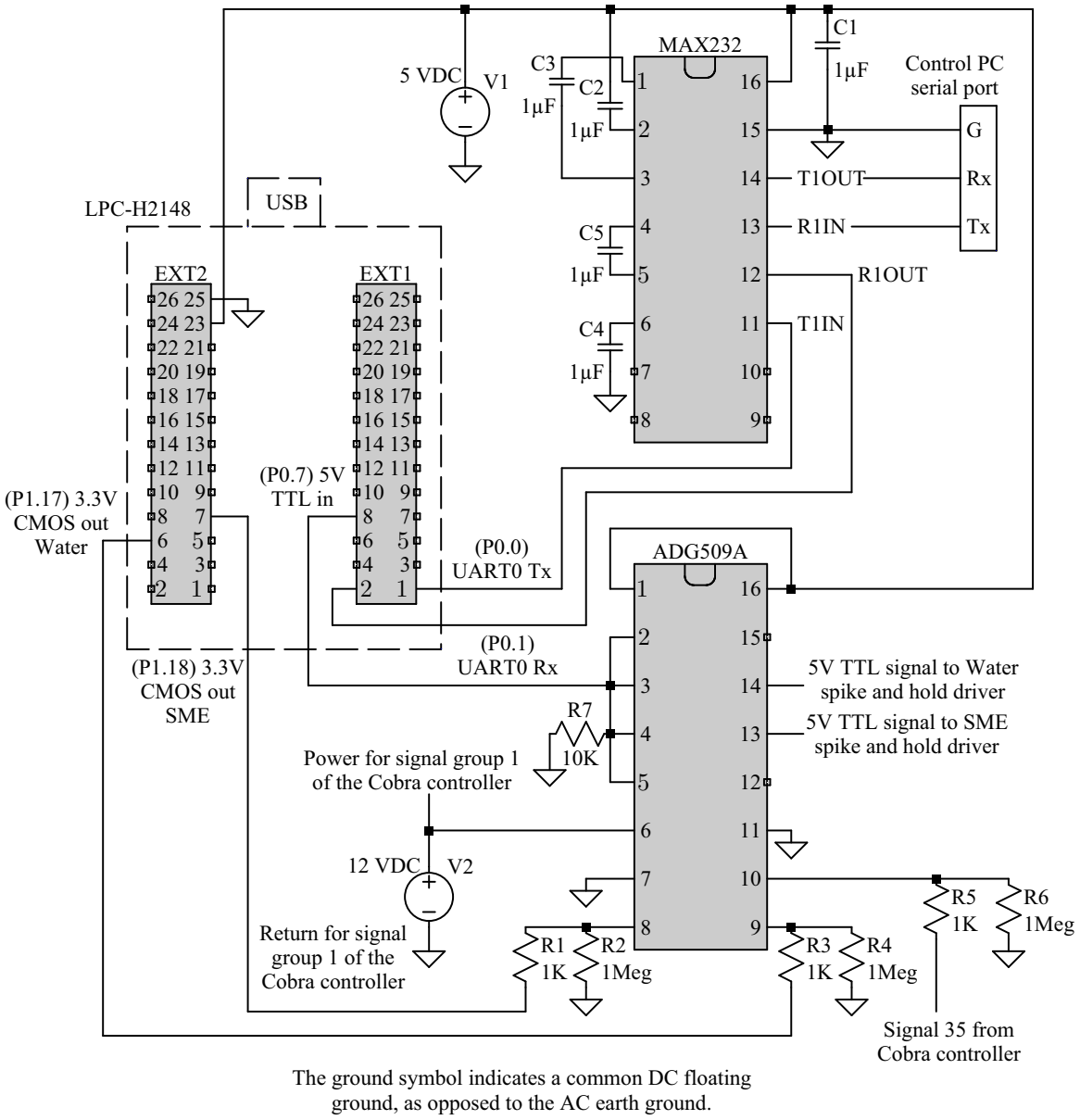


Figure 2–9: Valve controller wiring diagram

for laser operation during part construction, since the rate of USB communication is 12 Mbit/s, while the RS-232 rate is only 115 200 bit/s. However, USB control cannot be *automated* at this time, because it only accepts commands from the graphical user interface. Upon consultation with Keyence engineers, USB control could be adapted for automated operation, using the same commands as those used for RS-232, if a new hardware driver were developed.

2.3.3 Temperature Control

Temperature control is achieved with three ON/OFF temperature controllers, each using a thermocouple for temperature measurement and resistance heating rope for heating. The temperature controller units are powered by regular 120 VAC power; the heating coils can be driven by any voltage, AC or DC, from 0–120 V. The following three sections of the end effector are temperature-controlled: (a) the end-effector tip, with 36 V applied to the heating coil; (b) the laser emission location, with 24 V applied to the heating coil; and (c) at the joint where the flexible cable conduit intersects the rigid EE, with 36 V applied to the heating coil. System (b) is used for redundancy, since the laser is the most valuable system component, after the robot itself; additionally, if system (a) fails, system (b) can still produce enough heat to prevent the microsolenoid valves from freezing.

During fluid deposition, system (a) is off most of the time because the microsolenoid valves produce a large amount of heat. During laser measurements, however, the microsolenoid valves are off and the (a) heating coils are used much more heavily. Therefore, system (c) is added to prevent significant temperature variation and freezing of the fluid supply lines.

The heating rope would degrade and eventually break if exposed to the scaffolding material, shortening methyl ester (SME). Therefore, SME-compatible Kynar heatshrink tubing is used to protect the rope.

2.4 The Scaffolding Material

Scaffolding is needed to support objects with features that overhang free space, i.e., if any free space is intersected when projecting the object *downward*. An ideal scaffolding material would have the features listed below:

- inexpensive (i.e., less than $10/kg)
- easily removable after construction is completed
- does not *degrade* the ice part during construction or during scaffolding removal

- viscosity and liquid-solid phase change temperature are close to those of water
- non-flammable, environmentally-friendly, and poses no health risks

The most obvious choice for an RFP scaffolding material is a water-based solution. Since all such solutions have a lower freezing point than that of pure water, they can be removed following construction by placing the part in an environment slightly below 0°C. A sodium chloride (NaCl) salt solution was initially used as the scaffolding material for the Cobra 600 RFP system, but was observed to significantly *degrade* ice parts, due to diffusion during construction and melting at the scaffolding-part interface during scaffolding removal. This degradation is unsurprising, given that salt is commonly used to melt ice and snow in the winter.

The diffusion and melting problems described above would exist for any water-based solution, though they can be minimized by selecting a less active solute such as sucrose [4]. However, both problems can be eliminated by selecting an oil-based material. Given the desired scaffolding characteristics listed above, vegetable oils are potential candidates. These, however, have a viscosity of approximately forty times that of water at room temperature. At this viscosity, the flow rate transition from dripping to jetting is too high for the selected microsolenoid valves. Fortunately, the viscosity of any vegetable oil can be lowered sufficiently by using a process called transesterification, commonly used to produce biodiesel [56].

The liquid-solid phase change temperature is used to select an appropriate vegetable oil. Common vegetable oils such as canola oil, olive oil, and shortenening, are made up of a combination of fatty acids, all with different melting points. In practice, this means there is no specific melting point but rather a gradual transition from liquid to solid. A cloud point is defined, at which the substance first starts to solidify; as the temperature is lowered further, the ME gradually becomes harder. An abrupt phase change would be preferred, and one of the constituent fatty acids could be isolated that possesses all of the desired phase change characteristics. However,

these chemicals are not commonly available and are much more expensive than the desired material cost.

Ideally, the selected ME should melt *below* 0°C, to facilitate scaffolding removal. However, the ME should also freeze *quickly* and remain relatively *stiff* at −20°C, the temperature in the freezer at which the water and scaffolding are deposited. After trying MEs such as olive oil methyl ester and canola oil methyl ester, shortening methyl ester (SME) was found to be the best compromise among the desired phase change characteristics. Unfortunately, the melting point of SME is about 5°C, so phase-change difference cannot be used for scaffolding removal. Instead, chemical removal is used, as explained in Sec. 2.5. Other ME's, such as olive oil ME and canola oil ME, do melt slightly below 0°C, but they are insufficiently stiff at −20°C; tall, thin scaffolding features built with these ME's would collapse under their own weight.

2.5 Part-Construction Process

The process used to construct an ice part with the Cobra 600 RFP system can be divided into the steps below:

1. The input for the ice construction process is a CAD part in the STL or PLY format. Based on the CAD part, the water and SME paths needed for construction are computed using RPSLICE, as explained in Ch. 3.
2. Deposition paths are used to produce positioning and valve control data using minimum-time trajectory shaping (MTTS), as explained in Ch. 4.
3. The part is constructed automatically; surface mapping feedback (SMF), is used to detect and correct geometry errors, as explained in Ch. 5.
4. Most of the scaffolding is removed *manually*, then melted, filtered and *re-used*.
5. The part is placed in kerosene at −5°C to dissolve the remnants of the scaffolding, a process that requires anywhere from a few hours to a few days.
6. The completed ice part is stored in kerosene at −25°C, to prevent sublimation and frost buildup, which can cause significant loss of detail over time.

The construction process for a 1/6-scale bronze miniature of the statue of James McGill, located at the McGill University downtown campus, is shown in Figs. 2–10 and 2–11. Several completed ice parts are shown in Figs. 2–12 and 2–13. For the parts shown in Fig. 2–13, the CAD files were obtained: from the Stanford 3D scanning repository for Lucy[1]; by following a modeling procedure on the MeshLab blog for the Voronoi sphere[2]; from Quality Transmission Components for the spiral bevel gear[3]; and from Dr. George Hart for the toroidal echinoderm[4].

[1]http://graphics.stanford.edu/data/3Dscanrep/

[2]http://meshlabstuff.blogspot.com/2009/03/creating-voronoi-sphere.html

[3]http://www.qtcgears.com/KHK/newgears/KHK206.html

[4]http://www.georgehart.com/rp/rp.html

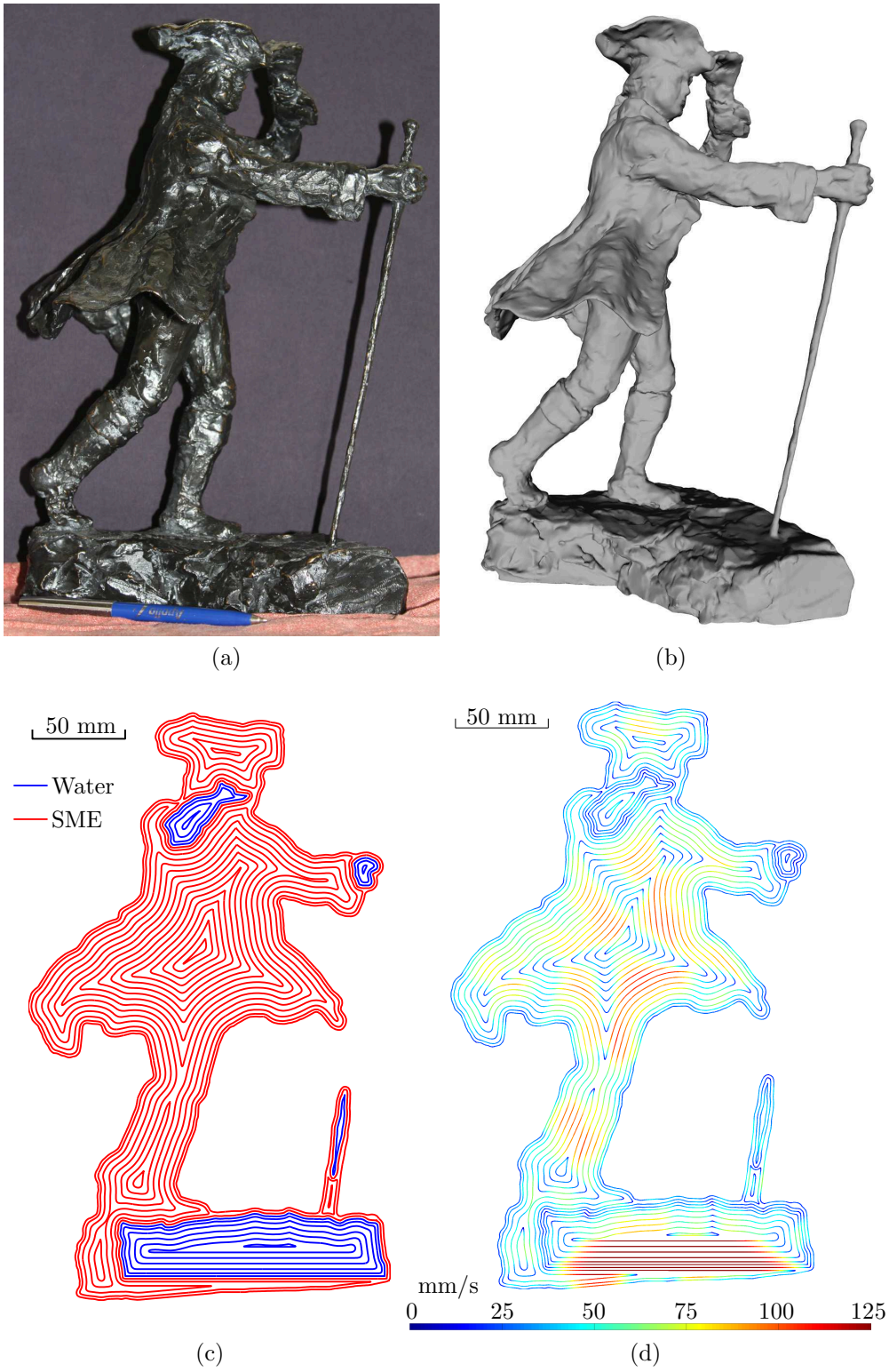
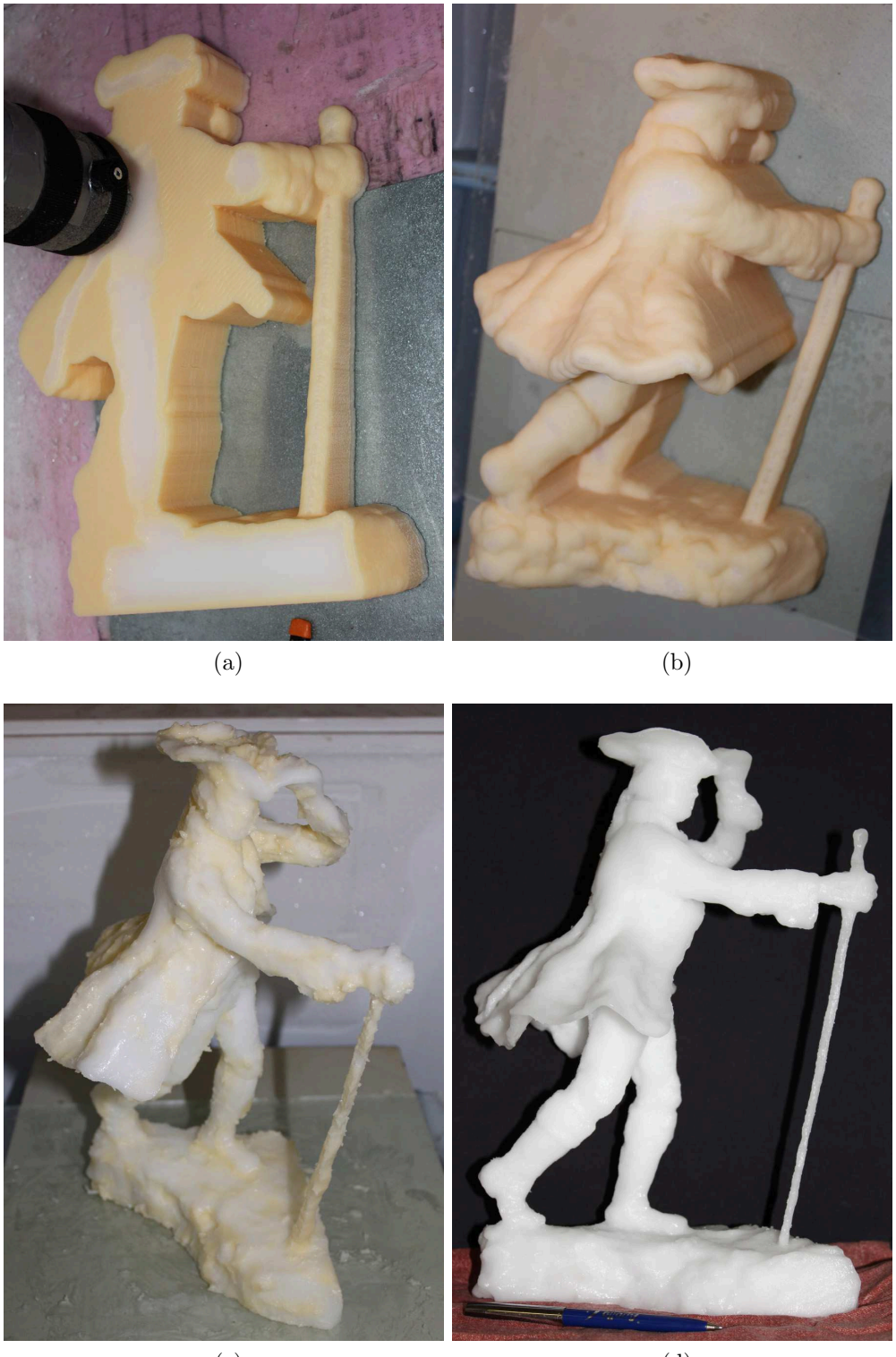


Figure 2–10: Construction process for the James McGill statue: (a) bronze statue, 30 cm high; (b) STL file, 1 million triangles; (c) deposition paths for one layer, 25 mm from the deposition surface; (d) robot end effector speeds along the paths



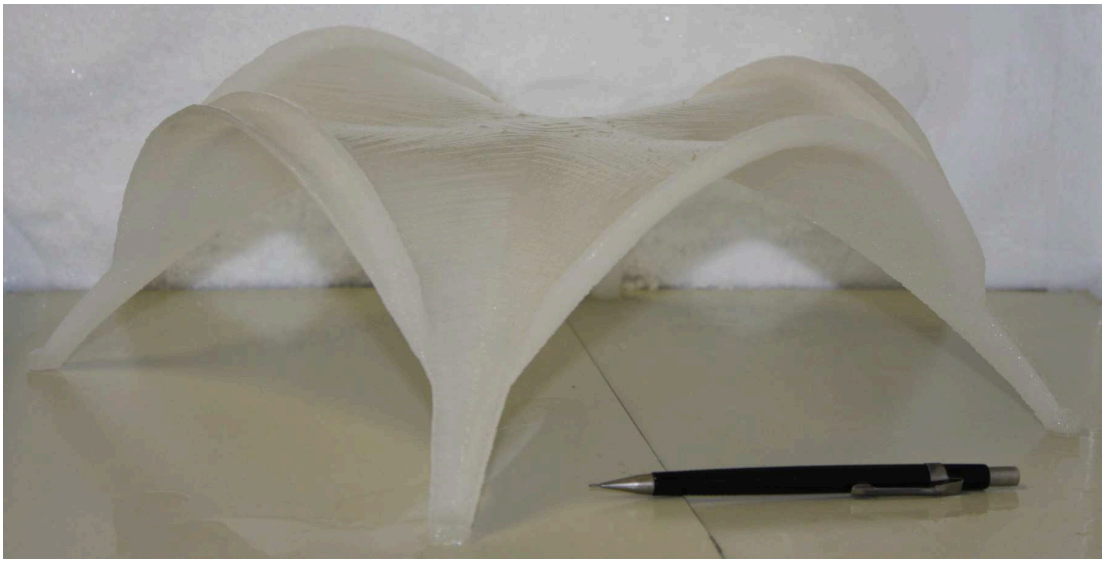
(a) (b)

(c) (d)

Figure 2–11: Construction process for the James McGill statue (continued): (a) with construction half-finished; (b) with construction complete; (c) after manual removal of most of the scaffolding; (d) after removal of the scaffolding remnants using kerosene

(a)

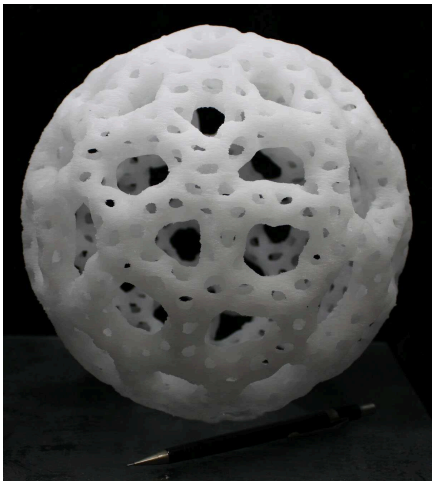
(b)



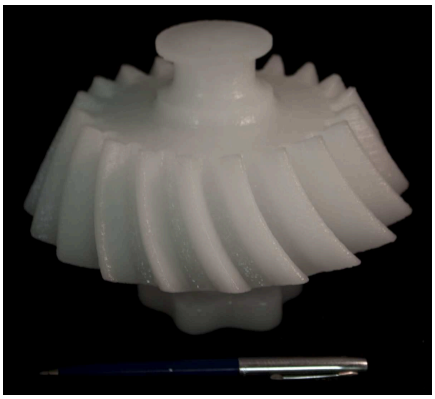
(c)

Figure 2–12: Ice parts built with the Cobra 600 RFP system: (a) McGill coat of arms (220 mm wide); (b) twisted Koch fractal structure (no scaffolding used); (c) 1:120 scale model of the Bacardi bottling plant in Cuautitlán, Mexico, a hyperbolic paraboloid structure designed by Félix Candela

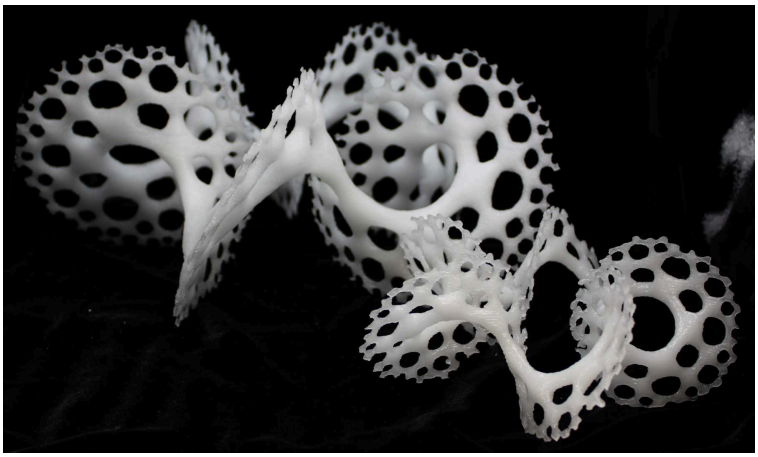
(a)



(b)



(c)



(d)

Figure 2–13: Ice parts built with the Cobra 600 RFP system (continued): (a) Lucy, a Christian angel (large model is 450 mm high); (b) Voronoi sphere; (c) spiral bevel gear; (d) toroidal echinoderm (large model is 300 mm in diameter)

CHAPTER 3
Part-Slicing Algorithm

A rapid prototyping system should include a software solution, which we will refer to as a slicing algorithm, for converting 3D CAD data into layers of 1D deposition paths. Ideally, this algorithm should be capable of generating the deposition paths needed to construct *any* 3D geometry, within the limitations of the positioning system and the deposition material properties. Of course, 3D parts can also be constructed by programming robot paths *directly*, which can be preferable for certain geometries. For example, the paths used to construct the part shown in Fig. 2–12(b) were produced by modifying a Koch fractal algorithm.

Many of the factors that were considered when choosing the development platform, as described at the start of Ch. 2, are also relevant in selecting the software solution for the Cobra 600 RFP system. The code for an off-the-shelf RP system could be adapted, which would minimize the associated development time. Low-cost RP systems tend to include open-source software that can be customized but is quite limited. For example, the Fab@Home 3D printer does not automatically produce scaffolding path data, and frequently fails for certain part geometries. More expensive RP systems would likely produce higher-quality RP path data, and some of the customizations needed for RFP could be accomplished by post-processing these data. For example, path data could be readily offset for different deposition tools.

The slicing code for such a system would most likely be closed-source, however, which places too severe a restriction on the system versatility. Path data generated for a traditional RP system, which typically uses extrusion-based material deposition,

are not normally suitable for RFP, which uses valve-nozzle deposition. Fundamental slicing parameters that might need to be customized for RFP include boundary offsetting and fill path geometry. Additionally, the scaffolding solution used with many RP systems is not suitable for RFP. For example, many extrudable materials can be reliably cantilevered, which reduces the scaffolding volume required. For RFP, however, it is desirable to have the option of a full scaffolding solution, where scaffolding will fill the entire volume intersected when projecting the part downward. Finally, a customized algorithm is preferable because the requirements for the algorithm are expected to evolve during the development of a novel platform like the Cobra 600 RFP system.

Given the considerations listed above, a custom slicing code was developed, called RPSLICE [10], which is an improved version of a previous Matlab code [21]. RPSLICE is not adapted specifically for the Cobra 600; it can be used to generate control trajectories for other rapid prototyping systems. The previous code produced self-intersecting paths for some parts with complex geometries; this problem has been eliminated with RPSLICE through the implementation of a new path buffering technique. Additionally, the data storage and processing efficiency have improved considerably, allowing parts with a high level of detail to be sliced.

Part slicing involves several steps, although literature on the subject typically focuses on one of the steps [17–20]. Matlab is used as the coding environment for RPSLICE because it offers many useful functions and less development time compared to lower-level programming languages such as C or C++. Specifically, many functions available in the mapping toolbox are useful for rapid prototyping. The performance of RPSLICE is measured in terms of the quality of path data generated and the computational time required. All computations were performed with a computer that had the following specifications: Intel Core i7-2720QM, 4 CPUs @2.2 GHz (turbo 3.3 GHz), 16 GB RAM, 240 GB solid state drive. Additionally, all algorithm steps

(a) (b)

Figure 3–1: James McGill STL part: (a) One million facet model; (b) Decimated model with 3906 facets

are programmed to execute on multiple CPUs, unless otherwise indicated. In some cases, jobs are explicitly created for parallel execution, while in others, the Matlab parallel for loop PARFOR is used in a MATLABPOOL session. A CAD model of James McGill[1], shown in Fig. 3–1, is used to demonstrate the capabilities of RPSLICE.

3.1 Data Import and Transformation

The first step in RPSLICE is to import a control file, where several slicing parameters are specified; the main entries of this file, for the part shown in Fig. 3–1(a), are included in Table 3–1.

[1]James McGill (1744–1813) bequeathed an estate and funds in his will for the construction of McGill College (later McGill University), officially founded in 1821. A natural scale original statue of James McGill features prominently in the McGill downtown campus.

Table 3–1: Control parameters for RPSLICE

Parameter [units][a]	[Data format] (Data restrictions) Description
CAD PARTS	
1	[int] (> 0) Number of STL and/or PLY files to import
jamesmcgill.stl	[char char] STL and/or PLY file names; no spaces are allowed within each name)
COMPUTATION CONTROL	
4	number of CPUs to use in parallel computations
SLICING CONTROL	
1.5 1.5 [mm]	[float float] (> 0) [part scaffolding] deposition path width
0.25 [mm]	[float] (> 0) slice thickness
30 [°]	[float] (> 0) maximum angle between adjacent path segments that is considered to be *smooth*
3.3 [mm]	[float] Scaffolding buffer
PART TRANSFORMATION	
0 0 0 0	[1/0 float float float] (> 0) [(on/off) Sx Sy Sz] Scaling vector
1 -85 0 0 [°]	[1/0 float float float] [(on/off) Rx Ry Rz] Rotation vector
1 2 3	[int int int] (1–3) Rotation axis order
0 0 0 0 [mm]	[1/0 float float float][b] [(on/off) Tx Ty Tz] Translation vector
REFERENCE FRAMES	
260 -100 [mm]	[float float] [EEx EEy] origin of the end effector (EE) frame in the Cobra 600 base frame
-0.1 -0.3 [mm]	[float float] [Wx Wy] origin of the water building frame in the EE frame
0.2 8.2 [mm]	[float float] [SCx SCy] origin of the scaffolding building frame in the EE frame
-30 19 [mm]	[float float] [Mx My] origin of the laser measurement frame in the EE frame

[a] The parameters used for the James McGill ice statue STL file are shown here.
[b] There are three options for each axis: numerical entries [float] indicate the minimum coordinate for the part on an axis; [cent] entries will center the part on an axis; [orig] entries will keep the part centered at its original location on an axis.

3.1.1 Facet Data Importation With FACETREAD

PLY or STL facet data are imported with our FACETREAD function, which can read binary- or ASCII-formatted files. Table 3–2 shows the storage and processing efficiencies for different formats and settings, using the James McGill statue shown

Table 3–2: A comparison of different techniques for reading in the facet data for the part shown in Fig. 3–1(a)

File format	Technique	File size (MB)	Import time (s)
binary PLY	two FREAD statements	18.1	0.5
ASCII PLY	two FSCANF statements	34.9	5.3
binary STL	one FREAD statement	47.6	0.7
binary STL	FOR loop with one FREAD statement	47.6	21.6
ASCII STL	one FSCANF statement	257.8	28.6
ASCII STL	WHILE loop with multiple FSCANF statements	257.8	104.8

in Fig. 3–1(a) as an example. FACETREAD is one of the steps of RPSLICE that does not use parallel computing; parallelization would not improve performance in this case, since facet importation involves sequential disk reading.

It can be seen that ASCII files are considerably less efficient in both data-storage and processing time. Typically, programs will output ASCII STL files that store numbers using twelve bytes: seven significant figures, four characters for the exponent, and one for the decimal place. Binary STL files are typically output using single-precision floating-point numbers, which require only four bytes per number. Additionally, for STL files in the ASCII format, several text labels surround the data for each facet. Both of these factors contribute in making STL ASCII file sizes approximately five times larger than their binary counterparts. Reading in ASCII data also takes much longer, because file sizes are larger, data must be encoded into machine format, and data are stored in variable-width fields. The advantage of using the ASCII format is that it can be read directly by people, which can be useful for debugging. However, since the binary format is significantly more efficient in terms of processing time and data-storage, it should be used in almost all cases.

The PLY format also consists of triangular facets, which are stored much more efficiently. In the STL format, each facet is stored as nine floating-point numbers, which represent the three facet vertices in three-dimensional space. Since adjacent

facets share a common edge, every vertex is repeated several times in the file. In the PLY format, each unique vertex is stored as three floating-point numbers. Each facet is stored as three integers, which are the indices of its three vertices. Table 3–2 shows that the PLY format is significantly more storage-efficient than the STL format.

To exploit Matlab's vectorization capabilities and minimize import time, looping structures should be avoided when importing large amounts of data. For some of the formats described above, this can be difficult, since each facet contains several different data types, and in some cases, extraneous information. However, with Matlab's FSCANF and FREAD commands, used for ASCII and binary data, respectively, it is possible to skip unwanted information and read in all facet data with a single command.

Throughout RPSLICE, looping code structures are avoided whenever possible to reduce processing time; this is especially important when large, detailed models are sliced. Simple computations are vectorized, and more complex operations are accomplished with cell functions. Both of these techniques lead to significant reductions in processing time. However, they can also be highly memory-intensive, especially when cell functions are used. For some especially complex or memory-intensive operations, FOR loops or PARFOR loops are used.

3.1.2 Transformation of Facet Data

The RPSLICE algorithm translates, rotates, and scales the triangular facets based on the data imported from the control parameter text file. This feature is especially useful when a STL or PLY file is the only CAD model available. If rotation or scaling is desired, the model is initially moved to a location where its minimum value in each dimension is zero. Rotation is performed about one axis of a *space-fixed frame* at a time, in a user-defined order. This technique is used because rotations are typically defined based on a user's observations of the part orientation in the CAD

file. Normally, zero, one, or two rotations are required to obtain a near-optimal orientation of the part. *Optimal* typically refers to minimization of part-construction time, though other criteria may be more important in some cases. For example, part orientation might be configured to place more important features face-up, since this results in a better surface finish.

3.2 Part Boundary Paths

The boundary contours for each slice are formed using a function called FACETSLICE, composed of the three subfunctions INTERSECTFACET, SEGMERGE and PATHFILTER. Parallel processing is enabled for INTERSECTFACET, where the Matlab parallel for loop PARFOR is used, with each loop iteration corresponding to one layer or slice of the part. First, INTERSECTFACET is used to identify the facets that intersect each layer. Then, for each slice, the intersections between the horizontal plane and facet triangles are computed, forming an array of disconnected line segments. This is accomplished with a single command, rather than performing the plane-plane intersections one facet at a time. INTERSECTFACET accepts a matrix of facets of dimensions $n_{\text{facets}} \times 9$ and returns a matrix of line segments of dimensions $2n_{\text{facets}} \times 2$. The effect of this vectorization on computational time becomes more and more apparent as the number of facets increases; for the one-million facet James McGill CAD model, unvectorized INTERSECTFACET requires 28.5 seconds, while vectorized INTERSECTFACET requires 0.6 seconds.

The line segment matrix output by INTERSECTFACET is *unordered*, since the facets in an STL or PLY file are unordered. Additionally, the line segments, when linked to form the part boundary paths or contours, can form tens or even hundreds of separate polygons. A function called POLYMERGE exists in Matlab to link such a disconnected line segment array, though it is very memory-intensive and has a high computational cost. A new segment-linking function, called SEGMERGE, has been developed to form the part boundary paths iteratively, segment-by-segment: at each

Table 3–3: A comparison of functions for forming contours from segments with matching endpoints, for all 691 layers of the part in Fig. 3–1

Facets	Computational Time (s)			Segments Before PATHFILTER	Segments After PATHFILTER
	POLYMERGE	SEGMERGE	SEGMERGE2		
3906	3.3	0.8	0.5	118 604	112 044
1 000 000	521.8	25.9	7.0	1 739 712	302 332

iteration, the last point of the partially-completed polygon is matched with an endpoint of a segment in the disconnected array. A second function, called SEGMERGE2, has also been developed to further improve computational efficiency by pre-sorting the disconnected line segments and greatly reducing the number of searches needed, when compared to SEGMERGE.

SEGMERGE2 is therefore the preferred technique, though it will fail if a part contains geometry errors such as non-manifold edges, which occur when more than two facets are incident on an edge. Failure of SEGMERGE2 is detected automatically, and FACETSLICE reverts to SEGMERGE, which is less efficient but more robust. In fact, POLYMERGE and SEGMERGE are both more robust than SEGMERGE2, since they accept a tolerance variable, which is used to link matching segment endpoints that are not identical.

A comparison between POLYMERGE, SEGMERGE, and SEGMERGE2 is included in Table 3–3. For RFP, the model is sliced at increments of 0.25 mm, which produces 691 layers when the model is built on its side, as shown in Figs. 2–10 and 2–11.

Contour segment counts for POLYMERGE and SEGMERGE2 are identical; counts for SEGMERGE are slightly lower, most likely since it does not rely on *exact* matching of segment vertices.

The last step of FACETSLICE is the removal of unnecessary detail from boundary contour data by imposing a minimum point spacing of 0.4 mm, using a function called

PATHFILTER. Table 3–3 displays the number of path segments required, before and after this operation, for the part in Fig. 3–1. PATHFILTER is also vectorized such that all operations apply to *entire contours* and never to individual contour points. Since the speed of most subsequent steps of the RPSLICE algorithm is directly dependent on contour resolution, PATHFILTER has a significant impact on computational time.

3.3 Scaffolding Boundary Paths

During deposition, the RFP system alternates between part and scaffolding slices. It is possible to model the scaffolding with CAD software by "subtracting" the part from a block of material. However, this technique is not always straightforward and will usually result in substantial waste of scaffolding material, since support is only needed *below* overhanging features of the part. Additionally, if the part is only available in a surface rather than a solid format, this technique is difficult to implement. These factors motivate the development of a function that generates scaffolding contours based *only* on contour data output from FACETSLICE.

Chalasani et al. [57] developed a rapid prototyping technique in which they consider only the 2D contours in each slicing plane, while others [17, 58] consider the 3D model, analyzing and processing the triangular facets. Our RPSCAF function is similar to the first technique, with additional options included to accommodate the specific characteristics of the Cobra 600 RFP system.

Our RPSCAF function works in the following manner. A "merged" region is first formed for every layer, which consists of the Boolean union of all of the part layers above. The regions are formed one layer at a time, starting at the top of the part. Since each region depends on the region for the layer above, this process is constrained to run on a single CPU. Scaffolding regions are then formed for each layer *independently*, by computing the Boolean subtraction of the part region from

the merged region for that layer. Parallel jobs are created for this step, splitting computations among multiple CPUs.

This technique is fast and efficient, though it also produces many thin scaffolding features. Since thin features are built less accurately by the RFP system, the merged regions are buffered outward, by an amount specified in the input configuration file, before performing the Boolean subtraction. This is accomplished using the BUFFERF function, described in Sec. 3.4. All ice features in every slice are thus completely surrounded by scaffolding; obviously, this increases the amount of scaffolding material and construction time needed. However, it also significantly increases the part accuracy, since all ice boundaries are constrained by scaffolding. The feasible layer thickness is also higher, since the SME is hydrophobic and deposited water is therefore inhibited from spreading. Typically, with RFP, the maximum layer thickness is 0.25 mm when ice regions are completely surrounded by scaffolding; otherwise, this value reduces to 0.15–0.2 mm. In most cases, this means that part construction is considerably *faster* when the extra scaffolding is used.

Figure 3–2 shows the scaffolding boundary formation steps for one layer of the part of Fig. 3–1. In Fig. 3–2(b), part and scaffolding regions are shown *after* initial path buffering, described in Sec. 3.4.

3.4 Path Buffering

Path buffering, or offsetting, is used for two purposes with the Cobra 600 RFP system: (a) to adjust for the deposition path width at part boundaries; (b) to generate the fill paths. Since the path width w used with the Cobra 600 RFP system is 1.5 mm, if the path and scaffolding boundaries were used directly as deposition paths, an error of $w/2 = 0.75$ mm would be introduced. To avoid this error, all boundary paths are buffered inward by $w/2$ to form deposition paths. A custom contour-buffering technique was implemented in [21], but it was not computationally efficient, and produces self-intersecting paths for certain complex geometries. For RPSLICE, path

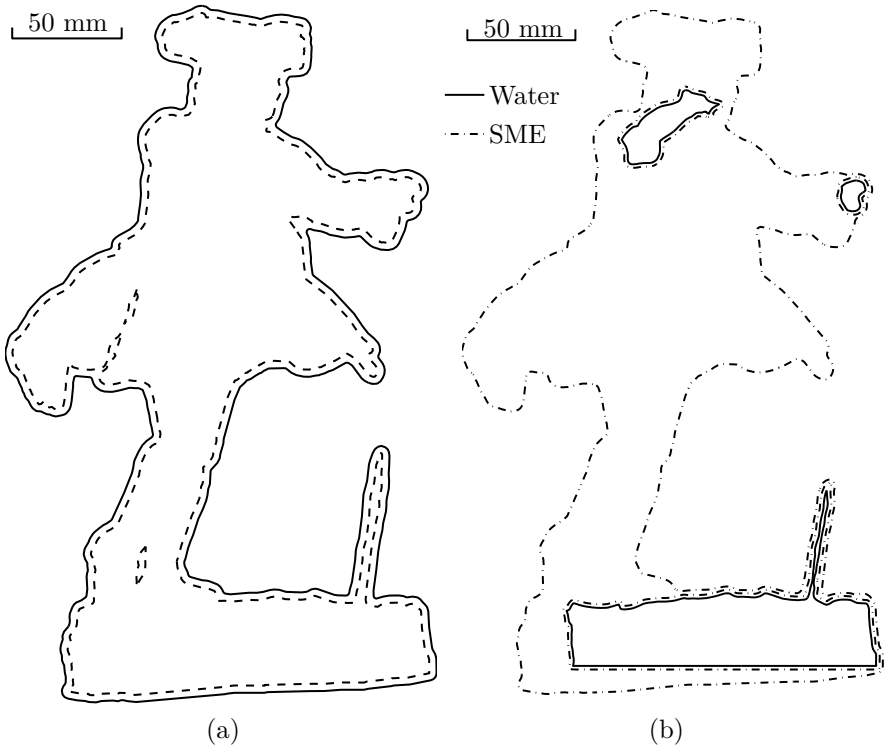


Figure 3–2: Boundary contours for one layer, 25 mm from the substrate of the James McGill statue: (a) Merged contours, before (dashed) and after (solid) buffering; (b) Water and SME contours

buffering is performed with BUFFERF, a modified version of the BUFFERM function available in Matlab's mapping toolbox.

Even when path buffering is used, some error will occur because 1D paths are being used to *approximate* a 2D region. This approximation error is most significant when paths contain sharp changes in direction. It can be reduced by lowering the value of w, though this would also significantly increase construction time.

3.4.1 The Matlab BUFFERM Function

The Matlab BUFFERM function is used to produced buffered or offset regions for polygons or contours composed of latitude-longitude coordinates. BUFFERM introduces several conventions for storing contour data. A polygonal region is composed of several contours, stored in the NaN-delimited vector format or in separate cells of

an array. Vertices for external contours are ordered clockwise, while internal contours, or holes, are ordered counterclockwise. Internal contours are thus buffered in the opposite direction to the commanded buffering direction.

The syntax of BUFFERM is: `[latb,lonb] = bufferm(lat,lon,...`

`dist,direction,npts,outputformat)`

`[latb,lonb]`	output latitude-longitude region
`[lat,lon]`	input latitude-longitude region
`dist`	buffering distance
`direction`	buffering direction (`in` or `out`)
`npts`	number of points used for vertex buffer circles
`outputformat`	data output format (`cell`, `vector`, or `cutvector`)

The offsetting technique used by BUFFERM is summarized in Fig. 3–3. The input contours are buffered one at a time, with the procedure for a single contour as follows: a rectangle of width 2`dist` and a circle of diameter 2`dist` are formed around the first segment and first vertex, respectively, of the contour. A new polygon is formed by computing the Boolean union of these two, using the Matlab function POLYBOOL. Then, this new polygon is added to or subtracted from the subregion formed by the original contour, depending on the buffering direction. The process repeats until all segments and vertices in the contour have been processed.

A single input polygon can be transformed into several new polygons after offsetting. If there is more than one new polygon, BUFFERM forms a new contour group by merging them one at a time using Boolean unions. This eliminates any intersections among the new contours. A buffered contour group composed of one or more contours is formed for each of the input contours. The groups are then merged, one at a time, to form the output region, using Boolean unions for external contours and subtractions for internal contours.

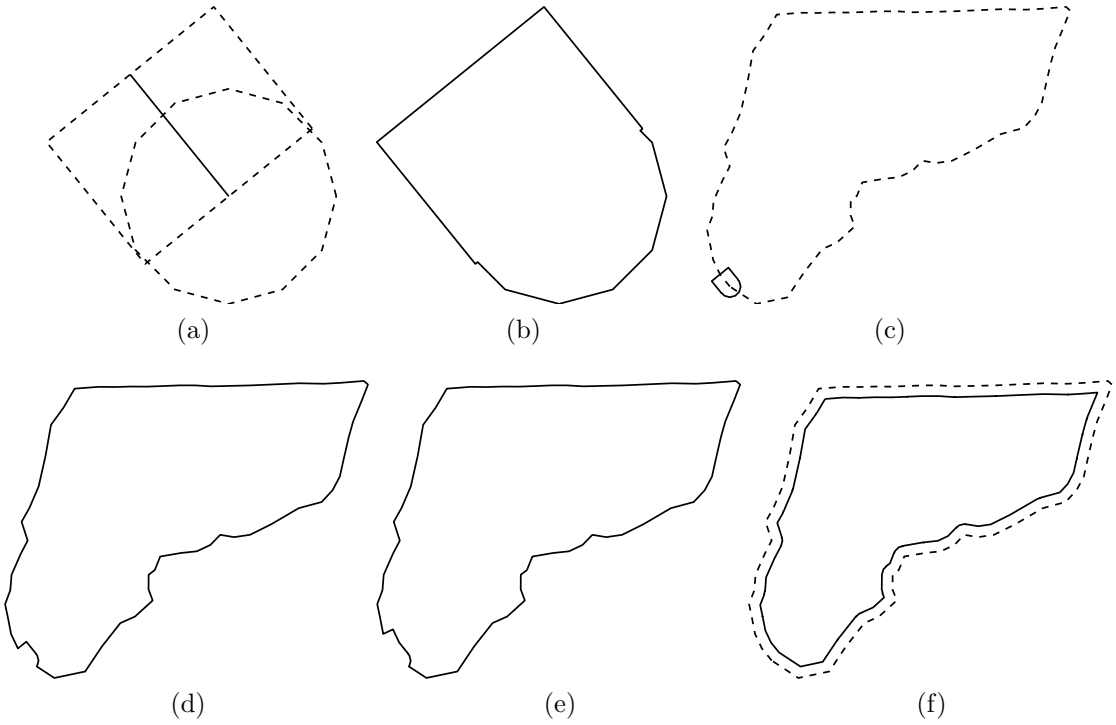


Figure 3–3: The Matlab BUFFERM function: (a–c) A rectangle and a circle are generated and merged for each segment of the contour; (d–f) Boolean operations are performed between these merged polygons and the original contour, one at a time, to produce an offset contour

3.4.2 BUFFERF, a Contour Buffering Function for Planar Regions

BUFFERF was produced by modifying BUFFERM, mainly to improve the computational efficiency. Contour data conventions used in BUFFERM are also used in BUFFERF.

Input data for BUFFERM are assumed to be latitude and longitude coordinates, and geometric calculations are thus done for a spherical rather than a planar surface. Therefore, if contour data pertain to a planar surface, this function will introduce some error when computing buffered contours. A workaround could be to scale down input contour data such that surface curvature has a minimal effect on the geometry. However, computations are simpler for planar geometry and it is preferable to work directly with input data. Therefore, all spherical surface computations in BUFFERM were replaced with their planar counterparts in BUFFERF.

The best feature of BUFFERM is its robustness: as long as input data are in the correct format, it will never produce self-intersecting paths. However, BUFFERM also has a very high computational cost; any modifications that can reduce this cost will significantly impact the overall processing time for RPSLICE. Nearly all of the computational time expended during BUFFERM is during calls to POLYBOOL, the function in Matlab that performs Boolean operations on polygonal regions. As a result, our modifications consist mostly of techniques for minimizing the number of calls to POLYBOOL.

The application of BUFFERF to a single contour is summarized in Fig. 3–4. The circle-and-rectangle technique used by BUFFERM is replaced with a function which computes a single "boundseg" polygon for each line segment, as shown in Fig. 3–4(a). This reduces the number of polygons to merge by half.

The iterative procedure whereby the "boundseg" polygons are merged one-by-one with the original contour has been replaced in BUFFERF by a handful of calls to POLYBOOL, exploiting the fact that a Boolean operation can be performed on polygonal *regions* composed of *multiple* contours. The only restriction is that contours within each region must not intersect with either each other or themselves. A PREBOOL function was thus written, which separates the "boundseg" polygons for one contour into a handful of non-intersecting contour groups. PREBOOL first finds the maxima and minima in the Cartesian dimensions for each "boundseg" polygon, forming a bounding rectangle, as shown in Figs. 3–4(a) . Then, contour groups are formed composed of contours whose bounding rectangles are non-intersecting, as shown in Figs. 3–4(b)–(e).

The contour groups are merged, to form the offset contours, as shown in Figs. 3–4(f)–(h). Since the contour of Fig. 3–4 is composed of 57 segments, 114 Boolean operations would be needed to buffer it using BUFFERM. Using BUFFERF, only three Boolean operations are needed. Additionally, this polygon has a relatively low

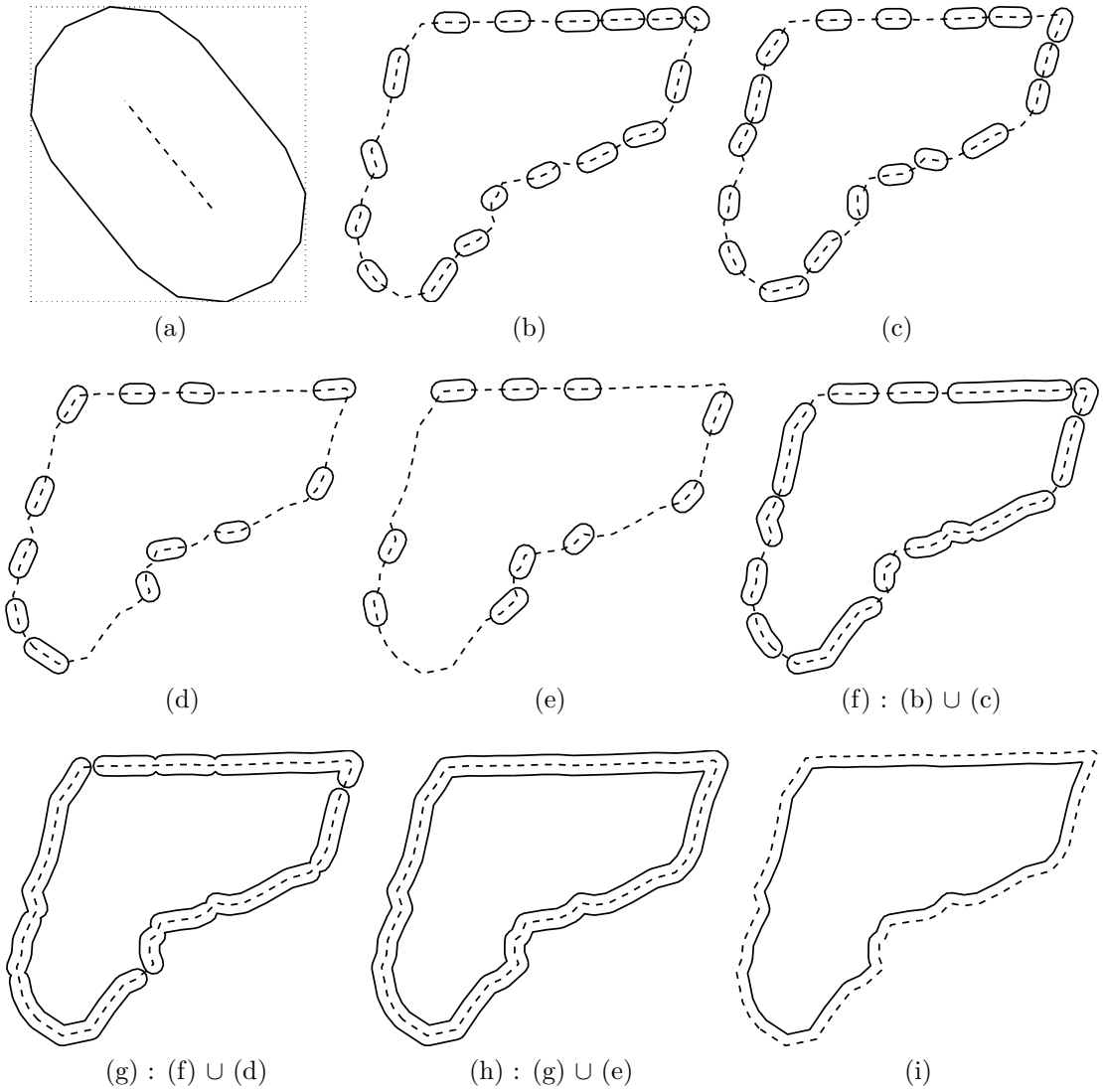
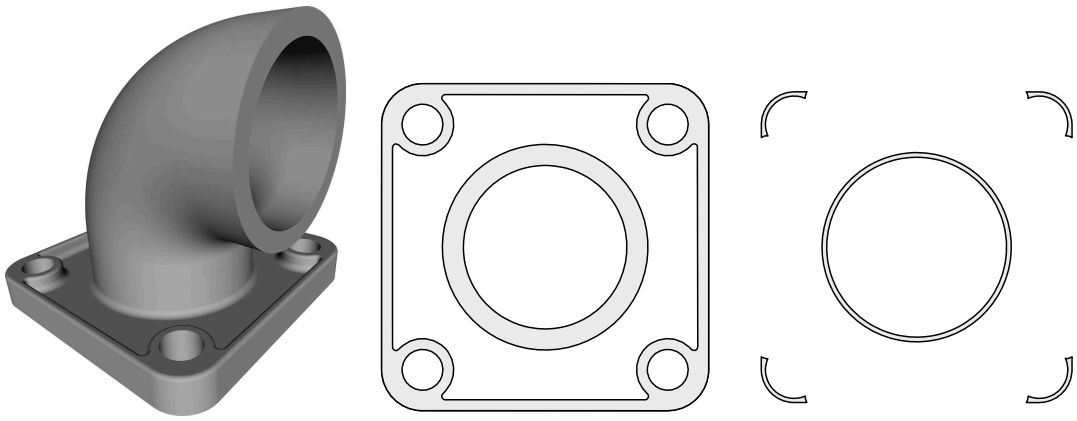


Figure 3–4: The BUFFERF function: (a) One contour segment, along with its "boundseg" polygon and bounding rectangle; (b–e) Groups of non-intersecting "boundseg" polygons; (f–h) Boolean operations to form offset polygons; (g) The desired [*in* or *out*] offset contour is selected

level of detail; the performance difference between the two functions increases with the polygon detail.

PREBOOL would produce even fewer contour groups to merge if "boundseg" polygon intersections were detected instead of computing the bounding rectangles. However, detecting polygon intersections incurs nearly as much computational cost



(a) (b) (c)

Figure 3–5: Buffering a region composed of several contours: (a) a pipe part; (b) One slice near the base of the part, before buffering; (c) after buffering

as performing Boolean operations and actually lowers the overall performance of PREBOOL.

The number of calls to POLYBOOL is reduced further for slices originally composed of multiple contours, as is the case for the part depicted in Figs. 3–5 and 3–6. Since POLYBOOL operations are only needed when polygonal regions are intersecting, non-intersecting polygons can simply be *labelled* and grouped with each other in cell arrays or the NaN-separated vector format. In RPSLICE, input contours are guaranteed to be non-intersecting, as long as the STL input file is properly defined. The nesting among contours, or the number of other contours each contour lies within, is also known. Even levels of nesting indicate external contours and odd levels of nesting indicate internal contours. This information, along with the "bounding rectangle" technique shown in Fig.3–4(a), is exploited to minimize function calls to POLYBOOL.

In BUFFERF, buffered contours are separated into groups at each nesting level, as shown in Fig. 3–6. If the buffering direction is inward, the buffered contours for each external nesting level are simply grouped together via array concatenation, since they cannot intersect. Contours in internal nesting levels could intersect, as is the case for the contours shown in Fig. 3–6(b). In this case, the "bounding rectangle" technique is used to form groups of contours, where no intersections exist within each group.

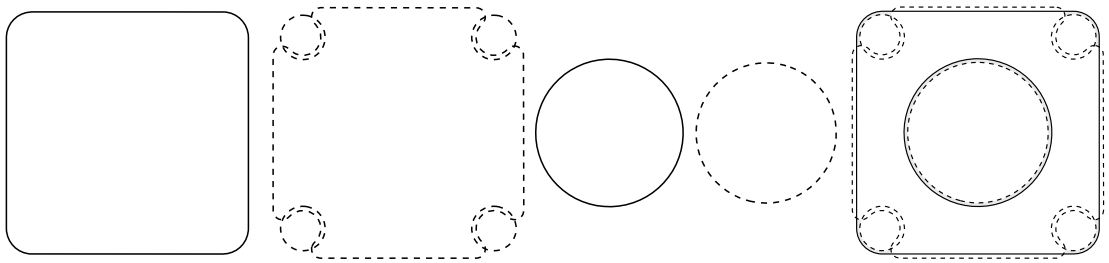


Figure 3–6: Contour nesting for the slice shown in Fig. 3–5(b), after buffering: (a) first nesting level; (b) second nesting level; (c) third nesting level; (d) fourth nesting level; (e) all contours

Using this technique, two groups are found for the contours shown in Fig. 3–6(b): one composed of the four corner circles and one composed of the other contour.

The final buffered region is produced by merging or concatenating the contour groups at each nesting level, one level at a time. For example, the two contour groups in Fig. 3–6(b) are first subtracted from the contour in Fig. 3–6(a), using two calls to POLYBOOL. Then, the contour in Fig. 3–6(c) is added to the group using concatenation, since it cannot intersect contours at the first or second nesting level. Then, the contour in Fig. 3–6(d) is subtracted to form the final buffered region. Therefore, three calls to POLYBOOL are needed to complete this step of BUFFERF; BUFFERM uses seven calls to produce the same region.

The above paragraph describes the technique used for inward buffering; the outward technique is necessarily different. In this case, external contours within each nesting level could intersect and internal contours will not intersect. Additionally, the contours within pairs of nesting levels, e.g., 1 and 2, 3 and 4, etc., cannot intersect. This information is also exploited to minimize calls to POLYBOOL.

Both BUFFERM and BUFFERF add several redundant path points to contour data. To remove unnecessary points, following each buffering iteration, all contours in a layer are filtered using PATHFILTER, as described in Sec. 3.1.

Table 3–4: Times required for boundary contour offsetting, for the James McGill statue in Fig. 3–1

	Facets	Contour Segments	Computational Time (s) BUFFERM[a]	BUFFERM[b]	BUFFERF
Part	3906	112 044	163.8	285.1	23.8
	1 000 000	302 332	667.4	1339.7	64.6
Scaffolding	3906	283 541	479.1	1244.2	60.3
	1 000 000	555 417	1271.4	4196.8	127.4

[a]Revision: 1.5.4.9 Date: 2008/11/24
[b]Revision: 1.5.4.13 Date: 2010/06/26

Table 3–4 shows the computational times needed for boundary contour offsetting, for the James McGill statue in Fig. 3–1. Two versions of BUFFERM are used for comparison: Revision 1.5.4.9, upon which BUFFERF is based, and Revision 1.5.4.13, which is included with Matlab r2011a. The BUFFERF function is always significantly faster than either version; surprisingly, the older version of BUFFERM is much faster than the new version.

3.5 Fill Paths

Many authors [20, 59, 60] reportedly use zig-zag paths to fill object areas in their rapid prototyping algorithms. With the zig-zag technique, typically a series of parallel lines are intersected with the boundary contours, adjacent lines being linked to produce paths that are very long and have many abrupt changes in direction, as shown in Fig. 3–7(a). Xu and Shaw [61] consider 2D material gradients within each slice to produce smooth filling paths.

An iterative inward path buffering technique was initially developed to form fill paths for the Cobra 600 RFP system, mainly to avoid abrupt changes in direction [21]. However, as this technique is computationally intensive, smooth paths are not produced in all cases, and it fails for certain complex geometries. To address these problems, a new filling-path technique, called ZZFILL, was developed, which is similar to the zig-zag filling technique, except that the parallel lines are simply not linked.

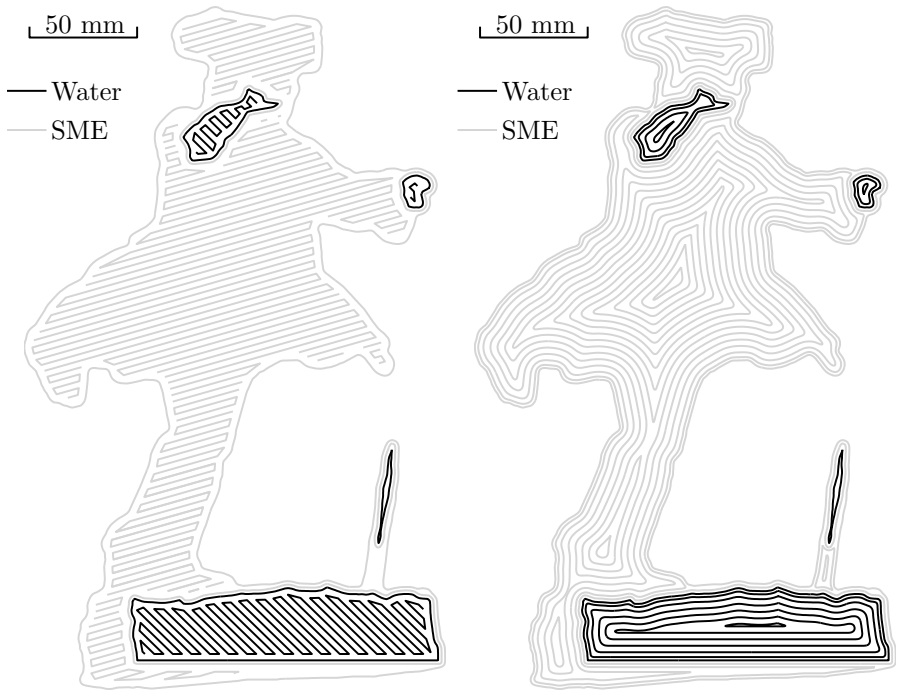


Figure 3–7: Filling techniques for one layer, 25 mm from the substrate, for the James McGill statue: (a) zig-zag technique (b) concentric technique

This technique was well-suited to the then-used trajectory control algorithm [62], which required that deposition only occurs when the tool is moving at a constant speed.

Variable-speed deposition paths, explained in Ch. 4, along with variable material flow, explained in Ch. 6, have since been implemented with the Cobra 600 RFP system, to reduce construction time. With these system improvements, ZZFILL no longer provides the optimal part-filling technique, mainly because it requires frequent, abrupt changes in direction, with deceleration to zero speed. Fill paths are now generated via iterative calls to BUFFERF, until no contours remain, as shown in Fig. 3–7(b). This procedure is computationally *expensive* and would be very time consuming without the optimization work implemented with BUFFERF, described in Subsec. 3.4.2.

Table 3–5: Breakdown of computational times for the slicing algorithm, for the James McGill statue in Fig. 3–1

	Computational Time (s)	
Facets	3906	1 million
FACETREAD	0.1	0.5
FACETSLICE	1.8	12.7
RPSCAF	247.7	243.9
BUFFERF (bounds)	84.1	192.0
BUFFERF (fill)	668.8	832.2
total	1002.5	1281.3

3.6 Results

Table 3–5 shows the breakdown of computational times for slicing the coarse and fine James McGill STL models of Fig. 3–1. Part resolution does not have as significant an impact on computational times as one might expect, because PATHFILTER is used during several steps of RPSLICE to eliminate any unnecessary detail.

Part resolution does have a significant impact on minimum-time trajectory shaping (MTTS), described in Ch. 4, which produces trajectory and valve control data based on the paths output from RPSLICE. The trajectory tracking speed produced by MTTS is greatly dependent on path *smoothness*. Since coarsely-faceted parts can contain artificial, abrupt changes in direction, tracking speed can be much slower than for an equivalent finely-faceted part. Ideally, an STL or PLY faceted part should have sufficient detail such that tessellation error is not higher than the minimum point spacing imposed during FACETSLICE. However, with most CAD software packages, it is difficult to control the detail this precisely. Additionally, as shown in Table 3–5, increasing the part detail does not greatly increase the computational time required. Therefore, it is generally preferable to produce CAD parts with much more detail than is needed. In practice, an input part should have anywhere from a several hundred thousand to a few million facets for optimal behaviour of RPSLICE and MTTS. Of

course, high resolution is unneccesary for certain shapes that can be described exactly with fewer facets.

3.7 Customized Features for RFP

The choice to pursue a customized slicing code was particularly beneficial because several desirable code features were identified and implemented in parallel with the rest of the RFP system development.

Firstly, for RFP, the initial offset from the part boundary must be *material-dependent*, due to the different properties of water and SME. While a path width of 1.5 mm can be reliably used for both materials, the initial offsets were set to 0.6 mm and 1.2 mm, for water and SME, respectively, to yield constructed parts with the correct dimensions.

Secondly, since both water and SME are deposited in liquid form with the RFP system, material spreading can occur if the flow rate is raised high enough. Obviously, this is not desirable for part boundaries, but it can be exploited for fill paths to reduce the deposition time needed. With the Cobra 600 RFP system, the path width for fill paths is currently set to 3.0 mm, which results in considerable part-construction time savings.

Thirdly, the slicing code was modified to completely surround all ice regions with scaffolding, as described in Sec. 3.3, which prevents ice features from rounding off near the part boundaries and permits the use of deposition layer heights of up to 0.25 mm.

Finally, for certain parts, the SME scaffolding material was observed to bend upward off the deposition surface, after several layers had been deposited, causing part failure due to loss of accuracy. To prevent this situation from arising, the slicing code was modified to include the option of constructing an *ice hook wall* at the base of the part to secure the scaffolding to the glass deposition surface. The wall is created by replacing the outermost scaffolding paths for the first 10 layers, and the two outermost paths for layers 10–20, by ice paths. Additional scaffolding contours,

(a) (b)

Figure 3–8: Implementation of the ice hook wall to prevent SME scaffolding from curling up: (a) Without hook wall; (b) With hook wall

offset outward, can be produced if necessary for these layers to prevent the ice hook wall from merging with the ice part. Figure 3–8 shows the impact of using the ice hook wall for constructing the James McGill statue of Figs. 2–10 and 2–11.

3.8 Summary

The major steps involved in a slicing algorithm for Rapid Freeze Prototyping are reported in this chapter. Some functions used in the algorithm were developed from scratch, while others were produced by either modifying Matlab functions or using them directly. The paths produced with the algorithm were used to generate the control trajectories for building the James McGill ice statue of Fig. 2–11(d).

CHAPTER 4
Minimum-Time Trajectory Shaping

RPSLICE, described in Ch. 3, produces deposition paths for rapid prototyping, but does not specify anything about trajectory control along the paths. A technique is thus needed to convert path data into trajectory and valve control data for rapid prototyping. The trajectory control problem involves the minimization of the time required to follow the deposition paths, while respecting certain constraints.

This problem is a special case of the time-optimal trajectory planning (TOTP) problem, discussed in Subsec. 1.2.2. Based on the analysis of available techniques for TOTP in the literature, we formulated the ideal characteristics of a TOTP technique, designed for a general *user* of a positioning system, who wishes to use off-the-shelf products without making extensive hardware modifications, and may not have access to low-level dynamic control:

1. Implementation is possible at the kinematic level, to be accessible to the widest range of potential users.
2. Time-consuming solution techniques are avoided, since some applications involve thousands, or even millions of paths to be followed.
3. No restrictions are placed on path geometry: abrupt changes in both direction and curvature are allowed.
4. A wide range of constraints can be applied: velocity, acceleration, and jerk constraints are allowed at both the joint level and the Cartesian level.

The proposed technique, minimum-time trajectory shaping (MTTS), exhibits all of the required features listed above. MTTS is outlined in [12], the current chapter

providing the details. MTTS is most similar to the technique proposed in [35]; both methods use procedural techniques consisting of multiple subalgorithms. With MTTS, however, the problem is solved using simple, robust, graphical techniques to avoid nonlinear optimization problems.

This chapter initially introduces the MTTS algorithm, which can be divided into several distinct steps. Firstly, linear interpolation is used to produce paths with constant point-to-point distance in Cartesian space. At the same time, smoothing of path data can be performed, if desired. An initial maximum-speed boundary curve is then established for the path, based on the kinematic constraints of the system. Discontinuities in acceleration along this trajectory are corrected in the next step. Then, jerk discontinuities are identified and eliminated; finally, acceleration curves from and to rest are integrated.

A case study is then presented, showing the application of MTTS to the Cobra 600 rapid freeze prototyping system. A test path is used to highlight the benefits of MTTS: joint encoder readings and accelerometer measurements are taken when the Cobra 600 follows the path, with and without MTTS-generated trajectory control data. Then, the performance gains realized when constructing entire ice parts with MTTS-generated control data are discussed.

4.1 Minimum-Time Trajectory Shaping (MTTS)

As mentioned above, MTTS consists of simple, robust steps requiring minimal computation. These features are critical for the rapid prototyping application, where a typical part is produced with over 10 000 paths and several million trajectory points. Some of the steps in the algorithm could possibly be combined in the interest of compactness, though this would also inevitably involve solving more complex problems.

4.1.1 PATHSPLIT: Path Splitting and Point Redistribution

A path to be followed is received by MTTS as a series of n supporting points, with m positioning and orienting variables to be controlled at each point. The path is

first broken into subpaths at abrupt changes in direction, where deceleration to zero speed will be imposed. The threshold angle, beyond which a path split is needed, is a user-configurable parameter, which is typically smaller than 45°. Next, each subpath is transformed to be uniformly sampled in Cartesian space. If no smoothing is desired, this step is achieved via linear interpolation. If some smoothing is permitted, more advanced techniques such as local polynomial regression or cubic smoothing splines can be used. Application-permitting, smoothing should be used, since it will lead to faster robot motion for the final trajectory. The subsequent subfunctions VCON, ACON, JCON, and AREST are applied *separately* to each subpath, though subpaths will be referred to as paths from now on for conciseness.

4.1.2 VCON: Formation of a Maximum-Speed Boundary

For the VCON, ACON, and JCON steps of MTTS, the n supporting path points are not modified; instead, a point-to-point duration vector is formed and adjusted. First, the path position vector $\mathbf{s}$ of dimension n is formed, where each entry is the distance travelled along the path from the start point. A time vector $\mathbf{t}$ is then defined as an array of instants at which the robot operation point P passes through each path point. The point-to-point duration vector, or vector of first-order differences, $\Delta\mathbf{t}$, and the second-order difference vector $\Delta^2\mathbf{t}$, then become

$$\Delta\mathbf{t} = \begin{bmatrix} t_2 - t_1 \\ t_3 - t_2 \\ \vdots \\ t_n - t_{n-1} \end{bmatrix} \quad \Delta^2\mathbf{t} = \begin{bmatrix} t_3 - 2t_2 + t_1 \\ t_4 - 2t_3 + t_2 \\ \vdots \\ t_n - 2t_{n-1} + t_{n-2} \end{bmatrix},$$

with higher-order difference vector arrays defined likewise. Linear interpolation is used to transform $\Delta^2\mathbf{t}$ and $\Delta^3\mathbf{t}$ to $\Delta^2\mathbf{t}^*$ and $\Delta^3\mathbf{t}^*$, vectors of the same dimension as $\Delta\mathbf{t}$. Difference vectors $\Delta\mathbf{s}$, $\Delta^2\mathbf{s}^\dagger$, and $\Delta^3\mathbf{s}^\dagger$ for $\mathbf{s}$ are then defined as

$$\begin{aligned}
\Delta s_i &= \sqrt{(\Delta x_i)^2 + (\Delta y_i)^2 + (\Delta z_i)^2} \\
\Delta^2 s_i^\dagger &= \sqrt{(\Delta^2 x_i^*)^2 + (\Delta^2 y_i^*)^2 + (\Delta^2 z_i^*)^2} \\
\Delta^3 s_i^\dagger &= \sqrt{(\Delta^3 x_i^*)^2 + (\Delta^3 y_i^*)^2 + (\Delta^3 z_i^*)^2} \\
& i = 1 \ldots n-1
\end{aligned} \tag{4.1}$$

where x_i, y_i, and z_i denote the Cartesian position P_i along the path, in the robot base frame $\mathcal{F}_0$. The $\dagger$ superscript is used in $\Delta^2\mathbf{s}^\dagger$ and $\Delta^3\mathbf{s}^\dagger$ to denote the *total* acceleration and jerk, which should be distinguished from $\Delta^2\mathbf{s}$ and $\Delta^2\mathbf{s}$, which represent the *tangential* acceleration and jerk. The path orientation vector $\boldsymbol{\zeta}$ is defined similarly as a function of $\boldsymbol{\phi}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\psi}$, the arrays of yaw, pitch, and roll values of the robot end-effector in $\mathcal{F}_0$ along the path.

The maximum speed boundary for all points along the path is produced by first imposing kinematic constraints on the path shape. Velocity, acceleration, and jerk upper bounds are formed for $\mathbf{s}$ and $\boldsymbol{\zeta}$, and similar bounds are established for each joint j of an m-axis manipulator, as follows

$$\left.\begin{array}{lll}
\dot{s}_{\max} & \dot{\zeta}_{\max} & \dot{p}_{j,\max} \\
\ddot{s}^\dagger_{\max} & \ddot{\zeta}^\dagger_{\max} & \ddot{p}_{j,\max} \\
\dddot{s}^\dagger_{\max} & \dddot{\zeta}^\dagger_{\max} & \dddot{p}_{j,\max}
\end{array}\right\} \quad j = 1 \ldots m \tag{4.2}$$

where $\mathbf{p}_j$ denotes the position vector for joint j. For each joint, we thus have the constraint equations

$$\left|\frac{dp}{dt}\right| \leq \dot{p}_{j,\max}, \quad \left|\frac{d^2p}{dt^2}\right| \leq \ddot{p}_{j,\max}, \quad \left|\frac{d^3p}{dt^3}\right| \leq \dddot{p}_{j,\max} \tag{4.3}$$

from which we obtain, upon discretizing and combining,

$$\left.\begin{aligned} \Delta t_i &\geq \left|\Delta p_i\right| / \dot{p}_{j,\max} \\ \Delta t_i &\geq \left(\left|\Delta^2 p_i^*\right| / \ddot{p}_{j,\max}\right)^{1/2} \\ \Delta t_i &\geq \left(\left|\Delta^3 p_i^*\right| / \dddot{p}_{j,\max}\right)^{1/3} \end{aligned}\right\} \quad i = 1 \ldots n-1. \tag{4.4}$$

The global positioning constraints produce the constraint relations

$$\begin{aligned} \Delta t_i &\geq \sqrt{(\Delta x_i)^2 + (\Delta y_i)^2 + (\Delta z_i)^2} / \dot{s}_{\max} \\ \Delta t_i &\geq \left(\sqrt{(\Delta^2 x_i^*)^2 + (\Delta^2 y_i^*)^2 + (\Delta^2 z_i^*)^2} / \ddot{s}^{\dagger}_{\max}\right)^{1/2} \\ \Delta t_i &\geq \left(\sqrt{(\Delta^3 x_i^*)^2 + (\Delta^3 y_i^*)^2 + (\Delta^3 z_i^*)^2} / \dddot{s}^{\dagger}_{\max}\right)^{1/3} \end{aligned} \tag{4.5}$$

with the global orientation constraints similarly obtained. Using the notation introduced thus far, we formulate the MTTS optimization problem as

$$\min_{\Delta \mathbf{t}} t_n \tag{4.6}$$

subject to the constraints listed in Eqs. (4.4) and (4.5), where t_n is the time elapsed since the start of the trajectory when the manipulator reaches the final path point.

The *geometric* significance of this process is to be highlighted. Velocity constraints simply limit the overall positioning, orientation, and joint speeds. Acceleration constraints, on the other hand, limit the acceleration due to path curvature, which, in the planar motion case, reduces to the radial acceleration. Similarly, jerk constraints do not affect straight-line motion at this stage, but do have an impact in curve locations where there is jerk discontinuity, such as the transition from a circular round to a straight line.

The calculations needed to produce $\Delta \mathbf{t}$ are all explicit and straightforward. As mentioned above, $\Delta \mathbf{t}$ represents a maximum-speed envelope for the path. Subsequent steps of MTTS are constrained to shape the path *within* this envelope.

4.1.3 ACON: Elimination of Acceleration Discontinuities

VCON produces a trajectory that respects velocity constraints but violates acceleration and jerk constraints. ACON is used to identify and correct path regions where there are violations of acceleration constraints and/or "speed-increase" jerk constraints. Speed-decrease jerk constraints cannot be corrected with ACON; they are satisfied using JCON, as described in Subsec. 4.1.4. ACON is applied *separately* for global positioning, global orientation, and for each joint. The most restrictive constraint should be imposed first, to reduce the number of computations. Since the constraints are not imposed simultaneously, new discontinuities can be produced, though these can be eliminated by iterating the process.

Below, ACON is applied for global positioning; the application for global orientation and for each joint is similar. Once all constraint-violating path points have been found, subgroups of *adjacent* violating points are identified; in each subgroup, the point i with the lowest speed is selected as the starting point for an acceleration curve. The initial jerk is set at $\dddot{s}_{\max}$. If adjacent points on both sides of i also violate constraints, the acceleration a_0 at i is set to zero. Otherwise, a_0 is set to the acceleration on the side of i that complies with the constraints. The acceleration curves are then produced *one point at a time*: at each point, the minimum point-to-point duration Δt_{i*} that complies with the constraints is computed; Δt_{i*} replaces Δt_i if $\Delta t_{i*} > \Delta t_i$. Otherwise, i.e., when the original Δt vs. s curve is intersected, the acceleration curve is complete. ACON is thus implemented by adjusting entries of the $\Delta\mathbf{t}$ vector; the path point *locations* are not changed.

At each step, the most complex operation to be performed is the solution of the cubic equation

$$\frac{1}{6}j_0\left(\Delta t_{i*}\right)^3 + \frac{1}{2}a_0\left(\Delta t_{i*}\right)^2 + v_0\left(\Delta t_{i*}\right) - \Delta s_i = 0 \tag{4.7}$$

with j_0, a_0, v_0 being the jerk, acceleration, and velocity in the interval and Δs_i being the distance traveled. Careful definition of these quantities ensures they are all *positive* and thus, of the three roots Δt_{i*}, two are complex and one is real and positive. Root-finding is therefore unnecessary; the latter solution is explicitly specified. The quantities j_0, a_0, and v_0 are updated after each step of acceleration curve generation. The complexity of this process will be dependent on the particular application and the strictness of the constraints. For example, it may be acceptable to use tangential jerk as the estimate for j_0, which simplifies calculations. The application of ACON is straightforward for $\mathbf{s}$ and $\boldsymbol{\zeta}$, whose entries are, by definition, monotonically increasing. The joint angles and joint rates, however, can assume both positive and negative values; therefore, each acceleration curve produced by ACON is restricted to path sections where the sign of the joint rate does not change.

4.1.4 JCON: Elimination of Jerk Discontinuities

As mentioned above, ACON cannot be applied to address violations of "speed-decrease" jerk constraints. Therefore, a different technique is used to eliminate this type of jerk: cubic interpolating polynomials at the velocity level are used to *distribute* jerk where the maximum jerk is exceeded; this is achieved by updating the $\Delta\mathbf{t}$ vector. In each path region where the jerk is to be smoothed, the *sign* of the acceleration is not allowed to change, or equivalently, no inflection point is permitted in the polynomial curve. Without this restriction, uneven jerk distribution and unwanted speed variations could result. This step is applied separately to the global positioning and orientation, and then to each joint individually.

4.1.5 AREST: Integration of Acceleration From and to Rest Curves

Acceleration from and to rest must be considered separately because the evenly-spaced position data used up to this point have insufficient resolution to produce smooth curves at low speeds in the time domain. Again, acceleration curves are produced separately for the global positioning, orientation, and for each individual joint.

One of these sets of constraints—ideally the most restrictive—is used to produce initial acceleration curves, formed by adding extra entries to the $\mathbf{s}$ vector, evenly-spaced in time, at the start and end of the path. This process is similar to that used for ACON, except that it operates via even time-sampling rather than even position-sampling. Once one of the sets of constraints has been imposed, the rest of the constraints are imposed by modifying the entries of the new $\Delta\mathbf{t}$ vector. Following AREST, JCON is reapplied to eliminate any new speed-decrease jerk violations. Then, all subpaths produced by PATHSPLIT are reassembled.

4.2 Case Study: The Cobra 600 Rapid Freeze Prototyping System

The Cobra 600 rapid freeze prototyping (RFP) system, shown in Fig. 1–1, is used to demonstrate MTTS. Given that the Cobra 600 system is an experimental RP platform, however, it might be argued that the benefit of implementing MTTS is not worth the cost in development time and increased system complexity, when compared to other potential construction-time minimization options, such as adaptive slicing [63], or optimization of part orientation during construction [64]. Therefore, we will first argue why these and other options are expected to offer little or no performance gain, while the expected gain associated with implementing MTTS, is significant.

For adaptive slicing, a variable deposition layer height is used to optimize speed and accuracy. This technique can be applied successfully for RP systems that use semi-solid material extrusion, since semi-solids exhibit high deposition aspect ratios. However, for liquid materials, the aspect ratio can be as low as 0.1, so this technique has limited applicability. For example, with the Cobra 600 RFP system, the maximum layer height of 0.25 mm is always used; higher values will result in non-uniform spreading and loss of boundary definition.

Part orientation during construction is typically optimized to minimize the necessary scaffolding volume. However, in some cases, other criteria are more important.

For example, with the Cobra 600 RFP system, part features that are constructed face-up have a better surface finish than features that are face-down. Therefore, manual control of orientation is preferred. Even with manual part-orientation control, close-to-optimal scaffolding use is normally easily accomplished.

An even simpler technique for reducing part construction time might be to exploit the fluid properties of water and the scaffolding material, SME, compared to more traditional, extruded, RP materials. For example, since the viscosity of both materials is low enough to jet through tiny nozzles, we might expect that fill paths are unnecessary and the needed fill volume can can simply be *dumped* and allowed to spread. In fact, this experiment has been carried out with the Cobra 600 system, and it has been observed that water will not spread evenly over the entire top surface of the part before freezing. This occurs because surface tension and viscosity become significant forces with respect to gravity at the scale of the part deposition layer height, which is at most 0.25 mm. A further complication of this technique is that it does not allow for the application of geometric feedback, described in Ch. 5, which can be used to correct geometric errors that have occurred for *any* reason.

A compromise between the material-spreading technique and the detailed fill-path technique can be reached by choosing a fill-path width that is greater than the boundary-path width. Using this method, accuracy is favoured at part boundaries, while speed is favoured for fill paths. Currently, the deposition-path width for the Cobra 600 RFP system is set at 1.5 mm, while the fill-path width is set at 3 mm.

Further optimization of the techniques described above is likely to result in minimal reduction in part-construction time. To establish the expected benefits using MTTS, we need to first establish the characteristics of the system used *before* applying MTTS.

As is the case with many rapid prototyping systems, *constant-speed* paths were used initially with the Cobra 600 RFP system. Boundary paths were followed at

25 mm/s, while straight-line fill paths, similar to those shown in Fig. 3–7(a), were followed at 100 mm/s. The maximum speed for boundary paths was set very low to limit the acceleration in path sections of extremely high curvature. For acceleration a in a horizontal plane, we have

$$a = \sqrt{a_t^2 + a_r^2} \tag{4.8}$$

where a_t is the tangential acceleration and a_r is the radial acceleration. If a maximum allowed acceleration $a_{\max}$ is defined, the maximum speed $c_{\max}$ is given by

$$c_{\max} = \sqrt{\frac{a_{\max}}{r_c}} \tag{4.9}$$

where r_c is the radius of curvature at a certain path location. The constant speed for a path must be chosen such that the maximum allowed acceleration $a_{\max}$ occurs at the minimum possible radius of curvature $r_{c,\min}$. For the Cobra 600 RFP system, $r_{c,\min} \approx 1\ mm$ and $a_{\max} \approx 500$ mm/s^2, leading to the maximum speed of 25 mm/s used for boundary paths. The maximum acceleration $a_{\max}$ is well below the capability of the Cobra 600 because it depends primarily on *inertial* effects, rather than on the maximum joint accelerations. Since the end effector forms a 0.5 m cantilever when the third vertical link is extended, inertial effects are significant enough to produce noticeable deflection and vibration of the end effector, if the acceleration is too high.

If constant-speed paths are used, the straight-line fill path technique is clearly faster than the alternative the concentric-contour technique, with both methods shown in Fig. 3–7. However, with variable-speed paths, the concentric-contour filling is preferred, since many fewer paths are required to approximate the planar slice geometry, and there are fewer abrupt changes in direction.

When MTTS is applied, path speed is dependent *mostly* on path curvature. Since the maximum speed is set to 125 mm/s, significant construction-time savings are

expected for some parts. However, the magnitude of the savings is highly-dependent on the part geometry and the associated deposition paths produced using RPSLICE.

For highly-detailed parts, the construction-time savings using MTTS will be much smaller. However, MTTS is still superior to the previously-used technique because velocity, acceleration, and jerk constraints can be imposed in Cartesian- and in joint-space. For example, jerk constraints are especially important with the Cobra 600 system because the end effector, a 0.5 m cantilever, is susceptible to vibration. Additionally, joint-space constraints are needed because Cartesian constraints are not restrictive enough when the robot nears workspace boundaries.

4.2.1 Implementation of MTTS on the Cobra 600 RFP System

It should be noted that the Cobra 600 RFP system exhibits several features that are needed to implement MTTS. With some RP systems, the implementation described below would need to be modified, or might not be possible at all. Firstly, many RP systems use open-loop positioning control, for which accurate control of velocity, acceleration, and jerk is generally not possible. Secondly, many RP systems cannot vary the material deposition rate quickly or accurately enough to match variations in tool positioning speed.

MTTS, implemented with Matlab, transforms the deposition paths produced by the part-slicing algorithm into suitable control data for the robot controller. The data input format for MTTS for one path is two vectors, $\mathbf{x}$ and $\mathbf{y}$, which are simply the coordinates of all path points. The output format is two joint angle vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, with a point-to-point time resolution of 0.016 seconds. This resolution is chosen to obtain a compromise between path resolution and data transmission constraints.

For the Cobra 600 RFP system, the following kinematic constraints are imposed:

$$
\begin{aligned}
\dot{\theta}_{1,\max} &= 33\,\mathrm{deg/s} & \dot{\theta}_{2,\max} &= 39\,\mathrm{deg/s} \\
\ddot{\theta}_{1,\max} &= 165\,\mathrm{deg/s^2} & \ddot{\theta}_{2,\max} &= 195\,\mathrm{deg/s^2} \\
\dddot{\theta}_{1,\max} &= 1320\,\mathrm{deg/s^3} & \dddot{\theta}_{2,\max} &= 1560\,\mathrm{deg/s^3} \\
\dot{s}_{\max} &= 125\,\mathrm{mm/s} & \ddot{s}^{\dagger}_{\max} &= 500\,\mathrm{mm/s^2} \\
\dddot{s}^{\dagger}_{\max} &= 4000\,\mathrm{mm/s^3}
\end{aligned}
\tag{4.10}
$$

where θ_1 and θ_2 indicate the positions of revolute joints 1 and 2 of the Cobra 600, shown in Fig. 2–2. Joint 3 is the vertical stroke joint for the Cobra 600; since it does not move during deposition paths, no constraints are needed. Joint 4 is a revolute joint that is used to maintain a constant end-effector orientation in the inertial robot base frame:

$$
\theta_4 = \phi - \theta_1 - \theta_2. \tag{4.11}
$$

This simplifies trajectory planning for the two deposition nozzles and the laser displacement sensor, which are all offset from the Joint 4 axis, as shown in Fig. 4–1. No kinematic constraints are required for θ_4, since the inertia it produces is comparatively low. The optimization problem for a single path is thus given by (4.6), subject to the constraints listed in (4.10).

The strictness of adherence to the constraints listed in Eq.(4.10) varies: speed, acceleration, and jerk constraints are decreasingly strict. The constraints are mainly dependent upon the limitations of the flow control system and the cantilevered end effector. This means that if these constraints are respected, the joint rates and inertia loads listed in Table 2–1 are also respected, so the Adept C40 controller should be capable of performing the necessary dynamic control.

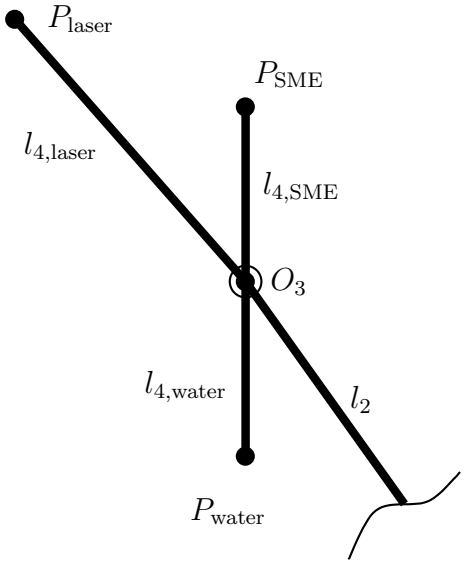


Figure 4–1: Layout of Cobra 600 end effector tools

The global positioning speed constraint listed in Eq.(4.10) is primarily dependent on the available flow rate range of the deposition system used for the Cobra 600 RFP system. This range is limited by the flow control hardware and by the required flow resolution, in terms of drops deposited per distance traveled.

The Cobra 600 RFP system is equipped with a geometric feedback system, which detects part errors with a laser displacement system and corrects them by adjusting subsequent valve control data [11]. Measurements are made using the same paths produced for deposition, except offset to the coordinate frame of the laser emission point. Scaffolding measurement paths are followed using the regular $\dot{s}_{\max} = 125$ mm/s constraint. However, water measurement paths are constrained by $\dot{s}_{\max} = 50$ mm/s to minimize errors produced when measuring the reflective surface of the ice.

The global acceleration constraint is chosen partly to minimize vibrations of the end-effector. Also, since the end-effector is a relatively long cantilever, low acceleration is needed to limit inertial forces and prevent it from deflecting. The value listed in Eq.(4.10) was found to be reasonable based on observations of system performance. The global jerk constraint is based on the assumption that the path should start with

zero velocity, acceleration, and jerk, and should also exhibit zero acceleration and jerk when the maximum speed is reached. With these restrictions, times of 0.5, 0.25, and 0.125 seconds are required to reach the maximum speed, acceleration, and jerk, respectively, and yield the $\dddot{s}_{\max}$ value listed in Eq.(4.10).

The global positioning constraints alone produce reasonable trajectories as long as the robot remains far from singularities in its workspace. When operating near singularities, however, particularly when the robot approaches its maximum reach, the global positioning constraints do not restrict the motion sufficiently. Therefore, the joint constraints listed in Eq.(4.10) were selected such that they affect the robot minimally in most of the workspace but restrict the motion increasingly as the robot approaches its maximum reach.

The path shown in Fig. 4–2 is used to demonstrate MTTS; Figure 4–3 shows the distinction between good and poor placement of the path in the Cobra 600 workspace. This path has several important features that are useful for testing the performance of MTTS and comparing it to alternative techniques. Firstly, there is an abrupt change in direction, which should prompt a deceleration to zero speed. Secondly, there are rounds of several different radii, which should produce varying degrees of speed reduction. Finally, there are abrupt changes in path curvature, which should trigger jerk constraints. The MTTS trajectory-shaping steps are summarized graphically in Fig. 4–4.

4.2.2 PATHSPLIT

The first step of MTTS is to split paths at abrupt changes in direction. For the Cobra 600 system, the path-splitting angle is set at 30°, based on observations of robot performance. Therefore, the path of Fig. 4–2 is split into two pieces at the 90° corner. Linear interpolation is used to obtain the desired point-to-point spacing of 0.4 mm, for each of the new subpaths.

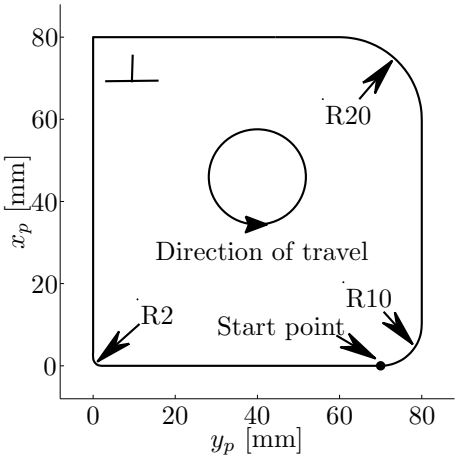


Figure 4–2: Properties of the test path used to demonstrate MTTS. The subscript p denotes a reference frame offset from the Cobra 600 base frame $\mathcal{F}_0$.

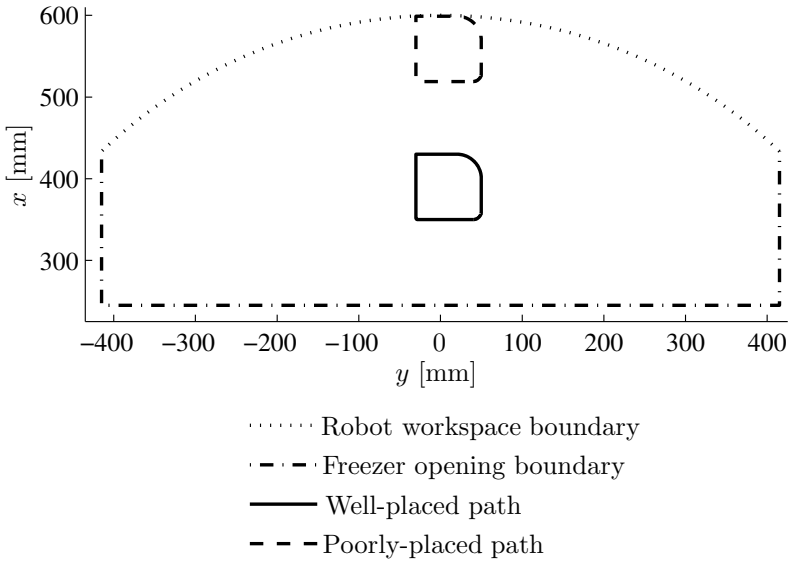


Figure 4–3: Path placement in the Cobra 600 base frame $\mathcal{F}_0$

4.2.3 VCON

Nine inequality constraints are imposed using VCON, based on Eqs. (4.4), (4.5), and (4.10). Six of the relations are produced by substituting θ for p in Eq.(4.4), with j ranging from 1 to 2. The other three equations are produced from Eq.(4.5), with z omitted. Global orientation constraints are not needed. The vector of minimum point-to-point durations $\Delta\mathbf{t}$ is then produced, according to the procedure described in Subsec. 4.1.2.

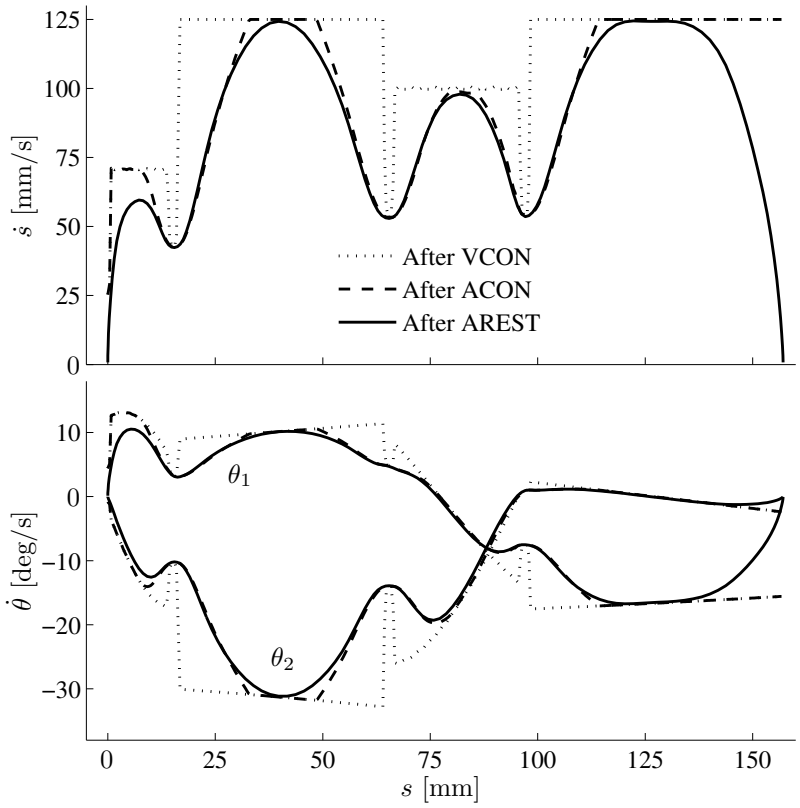


Figure 4–4: Plots of total speed and joint angular velocities vs. path position, showing some steps of the MTTS algorithm, for the path shown in Fig. 4–2 (only the path section from the start point to the 90-degree corner)

4.2.4 ACON

Path points where $\ddot{s}^{\dagger}_{\max}$ or $\dddot{s}^{\dagger}_{\max}$ are exceeded by 10% and 20%, respectively, are identified. As mentioned earlier, strict adherence to these constraints is not necessary, though they should not be greatly exceeded. Since interpolation and filtering can produce small changes in acceleration and jerk values at path points, it is desirable that the constraints *imposed* on the path are slightly stricter than those used to *detect* violations.

The algorithm used for generating a single acceleration curve for a constraint-violating point i is shown in Fig. 4–5. Firstly, the radial acceleration $a_r = c_i^2/r_i$, is computed, with c_i being the speed and r_i the radius of curvature at point i. The tangential acceleration a_t is then computed and used to produce the minimum time

procedure (i, $\mathbf{c}$, $\mathbf{r}$, a_0, $\Delta\mathbf{t}$, $\Delta\mathbf{s}$, $\ddot{s}^{\dagger}_{\max}$, $\dddot{s}^{\dagger}_{\max}$)
  $f \leftarrow 1$
  repeat
    $a_r \leftarrow c_i^2 / r_i$
    $a_t \leftarrow \sqrt{(\ddot{s}^{\dagger}_{\max})^2 - a_r^2}$
    $\Delta t_{ia} \leftarrow \left(-c_i + \sqrt{c_i^2 + 2a_t \Delta s_i}\right) / a_t$
    $\Delta t_{i*} \leftarrow \mathbf{solve}\Big(\dddot{s}^{\dagger}_{\max} (\Delta t_{i*})^3 / 6 + a_0 (\Delta t_{i*})^2 / 2$
      $+ c_i \Delta t_{i*} - \Delta s_i = 0\Big)$
    if $\mathbf{max}(\Delta t_{ia}, \Delta t_{i*}) < \Delta t_{i+1}$ **then**
      $f \leftarrow 0$
    else
      $i \leftarrow i + 1$
      if $\Delta t_{ia} > \Delta t_{i*}$ **then**
        $\Delta t_i \leftarrow \Delta t_{ia}$
        $a_1 \leftarrow a_t$
        $j_1 \leftarrow 0$
      else
        $\Delta t_i \leftarrow \Delta t_{i*}$
        $a_1 \leftarrow a_0$
        $j_1 \leftarrow \dddot{s}^{\dagger}_{\max}$
      end if
      $c_i \leftarrow v_{i-1} + a_1 \Delta t_i + j_1 (\Delta t_i)^2 / 2$
      $a_0 \leftarrow a_1 + j_1 \Delta t_i$
    end if
  until $f = 0$
end procedure

Figure 4–5: ACON: Acceleration constraint algorithm

difference Δt_{ia} to reach $i + 1$, while respecting the $\ddot{s}^{\dagger}_{\max}$ constraint. Similarly, Δt_{i*} is the minimum time that respects the $\dddot{s}^{\dagger}_{\max}$ constraint, explicitly defined as the only real, positive root of the cubic equation displayed in Fig. 4–5, as justified in Subsec. 4.1.3. If either Δt_{ia} or Δt_{i*} is greater than Δt_{i+1}, the larger of the two replaces this entry, all other necessary parameters then being updated; otherwise, the algorithm ends. As presented, the algorithm can be applied to produce *forward* accelerations; *backward* accelerations are produced by changing the increment from +1 to −1. Non-tangential jerk is ignored by this algorithm; this is acceptable because strict

procedure (i, $\mathbf{a}$, $\Delta\mathbf{t}$, $\dddot{s}^{\dagger}_{\max}$)
  $f_l \leftarrow 1$
  $f_r \leftarrow 1$
  $l \leftarrow i$
  $r \leftarrow i$
  repeat
    if $f_l = 1$ **then**
      if $a_l > 0$ **or** $l = 1$ **then**
        $f_l \leftarrow 0$
      else
        $l \leftarrow l - 1$
      end if
    end if
    if $f_r = 1$ **then**
      if $a_r > 0$ **or** $r = \mathbf{length}(\mathbf{a})$ **then**
        $f_r \leftarrow 0$
      else
        $r \leftarrow r + 1$
      end if
    end if
    $\Delta t_* = \sum_{k=l+1}^{r} \Delta t_k$
    $j_t = |a_r - a_l| / \Delta t_*$
  until $j_t < \dddot{s}^{\dagger}_{\max}$ **or** $f_l + f_r = 0$
end procedure

Figure 4–6: JCON: Jerk constraint algorithm

adherence to jerk constraints is not needed. The algorithm is modified for application of the joint acceleration constraints, as mentioned in Subsec. 4.1.3. Also, there is no radial acceleration for the joints, which leads to some additional simplification. The dashed lines of Fig. 4–4 show the shape of the speed curves following the application of ACON.

4.2.5 JCON

As mentioned in Subsec. 4.1.3, speed-decrease jerk constraint violations cannot be eliminated using ACON; in fact, most of these violations are actually *produced* by ACON. JCON smooths out speed-decrease jerk spikes by using cubic polynomial interpolation, ensuring continuity in the velocity and acceleration domains. Sufficient jerk dissipation is ensured by appropriately selecting the interval of the polynomial

interpolant in the path position domain. The global positioning constraint is imposed first, followed by the two joint constraints.

The algorithm for finding one jerk-dissipation interval is summarized in Fig. 4–6. The locations of speed-decrease jerk spikes are first identified. Subgroups of adjacent points are then formed among the identified points. Each subgroup is gradually expanded by incrementing the leftmost and rightmost points left and right, respectively, until the change in acceleration over the entire subgroup is low enough to respect the jerk constraint. Leftward expansion of the interval is halted if a change in acceleration sign is encountered; rightward expansion is halted on the same condition, but separately. Once the interpolation interval Δt_k has been found, the speed over this interval is given by

$$\begin{aligned} v &= D\tau^3 + C\tau^2 + B\tau + A, \qquad \tau = 0 \ldots 1 \\ A &= v_0, \qquad C = 3v_1 - a_1\Delta t_k - 3A - 2B, \\ B &= a_0\Delta t_k, \qquad D = v_1 - A - B - C \end{aligned} \tag{4.12}$$

where v_0, v_1, a_0, and a_1 are the velocity and acceleration at the start and end of the interval, and τ is the normalized time. As described in Subsec. 4.1.4, this procedure is approximate, since the change in acceleration is computed based on the point-to-point durations and these durations will change after application of the polynomial interpolant. However, since the durations will only increase using this procedure, the jerk estimate is slightly higher than the final jerk exhibited.

4.2.6 AREST

Acceleration from and to rest is dealt with separately to obtain sufficient resolution in the time domain. During this process, careful consideration of the robot *positioning resolution* is also necessary. Ideally, a subpath would start with position, velocity, acceleration, and jerk all zero. However, the Cobra 600 has finite positioning

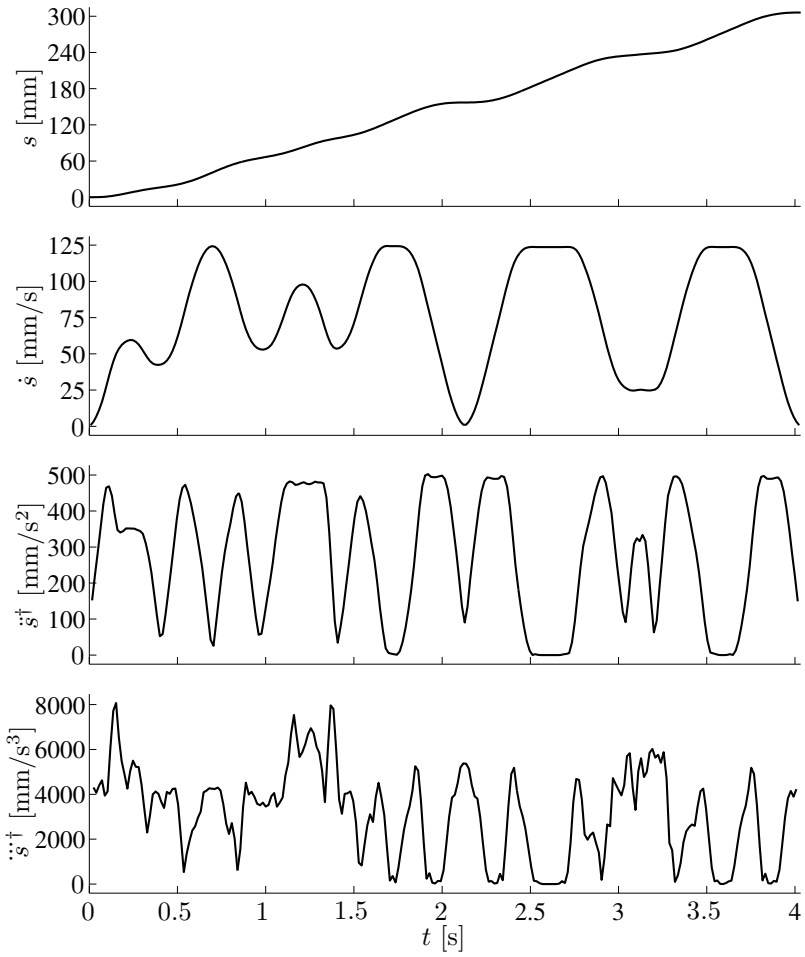


Figure 4–7: Plots of position, speed, acceleration, and jerk vs. time for the well-placed path in Fig. 4–3

resolutions of 0.00045° and 0.00072° per encoder count for joints 1 and 2, respectively [32]; the commanded displacements at the start and end of the trajectory will be below these resolutions if the ideal path end conditions are imposed. To produce displacements above the joint positioning resolutions, an initial and final jerk of $\dddot{s}^\dagger_{\max}$, with an initial and final acceleration of 80 mm/s^2, are imposed. Subsequent points on the acceleration from rest curve are produced as in the algorithm of Fig. 4–5, except that they are spaced by 0.008 s, which is half the desired time resolution of the final, output trajectory. This procedure thus produces new $\mathbf{s}$ and $\mathbf{t}$ vectors. The joint acceleration from rest constraints are imposed by modifying entries of the new $\Delta\mathbf{t}$ vector.

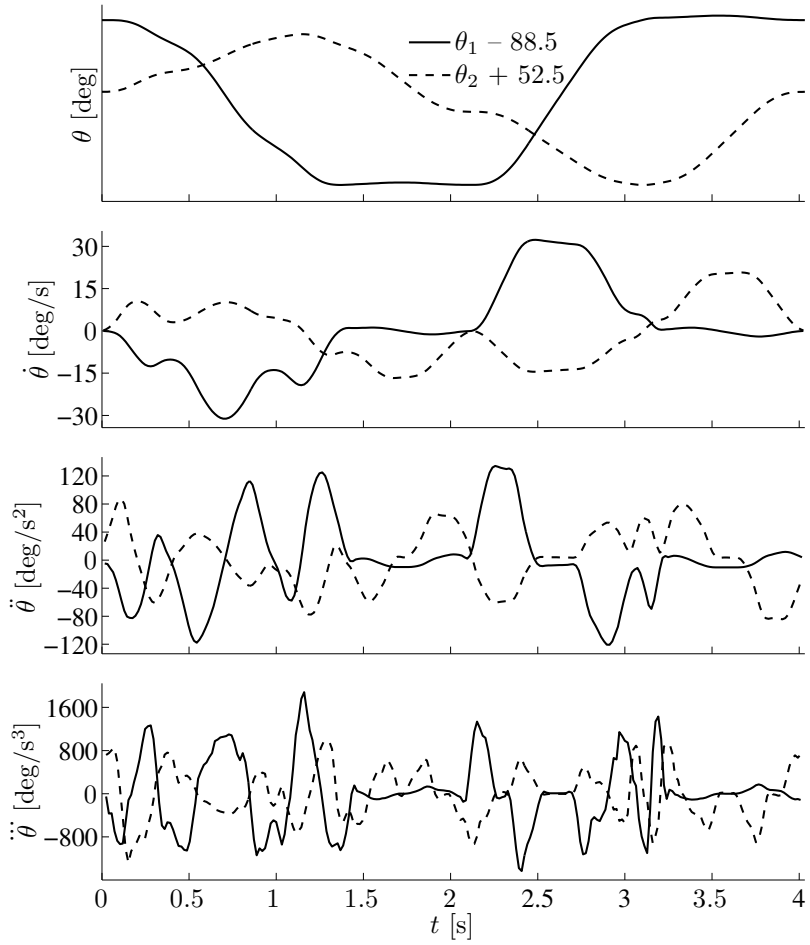


Figure 4–8: Plots of joint angle position, speed, acceleration, and jerk vs. time for the well-placed path in Fig. 4–3

Following AREST, linear interpolation is used to produce subpaths equally sampled at the desired time resolution of 0.016 seconds; then, JCON is reapplied. The solid line in Fig. 4–4 shows the shape of the speed curve at this stage. The subpaths produced during PATHSPLIT are then reassembled into the original paths. Finally, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ joint angle vectors for the full path are smoothed separately using a 7-point, third-order Savitzky-Golay filter. This filter is chosen because it eliminates a significant amount of noise that is apparent in the jerk plots but does not significantly change the path shape or modify the total path length.

4.3 Simulation Results

Figure 4–7 shows plots of position, speed, acceleration, and jerk vs. time for the well-placed path of Fig. 4–3. It is interesting to note the shape of the speed plot along the rounds. If no jerk constraint had been imposed, constant speed would be exhibited along these rounds. However, acceleration discontinuities, which are detected by the jerk constraint, occur at the points of transition between the rounds and the straight sides of the path. MTTS imposes local minima in speed at these curvature discontinuities, and speed increases to local maxima at the centers of the rounds. Figure 4–7 also shows the jerk exceeding $\dddot{s}^{\dagger}_{\max}$ at several locations. However, this is acceptable, since the maximum jerk is the least strict of the constraints. Figure 4–8 shows similar plots for joints 1 and 2. The joint angle positions θ_1 and θ_2 are offset to fit on the same plot.

Several tests are performed to demonstrate the performance of MTTS. Figure 4–9 shows a comparison of the trajectories produced for the well-placed and poorly-placed paths of Fig. 4–4. As expected, the joint constraints significantly affect the motion for the poorly-placed path, which lies within 1 mm of the robot maximum reach. It can be observed that motion along the x axis is significantly restricted, while motion along the y axis is mostly unchanged. This occurs because the x-axis motion requires relatively large joint angle changes when compared to the y-axis motion. The joint rate vs. position plots show that the $\dot{\theta}_{2,\max}$ constraint has been imposed in both regions where the motion has been significantly affected.

While the simple test path from Fig. 4–4 effectively demonstrates the steps of MTTS, a broader characterization of the algorithm performance is needed. To achieve this, MTTS is applied to deposition paths produced by slicing a CAD model of Lucy, a Christian angel, downloaded from the Stanford 3D Scanning Repository[1], shown in

[1]http://graphics.Stanford.edu/data/3Dscanrep/

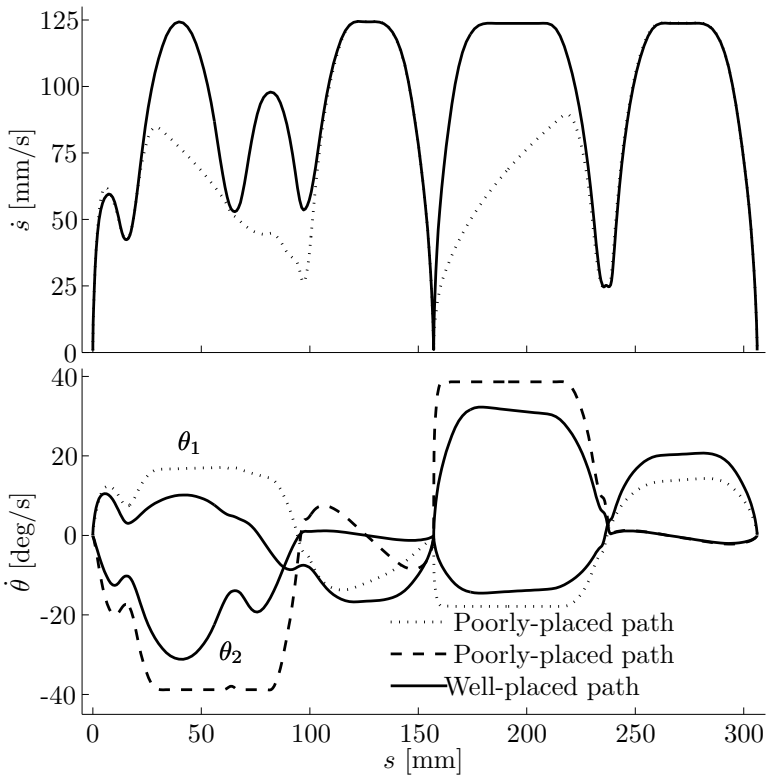


Figure 4–9: The speed vs. position profiles for the well-placed path and the poorly-placed path shown in Fig. 4–3

Fig. 4–10(a). A slicing plane is shown passing through the model; the construction orientation is face-up. The rapid prototyping paths for this slice are shown in Fig. 4–11; these paths will produce a 0.25 m-high version of the statue, when standing upright. The application of MTTS for this slice is also shown in Fig. 4–11: the speed at all path locations is *embedded* in the point-to-point spacing, since the point-to-point duration is held constant at 0.016 seconds. The spacing is very dense in sharp corners and becomes coarser in regions of low curvature. Increased spacing leads to increased speed and less path detail, though fewer points are needed to describe path regions of low curvature. This leads to a more compact data set and thus reduces data transfer time during part construction, when compared to paths with constant point spacing. For the Cobra 600 RFP system, this is significant, since the robot controller has a relatively low data transfer speed, when compared to modern computers.

(a) (b)

Figure 4–10: (a) STL file courtesy of the Stanford 3D Scanning Repository, produced by scanning a statue of Lucy, a Christian angel; (b) the ice statue, 0.5 m high, built with the Cobra 600 rapid freeze prototyping system

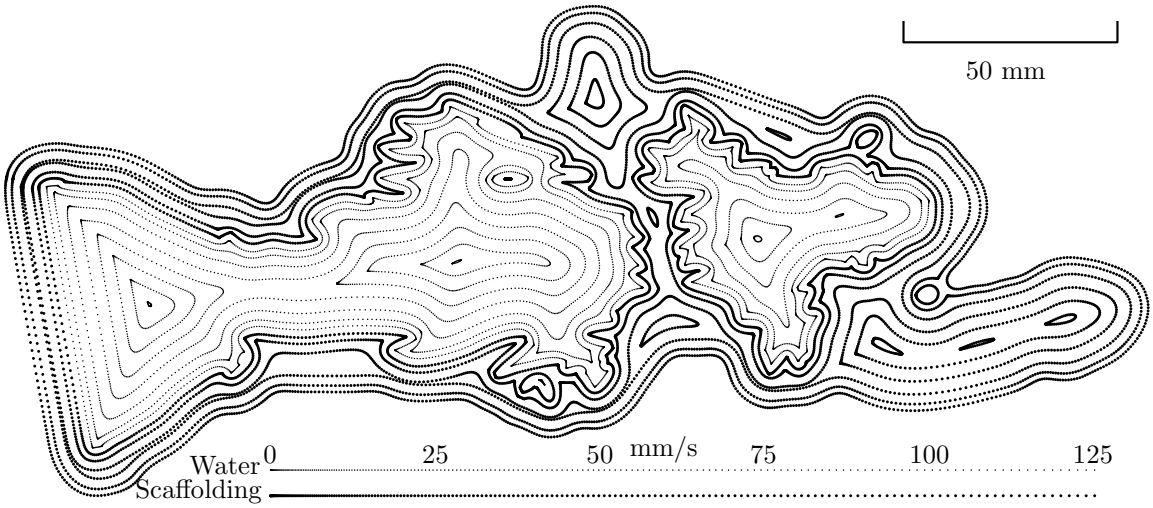


Figure 4–11: Deposition path speeds produced using minimum-time trajectory shaping for the layer shown in Fig. 4–10(a), to produce a 0.25 m high part. Speeds are encoded in the point-to-point distance, holding the point-to-point duration constant.

To benchmark the computational performance of MTTS, control data are produced for a scale model of the Lucy statue, 0.5 m high when standing right-side up.

At this scale, about 85 hours are required to build the statue in the face-up orientation, with the Cobra 600 RFP system. The model is sliced into 572 layers, consisting of about 48 000 total paths, covering a total distance of about 10 000 m, with approximately 14 million path points. About 20 minutes are required to run MTTS for this part on an Intel Core i7-2720QM processor, completing computations in four parallel threads whenever possible. These statistics include the synthesis of water, scaffolding, and measurement trajectories.

4.4 Experimental Results

While the trajectory produced using MTTS is near-optimal, this does not guarantee that the robot will actually follow the data in a smooth manner. The Cobra 600 controller is highly sensitive to the manner in which trajectory data are provided: two control techniques which might seem equivalent could produce very different results. For example, one representation of a path could be a series of points equally separated in time, with variable position spacing, while a second representation could be variable point-to-point duration with constant position spacing. While both representations are basically equivalent in theory, the first will produce the desired smooth motion, while the second will produce discontinuous, jerk-filled motion.

To verify that the MTTS control data produce the desired results, the Cobra 600 was commanded to follow the well-placed path of Fig. 4–4. Figure 4–12 shows the kinematic parameters exhibited by the robot when following this path, based on joint-encoder readings. Joint-encoder data are filtered using a 128-point, cubic polynomial least-square filter (LSF). This technique is similar to that shown in [65], except that, when computing the polynomial fitting curves, encoder data are used both before and after the point of interest, since the data are being used for analysis and not for real-time control. Figure 4–13 shows the *actual* path followed by the robot, based on the joint encoder readings.

Figure 4–14 shows acceleration readings taken with an accelerometer fastened at the end-effector tip. The accelerometer is the LIS3L02AQ three-axis linear accelerometer, which has been combined with a wireless transmitter and data acquisition system by Pultronics, Inc[2]. The sampling frequency of this accelerometer is 200 Hz per axis; the data shown in Fig. 4–14 are produced by applying a 4-Hz low-pass filter.

To highlight the benefits of MTTS, a different control technique, called the "default" technique in Figs. 4–12 and 4–14, is also used to command the robot along the test path. With this technique, the path is supplied to the robot as a series of supporting points, but the point-to-point duration is not specified. Only basic commands available in the Adept V+ programming language are used to control kinematic parameters: the maximum speed is set at 125 mm/s and the acceleration is adjusted until the time of travel is approximately the same as that used for the MTTS path. The acceleration is keyed in as a percentage of the maximum robot acceleration; absolute values are not permitted.

As shown in Fig. 4–12, the test path is followed very closely by the robot when the trajectory control data supplied by MTTS is used. Several problems are apparent for the default technique. Firstly, as shown in Fig. 4–13, the path corner where there is an abrupt 90° change in direction is rounded off to a radius of approximately 10 mm. This is unacceptable for rapid prototyping, since it is desirable to reproduce the modeled part as accurately as possible. This problem can be eliminated by detecting abrupt changes in direction and inserting a `BREAK` command in the Adept V+ programming language used to control the robot, which would force deceleration to zero and exact tracking of the sharp corner. Of course, the robot would then take longer to follow the path. A second significant problem with the default technique is the strong dependence of robot speed on point spacing, despite the commanded speed

[2]http://www.pultronics.com

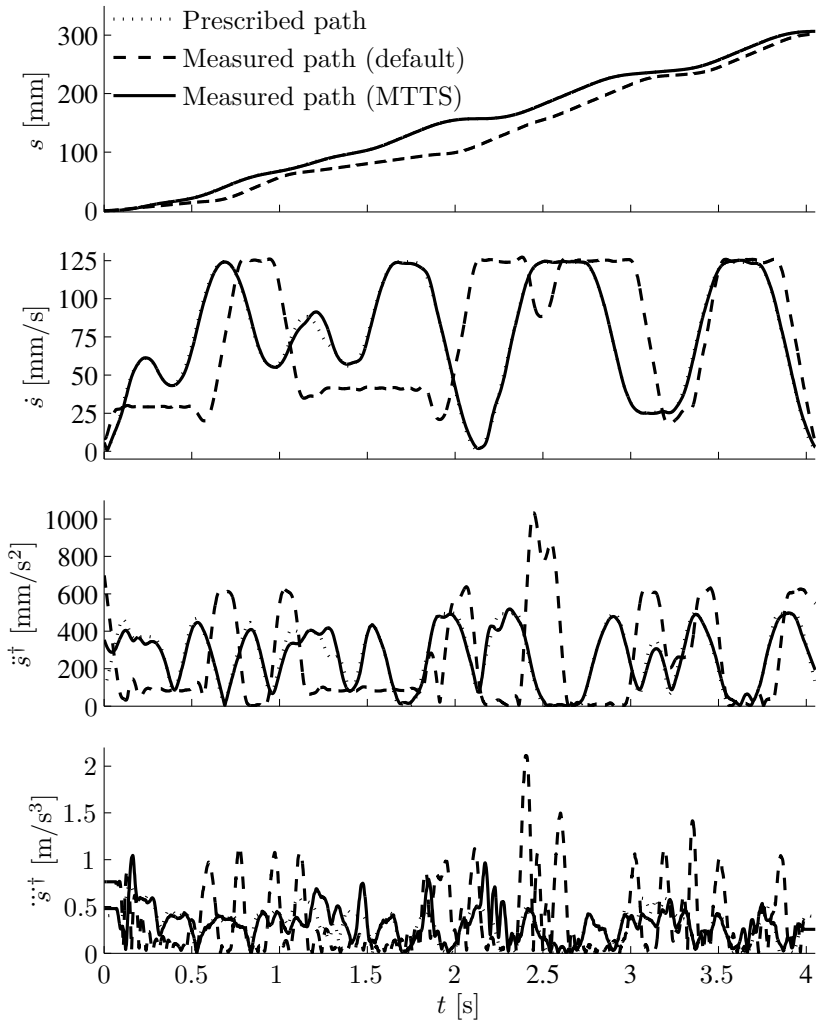


Figure 4–12: Kinematic parameters, based on joint encoder measurements, for the well-placed path shown in Fig. 4–3

of 125 mm/s: the robot follows the 10 mm and 20 mm rounds very slowly, when compared to MTTS. The only way to raise the speed using the default technique is to coarsen the point spacing. The kinematic performance, defined as the degree to which the kinematic constraints are respected, is reasonably good for the default path, though noticeably worse than that for MTTS. The accelerations reached are all higher for the default technique, with a significant spike in acceleration near the center of the motion. The jerk is also much smaller for MTTS; jerk spikes could be

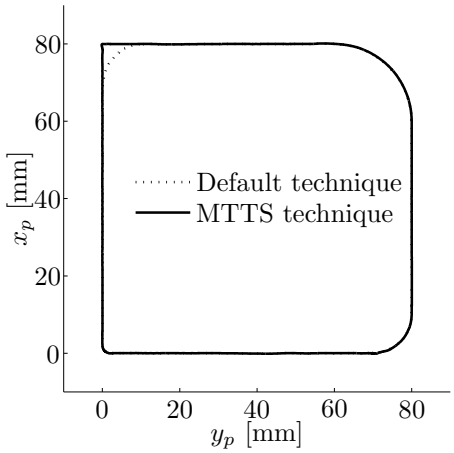


Figure 4–13: The actual path followed by the Cobra 600, based on joint encoder readings, for the well-placed path shown in Fig. 4–3.

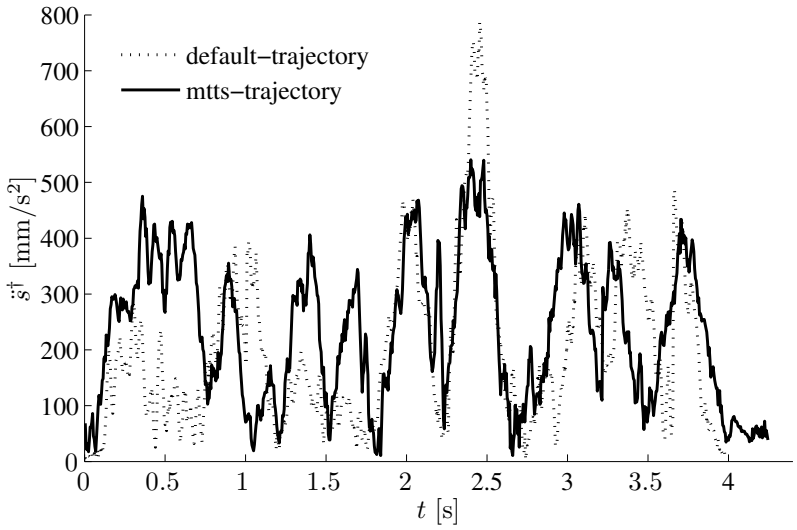
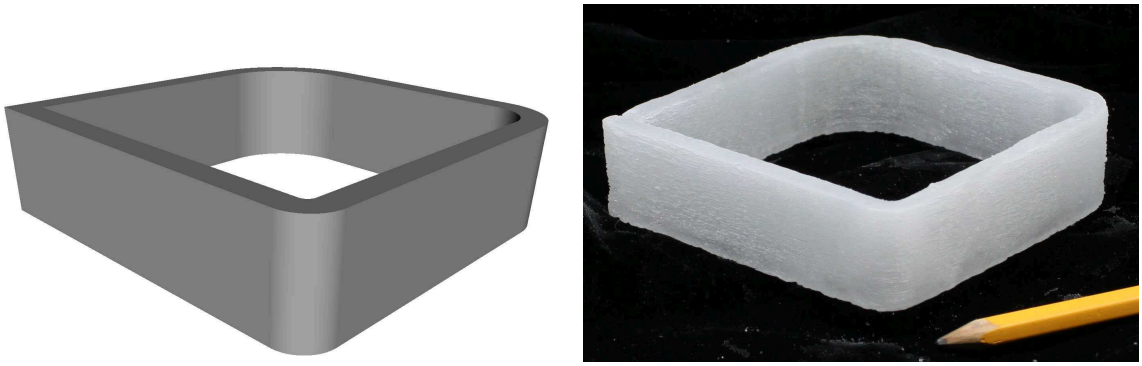


Figure 4–14: Accelerations measured with a three-axis accelerometer for the well-placed path shown in Fig. 4–3.

reduced for the default method through the use of trapezoidal acceleration profiles, though this would also lengthen the time of travel for the path.

As mentioned above, the "default" technique cannot be used for rapid prototyping, because of the corner-rounding effect would produce unacceptable geometric errors. However, it is desirable to quantify the performance of the Cobra 600 RFP system *before* and *after* MTTS is applied. The previously-used technique, consisting of boundary paths followed at 25 mm/s and straight-line fill paths followed at 100 mm/s,



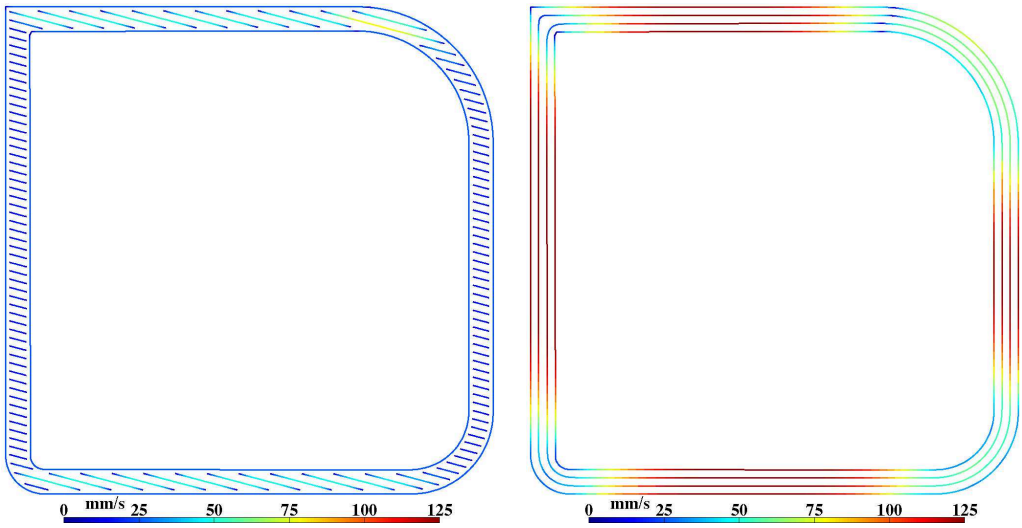
(a) (b)

Figure 4–15: Part with interior boundary the same as that shown in Fig. 4–2: (a) STL file; (b) Constructed ice part

is described in Sec. 4.2. However, several other improvements to the Cobra 600 RFP system have been made since that system was used, so direct comparison, highlighting the improvements of MTTS, is difficult. Additionally, many of the kinematic constraints applied by MTTS were not applied with the previous system, which led to unwanted inertial effects and vibrations.

Therefore, the benefits of MTTS will be highlighted by comparing two cases: "constant speed" and "variable speed". The variable-speed case will involve standard application of MTTS, as described above, with a maximum speed of 125 mm/s. For the constant-speed case, the maximum speed will be set to 25 mm/s for boundary paths, the value that will not produce radial acceleration above the maximum allowed value. Fill paths are produced with the straight-line technique, with a maximum speed of 125 mm/s. As a test part, a box with rounded corners, shown in Fig. 4–15, is used. The deposition paths for one layer are shown in Fig. 4–16. The total time required to build this part is 11.6 hours using the constant-speed technique and 2.6 hours using the variable-speed technique.

It can be seen that this part is reasonably well-suited to MTTS, since there are many long, straight paths that can be followed at high speed. However, an ideal part would be a thin-walled cylinder, with no regions of high curvature, as this would



(a) (b)

Figure 4–16: Comparison of trajectory control techniques: (a) "constant-speed" technique; (b) "variable-speed" technique, using MTTS

avoid any need for reducing speed, except at the endpoints. Additionally, many fewer paths are needed with the concentric-contour fill technique. A third case, is therefore considered, where the concentric-contour fill technique is used with constant-speed paths at 25 mm/s. For this case, the required construction time is 7.4 hours.

Clearly, MTTS provides significant construction-time savings for the box with rounded corners. For a part with more complex geometry, such as the Lucy statue, shown in Fig. 4–10, the construction-time savings are reduced, but still significant. MTTS was used to produce the 0.5 m-tall version of Lucy shown in Fig. 4–10, the construction time being 3.5 days. Without MTTS, using the same parameters as above for the constant-speed case with straight-line fill paths, construction time is predicted to be over 5 days.

4.5 Summary

This chapter introduced Minimum-Time Trajectory Shaping (MTTS), a trajectory control technique for synthesizing time-optimal trajectories for paths of fixed geometry, with several kinematic constraints. This optimization problem is complex:

rather than attempting to solve it with a single optimization formulation, MTTS implements a series of simple, robust, efficient steps. This technique also makes MTTS adaptable to different systems and the addition or modification of constraints. The application of MTTS to the Cobra 600 rapid freeze prototyping system has been shown to yield far superior results to traditional trajectory control techniques available within the Adept V+ programming language. Most importantly, MTTS does not round off paths at abrupt changes in direction, and does not limit the speed according to the input path point spacing. MTTS has been integrated with the control software for the Cobra 600 RFP System and several test parts have been successfully constructed.

CHAPTER 5
Surface Mapping Feedback

Part accuracy is one of the most important measures of performance for rapid prototyping (RP) systems. Unfortunately, for many *deposition-based* RP systems, part error accumulates as more and more layers of material are added. Control of part geometry is therefore *open-loop*, since no verification is made by the system to confirm that the printed geometry conforms to the nominal geometry. To address this problem, a closed-loop geometry control system is needed. In Subsec. 1.2.3, a review of previous research into closed-loop control systems for rapid prototyping is presented. However, only one of these systems could be called true geometric feedback [41].

Here, we propose a new control technique, surface mapping feedback (SMF), which is a true geometric feedback system that detects and corrects part geometry errors. SMF can be applied to both drop-on-demand and continuous-flow deposition; geometry adjustment can be performed by adjusting material flow and/or deposition tool speed. Drop-on-demand refers to a deposition system whereby the target location of each drop of material deposited is controlled, typically while the tool is stationary. Continuous-flow refers to a system whereby a continuous stream of material is deposited, typically while the tool is moving.

In this chapter, the general application of SMF is first introduced. Then, the implementation of SMF for the Cobra 600 RFP system is described, and the performance is characterized, compared to open-loop geometry control. Finally, a one-dimensional version of the SMF controller is used to perform stability analysis and controller gain optimization.

An important consideration is whether closed-loop geometry control is necessary in *both* the horizontal and the vertical directions. For the Cobra 600 RFP system, under open-loop geometry control, part error in the vertical direction was typically observed to be at least one order of magnitude higher than that in the horizontal direction. For this reason, a geometric feedback system is developed that regulates part geometry *only* in the vertical direction. The validity of this choice is justified by measuring several dimensions of a constructed part with a vernier caliper and comparing them to the those of the virtual CAD part.

5.1 The SMF Technique

Surface mapping feedback (SMF) involves three main steps: part measurement, control loop iteration, and deposition adjustment. From a control perspective, it is important to distinguish between online and offline control. For RP systems, we can define an online feedback system as one where the measurement and control operate in parallel with the material deposition. For the proposed surface mapping feedback control, online control is possible with the appropriate system hardware, including adequate computational resources and measurement and deposition systems that do not interfere with each other. However, an offline implementation, involving part measurement in between periods of material deposition, is much more tolerant of system hardware limitations.

5.1.1 Surface Measurement

Measurement accuracy is obviously critical for SMF; it should be significantly higher than the desired part accuracy. Additionally, measurement should be *fast*, meaning that the part construction time with SMF on should not be significantly longer than with SMF off.

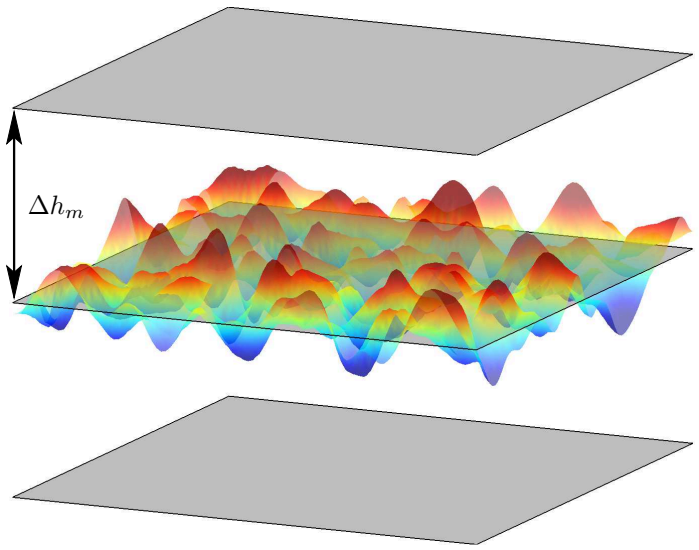


Figure 5–1: Surface mapping feedback (SMF): The rippled surface represents the current part height, while the gray plane intersecting it is the nominal height. All gray planes indicate nominal heights where the part surface would be mapped.

The two main classes of distance-measurement devices are contact and non-contact. Since detailed mapping of the part surface would necessarily be time-consuming with contact-type devices, they can be immediately ruled out. The non-contact class includes computer vision, photographic, laser, and ultrasonic devices. It is noteworthy that the measurement ability of all non-contact displacement sensors is highly dependent on part-surface characteristics. For example, many optical sensors are unable to correctly measure distance to transparent or translucent objects.

A key feature of SMF is that the feedback loop operates at the *surface* level rather than at the *point* level. During measurement, the *entire* top surface of the part and scaffolding is mapped, generating a surface interpolant, which is then used to adjust subsequent material deposition. The interval Δh_m, defined as the vertical distance between measurement layers, determines the frequency of surface measurements. Figure 5–1 provides a graphical representation of SMF.

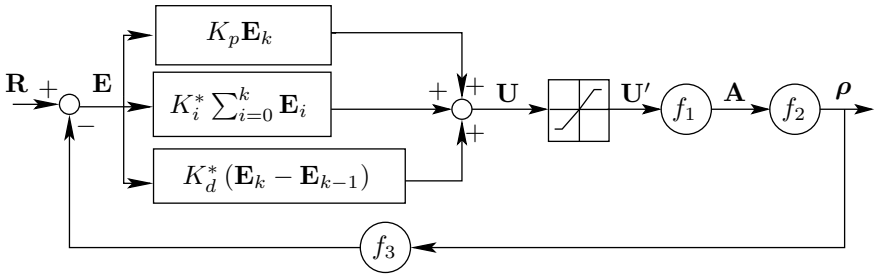


Figure 5–2: Surface mapping feedback PID controller

5.1.2 Control Technique

The main advantage of having the controller operate at the surface level is that measured locations need not correspond to controlled locations. Additionally, the computational cost of running a single controller loop at the surface level is much lower than operating many loops at the point level. No specific controller is required, though we will demonstrate the principles of SMF control with a PID controller, since it has a relatively simple implementation and widespread use, with 95% of controllers used in process control of the PID or PI type [66, p.1]. The SMF PID controller is shown in Fig. 5–2.

Normally, PID control is implemented in the time domain, while for SMF, the spatial domain of the part height h is used; the time step then becomes the constant interval Δh_m. Additionally, PID control for SMF must be *discrete*, since the smallest possible value that Δh_m can assume is the deposition layer thickness Δh_d. The error signal $e(t)$ of PID controllers becomes an error surface, defined by the function $E(x, y, h)$, or, upon discretization, an error matrix $\mathbf{E}_k$, with the dimensions of a grid which has a user-specified resolution and boundaries defined by the footprint of the part and scaffolding at height h_k, for measurement layer k. Equivalently, the controller output $u(t)$ becomes $U(x, y, h)$ or, upon discretization, matrix $\mathbf{U}_k$. The proportional, integral, and derivative gains are, as usual, K_p, K_i, and K_d. The functions f_1 and f_2 depend on the process parameter used to implement the control action, as explained in Subsec. 5.1.3. The function f_3 depends on the surface measurement technique used.

Updating and discretizing the standard PID controller, we obtain the control law

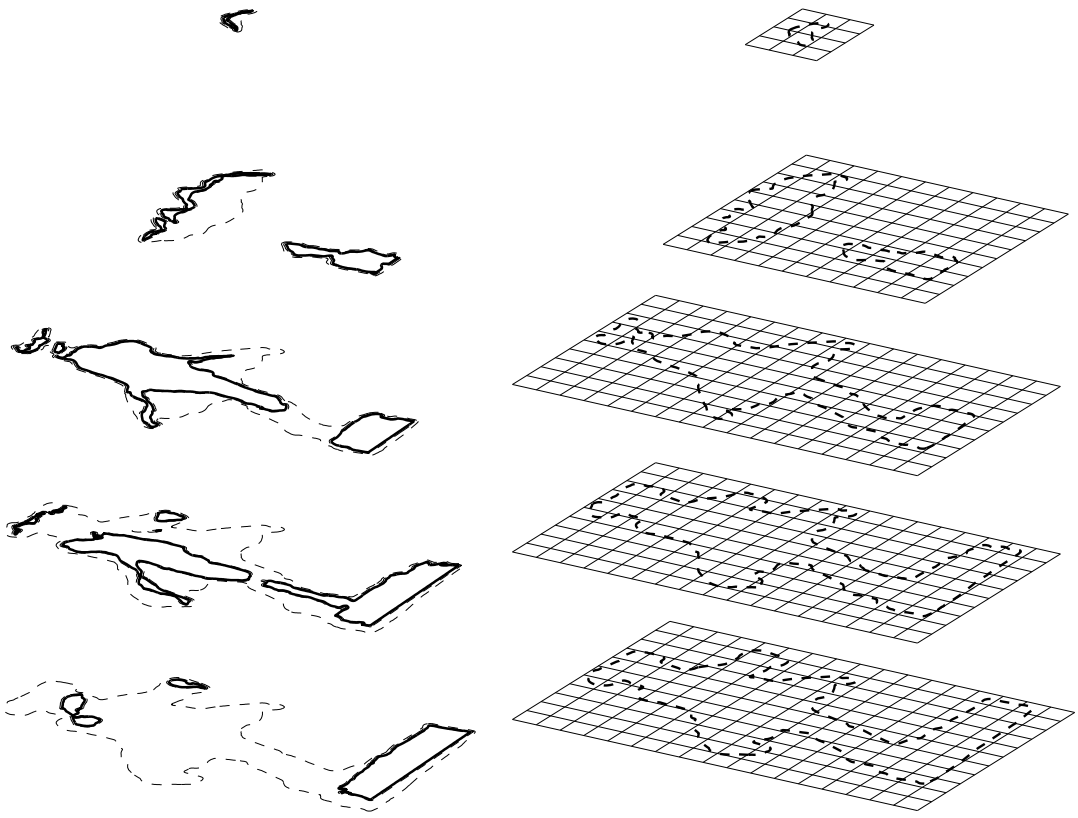
$$\mathbf{U}_k = K_p\mathbf{E}_k + K_i\sum_{i=1}^{k}\mathbf{E}_i\Delta h_m + \frac{K_d}{\Delta h_m}\left(\mathbf{E}_k - \mathbf{E}_{k-1}\right) \tag{5.1}$$

where $k > 1$; for $k = 1$, the derivative term is left out. Integral and derivative gains are modified to produce adjusted gains for Eq.(5.1) that all have the same units:

$$K_i^* = K_i\Delta h_m, \qquad K_d^* = \frac{K_d}{\Delta h_m}. \tag{5.2}$$

The integral and derivative terms in Eq.(5.1) call for the summation of surfaces defined on different (x, y) grids. Since this summation cannot be performed directly, the matrix terms from the previous measurement layer $k-1$ are first evaluated at the points on the (x, y) grid for layer k. This transformation is straightforward using scientific software. For example, in Matlab, a surface interpolant `F` for a grid of $m \times n$ points is expressed as a structure where `F.X` is a $2\times mn$ matrix of (x, y) coordinates and `F.V` is a mn-dimensional vector of surface heights. `F` can be expressed in a new grid `X'` using the commands `F.V=F(X')` and `F.X=X'`. If the grid for h_k is larger than that for h_{k-1}, extrapolation could be used, or the initial integral and derivative contribution for new gridpoints could simply be set to zero. A graphical representation of the different matrix dimensions at different slice elevations for a CAD part is shown in Fig. 5–3.

The SMF PID control implementation described above is quite unique; the most closely-related application would be temperature control of a plate, which has been implemented by Gorinevsky et al. [67], among others. However, there are several key differences: firstly, the domain for temperature control is time; secondly, the variable being controlled is the 2D surface representing the temperature on the entire surface of the plate; lastly, the domain or grid of points for the 2D temperature surface is unchanging. In SMF PID control, the domain is the nominal part height, which is spatial, the variable being controlled is the 2D surface representing the actual part



(a) (b)

Figure 5–3: SMF 3D PID controller matrix dimensions for the part shown in Fig. 3–1(a): (a) Part (solid) and scaffolding (dashed) contours for five layers of the part; (b) Merged regions for the same five layers, with the grids at each layer defining the dimensions of the matrices in Eq.(5.1)

height at the nominal part height, which is also spatial, and the size of the 2D surface changes at each iteration of the controller loop.

5.1.3 Deposition Adjustment

For drop-on-demand systems, probably the simplest implementation of SMF would entail meshing the part and scaffolding cross-sections at each height h_k and using the **U** matrix to determine how many drops per gridpoint are needed to reach the next measurement plane. A more complex formulation, which could lead to enhanced accuracy, might consist of using warped pixels, defined by the part boundaries, along with interior pixels to fill in the shells.

For continuous-flow systems, a nominal deposition control data set is assumed to be available for adjustment. This data set is divided into deposition layers, each consisting of a series of deposition paths. Each path is assumed to include a mathematical description of the path itself, along with any other variables needed for controlling deposition.

Part measurement takes place every n_d deposition layers, where n_d is a user-defined parameter. With Δh_d being the nominal layer height for the material, we have $\Delta h_m = n_d \Delta h_d$. Deposition adjustment at measurement layer k is thus achieved by applying $\mathbf{U}_k$ to the n_d deposition layers above h_k.

Adjustment consists of manipulating the material volume deposited per distance traveled

$$V' = \frac{Q}{c} \tag{5.3}$$

where Q is the material flow rate and c is the deposition path speed.

The controller output $\mathbf{U}$ can be applied to any process parameter ρ that can be varied precisely at all points in the RP workspace and has a reasonably well-known impact on Q and/or c. The entries of $\boldsymbol{\rho}$, the system output, are the value of ρ, at each point along a deposition path. Extrusion temperature, deposition line pressure, and many other variables cannot be used for ρ, since they cannot be varied *quickly* enough to achieve the needed flow rate variations along a deposition path.

One highly-responsive process parameter that directly impacts flow rate for some RP systems is the control signal used for a valve, discussed in detail in Subsec. 1.2.4. Pulse width modulation (PWM) or frequency modulation (FM) of this signal could be used to vary the flow rate precisely and rapidly along a deposition path. The path speed c can also be varied to implement the desired control, though for some RP systems, c cannot be varied precisely along a deposition path.

The controller output $\mathbf{U}$ is used to update ρ along each path in the n_d layers to be adjusted, as shown graphically in Fig. 5–4. For this figure, ρ is a parameter that should *decrease* with the controller output, such as the deposition path speed c. The control action matrix $\mathbf{A}$ is produced using function f_1 in Fig. 5–2, which reduces to

$$\mathbf{A}_k = C_u \left(\mathbf{\Gamma}_k + \mathbf{U}'_k \right) \tag{5.4}$$

where $\mathbf{\Gamma}_k$ is a matrix with entries all equal to 1, while C_u is a constant of proportionality. The vector $\boldsymbol{\rho}$, whose entries indicate the value of the process parameter at each point i along the path, is produced according to

$$\rho_i = A(x_i, y_i, h_k) \tag{5.5}$$

where $A(x, y, h_k)$ is the surface interpolant representation of $\mathbf{A}_k$. Equation (5.5) implements function f_2 in Fig. 5–2.

In the above derivation, it is assumed that detected height errors are *small*, i.e., smaller than Δh_m. If large errors are present, large changes in ρ along a path will be called for, which might not be possible, since many process parameters that could be useful in implementing SMF will only be *partially* controllable. For example, if the path speed c is used as the process parameter, velocity, acceleration, and jerk constraints need to be respected.

This problem can be minimized by imposing saturation at different steps in the algorithm. A simple but robust implementation would involve the definition of upper and lower saturation planes to bound the controller output U, as shown in Fig. 5–4. Additionally, filtering can be used to prevent rapid changes in ρ along a path.

5.2 Test Platform: The Cobra 600 RFP System

Surface mapping feedback has been implemented with the Cobra 600 rapid freeze prototyping (RFP) system, which is of the continuous-flow type. SMF is integrated with part-slicing and trajectory control, the subjects of Chs. 3 and 4, respectively.

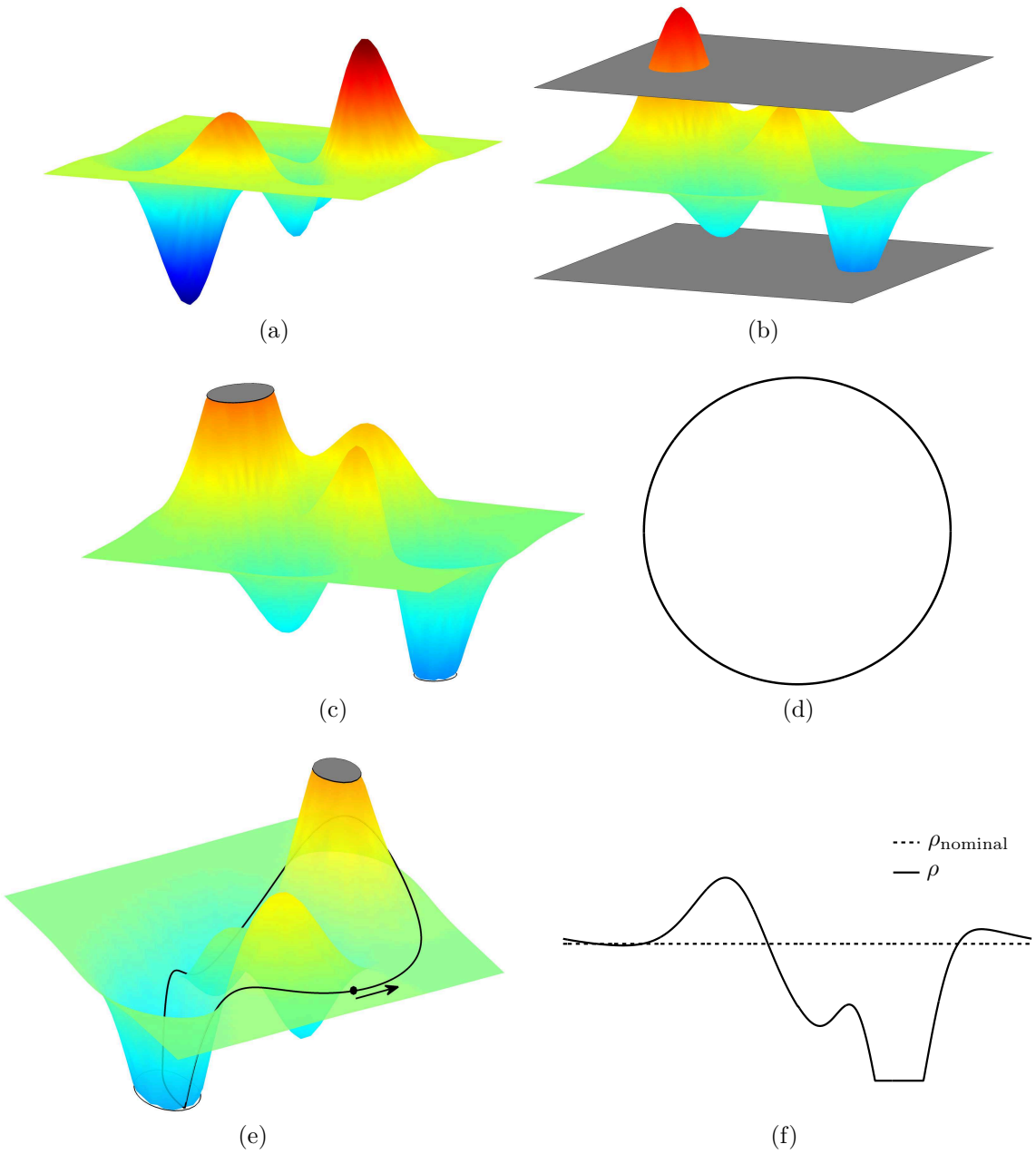
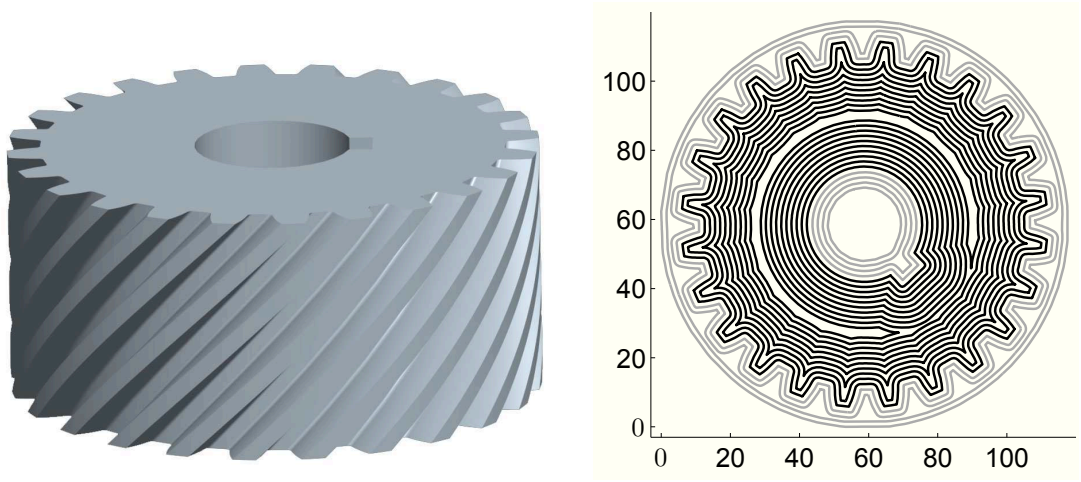


Figure 5–4: SMF PID controller steps: (a) measured surface; (b) controller output U; (c) saturated controller output U'; (d) a deposition path; (e) control action A applied to produce the process parameter ρ along the deposition path; (f) 2D representation of (e), in the counterclockwise direction

The SMF software implementation consists of a robot control program written in the V+ programming language developed by Adept Technology, which runs on the Cobra controller, along with a program written in Matlab code, which runs in parallel



(a) (b)

Figure 5–5: (a) CAD model of a helical gear used to compare SMF to open-loop deposition control: height is 50 mm and outside diameter is 107.6 mm; (b) Measurement paths for one layer for the part shown in (a). Gray lines indicate scaffolding paths while black lines indicate part paths. Axes indicate distance in mm.

on the control PC. Since SMF is computationally intensive and the computational resources of the controller[1] are limited, the bulk of the communication and processing load is accomplished with the Matlab code, on the control PC. A diagram of signal and data traffic during part construction is shown in Fig. 2–5.

5.2.1 Surface Measurement

The measurement of distance to an ice part is a particularly difficult problem because of the reflective properties of ice. After consideration of several devices, a laser displacement system from Keyence, Inc., consisting of the LK-G32 laser and the LK-G3001P laser controller, was selected; relevant specifications for this system are listed in Table 2–3.

The function f_3 in Fig. 5–2 corresponds to the laser measurement operation. Vertical measurements are taken with the Keyence system while the robot follows a

[1]An Adept C40 Compact Controller with an AWC-II 040 Processor (25 MHz), 32 MB RAM, and a 128 MB CompactFlash disk

series of paths in a reference frame defined for the laser. The part-slicing algorithm RPSLICE, described in Ch. 3, is used to produce these measurement paths, before construction begins. While the EE follows each path, measurements are accumulated at a rate of 100 Hz; after each path is followed, these data are sent to the control PC via RS-232C. The measurement paths for one layer of a helical gear part are shown in Fig. 5–5(b). The laser measurements are concatenated into an array of (x, y, z) coordinates, which are combined to form an interpolation surface using the Matlab function TRISCATTEREDINTERP. This surface is then evaluated on the rectangular grid spanning all part and scaffolding regions for the layer, shown graphically in Fig. 5–3.

It was found experimentally that the laser produces the highest quality measurement data if the surface being measured has *uniform* reflective properties and long paths are followed at low speed. To satisfy these requirements, the measurement fill paths are produced with the concentric contour-shrinking technique BUFFERF, described in Ch. 3. Additionally, only one of the two materials is measured along a single measurement path; the laser measurement settings are thus optimized separately for the scaffolding and the ice.

The laser is mounted onto robot end effector, near the deposition nozzles. The position of the laser in the Cobra base frame must be known within 0.2 mm, and the latency of laser measurements with respect to EE position on the commanded paths must both be known within 0.002 s. These tolerances are chosen to be well within the desired part accuracy of 0.5 mm. The latency tolerance introduces a maximum error of 0.25 mm, since the maximum measurement paths speed is 125 mm/s. Machining and assembly error are sufficient to prevent the determination of laser position within the desired accuracy, based only on the EE geometry. Additionally, measurement latency depends partly on the laser controller latency upon receiving a command signal, which is known within 0.001 s, and partly on the delay between the sending of

this signal from the robot controller and the start of the motion by the robot, which can only be estimated.

Manual calibration procedures are therefore used to determine the laser offset and measurement latency within the desired accuracy. The laser offset is found by passing the end effector over a test jig with known elevation changes and recording the position indicated by the joint encoders when expected elevation transitions are indicated by the laser. The measurement latency is found by mapping an object with a sharp elevation change. The laser is passed over this object using parallel paths traversed at a relatively high speed but in opposite directions. Laser measurements are used to produce an interpolation surface that should reproduce the actual topography of the object. Any error in the latency is thus indicated by skewed topography at the common edge between paths traversed in opposite directions.

The SMF PID controller loop, shown in Fig. 5–2, can operate in dimensioned or dimensionless form. However, it is desirable to normalize quantities in the loop by the measurement interval Δh_m, to simplify calculations. The controller input $\mathbf{R}_k$, which is a horizontal plane, then becomes the normalized nominal height of the part, given simply by $k\mathbf{\Gamma}_k$, where $\mathbf{\Gamma}_k$ is a matrix of entries identical to 1 with dimensions equal to those of $\mathbf{E}_k$. The function f_3 therefore produces a matrix which represents the measured height of the part, normalized, at all locations on the horizontal grid for layer k.

5.2.2 Control Technique and Deposition Adjustment

The generalized version of the PID control technique introduced in Subsec. 5.1.2 must be adapted for the specific properties of the Cobra 600 RFP system. This consists of defining the saturation blocks and the functions f_1 and f_2 of Fig. 5–2.

Both material flow and EE speed could be used to implement the SMF control action. Material flow could be varied along a path through pulse-width modulation or frequency modulation of the valve control signal, as explained in Subsec. 5.1.3.

EE speed c can be precisely controlled with the Cobra 600 RFP system, but not for gantry-type RP systems that use open-loop positioning control.

For this section, the path speed c is therefore used as the process parameter; this version of the SMF PID controller will be referred to as the "c-controller" from now on, which is distinguished from the valve frequency version, called the ν-controller, introduced in Sec. 5.4 and used in Ch. 6. The gains for the c-controller, which have been manually tuned, are $K_p = 1$, $K_i^* = 0.25$, and $K_d^* = 0.25$.

Nominally, i.e. if the error signal is zero, the flow rate Q is given by

$$Q = cw\Delta h_d \tag{5.6}$$

Therefore, if Q and w are held constant, the path speed required by the controller is

$$c = \frac{Q}{(u+1)w\Delta h_d} = \frac{c_{\text{nom}}}{(u+1)} \tag{5.7}$$

The (i, j) entries of the control action matrix $\mathbf{A}$ are thus given by

$$A_{ij} = \frac{Q}{(u_{ij}+1)w\Delta h_d} = \frac{c_{\text{nom}}}{(u_{ij}+1)} \tag{5.8}$$

Each entry of $\mathbf{A}$ represents the end-effector speed required at a point in the rectangular grid spanning the top surface of the part for layer k. Equation (5.8) corresponds to Fig.5–4(c).

The surface function $A(x, y, h_k)$ is evaluated via interpolation among the entries of $\mathbf{A}$. $A(x, y, h_k)$ is used to produce a vector $\mathbf{c}$ of required speeds along each path for deposition layers $k+1$ to $k+n_d$, according to

$$c_i = A(x_i, y_i, h_k) \tag{5.9}$$

for all path points i of $\mathbf{c}$. The application of Eq.(5.9) to a single path is shown in Fig. 5–4(e–f). Equations (5.8) and (5.9) correspond to functions f_1 and f_2 in Fig. 5–2, respectively. The required speed vectors produced by Eq.(5.9) cannot be used directly, since they could violate kinematic and dynamic constraints for the robot. Additionally, if the flow rate per speed Q/c is too high or too low, flow characteristics could become unacceptable: high values of Q/c could produce runoff of liquid material following deposition; low values of Q/c could produce noticeably discontinuous paths. Q/c is kept within acceptable limits by specifying saturation limits on $\mathbf{U}$, as shown in Fig.5–4(b). For the c-controller, saturation planes for $\mathbf{U}$ are defined such that c remains within 30% of c_{nom}.

Saturation limits imposed on $\mathbf{U}$ could lead to large growth of the integral term, without affecting the saturated control output. If this situation arises for a subarea of the part and the nominal height is then subsequently reached in this subarea, the integral term could contribute to a controller output that moves this subarea *away* from the setpoint. For the RFP system, this most often arises when the system shifts from depositing water to SME or vice versa in the subarea. To avoid this situation, saturation is also imposed on the integral term.

An offline control method is implemented for the Cobra 600 RFP system, for two reasons. Firstly, the offset between the laser and the deposition nozzles would complicate path planning in an online system, since the EE would need to follow the same path in two reference frames. Secondly, online control would need to be implemented on the Cobra controller, which cannot handle the proposed, computationally-intensive, control scheme.

It should be noted that path speed c cannot be controlled directly along a path in V+, the programming language used by the Adept C40 controller. Indirect control by adjustment of point-to-point spacing or point-to-point duration is used instead. Since point-to-point duration is inversely proportional to path speed, the adjusted vector of

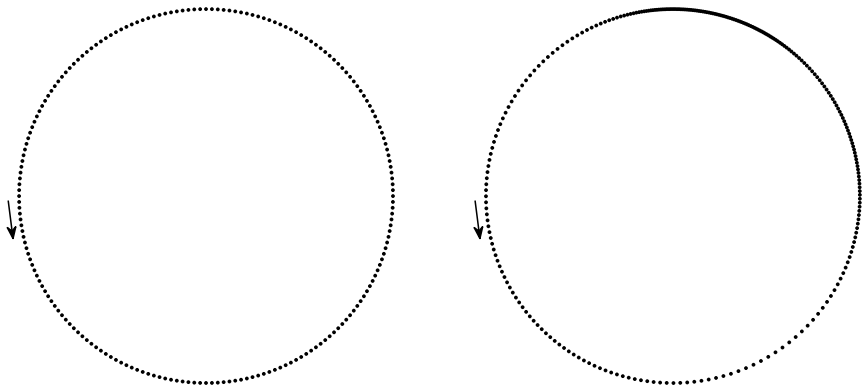


Figure 5–6: Variable point-spacing, with constant point-to-point duration, used to produce the speed required along the path shown in Fig. 5–4: (a) nominal, constant point-spacing; (b) variable point-spacing (spacing variation has been exaggerated to be more visually discernible)

point-to-point durations $\boldsymbol{\delta}_{\text{adj}}$ for a path is produced by modifying the nominal version $\boldsymbol{\delta}_{\text{nom}}$, according to

$$\delta_{i,\text{adj}} = U(x_i, y_i)\delta_{i,\text{nom}}. \tag{5.10}$$

Unfortunately, unpredictable, jerky motion is often exhibited when the Cobra 600 is commanded to follow trajectories with highly variable δ values, even when the trajectories are theoretically smooth. Therefore, path point-spacing is adjusted instead, holding point-to-point duration constant. This technique is shown in Fig. 5–6. The conversion from uniformly-sampled space paths to uniformly-sampled time paths is accomplished using linear interpolation. Figure 4–11 shows the variable point-spacing technique applied to an entire layer of a part.

5.2.3 The Drop-On-Demand Case

If a drop-on-demand (DOD) deposition system were used with the Cobra 600 RFP system, SMF control would be much simpler, since a nominal path data set would not be needed and saturation limits on the control output would probably not be necessary. A logical process parameter would be droplets per gridpoint, for a grid

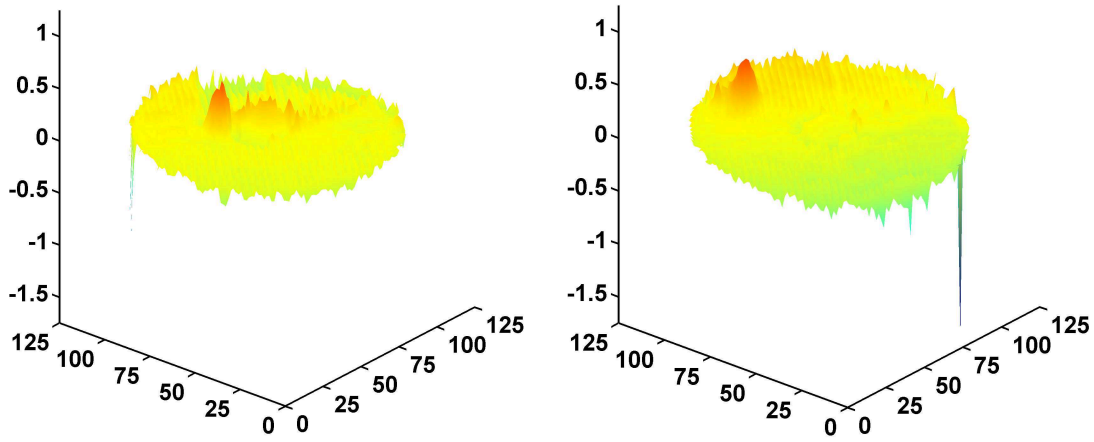
Figure 5–7: The final helical ice gear, constructed with SMF, after the scaffolding is removed

defined as in Subsec. 5.1.3. The number of drops required for one gridpoint would be proportional to the measured height error there. DOD is not used with the Cobra 600 RFP system because low or discrete material flow is less predictable with the currently used flow-control hardware. Additionally, part construction times would be much longer with DOD.

5.3 Results With The Cobra 600 RFP System

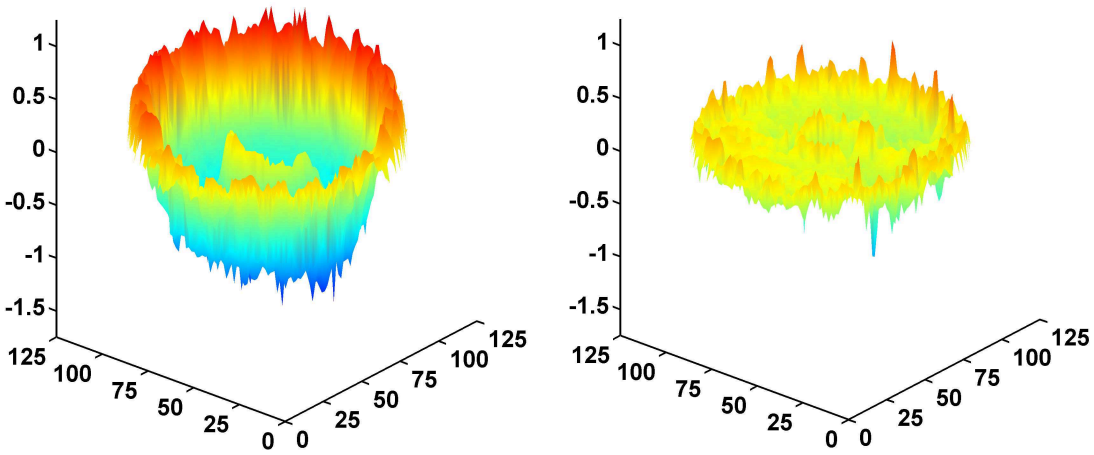
The helical gear part shown in Fig. 5–5 was built with the Cobra RFP system to demonstrate the influence of SMF on part dimensional accuracy. The measurement interval Δh_m used was 1.2 mm. Figure 5–7 shows the final part, constructed with SMF, after the scaffolding has been removed. Figure 5–8 shows surfaces measured with the laser for both the SMF and open-loop cases. The surfaces shown indicate *raw* height measurements, which are smoothed using a N-D smoothing function developed by Garcia [68], before path adjustment. The RMS deviation from the nominal or zero plane σ_l^* for each measured surface, in both the SMF and open-loop cases, is plotted as a function of part height in Fig. 5–9.

In both the open-loop and SMF cases, part construction time, to a height of 31.2 mm was approximately 16 hours, since surface measurements were taken in both



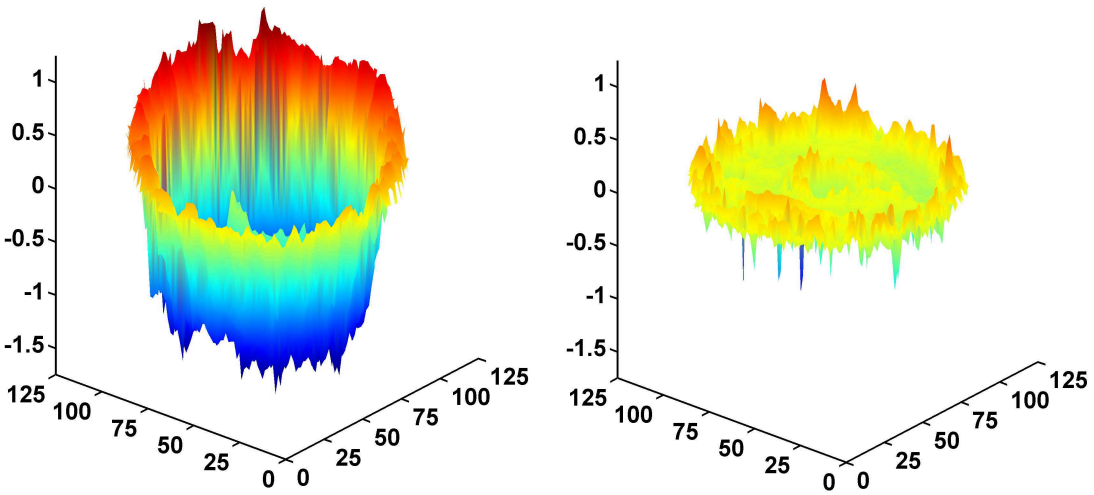
(a) SMF off: $h =$ 0.4 mm, $\sigma_l^* = 0.072$ mm

(b) SMF on: $h =$ 0.4 mm, $\sigma_l^* = 0.097$ mm



(c) SMF off: $h =$ 15.6 mm, $\sigma_l^* = 0.469$ mm

(d) SMF on: $h =$ 15.6 mm, $\sigma_l^* = 0.106$ mm



(e) SMF off: $h =$ 31.2 mm, $\sigma_l^* = 0.843$ mm

(f) SMF on: $h =$ 31.2 mm, $\sigma_l^* = 0.116$ mm

Figure 5–8: 3D surface plots, with and without SMF, for the helical gear part shown in Fig. 5–5. All axes indicate distance in millimetres; σ_l^* is the RMS deviation of the surface from the nominal or zero plane, for one layer

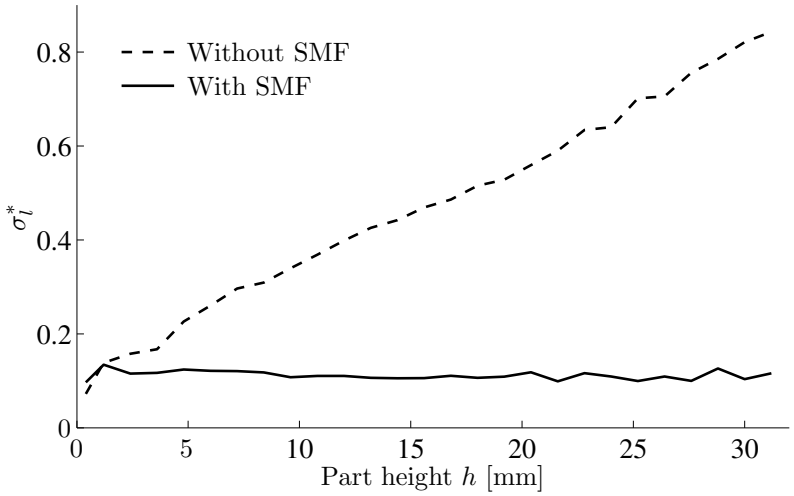


Figure 5–9: Comparison of part accuracy, with and without SMF, for the helical gear part shown in Fig. 5–5, built 31.2 mm high (27 measurement intervals)

cases. Deposition time per measurement interval Δh_m was 30 minutes, while measurement time was 10 minutes. Therefore, construction time was about 33% higher using SMF than it would be by using open-loop control. To reduce the total measurement time, Δh_m could be increased and/or the measurement path resolution, shown in Fig. 5–5(b), could be coarsened, though there would likely be an associated sacrifice in part accuracy.

The RMS deviation from the nominal height for the final layer is 0.12 mm with SMF online, and 0.84 mm for open-loop control, showing that SMF greatly enhances the part dimensional accuracy; for this case, vertical accuracy is the same or better than in the horizontal directions. As can be seen in Fig. 5–9, the RMS height error with SMF stays relatively constant, near 0.1 mm, while the error without SMF is roughly proportional to part height. We can thus claim that SMF successfully maintains the vertical error below a certain threshold, in this case, approximately 0.13 mm. Additionally, for open-loop control, it can be seen that the height error surface deviates from the nominal height by more than 1 mm in some locations, which is dangerous, as the EE could eventually scrape or collide with high part sections. Also, when the EE-part clearance becomes too high, stream geometry is less predictable

and part errors tend to increase at a faster rate. In these cases, the part construction must often be aborted or adjustment of the process parameters is needed during construction, rendering the system only *semi-automated*.

The CAD part shown in Fig. 4–15 was constructed to verify the dimensional accuracy of the Cobra 600 RFP system, with SMF, MTTS, and VFVC all online. MTTS, or minimum-time trajectory shaping, and VFVC, or variable flow valve control, are described in Chs. 4 and 6, respectively. Each measurement shown in Fig. 5–10 was taken at four different locations using a vernier caliper; the measurements are compared to the dimensions of the CAD file in Table 5–1. It can be seen that the differences between the nominal and measured values for most of the measurements made lie within the claimed accuracy of 0.5 mm. Additionally, with the exception of l_3, the measured values are all larger than the nominal values. This is to be expected, since the ice surface is rough, and the caliper measures to the outer edge of the roughness. Slicing parameters introduced in Ch. 3 could be adjusted to centre most of the measurements about the nominal values, if desired. The diagonal dimension l_3 is smaller than the nominal value, which is expected, since the 90° corner is rounded off by one half the deposition path width. Also, measurement with the caliper crushes the sharp edge slightly.

These measurements show that the error in the vertical and horizontal directions is approximately the same, confirming that closed-loop geometry control is only needed in the vertical direction for the Cobra 600 RFP system.

5.4 One-Dimensional Version of the SMF PID Controller

To reduce the complexity of the SMF PID controller, a one-dimensional version is now introduced for use in stability analysis and gain optimization. The c- or speed version of the SMF PID controller for the Cobra 600 RFP system was introduced in Sec. 5.2. Here, we introduce the ν- or valve frequency version. Figure 5–11 shows a diagram of the ν-controller, which can be described with the relations below:

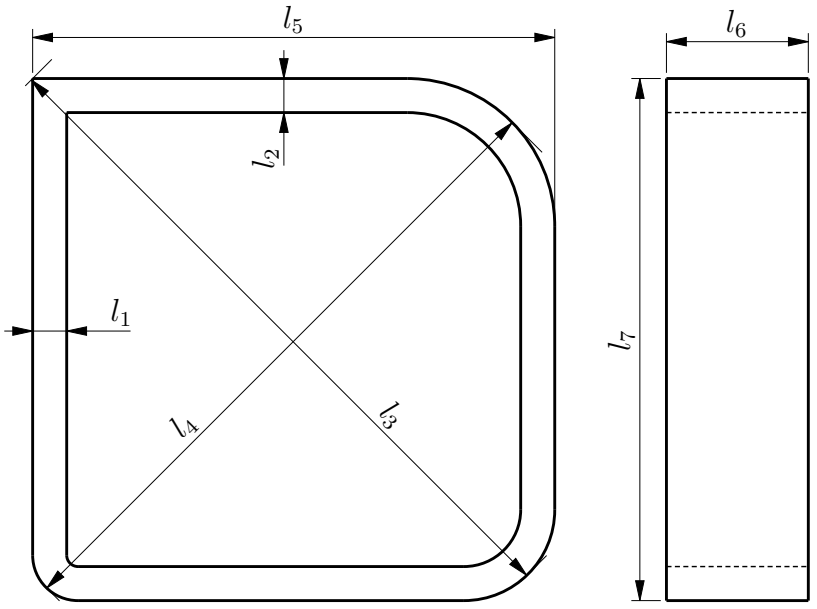


Figure 5–10: Dimensions of the CAD part shown in Fig. 4–15

Table 5–1: Nominal values for the parameters indicated in Fig. 5–10, compared to measured values for the constructed ice part

Parameter	Nominal (mm)	Measured (mm)			
		#1	#2	#3	#4
l_1	6.0	6.2	6.2	6.6	6.3
l_2	6.0	6.0	6.1	6.3	6.4
l_3	123.5	122.9	122.9	122.7	122.5
l_4	116.0	116.0	116.0	116.4	116.2
l_5	92.0	92.3	92.2	92.4	92.3
l_6	25.0	25.7	25.7	25.8	25.6
l_7	92.0	92.5	92.5	92.8	92.7

$$u_k = K_p e_k + K_i \sum_{i=1}^{k} e_i \Delta h_m + \frac{1}{\Delta h_m} K_d \left(e_k - e_{k-1}\right) \tag{5.11a}$$

$$\nu_k = C_f \left(u_k + 1\right) \tag{5.11b}$$

$$e_{k+1} = r_k - \sum_{i=1}^{k} \left(\frac{\nu_i}{C_f} + d_i \right) \tag{5.11c}$$

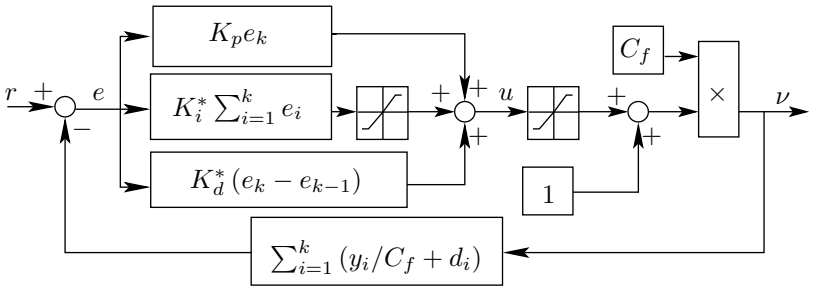


Figure 5–11: Surface mapping feedback 1D PID controller

where h is the part height, Δh_m is the layer thickness, and $e_k = e(h_k)$, with u_k, y_k, and d_k defined similarly. Modified integral and derivative gains are introduced to eliminate Δh_m from Eq.(5.11a), as shown in Eqs.(5.2). The quantities e_k, u_k, y_k, and d_k are *normalized* by the measurement interval Δh_m, to make the controller dimensionless. Dimensioned quantities can be obtained, if needed, through multiplication by Δh_m.

The input signal r is the part height *in layers*, given by

$$r_k = \frac{h_k}{\Delta h_m} = k. \tag{5.12}$$

C_f is the constant of proportionality for converting the controller output into a required valve frequency. The valve flow rate Q is given by

$$Q = (u + 1) c w \Delta h_d. \tag{5.13}$$

The valve frequency, which is the controller output y, is related to the valve flow rate by $y = c_{\text{ff}} Q$. Combining this with Eq.(5.13), we have

$$C_f = c_{\text{ff}} c w \Delta h_d. \tag{5.14}$$

Disturbances are lumped into the quantity d_k. Uncertainty in parameters c_{ff}, v, and w all affect d_k. Other contributions to d_k include debris falling on the part and variation of system performance between the edge of the part and the interior.

It is important to note that the lowest block of Fig. 5–11,

$$\sum_{i=1}^{k} \frac{\nu_i}{C_f} + d_k, \tag{5.15}$$

assumes this form only for *simulation* purposes. For the Cobra 600 RFP system, the laser displacement sensor measures the quantity $-e$ directly, because it is mounted onto the EE and undergoes vertical displacements *identical* to those of the deposition tools.

5.4.1 Model Simplifications and Assumptions

Three major simplifications/assumptions have been made in producing the controller model, described by Eqs.(5.11): firstly, the three-dimensional spatial PID controller has been reduced to a one-dimensional spatial controller; secondly, saturation terms have been neglected; thirdly, the controller is assumed to be linear shift-invariant.

In the context of SMF, the one-dimensional model corresponds to controlling the height of a single gridpoint or pixel in the horizontal plane, from one measurement stage to the next. However, the 3D PID control loop is a matrix equation, where the matrix dimensions for each measurement stage k are defined by the uniform rectangular grid that spans Boolean union of the part and scaffolding regions for the last layer to be deposited. Therefore, the matrix changes dimensions at each k, with the dimensions always *decreasing*, given that scaffolding must fill all empty space intersected when projecting the part geometry downward; this can be observed in Fig. 5–3.

For the 3D PID controller, the system output for one deposition path is the vector $\boldsymbol{\nu}$, whose entries are the value of the valve frequency ν at each path point, formed via interpolation among the entries of $\mathbf{A}$, at the horizontal coordinates of the path points.

If stability is shown individually for all possible points on the surface, it may then be argued that stability is also shown for interpolated points on the surface and

then for the surface itself. However, this statement, along with any other conclusions reached using the 1D controller, is only theoretically valid if the pixels behave *independently*. Experimentally, the validity of the analysis below has been confirmed for parts with unrestricted 3D geometry.

Saturation terms have also been been neglected in Eq.(5.11) to reduce model complexity. Including saturation functions would greatly complicate stability analysis and gain optimization. In practice, the saturation blocks are very important because they act as a fail-safe system. If valve frequency is the control action, saturation keeps the requested frequency within the range of valve capabilities. If EE speed is the control action, saturation is even more important because it prevents negative and/or extremely high speeds.

Including saturation in Eq.(5.11) is not necessary because, during normal operation, the system should only saturate and remain saturated in three extreme cases:

- the requested part geometry is obviously impossible for the robotic system to build
- the user has made a significant error in parameter configuration during part-slicing, described in Ch. 3
- one of the fluid tanks has become empty or a fluid line has become blocked

The controller is assumed to be linear shift-invariant, which is the discrete equivalent of linear time-invariance. For SMF, there is a linear relationship between the input r and the controller output u, though not between r and the valve frequency ν, since there is a nonlinear relationship between flow rate Q and ν. However, since the nonlinearity is contained within the plant, it does not affect the controller analysis because the disturbance d is introduced during feedback, as shown in Fig. 5–11. The system is also not space-invariant in the spatial domain h because h appears in the input. However, in the stability analysis below, a new system is derived where the disturbance d_k is the input and the error e_k is the output; this system is invariant,

i.e. identical input disturbances introduced at k and $k+K$, with K being a positive integer, will produce the same output error.

5.5 Stability Analysis

An analysis is now conducted to identify under which conditions the 1D SMF PID controller is stable. First, Eqs.(5.11) are recast into a more standard form. Taking the difference between adjacent layers for Eq.(5.11c), we obtain

$$e_{k+1} - e_k = 1 - \frac{\nu_k}{C_f} - d_k. \tag{5.16}$$

Substituting Eq.(5.16) into Eq.(5.11b) and solving for u_k yields

$$u_k = e_k - e_{k+1} - d_k. \tag{5.17}$$

Substituting Eq.(5.17) into Eq.(5.11a) and again taking the difference between adjacent layers yields

$$e_{k+1} - 2e_k + e_{k-1} + K_d^* \left(e_k - 2e_{k-1} + e_{k-2}\right) + K_p \left(e_k - e_{k-1}\right) + K_i^* e_k = d_{k-1} - d_k. \tag{5.18}$$

Regrouping Eq.(5.18), we obtain

$$e_{k+3} + \left(K_p + K_i^* + K_d^* - 2\right) e_{k+2} + \left(1 - K_p - 2K_d^*\right) e_{k+1} + K_d^* e_k = d_{k+1} - d_{k+2}. \tag{5.19}$$

To convert Eq.(5.19) into state-space form, we set $f_k = e_{k+1}$ and $g_k = e_{k+2}$ to obtain

$$\begin{bmatrix} e_{k+1} \\ f_{k+1} \\ g_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ C_1 & C_2 & C_3 \end{bmatrix} \begin{bmatrix} e_k \\ f_k \\ g_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ d_{k+1} - d_{k+2} \end{bmatrix} \tag{5.20}$$

with $C_1 = -K_d^*$, $C_2 = K_p + 2K_d^* - 1$, and $C_3 = 2 - K_p - K_i^* - K_d^*$. Alternatively, applying the the z-transform to Eq.(5.19) [69, p. 92], with d_k and e_k being the input and output, respectively, we find

$$\begin{aligned} zD(z) - z^2D(z) = z^3E(z) + (K_p + K_i^* + K_d^* - 2)\, z^2E(z) \\ + (1 - K_p - 2K_d^*)\, zE(z) + K_d^*E(z). \end{aligned} \tag{5.21}$$

The system transfer function $H(z)$ is then given by

$$H(z) = \frac{E(z)}{D(z)} = \frac{z - z^2}{z^3 + (K_p + K_i^* + K_d^* - 2)\, z^2 + (1 - K_p - 2K_d^*)\, z + K_d^*}. \tag{5.22}$$

According to the Jury stability criterion [44, p. 80], the system is stable if all poles of Eq.(5.22) lie within the unit circle centered at the origin of the complex plane. Equivalently, the system is stable if the module of each eigenvalue is smaller than unity, where the module of a complex number $a + bi$ is given by $\sqrt{a^2 + b^2}$. The poles are given by the cubic roots of the denominator of Eq.(5.22). A closed-form solution for each of the three roots can be obtained, though the expressions are quite long and need not be shown here. If we denote the three poles by $z_{p,1}$, $z_{p,2}$, and $z_{p,3}$, the Jury stability criterion can be imposed through the inequalities

$$|z_{p,i}| < 1, \qquad i = 1 \ldots 3. \tag{5.23}$$

Since the restrictions imposed on K_p, K_i^*, and K_d^* by Eq.(5.23) are not immediately obvious, Maple 14 is used to solve the equations $|z_{p,i}| = 1$ symbolically, thereby producing the plane equations

$$K_p = 0, \qquad K_i^* = 0, \qquad K_d^* = 0, \qquad \frac{K_p}{2} + \frac{K_i^*}{4} + K_d^* = 1 \tag{5.24}$$

along with several extremely long constraints. It can be readily realized that the fourth plane equation corresponds to a pole of Eq.(5.22) at $z = -1$. Additionally,

test points can be tried to verify that the these equations define a gain-stable volume described by the four planar inequalities below:

$$K_p > 0, \qquad K_i^* > 0, \qquad K_d^* > 0, \qquad \frac{K_p}{2} + \frac{K_i^*}{4} + K_d^* < 1. \tag{5.25}$$

It is likely that the constraints listed by Maple on Eq.(5.24) apply to regions outside the gain-stable volume. To confirm this, we verify that the module of the maximum pole of Eq.(5.22), denoted $z_{p,\max}$, lies within the unit circle of the complex plane for all values of K_p, K_i^*, and K_d^* within the gain-stable volume. A visualization of the variation of $z_{p,\max}$ is also useful for optimizing the gains, since it can be used to estimate how far the controller is from instability for a particular configuration. Since the function $z_{p,\max}(K_p, K_i^*, K_d^*)$, subject to constraints (5.25), describes a hypervolume in $\mathbb{R}^4$, the desired visualization is produced through projections onto $\mathbb{R}^3$ and $\mathbb{R}^2$, as shown in Fig. 5–12.

The gain-stable volume is first divided into a three-dimensional grid of one-million points, with each gain axis divided into 100 points. The value $z_{p,\max}$ is found for each point using the ROOTS function in Matlab. The left-hand plots of Fig. 5–12 show the projection of the hypervolume in $\mathbb{R}^4$ onto planes in $\mathbb{R}^2$. Projections onto $\mathbb{R}^3$ produce volumes that have an upper and a lower surface; the right-hand plots of Fig. 5–12 show contour plots of the lower surfaces for each pair of gains.

Figure 5–12 shows that the controller is indeed stable within the gain-stable volume, since $z_{p,\max}$ is always less than unity. In addition to this conclusion, Fig. 5–12 provides some insight into the optimal values of the gains. We expect the optimal gains to produce a value of $z_{p,\max}$ much smaller than unity. It can be seen in Fig. 5–12 and verified in Eq.(5.24) that the values $K_p = 1$, $K_i^* = 1$, and $K_i^* = 0$ produces a triple pole at $z = 0$. Although no claim is made that minimizing the value $z_{p,\max}$ yields the optimal gains, this provides an interesting reference for comparison.

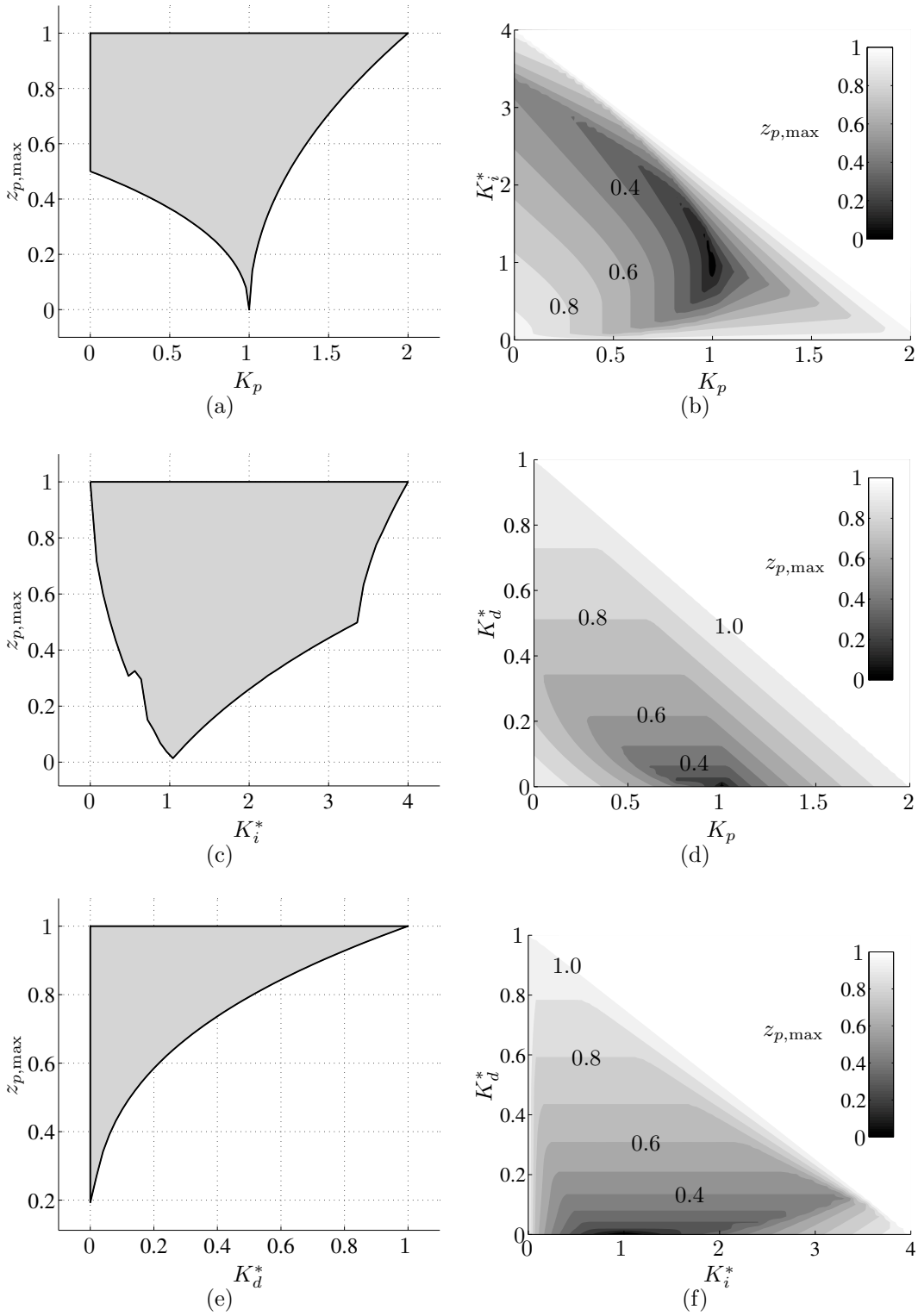


Figure 5–12: The maximum pole $z_{p,\max}$ of Eq.(5.24), shown at all locations within the gain stable volume, constrained by Eqs.(5.25). Since this is a four-dimensional relation, it is shown graphically by projections onto each gain axis (left-hand plots), and contour plots of the minimum values of $z_{p,\max}$ for each gain axis pair.

5.6 Optimization of the PID Controller Gains

In addition to ensuring that the gains produce a stable controller, it is desirable to tune the gains for the process at hand. There exist many different techniques for tuning PID Controller Gains. Manual gain-tuning is implemented first, using a technique similar to the Ziegler-Nichols (ZN) method [45]. However, the ZN method is time-based and continuous, so it is not directly applicable to the spatial-discrete PID controller used for SMF. First, K_i and K_d are set to zero. K_p is increased until the output begins to noticeably oscillate; then, K_p is set at one-half this value to obtain the quarter-amplitude decay recommended by Ziegler and Nichols as a good compromise for most applications. Then, K_i is increased until any steady-state error is eliminated in sufficient time. Finally, K_d is increased to improve the system response to sudden disturbances; K_d can also be used to limit the overshoot caused by K_i. Using this procedure, values of $K_p = 0.8$, $K_i^* = 0.2$, and $K_d^* = 0.1$ were found for the $\nu-$version of SMF PID controller.

In Sec. 5.3, the performance of the $c-$version of the controller during the construction of an ice part is reported. Values of $K_p = 1$, $K_i^* = 0.25$, and $K_d^* = 0.25$ were used in this case. These gains are normalized by the measurement height interval Δh_m, as explained in Subsec. 5.1.2.

The ZN gain-tuning method, and other methods that rely on trial-and-error, produce results that are stable and acceptable, but not optimal. Other tuning techniques exist that optimize the controller gains based on performance criteria. Lopez et al. developed tuning relations for three potential optimization criteria [46]: the integral of the squared error (ISE), the integral of the absolute value of the error (IAE), and the integral of the time multiplied by the absolute error (ITAE), given by the equations

$$ISE = \int_0^\infty e^2(t)dt \tag{5.26a}$$

$$IAE = \int_0^\infty |e(t)|\, dt \tag{5.26b}$$

$$ITAE = \int_0^\infty t\,|e(t)|\, dt. \tag{5.26c}$$

With the IAE method, the controller will be proportionally sensitive to the magnitude of the error; with the ISE method it will be much more sensitive to large errors. The ITAE method is insensitive to initial errors and hypersensitive to late errors. For a rapid prototyping system, the purpose of the PID controller is to minimize the total part error during the entire construction process, and correcting large errors quickly is also desirable. Therefore, the ISE method is the most appropriate. However, since the SMF PID controller is discrete, the optimization problem involves the minimization of the sum of the squared error (SSE). Minimization of the SSE is equivalent to minimizing σ_p^*, which is the RMS deviation from the nominal height for all measurements taken during part construction, given by

$$\min_{\mathbf{k}} \left(\sigma_p^*\right), \qquad \sigma_p^* = \left(\frac{1}{n_{\text{layers}}} \sum_{k=1}^{n_{\text{layers}}} e_k^2\right)^{1/2} \tag{5.27}$$

where $\mathbf{k}$ is the vector of design variables, which are the controller gains

$$\mathbf{k} = \begin{bmatrix} K_p & K_i^* & K_d^* \end{bmatrix}^T. \tag{5.28}$$

The optimal value of $\mathbf{k}$ obviously depends on the disturbance d_k. One option would be to set d_k to the highest value ever to be encountered in the system. This would produce very high gains, but then the controller would be suboptimal most of the time, and would most likely be less stable. A better option would be to set d_k to the maximum value *frequently* encountered in the system.

The disturbance can be divided into pulse, step, and random components:

$$d_k = d_{k,\text{pulse}} + d_{k,\text{step}} + d_{k,\text{random}}. \tag{5.29}$$

The pulse component represents deviation from nominal system performance for a single layer. Contributions to $d_{k,\text{pulse}}$ include debris falling on the part and deviation from flatness of the initial deposition surface. The step component represents sustained deviation from nominal system performance. Contributions to $d_{k,\text{step}}$ include inaccuracies in the valve frequency-flow rate relationship, error in approximating the sliced area with a series of one-dimensional paths, and variation of system performance at the edge of the part, compared to the interior. The random component $d_{k,\text{random}}$ represents disturbances that do not follow the pattern of the other two components. Contributions to $d_{k,\text{random}}$ can include laser measurement error, alignment error of the nozzles and the measurement laser, and variation of the frozen part geometry, depending on the geometry of the cross-section being deposited.

For a single-pulse disturbance of magnitude d_p at layer k_p, we have

$$d_{k,\text{pulse}} = \begin{cases} d_p & \text{for } k = k_p \\ 0 & \text{otherwise} \end{cases}. \tag{5.30}$$

For a single-step disturbance of magnitude d_s at layer k_s, we have

$$d_{k,\text{step}} = \begin{cases} d_s & \text{for } k \geq k_s \\ 0 & \text{otherwise} \end{cases}. \tag{5.31}$$

For random noise with amplitude d_r we have

$$d_{k,\text{random}} = d_r \left(RN - 0.5\right). \tag{5.32}$$

where RN is a random number between zero and one, separately generated for each k.

For the optimization problem at hand, only $d_{k,\text{pulse}}$ and $d_{k,\text{step}}$ need to be modeled, since $d_{k,\text{random}}$ is not expected to affect the optimal gains and could prevent convergence of the optimization algorithms. Random noise will be simulated when testing the optimal gains, to demonstrate the controller robustness. For the Cobra 600 system, the maximum values typically encountered are $d_p = 0.5$ and $d_s = 0.2$. Using these values, and setting $k_p = 60$ and $k_s = 30$, the problem has been fully formulated. The values d_p, d_s, k_p, and k_s listed here have been normalized by the measurement layer interval Δh_m, to be compatible with the other normalized quantities used in the derivation of Sec. 5.4; dimensioned values can be obtained through multiplication by Δh_m.

An important consideration is whether to treat this nonlinear multivariable minimization problem as constrained or unconstrained. If we constrain the problem to search within the gain-stable volume formed by the inequality relations of Eq.(5.25), stability is assured. However, even if the problem is unconstrained, the optimal solution is expected to lie within the gain-stable volume, since unstable gains will usually produce oscillations, leading to a higher value for the objective function σ_p^*.

Given these considerations, the optimization is performed both with and without constraints listed in Eq.(5.25), and the two cases are compared. The one-dimensional version of the SMF PID controller is used, shown in Fig. 5–11 and described by Eqs.(5.11). For unconstrained nonlinear multivariable minimization problems, there are three methods of solution that can be used: direct, gradient, and Newton. Direct methods are the most general, easiest to implement, and have the slowest rate of convergence. Gradient methods exhibit a linear convergence rate but require knowledge of the gradient, which is the vector first-order derivatives of the objective function,

with respect to the design variables. Newton methods exhibit a quadratic convergence rate but require knowledge of the gradient and the Hessian, which is the matrix of second-order derivatives, formed by the gradient of the gradient of the objective function.

Obviously, Newton methods are preferred, if the required information is available. The gradient and Hessian should be computed using analytical expressions, if possible, although they can be approximated using finite difference methods. However, finite differences require the evaluation of the objective function several times at each iteration of the solver, which leads to longer solution times. Additionally, finite difference approximation involves round-off and truncation error, which can lead to numerical instability and, in some cases, algorithm failure.

For the problem at hand, the gradient of the objective function $\nabla\sigma_p^*$ is given by

$$\nabla = \begin{bmatrix} \dfrac{\partial}{\partial K_p} & \dfrac{\partial}{\partial K_i^*} & \dfrac{\partial}{\partial K_d^*} \end{bmatrix}^T. \tag{5.33}$$

Applying the ∇ operator to Eq.(5.27), we obtain

$$\nabla\sigma_p^* = \frac{1}{\sigma_p^* n_{\text{layers}}} \sum_{k=1}^{n_{\text{layers}}} e_k \nabla e_k. \tag{5.34}$$

The gradient is applied to Eq.(5.19) to find ∇e_k

$$\begin{aligned} \nabla e_k = &\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} e_{k-1} + \left(2 - K_p - K_i^* - K_d^*\right) \nabla e_{k-1} + \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix} e_{k-2} \\ &+ \left(K_p + 2K_d^* - 1\right) \nabla e_{k-2} + K_d^* + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} e_{k-1} + K_d^* \nabla e_{k-3}. \end{aligned} \tag{5.35}$$

Matlab is used to perform both the constrained and unconstrained minimizations, where the FMINUNC and FMINCON functions are used, respectively. For FMINUNC, the objective function must be continuous and only local solutions are

reached; two algorithm options are available, `large-scale` and `medium-scale`. The large-scale option uses sparse linear algebra that neither stores, nor operates on, full matrices; this leads to more efficient memory use for problems that involve large matrices. Since the dimensions of largest matrices used in the problem at hand are 3×3, the `medium-scale` option is used.

The FMINUNC Matlab function, with the `medium-scale` option specified, uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton method [70–73] with a cubic line search procedure. The BFGS method is an improved version of the Davidon-Fletcher-Powell (DFP) method [74, 75], with the only difference being the formula used for updating the approximation of the Hessian matrix. By default, the gradient is computed using finite differences, with a user-supplied gradient being optional.

The FMINCON Matlab function offers several algorithms for solving the inequality-constrained nonlinear optimization problem given by the objective function (5.27), subject to the constraints listed in (5.25). The algorithm options are `interior-point`, `sqp`, `active-set`, and `trust-region-reflective`, the last of which cannot be used for problems with inequality constraints. For all algorithms except `trust-region-reflective`, a user-supplied gradient is optional. `Interior-point`, a large-scale algorithm, is the most general [76–78], though computational savings can be realized with `sqp` or `active-set` for small- to medium-scale problems. Therefore, the optimization problem at hand is solved using the FMINCON `interior-point` (IP) algorithm, and the solution is compared to that reached with the FMINUNC `medium-scale` (MS) algorithm.

For both methods, algorithm convergence occurs in Matlab when one of several stopping criteria are satisfied. For the problem at hand, the most appropriate criterion is convergence of the objective function. Convergence is deemed to occur in

Matlab if the size of the latest change in objective function value or the value of first-order normality condition (FONC) is less than the function tolerance ϵ. We therefore set $\epsilon = 10^{-8}$, while all other stopping criteria tolerances are set to 10^{-20}. For the unconstrained problem, the FONC is given by

$$\left\|\nabla\sigma_p^*\right\|_\infty < \epsilon \tag{5.36}$$

where $\|\cdot\|_\infty$ is the infinity norm. For the constrained problem, we have

$$\left\|\nabla\sigma_p^* + \boldsymbol{\mu}^T\mathbf{g}(\mathbf{k})\right\| < \epsilon, \tag{5.37}$$

where $\mathbf{g}(\mathbf{k}) \leq \mathbf{0}_4$ is the vector equation of inequality constraints and $\boldsymbol{\mu}$ is a vector of Lagrangian multipliers. For the problem at hand, the inequality constraints are the four planes of the gain stable volume, given by Eqs.(5.25), so we have

$$\mathbf{g}(\mathbf{k}) = \begin{bmatrix} 0.5 & 0.25 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} K_p \\ K_i^* \\ K_d^* \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.38}$$

Equation (5.37) is an implementation of the Karush-Kuhn-Tucker conditions [79, 80], with no equality constraints present.

For each method used, the following items are listed in Table 5–2: the initial guess $\mathbf{k}_0$, the FONC upon convergence, given by Eq.(5.36) or Eq.(5.37), the computational time[2] t_{comp}, the number of algorithm iterations to convergence, and the type of gradient computation. For every case, the optimal value of the design vector found is $\mathbf{k}_{\text{opt}} = \begin{bmatrix} 1 & 0.33 & 0 \end{bmatrix}^T$, with the value of the objective function being $\sigma_p^* = 0.061$.

[2]The computer used for computations had the following specifications: Intel Core i7-2720QM 4 CPUs @2.2 GHz (turbo 3.3 GHz), 16 GB RAM, 240 GB solid state drive

Table 5–2: Optimization of the vector of gains $\mathbf{k}$

Alg.	$\mathbf{k}_0$	t_{comp} (sec)	Alg. iter.	User grad.	FONC
MS	$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$	0.24	14	no	4.5×10^{-9}
	$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$	0.22	14	yes	4.9×10^{-9}
	$\begin{bmatrix} 0 & 0 & 0.5 \end{bmatrix}^T$	0.33	45	no	2.5×10^{-9}
	$\begin{bmatrix} 0 & 0 & 0.5 \end{bmatrix}^T$	0.25	45	yes	2.5×10^{-9}
IP	$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$	0.47	25	no	4.8×10^{-9}
	$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$	0.43	25	yes	4.8×10^{-9}
	$\begin{bmatrix} 0 & 0 & 0.5 \end{bmatrix}^T$	0.50	31	no	1.9×10^{-9}
	$\begin{bmatrix} 0 & 0 & 0.5 \end{bmatrix}^T$	0.45	35	yes	6.3×10^{-9}

From Table 5–2, we can see that MS is about twice as fast IP in all cases, and supplying an analytical expression for the gradient always improves the speed slightly. The `large-scale` algorithm of FMINUNC, and the `sqp` and `active-set` algorithms of FMINCON, were also tried, and the same solution was obtained. However, MS and IP exhibited the lowest computational times for FMINUNC and FMINCON, respectively.

It is important to note that the gains $\mathbf{k}_{\text{opt}}$ are optimal only for the specific problem solved above. If the disturbance model changes, K_i^* will change, but K_p will not; K_i^* increases with the magnitude of the step disturbance. For example, with $d_s = 0.5$, $K_i^* = 0.618$. As mentioned earlier, however, the magnitudes of the pulse and step disturbances are rarely above 0.5 and 0.2, respectively, so we need only verify that the controller does not deviate far from optimality within these bounds. Additionally, K_d^* can be used to balance the controller performance in the entire expected disturbance range. Normally, K_d^* is used to reduce overshoot caused by the integral component, though, for rapid prototyping, minimizing σ_p^* is much more important than reducing overshoot.

A second optimization problem is therefore formulated,

$$\min_{K_d^*} \left(\sigma_p^*\right), \qquad \sigma_p^* = \left(\frac{1}{n_{\text{layers}}} \sum_{k=1}^{n_{\text{layers}}} e_k^2\right)^{1/2}. \tag{5.39}$$

In this case, $\nabla = \partial/\partial K_d^*$, so the gradient is identical to Eq.(5.34), except

$$\begin{aligned} \nabla e_k = &-e_{k-1} + \left(2 - K_p - K_i^* - K_d^*\right) \nabla e_{k-1} + 2e_{k-2} \\ &+ \left(K_p + 2K_d^* - 1\right) \nabla e_{k-2} + e_{k-3} + K_d^* \nabla e_{k-3}. \end{aligned} \tag{5.40}$$

The proportional and integral gains are set to their optimal values, $K_p = 1$ and $K_i^* = 0.33$. The pulse and step magnitudes, d_p and d_s, are varied up to 0.5 and 0.2, respectively, to see the impact on the optimal values of K_d^* and the value of the objective function. Variation of d_s was found to have no impact on K_d^*, so the results obtained when varying d_p are shown in Table 5–3. For both algorithms, an analytical expression is used for the gradient.

Table 5–3: Optimization of the normalized derivative gain K_d^*

Method	$K_{d,0}^*$	$K_{d,\text{opt}}^*$	σ_p^*	d_p	Comp. time (sec)	Alg. iter.	FONC
MS	0	0	0.061	0.5	0.07	0	2.7×10^{-9}
		0.023	0.038	0.25	0.09	4	2.6×10^{-11}
		0.126	0.027	0	0.10	6	1.0×10^{-10}
		0	0.061	-0.5	0.09	2	8.7×10^{-12}
	0.2	0	0.061	0.5	0.10	7	1.8×10^{-11}
		0.023	0.038	0.25	0.10	7	1.4×10^{-10}
		0.126	0.027	0	0.10	5	8.1×10^{-9}
		0	0.061	-0.5	0.10	7	1.7×10^{-11}
IP	0	0	0.061	0.5	0.11	0	4.8×10^{-9}
		0.023	0.038	0.25	0.22	12	4.0×10^{-9}
		0.126	0.027	0	0.20	9	4.2×10^{-9}
		0	0.061	-0.5	0.11	0	5.1×10^{-9}
	0.2	0	0.061	0.5	0.23	17	4.6×10^{-9}
		0.023	0.038	0.25	0.22	13	4.0×10^{-9}
		0.126	0.027	0	0.21	9	2.0×10^{-9}
		0	0.061	-0.5	0.23	17	4.6×10^{-9}

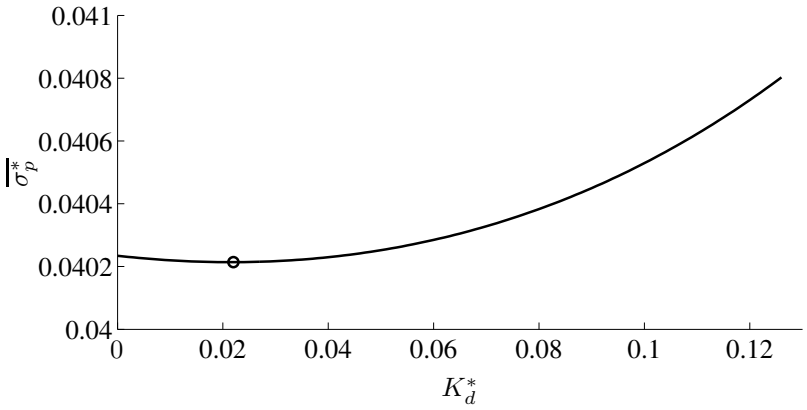


Figure 5–13: Optimization of the derivative gain K_d^*

The MS and IP algorithms converge to the same values for $K_{d,\text{opt}}^*$, when the pulse disturbance amplitude d_p is the same. Algorithm performance is similar to that for the first optimization problem. The results in Table 5–3 indicate that the optimal value of K_d^* lies in the range $[0, 0.126]$. We can define the best value for K_d^* as that which yields the smallest average value of σ_p^* for all values of d_p in the range $[0, 0.5]$. The relation between $\overline{\sigma_p^*}$ and K_d^* is plotted in Fig. 5–13, the optimal value of K_d^* being about 0.02. Also, any value of K_d^* in the range $[0, 0.06]$ could be used, since $\overline{\sigma_p^*}$ is almost constant there.

5.7 SMF PID Controller Simulations

A series of simulation runs are performed, shown in Fig. 5–14, to characterize the system performance with the optimal gains $K_p = 1$, $K_i^* = 0.33$, and $K_d^* = 0.03$. Parameters for each simulation are listed in Table 5–4. Valve frequency ν is calculated using Eq.(5.14), with $c_{\text{ff}} = 4$ Hz/(mm^3/s), $w = 1.5$ mm, $\Delta h_d = 0.25$ mm and $v = 25$ mm/s. These are values typically encountered using the Cobra 600 RFP system.

Figure 5–14 shows that, with the optimal gains, the SMF PID controller is stable and rapidly corrects any disturbances likely to be encountered. For further verification, the set of unstable gains $\mathbf{k} = \begin{bmatrix} 1 & 0.33 & 0.45 \end{bmatrix}^T$ is used and the controller performance is shown in Fig. 5–15. This set of gains fails the Jury stability criterion, with one of the poles of Eq.(5.22) having a module of 1.05.

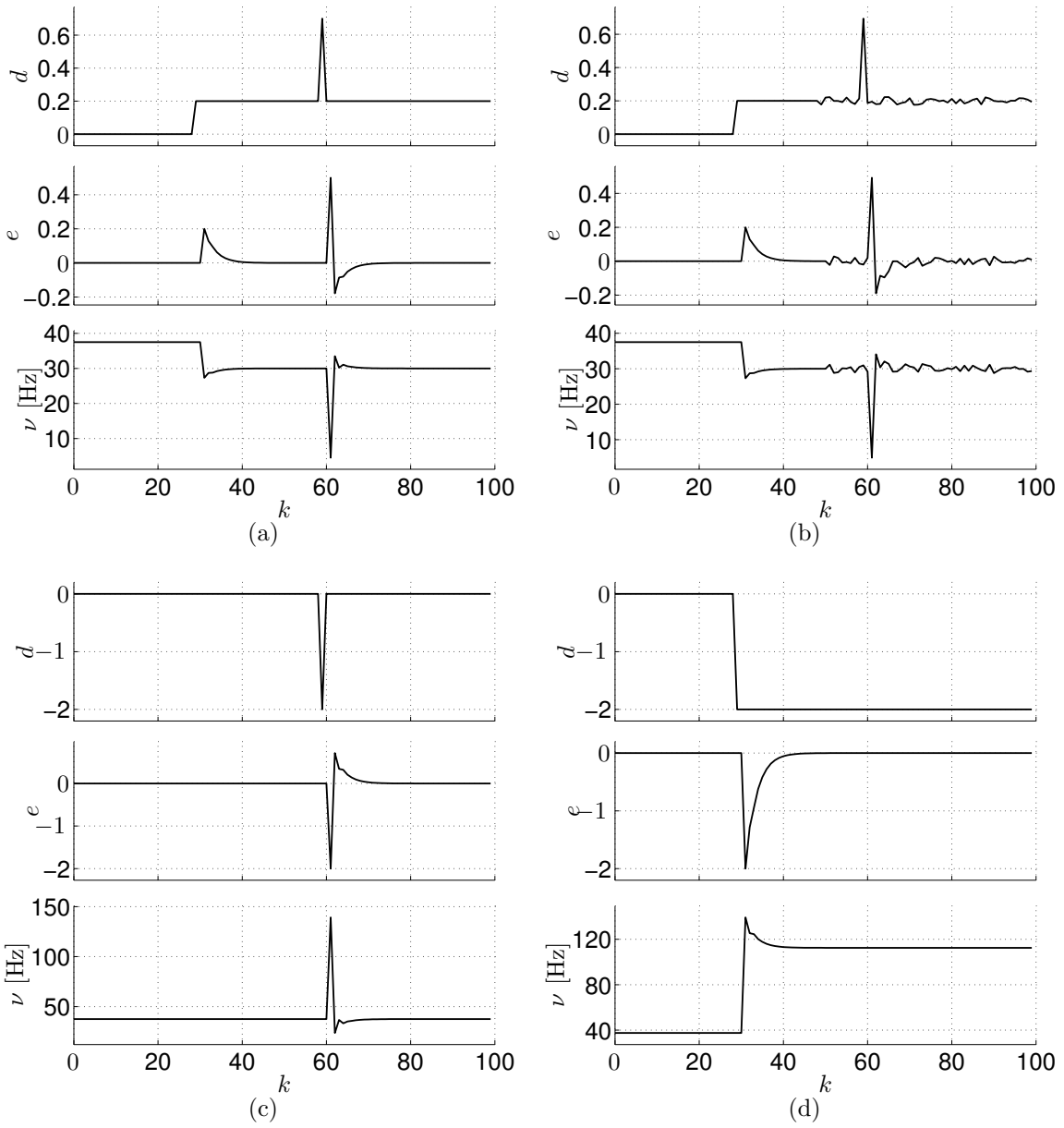


Figure 5–14: Matlab simulations of the PID controller with optimal gains: disturbance values d_k for each subfigure, are listed in Table 5–4

5.8 Summary

Surface mapping feedback (SMF), a novel method of geometric feedback for rapid prototyping systems, was introduced in this chapter. The technique is applicable to

Table 5–4: Contributions to the disturbance d_k for each subfigure of Fig. 5–14

Subfig. of Fig. 5–14	d_p	k_p	d_s	k_s	d_r	k_r
(a)	0.5	60	0.2	30	0	-
(b)	0.5	60	0.2	30	0.05	50
(c)	-2	60	0	-	0	-
(d)	0	-	-2	30	0	-

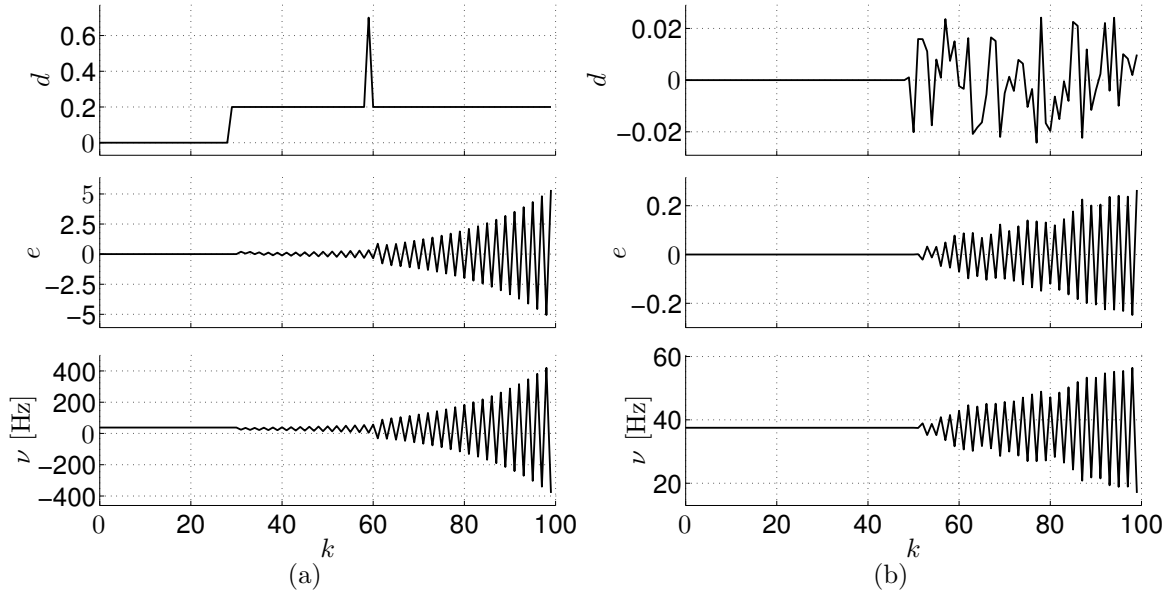


Figure 5–15: Matlab simulations of the PID controller with unstable gains $K_p = 1$, $K_i^* = 0.33$, and $K_d^* = 0.45$; disturbance values d_k for each subfigure are listed in Table 5–5

both drop-on-demand and continuous-flow RP systems. A prerequisite for implementation of SMF is a process parameter that has a well-known impact on the part geometry. For continuous-flow systems, continuous adjustment of this parameter along deposition paths must also be possible. Also, a measurement system is needed for mapping part surface geometry. The implementation of SMF can lead to a significant increase in part accuracy, as has been shown for the Cobra 600 RFP system. Additionally, the system can help avert construction failures caused by an RP system drifting away from the nominal part height.

Table 5–5: Contributions to the disturbance d_k for each subfigure of Fig. 5–15

Subfig. of Fig. 5–14	d_p	k_p	d_s	k_s	d_r	k_r
(a)	0.5	60	0.2	30	0	-
(b)	0	-	0	-	0.05	50

A one-dimensional version of the SMF PID controller was produced to perform a stability analysis and gain optimization. With the stability analysis, a region in the three-dimensional space of the controller gains K_p, K_i^*, and K_d^* was identified, within which the system is stable. Optimal gains were then found for the Cobra 600 RFP system by minimizing the sum of the squared error when typically encountered disturbances are simulated.

CHAPTER 6
Material Flow Control

In Subsec. 1.2.4, various options for valve flow control are discussed, with the conclusion being that digital control via frequency modulation (FM), implemented offline, is the best option for a rapid prototyping system. Here, we first briefly discuss a previously used constant flow control technique used with the Cobra 600 rapid freeze prototyping (RFP) system, and then explain the advantages that variable flow via offline control carries. Then, the capabilities of the specific valve used for the Cobra 600 system are described, and the procedure used for synthesizing the variable-flow signal is discussed.

In a previous implementation of the Cobra 600 system, constant material flow was used along deposition paths. Compared to variable-flow control, constant-flow hardware and software are both much simpler to implement . However, a variable-flow system is desirable for two reasons. Firstly, variable flow is necessary if variable end-effector speed is to be used, which is the case for minimum-time trajectory shaping (MTTS), discussed in Ch. 4. Secondly, variable flow offers a natural means of correcting geometry errors during the implementation of surface mapping feedback (SMF).

As discussed in Ch. 5, variation of the end effector speed c, or the material flow rate Q, can be used to correct geometry errors with SMF. Variable speed control, however, has many more restrictions associated with it. Most importantly, at each path point, there are strict restrictions on the range of allowed speeds, since acceleration and jerk constraints limit the change in speed allowed from one point to the

next. Additionally, a *minimum* amount of material must be deposited at all path locations, since Q is held constant and c has an upper bound. Conversely, with Q as the process parameter, it is possible to completely turn off material deposition at any point along a path. Lastly, c should not be used as the process parameter if minimum-time trajectory shaping (MTTS), explained in Ch. 4, is being used. Since c is optimized using MTTS, if it were then used as the process parameter for SMF, the optimality would be lost.

6.1 Variable Flow Control

The valves used for primary flow control with the Cobra 600 RFP system are of the digital (on/off) variety, actuated by microsolenoids[1]. The only variable that can be used to both greatly impact the flow rate and vary it quickly and accurately enough for synchronization is the microsolenoid valve control signal.

Three aspects of the valve control signal are user-configurable: spike time t_{spike}, pulse time t_{pulse}, and period time t_{period}, all depicted in Fig. 2–8. The waveform shown in Fig. 2–8(b), is referred to as "spike-and-hold" by the manufacturer: a 24 V spike is required to open the valve, and a 3.5 V signal is sufficient to hold it open. This type of signal is used to minimize the power required and the heat produced by the valve. The valve frequency f is given by

$$f = 1/t_{\text{period}}. \tag{6.1}$$

Spike time is set using a screw on the spike and hold driver for the valve, so high speed variation of this parameter is not possible. Additionally, variation of t_{spike} does not greatly impact Q, and should not be raised too high, to avoid overheating.

[1]Manufactured by the Lee Company, with following specifications: Part name VHS-M/2-CRDV-CHROME CORE - 24V; Part number INKX0515950A

Therefore, variation of t_{pulse}, t_{period}, or both could be used to achieve the desired flow control. Pulse width modulation (PWM) is achieved through variation of t_{pulse}, while holding t_{period} constant. Frequency modulation (FM) is achieved through variation of t_{period}, while holding t_{pulse} constant.

For PWM, the maximum flow rate is achieved when $t_{\text{pulse}} = t_{\text{period}}$. The minimum pulse width is 500 μs, defined by the manufacturer. The frequency used should be high enough such that a continuous path is produced; practically this means that at least one drop should be deposited per millimeter of path; thus for the maximum speed of 125 mm/s defined in Ch. 4, the valve frequency should be at least 125 Hz. The maximum pulse time for this frequency is 8 ms.

For FM, the minimum pulse width is still 500 μs, and the maximum frequency is given by $\nu_{\max} = 1/t_{\text{pulse}}$. For example, the maximum frequency for a pulse width of 2 ms is 500 Hz. The maximum usable frequency of the microsolenoid valve is 1000 Hz.

When implementing variable flow for matching variable path speeds produced by MTTS, the objective is to maintain a constant material volume deposited per distance travelled. With FM, this objective naturally extends to maintaining a constant frequency per path speed ν/c. We can also think of ν as the valve *speed*, which is to be matched with EE speed c. With PWM, on the other hand, ν is constant, and flow control is achieved much more awkwardly through variation of the pulse width.

While it is evident that FM offers a more natural means to implement variable-flow control compared to PWM, we can quantify the most suitable type of flow control as that which maximizes the flow rate range available. Using PWM, the available flow rate range is limited by the pulse width range of $[0.5, 8]$ ms. The restriction of a minimum flow rate is problematic because it prevents the matching of low EE speeds at the start of a path with low flow rates. With FM, on the other hand, valve frequency, and by extension flow rate, can be reduced to zero. Additionally, a much larger flow rate range is possible, since allowed frequency range can be as large as $[0, 1000]$ Hz.

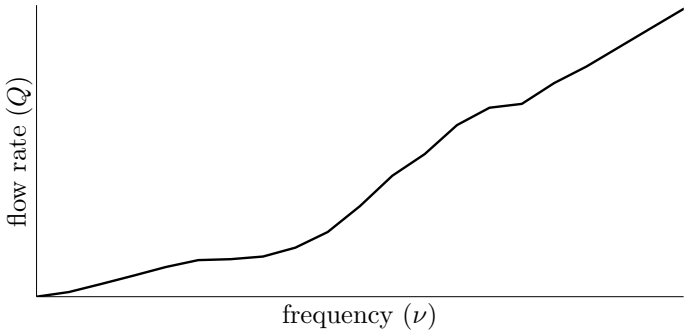


Figure 6–1: Typically observed frequency-flow rate relationship for the microsolenoid valves used with the Cobra 600 RFP system

Based on these considerations, FM was implemented for the Cobra 600 rapid freeze prototyping system. PWM remains a feasible option, however, particularly if it were used in some combination with FM.

When holding all other parameters constant, it is nominally expected that flow rate will be *proportional* to valve frequency. In practice, however, this is not the case; Fig. 6–1 shows the type of relationship typically observed with the microsolenoid valves that are used. The specific frequency-flow rate relation $\nu = h(Q)$ for a particular material and system configuration is produced offline. Approximately 20 values in the frequency range to be used are selected; for each value, the mass of liquid deposited during a certain amount of time is found. A frequency-flow rate table is thus produced, the function $h(Q)$ then amounting to linear interpolation in this table.

6.2 The Variable-Flow Valve Control Signal

For every deposition path, the variable-flow valve control (VFVC) signal is dependent on three factors: the frequency-flow rate relation $\nu = h(Q)$; the [variable] path speed c, produced by MTTS, described in Ch. 4; and the control action surface A, produced by surface mapping feedback (SMF), described in Ch. 5. For each deposition path, we define the n-dimensional vector arrays $\boldsymbol{\nu}$, $\mathbf{c}$, $\mathbf{u}$,and $\mathbf{q}$, with n denoting the number of points on the path; these arrays list the value of parameters ν, c, the controller output u, and Q at each path instant t_i.

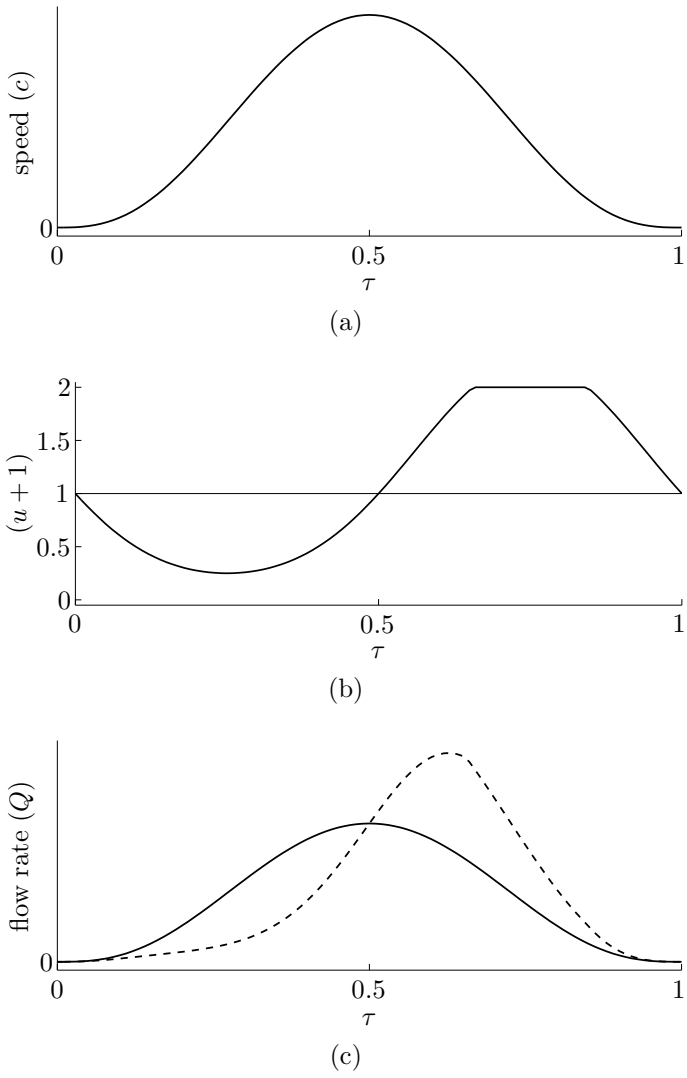


Figure 6–2: Computation of the valve flow rate for a straight-line test path, based on the path speed and the control action: (a) path speed; (b) controller output along the path; (c) required flow rate, before and after applying the controller output

The variable-flow valve control signal is sent to the valve microcontroller, the Philips LPC-H2148, as a list whose entries indicate the elapsed time along the path at which every pulse is to be output. Custom firmware written for the LPC-H2148, along with input and output circuitry, has been configured to produce the required digital output signal, based on the list of pulse instants. This is explained in more detail in Subsec. 2.3.1; the LPC-H2148 configuration is shown pictorially in Fig. 2–7 and schematically in Fig. 2–9.

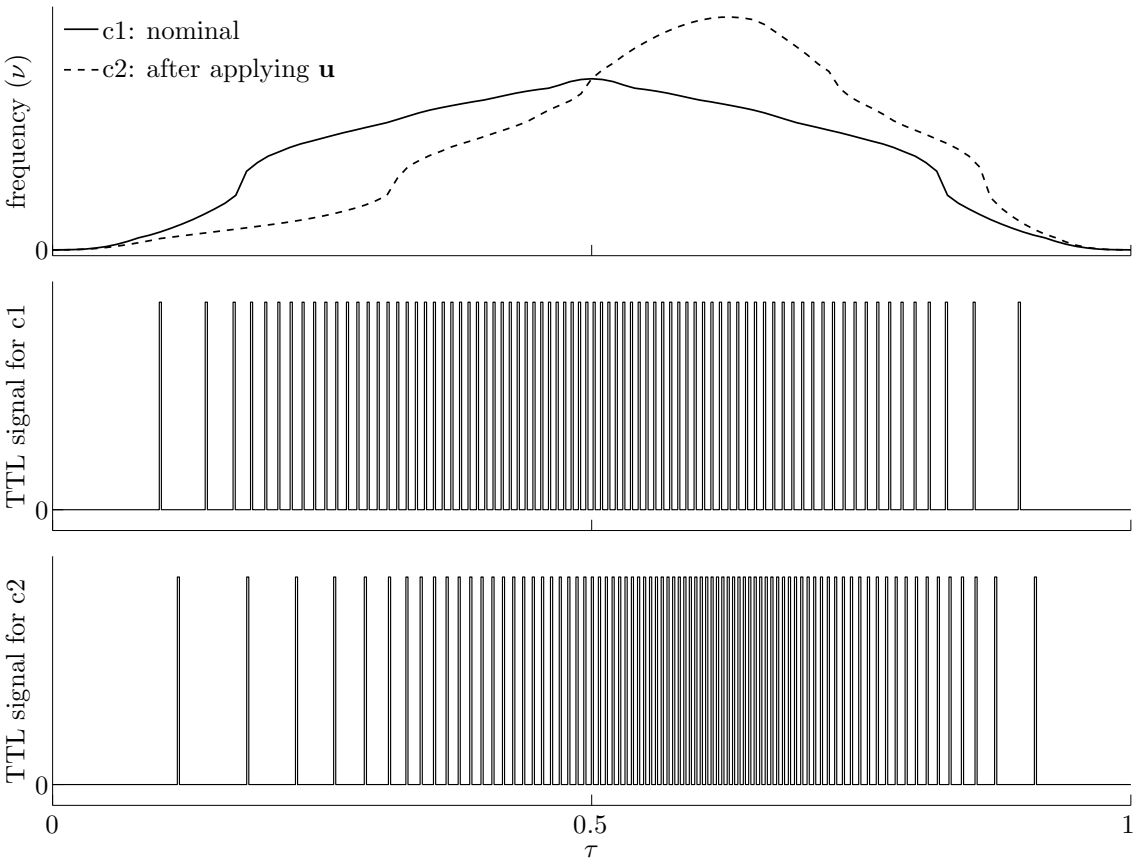


Figure 6–3: Formation of the valve pulse signal for a straight-line test path

The procedure used to produce the valve transistor-transistor logic (TTL) signal for a straight-line path is depicted graphically in Figs. 6–2 and 6–3. Normalized time is used, such that $\tau = t/T$, where T is the total time of travel for a path. Nominally, the flow rate along the path is given by

$$\mathbf{q}_{\text{nom}} = w\Delta h_d \mathbf{c} \tag{6.2}$$

where w is the deposition path width and Δh_d is the deposition layer thickness. Applying the control action, the corrected flow rate then becomes

$$q_{i,\text{cor}} = A(x_i, y_i) = (u_i + 1)\, q_{i,\text{nom}}, \qquad i = 1 \ldots n \tag{6.3}$$

The nominal and corrected flow rates are shown in Fig. 6–2. The nominal required frequency along the path, shown in Fig. 6–2(b), is given by

$$\nu_i = h(Q_{i,\text{cor}}), \qquad i = 1 \ldots n \tag{6.4}$$

The required number of pulses at each path point i is given by

$$n_{\text{pulses},i} = \delta \sum_{j=1}^{i} \nu_j, \qquad i = 1 \ldots n \tag{6.5}$$

where δ is the point-to-point duration along the path. The entries of $\mathbf{n}_{\text{pulses}}$ in Eq.(6.5) are decimal numbers, evenly-sampled in *time*. The desired output data format, which is a list of instants at which each pulse is to be output, is produced by linear interpolation of the evenly-sampled time array $\delta(1, \ldots, n)$, at the following evenly-spaced pulse values

$$n_{\text{pulses}} = 1, 2, \ldots, \lceil \max(n_{\text{pulses}}) - 0.5 \rceil - 0.5. \tag{6.6}$$

where $\lceil \cdot \rceil$ is the ceiling function, which maps a real number to the smallest following integer.

6.3 Computational Cost

Typically, a deposition layer will contain hundreds of paths and a part will contain over 10 000 paths. After each measurement layer, VFVC is used to update all of the valve control data for subsequent deposition layers until the next measurement layer. Normally, these computations require less than five seconds. The computational cost of the technique is low because only simple operations like vector multiplication and linear interpolation are used. Compared to RPSLICE, described in Ch. 3, and MTTS, described in Ch. 4, the cost of VFVC is almost negligible. Of course, RPSLICE and MTTS are also implemented prior to part construction, while VFVC is implemented during part construction.

6.4 Summary

After considering several options for implementing flow variation, variable-flow valve control (VFVC) via frequency modulation (FM) was selected as the best control strategy. VFVC has several advantages compared to geometric error correction with the c-controller. Firstly, VFVC does not affect trajectory control that has been *separately* optimized with MTTS. Secondly, VFVC *simultaneously* synchronizes material flow rate with end-effector speed and corrects geometry errors with SMF. Compared to alternative variable-flow techniques, VFVC with FM produces a very large usable flow rate range.

One disadvantage of VFVC is increased complexity when compared to previously-used constant-flow techniques, since additional hardware and more advanced programming are needed. Another disadvantage is the variability of the frequency-flow rate relation $h(Q)$, which is nonlinear and also varies with valve usage. For example, with a high rate of use, the microsolenoid valve temperature increases, and the flow rate for a particular frequency will increase by as much as 20%. This effect occurs *gradually*, however, and is readily corrected by SMF.

CHAPTER 7
Conclusions

In this thesis, a novel rapid prototyping system for ice is introduced. The system hardware layout is described in Ch. 2, as an interconnection of many subsystems. Among these, the Cobra 600 robot, the LK-G32 laser displacement sensor, and the LPC-H2148 microcontroller, should be singled out. Without these devices, many of the innovative control algorithms described in Chs. 4–6 would not be possible.

Firstly, the Cobra 600 robot, used for positioning, is atypical for a rapid prototyping system: normally a Cartesian positioning system is used rather than a revolute manipulator. The four-axis architecture of the Cobra 600 is suitable for rapid prototyping, however, since it provides three spatial degrees of freedom, as well as orientation in the horizontal plane. The closed-loop positioning control available with the Cobra 600 is also ideal for implementing the variable-speed control described in Ch. 4.

The Cobra 600 is also well-suited for RFP because it can be installed *outside* the freezer, with the distal link extending into the freezer. This configuration is advantageous because the robot does not need to tolerate the harsh freezing environment. Additionally, the robot does not occupy any portion of the space inside the freezer, which allows for the maximization of the deposition workspace. Neither of these advantages would be realized with a typical Cartesian positioning system, which would need to be installed inside the freezer.

Secondly, the laser displacement measurement system, describeded in Ch. 5, is a critical component of surface mapping feedback. The most important feature of

the chosen laser is its ability to reliably measure ice surfaces; many similar laser displacement systems produce measurement errors due to the reflective properties of ice. Thirdly, the chosen valve microcontroller is critical for the variable flow control discussed in Ch. 6 because of its versatile programming features and the necessary memory and communication options it includes.

Based on the successful conversion of the Cobra 600 to an RP system, it is expected that a five- or six-axis manipulator could also be successfully modified for rapid prototyping. Compared to a three- or four-axis system, such a manipulator provides the potential for two- or three-axis control of nozzle orientation, which means deposition need no longer be limited to the vertical direction. This additional control could be useful for building sloping walls, avoiding the unwanted "staircase" effect inherent in most RP systems. Additionally, orientation control could be useful for adding detailed surface features to constructed parts.

Chapter 3 introduces RPSLICE, a slicing code for producing the deposition paths needed to build a CAD part with a 3D printer. While many CAD-slicing algorithms exist in the literature, several of the features of the code described in this thesis are novel. Firstly, the code is able to process extremely large CAD files, composed of tens of millions of facets. Secondly, scaffolding paths are produced automatically; a scaffolding CAD file is not needed. Thirdly, fill paths are produced using a buffering algorithm that is much more efficient than the Matlab algorithm on which it is based. This is significant, since path buffering is the most computationally-intensive step of the slicing algorithm. In addition to rapid prototyping, subfuntions of RPSLICE are applicable to mapping, manufacturing, and computer graphics.

As justified in Ch. 3, the versatility provided by a customized slicing algorithm such as RPSLICE was well worth the extra cost in development time. During the development of the RFP system, several specific features were implemented that are

unavailable in algorithms designed for other RP systems. Additionally, with a customized slicing algorithm, the system is more adaptable to future modifications.

In Ch. 4, a technique for implementing variable-speed positioning control along deposition paths, minimum-time trajectory shaping (MTTS), is described. MTTS is relevant for any application where a device is to follow a prescribed path in a minimum time, especially in cases where system constraints are only available kinematically. One of the most important features of MTTS is the ability to correctly synthesize trajectories for *any* path geometry. For the Cobra 600 RFP system, MTTS has reduced part-construction time by a factor of up to five for a typical part, when compared to the previously-used, constant-speed trajectory control. MTTS represents a novel, robust solution to the time-optimal trajectory control problem, with applications in rapid prototyping and robotics.

A significant feature of MTTS is the implementation of trajectory control using *variable* point spacing, with constant point-to-point duration. This technique was well-suited to trajectory control with the Cobra 600 RFP system, using the Adept V+ DURATION command, but could also be adapted to other positioning systems. For example, many three-axis CNC machines use positioning data coupled with feedrates for control, and MTTS output data can be readily converted to this format [35, 81]. Additionally, MTTS produces output data that are automatically optimized for storage and resolution: point resolution increases with path curvature, providing extra detail where it is needed.

Chapter 5 introduces surface mapping feedback (SMF), a geometric feedback algorithm for rapid prototyping. This algorithm is novel and could be applied to geometrically "close the loop" in many deposition-based RP systems, thereby improving system accuracy and reliability. Additionally, the algorithm is unique because of the three-dimensional spatial PID control loop implemented, compared to a traditional, one-dimensional time control loop.

SMF is a critical component the Cobra 600 RFP system. Before its implementation, part geometry fidelity was accomplished indirectly by regulating process parameters including temperature, reservoir pressure, and the control signal of the microsolenoid valve. However, control of of these variables was insufficiently precise for producing ice objects with the desired accuracy. Often, several parameter adjustments were needed and parts had to be constructed several times before an acceptable accuracy was achieved. Additionally, for many parts, manual intervention was often necessary to prevent the end effector from scraping areas of the part where the height was increasing too rapidly. Also, it was observed that the ideal control parameters varied from one part to the next; in fact, the parameters could even vary during the construction of a single part. This variation was mostly caused by the valve flow rate being dependent on *usage* in addition to the control signal. For the same control signal, the flow rate tends to be higher when the valve usage is high, i.e. when a large amount of material must be deposited for the layer. This effect is likely caused by the valve temperature increasing with increased usage. SMF responds to all of these undesirable effects; in fact, process parameters can now be set within approximately 20% of their nominal values and SMF will adjust the deposition as needed to produce the desired geometry.

Although process parameter control is sufficiently precise that geometric feedback is not needed for many other RP systems, SMF could definitely enhance part geometry accuracy in some cases. Additionally, SMF could be used to investigate novel materials for RP, which may have previously been rejected for consideration because they are too uncontrollable.

A stability analysis of the SMF PID controller was also conducted, to establish a region in the three-dimensional space of the controller gains K_p, K_i^*, and K_d^*, within which the system is stable. The controller gains were then optimized for the Cobra

600 RFP system, using several different methods for solving a nonlinear minimization problem.

The stability analysis and gain optimization were both performed using a 1D version of the SMF PID controller. In the rapid prototyping context, such a 1D controller would regulate the height of a single pixel or point. In practice, a grid of pixels, representing a surface interpolant, is to be controlled. The stability analysis and gain optimization would be completely applicable to this grid, if each pixel behaved *independently*. However, since the pixel grid is being used to represent a continuous surface, each pixel is coupled with its neighbours and its behaviour is dependent on theirs.

The interdependence between adjacent pixels could be modeled through the introduction of constraints on the surface continuity, though this would greatly complicate the analysis. Instead, practical insight can be used to argue the applicability of the 1D analysis to the more general 3D system. Pixel interdependence actually *increases* the system stability, since it has a damping effect: as geometric error increases at a particular pixel location, it becomes increasingly *distributed* among the pixels nearby. Error concentration is thus avoided, and the desired accuracy threshold is less likely to be exceeded. This effect also makes the system more tolerant of errors in reference-frame offsets and positioning/deposition and positioning/measurement synchronization.

The enhanced stability due to pixel interdependence can be observed experimentally during part construction with the Cobra 600 RFP system: interior pixels, which are completely surrounded by other pixels, exhibit much less height variation compared to boundary pixels. This characteristic is exploited by completely surrounding all ice features with scaffolding, using the slicing code, as explained in Sec. 3.3. Therefore, boundary pixels, and the associated increase in error, can only occur for the scaffolding.

Gain optimization with the 1D controller amounts to minimizing the RMS error for a single pixel on the surface. When generalizing to the 3D controller, SMF addresses error by adjusting deposition *separately* for each pixel and does not perform a global adjustment based on the RMS error for the entire measured surface. For example, if an undulating surface is measured, SMF will modify control data such that more material is deposited in low regions and less material is deposited in high regions.

In Ch. 6, variable-flow valve control (VFVC) is introduced. VFVC is implemented via frequency modulation (FM) of the TTL control signal used to open and close the microsolenoid valve. VFVC serves two functions: nominally, flow variation is used to synchronize flow with the variable path speed implemented with MTTS; the nominal flow signal is then adjusted to correct geometry errors detected with SMF. This flow variation is not possible with standard dispensing flow control techniques, which typically maintain flow proportional to end-effector speed. FM is a unique method for implementing variable-flow control that produces a significantly larger usable flow-rate range, when compared to pulse-width-modulation (PWM) flow control, which is also commonly-used in industrial applications.

To permit maximum flexibility of the valve control waveform, it is generated offline. A major challenge for the implementation of VFVC was thus the *synchronization* between positioning and deposition. As described in Subsec. 2.3.1, digital timing signals from the Cobra 600 are used to synchronize valve and positioning control timers at the start of a path. During the path, positioning and valve control are implemented *independently*; timing control of each system is sufficiently precise that synchronization is maintained. Similar timing and synchronization is accomplished with the Keyence LK-G32 laser during measurement paths.

The software and control implementations of Chs. 3–6 represent the main contributions to knowledge of this thesis. Individually, they represent contributions to a

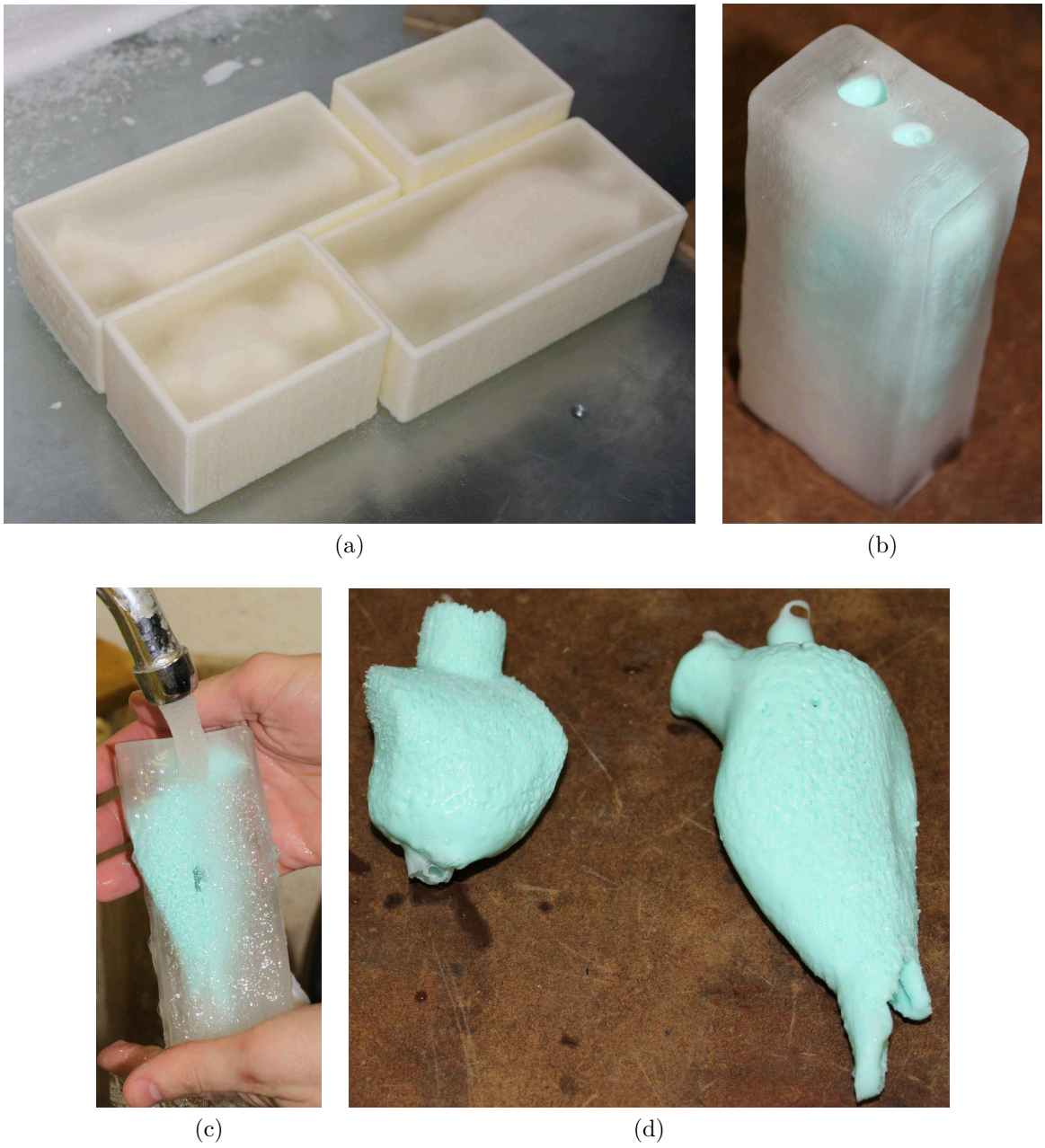
wide variety of research fields. When implemented together in the Cobra 600 RFP system, a reliable, fast, accurate, automatic 3D ice printer is produced, capable of forming any 3D structure that can be modelled in CAD software.

A common theme in the development of Cobra 600 RFP system software, including slicing, MTTS, SMF, and VFVC, was the development of open-source algorithms. The integration of closed-source software was deemed to be undesirable, because it would have placed unwanted restrictions on the future system development. This choice has proved to be highly beneficial, since the system has continuously evolved, as have the software requirements.

The main application of RFP is the production of ice sculptures for the ice tourism industry, including ice hotels and ice festivals. However, the Cobra 600 RFP system has also proved to be competitive with other RP systems, in certain cases. Obviously, the main advantage of RFP compared to other RP systems is its minimal cost, while the main disadvantage is the limited durability of the ice models. In many cases, however, a prototype is only needed for *visualization*. Additionally, RFP can be used to create moulds for casting other materials. In fact, this procedure has been used to produce a silicone version of the atrium and ventricle of a calf heart, as shown in Fig. 7–1. The silicone models were used for investigating a percutaneous annuloplasty procedure for mitral valve repair [82].

7.1 Recommendations for Future Work

Several aspects of the Cobra 600 rapid freeze prototyping system (RFP) can be improved, including the system user-friendliness, the scaffolding material, and most importantly, the end-effector design. The choice to use open-source software also makes this system an ideal testbed for performing other, related research. For example, one of the planned future projects is to develop a large-scale rapid prototyping system; the software algorithms introduced here are adaptable to such a system. In



(a) (b)

(c) (d)

Figure 7–1: Silicone models of the atrium and ventricle of a calf heart, produced using ice moulds printed using the Cobra 600 RFP system.

particular, SMF could be crucial, since it could allow the use of materials previously thought to be too uncontrollable for RP.

The Cobra 600 RFP system user-friendliness should be improved. Currently, the system is configured for development: basic and advanced features are both configured using text files and manually editing code; a casual user would probably find this

configuration overwhelming. To address this problem, a target user with a specified level of expertise should be identified and a suitable interface should be designed. For example, a simple graphical user interface could probably be designed for a user who simply wants to print parts at a certain scale and has no need of advanced configuration options.

It is possible that a better scaffolding material could be found. The desired properties, along with the selection procedure for the scaffolding material, are described in Sec. 2.4. Shortening methyl ester (SME) was selected as the best option among the candidates considered. The main deficiency of SME is its non-ideal phase change characteristics: it does not have an abrupt solid-liquid phase change, and is only completely liquid above 5°C. To cope with this behaviour, manual removal of most scaffolding, followed by chemical removal of scaffolding remnants, is required. Ideally, the scaffolding material should have an abrupt liquid-solid phase change temperature at approximately −5°C; then scaffolding could simply be melted away by raising the part temperature to between −5°C and 0°C. The difficulty lies in the identification of such a material that also possesses the other desired properties listed in Sec. 2.4.

If a new EE is constructed, a major revision of the overall design described in Ch. 2 is recommended. Firstly, a single-piece EE should be used, i.e., one piece of aluminum that connects to the Cobra 600 user flange, with the valves and laser mounted *directly* to this piece, if possible. This modification would reduce the variations in component locations following disassembly and assembly of the EE.

Secondly, the number of valve/nozzle positions should be increased from two to four or more. Additional valves and nozzles would be useful for testing new configurations without affecting the main deposition system. Tests could include new scaffolding or part materials, and coloured water. Several nozzles could also be used to provide a larger usable flow-rate range. For example, a low flow rate nozzle could

be used to deposit boundary contours and achieve a high degree of detail, while a high flow-rate nozzle could be used for filling.

One modification to the EE that could be seen as a potential improvement is the extension of the Cobra 600 distal link by moving the valves and laser to be further offset from the joint-4 axis. This modification could be used to reduce resonance, to maximize a robot kinetostatic performance index for the robot [83, p. 201], or simply to enlarge the workspace. However, this modification would be dangerous for three reasons. Firstly, since joint 4 is only capable of a 720° rotation, rapid de-spin of the EE can occur if paths are not carefully programmed. Secondly, there is an increased danger of collision with the sides of the freezer. Thirdly, path synthesis would be more complicated and calibration of deposition and measurement reference frames would be more difficult. For these reasons, a long distal link is not recommended.

The characterization of system accuracy discussed in Sec. 5.3 demonstrates that the Cobra 600 RFP system can reproduce the part of Fig. 4–15 within the claimed accuracy of 0.5 mm. The measurement procedure used there should be applied to a wider range of parts, to demonstrate that this accuracy is not part-dependent. Ideally, complex objects, such as the helical gear and the Lucy statue, should be built and measured. However, more advanced metrology techniques such as 3D scanning would be needed to measure the constructed part geometry.

BIBLIOGRAPHY

[1] P. Sijpkes, E. Barnett, J. Angeles, and D. Pasini, "The architecture of phase change at McGill," in *Archit. Res. Cent. Consort. Spring Conf. (ARCC 2009)*, San Antonio, TX, Apr. 15–18, 2009, 6 pages.

[2] W. Zhang, M. C. Leu, Z. Yi, and Y. Yan, "Rapid freezing prototyping with water," *IEEE Spectr.*, vol. 20, pp. 139–145, 1999.

[3] F. D. Bryant, G. Sui, and M. C. Leu, "A study on the effects of process parameters in rapid freeze prototyping," *Rapid Prototyp. J.*, vol. 9, no. 1, pp. 19–23, 2003.

[4] F. D. Bryant and M. C. Leu, "Modeling and experimental results of concentration with support material in rapid freeze prototyping," *Rapid Prototyp. J.*, vol. 55, no. 5, p. 041020 (9 pages), 2009.

[5] ——, "Predictive modeling and experimental verification of temperature and concentration in rapid freeze prototyping with support material," *ASME J. Manuf. Sci. Eng.*, vol. 131, no. 4, pp. 317–324, 2007.

[6] Q. Liu, M. C. Leu, V. L. Richards, and S. M. Schmitt, "Dimensional accuracy and surface roughness of rapid freeze prototyping ice patterns and investment casting metal parts," *Int. J. Adv. Manuf. Technol*, vol. 24, pp. 485–495, 2004.

[7] Q. Liu and M. C. Leu, "Investigation of interface agent for investment casting with ice patterns," *J. Manuf. Sci.*, vol. 128, pp. 554–562, 2006.

[8] E. Barnett, J. Angeles, D. Pasini, and P. Sijpkes, "Robot-assisted rapid prototyping for ice structures," in *IEEE Int. Conf. Robot. Autom.*, Kobe, JP, May 12–17, 2009, pp. 146–151.

[9] E. Barnett, "Cobra 600 rapid freeze prototyping system user's guide," Tech. Rep., TR-CIM-11-10, Centre for Intelligent Machines and Department of Mechanical Engineering, McGill University, Montreal, QC, Oct. 2011.

[10] E. Barnett, J. Angeles, D. Pasini, and P. Sijpkes, "A heuristic algorithm for slicing in the rapid freeze prototyping of sculptured bodies," in J. Angeles, B. Boulet, J. J. Clark, J. Kövecses, and K. Siddiqi, Eds. *Brain, Body, and Machine*. Berlin: Springer-Verlag, 2010, pp. 149–162.

[11] ——, "Surface mapping feedback for robot-assisted rapid prototyping," in *IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 15–19, 2011, pp. 3739–3744.

[12] ——, "Minimum-time trajectory shaping for robot-assisted rapid prototyping," *IEEE Trans. Autom. Sci. Eng.*, 2012, (submitted).

[13] D. S. Ebertt, R. Yagelt, J. Scott, and Y. Kurzion, "Volume rendering methods for computational fluid dynamics visualization," in *Proc. Conf. Visualization*, Washington, DC, USA, Oct. 17–21, 1994, pp. 232–239.

[14] T. McLoughlin, R. S. Laramee1, R. Peikert, F. H. Post, and M. Chen, "Over two decades of integration-based, geometric flow visualization," *Comp. Graphics Forum*, vol. 29, no. 6, pp. 1807–1829, 2010.

[15] R. W. Cox, "AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages," *Comp. Biomed. Res.*, vol. 29, pp. 162–173, 1996.

[16] The MathWorks, Inc., "Matlab 3-D visualization, r2012a," 2012.

[17] S. Allen and D. Dutta, "Determination and evaluation of support structures in layered manufacturing," *J. Des. Manuf.*, vol. 5, no. 3, pp. 153–162, 1995.

[18] S. H. Choi and K. T. Kwok, "A tolerant slicing algorithm for layered manufacturing," *Rapid Prototyp. J.*, vol. 8, no. 3, pp. 161–179, 2002.

[19] P. Haipeng and Z. Tianrui, "Generation and optimization of slice profile data in rapid prototyping and manufacturing," *Rapid Prototyp. J.*, vol. 187–188, pp. 623–626, 2007.

[20] R. C. Luo, Y. L. Pan, C. J. Wang, and Z. H. Huang, "Path planning and control of functionally graded materials for rapid tooling," in *IEEE Int. Conf. Robot. Autom.*, Orlando, FL, May 15–19, 2006, pp. 883–888.

[21] A. Ossino, E. Barnett, J. Angeles, D. Pasini, and P. Sijpkes, "Path planning for robot-assisted rapid prototyping of ice structures," *Trans. Can. Soc. Mech. Eng.*, vol. 33, no. 4, pp. 689–700, 2009.

[22] G. Bolmsjö and G. Nikoleris, "Task planning for welding applications," in *IEEE Int. Conf. on Systems, Man and Cybernetics-Systems Engineering in the Service of Humans*, Le Touchet, France, 1993, pp. 515–519.

[23] E. J. Lima II and A. Q. Bracarense, "Trajectory generation in robotic shielded metal arc welding during execution time," *Ind. Robot: An Int. J.*, vol. 36, no. 1, pp. 19–26, 2009.

[24] M. de Graaf, R. Aarts, B. Jonker, and J. Meijer, "Real-time seam tracking for robotic laser welding using trajectory-based control," *Control Eng. Practice*, vol. 18, pp. 944–953, 2010.

[25] E. Bassi, F. Benzi, F. Calegari, and A. Erba, "Control strategies for very low speed trajectories of an industrial robot," in *IEEE Int. Symp. Ind. Electr.*, Ajaccio, France, May 4–7, 2004, pp. 687–692.

[26] A. P. Pashkevich, A. B. Dolguia, and K. I. Semkinb, "Kinematic aspects of a robot-positioner system in an arc welding application," *Control Eng. Practice*, vol. 11, no. 1, pp. 633–647, 2003.

[27] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, 1985.

[28] K. G. Shin and N. D. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Trans. Autom. Control*, vol. AC-31, no. 6, pp. 491–500, 1986.

[29] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *J. Robot. Autom.*, vol. RA-3, pp. 115–123, Apr. 1987.

[30] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, pp. 2318–2327, Oct. 2009.

[31] A. Obradović, J. Vuković, N. Mladenović, and Z. Mitrović, "Time optimal motions of mechanical system with a prescribed trajectory," *Meccanica*, vol. 46, pp. 803–816, 2011.

[32] Adept Technology Inc., "Instruction handbook: Adept Cobra 600 and Adept Cobra 800 Robot," 1999, 00560-00100, Rev. B.

[33] A. Gasparetto and V. Zanotto, "A technique for time-jerk optimal planning of robot trajectories," *Robot. Comp. Integ. Manuf.*, vol. 24, pp. 415–426, 2008.

[34] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 42–52, 2003.

[35] J. Dong, P. M. Ferreira, and J. A. Stori, "Feed-rate optimization with jerk constraints for generating minimum-time trajectories," *Int. J. Mach. Tools Manuf.*, vol. 47, pp. 1941–1955, 2007.

[36] Y. J. E. Wu and J. J. Beaman, "Contour following for scanning control in SFF application: Control trajectory planning," in *Proc. Solid Freeform Fabr. Symposium*, Austin, TX, 1990, pp. 126–134.

[37] G. M. Fadel and R. Ganti, "Parametric based controller for rapid prototyping applications," in *Proc. Solid Freeform Fabr. Symposium*, Austin, TX, 1998, pp. 236–243.

[38] C. C. Doumanidis, "In-process control in thermal rapid prototyping," *IEEE Contr. Syst. Mag.*, vol. 17, no. 4, pp. 46–54, 1997.

[39] J. Mazumder, A. Schifferer, and J. Choi, "Direct materials deposition: Designed macro and microstructre," *Mat. Res. Innovat.*, vol. 3, pp. 118–131, 1999.

[40] G. Muscato, G. Spampinato, and L. Cantelli, "Robot-assisted shape deposition manufacturing," in *IEEE Int. Conf. Emerging Tech. and Factory Autom.*, Hamburg, Germany, Sept. 15–18, 2008, pp. 1080–1083.

[41] D. L. Cohen and H. Lipson, "Geometric feedback control of discrete-deposition SFF systems," *Rapid Prototyp. J.*, vol. 16, no. 5, pp. 377–393, 2010.

[42] E. J. Routh, *A Treatise on the Stability of a Given State of Motion: Particularly Steady Motion.* Macmillan and Co., 1877.

[43] A.Hurwitz, "On the conditions under which an equation has only roots with negative real parts," in *Selected Papers on Mathematical Trends in Control Theory*, R. T. Ballman et al., Ed. Dover, 1964.

[44] E. I. Jury, *Theory and Application of the z-Transform Method.* New York: John Wiley & Sons, Inc., 1964.

[45] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Trans. ASME*, vol. 64, pp. 759–765, 1942.

[46] A. M. Lopez, J. A. Miller, C. L. Smith, and P. W. Murrill, "Tuning controllers with error-integral criteria," *Instrumentation Technology*, vol. 14, no. 11, pp. 57–62, 1967.

[47] N. Nguyen and S. T. Wereley, *Fundamentals and applications of microfluidics.* Boston: Artech House, 2006, vol. 2.

[48] J. L. Dobson and B. R. F. Kendall, "Controllable leaks using electrically pulsed valves," *J. Vac. Sci. Technol. A*, vol. 8, no. 3, pp. 2790–2794, 1990.

[49] T. Yi, M. Cornwell, and E. J. Gutmark, "Mean flow regulation of a high frequency combustion control valve based on pulse width modulation and system identification," in *American Control Conference*, Portland, OR, June 8–10, 2005, pp. 1132–1137.

[50] A. M. Noll, *Principles of modern communications technology.* Norwood, MA: Artech House, 2001.

[51] Adept Technology Inc., "Dispense module: User's & reference guide," 1998, RDASG-C0002, Rev. 3.2 A.

[52] E. Malone and H. Lipson, "Fab@home: The personal desktop fabricator kit," *Rapid Prototyping J.*, vol. 13, no. 4, pp. 245–255, 2007.

[53] K. Hartmann, R. Krishnan, R. Merz, G. Neplotnik, F. B. Prinz, L. Schultz, M. Terk, and L. Weiss, "Robot-assisted shape deposition manufacturing," in *IEEE Int. Conf. Robot. Autom.*, San Diego, CA, May 8–13, 1994, pp. 2890–2895.

[54] R. E. Tate, K. C. Watts, C. A. W. Allen, and K. I. Wilkie, "The viscosities of three biodiesel fuels at temperatures up to 300°C," *Fuel*, vol. 85, pp. 1010–1015, 2006.

[55] J. A. Kinast, "Production of biodiesels from multiple feedstocks and properties of biodiesels and biodiesel/diesel blends," Gas Technology Institute, Tech. Rep. NREL/SR-510-31460, 2003.

[56] F. Ma and M. A. Hanna, "Biodiesel production: a review," *Bioresource Tech.*, vol. 70, no. 5, pp. 1–15, 1999.

[57] K. Chalasani, L. Jones, and L. Roscoe, "Support generation for fused deposition modeling," in *Proc. Solid Freeform Fabr. Symposium*, Austin, TX, Aug. 7–9, 1995, pp. 229–241.

[58] X. Huang, C. Ye, S. Wu, K. Guo, and J. Mo, "Sloping wall structure support generation for fused deposition modeling," *Int. J. Adv. Manuf. Tech.*, vol. 42, pp. 1074–1081, 2008.

[59] H. Chen, N. Xi, W. Sheng, Y.Chen, A. Roche, and J. Dahl, "A general framework for automatic CAD-guided tool planning for surface manufacturing," in *IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, Sept. 14–19, Sep. 2003, pp. 3504–3509.

[60] R. C. Luo, C. L. Chang, J. H. Tzou, and Z. H. Huang, "Automated desktop manufacturing: Direct metallic rapid tooling system," in *IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 18–22, May 2005, pp. 584–589.

[61] A. Xu and L. L. Shaw, "Equal distance offset approach to representing and process planning for solid freeform fabrication of functionally graded materials," *Comput. Aided Des.*, vol. 37, pp. 1308–1318, 2005.

[62] E. Barnett, J. Angeles, D. Pasini, and P. Sijpkes, "Trajectory control for an innovative rapid freeze prototyping system," in *Proc. ASME 2010 Int. Des. Eng. Tech. Conf.*, Montreal, QC, Canada, Aug. 15–18, 2010, pp. 1289–1297.

[63] R. C. Luo and J. H. Tzou, "Implementation of a new adaptive slicing algorithm for the rapid prototyping manufacturing system," *IEEE/ASME Trans. Mechatronics*, vol. 9, pp. 593–600, Sep. 2004.

[64] W. Rattanawong, S. H. Masood, and P. Iovenitti, "A volumetric approach to part-build orientations in rapid prototyping," *J. Mat. Proc. Tech.*, vol. 119, pp. 348–353, 2001.

[65] P. S. Carpenter, R. H. Brown, J. A. Heinen, and S. C. Schneider, "On algorithms for velocity estimation using discrete position encoders," in *Proc. of the IEEE*

IECON 21st Int. Conf. on Industrial Electronics, Control, and Instrumentation, vol. 2, Orlando, FL, Nov. 1995, pp. 844–849.

[66] K. J. Åström and T. Hägglund, *Advanced PID Control*. Research Triangle Park, NJ: Instrumentation, Systems, and Automation Society, 2006.

[67] D. Gorinevsky, S. Boyd, and G. Stein, "Design of low-bandwidth spatially distributed feedback," *IEEE Trans. Autom. Control*, vol. 53, no. 2, pp. 257–272, 2008.

[68] D. Garcia, "Robust smoothing of gridded data in one and higher dimensions with missing values," *Comput. Stat. Data Anal.*, vol. 54, pp. 1167–1178, 2010.

[69] T. Kailath, *Linear Systems*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1980.

[70] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms," *J. Inst. Math. Applic.*, vol. 6, pp. 76–90, 1970.

[71] R. Fletcher, "A new approach to variable metric algorithms," *Comp. J.*, vol. 13, pp. 317–322, 1970.

[72] D. Goldfarb, "A family of variable metric updates derived by variational means," *Math. Comp.*, vol. 24, pp. 23–26, 1970.

[73] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comp.*, vol. 24, pp. 647–656, 1970.

[74] W. C. Davidon, "Variable metric method for minimization," A.E.C. Research and Development, Tech. Rep. ANL-5990, 1969.

[75] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Comp. J.*, vol. 6, pp. 163–168, 1963.

[76] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Opt.*, vol. 9, no. 4, pp. 877–900, 1999.

[77] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Math. Prog.*, vol. 89, no. 1, pp. 149–185, 2000.

[78] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Math. Prog.*, vol. 107, no. 3, pp. 391–408, 2006.

[79] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proc.* 2nd *Berkeley Symposium*. Berkeley: University of California Press, 1951, pp. 481–492.

[80] W. Karush, "Minima of functions of several variables with inequalities as side constraints," Master's thesis, Univ. of Chicago, 1939.

[81] K. Zhang, C. H. Yuan, X. S. Gao, and H. Li, "A greedy algorithm for feedrate planning of CNC machines along curved tool paths with confined jerk," *Robot. Comp. Int. Manuf.*, vol. 28, pp. 472–483, 2012.

[82] T. Azar, J. Angeles, and J. Kövecses, "The mathematical model of a procedure for percutaneous annuloplasty," in *Proc. ASME 2010 Int. Des. Eng. Tech. Conf.*, Montreal, QC, Canada, Aug. 15–18, 2010, pp. 1219–1224.

[83] J. Angeles, *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*, 3rd ed. New York: Springer-Verlag, 2008.