Control, simulation, and appearance modeling for real-time physics-based hand animation

by

Sheldon Andrews

B.Eng., Memorial University of Newfoundland (2004)
M.A.Sc., University of Ottawa (2007)

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

Doctor of Philosophy

School of Computer Science
McGill University, Montreal

December 2014

English abstract

Digital human characters are a mainstay of video games, film, and interactive computer graphics applications. However, animating hands remains a challenging aspect of human character animation: posing the hand involves coordinating many degrees of freedom, synthesizing a plausible grasp requires careful placement of contacts, and realistic rendering must account for intricate colour and texture variations. Traditional solutions to these problems require significant manual effort by skilled artists. It is therefore of great interest to computer animation researchers to develop fast and automatic methods for animating hands. This thesis presents methods for improving the realism of hands in real-time physics-based virtual environments.

We begin by presenting a framework for skilled motion synthesis, wherein reinforcement learning and non-linear continuous optimization are used to generate controllers for single-handed reorientation tasks. A mid-level multiphase approach breaks the problem into three parts, providing an appropriate control strategy for each phase and resulting in cyclic finger motions that accomplish the task. The exact trajectory is never specified, as the task goals are concerned with the final orientation and position of the object. Offline simulations are used to learn controller parameters, but the resulting control policy is suitable for real-time applications.

We then describe a method for the simulation of compliant articulated structures using an approximate model that focuses on plausible endpoint behaviour. The approach is suitable for simulating physics-based characters under static proportional derivative control and stiff kinematic structures, like robotic grippers. The computation time of the dynamical simulation is reduced by an order of magnitude, and faster than real-time frame rates are easily achieved. Additionally, the state of internal bodies is computed independently, and in a parallel fashion.

We also demonstrate an approach for synthesizing colour variation in fingers due to physical interaction with objects. A data-driven model relates contact information to visible colour changes for the fingernail and surrounding tissue on the back of the fingertip. The model construction uses the space of hemoglobin concentrations, as opposed to an RGB colour space, which permits transferability across different fingers and different people. Principal component analysis (PCA) on the

sample images results in a compact model, enabling efficient implementation as a fragment shader program.

Finally, we introduce a system for capturing grasping and dexterous interactions with real-world objects. A novel sensor ensemble collects information about joint motion and pressure distributions for the hand, and the data is used to design grasping controllers for a physics-based climbing simulation. Additionally, we speculate on how the interaction data can be used to derive future control strategies in physics-based animation. Combining interaction data with physical models is a promising approach for skilled motion synthesis involving hands.

French abstract

Les personnages humains numériques jouent un rôle primordial dans les jeux vidéo, les films et les logiciels d'animation interactifs. La main reste toutefois une partie du corps difficile à animer: positionner une main implique de coordonner plusieurs degrés de liberté, la synthèse d'un mouvement réaliste de préhension requiert un placement minutieux des points de contact, et un rendu photoréaliste comporte des difficultés au niveau des variations de couleur et de texture. Les solutions traditionnelles à ces problèmes demandent l'effort de plusieurs artistes hautement qualifiés. Cela motive les chercheurs en animation le développement d'outils permettant d'animer une main automatiquement. Cette thèse présente plusieurs méthodes qui augmentent le réalisme des mains dans des simulations interactives basées sur la physique.

Nous commençons par présenter un cadre technique pour la synthèse de mouvements adroits dans lequel un apprentissage par renforcement et une méthode d'optimisation non-linéaire sont utilisés pour générer des contrôleurs d'orientation lors de tâches comportant une seule main. Une méthode à phases multiples basée sur l'état de la simulation divise le problème en trois parties, fournissant une stratégie de contrôle appropriée pour chaque phase et produisant un mouvement cyclique des doigts qui accomplit la tâche souhaitée. La trajectoire exacte des doigts n'est jamais précisée, car les objectifs de la tâche dépendent de la position et de l'orientation de l'objet.

Nous exposons ensuite une méthode pour la simulation de structures articulées adaptatives en utilisant un modèle basé sur les comportements plausibles des extrémités. Cette méthode convient à la simulation de personnages physiques régis par un régulateur proportionnel dérivé statique et constitués de liens structurels raides, comme une pince robotique. Le temps de calcul pour une simulation dynamique est réduit par un ordre de grandeur, et une fréquence de rafraîchissement supérieure au niveau interactif est facilement observable. De plus, l'état des liens internes est calculé de façon indépendante et en parallèle.

Nous démontrons aussi une méthode pour la synthèse de variations de couleur au niveau des doigts provenant des interactions avec les objets. Un modèle basé sur des données permet de faire le lien entre les informations du contact et les changements de couleur visibles des ongles et des tissus adjacents. En s'inspirant du modèle RGB pour les couleurs, notre modèle utilise un espace

paramétrique de la concentration d'hémoglobine, ce qui permet une portabilité entre différents doigts et entre différentes personnes. Une analyse des composantes principales de photographies de doigts en action produit un modèle compact, ce qui rend possible une mise en application efficace de la méthode dans un nuanceur.

Finalement, nous introduisons un système permettant l'enregistrement de mouvements de préhensions et de manipulations adroites avec des objets réels. Un groupe de capteurs révolutionnaire récolte l'information provenant du mouvement des articulations et de la distribution de la pression. L'information récoltée est ensuite utilisée dans le développement de régulateurs de préhension lors d'une simulation basée sur la physique d'un personnage escaladant une paroi. De plus, nous spéculons sur la possibilité de tirer avantage de cette information dans le développement d'autres stratégies pour l'animation basée sur la physique. La combinaison de données d'interaction et d'un modèle physique est une méthode prometteuse pour la synthèse de mouvements adroits des mains.

Acknowledgments

First and foremost, I offer my sincerest gratitude to my supervisor, Dr. Paul G. Kry. You gave me the intellectual freedom I craved and motivated me every step of the way. You are a continuing source of guidance, knowledge, and inspiration. I could not ask for a better mentor.

To my parents, Paul and Doris, thank you for always supporting me in my determination to realize my full potential. Thank you to my in-laws, David and Helena, for welcoming me into your family and providing me with a home away from home.

A very special thank you to my loving wife, Marie-Anne. She strengthened my resolve on many occasions and supported me, practically and emotionally, throughout this whole endeavour. This thesis could not have been written without her.

Thank you to Catherine Bradley for lending her time and expertise to modify "the glove" used for our interaction capture experiments. Thank you to all of my labmates at the McGill Computer Animation Lab for the interesting discussions over the years. Likewise, thanks to Marek Teichmann for his advice and willingness to pose compelling research problems.

Lastly, I wish to thank the funding bodies and companies for their financial support during my degree, including NSERC, the Walter C. Sumner Foundation, GRAND, MITACS, CINQ, and CMLabs Simulations.

Contribution of authors

The material from Chapters 3, 4, and 5 have been published in peer-reviewed academic journals and conferences. My advisor Dr. Paul G. Kry helped in writing all of the manuscripts. Also, some of the work from Chapter 6 was presented as a poster. The relative contributions of co-authors are discussed below.

Chapter 3

Published as Andrews and Kry (2012), "Policies for goal directed multi-finger manipulation", *The 9th Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)* and an extended version as Andrews and Kry (2013), "Goal directed multi-finger manipulation: Control policies and analysis", *Computers & Graphics*, 37(7). I implemented the control framework, built the simulation assets, and conducted all experiments. I assisted in writing the manuscripts. I also presented the work in Darmstadt, Germany in 2012.

Chapter 4

Published as Andrews, Teichmann, and Kry (2014), "FORK^{-1}S: Interactive compliant mechanisms with parallel state computation", *ACM Symposium on Interactive 3D Graphics and Games (I3D)*. The method was developed in consultation with Marek Teichmann, CTO of CMLabs Simulations. He was the industrial contact for the MITACS Accelerate internship and contributed with some helpful discussions. I implemented the simulation framework, conducted experiments, and collected results. I also presented the work in San Francisco, USA in 2014.

Chapter 5

Published as Andrews, Jarvis, and Kry (2013), "Data-driven fingertip appearance for interactive hand simulation", *ACM SIGGRAPH Conference on Motion in Games (MIG)*. Marc Jarvis is an undergraduate student at McGill University's School of Computer Science whom I co-supervised for a summer research project. He collected pressure and image data for the fingertip appearance model described in Chapter 5. His expertise was also used to post-process images in Photoshop, create the skinned hand model shown in the results, and implement a preliminary version of the shader program. He also presented the work in Dublin, Ireland in 2013. I performed the model fitting steps and implemented the simulation software to demonstrate the results. I also wrote the final version of the shader program.

Chapter 6

Presented as Olsen, Andrews, and Kry (2014), "Computational climbing for physics-based characters", *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA) poster session*. Tim Gorm Kaas-Rasmussen Olsen is a Masters student from the University of Copenhagen. He visited the computer animation lab at McGill University where we collaborated on a project for dynamic climbing characters. He created the character model and some of the controllers used to generate the climbing sequence. He also helped with data collection. I implemented the optimization algorithm used to learn grasp poses, and conducted the simulation experiments demonstrated in the chapter.

Contents

List of figures

List of tables

List of algorithms

Acronyms

CFM constraint force mixing.

CMA-ES covariance matrix adaptation evolution strategy.

CPU central processing unit.

DAG directed acyclic graph.

DOF degrees of freedom.

FTR force transmission ratio.

GLSL OpenGL shading language.

GPGPU general-purpose computing on graphics processing units.

GPU graphics processing unit.

GWS grasp wrench space.

IK inverse kinematics.

k-NN k-nearest neighbours.

MPC model predictive control.

PC principal component.

PCA principal component analysis.

PD proportional derivative.

RBF radial basis function.

RL reinforcement learning.

SVD singular value decomposition.

Chapter 1

Introduction

The work presented in this thesis focuses on developing methods for physics-based character animation. Specifically, our target is the animation of human hands, and similar mechanical structures, for real-time applications. Simulation, control, and appearance modeling are pillars of physics-based character animation. Therefore, the objectives of our work are threefold: (1) to facilitate complex manual interactions between animated 3D characters and their environment, (2) to increase the overall richness and plausibility of grasping and dexterous manipulation animations by rendering cues that highlight these interactions, and (3) to develop methods that allow grasping and manipulation whilst maintaining the high frame rates demanded by real-time applications.

1.1 Motivation

Hands have long been identified as a distinguishing feature of humans; Aristotle once said that the hand is the "tool of tools".[1] They are instruments of the human mind: we use them to express ourselves, perform manual tasks, and realize complex interactions with the world around us. This compelling relationship is reflected by the significant body of literature devoted to reproducing the aesthetics (Hogarth, 1988), form (Yasumuro et al., 1999), and function of hands.

[1] Full quote: "It follows that the soul is analogous to the hand; for as the hand is a tool of tools, so the mind is the form of forms and sense the form of sensible things." (see Aristotle, 350 B.C.E.)

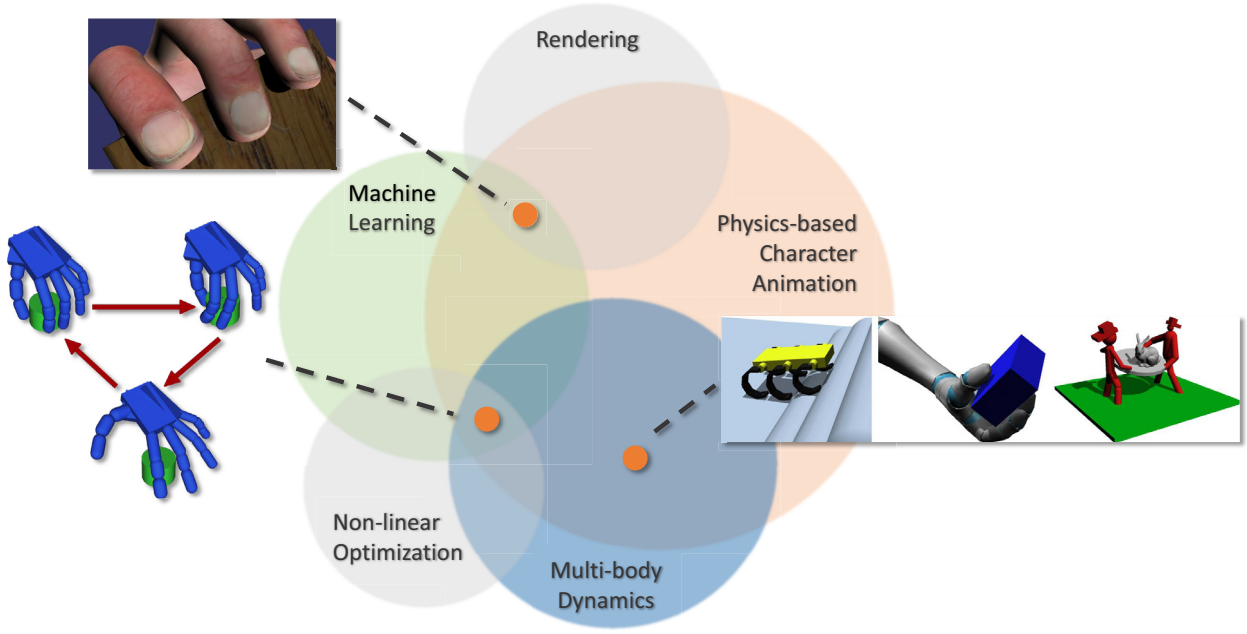

Figure 1-1: *We explore techniques from machine learning, non-linear optimization, robotics, multi-body dynamics, and rendering in order to address the overall problem of physics-based animation for hands.*

In computer animation, human characters are a mainstay, appearing in film and video games. However, the animation of hands is a difficult and time consuming task. Whether designed from scratch or recorded by a motion capture device, generating a motion that exhibits natural behaviour while satisfying task-specific or artistic constraints is challenging. Physics simulation may be used to address some difficulties, but depends on many variables. For instance, when designing a grasping animation, the shape, size, texture, and other physical properties of the object influence the success of the simulation. Also, human hands and similar articulated structures are typically modeled as kinematic chains of rigid bodies and joints. As a result, the simulation of physics-based grasping presents some challenging obstacles. Scenarios involving complex contact are computationally expensive and require special treatment by the dynamics solver. Notably, the constraints generated by hand-object contact can result in numerical issues, due to loops in the constraint topology and degenerate linear systems (Arechavaleta et al., 2009). Therefore, methods that address these concerns and reduce the overhead and complexity of grasping simulation are an active area of research.

In addition to presenting simulation difficulties, control and manipulation synthesis are also a chal-

lenge. Common approaches are similar in flavour to robotics, employing motion planning, to move within reach of a target object, and contact planning, to maintain a stable grasp while simultaneously applying wrenches to perform a manipulation task. This involves coordination of the large number of degrees of freedom (DOF) for the human hand model, making control algorithms susceptible to the *curse of dimensionality*. Also, since manipulation occurs by contact between the hand and the object, the planning landscape is wrought with non-linearities and discontinuities introduced by contact forces and friction.

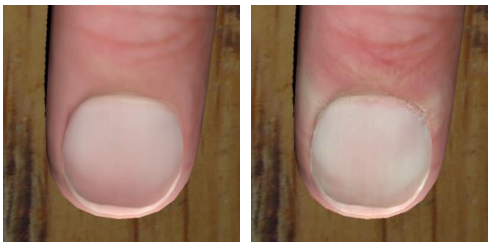

Figure 1-2: *A side-by-side comparison of rendering without and with appearance changes due to interaction forces.*

Even if dynamical simulation and dexterous motion synthesis presented no problems, standard skinning and rendering methods ignore the fine visual details associated with manual interaction. For example, consider the colour variations present in a hand when tightly gripping a cliff ledge. The whitening of the fingers and knuckles provides important cues about physical interactions with an object. Figure 1-2 shows simulations where these details are included as opposed to being overlooked. In a side-by-side comparison, it is clear that these visual specifics dramatically improve the richness of human hand animations. However, work in this area for interactive applications has mainly focused on colour adjustments to the face, and modeling surface deformations due to self contact.

Grasping and dexterous manipulation are contact-centric behaviours, and there is significant room for improving the performance and complexity achievable by current methods. In this regard, methods proposed by this thesis aim to push the state-of-the-art of physics-based animation toward likelife hands at real-time frame rates.

1.2 Contributions

The work presented in this thesis presents a solution for the real-time animation of interactive characters with hands. We aim to address many of the challenges discussed above by exploring methods from a number of different research areas. This diversity is due to the fact that our work is inspired by ideas from many fields, including robotics, machine learning, real-time rendering,

physics simulation, multi-body dynamics, and non-linear optimization (see Figure 1-1). We also draw inspiration from work in the biomechanics and neurobiology communities. By standing on the shoulders of giants, we realize novel methods for the purpose of 3D character animation and the contributions of this thesis are discussed below.

Skilled motion synthesis. We describe an approach for single-handed dexterous manipulation. This approach is one of the first methods for online motion synthesis involving object manipulation. The algorithm uses a combination of machine learning and offline optimization to generate a control policy for complex manipulation tasks involving contact. Only the end configuration of an object is specified by the user; the full trajectory is not known a priori. The approach was successfully applied to a 23 DOF physics-based hand model that was subsequently able to manipulate various objects in a series of re-orientation tasks.

Physics simulation. We propose a reduced, accelerated model for the simulation of articulated structures containing loops. The approach uses a first-order model to approximate the behaviour of compliant kinematic chains, effectively speeding up the simulation by an order of magnitude. Additionally, the process of updating the configuration of rigid bodies is done in a parallel fashion and is suitable for implementation on multi-core platforms, for instance, as a GPGPU program. Although developed with grasping simulation in mind, the method is also useful for simulating other scenarios involving articulated mechanisms, such as the suspension of a vehicle or a tree swaying in the wind. Therefore, it is relevant to a wide variety of real-time virtual environment applications, such as video games and training simulations.

Appearance modeling. We propose a method for synthesizing colour changes that occur in the fingertip when grasping or manipulating a physical object. Colour changes are due to local redistribution of hemoglobin concentrations; a function approximator is learned to estimate these concentrations from finger pad pressures. The method requires a small memory footprint and is efficiently computed using a compact shader program. Integrating with a new or existing simulation application is straightforward, and requires only marking areas where colour variations occur. This is the first work that we know of that focuses on providing fast colour changes, due to interaction forces, with a grasped object.

Interaction capture. Finally, we introduce a novel sensor ensemble for performing interaction

capture. The setup uses off-the-shelf components to capture pressure distributions across the palm and fingers, as well as joint motion information. Data collected from a human subject is used to build a postural synergy for climbing tasks. An optimization algorithm is used to learn grasps for handholds, allowing a dynamic character to grip and support itself on a climbing wall. The chapter ends with a discussion on the role of interaction capture for controller design.

1.3 Outline

In Chapter 2, we briefly review the state-of-the-art of physics-based character animation, grasping control, and human appearance modeling. We then present our control framework for single-handed dexterous manipulation in Chapter 3. This is followed by a discussion about FORK^{-1}S in Chapter 4, which demonstrates our reduced accelerated model for simulating articulated structures containing loops. In Chapter 5, we present our appearance model for synthesizing realistic colour changes due to physical contact between a finger and an object. A novel capture system for collecting hand motion data and information about interaction forces is introduced in Chapter 6. Finally, we provide a summary of our contributions and discuss future research directions in Chapter 7.

Chapter 2

Related work

In this chapter, we provide a state-of-the-art literature survey on methods related to those presented in this thesis. This is not limited to work on physics-based character animation; it includes work from robotics, rendering, and machine learning. We first review motion synthesis techniques, focusing on grasping and manipulation tasks. We then review physics modeling and simulation for articulated mechanisms. Finally, we examine methods for appearance modeling of skin colour changes and surface deformation.

2.1 Control for grasping and manipulation tasks

In physically based computer animation, synthesizing realistic character motions is essentially a control problem. Forces and torques are generated by the control algorithm in order to accomplish a task or desired motion. Where manual interaction is concerned, contact is a critical aspect, and a variety of control strategies can be used to accomplish object manipulation.

Previous approaches to the problem of control for grasping and dexterous manipulation can be categorized, roughly, as optimization or data-driven. Optimization techniques determine the exact contact forces needed to perform a manipulation task, at each instance in time or over a short planning horizon. Often this limits the type of contact that may occur (e.g., no sliding or rolling) and where contacts may form. Likewise, the exact trajectory of the object must be known ahead of time. Data-driven methods attempt to replay motion captured from a sensor device, such as

an optical tracker with markers and camera. Although very natural motion can be reproduced with this type of approach, it often fails to accomplish the task when conditions of the virtual environment are disparate from the captured environment. Also, it can be costly to capture motions for successful replay in every situation.

The grasping control work presented in this thesis includes aspects of both optimization and data-driven approaches. Continuous optimization is used to learn suitable motor control parameters which are analogous to short horizon planning. A corpus of hand poses is used to compute a latent parameter space in which motor control programs are learned. Also, inspired by the success of reinforcement learning methods for controlling characters in locomotion tasks, our control work builds a policy of controllers that generalizes to many simulation and task states.

A common approach for control of physics-based characters is to break the problem into phases. This is also the recipe adopted in our work, where automata based controllers use contact changes to trigger transitions in the phases of motion. Such controllers are a natural choice for modeling virtual motor control involving environmental interactions, such as grasping, manipulation, and locomotion. In other work, Pollard and Zordan (2005) present a physically based grasping simulation that combines a finite state machine with motion capture at the wrist and selected key poses. However, their method only performs a grasp and release of an object, whereas our controllers are capable of performing complex re-orientation tasks. Also, the state machine transitions use simple heuristics for triggering the release of the object, as opposed to a grasp quality metric.

Other animation work has performed dexterous manipulation from a grasping pose by optimizing the forces necessary to move a manipulated object on a pre-specified trajectory (Liu, 2009). These optimized forces are then used to drive finger motions with appropriate torques at the joints. However, the complete trajectory of the object is known a priori, and an initial grasp is required. Our work differs in that only the start and goal start of the object are required. Similarly, Ye and Liu (2012) use contact sampling to animate fingers, given motion capture data for the object. Interestingly, they note that it is important that the motion of the object comes from a captured manipulation, as opposed to a key-framed trajectory, for finger motions to appear natural. This is not unexpected, and provides motivation for using a goal-based approach, as opposed to simplifying the problem by first scripting or planning a path for the object.

Mordatch et al. (2012a) present a solution that produces impressive results, and it does not require a pre-scripted trajectory for the object. Their approach solves a sequence of space-time constraints with special treatment for contact. They avoid optimizing the motion of each joint by considering only end effector positions; finger poses are reconstructed with inverse kinematics (IK). An alternative here would be to simulate all of the finger joints, with an optimization that takes the form of a shooting method, as opposed to encoding physics as kinematic constraints. Such an approach would produce solutions that have better physical plausibility and hard contacts, but with higher computational cost due to the fidelity of the simulation. Liu et al. (2010) use a shooting technique to synthesize compelling character motions involving complex contact scenarios. However, their approach requires that a similar motion trajectory has previously been recorded.

In robotics work, Huber and Grupen (2002) demonstrate robust finger gaiting from simple closed-loop controllers. While not optimal, in the sense that they do not maximize stability or manipulability, the grasps generated by their technique do exhibit force closure. However, the motions are restricted to clockwise and counter-clockwise rotations, with a state machine controller designed specifically for the task. Their work is similar to ours, but they do not use a latent parameter space to compute control poses. Ideally, controller parameters could be learned automatically and result in a control policy capable of adapting to scenarios with changing goals, like the work presented in this thesis.

Han and Trinkle (1998a,b) introduce a framework for dexterous manipulation using two- and three-fingered robotic grippers. Although limited to spherical objects, their approach provides a recipe for manipulation control that includes contact planning, coordinated motion, and grasp stability. However, like Huber and Grupen (2002), Han and Trinkle (1998a) do not consider complex contact scenarios involving the full kinematic chain of the gripper.

Other work uses multi-modal control approach to perform motion planning for full body manipulation tasks. Hauser et al. (2007) break down the planning problem for robot pushing tasks into a sequence of walking, reaching, and pushing motions. These modes are high-level compared to the phases used by our controller framework. Their framework uses short planning phases (10-100 ms), making exploration costly for scenarios where high branching factors exist. Therefore, synthesizing motions for manipulation tasks with complex contact scenarios is protracted. This over-

head could be reduced by determining phase transitions not based on time, but contact changes. Our work schedules phase transitions according to discrete events within the simulation, resulting in longer phase durations, typically 200-1000 ms. Okamura et al. (2000) provide an excellent overview of dexterous manipulation in robotics and discuss the idea of mid-level control. In such a framework, transitions between phases are triggered by contact events. This is an appropriate strategy for manipulation tasks, and one which we use in our own control work.

More recent robotics work has used synergy-space planning for grasping and object manipulation tasks (Ciocarlie et al., 2007; Kumar et al., 2014). This effectively reduces the dimensionality of the controller parameter search, making the problem of contact and motion planning for a gripper with high DOF a tractable one. Ben Amor et al. (2012) use a low-dimensional sub-space built from a database of recorded human grasping postures to perform grasp optimization on a robot. The focus of their work is complementary to ours in that their approach synthesizes motions for "reach-and-grasp" tasks. Also, they must deal with correspondence problems between human and robot kinematics. This could be avoided by allowing the user to supply a pose corpus directly, using a simulation model.

Recognition by the robotics community that the grasping problem can be described in a low-dimensional manner is supported by the biomechanics literature. Santello et al. (1998) show that the variation in final imagined grasp poses for a large number of objects is quite small, with well over 80% of the variation explained by only two principal components. Similarly, Flanagan et al. (2006) observe that changes in motor control are triggered by discrete events, with the contact information provided by different mechanoreceptor signals. When this information is suppressed, it becomes difficult to perform fine manipulation (e.g., imagine trying to open a combination lock with fingers numbed by cold). This justifies adopting a phase-based approach to enable control for manipulation tasks, with phase transitions determined by contact changes.

In other work, there has been progress in resynthesizing human grasping motions. Kry and Pai (2006) describe a method where force measurements are used in conjunction with motion capture to estimate finger stiffnesses for use in a grasping simulation. The controller in this case is entirely feed-forward. While the resynthesized interactions appear natural due to the estimated compliance, there is no feedback to ensure that the resulting final object position and orientation match a desired

goal.

An important part of our motion synthesis work is that we use continuous optimization and machine learning to compute successful controllers, which produces a policy that can be used in a real-time simulation. In the context of locomotion, Coros et al. (2009) use reinforcement learning to create a control policy for performing a series of walking tasks, such as walking on a line. Their controllers benefit from a learned control policy in that they are made more robust by interpolating optimal control parameters from nearby states. Similarly, Wang et al. (2009) perform optimization for walking controllers that anticipate perturbation.

Model predictive control (MPC) is emerging as an important tool for generating plausible motion at interactive rates. The general approach here is that an underlying physics simulation is used to explore short-term solutions that make progress towards a goal state. Controller parameters are determined as part of an online sampling process, and MPC approaches tend to use importance sampling and pruning techniques to efficiently explore solutions (Hämäläinen et al., 2014; Kumar et al., 2014). However, these approaches still have significant computational overhead, and further development of these techniques is required to achieve true real-time frame rates.

2.1.1 Grasp quality and planning

Determining the quality of a grasp is an important aspect of planning. Grasp quality metrics evaluate the ability of a grasp to resist external perturbations (stability) and make progress on the task at hand (manipulability). Although Okamura et al. (2000) and Bicchi (2000) provide excellent overviews of these topics, this section highlights seminal work in this area.

Ferrari and Canny (1992) introduce a metric that computes quality as the maximum finger force required to resist an external wrench. This involves taking the convex hull of the grasp wrench space (GWS) over all contact forces. The quality is the minimum distance of the surface of the hull to the origin. This metric depends on the choice of the reference frame used to compute torques (e.g., the center-of-mass of the object), and an optimum grasp with respect to one reference system may not be optimal with respect to another. The metric developed by Teichmann (1996) avoids this problem by computing the quality measure as the radius of the largest sphere inscribed in the set of wrenches that a grasp can resist. However, this requires solving a linear programming problem, and

the methods developed by Ferrari and Canny (1992) and Teichmann (1996) are computationally expensive. Specifically, computing the convex hull of the GWS is costly, with naive algorithms having $O(n^3)$ time complexity. If this is done as part of online grasp planning, it can severely impact the performance of interactive and real-time physics simulations.

Pollard (1996) characterizes the quality of a grasp in terms of the task. In addition to the set of disturbance wrenches, the task wrench space (TWS) contains the set of wrenches applied on an object to achieve a given objective. Similar to the GWS, this space is effectively represented by a 6D convex hull, although it is commonly approximated by an ellipsoid (see Li and Sastry, 1988). A task-based quality is used by Li et al. (2007) to filter a collection of candidate poses obtained from a hand pose corpus. A shape-matching algorithm re-targets a database of grasps to new objects, choosing a grasp with the ability to apply forces that best match those required by the task. This approach is limited to static enveloping grasps, but it significantly reduces the manual effort required to synthesize a plausible grasp for hand animations.

In a somewhat different approach, Kyota and Saito (2012) incorporate a grasp taxonomy as part of their interactive application for designing hand animations. Their approach is capable of synthesizing precision grasps, in addition to the power type of grasps generated by Li et al. (2007). The taxonomy represents a hierarchical categorization of grasps, providing a coarse alignment of the hand and fingers depending on the size of the object being grasped and required dexterity (Cutkosky, 1989).

2.2 Simulating articulated structures

A main objective of our work is physics simulation at real-time frame rates. This is a common goal for many aspects of physics-based animation, such as simulating deformation, friction, contact, and collision. We now examine work on modeling and simulating physical systems at different levels of fidelity. Since our focus is on human hand models, literature on articulated multi-body dynamics is relevant. The book by Murray et al. (1994a) provides an excellent overview of the mathematics of rigid motion, twists, wrenches, and adjoint transformations between different coordinate systems. We consider it essential background reading for this thesis, particularly for the material in

Chapter 4. A brief introduction to rigid body kinematics and related definitions is also provided in Appendix A.

One approach for the simulation of multi-body mechanical systems is to use a constrained full-coordinate formulation. Such systems can be solved quickly with sparse methods, and linear time solutions are possible when the structure has the connectivity of a tree (Baraff, 1996). This type of constrained, multi-body simulation is popular for its simplicity, and is available in a number of different software libraries including Vortex, PhysX, Havok, Box2D, and the Open Dynamics Engine. However, numerical drift must be addressed using stabilization techniques (Ascher and Petzold, 1998). Loops also result in redundant constraints that require special attention in the solution of such systems (Ascher and Lin, 1999; Faure, 1999). The simulation approach we later demonstrate in Chapter 4 uses a spatial coordinate formulation, but does not require any specialized solvers or constraint equations for the internal structure. Since we are only concerned with the end bodies in the kinematic chain, the behaviour is well approximated by a reduced set of equations resembling a mass-spring-damper system.

An alternative approach to a full-coordinate representation is to formulate the system using minimal coordinates (Featherstone and Orin, 2000). in essence, the joint angles. Straightforward linear time solvers have been used for decades, while divide-and-conquer approaches permit parallel algorithms with logarithmic time complexity (Mukherjee and Anderson, 2006; Featherstone, 2008). Mechanical structures with loops likewise require special treatment and modified solvers. Various libraries based on minimal coordinates exist, such as SD/FAST, which is commonly used in mechanical engineering applications, and RTQL8, which is designed specifically for character animation and simulated robots. Other approaches have been proposed for reducing multi-body dynamics. These resemble neither minimal coordinates, nor a full coordinate constrained multi-body system. Redon et al. (2005) use adaptive dynamics and a reduced coordinate formulation to perform rigidification of selected joints, which allows for faster simulation.

We note that IK techniques provide a possible solution for determining the internal configuration of articulated mechanisms. There are a variety of fast methods for solving over-constrained IK problems using singularity-robust inverse computations (Yamane and Nakamura, 2003) or damped least squares (Buss and Kim, 2004). Again, special treatment is required for loops in the kinematic

structure. While the motion generated by these methods can appear natural, physical properties such as mass and joint stiffness are not considered; dynamical effects, such as secondary motion, are missing from the system.

The dynamical equations of motion are important in robot control and analysis. Khatib (1987) uses projections of system dynamics as a central part of an operational space formulation. An incremental projection process is used to produce a reduced model for the dynamics of a robotic manipulator located at the end of a kinematic chain. Effective end-point dynamics can also be estimated from data. For instance, model fitting has been applied to human fingertips and hands (Hajian and Howe, 1997; Hasser and Cutkosky, 2002). However, a dynamics projection approach is preferable because the fitting process can become difficult when data are complex. Fitting a simple 6D linear mass-spring is undesirable when the force-to-displacement relationship in the full model exhibits non-linearities and bifurcation behaviour. Sampling the system behaviour can be expensive and does not fit the desired work flow of interactive simulators. It is not unreasonable to impose a simple model and to identify its behaviour based on a projection of linear compliant or PD controlled behaviour of joints. The simplification produces a computationally inexpensive first-order model, with a plausible response corresponding to a slightly modified set of non-linear joint controllers.

In contrast to redundant coordinate formulations, modal reduction of rigid articulated structures is possible, and has been used to animate animals (Kry et al., 2009) and synthesize stylistic character motions (Nunes et al., 2012). There has also been a vast amount of relevant work in computer graphics which exploits modal vibration for reduced models in soft-body simulation and deformation. A good survey can be found in a state-of-the-art report by Nealen et al. (2006), while other alternative elastic simulation reduction techniques continue to be an active area of research (Nesme et al., 2009; Barbič and Zhao, 2011; Kim and James, 2012; Harmon and Zorin, 2013). Frictional contact computations can also be simplified in a variety of ways, such as exact Coulomb friction cones, discretized friction pyramids, box constraints, or penalty based methods (Duriez et al., 2006; Yamane and Nakamura, 2006; Parker and O'Brien, 2009). With respect to the contact equations, the contact patches can be discretized at arbitrary resolutions (Allard et al., 2010), and collision detection and response can be modified to produce various plausible animations with

different fidelity levels (O'Sullivan and Dingliana, 2001).

2.3 Deformation and appearance changes of skin

Although producing plausible motion is a central concern for methods in human character animation, synthesizing appearance qualities, such as texture, colour, and skin deformation, are equally important. Over the years, many improvements have been proposed to enhance the appearance of animated characters beyond the standard linear blend skinning model (Magnenat-Thalmann et al., 1988). This section provides an overview of such techniques, with a focus on appearance modeling for hands.

Kinematic quantities, like joint angles and posture, have successfully been used to drive geometry changes in the character model. For instance, pose space deformation interpolates geometry corrections using radial basis functions (Lewis et al., 2000), and shape-by-example interpolates corrections mapped back into the rest pose with the inverse skinning transform (Sloan et al., 2001). Likewise, the EigenSkin technique interpolates shapes within a basis computed using PCA (Kry et al., 2002). This provides a memory-efficient representation for interpolating the geometric changes. The appearance modeling work presented in this thesis makes the observation that similar techniques are appropriate for modeling skin colour variations. Instead of interpolating geometric shape, we interpolate a blood concentration texture and use this to compute a correction to the skin colour.

In the context of physics-based animation, simulation quantities may also drive appearance updates. Borshukov et al. (2007) use playable universal capture where the texture appearance is driven by simulated attributes (i.e., contacts in the physics simulation), rather than a previously recorded appearance trajectory. Our data-driven appearance model interpolates textural changes based on contact forces, and a similar reduction method to Kry et al. (2002) permits a compact implementation as a shader program.

Sueda et al. (2008) propose an approach wherein geometric features on the back of hand are driven by tendon simulations. Although anatomically accurate, this type of simulation is costly, and human character models are often not designed with the required level of fidelity. Alternatively,

Huang et al. (2011) capture and model pose dependent wrinkles on hands, while other popular approaches build wrinkle texture maps into the character rig to provide these details without the cost of geometric modeling (Oat, 2007; Dutreve et al., 2011).

There are important changes in the appearance of the fingertips during posture changes and contact. In fact, it is possible to estimate posture and touch force using sensors that measure colour changes under the fingernails (Mascaro and Asada, 2004). Inspired by this difficult inverse problem, our focus is the creation of a data-driven model suitable for synthesizing these appearance changes. Also, while it may be convenient to model appearance variations in RGB colour space, it is preferable to work in the space of changing hemoglobin concentrations. Displacement of blood due to contact is the primary explanation for the change in appearance; building a model that works directly in the space of hemoglobin concentrations allows for easier reuse of captured data across different fingers and different people. A variety of techniques have been devised for estimating these quantities for medical purposes. For instance, camera-based sensors can be constructed to measure melanin, hemoglobin, and oxygenation of tissue, through the use of a known spectral illumination (Jakovels et al., 2011). More recently, it has been shown that Wiener estimation methods can produce estimates from an appropriately white balanced RGB camera (Nishidate et al., 2013). In computer graphics, Tsumura et al. (2003) use independent component analysis to estimate melanin and hemoglobin components using a single image, such as the image of a face.

Our appearance modeling is also closely related to that of Tsumura et al. (1999). They use independent component analysis in a negative log colour space to model appearance changes due to hemoglobin and melanin concentration, with the assumption that a simple Lambert-Beer scattering law explains skin colour. Parameters used by their model are conveniently estimated from example images. The difference with our work is that we do not need to extract melanin concentrations because we can focus entirely on the appearance variation due to hemoglobin concentration variations across an image dataset.

In contrast, Jimenez et al. (2010) produce skin colour variations from a lookup table indexed by hemoglobin and melanin concentrations. Their objective is to model the appearance of dynamic faces using a skin appearance rig. They use hemoglobin maps to control skin colour, permitting variation of appearance under the deformation of different blend shapes, or due to other conditions,

such as exercise or alcohol consumption. Kider et al. (2011) use a data-driven appearance model, driven by fatigue, which includes models of flushing and perspiration appearance. Boukhalfi (2012) focuses on face geometry during strenuous exercises. Other work uses hemoglobin estimation methods to produce skin colour variations (Tsumura et al., 1999, 2003; Jimenez et al., 2010).

Donner et al. (2008) specifically focus on hands. They fit a multi-spectral layered model to data collected from human skin samples. By modifying parameter maps, they show the possibility of appearance changes due to blood being squeezed out of areas deformed by contact. Within the context of our work, an interactive physics simulation could also be used to drive realistic colour changes.

Chapter 3

Multi-finger manipulation control policies

This chapter presents our approach for physics-based one-handed manipulation. A key feature of this approach is that it does not require a scripted path for the object. Instead, only the goal configuration of the object is specified and allows the trajectory to be influenced by hand geometry and dynamics of the finger and palm. This is useful for generating plausible and natural manipulation motions, and is relevant to many scenarios where only the final object configuration is important. For instance, preparing a coin for insertion into a vending machine, rotating a small package to read its label, or orienting small parts as part of a larger assembly task.

3.1 Overview

Problems related to simulated grasping and manipulation have received significant attention from the computer animation and robotics communities, with extensive work addressing the issues of motion planning, contact placement, and grasp quality. In contrast to high-level motion planning techniques that solve complex problems through a sequence of actions, we use a mid-level control approach (Okamura et al., 2000). By introducing an automata-based structure, the controller produces cyclic finger gaiting actions, wherein contact related events trigger different low-level controller phases. Adjusting a volume dial, manipulating a ball held in the hand, or removing a lid from a jar are examples that work well with this approach; the goal can be achieved by chaining

together a number of similar turning actions with repeated releasing and re-grasping interleaved to reposition the contacts. These three phases are termed *approach*, *actuate*, and *release*.

Continuous optimization is used to compute the parameters necessary for our mid-level controller phases. The objective is to successfully perform a manipulation task, but also to produce a physically plausible, natural motion sequence. The controllers tend to work well for a collection of nearby states, and because several cycles are often necessary to reach farther goals, a control policy is built using reinforcement learning (RL) and interpolation of controller parameters. Unlike traditional RL, a discretized action space is not used and parameter selection occurs in a continuous manner for each state. This learning approach helps to adjust the release phases so that fingers are better positioned for improved progress toward the goal in future cycles. More importantly, once the policy has been computed, it is useful for simulation of goal-oriented manipulation in real-time.

Although motion capture is not used, a selection of natural hand poses is used to compute a reduced parameter space for the low-level controllers. This limits the number of degrees of freedom that are needed to include in searching for solutions, and encourages the use of natural hand poses. Despite the reduced degrees of freedom, we still have a full simulation that produces poses outside of the reduced pose space, with finger joints bending to accommodate contacts.

This method makes important progress toward the development of improved virtual humans that can perform successful goal-oriented physically based interactions with virtual objects in real-time. The contributions of this work are as follows:

- A novel framework for synthesizing motions for human manipulation problems where the generated motions exhibit finger gaiting;
- A reduced search space based on natural poses to increase the performance of our method while ensuring the use of plausible hand shapes;
- Learned control policies that run in real-time;
- An analysis of the robustness of the control policies, providing insight into the selection of learning parameters, as well as providing indications on how to improve the low-level controllers used by our framework.

3.2 Multi-phase controllers

Our approach is motivated by the observation that human finger motion exhibits pseudo-cyclic characteristics, or finger gaiting, for a broad range of hand manipulation tasks. The fingers move in a coordinated fashion with an effort determined by one of three distinct phases: (i) a pre-shaping and finger planting phase wherein the hand forms a stable grasp around the object, (ii) an actuation phase in which wrenches due to contact forces are used to translate and rotate the object toward some desired configuration, and (iii) a release phase wherein the fingers adjust to a pose that is suitable for a subsequent approach phase. We refer to these phases simply as approach, actuation, and release (see Figure 3-1).

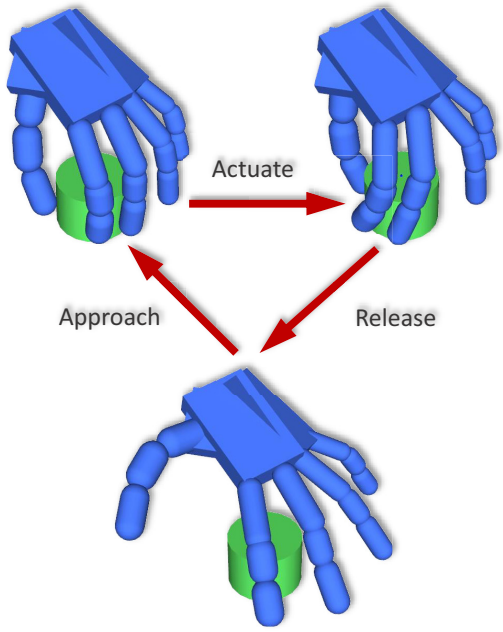

Figure 3-1: *Our mid-level control strategy.*

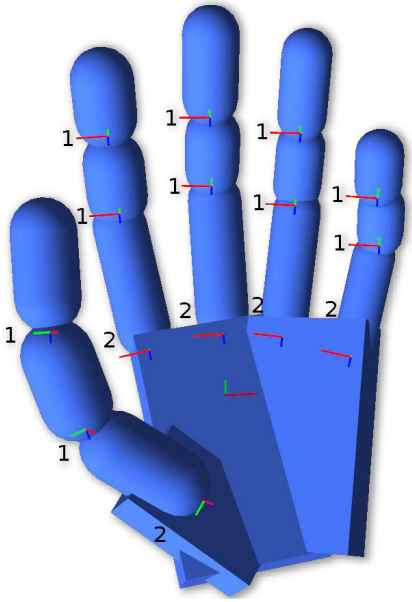

Figure 3-2: *The hand model used in manipulation experiments. The associated DOF is shown for each joint.*

These phases represent strategies that are encompassed by an automata-based controller architecture. Individual controllers use a set of three reference poses, $(\tilde{q}_0, \tilde{q}_1, \tilde{q}_2)$, to guide the hand in order to accomplish a manipulation task. Each pose consists of 20 joint angles corresponding to the degrees of freedom of the hand. The method for selecting target poses is discussed later, in Section 3.3.1. During the ith phase, we apply joint torques, τ, computed as

$$\tau = K\left(\tilde{q}_i - q\right) - D\dot{q}\,, \tag{3.1}$$

where K and D are the joint stiffness and damping matrices, respectively. The hand model used in our experiments is shown in Figure 3-2.

Notation. We use $\tilde{\cdot}$ to denote a target, or desired, quantity. Throughout the rest of this chapter we use the integer subscripts $0, 1, 2$ on scalar and vector parameters to denote a correspondence with each of the approach, actuation, and release phases, respectively.

3.2.1 Phase transitions

Phase transitions occur asynchronously and are tied to contact and joint limit events occurring within the simulation. In this section, we describe the conditions used to trigger transitions between phases.

The initial obstacle faced in many grasping problems is determining where to form finger-object contacts so that manipulation may be performed. This is the main objective of the approach phase, wherein pre-shaping occurs and finger end effectors ultimately make contact. It is most beneficial to end in a configuration that results in a stable grasp and good dexterous potential for manipulating the object. The subsequent phase involves actuating the object using contact forces, typically until further actuation is no longer possible (i.e., due to joint limits) and some or all fingers break contact. The hand then moves toward a recovery pose where pre-shaping and approach can begin again, repeating the cycle.

The approach phase ends once the fingers have planted and the desired grasp quality, $\tilde{Q}$, has been achieved. Quality here means that some stable, dexterous manipulation is possible, and there are several possibilities for measuring this quantitatively. We use a metric that is computationally inexpensive, but effective. Details about estimating grasp quality are provided in Section 3.3. Once the grasp quality condition is met, the approach phase transitions to the actuation phase.

At this point, the fingers are ready to manipulate the object. The direction of manipulation is a result of contact with the object and the accumulation of joint torques as computed by the PD control given in Equation 3.1.

During actuation, joint torques are applied until the grasp quality drops below an acceptable threshold, indicating that dexterous manipulability is no longer possible and the controller transitions to the release phase.

The transition between release and approach occurs when the total joint velocity of the fingers

becomes small, indicating that the desired pose has been reached, or that motion is hindered due to contact forces. There is also a transition when the allotted time for the phase has elapsed. The controller state is set to the approach phase and the cycle repeats. Contact information is not used to trigger a transition out of the release phase.

There is no assurance that a stable grasp is maintained during the release phase. However, it is assumed that "good" trajectories ultimately lead to the goal state; this information is encoded in the value function, $V(s)$. We include the value function as part of the selection process for controller parameters. This is a subtle, but important, aspect of our approach and further details are provided in later sections.

For all phases, we force a transition to the next phase if the joint velocities of the hand become small or the duration of a phase exceeds a maximum value, T_{max}. The one exception is when the goal has been reached, in which case the hand holds in either the actuate or release phase, waiting for the goal to change. The choice here is to let the hand remain in the actuation phase, ready to apply forces to achieve a new goal, or to remain in the release phase, allowing the hand to be moved between objects as part of a higher level control.

3.3 Control policy creation

In this section, we provide details on how to build a control policy for object manipulation tasks, beginning with a description of the simulation environment. Since our work focuses on single-handed manipulation tasks, state information regarding a full character skeleton is ignored; only the wrist and fingers joints are considered. Therefore, each state vector, s, contains the joint angles of the hand, q, the orientation of the object, θ, and its 3D position, x. The object orientation is stored as a quaternion, and both the position and orientation use a coordinate frame affixed to the wrist of the hand model. A homogeneous transformation matrix is used to transform the object's position and orientation from a global coordinate frame to a hand centric frame. The state vector is updated at each time step.

Other components of the simulation state, such as the linear and angular velocity of the object and hand joint velocities, are used to initialize the dynamics simulation when evaluating control pa-

rameters. However, we found these state components had little effect on the results when querying the control policy for optimal control parameters. This can be partly explained by the quasi-static nature of the hand based on the stiffness and damping control parameters we use. Therefore, we exclude all velocity level quantities from the state when building our control policy function.

An action, a, is represented by the tri-phase controller described in the previous section. Each action is composed of a sequence of three desired hand poses, $(\tilde{q}_0, \tilde{q}_1, \tilde{q}_2)$, which are determined during the offline continuous optimization stage, as described in Section 3.3.1. The control policy, $\Pi(s)$, provides a mapping from the environment state to an optimal action, a^*, whose control parameters are used to bring the environment to a higher valued state by making progress on the task. Progress occurs by means of a forward dynamics simulation, and the value of being in state s is stored in the value function, $V(s)$.

The value and control policy functions are represented by a k-nearest neighbours (k-NN) function approximator. Distance between neighbouring states is computed as a combination of state components. The distance between states s_a and s_b is computed as

$$d(s_a, s_b) = \beta_q \left\| q_a - q_b \right\| + \beta_x \left\| x_a - x_b \right\| + \beta_\theta \left\| \log(\theta_a^{-1}\theta_b) \right\|.$$

Of these components, the distance between hand postures is most critical for selecting the most appropriate action. Our implementation represents the object's position in centimeters. The range of values for this component is similar in magnitude to the angular component, which is measured in radians and computed by the logarithm of quaternions.[1] Hence, the distance between hand postures tends to dominate the function $d(s_a, s_b)$. The scalar values β_q, β_x, β_θ are used to weight contributions of the pose, object position, and orientation components, respectively. Through empirical evaluation we determined that $\beta_q = \beta_x = \beta_\theta = 1.0$ gave good results, and this is coincidentally equivalent to an unweighted metric.

The interpolation weight for the ith neighbouring state is computed using an inverse distance-squared kernel,

$$w_i = \frac{1}{\sigma} \frac{1}{\left(d(s, s_i)\right)^2}.$$

[1] Briefly, $\log(\theta_a^{-1}\theta_b)$ gives the geodesic curve connecting the unit quaternions θ_a and θ_b (see Kim et al., 1995).

Procedure 3-1 The value iteration algorithm used to build the control policy.

while not converged **do**
  for $s \in S$ **do**
    $a^* = \text{OPTIMIZE}(s)$
    $s' \leftarrow \text{DYNAMICS\_SIMULATION}(s, a)$
    $\tilde{V}(s) = R(s, a^*) + \gamma V(s')$
    $V(s) \leftarrow \alpha \tilde{V}(s) + (1 - \alpha) V(s)$
    $\Pi(s) = a^*$
    if $\text{ISNOVEL}(s')$ **then**
      $S \leftarrow S \cup s'$
    end if
  end for
end while

Convexity is ensured by computing a normalizing factor $\frac{1}{\sigma}$ such that $\sum_{i=1}^{k} w_i = 1$. The optimal action for an arbitrary state is estimated by interpolating the actions for the k closest states in the policy,

$$a^* = \sum_{i=1}^{k} w_i \, a_i \, .$$

3.3.1 Fitted value iteration

A control policy is learned using the value iteration method (Sutton and Barto, 1998). The collection of states, S, that make up the instance-based functions $\Pi(s)$ and $V(s)$ is bootstrapped with random states that are chosen uniformly across the pose and task space. Additionally, we make sure these are physically valid states. Specifically, no intersections occur between the fingers and the joint angles are within the constraint limits of the hand model.

Controller parameters are determined by performing a multi-objective optimization. For each state $s \in S$, candidate actions are evaluated using a forward dynamics simulation. This means that the parameter search occurs across a non-linear, rugged landscape.

The value function is updated using the reward and value of the proceeding state, s', at the end of the simulation. The method $\text{ISNOVEL}(s')$ uses a threshold, ϵ, to determine if the state is sufficiently novel and the method returns true if $d(s, s') > \epsilon$ for all $s \in S$. Novel states are added to S. Note that for the states used to bootstrap the control policy, $\text{ISNOVEL}(s')$ is true. Pseudo-code for the

value iteration algorithm is provided in Procedure 3-1.

CMA-ES (Hansen, 2006) is used to determine the optimal controller parameters at each state s. The phases are not optimized in isolation, and instead CMA-ES optimization is performed over one complete cycle of the state machine. The coupling between phases is key to our approach, since evaluation of the success of a controller is determined not only by the minimization of a set of objective terms for each phase independently, but by their performance as a sequence.

Each objective term is a function of the simulation and task state across all controller phases, leading to the composite objective function

$$\min_{a^*} L_0 + L_1 + L_2 + L_g \ .$$

Here, L_0, L_1, L_2 pertain to the approach, actuation, and release phases, respectively; L_g is an aggregation of global terms pertaining to all phases. The contents of the phase specific and global objective functions are discussed in later sections.

Notation. The terms of the composite objective function are denoted by the symbol L, and subscripts are used to distinguish individual quantities pertaining to phase-specific features. These features are quantities accumulated throughout a phase, or computed at phase transitions. The notation $\sum_{t \in T_i}$ is used to indicate a term that is accumulated over period T_i, with its value being sampled at each time step.

3.3.2 Multi-objective optimization

Here we discuss the details of the optimization function. The explanation is organized according to phase, starting with approach. This also includes details on the global terms, which are applicable to all phases.

Approach. The approach is a pre-shaping phase, wherein the agent (hand) makes contact with the object in preparation for actuation. The objective function for this phase is simply

$$L_0 = l_0 \max\left(0, \tilde{Q} - Q_{T_0}\right) \ ,$$

which ensures a penalty if the grasp quality at the end of the phase, Q_{T_0}, is below the desired grasp quality, $\tilde{Q}$. As such, the duration of the approach phase T_0 can equal the time out T if the grasp quality was not achieved, in which case L_0 will be some positive value to penalize this action. Alternatively, if grasp quality is achieved, then T_0 is the time at which the approach phase transitions into activation, and the objective L_0 will simply be zero. Overall, this encourages the optimization method to find solutions where the fingers are planted and ready to actuate the object.

Grasp quality.

In this section, we describe our method for computing the grasp quality. A common metric for determining grasp quality is by computing the *force closure* of the GWS (Murray et al., 1994b). If the convex hull of the GWS contains the origin, it has closure, and any external wrench acting on the object can be resisted by a convex combination of the grasp wrenches. However, computing the convex hull at each simulation step is a computational bottleneck, and instead we estimate the grasp quality with an ellipsoid, similar to the method outlined by Klein and Baho (1987).

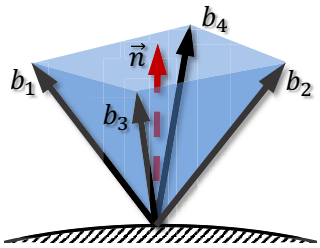

Figure 3-3: *Discretized friction cone showing the contact normal, $\vec{n}$, and the frictional basis vectors, $b_{1...4}$.*

The matrix $G \in \mathbb{R}^{6\times(mN)}$ is assembled from a set of representative vectors, accounting for N finger-object contacts and m basis vectors at each contact that approximate the Coulomb friction cone (as shown in Figure 3-3). The positive linear span of the friction cone represents the set of potential forces a contact may apply to the object. For our experiments $m = 4$, and the basis for the ith contact is denoted by $(b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4})$.

The matrix, $\Gamma_i \in \mathbb{R}^{6\times3}$, is used to map contact forces in the global coordinate frame to wrenches in the object's local frame. This matrix has the block form $\Gamma = \left[R^T - \hat{p}R^T\right]^T$, where R is a rotation matrix transforming vectors in the global coordinate frame to the object's coordinate frame, and p is the contact location used to form the skew symmetric cross product matrix $\hat{p}$. This gives the set of wrench vectors

$$W_i = \begin{bmatrix} \Gamma_i b_{i_1} & \Gamma_i b_{i_2} & \Gamma_i b_{i_3} & \Gamma_i b_{i_4} \end{bmatrix}, \tag{3.2}$$

and G becomes the block row matrix $G = [W_1 \ \ldots \ W_N]$.

We compute the singular value decomposition $G = U\Sigma V^T$, and estimate grasp quality, Q, as the smallest singular value of G. By avoiding poses where the singular value is equal or close to zero, the optimization is more likely to select non-singular grasp configurations. The columns of U provide the axes of the wrench ellipsoid, and the axis corresponding to the smallest singular value provides a direction in which the least amount of force and torque is needed to break the grasp. Informally, the smallest singular value corresponds to a GWS direction that is "weakest".

The example illustrated in Figure 3-4 shows the convex hull computed for a grasp in a 3D wrench space. The grasp lacks closure since the hull does not contain the origin yet the wrench ellipsoid is not degenerate and does contain the origin. Therefore, as an additional check, we ensure that the cone spanned by the contact wrenches is at least π, otherwise $Q = 0$. Although this approximated metric often leads to grasps which have the force closure property, there is no assurance of closure grasps throughout a manipulation. Rather, this heuristic sufficiently guides the optimization towards manipulable and plausible grasps.

We justify the use of the wrench ellipsoid heuristic by considering computational performance. For complex scenarios involving approximately 10 contacts, the time required to compute the wrench hull with an Intel 3.2 GHz processor is 15 ms compared to 0.1 ms for the wrench ellipsoid based quality metric. In our experiments, there was little difference in the motions generated using these two quality metrics, so we choose the one that is less expensive to compute.

Actuation. It is during the actuation phase that most of the progress is made on the task. Wrenches acting on the object change its position and orientation such that it moves toward the goal state. Based on this assumption, it is necessary that the predominant objective for this phase is to minimize the task-based objective function

$$L_T = l_x\|\tilde{x} - x\| + l_\theta\|\log(\tilde{\theta}^{-1}\theta)\|.$$

Note that $L_T \geq 0$, and the minimal value occurs when the goal state is reached.

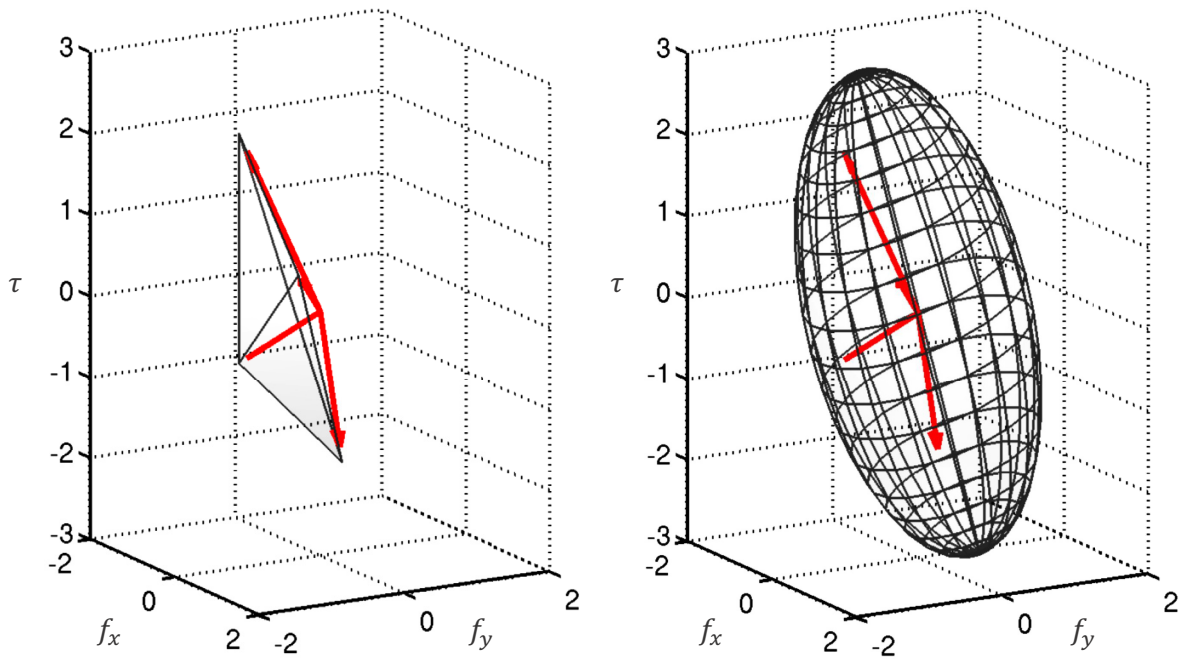

Figure 3-4: *Left, the wrench convex hull of a 2D grasp, with two linear force components (f_x, f_y) and an angular torque (τ); the 3D wrenches are drawn in red. The grasp does not have closure (i.e., the hull does not contain the origin), yet the wrench ellipsoid, shown on the right, is not degenerate. Therefore, an additional check is performed to ensure that the GWS spans a cone of at least π.*

The value function should reflect the optimality of the task state. By using the reward function

$$R(s) = -\|\tilde{x} - x\| - \|\log(\tilde{\theta}^{-1}\theta)\|,$$

this results in a $V(s)$ that is non-positive for any state s. Choosing an action that minimizes the objective term L_T will maximize the return of the reward function, meaning that in a greedy sense progress is made on the task. This duality is also exploited during the release phase for choosing a recovery posture by incorporating $V(s)$ as an objective term, maximizing future rewards.

During manipulation, it is also a requirement that the fingers maintain a certain degree of stability for the object. Using the same metric from the approach phase, a minimum level of grasp quality is maintained throughout the actuation phase by the objective

$$L_Q = l_Q \frac{1}{T_1} \sum_{t \in T_1} \max\left(0, \tilde{Q} - Q_t\right).$$

Here, T_1 is the duration of the actuation phase and Q_t is the quality at time t of the actuation phase. Note that this is similar to averaging the difference from the desired grasp quality throughout the actuation phase, but only accounts for frames when the quality falls below a threshold.

Additionally, we include a penalty term allowing the user to specify which of the M fingers participate in the actuation. An array of boolean values, p, contains an entry for each finger, indicating if it should participate (i.e., *true* if participating, *false* otherwise). The summed magnitude of contact forces affecting each finger is computed as

$$F_j = \sum_{k}^{N_j} \|f_{j,k}\|,$$

where $f_{j,k}$ is the kth of N_j contact forces between the object and finger j. If the value of F_j for a non-participating finger exceeds a threshold, η, a penalty proportional to the contact force is added. Conversely, if F_j for a participating finger falls below η, a penalty is also added. The penalty is accumulated at each time step of the phase as

$$L_P = l_P \sum_{t \in T_1} \sum_{j}^{M} \begin{cases} \eta - F_j & \text{if } F_j < \eta \text{ and } p_j \\ F_j - \eta & \text{if } F_j > \eta \text{ and not } p_j \\ 0 & \text{otherwise} \end{cases}.$$

For the results shown in this chapter, $\eta = 2.0$.

Assembling each of the task, quality, and participating finger penalty terms, the objective function for the actuation phase becomes

$$L_1 = L_T + L_Q + L_P.$$

Release. The primary objective of the release phase is to let the fingers break contact and move them in preparation for another approach. Since our tasks focus on re-orientating and re-positioning an object, spatial velocity of the object during this phase is penalized in order to ensure that progress made during the actuation phase is not undone. We introduce objective terms that pe-

nalize any spatial velocity throughout this phase:

$$L_\omega = l_\omega \sum_{t \in T_2} \|\omega_t\|_2$$
$$L_v = l_v \sum_{t \in T_2} \|v_t\|_2 \,.$$

Here, ω_t and v_t are the angular and linear velocities of the object at time t.

However, simply minimizing the spatial velocity of the object will not ensure successful manipulation, since the hand must recover to a pose where it can make progress during the subsequent approach phase. Upon transitioning to this phase, if the object has reached the target configuration, the hand simply maintains a static grasp on the object. Consider the case where the configuration of the object is not the goal state. Progress must be made in subsequent phases. Given the control policy, $\Pi(s)$, states corresponding to future progress are considered high value states. High value states are discerned using the value function, $V(s)$, and we include it as part of the optimization for the release phase to determine a good recovery posture.

Conveniently, like the reward function, the value function has an upper bound of 0 and lower bound of $-\infty$. Therefore, we can use its value directly as a penalty term in the controller optimization problem as

$$L_V = -l_V \, V\left(s_{T_2}\right),$$

where s_{T_2} is the state of the simulation at the end of the release phase. Note that a negative scalar is used to weight this objective term. By minimizing L_V, postures that result in good finger planting at the subsequent approach phase are encouraged.

At the beginning of each inner loop, the covariance matrix entries pertaining to the release phase are reset. This is necessary because the value function changes at each iteration of the algorithm. Thus, the solution found for the release pose in a previous iteration may no longer minimize $V\left(s_{T_2}\right)$.

Combining the velocity penalty and value function terms, the objective function for the release phase is

$$L_2 = L_\omega + L_v + L_V.$$

Global terms. In addition to the individual objectives for each phase, a global penalty term was added to minimize the energy used to perform the action, and to discourage contacts that use surfaces on the back of the fingers. The global component of the objective function is

$$L_g = l_\tau \sum_{t \in T'} \| K\left(\tilde{q} - q(t)\right) \| + l_h \sum_{t \in T'} \sum_{j}^{M} h_j(t),$$

where $h_j(t)$ is equal to 1 if there is contact on the back of finger j at time t, or 0 otherwise. This is determined by comparing the contact force with a vector defining the backhand direction of each finger segment. We use T' to denote the time interval necessary to complete a full controller cycle.

3.3.3 Tractability and implementation

In addition to using a grasp quality metric which is simpler and faster to compute, we have made a few other technical choices which significantly reduce the computing time of the CMA-ES optimization required for controller selection. For instance, we use a parallelized implementation of the OPTIMIZE(s) method. On a modern multi-core CPU, this reduced the time required to compute an optimal action by nearly an order of magnitude.

Rather than performing optimization in the full coordinate space, we were inspired by the work of Santello et al. (1998), which suggests human hand postures, when interacting with tools and everyday objects, may be represented using just a reduced postural synergy. Using a set of approximately 20 grasp poses, from which the user may select some or all, a reduced pose basis is constructed. Figure 3-5 shows examples of the poses we use for building our reduced pose space. The poses are manually designed and selected to be representative of grasps necessary for the tasks in our experiments.

Principal component analysis (PCA) is used to generate a set of orthogonal basis vectors in which to perform the controller optimization. Basis vectors are selected according to their component scores and the set of vectors remains the same for all phases. For the tasks shown in this chapter, the parameter space for controlling the joints of the hand is reduced from 20 per phase to just 3 to 5 per phase, depending on the task, giving a total of 9 to 15 parameters for each tri-phase controller. Note that the process of bootstrapping a control policy is done using the reduced pose vectors.

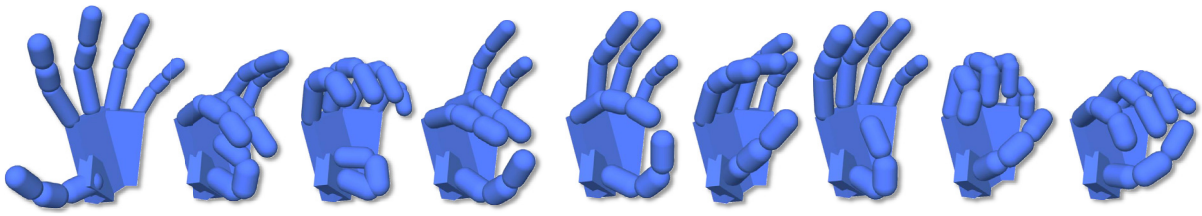

Figure 3-5: *Selected example poses from the corpus used to build a reduced basis for the control parameter search.*

3.3.4 Practical considerations

Constraint force mixing (CFM) is a feature of several off-the-shelf rigid body physics engines and is often a required, practical consideration for building stable simulations of complex systems. We use CFM to implicitize the joint torques needed in Equation 3.1. This permits us to take large time steps ($\Delta t = \frac{1}{60}$ s) while remaining stable. For the range of values of the stiffness and damping coefficients used by our experiments, it would require advancing the simulation at sub-millisecond time steps if PD control torques were computed explicitly (see Pollard and Zordan, 2005). This gives a significant speedup in the time required perform the CMA-ES optimization.

This approach is similar in spirit to the work of Tan et al. (2011), and although the formulation is different, the results are equivalent. They correctly identify the analogy between PD controllers and penalty based constrained dynamics. The controllable DOFs may be considered regularized constraints, where some violation of the position is allowed. For further details, we refer readers to the related literature on CFM (see Erleben et al., 2005).

3.3.5 Parameter tuning

For all results presented in this chapter, $\alpha = 0.8$ and $\gamma = 0.7$ are used as the learning rate and discount factor values, respectively, in the fitted value iteration. These are determined by experimentation and we found that convergence of the policy building algorithm was reasonably insensitive to changes in these values. Scaling is used to bring the range of values across objective terms to within an order of magnitude, and approximately unity across exemplary sequences. Scaling factors are computed once for each task, and are determined empirically by data collected from manually defined manipulation sequences. By scaling each objective term in this way, it simplifies the process of the tuning the weights, l, for the multi-objective optimization problem. These

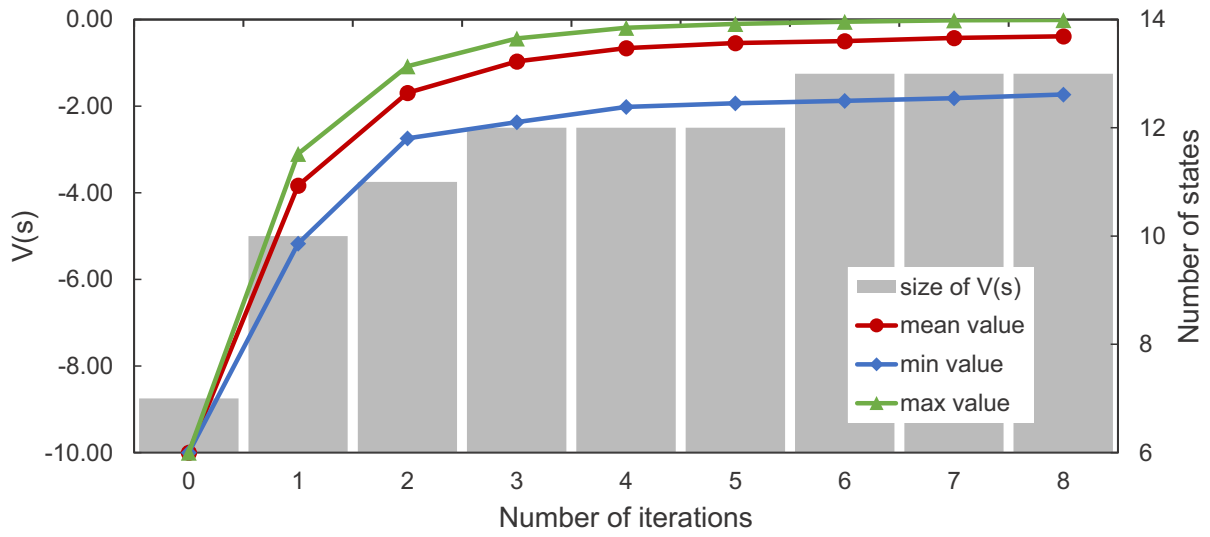

Figure 3-6: *Convergence of a dial turning control policy with $\alpha = 0.8$ and $\gamma = 0.7$. The mean value across states for the instance based $V(s)$ approaches zero, indicating controller sequences that achieve the goal are being found. Novel states are added mostly during early iterations of the algorithm.*

scaling factors are omitted from the equations in previous sections to improve readability. Finally, tuning the CMA-ES step size σ and population size λ is also done experimentally, such that the optimization finds suitable controller parameters within 30-60 seconds for a single simulation state.

3.3.6 Policy convergence

Figure 3-6 shows the evolution of $V(s)$ during each iteration of Procedure 3-1 for a dial turning task. The policy is initialized with 7 random states and $V(s) = -10$ for each state s. The policy grows in size as novel states are added at each iteration. Novel states are encountered mainly during preliminary iterations of the algorithm, occurring less frequently as the policy converges. Note that the mean and max of the value function approach zero, indicating controller sequences are being found that reach the goal state, since $R(s, a) = 0$ once the goal state is achieved.

3.4 Manipulation results

In this section, we provide results for two examples: dial turning and ball-in-hand manipulation. The tasks involve re-positioning and re-orienting an object to match a desired configuration. A

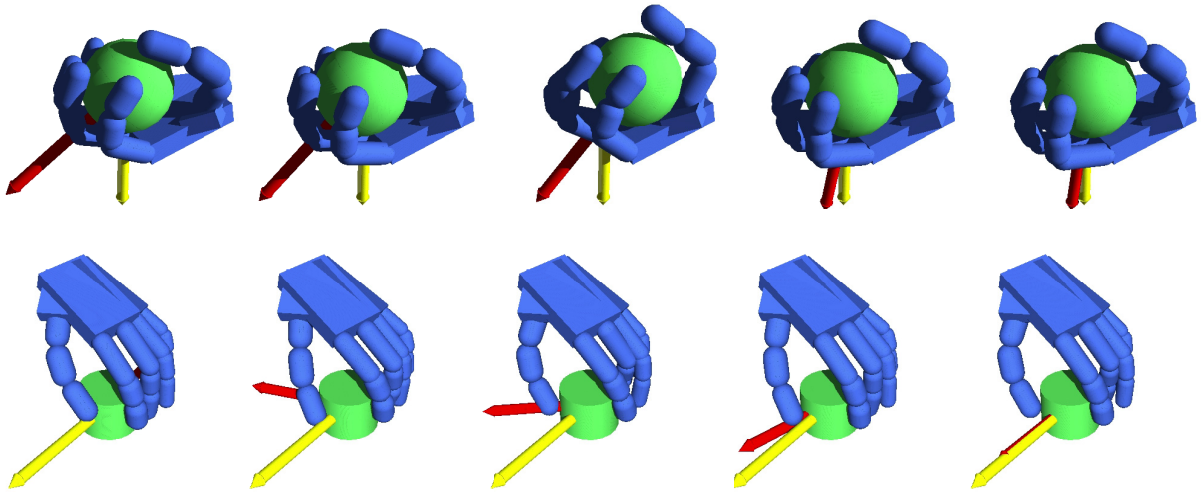

Figure 3-7: *Showing hand motion sequences for the ball-in-hand example (top) and dial turning example (bottom). The desired (yellow) and current (red) configuration are shown.*

collection of capsule and box collision geometries is used to model the hand, with finger segments being actuated by joints with 1 and 2 DOF (see Figure 3-2), for a total of 20 joint angles.

The Vortex[2] toolkit is used to simulate the forward dynamics, including contact and gravity forces. All results were obtained using a 6-core Intel i7 3.2 GHz processor and running 12 simulation threads for the CMA-ES optimization. Conveniently, Vortex supports the method of PD control discussed in Section 3.3.4 for hinge and universal joints; stiffness and damping coefficients may be assigned directly through the programming interface.

Although learning times may take hours, the result is a policy that runs in real-time. Individual iterations of our algorithm work like a space-time constraints optimization problem, giving a partial solution for performing the overall task. Learning times for specific tasks are discussed in Section 3.4.1 and Section 3.4.2.

Figure 3-7 shows a temporal sequence of hand poses generated for two examples. Complete animations are available on the project page.[3]

[2] http://www.vxsim.com/
[3] http://www.cs.mcgill.ca/~sandre17/multifinger/

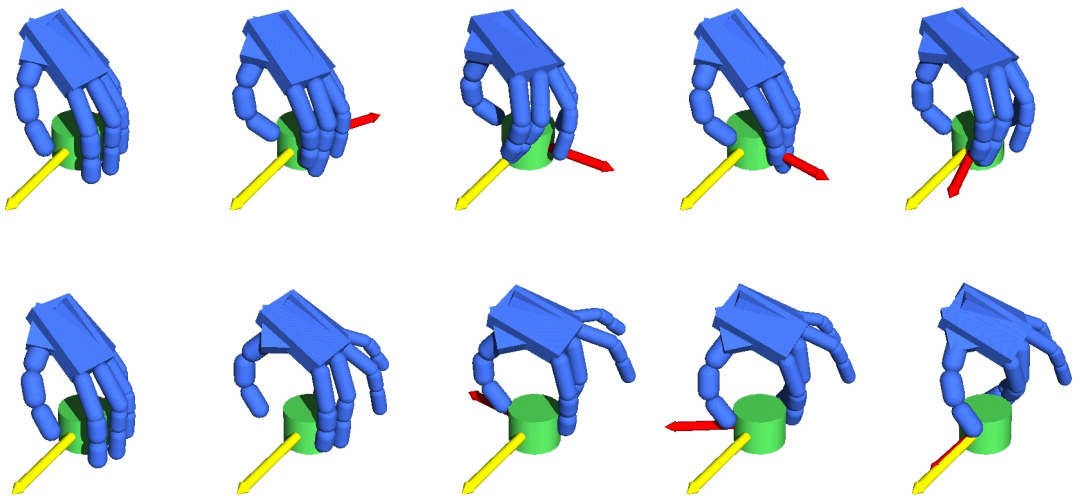

Figure 3-8: *Variations in the synthesized motion may be achieved by changing parameters of the multi-objective optimization.*

3.4.1 Dial turning

For the dial turning example, a cylindrical object is constrained using a hinge joint, similar to a mounted dial or volume knob. Since the object is constrained to rotate about a single axis and no linear motion is allowed, the reward function simplifies to

$$R(s) = -\left|\tilde{\theta}_{\text{hinge}} - \theta_{\text{hinge}}\right|$$

where $\tilde{\theta}_{\text{hinge}}$ and θ_{hinge} are the desired and current angle of rotation about the hinge axis, respectively. One of the trajectories generated by the learned policy is shown in the bottom row of Figure 3-7.

The mean wall clock time to perform the controller optimization for a single state s is approximately 18 seconds. For the example shown in Figure 3-7 (bottom row), the policy was bootstrapped with 16 states, finally growing to 32 states after 23 minutes of running our algorithm.

The CMA-ES parameters used by this example are $\sigma = 0.2$, $\lambda = 30$. The multi-objective weights and other parameter values are provided in Table 3.1.

	l_0	l_Q	l_x	l_θ	l_ω	l_v	l_h	l_V	l_P	l_τ	T_{max}
Dial turning	0.4	0.4	1.0	1.0	0.5	0.5	10	0.6	0.2	0.1	0.6 s
Ball-in-hand	0.4	0.4	1.0	1.0	0.5	0.5	10	0.6	0.0	0.1	1.0 s

Table 3.1: *Parameter values used to learn example tasks.*

3.4.2 Ball-in-hand

The ball-in-hand example involves re-orienting and re-positioning an unconstrained ball. The task state consists of the 3D position and orientation of the object in the hand frame. There is special consideration for states s' where, at the end of the controller optimization, the ball is no longer in contact with the hand. These are not added to S.

On average, it takes approximately 45 seconds to perform the controller optimization for each state $s \in S$. For the example shown in Figure 3-7 (top row), the policy was bootstrapped with 40 states, finally growing to 320 states after 4 hours 21 minutes of running our algorithm.

The CMA-ES parameters used by this example are $\sigma = 0.4$, $\lambda = 60$. The multi-objective weights and other parameter values are provided in Table 3.1.

3.4.3 Task variations

It is possible to synthesize variations of motion for a task by changing the weights of the multi-objective optimization problem and the maximum phase duration, T_{max}. Figure 3-8 shows two styles of motion generated for the dial turning task. For the controllers with shorter phase duration (top row), multiple cycles are required to reach the goal. The controller with a longer phase duration reaches the goal in one cycle, but only the thumb, index and middle fingers are flagged as participating.

3.4.4 Robustness of controllers

To evaluate the robustness of the control policy and individual controllers, we performed a series of experiments on the success of performing a task as a function of perturbations to s. A control policy was built using a small number of states (approximately 12) with the value of θ and $\tilde{\theta}$ remaining fixed. The states were carefully selected so that the learned controllers were capable of

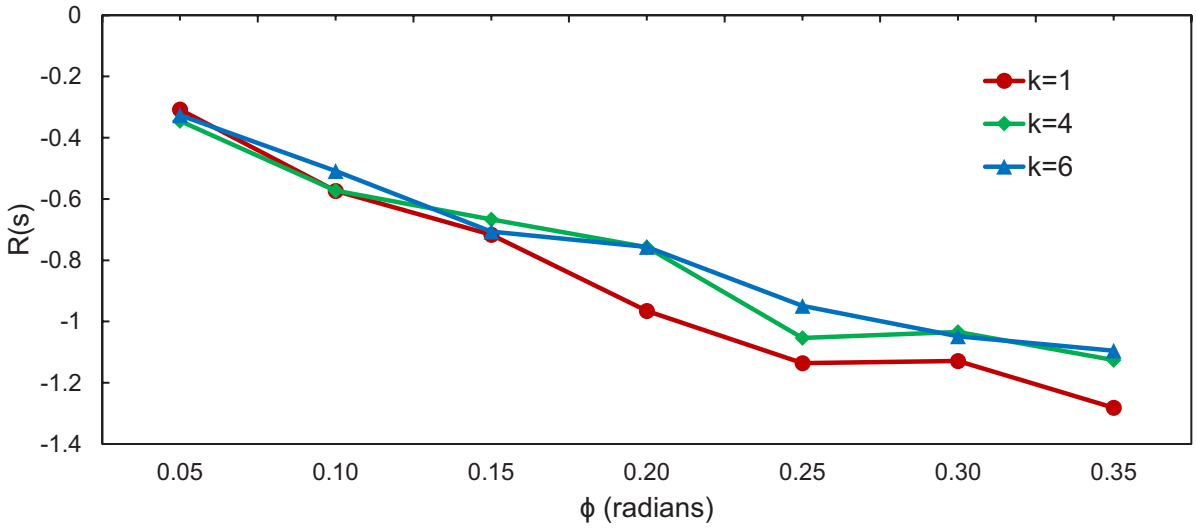

Figure 3-9: *The effect of perturbation, ϕ, on the effectiveness of the controller. Random joint perturbations in the range $[-\phi, \phi]$ are applied to the hand joints. Effectiveness is measured as the reward at the end of the controller cycle, $R(s')$. Note that the reward is averaged over 100 trials.*

reaching the goal state at the end of the release phase, or $R(s') = 0$. For this experiment, $\epsilon = \infty$ to avoid updates to the control policy.

The state of the simulation is initialized to $s_\delta = s + \left(\delta(\phi)^T, 0, 0\right)^T$, where $s \in S$, and $\delta(\phi)$ is a function used to generate a randomized perturbation to the joints of the hand. Specifically, $\delta(\phi) = (g(\phi), \cdots, g(\phi))^T$. The function $g(\phi)$ generates a random number with a uniform distribution in the range $[-\phi, \phi]$.

Figure 3-9 shows the reward as ϕ was increased from 0.05 to 0.35 radians. Each data point in the plot represents the mean value of $R(s')$ over 100 trials. As expected, when ϕ is small, the controllers perform well. However, as ϕ increases, performance decreases. Increasing k for the nearest neighbour interpolation mitigated this problem.

These experiments provide insight into the selection of learning parameters for the instance-based function approximators. For our experiments, $k = 6$ and $\epsilon = 0.22$, giving a trade-off of performance versus learning times.

Further analysis gives additional insight as to why the performance drops as ϕ grows. Figure 3-10 shows the duration of the approach and actuation phase as ϕ is increased. In the same graph, the average grasp quality at the end of the approach phase, Q_0, and throughout the actuation phase, Q_1,

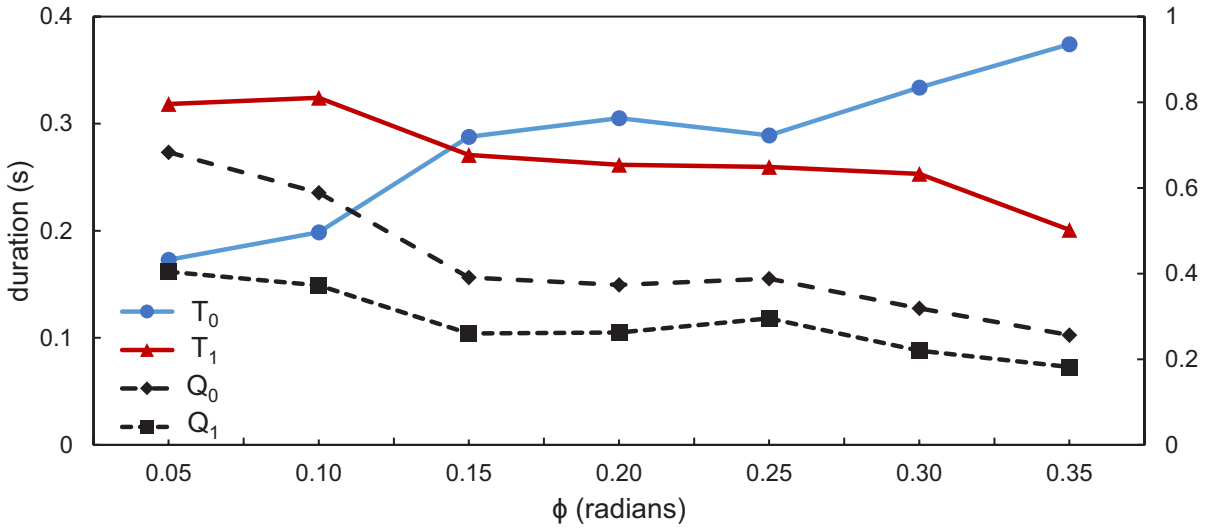

Figure 3-10: *The effect of perturbation, ϕ, on the duration of controller cycles and the grasp quality. The duration of the approach phase, T_0, is lengthened and the duration of the actuation phase, T_1, is shortened. Similarly, the average grasp quality at the end of the approach phase, Q_0, and throughout the actuation phase, Q_1, drops, indicating poor progress on the task.*

is plotted. Recall that, during the actuation phase, when the grasp quality drops below a minimum value, it triggers a transition to the subsequent phase. The plot indicates that the controller phases are transitioning early due to an insufficient grasp on the object, and less progress is being made on the task. The controllers used by our framework are feed forward in nature, and ignores aggregate features of the simulation, such as grasp quality, throughout the duration of the controller cycle. We speculate that incorporating some feedback about these features will not only improve the performance of our controllers, but also lead to sparser control policies.

3.5 Discussion

The results we have shown involve the manipulation of convex and symmetric objects. Generating control policies that perform well across variations in object size, shape, and mass is a challenge for our framework. Although our method does not exploit these properties for successful manipulation, and no examples involving non-convex or asymmetric objects are provided, there is no evidence indicating that control policies cannot be learned for other classes of objects. This is mainly attributed to the robustness of the CMA-ES method and its ability to find solutions that successfully achieve task objectives. However, complex object geometry may void the smoothness

assumption we make by interpolating controller parameters for a k-NN representation of $\Pi(s)$. One solution may be to increase the density of the function approximator and exploring more states. Similarly, tasks where object momentum is exploited, such as contact juggling, require adding simulation information to the state, such as the object's velocity. Both of these changes would result in longer learning times.

The work presented in this chapter is targeted at synthesizing cyclic, single-handed manipulation tasks. An interesting extension would be to modify the framework to perform a general class of manipulation, including those that involve aperiodic motion. One obvious approach is to include additional automata as part of the controller state machine, giving the controller a form wherein multiple outgoing transitions may occur at any state and additional parameters for the associated phases of motion. This makes batch optimization of controller parameters problematic, since branching determined by each new transition needs to be optimized independently. Using a hierarchical approach could be beneficial here, wherein each possible circuit of the controller graph is represented by a different policy. At run-time, a top-level agent then chooses the best action across all learned control policies. However, this increases the complexity of our approach, and would certainly increase the offline learning stage of our algorithm.

Large-scale motion planning is not considered by our work, since we focus only on single-handed manipulation. In all of our examples, the wrist is immobile, which may be unrealistic for some tasks where motion of the arm, elbow, shoulder, and other body joints may be useful. Note that for the examples shown in Figure 3-8, additional control parameters (2 joint angles per phase) are used to allow motion of the wrist. However, providing additional degrees of freedom to control other body joints is not appropriate given the restrictions of our phase-based controller architecture. Furthermore, aperiodic finger gaiting styles may not be synthesized without modifications to the controller framework.

This framework provides a powerful approach for online motion synthesis of dexterous manipulation tasks. The learning approach is an incremental one, where progress is learned over short motion phases. This style of motion synthesis has been gaining traction in the field (Mordatch et al., 2012a,b). However, the offline optimization step used to build our control policies is a protracted aspect of the method. This is due to running many forward dynamical simulations to evaluate

controller parameters. It can take several hours to learn manipulation strategies for a single object.

One way to reduce the time of the the policy building step is to speed up the simulation. In the next chapter, we introduce an approximate model for articulated mechanisms that accelerates the dynamical simulation by an order of magnitude.

Chapter 4

FORK^{-1}S: Compliant mechanisms with parallel state computation

Real-time physics simulation has emerged as a fundamental component of interactive virtual environments. There are many important applications, for example, in training operators of robots and heavy equipment, design of robots and mission planning, and simulating virtual humans in video games. In this chapter, we describe a technique for improving interactive simulations of scenarios involving complex multi-body mechanisms with contact. For instance, a grasping simulation of a human or robotic hand involves a complicated chain of compliant joints and distributed contacts. Collaborative grasping and manipulation with multiple people, multi-legged robots, and vehicle suspension systems can produce similarly challenging computational scenarios. Simulating these kinds of systems is difficult because they result in large, over-constrained systems of equations that, in general, require considerable computational effort to solve. Furthermore, special attention must be paid to the parameters of both the system and simulation to ensure stability. The technique demonstrated here simplifies these kinds of systems, allowing for complex interactive mechanisms to be simulated while meeting real-time requirements.

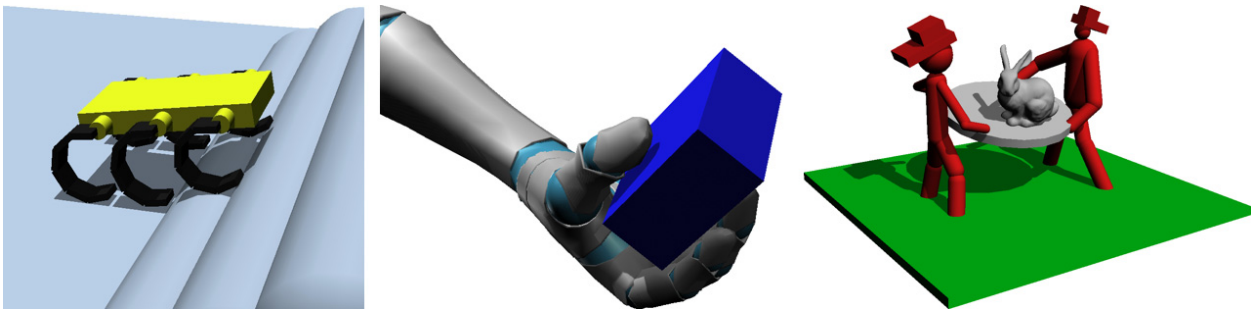

Figure 4-1: *Example simulations used to demonstrate the FORK^{-1}S technique include a multi-legged robot on uneven terrain, a humanoid robot grasping, and two firemen catching a bunny. The reduced model in the firemen scenario involves only one body, the trampoline, while the reduced models for the legged robot and hand involve multiple, coupled end effectors.*

4.1 Overview

Our approach is based on two main assumptions. The first assumption is that there are a limited number of surfaces at which our articulated systems experience contact. Thus, we can focus on the effective mechanical properties of a small collection of bodies in the system. This is a reasonable assumption for many scenarios, such as the wheel ground contact for a vehicle, the fingertips of a hand during grasping, or simulated tool-use by virtual humans and robots. The second assumption is that the multi-body system has a reference pose, which is held due to linear springs at the joints. This is certainly true for systems that have passive linear elastic joints, but also reasonable for virtual humans following the equilibrium point hypothesis of motor control, and in simulated robots using PD control.

In the case of a single interaction surface, our approach simplifies an entire system to a single 6D mass-spring system. When there are multiple bodies with surfaces experiencing contact, we use a collection of compliantly coupled bodies. We present an incremental algorithm for computing the dynamics model, which walks the body connectivity graph. The result is a system that is much simpler than the original, and is also stable and fast to compute. Note that we do not assume that the structure has the topological structure of a tree, as is necessary for fast computation in many alternative multi-body algorithms.

At simulation time, external forces produce a dynamic transient behaviour for the bodies that we include in the model. The configuration of all the remaining bodies is updated by computing a

compatible state. Rather than using IK, we compute linear maps that provide twists for each body as a function of the reduced system state and use the exponential map to compute the body positions. Thus, non-interacting body configurations can be computed independently and in a parallel fashion. Although the twists only model the linear response, we observe that the exponential map gives good behaviour. There is little separation at joints for an adequate range of interaction forces, and we discuss the size of the errors produced. Because the position updates can all happen in parallel, our method is well suited to parallel implementation on modern CPUs and GPUs. With new hardware from smart phones to desktop computers primarily gaining additional computation power through increasing core counts, we believe it is important for future algorithms to exploit parallelism, and we identify this separable computation of the internal state as one of the important contributions of our work.

Another important aspect in our work is that we have control over the fidelity of the physics simulation, and can dial it up or down as necessary. For instance, we can simplify a grasping system to just the fingertips of a hand in frictional contact with a grasped object. Alternatively, assuming no sliding or rolling at contacts, we can reduce the system to model the grasped body alone, which may be of interest in the simulation of a peg-in-hole insertion task. Likewise, if we know that additional contacts will occur at other bodies in the multi-body system, these can be included as interaction surfaces in our model.

The work presented in this chapter is an important new tool for simulating first-order reduced compliant systems, or FORK^{-1}S. The contributions of this work are summarized as follows:

- An approximate model for the simulation of complex kinematic structures;
- A model building process that uses a principled approach based on first-order projections of the dynamical behaviour of the system;
- Methods for efficient and parallel updates of the internal bodies;
- Concise recursive algorithms for constructing the FORK^{-1}S model, for which we provide pseudo-code;
- A demonstration of various important scenarios that show the utility of our approach and the models that we can produce.

4.2 Dynamics projection and notation

Our method targets mechanisms made up of articulate chains of compliant joints where external interaction occurs only with a small collection of bodies at the interface. We call these bodies the *end effectors*, following the terminology used in robotics literature. For simplicity, we will initially present our approach for the case where the base, or root, of the mechanical system is fixed in the world (the case of a free-body base link is discussed in Section 4.5). There are numerous simulation and animation applications where this type of configuration occurs, such as the previously described grasping and manipulation examples.

In this section, we explore the simple case of dynamics projection for a single body with one rotational joint, and provide a preliminary discussion of how we can incrementally build a projection for a complex mechanical system.

4.2.1 Projection for a single link

Consider the behaviour of a rigid body link x rotating about a hinge joint, as illustrated by Figure 4-2. For simplicity, we assume the hinge constraint is fixed in the global coordinate frame. The hinge constrains the motion of the body to a single degree of freedom and the admissible twists are rotations about the joint axis. Therefore, the twist of the rigid body $\xi \in \mathbb{R}^6$ is a function of the joint angle $\theta \in \mathbb{R}$, or $\xi = f(\theta)$. The Taylor expansion of $f(\theta)$ gives the approximation

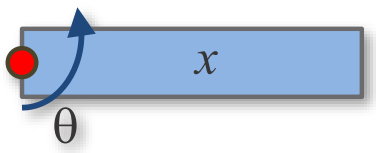

Figure 4-2: *The simplest mechanism: a single rigid body x rotating about a compliant joint with angle θ.*

$$\xi \approx f(0) + \frac{\delta f}{\delta \theta}\Delta\theta, \tag{4.1}$$

where $J = \frac{\delta f}{\delta \theta}$ is the Jacobian of the kinematic configuration of the body. Without loss of generality, let $f(0) = 0$, giving the first-order kinematic relationship $\xi \approx J\Delta\theta$.

Assuming a stiff joint, any displacement of the hinge from its rest or initial configuration will generate a torque about the joint axis. The torque τ and joint displacement $\Delta\theta$ of the hinge are related by

$$K_\theta^{-1}\tau = \Delta\theta, \tag{4.2}$$

where K_θ^{-1} is the compliance, or inverse stiffness, of the joint.

Body wrenches $w \in \mathbb{R}^6$ corresponding to joint torques are computed by the transpose of the Jacobian,

$$\tau = J^T w. \tag{4.3}$$

Combining Equations 4.2 and 4.3 and multiplying on the left throughout by J gives

$$JK_\theta^{-1}J^T w = \xi. \tag{4.4}$$

Here, $K_x^{-1} = JK_\theta^{-1}J^T$ is the effective compliance of the body x in spatial coordinates. The twist of the body ξ resulting from an applied external wrench w_{ext} is computed as

$$K_x^{-1} w_{ext} = \xi \tag{4.5}$$

and the homogeneous transformation of the body's displacement is computed by the exponential map.

It is convenient to use compliance to model the behaviour of the body in full coordinates because this compliance will be zero for motions not permitted by the joint. As such, K_x^{-1} is rank deficient, and a robust method for computing the matrix inverse is necessary to compute the stiffness K_x. Our work uses a truncated SVD (Hansen, 1990) to compute the inverse when needed.

We can perform a very similar projection of the rotational mass matrix M_θ. From the equation of motion $M_\theta^{-1}\tau = \ddot{\theta}$, and assuming the body acceleration $\dot{\phi} \in \mathbb{R}^6$ is approximately $J\ddot{\theta}$, we find $JM_\theta^{-1}J^T w = \dot{\phi}$, thus, $M_x^{-1} = JM_\theta^{-1}J^T$. We follow the same projection for the damping matrix to produce the second-order system

$$M_x\dot{\phi} + D_x\phi + K_x\xi = w. \tag{4.6}$$

Note that the static solution of this system exactly matches that of the original. Also, this example is more instructional than useful given that Equation 4.6 has 6 dimensions, but it is constructed from a 1D joint. Nevertheless, these projections are useful and a central part of the approach we

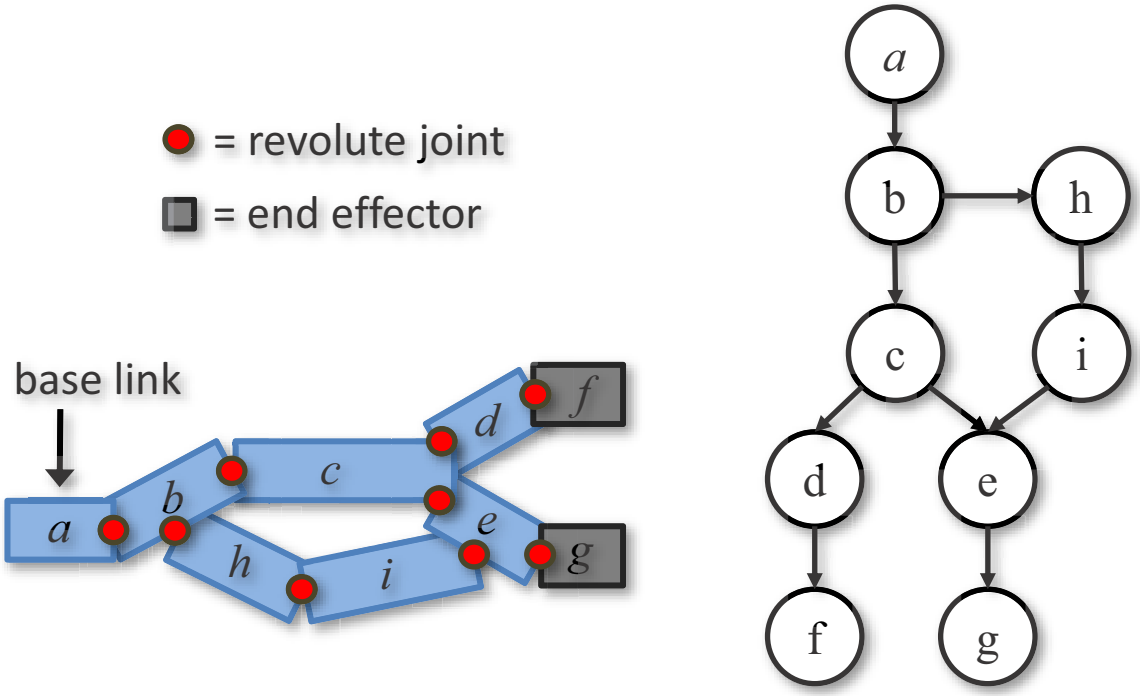

Figure 4-3: *The connectivity of bodies used in constructing the effective coupled stiffness, damping, and mass of end effectors.*

describe in Section 4.3. Specifically, the end effector will be part of a complex system of joints and rigid bodies. The effective compliance at the end effector x is due not only to the compliant behaviour of the directly attached joint, but also depends on the compliance of its parent (and the rest of the system). As such, we will describe an incremental approach for computing the effective stiffness, damping, and mass, at each link in an articulated system.

4.2.2 Notation

Throughout the rest of the chapter, we use different coordinate frames to explain our method. Preceding superscripts are used to denote the coordinate frame in which a vector is expressed. For instance, the velocity $\phi \in \mathbb{R}^6$ of body i in frame j is denoted by ${}^j\phi_i$. Likewise, a wrench acting on body i in frame j is denoted by ${}^jw_i$. The adjoint matrix ${}^k_j\text{Ad}$ maps the twist of a body in frame j to frame k, while its inverse transpose is used to change the coordinates of a wrench from frame j to frame k.

The joint structure of the mechanism has a dual representation as a directed acyclic graph (DAG), as seen in Figure 4-3. The graph's nodes are links (i.e., rigid bodies), and the edges correspond to

joint constraints, with the direction denoting the parent-child relationship. Specifically, the terms *parent link* and *child link* refer to the rigid bodies associated with the outgoing and incoming vertex of the edge, respectively. We use P_i and C_i to denote the set of parents and children of link i. Also, A_j is the set of ancestors of link j where branching occurs (e.g., see Figure 4-3), A_f contains c and b. Finally, let E denote the set of end effectors.

4.3 Incremental FORK^{-1}S construction

We incrementally build our approximated model by starting at the base link and working towards the end effectors. The process is simplified by examining three fundamental cases: *chaining*, *splitting*, and *merging*. In Figure 4-3 we can observe the three cases. Body b is a chained extension from body a. There exists a split at body b because both c and h are children. Finally, the only merge exists at body e, which has the two parents c and i. It is interesting to note that the parent-child relationship between bodies c and e can be set in either direction without affecting the final projection.

4.3.1 Chaining

The effective compliance of body b in the chaining case is the sum of the compliance of the parent link a and the compliance due to the stiffness of the joint between the two bodies, K_θ. Assuming the effective compliance of the parent link has already been constructed, it is straightforward to compute the effective compliance of the child link k. The twist-wrench relationship at link k is

$$K_{b,b}^{-1} = J K_\theta^{-1} J^T + {}^b_a\mathsf{Ad}\ K_{a,a}^{-1}\ {}^b_a\mathsf{Ad}^T, \tag{4.7}$$

where $K_{a,a}^{-1}$ is the effective compliance of the parent link, J and K_θ^{-1} are respectively the kinematic Jacobian and compliance of the common joint. Multiplying the compliance $K_{b,b}^{-1}$ by a wrench ${}^b w_b$ produces a twist, where the total twist is the sum of two parts: the twist due to the parent's motion and the twist due to the common joint motion.

Note that we use two identical subscripts to denote the compliance $K_{b,b}^{-1}$ because we want the motion of body b due to wrenches on body b. In the sections that follow, we will also need to

capture the motion of one body due to a wrench on another body in the system. Also notice that the compliance matrix could be written ${}^{b}_{b}K^{-1}_{b,b}$ to denote that it provides twists expressed in the coordinate frame of body b and must be given wrenches expressed in the same coordinate frame. We will drop these preceding scripts when the coordinate frames are clear due to context.

4.3.2 Splitting

In the case where two or more links share a common parent, their motion is coupled through their common parent. For a link with m children, the linear system determining the twist-wrench relationship of the child links is $\Phi = \mathsf{K}^{-1}\mathsf{w}$ where

$$\mathsf{K}^{-1} \equiv \begin{bmatrix} K^{-1}_{1,1} & \dots & K^{-1}_{1,m} \\ \vdots & \ddots & \vdots \\ K^{-1}_{m,1} & \dots & K^{-1}_{m,m} \end{bmatrix} \tag{4.8}$$

and $\Phi = ({}^{1}\phi^T_1 \cdots {}^{m}\phi^T_m)^T$, $\mathsf{w} = ({}^{1}w^T_1 \cdots {}^{m}w^T_m)^T$. In block matrix K^{-1} the diagonal block $K^{-1}_{i,i}$ is the compliance of child link i computed as described for the chaining case, while the off-diagonal blocks provide the coupling. That is, $K^{-1}_{i,j}$ determines the twist at link i due to a wrench applied to link j. To create these off diagonal blocks, an adjoint transform is used to first map wrenches in link j to the common parent k. The resulting twist is determined by the compliance of the parent, which is then mapped from frame k into the local coordinate frame of link i:

$$K^{-1}_{i,j} = {}^{i}_{k}\mathsf{Ad}\ K^{-1}_{k,k}\ {}^{j}_{k}\mathsf{Ad}^T. \tag{4.9}$$

Chaining additional links after a split is very similar. The diagonal blocks are updated as per Equation 4.7, while the off diagonal blocks are updated using Equation 4.9, where k is the *lowest common ancestor* of the two links i and j.

4.3.3 Merging

Unlike the cases of serial chain and splitting which work with compliances, merging uses the effective stiffness of the parent links. The effective stiffness is a parallel combination of the effective stiffness of the parent links, and includes the coupled stiffness due to a common ancestor.

Procedure 4-2 Recursive algorithm for computation of K^{-1}. Recursion is initiated by RECURSECOMPLIANCE(*base*).

```
function RecurseCompliance(i)
  if |Parents(i)| == 1 then
    K^{-1}_{i,i} ← Chain(i)                    // apply Equation 4.7
  else if |Parents(i)| > 1 then
    K^{-1}_{i,i} ←Merge(i)                     // apply Equation 4.11
  end if
  for j ∈ Children(i) do
    RecurseCompliance(j)
  end for
  if |Children(i)| > 1 then
    Split(i)                                   // apply Equation 4.9
  end if
end function
```

The compliant behaviour of link k results from multiple coupled chains attached to a single rigid body. For a link with m parents, we consider that the motion of k is the result of multiple superimposed versions of the link, with virtual link labels $1, \ldots, m$, and coupled compliance matrix K^{-1} computed with Equation 4.8. Let us now write the linear system describing the wrench-twist relationship using the stiffness as

$$\mathsf{w} = \mathsf{K}\Phi \tag{4.10}$$

where $\Phi = ({}^k\phi_1^T \cdots {}^k\phi_m^T)^T$ and $\mathsf{w} = ({}^kw_1^T \cdots {}^kw_m^T)^T$. Note that we use coordinate frame k for all blocks, and observe that ${}^k\phi_1 = {}^k\phi_2 = \cdots = {}^k\phi_m$ because the twist motion of all the virtual links must be identical. Also, the accumulation of wrenches equals the total wrench at link k, that is, ${}^kw_k = \sum_{i=1}^m {}^kw_i$. Therefore, the effective stiffness at link k is

$$K_{k,k} = \mathrm{I}\,\mathsf{K}\,\mathrm{I}^T \tag{4.11}$$

where $\mathrm{I} = (I \cdots I)$, and I is the 6×6 identity matrix. That is, the stiffness is the sum of all the blocks of the inverted coupled compliance matrix. The effective compliance at link k is computed by inverting $K_{k,k}$ using the truncated SVD method. As an aside, performing matrix inversion by a truncated SVD is used extensively in our work. The tolerance parameter for inclusion of singular values is tuned according to the stiffnesses of mechanism joints, and this is done by inspection.

Instead of creating the coupled compliance incrementally, starting from the base and moving out to the end effectors, we use the recursive approach outlined in Procedure 4-2, which combines the chaining, merging, and splitting techniques described in this section. This process is initiated by a single call RECURSECOMPLIANCE(a), where parameter a is the base link of the mechanism.

4.4 Wrench and twist maps

With the method described in the previous section, we can construct the coupled compliance, damping, and mass matrix of the end effectors. This allows us to simulate a reduced system consisting only of the end effectors. However, we still need to visualize the positions of all links in the structure. For this, we use the static pose twist of internal links as computed from a set of end effectors wrenches, and we call this the *twist map*. At a given time step of the reduced simulation, we compute wrenches that explain the current state, i.e., current twists, thus producing a compatible pose for the internal links.

In order to compute the *twist map*, we first describe the construction of a *wrench map* that distributes wrenches applied at the end effectors to the internal links. These maps are built incrementally. Our algorithm starts at each end effector and traverses the DAG of the mechanism in reverse order. First, a local wrench map is found that distributes wrenches from child links to their parents. Then, a global wrench map is computed by a compound matrix transform along the kinematic chain.

4.4.1 Local wrench map

Given the wrench at link k, the local wrench map may be used to compute the wrenches distributed amongst its parent links. Naively, we could simply divide the wrench by the number of parent links and compute the wrench to transfer to the parent links using the appropriate adjoint transforms. However, this division of force will not be correct because the wrenches transmitted down different chains will depend on the effective compliance of each chain. Thus we construct a linear system to ensure that wrenches are distributed in a plausible manner that respects the internal joint constraints and compliances.

Note that the sum of the wrenches at the parent links must equal the wrench applied at the child link k. That is,

$$ {}^{k}w_k = \sum_{i \in P_k} {}^{i}_{k}\mathsf{Ad}^T \, {}^{i}w_i. \tag{4.12} $$

Consider the case of m superimposed virtual links that was presented in Section 4.3.3. We can write the following constrained linear system to determine the distribution of wrenches on the different chains:

$$ \begin{bmatrix} \mathsf{K}^{-1} & \mathrm{I}^T \\ \mathrm{I} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ {}^{k}w_k \end{bmatrix}. \tag{4.13} $$

The constraint here is the same as Equation 4.12, except that all quantities are represented in frame k, and thus the adjoints are 6×6 identity matrices, i.e., $\mathrm{I}\,\mathsf{w} = {}^{k}w_k$. To compute a local wrench map that takes the wrench from k and divides it among its parents $1, \ldots, m$, we invert the system above, replacing the right hand side by a block column matrix that will provide the desired vector when right multiplied by ${}^{k}w_k$. That is,

$$ \begin{bmatrix} {}^{1}W_k \\ \vdots \\ {}^{m}W_k \\ * \end{bmatrix} = \begin{bmatrix} \mathsf{K}^{-1} & \mathrm{I}^T \\ \mathrm{I} & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{4.14} $$

This gives us a block column vector containing the wrench map for each parent link, with a block $*$ due to the Lagrange multipliers that we can ignore. Note that forming left hand side blocks ${}^{i}W_k$ requires computing the inverse of a system that may be rank deficient due to the coupled compliance. Again, we use a truncated SVD in its computation.

Finally, while these wrench maps only consider the difficult case of merging (multiple parents) by using superimposed virtual parent links, the transmission of wrenches along simple serial chains is easy. It simply involves a change of coordinates with an adjoint inverse transpose. For parent link a and child link b in a serial chain, the wrench map is simply

$$ {}^{a}W_b = {}^{b}_{a}\mathsf{Ad}^T. \tag{4.15} $$

Procedure 4-3 Recursive algorithm to compute all wrench maps ${}^{i}W_{e}$. Here, i is a mechanism link. Initiate recursion by calling RECURSEWRENCHMAP(e, e, I) for every end effector e, where I is the identity matrix.

```
function RecurseWrenchMap(i, e, iW_e)
  for j ∈ Parents(i) do
    jW_i ← LocalWrenchMap(i)          // apply Equation 4.14 or 4.15
    jW_e ← jW_i iW_e                  // apply Equation 4.16
    RecurseWrenchMap(j, e, jW_e)
  end for
end function
```

4.4.2 Global wrench map

The matrix ${}^{i}W_{k}$ gives a local mapping for wrench distribution between child link k and parent link i. Since the local wrench map only needs to be computed once, this makes it possible to construct a global wrench map for computing the wrench at internal links due to applied wrenches at the end effectors. Keeping with our scheme of incremental model building, we use the local map to compute the wrenches distributed to internal links due to wrenches applied at the end effectors.

Let ${}^{j}W_{i}$ be the matrix mapping wrenches from link i to its parent link j. The matrix mapping wrenches from end effector e to link j is simply the compound matrix transform of the wrench map for each body in the path $1, \ldots, n$ between e and j,

$$ {}^{j}W_{e} = {}^{j}W_{1} \left(\prod_{i=1}^{n-1} {}^{i}W_{i+1} \right) {}^{n}W_{e}. \tag{4.16} $$

By accumulating the wrenches due to all end effectors, the wrench affecting an internal link is

$$ {}^{i}w_{i} = \sum_{e \in E} {}^{i}W_{e} \, {}^{e}w_{e}. \tag{4.17} $$

We use a recursive algorithm to explore the DAG while computing the global wrench map for each link. Procedure 4-3 gives an overview of how the equations described in this section are used to build the maps.

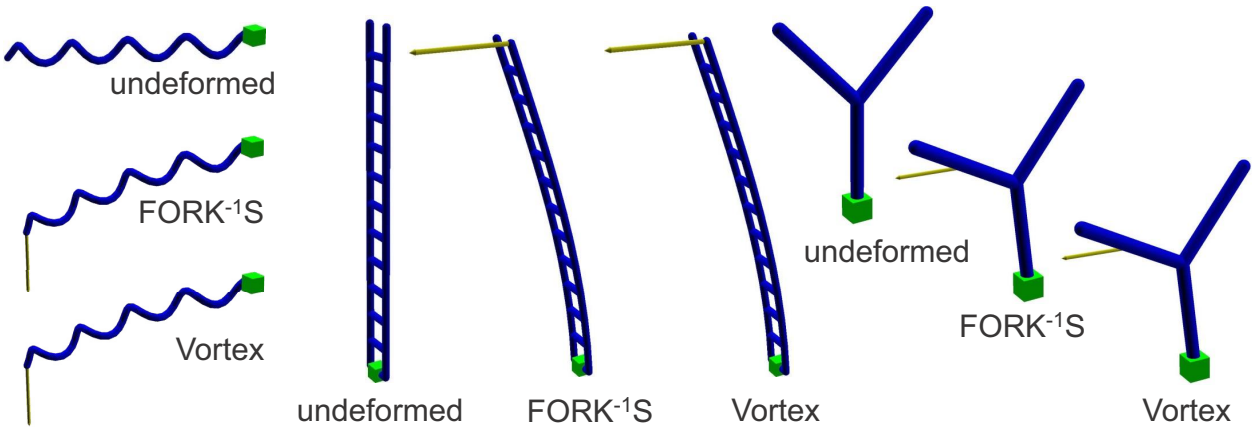

Figure 4-4: *Deforming a helix (left), spring ladder (middle), and "Y" mechanism (right). The rest configuration is shown, as well as a comparison between the static solution reached by simulation with FORK-1S versus a constrained rigid body simulator (Vortex). In each case, a* $100\,N$ *force is applied (yellow arrow); all joints use a stiffness of* $K = 1000$.

4.4.3 Global twist map

The twist map provides the static solution of the compliant joint chain due to wrenches applied at the end effectors. For a serial chain of compliant joints, the twist at an internal link i is computed as

$$ {}^{i}\phi_i = \sum_{e \in E} {}^{i}K_{i,i}^{-1}\ {}^{i}W_e\ {}^{e}w_e. \tag{4.18} $$

However, for more complex mechanisms, special consideration must be given to the coupled motion due to splitting and merging of the kinematic chain. The contributed motion of links that share a common ancestor with link i must also be considered, and the general version of the twist map in Equation 4.18 is

$$ \sum_{e \in E} \left({}^{i}K_{i,i}^{-1}\ {}^{i}W_e + \sum_{a \in A} {}^{i}_{a}\mathrm{Ad}\ K_{a,a}^{-1} \left({}^{a}W_e - {}^{i}_{a}\mathrm{Ad}^{T}\ {}^{i}W_e \right) \right). \tag{4.19} $$

The twist has a component due to the wrench arriving from each ancestor, but also experiences motion due to that of its ancestors influenced by end effector wrenches. The subtraction in the last term ensures that we do not include the motion of the ancestor induced by the wrench transmitted through the chain containing link i, because it is already accounted for in the first term (see Equation 4.18).

4.5 Dynamic simulation

We simulate the reduced dynamic system using a backward Euler formulation (Baraff and Witkin, 1998). As such, we have a system matrix of the form $\mathsf{A} = \mathsf{M} - h^2\mathsf{K} - h\mathsf{D}$. To solve this system with frictional contacts, we use an iterative projected Gauss-Seidel solver similar to the one described by Erleben (2007). This involves a Schur complement of the form $\mathsf{G}^T\mathsf{A}^{-1}\mathsf{G}$, where G is the Jacobian for the contact and friction constraints. We note that it is only necessary to invert the system matrix once, and reuse this small dense inverse system for the duration of the simulation.

The solution to the reduced dynamic system only provides the positions (twists, ξ) and velocities of the end effectors links. Internal links are updated using the twists of the end effectors. Specifically, we compute equivalent static end effector wrenches as $\mathsf{w} = \mathsf{K}\xi$, and then from these, compute the configuration of internal links using the twist map in Equation 4.19.

4.5.1 Free-body base link

For simplicity, the discussion above has let the body frame of the base link be fixed in the world. To extend the reduced model to allow for motion at the base, we integrate a second equation of motion for a rigid body representing the base. We set the base mass and inertia matrix to be that of the entire structure in the rest pose. The motion of the base is driven by gravity, but also by the net external wrenches applied at the end effectors. While the coupled equations of motion could be derived from the Lagrangian, we only couple the base and end effector models through external forces, and assume that the omitted terms such as Coriolis forces are negligible when the base has large mass and is moving slowly. In this case, the contact and frictional constraints must be modified to use the combined velocity of the base link ${}^b\phi_b$ and velocity of the end effector link ${}^k\phi_k$ in contact frame c, ${}^c\phi_{k+b} = {}^c_k\mathsf{Ad}\ {}^k\phi_k + {}^c_b\mathsf{Ad}\ {}^b\phi_b$.

4.6 Results

We have integrated our approach with the Vortex simulator. In this section we demonstrate the utility of our approach and the models we can produce in various scenarios of importance, such as simulation of amphibious robot walking, and simulation of human grasping, as seen in Figure 4-1.

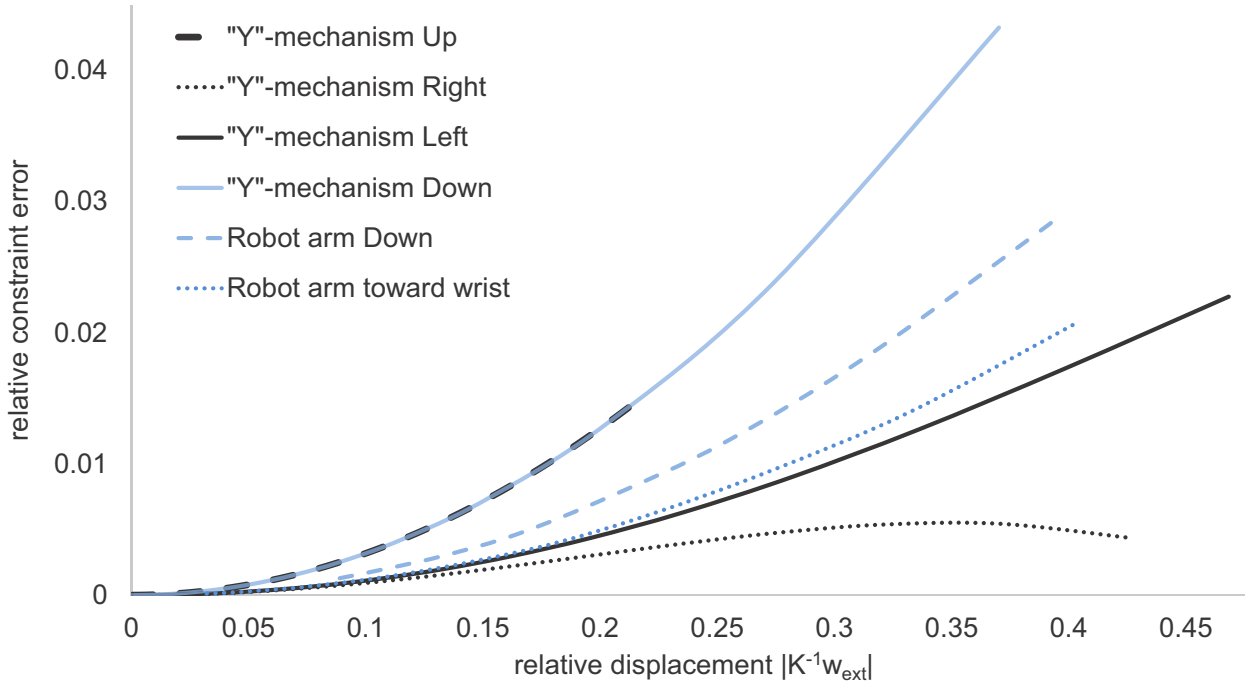

Figure 4-5: *The relative constraint error is measured for the "Y" shaped mechanism and the robotic arm as a single end effector is pulled in various directions. The vertical axis gives the constraint violation error, which is relative to the mechanism size. The horizontal axis gives the relative displacement of the end effector, which is also represented in proportion to the mechanism.*

Video results are available on the project page.[1] Note that all video results were obtained by FORK^{-1}S simulation, unless otherwise stated.

Figure 4-4 shows a comparison between simulations with our method versus a commercial physics engine. In each case, a constant force is applied at an end effector and simulated until a static equilibrium is reached. The final configurations are perceptually very similar between simulation with FORK^{-1}S and the standard Vortex physics engine.

Additionally, we have performed a number of experiments to explore how joint constraint violation errors grow as external forces are applied. The error at each joint is measured as the Euclidean distance between constraint attachment points of the body pairs, which are accumulated over all joints and scaled by the bounding sphere radius of the mechanism.

Figure 4-5 shows error plots for the "Y" mechanism and robotic arm as a single end effector is pulled in various directions. The relative error is computed as the position violation of the con-

[1] `http://www.cs.mcgill.ca/~sandre17/forks/`

Example	# links	Vortex	FORK^{-1}S		
			N=1	N=4	N=8
Helix	50	240	20	12	15
Helix	100	470	32	16	21
Helix	400	2150	112	84	76
Ladder	48	334	22	14	18
Robot arm	20	121	24	18	21

Table 4.1: *Mean computation time in μs per simulation step for various examples. Our method with N threads is compared against performing a full constrained rigid body simulation using the Vortex physics engine. Note that a 6-core Intel i7 processor with hyper-threading enabled was used to obtain these results.*

straint, accumulated over all joints and scaled by the bounding sphere radius r of the mechanism. This is plotted versus the relative displacement of the end effector, which is computed as the Euclidean norm of the twist with the linear component scaled by r and the angular component scaled by 2π. The rate of increase in the error is dependent on the direction of the applied force, but even for significant displacements of the end effector, the proportional constraint error remains low. Perceptually there is often no constraint violation, as demonstrated in the accompanying video. However, although the error remains relatively low, our method is based on a low-order approximation, and once the mechanism is sufficiently displaced from its initial configuration the error increases sharply.

4.6.1 Performance

Here, we compare the overhead of simulating a mechanism with a constrained rigid body physics engine versus the method outlined in this chapter. The computation times for solving the dynamical system in Section 4.5 and performing numerical integration is given in Table 4.1. Each mechanism listed in the table was simulated using a single threaded version of the Vortex physics engine, as well as our FORK^{-1}S implementation using different numbers of threads for the parallel update of internal links. Note that only moderate efforts were made to optimize our implementation.

The FORK^{-1}S method performed better in all cases, with the most drastic speedups observed when simulating long serial chains. Notably, there is a 28 times performance increase for the 400 link helix example. Also, as Table 4.1 suggests, there is a "sweet spot" in choosing the thread count for simulating a particular mechanism, with an increase of threads not necessarily giving better

performance.

One practical consideration that impacts performance significantly is grouping the internal links so that updates are performed in batches per thread. This avoids unnecessary context switching. Additionally, the associated data structures are stored contiguously in memory in order to minimize memory thrashing issues.

4.7 Discussion

When large external forces are applied, constraint errors result in the interior body reconstruction. These errors can be fixed by allowing rigid bodies to stretch, but there is a limit to how large external forces can grow before geometry modifications are visible. As such, we have imagined the addition of a model to reduce compliance as the system is pulled from its rest state. This would help address the fact that the structure should become stiffer as singular configurations are approached, and would help us approximately model geometric limits of the internal joints. We plan to implement such a non-linear compliance scaling method in future work.

Adaptively stiffening the system would help keep the state closer to the rest configuration, avoiding states where joint constraint violations would be visible. Nevertheless, we also note that it is possible to make small modifications to the geometry to correct the error at the expense of letting rigid links deform. Such a strategy has been used in repairing foot skate (Kovar et al., 2002), and such length changes are often not perceived (Harrison et al., 2004).

We note that the behaviour of our reduced model can differ from the full model. In general, we observe the reduced systems to be slightly stiffer than their fully simulated systems. This is not surprising, and we believe this occurs naturally due to the lower number of DOF and the linearization we impose. Higher levels of damping seen in the reduced system can be explained by our implicit integration, while Vortex uses a symplectic integration scheme.

Finally, the construction process assumes that we can walk from a base node in the graph to all end effectors. When there are loop closures between two end effectors that are on the far side of the graph from the base, the incremental algorithm will not find them. An alternative projection technique is necessary in this case.

Chapter 5

Fingertip appearance models for interactive hand simulation

Chapters 3 and 4 describe methods for controlling and simulating physically based hands, and other similar articulated mechanisms. These methods are useful for synthesizing motion at real-time frame rates. However, though the motion appears plausible, the hands still do not appear realistic. Even if a skinned hand model with high-resolution textured geometry is used, this does not effectively portray fundamental visual aspects associated with grasping and manual interaction. For instance, colour changes due to contact.

In this chapter, we present our method for synthesizing such colour changes; the visual cues associated with hand-object interaction. Simulating these colour changes better reflects the nature of hand-object interaction, and increases the richness of interactive character animations.

5.1 Overview

Digital characters have become increasingly realistic over the years. Skinning techniques have been developed to better model the deformation of skin; physically based simulation of garments has improved realism by providing valuable secondary motion to a character's movement; and control strategies have allowed for natural simulated motion that reacts to environmental changes

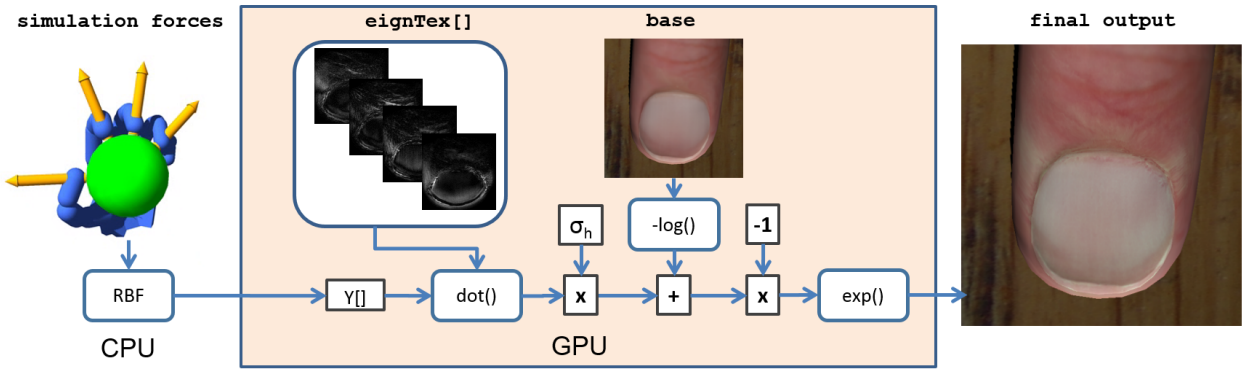

Figure 5-1: *An illustration of the texture modification process used by our method for appearance modeling of interaction forces.*

in a plausible manner. Control, deformation, and appearance can make use of procedural techniques, physics simulation, or data-driven models. Physically based deformation and rendering, in combination with anatomically detailed models, have become a popular approach for improving the quality of images and animation. We follow the alternative, which is to use captured data to create empirical models that replicate the behaviour of real-world example measurements.

While motion is central to the problem of character animation, and deforming geometry is a primary factor explaining overall appearance, colour variations across the character's skin are essential for rendering life-like images. Texture and bump mapping are now standard, while spatially varying surface scattering distribution functions and light transport approximations are becoming more common for highly realistic image production. In some cases, reflectance or scattering models can be parameterized. For instance, varying amounts of perspiration can drive glossiness to permit simple appearance changes, while it is also possible to model sub-surface blood variations due to pose, pressure, alcohol consumption, or exercise.

This is where our work fits in, as we are modeling colour changes due to the contact forces available in an interactive simulation. Our approach to modeling these colour changes is with a simple scattering model. Figure 5-1 shows an overview of our appearance modification process. A grasping motion controller and multi-body physics simulation is used to demonstrate the results of our method. The contributions of this work are summarized as follows:

- A simple capture process for collecting fingertip photos and finger pad pressure images;

- An analysis method that uses a custom, simplified hemoglobin concentration model;
- An efficient appearance model implementation within a fragment program, which integrates easily with a real-time interactive physics-based simulation of grasping control.

5.2 Data capture

Our capture process involves collecting photographs and force measurements of fingertips that are applying different amounts of pressure. We use a TekScan Grip system for measuring contact pressure at a fingertip. This system has a resolution of 6.2 *taxels* per square centimeter, where a taxel is a "tactile pixel" or "tactile element" within a pressure image. At the fingertip, the pressure image resolution is 4×4, giving a total of 16 taxels. We multiply the pressure value at each taxel by the area of an individual taxel, and sum up the force contributions of all taxels under the fingertip to produce a total force estimate. We also compute a center of pressure, but we currently only investigate the appearance change associated with forces centered on the finger pad. We have collected data with varying center of pressure to be used in future work.

We capture images with a consumer camera. Because we use a makeshift diffuse light box (consisting of David laser scanner calibration panels), we assume that the images effectively capture the albedo variation of the skin and fingernail. The camera white balance is set so that we can compute reasonable hemoglobin concentration estimates (see more details in Section 5.3).

Figure 5-2 shows our capture setup along with a sample of the captured pressure data. We have captured data for index and ring fingers of the right hand across three subjects. The process starts with the capture subject placing the distal finger pad of the selected finger at the center of the fingertip taxel grid. The capture subject interactively adjusts the positioning of their finger by applying a small amount of perpendicular force to the sensor surface. The fingertip position is adjusted until the center of pressure is aligned with the center of the sensor grid. Once capture begins, the finger remains in contact with the sensor, and if finger slippage occurs, the process is restarted. The back of the finger remains visible to the camera as the subject explores different pressure magnitudes (as well as varying the center of pressure in our extended future work data set). Pressure and image data are captured simultaneously. The interactive nature of the capture software

Figure 5-2: *Left, makeshift diffuse light box and TekScan capture setup. Right, example pressure data showing center of pressure.*

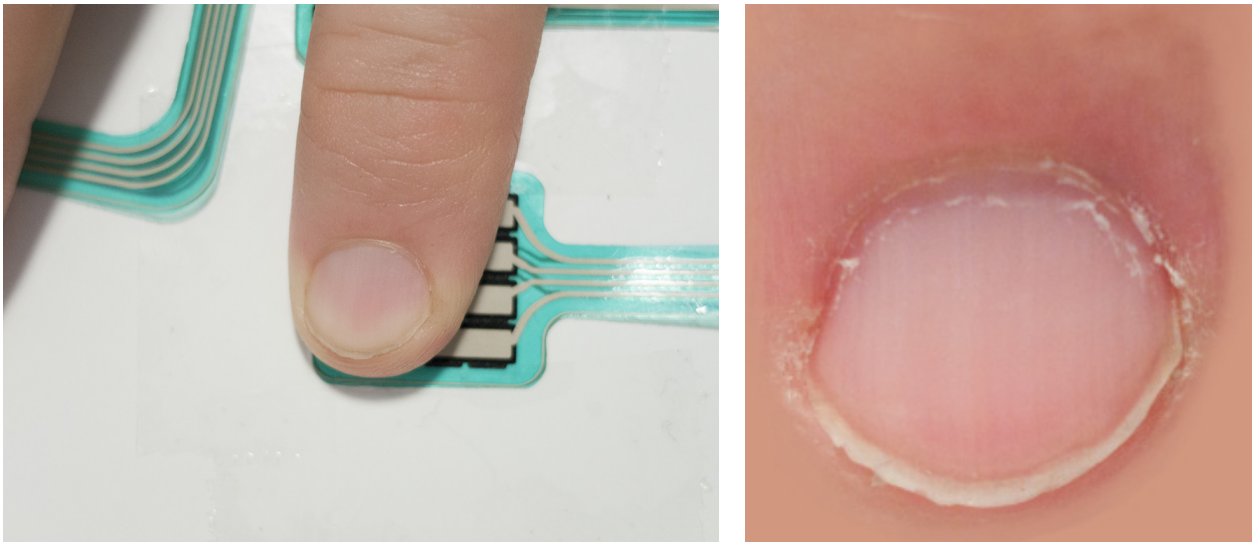

Figure 5-3: *Left, an example raw captured image, and right, the cropped and masked pre-processed image ready for analysis.*

allows the subject to avoid collecting multiple similar examples, which could be problematic for over-fitting during model construction.

Because the fingertip may translate and rotate slightly between different photographs when applying different amounts of pressure, the example images need to be warped into a consistently parameterized texture image (note that rotation is more pronounced in examples that involve vary-

ing centers of pressure). In particular it is important to match the shape of the fingernail. While it would be possible to develop a custom automated process to warp the images, we choose to do this manually for the examples that we collect (i.e., 10 samples used for our data-driven model). This pre-processing task is not onerous, and ultimately we observe that the estimated model for one fingertip is sufficiently general to apply to all fingertips and across individuals (validation of this can be found in Section 5.5). We use the Puppet Warp tool in Photoshop to deform the images, and apply a mask to avoid including any of the fingertip silhouettes and background in the analysis. Figure 5-3 shows an example photograph, along with a pre-processed example image which is ready for analysis.

5.3 Model fitting

While Mascaro and Asada (2004) note that finger posture also plays an important part in fingertip appearance, we focus solely on contact forces. Thus our appearance model can be seen as a function mapping a force on a finger pad to colour variations in the texture. Given that these colour changes are largely driven by the distribution of blood in the fingertip, we choose to do our regression in the space of hemoglobin concentrations.

In this section, we first explain our model for varying concentrations of hemoglobin, and the effect that it has on light scattering. We then describe how to estimate the absorption properties of hemoglobin in our images, and from there, how we work with hemoglobin concentration changes across the fingertip images. Finally, we describe an interpolation function that allows us to reproduce the collected examples.

5.3.1 Material concentration and light absorption

We assume a simple model for light scattering within human skin following previous work (Tsumura et al., 1999; Jensen et al., 2001; Tsumura et al., 2003). The transmittance T of light through a material at a given wavelength λ is defined as the fraction of the exiting spectral radiance $L(\lambda)$ over the initial incident radiance $L_0(\lambda)$. The Lambert-Beer law models transmittance as a function of the distance d that the light travels within the material, and the molar absorptivity and concentration

of a material $\alpha(\lambda)$ at the given wavelength. Specifically,

$$T = \frac{L(\lambda)}{L_0(\lambda)} = 10^{-\alpha(\lambda)d} \in [0, 1]. \tag{5.1}$$

Since we are using RGB images, we will take the convenience of defining $\alpha \in \mathbb{R}^3$ as a three-tuple of values corresponding to red, green, and blue wavelengths. When there exists a mixture of materials, we can write α as a sum of contributions from each of the materials. For instance, in our case it is useful to write the absorption as the sum of two parts, the first due to hemoglobin, α_h, and the second due to other constituents, α_o, such as melanin and keratin.

In the context of a diffuse reflection, we assume that our RGB images do not contain any lighting variation due to geometry, and in one sense, we can view the pixel values as varying surface albedo, which can be used as the diffuse material parameter in a lighting program. Here, we assume this per pixel reflected light percentage can be treated as transmission along a sub-surface light path of unknown fixed length within a homogeneous material. We further assume the light path d to be constant across all pixels. Thus, the negative logarithm of the pixel components gives us a proportional absorption property of the material mixture recorded at that pixel. That is, given a pixel's colour as $(R, G, B) \in [0, 1]^3$, we compute

$$\alpha = (-\log(R), -\log(G), -\log(B)) \in [0, \infty]^3. \tag{5.2}$$

Because the molar absorptivity of a material is constant, we factor α into a normalized *pigment* vector and a scalar *quantity*. Thus, we model colour changes in log space as changes in material quantity, and at any given pixel we can write

$$\alpha = q\sigma_h + \alpha_o, \tag{5.3}$$

where $q \geq 0$ is the hemoglobin quantity, $\sigma_h \in \mathbb{R}^3$ is the hemoglobin pigment, and α_o models the absorption of all other materials. We expect q at a given pixel to change depending on the distribution of blood in the fingertip, while the concentration of other materials, and in turn the absorption, remains constant.

5.3.2 Hemoglobin pigment estimation

With the white balance of the camera set correctly, our images will have colours appropriate for use in a texture map. However, because the illumination spectrum and camera response curves are unknown, the observed pigmentation of hemoglobin may not be fixed and must be estimated from the captured data. This is relatively straightforward because we have numerous sample images in which the only changes are due to the redistribution of blood in the fingertip tissues as different pressures are applied to the finger pad. Thus, the negative log colour values of different images at a given pixel should only vary in the direction σ_h.

We estimate σ_h as the first principal component of the negative log colour of all pixels in all images. Each pixel is centered independently to account for the varying amounts of other materials (i.e., α_o) across the fingertip. We use the negative log colour of the pixel in the zero pressure image as the center, though the mean negative log value would also be suitable (or likewise, that of any example image).

Note that the vector σ_h will have all positive components because absorption at different wavelengths is positively correlated (i.e., a material can only absorb more of any given light wavelength when its concentration is increased). Averaging across three data sets for three individuals, we estimate $\sigma_h = (0.19, 0.77, 0.61)$, which corresponds to higher absorption of green and blue wavelengths in comparison to red. Section 5.5 discusses estimates and validation further.

Letting σ_h be normal length, we can compute the hemoglobin quantity for any pixel in any image with a simple dot product,

$$q = \alpha \cdot \sigma_h, \tag{5.4}$$

while the residual provides absorption due to other materials,

$$\alpha_o = \alpha - q\sigma_h. \tag{5.5}$$

We compute hemoglobin quantity-change images for all of the examples to permit analysis and modeling (see Figure 5-4). While the pigment estimation is not sensitive to the center, we choose the zero pressure image as the center for the example hemoglobin quantity-change images. There-

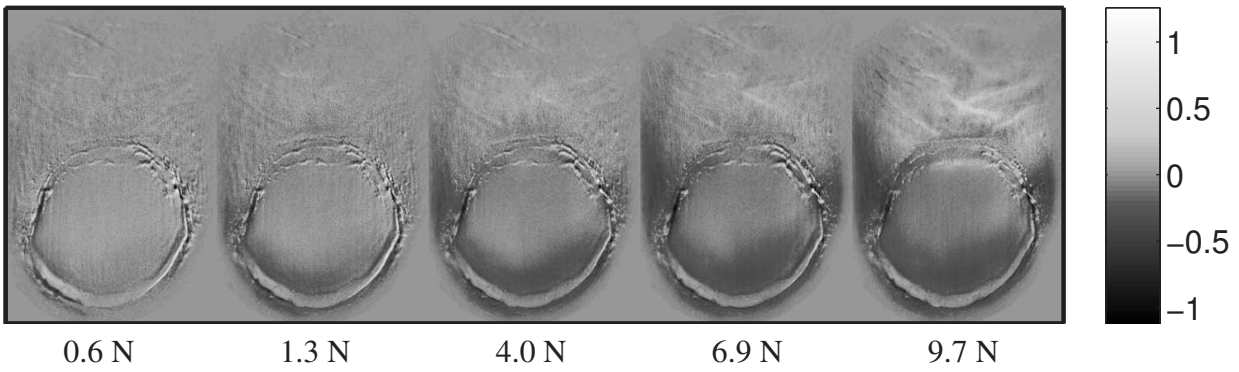

Figure 5-4: *Example quantity-change images showing considerable correlation across examples at different non-zero finger pad forces.*

fore, the quantity has zero change when there is zero force on the finger pad. We do not store the α or residual α_o vectors as they can easily be computed on the fly in the shader program as described in Section 5.4.

5.3.3 PCA model of hemoglobin quantity changes

Because there is considerable correlation among the hemoglobin quantity-change images that we observe with the different finger pad forces, it is appropriate to use PCA to compute an efficient basis to represent these changes in our example images.

There is camera noise in our example images, and because blood redistribution does not happen at the spatial frequencies captured by our full resolution example images, we down-sample the images to 256×256. Furthermore, we have considered applying a Gaussian blur to reduce camera noise, but this is not essential as we have not observed noise as having a significant effect on the first few principal components.

Figure 5-5 shows the principal component basis, along with the variation explained by the different basis vectors. We call these basis images *eigen textures*. Note that with just the first two we can capture over 94% of the variation observed in our examples.

Figure 5-6 shows reconstructions of the example RGB images using a reduced basis with only 4 components. The negative log colour image is modified by adding σ_h scaled by the per pixel quantity change computed with the truncated eigen texture basis. The result is negated, exponentiated,

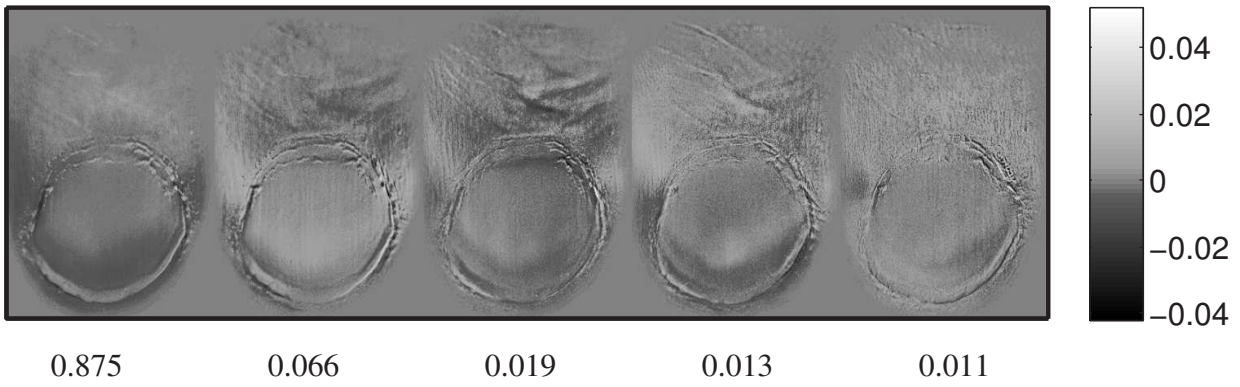

Figure 5-5: *The principal component basis vectors (i.e., eigen textures) explaining hemoglobin quantity-change across images. Below each is a value corresponding to the variation explained by each vector.*

and finally clamped to $[0, 1]$ to produce the reconstruction RGB image. Note that clamping is only necessary for extreme modifications and is typically unnecessary.

We are effectively modeling a process of appearance change due to blood moving about in the fingertip. As such, it would be ideal to use a data interpolation technique that appropriately takes into account mass transport (Bonneel et al., 2011). In the interest of having an inexpensive interpolation, we instead choose to model blood distribution variation in a principal component basis. The consequence is that we cannot do any better than blending examples, and if we do not have enough examples, then our results will exhibit less prominent differences from either the average or zero force appearance.

5.3.4 Interpolation

At this point, we have a collection of N examples consisting of a measured total finger pad force in newtons, $x \in \mathbb{R}$, and a hemoglobin concentration change that is represented in a truncated PCA basis, $y \in \mathbb{R}^M$. We use $M = 4$ because it is both sufficient and maps easily to graphics hardware.

We use a quadratic polyharmonic spline to interpolate the PCA coordinates at the sampled forces,

$$y(x) = \sum_{j=1}^{N} w_j \phi(\parallel x - x_j \parallel), \tag{5.6}$$

where $\phi(r) = r^2 ln(r)$ and x_j is force data for the jth sample. The weights w are computed

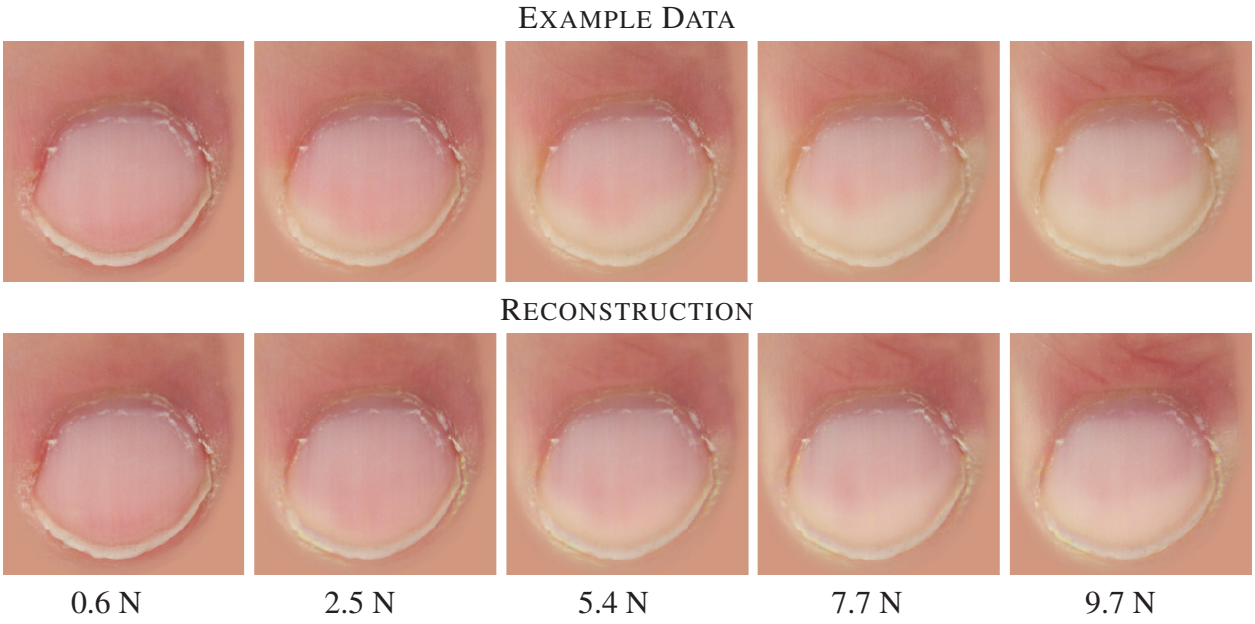

Figure 5-6: *Top row shows processed example images, while the second row shows reconstructions using 4 principal components. Note the whitening at the sides of the fingertip and under the nail at the tip as the pressure increases due to blood being squeezed out, into surrounding tissues. Also, note the similarity between examples and reconstruction.*

easily with a linear solve. We use this RBF interpolation approach because it will extend easily to interpolate other important features, such as the center of pressure on the finger pad, and the posture of the distal joint. In preliminary work in this direction, we noted that care is necessary in defining the distance metric for x such that it is meaningful. Ranges for different components must be taken into account to allow for valid comparisons. For example, position and force values need to be scaled to a comparable range, as do the joint angles of the finger if they are included.

Figure 5-7 shows plots of how well and how smooth the interpolations fit the data. The coefficients of the higher index principal components are noticeably smaller. This is because the basis vectors are all unit length, and the higher index vectors only capture a small portion of the variation.

5.4 Reconstruction implementation

In this section, we describe the reconstruction of the fingertip appearance in the context of rendering our full hand model. Thus, we will address how we pack information into textures and implement the reconstruction process in a fragment shader.

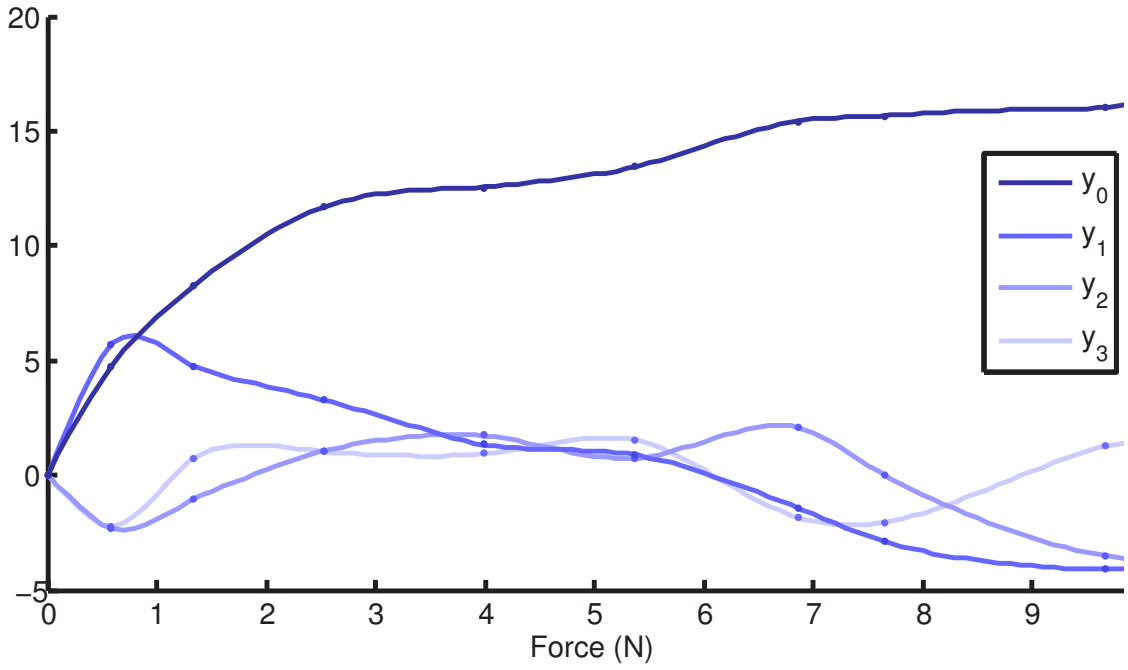

Figure 5-7: *Interpolation of the examples, shown with dots, in the coefficients of the PCA basis.*

Our model is suitable for any skinned hand rig. Preliminary experiments used a hand model exported from DAZ Studio. Our current work uses a hand model purchased from TurboSquid, with higher quality textures. Both models include linear transform blending weights. We use the FBX format for the convenience of being able to load and display the models easily within OpenSceneGraph.

5.4.1 Textures

In order to modify the colours of the diffuse texture map, we create a second texture map shown in Figure 5-8. We name this texture `stiaTex` to denote the contents of its four components: `s` and `t` texture coordinates, a finger index `i`, and an alpha mask `a`. The red and green channels contain the texture coordinates for the fragment's corresponding location in the pre-processed example image (to access the eigen textures for computing the fragment's hemoglobin quantity change). The blue channel contains an index i to identify the fingertip to which the fragment belongs (this allows us to look up the appropriate finger's PCA coordinates which were computed from the contact force).

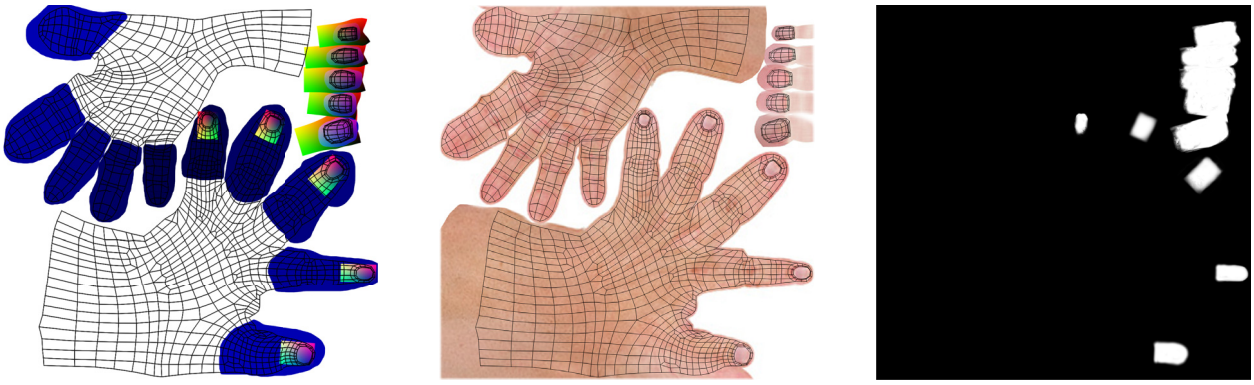

Figure 5-8: *Textures used in computing varying fragment colours at the fingertips. Left shows the* `stiaTex`*, with texture coordinates in red and green for looking up hemoglobin concentration changes, and the finger index encoded with different shades in the blue channel. Middle shows the base texture and the alpha mask with feathered edges used to blend the base texture with colour values modified due to finger pad forces is shown on the right.*

Finally, the alpha channel contains an alpha mask to allow a smooth blend between the modified colours and the base texture at the edges of the fingertip.

Note that the alpha component is shown separately at the right of Figure 5-8 to make it easier to see. Likewise, we overlay black lines to show the mesh coordinates within the texture to display how the texture content relates to the mesh geometry. Obviously, our run time textures do not contain these black lines. Furthermore, note that these mesh lines reflect the coarse level resolution of the mesh, while we use a subdivided mesh for better quality at run time.

The index information is easy to paint into the `stiaTex` texture image. However, a bit more work is involved to set the red and green channels to the appropriate texture coordinates. The process we use is to first prepare a red-green texture gradient identity map, and overlay this with our pre-processed fingertip example image. We then paste the red-green texture coordinate image, along with the zero pressure example image in a coupled layer, into the `stiaTex` image. The pasted layers must first be scaled, translated, and rotated to roughly match with the back of a fingertip in the base texture. We then use the Photoshop Puppet Warp tool to carefully deform the fingertip example image to match features in the base texture and mesh geometry. Once we have a good match, the warped red-green gradient can be *flattened* into the red and green channels of the `stiaTex` image, and we repeat the process for all fingers.

The alpha mask is not difficult to produce. We partially automate the process by using the eigen textures to identify where there are important changes in hemoglobin quantities. At each pixel, we sum up the absolute values of the first four eigen textures, and normalize to produce the mask shown in Figure 5-9. This is helpful because the pre-processed example images include areas which were masked and filled with a neutral skin texture to avoid the silhouette edges and background, and using this automatic alpha mask as a guide ensures that we exclude these regions in our blend. Ultimately, we edit this to produce the final alpha mask (e.g., we apply threshold, erosion, blur, and custom per finger modifications). The alpha mask is warped and flattened into the `stiaTex` image in the same process used for the red-green texture coordinate gradient.

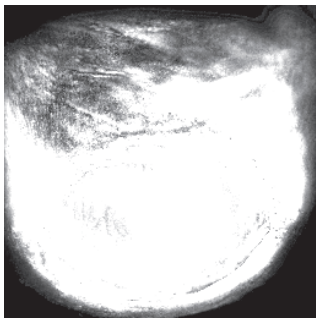

Figure 5-9: *The automatic alpha mask computed from eigen textures.*

While we could let our model produce colour changes on the base texture, we choose to replace the base texture with the warped zero pressure example image. This typically involves a small HSV space colour correction to the base texture to have it match our white balanced example images. We use the alpha mask generated above to smoothly blend the pasted example into the base texture.

Computing hemoglobin changes involves a chain of two texture lookups. This is a useful alternative to defining a new set of texture coordinates over the surface of the mesh. It allows for distortions in the mapping at the resolution of the texture, as opposed to the resolution of the mesh. This can be useful for closely matching small features, for instance, the length of fingernail may be longer or shorter in the geometry in comparison to our pre-processed images.

We compute hemoglobin changes using the first four eigen textures shown in Figure 5-5. These can be assigned to different texture units, but when only four are needed, it is efficient to pack them into the red, green, blue, and alpha components of a single floating point texture image. Note that in the previous section we use Matlab to compute the principal component analysis, and export the eigen textures as floating point tiff images, which are easily loaded into texture memory using OpenSceneGraph.

5.4.2 GLSL fragment program

Most of the work of modifying the appearance of fingertips is done in a fragment program. Using the total force on each finger pad, we evaluate the interpolation function in Equation 5.6 to compute the PCA basis coefficients for each finger's hemoglobin quantity change. We have 4 values for each fingertip. Thus, we compute a total of 20 values and store the results in a `vec4` GLSL uniform array. The GLSL fragment program code necessary to compute the appearance modification is quite small and is provided in Procedure 5-4.

To summarize the process, it begins with a base colour and `stia` value texture lookup, followed by a third texture lookup of the PCA basis at texture coordinates `st`. The negative log colour of the base texture is computed, and then modified in the direction of σ_h by the hemoglobin concentration change. Note that in our implementation, σ_h is stored as a constant global variable, or *uniform*, and only needs to be set once. The hemoglobin concentration change is computed with a simple dot product of the PCA coordinates and the four PCA basis vector components associated with the fragment. Note that some care is necessary when building the `stia` texture to ensure that the blue component can be correctly cast to an integer. The final diffuse colour is computed by exponentiating the modified negative log colour. The colour is clamped to $[0, 1]$, and blended with the base colour using the alpha mask.

5.4.3 Grasping control and simulation

The hand grasping animation is simulated with the Vortex real-time physics simulation software. We use a variety of simple PD feed-forward control trajectories to produce animation, as well as a grip strength controller that works from a single grasping pose based on the method of Liu (2009).

Our controller permits the computation of forces and torques at the fingertips that allow for a simple modulation of grip strength. From the forces at each contact, we compute control torques at the joints by multiplying with the Jacobian transpose (Murray et al., 1994b). This lets us set a reference pose for a proportional derivative controller. We use reasonable joint stiffness and damping values, which are set using the object-in-hand drop test suggested by Pollard and Zordan (2005).

Procedure 5-4 GLSL fragment shader code snippet summarizing the necessary additions to a standard per fragment lighting program.

```
// Fragment program
sampler2D baseTex;
sampler2D stiaTex;
sampler2D eignTex;
varying vec2 tex0;
uniform vec4 y[5];
uniform vec3 sigma_h;

vec3 base = sampler2D( baseTex, tex0.st );
vec4 stia = sampler2D( stiaTex, tex0.st );
vec4 U03  = sampler2D( eignTex, stia.st );
vec4 y03  = y[int(stia.b)];
float dq = dot( y03, U03 );
vec3 alpha = -log( base );
alpha += dq * sigma_h;
vec3 base2 = exp( -alpha );
base2 = clamp( base2, 0, 1 );
float mask = stia.a;
diffuseColour = mix( base, base2, mask );

// per pixel lighting follows...
```

When there is one or more contacts at a finger pad, we sum up the force contributions, and use the magnitude in the PCA coordinate interpolation functions. We note that even when the simulation's collision restitution is set to zero (i.e., no bounce), there can be a spike in the contact force which results in a brief white flash on the fingertips. In reality, the appearance changes should be gradual due to the fact that blood must physically move within the fingertip. Therefore, we add viscosity in the form of an exponential decay, which we apply to the force magnitude (conceptually it would make more sense to do this on the PCA coordinates, but there is no difference in our model). Our simulation runs at 100 Hz, and we note pleasing results by blending 0.95 of the previous time step's force magnitude with 0.05 of the current.

5.5 Validation and results

In this section, we discuss three types of validation that we have performed on our fingertip appearance modeling efforts: leave one out cross-validation of interpolated reconstructions, model

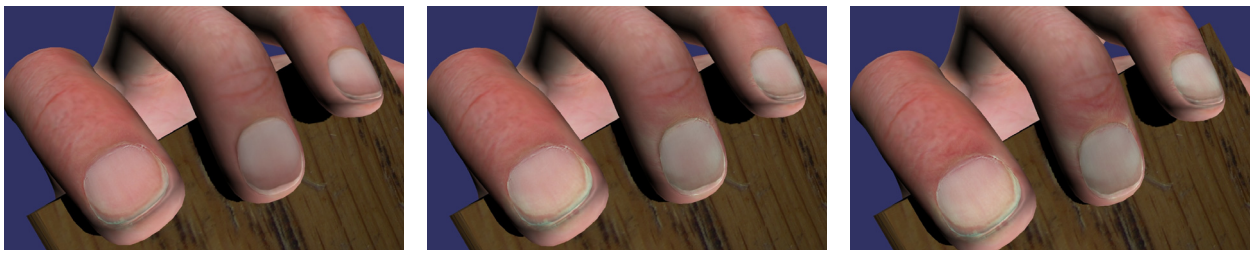

Figure 5-10: *A simulated grasping sequence where grip strength starts low and increases. Note the whitening of the nail while tissue near the cuticle takes on a more prominent red colour.*

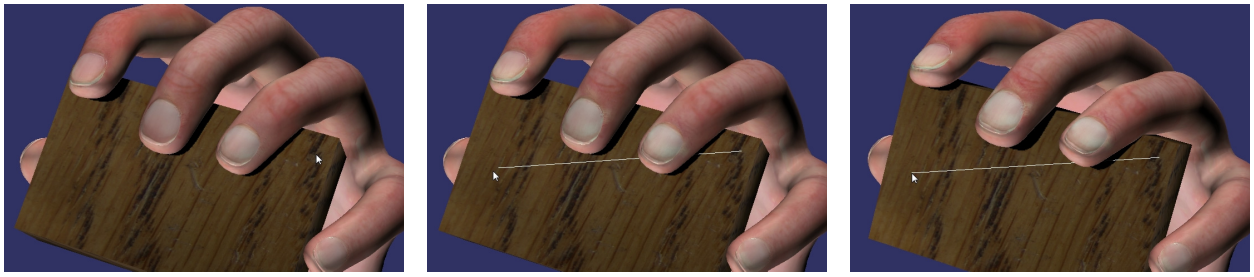

Figure 5-11: *A simulated interaction where a box can be pulled away from a grasp using a spring attached at the cursor location. Note the colour changes at the middle and ring fingers that subtly indicate increased contact forces.*

transferability between fingers and subjects, and verification of hemoglobin pigment consistency. We also show results from interactive simulations of grasping. Complete animations and video sequences are available on the project page.[1]

5.5.1 Cross-validation

Figure 5-12 shows a comparison of images synthesized with interpolation compared to pre-processed example images that were left out of the data analysis and fitting. The two examples show that our method successfully interpolates examples involving both small and large contact forces with little error. We compare our method to the naive solution, which simply uses the base fingernail image for all interaction forces. The absolute error is computed as the mean squared difference in RGB values of the validation and synthesized images. The relative error, computed by a ratio of absolute errors, is 0.6559 at 0.18 N, giving just a moderate improvement over the naive method. However, at larger interaction forces, the benefit of our method becomes clear, as the relative error drops to 0.1183 at 9.25 N.

[1] `http://www.cs.mcgill.ca/~sandre17/fingertip/`

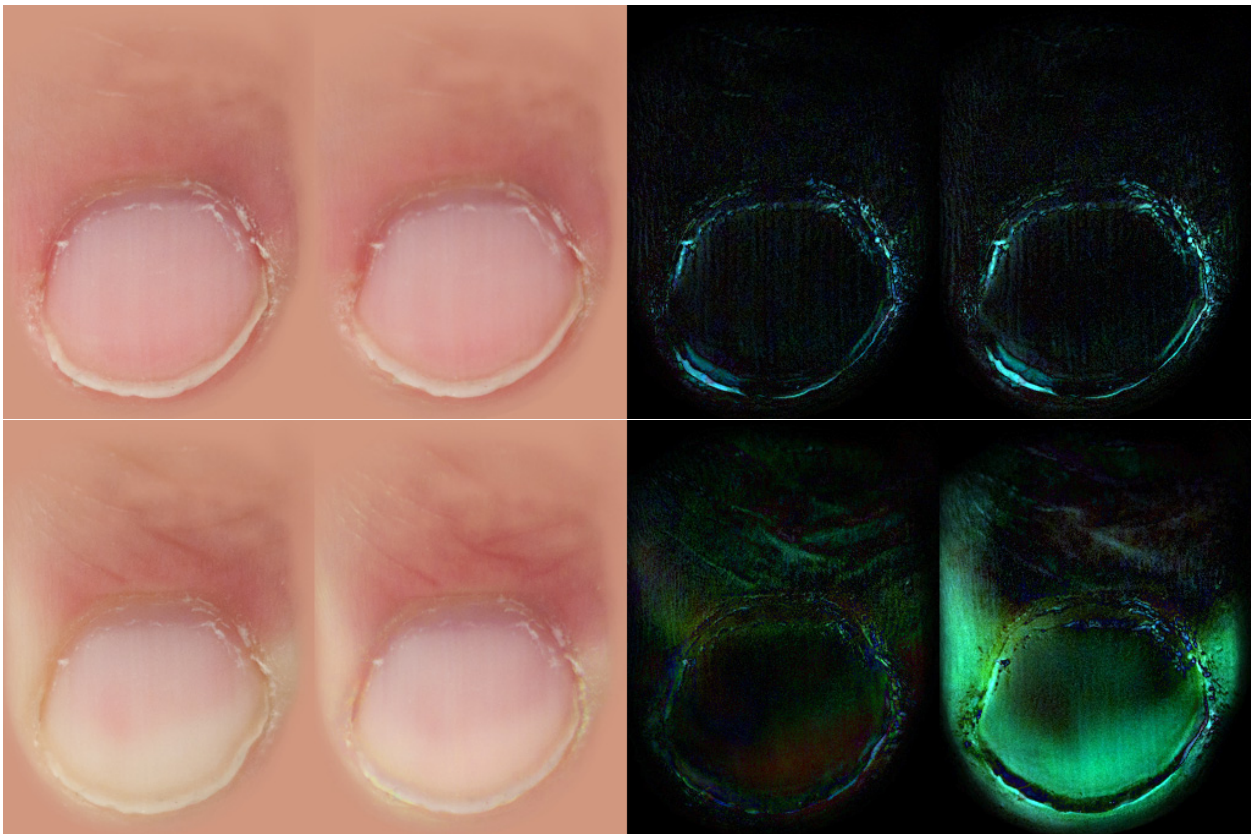

Figure 5-12: *Leave-one-out cross-validation of our synthesis approach with testing examples at 0.18 N (top) and 9.25 N (bottom). From left to right are the example images, synthesized images, the RGB error magnitude of the synthesized image, and the RGB error of the naive solution (errors scaled* $4\times$ *to reveal detail).*

5.5.2 Model transferability

To evaluate the transferability of our model across fingers and subjects, we performed the model fitting steps on three separate sets of 10 sample images gathered from three subjects. Figure 5-13 shows the first principal component from each set. These first eigen textures share qualitative similarities, with most changes occurring toward the distal part of the fingernail, and in the skin to the left and right. The Pearson correlation coefficient of the first eigen texture between the first two subjects is 0.6033, indicating a moderate correlation, but note that these two eigen textures do not share a common parameterization. For instance, we would expect a stronger correlation if image warping was applied.

We have also evaluated the constancy of our hemoglobin pigment estimation across different sub-

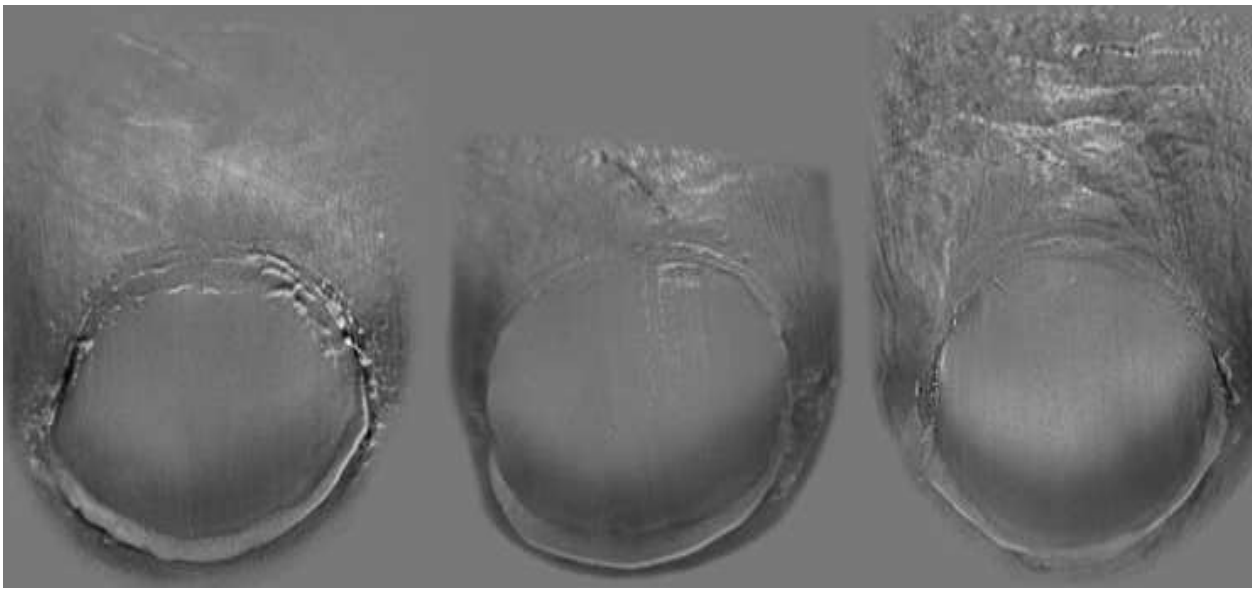

Figure 5-13: *The first principal component computed using different sets of example data from three different subjects. Shown are the index finger of subject one (left), the ring finger of subject two (middle) and the index finger of subject three (right).*

jects and example sets. The hemoglobin pigment vectors for the three subjects in Figure 5-13, number one on the left, two in the middle, and three on the left, were estimated to be

$$
\begin{aligned}
{}^{1}\sigma_h &= (0.1584, 0.7786, 0.6072), \\
{}^{2}\sigma_h &= (0.1905, 0.7810, 0.5948), \\
{}^{3}\sigma_h &= (0.2386, 0.7513, 0.6153).
\end{aligned}
$$

We note that the hemoglobin pigment vectors are remarkably similar, with ${}^{1}\sigma_h \cdot {}^{3}\sigma_h = 0.9964$, which is an angle of less than 5 degrees. The variance across the three pigment estimates is $(0.0016, 0.0003, 0.0001)$.

Figure 5-10 shows an example simulation where a virtual hand simulation is grasping an object. When we increase the desired grip strength parameter of the controller, the hand posture changes to produce larger torques at the joints and in turn larger contact forces. We see a change in finger posture for the stronger grip, but also colour changes at the fingertip due to blood distribution changes estimated by our data driven model. The effect can be seen more prominently in the supplementary video.

Figure 5-11 demonstrates both fingertip colour changes and the interactivity of our physics based grasping simulation. A mouse-spring interface is used to apply forces on the wooden block, causing it to rotate. The compliant finger joints bend to accommodate the motion, while the fingertips show colour changes due to the varying contact forces.

5.6 Discussion

Our real-time rendering method for fingertips produces hand animations that are undeniably richer versus using naive techniques. The demonstrated technique shares similarities with other work in appearance modeling, notably Tsumura et al. (1999) and Donner et al. (2008). However, our work focuses on fast appearance changes for fingers driven by contact forces in a multi-body dynamical simulation. An obvious target application for this work is video games, although the method is accessible to a range of applications and existing frameworks. The method can be readily integrated into an existing shader pipeline, and to facilitate re-use of our source code and data, we have released these materials on the project page.

The demonstrated results are compelling and we are motivated to expand on this work. Currently, we are considering building a more expressive model, and applying appearance changes to other regions of the hand. For example, the described method produces plausible colour changes for contact with the middle of the fingerpad. A limitation of our work is that it cannot faithfully reproduce details when rolling the fingertip. The center of pressure could therefore be incorporated as part of the model to address this issue.

Additionally, the viscosity parameter discussed in Section 5.4.3 requires some tuning to get plausible behaviour for colour changes. This is a minor drawback, but the parameter could be estimated from captured video and pressure data. For example, by performing a frame-by-frame reconstruction of a sequence where a dynamic pressure is applied at the finger pad, it should be possible to solve for a viscosity constant that best matches the captured sequence. However, this requires significant manual effort in order to pre-process and puppet warp individual video frames. An automatic technique could be devised for this purpose.

Finally, previous work has treated melanin and hemoglobin as chromophores that independently contribute to skin colour (Jimenez et al., 2010; Tsumura et al., 1999). An interesting validation experiment, which we intend to carry out as part of future work, is verifying that our model is transferable to the fingers of people with varying melanin contents, such as people of different ethno-racial backgrounds.

Chapter 6

Interaction capture system for grasp synthesis

The framework presented in Chapter 3 uses a combination of simulation, optimization, and a predetermined controller structure to successfully perform in-hand manipulation of objects. This is a common approach, whereby carefully selected features of the simulation are used to construct objective functions. This also means that a state machine needs to be designed for each class of motion or manipulation problem, and features of the simulation must be carefully selected in order to construct an objective function. Ideally, controllers for synthesizing natural human motion could be created automatically from real-world examples. With this goal in mind, we created a novel sensor ensemble to perform motion and interaction capture, and have taken steps towards building controllers from the data.

This chapter presents the details of the capture setup. This unique system may be used to examine human grasping and manipulation tasks. Additionally, we present experiments based on data collected from real-world interactions and preliminary results are discussed. Also, we discuss our initial work on controller design by interaction capture and our success in this area.

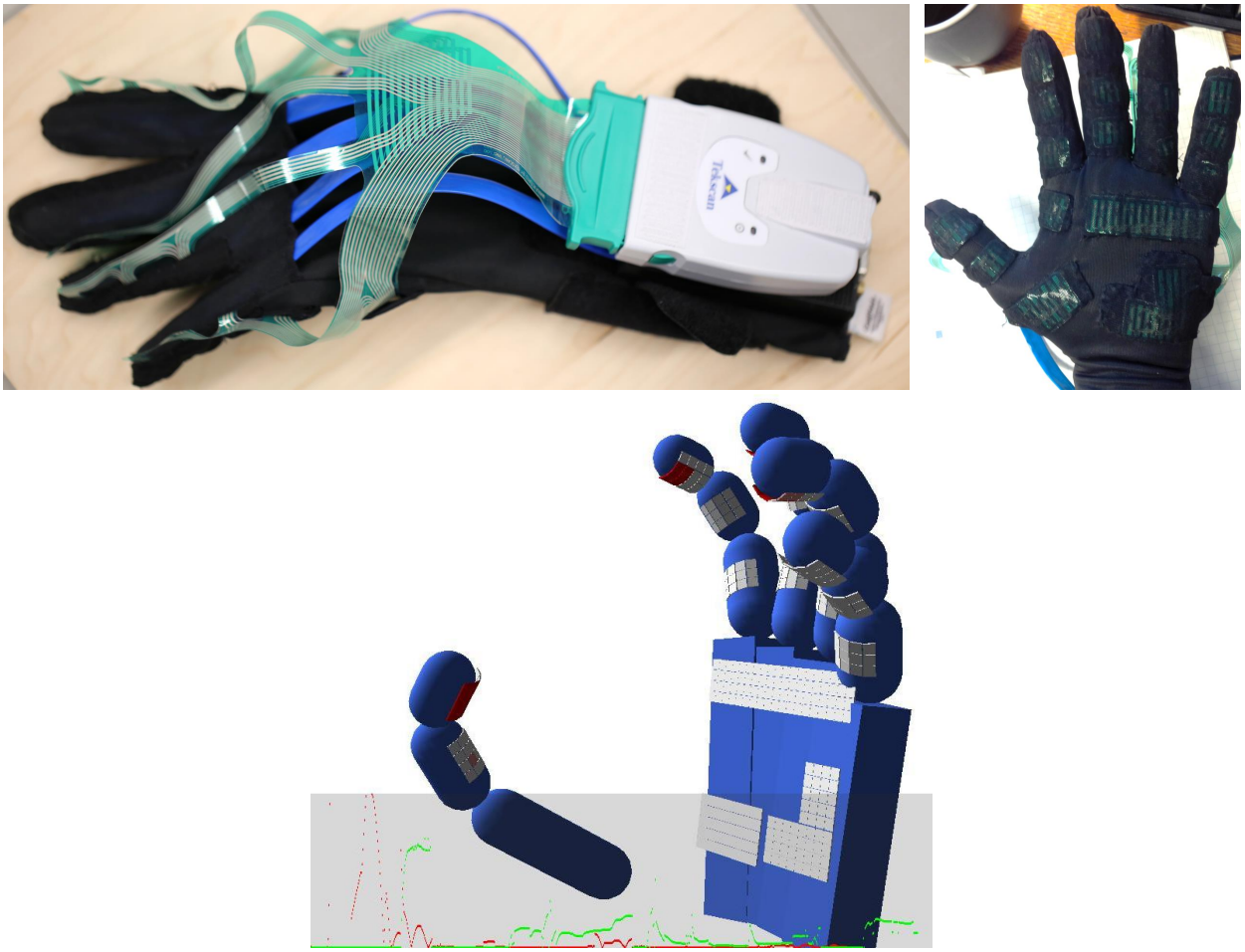

Figure 6-1: *Top row, showing the combined Tekscan Grip and Measurand ShapeHand sensors. Bottom row, a screen shot from the custom C++ application used to capture and visualize data from the sensors. The redness of the virtual taxels corresponds to the measured pressure, and a real-time plot at the bottom of the screen provides the user with an estimated grasp quality that is computed from sensor data.*

6.1 Sensors and software

One way to generate natural motion is by replaying data collected using a motion capture device. For example, an optical tracking system may be used to capture and reconstruct the joint motions of a human character model, including their hands. However, other devices are better designed for capturing hand and finger joint motions. One such device is the Measurand Shapehand,[1] which we use as one of the components. It uses 40 flexion sensors to reconstruct the joint motions of the fingers and wrist. Motion frames are sampled at a rate of 100 Hz. Each frame is a set of

[1] `http://www.shapehand.com/`

16 quaternions, representing the rotations of proximal, intermediate, and distal phalanges, as well rotation of the wrist.

We augment the Shapehand glove and combine it with a Tekscan Grip[2] pressure sensor. This type of sensor captures tactile and interaction forces at 18 sites across the finger pads and palm. Each site varies in size and has an array of tactile sensing elements, or taxels, which are capable of measuring pressure up to 50 psi. There are 349 taxels in total. The pressure data is stored as a 25×29 resolution image, which may be sampled at a rate of up to 750 Hz.

Figure 6-1 shows the assembled system without data cables. The pressure sensors are very thin and conveniently sewn into custom designed pockets on the front of the glove. The flexion sensors are inserted into pockets on the back of the glove. These pockets are provided as part of the original manufacturer's design. The signal processing units for both the joint motion sensors and pressure sensors are mounted on the forearm, behind the wrist, using a velcro strap.

We implemented a custom C++ application to collect data from the two sensor devices. The pressure sensors update at a much faster rate than the joint motion sensors. Therefore, frames from each device are buffered and a separate timer function is used to synchronize the data structures. The buffers are updated at a rate of 100 Hz. We estimate that sampling at this rate is sufficient to capture transient force and joint velocity information during grasping tasks. For instance, during pseudo-static grasping tasks, reaction times to disturbance forces are typically greater than 125 ms (Johansson et al., 1992); our system can collect dozens of pressure frames over this duration. Likewise, hand joint velocities across a variety of reaching and grasping tasks is typically less than 50 rad/s (see Spalding, 2010), a signal rate that we can faithfully reproduce based on the Nyquist-Shannon sampling theorem.

The sensor data is also used to update the joint configuration of a 3D hand model in a real-time simulation (see Figure 6-1). Pressure data is visualized by changing the colour of virtual taxels drawn on the hand. The application allows the user to adjust the dimensions and location of taxel arrays on the virtual hand, thus giving better correspondence with the real hand and glove. Also, a grasp quality metric (Ferrari and Canny, 1992) is computed at each time step using the position and

[2] `http://www.tekscan.com/grip-pressure-measurement`

orientation of virtual taxels, and contact forces estimated from the pressure measurements. This information is plotted to an on-screen display.

6.2 Postural synergies for grasp optimization

In this section, we discuss how data collected with our system is used to learn grasps for a climbing simulation. Recall that the work in Chapter 3 demonstrates how the challenge of determining controller parameters for a 23 DOF hand model can be mitigated by incorporating a user-defined pose corpus and computing a reduced pose basis. We use a similar approach here, but rather than manually defining the poses, a postural synergy is determined from real-world measurements obtained using the system described in Section 6.1.

The climbing application includes a fully dynamic articulated character with $n = 83$ DOF, which uses a collection of low-level controllers to support itself and make progress on a climbing wall. One of these, the grasping controller, pre-shapes the hand and plants the fingers on a target handhold. This section will discuss the optimization process used to learn grasping poses and its integration as part of the controller.

6.2.1 Reduced parameter space

We build a reduced pose basis using a sample set of 21 grasp postures captured during climbing trials. Each sample consists of a pose vector $q \in \mathbb{R}^n$, where n is the number of degrees of freedom of the hand skeleton. The elements of q correspond to the joint angles of the hand, excluding the wrist. The samples represent static grasp postures used by human subjects to support themselves on a climbing wall.

From the samples, PCA is used to construct $U \in \mathbb{R}^{n \times m}$ whose columns contain a reduced eigen basis of the postural synergies used in climbing. Note that m is much less than n, the total degrees of freedom for the hand. A target posture for the hand may be computed by the relationship $q \approx Up$, where $p \in \mathbb{R}^m$ is a coordinate vector in the reduced space. This allows us to search for a grasp posture using far fewer degrees of freedom. The postural synergies used by the optimization are shown in Figure 6-2, and for our experiments $m = 3$.

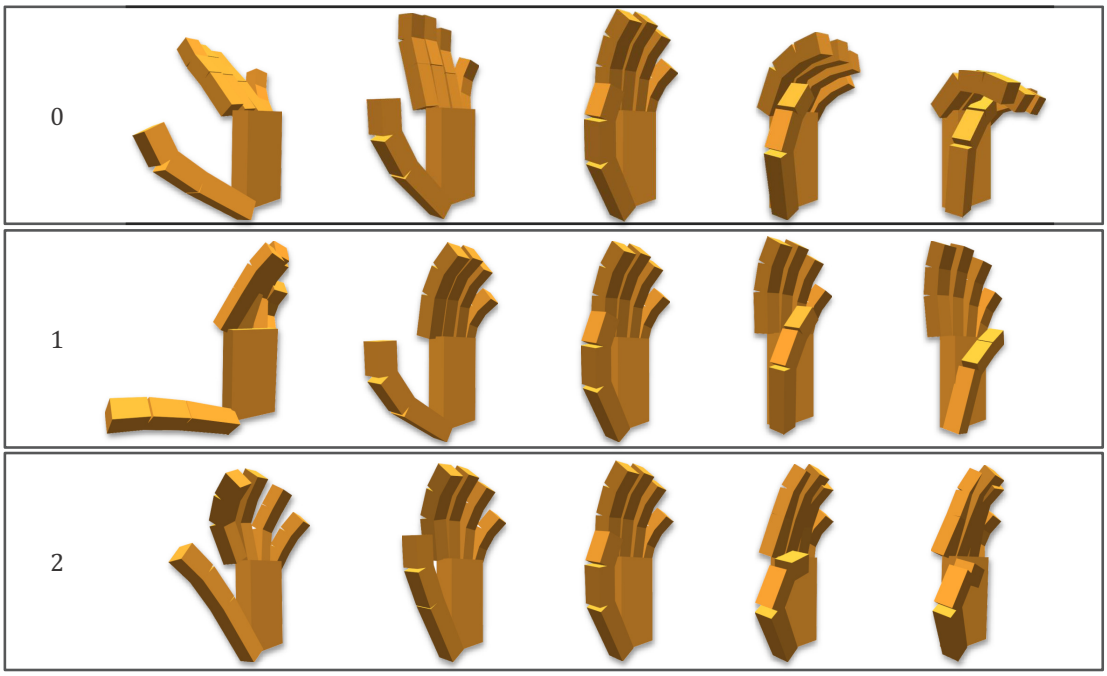

Figure 6-2: *The eigen postures used for grasp optimization. Each row shows poses from a single coordinate index, as the parameter value is gradually increased.*

6.2.2 Grasp optimization

A pose is learned by solving a multi-objective continuous optimization problem. At runtime, this pose is tracked using PD servos, actuating the joints of the fingers and wrist towards a reference configuration to successfully grasp a handhold. The controller parameters consist of a vector $\tilde{o} \in \mathbb{R}^6$, giving the desired position and Euler angle orientation of the wrist, and the target eigenvector pose $\tilde{p}$. We use $a = (\tilde{o}, \tilde{p})$ to refer to the wrist and the reduced coordinate parameters collectively. Please note that in the proceeding discussion, we use $\tilde{\cdot}$ to denote a target quantity, and $\hat{\cdot}$ for unit length vectors.

To determine suitable controller parameters for a given handhold, we solve a minimization problem for the multi-objective function

$$\min_a w_d L_d + w_b L_b + w_f L_f + w_v L_v.$$

This is done by running a forward dynamical simulation to evaluate candidate solutions for a, and the CMA-ES algorithm is used to find an optimal solution. The initial configuration for the

simulations has the hand located approximately 16 cm in front of the handhold with a neutral pose (see middle column Figure 6-2). Details about the objective terms, and how they are computed, are provided in the proceeding sections.

Fingertip distance. The term L_d is used to minimize the distance between the fingertips and the handhold at the end of the grasping phase. This helps the optimization find motions that move toward the handhold, and is useful when exploring regions of the parameter space where the final configuration of the hand is not in contact with the handhold. It is computed as the sum of the minimum distance between each finger segment and the handhold, or

$$L_d = \sum_{i \in K} \frac{1}{r} d(G_i, G_{\text{hold}}).$$

The parameters G_i and G_{hold} define the triangle mesh geometry of the ith finger segment and the handhold, respectively. Note that the Euclidean distance $d(G_i, G_{\text{hold}})$ is divided by the bounding sphere radius r of the hand, thus normalizing the range of values for this objective term. The distance function returns 0 if the finger segment intersects the handhold mesh. Also note that only the distal finger segments are contained in the set K. This was found to give satisfactory results for the optimization.

Bounds. The term L_b introduces a penalty for controller parameters that are outside the valid range. For parameters controlling the orientation of the wrist, a range of min-max values $[o^{min}, o^{max}]$ is selected that reflects the set of natural wrist postures. Similarly, the range of values $[p^{min}, p^{max}]$ for the eigen pose $\tilde{p}$ is determined by mapping the original set of sample poses onto the basis U^T and choosing the minimum and maximum value of each coefficient index. Parameter values outside this range are penalized, and the objective term becomes

$$L_b = \sum_{i=1}^{m+6} \begin{cases} \frac{|a_i - a_i^{max}|}{|a_i^{max} - a_i^{min}|}, & a_i > a_i^{max} \\ \frac{|a_i - a_i^{min}|}{|a_i^{max} - a_i^{min}|}, & a_i < a_i^{min} \\ 0, & \text{otherwise} \end{cases}.$$

Note that the penalty term is normalized, and here a_i is the ith element of the controller parameter vector.

Force transmission ratio. The term L_f attempts to maximize the force transmission ratio (FTR) due to fingers in contact with the handhold. The FTR is an estimate of the potential force that can be exerted in a given direction. We compute it as in the work by Chiu (1987). The force applied by the fingers must be in a direction opposite the force required to support the character in the climbing task, since contacts at the fingertips are used to resist the effects of gravity and inertia. For example, if climbing in a vertical direction, the character must apply a force in a downward direction. To compute the FTR, only contacts that have the property $\hat{n}_i^T \hat{v}_i < 0$ are considered. Here, $\hat{n}_i$ is the unit length contact normal at the ith finger segment and $\hat{v}_i$ is a unit length vector giving the front facing direction for that finger segment. By accumulating the ratio over each of the k finger segments, an estimate of the FTR for the grasp is determined. Therefore, the objective term becomes

$$L_f = \frac{1}{\sum_{i=1}^{k} \alpha(c_i, \hat{s})},$$

where $\alpha(c_i, \hat{s}) = \left(\hat{s}^T J_{c_i} {J_{c_i}}^T \hat{s}\right)^{-\frac{1}{2}}$ is used to compute the FTR. Note that J_{c_i} is the linear Jacobian computed at contact c_i, and the unit length vector $\hat{s}$ is the supporting direction for the climbing task. Note that we assume that $\alpha(c_i, \hat{s})$ is non-negative, and if no finger segment is in contact with the handhold, $L_f = 100$ (i.e., a large value to dominate other terms in the objective function).

Alignment. The final term L_a penalizes orientations of the forearm that are misaligned with the desired pulling or pushing direction, thus ensuring natural postures for the wrist. The objective term is

$$L_a = 1 + \sigma\, \hat{u}^T \hat{s},$$

where $\hat{u}$ is a unit vector pointing in the direction of a line connecting the elbow and wrist, and σ is $+1$ or -1, depending on if a pull or push is required.

Figure 6-3 shows some of the grasping poses found by the optimization algorithm for a target handhold. The optimization takes approximately 20 seconds on an six-core Intel 3.2 GHz processor per direction per handhold. A parallelized implementation of the CMA-ES algorithm is used. A sequence showing the character reaching, grasping, and supporting itself on a climbing wall is shown in Figure 6-4. The RTQL8[3] physics engine is used to simulate the character and climbing

[3] `http://www.cc.gatech.edu/~karenliu/RTQL8.html`

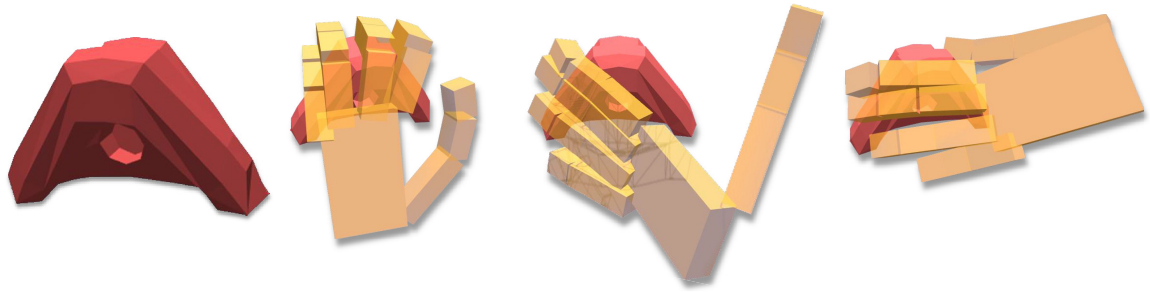

Figure 6-3: *Learned grasping postures for the climbing wall handhold shown on the left. The different poses found by the optimization depend on the direction of the supporting force for the character. Optimization parameters* $w_f = 5.0, w_b = 10.0, w_d = 0.2, w_c = 0.2, w_a = 1.0$ *were used to generate these results.*

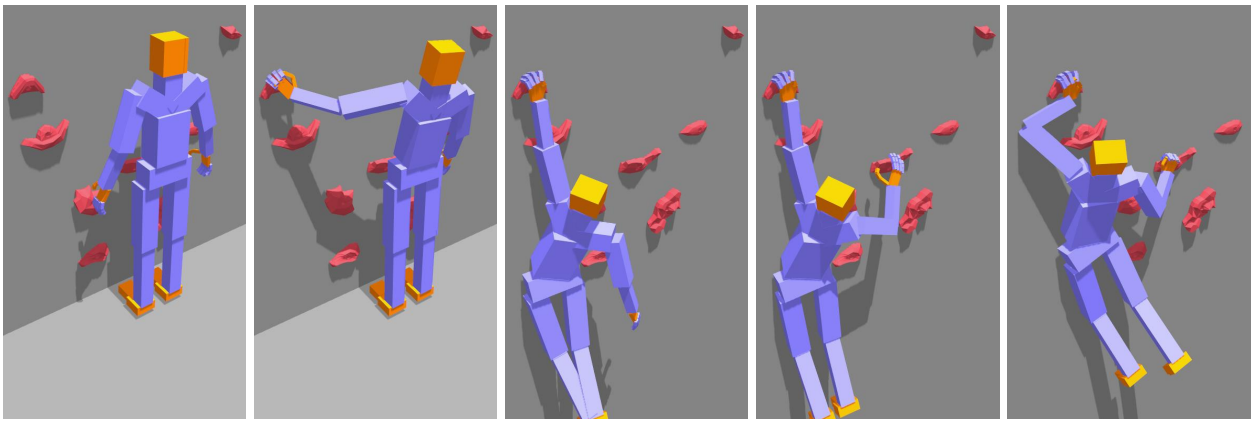

Figure 6-4: *A character supports itself by grasping handholds on a climbing wall. An offline optimization step is used to learn grasping postures for each handhold. At runtime, an appropriate grasp is selected based on the limb configuration of the character and the climbing direction.*

wall. The postures used for the pre-shaping phase are the ones learned by the optimization. A video clip of the climbing sequence is available on the project page.[4]

In this section, we have demonstrated how a collection of real-world data may be used to determine a synergy of climbing postures and plan grasps in a low dimensional parameter space. The resulting poses are plausible and have been successfully used in a virtual climbing application. Similar to the approach outlined in Chapter 3, the grasping optimization described here also uses an offline step to learn controller parameters. Although the latter method uses a forward dynamical simulation to evaluate controller parameters, a faster approach may be possible. For example, a dense set of grasp poses could be generated as a pre-computation step, and collision detection performed online

[4] `http://www.cs.mcgill.ca/~sandre17/climbing/`

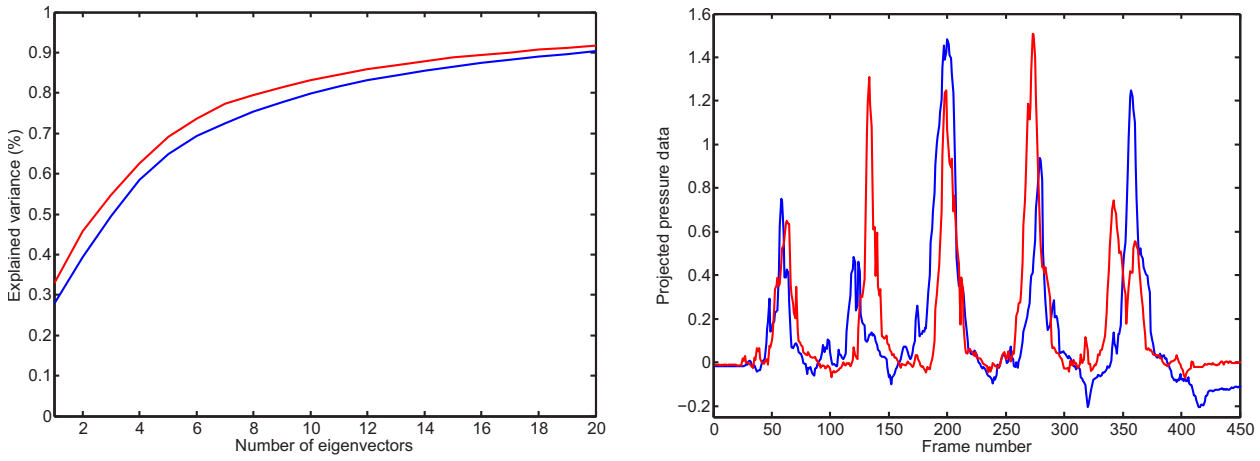

Figure 6-5: *Left, the singular values of the* 25×29 *pressure images is computed using PCA for a phone flipping sequence involving hundreds of frames. Two separate trials are shown, in red and blue. Although the dimensionality is high (725 pressure values), most of the variance (more than 90%) is accounted for by the first 20 principal component vectors. Right, the pressure data is projected onto the first principal component vector computed from each motion sequence. The phone was flipped five times, and this is easily determined by counting the peaks.*

against these configurations to quickly estimate the FTR of a grasp. This type of computation is relatively cheap and allows for parallelized implementation as a GPGPU program, since each configuration may be evaluated independently. We are currently exploring this approach, which could eliminate the need for a costly non-linear optimization and allow the character to interact with arbitrary 3D meshes.

6.3 Force synergies for manipulation tasks

In addition to postural synergies, we examine the pressure data collected for a variety of manipulation tasks. These tasks are similar to the ones discussed in Chapter 3, involving single-handed re-orientation of objects. Also, the motions for these tasks exhibit cyclic finger gaiting. One such task included in our analysis is turning over a cell phone. For example, consider when a person removes a cell phone from their pocket and the screen is facing the wrong direction. The phone may be re-orientated, or flipped, so that the screen is visible.

Pressure frames were captured from a real person performing the phone flipping task. This in-

volved the capture subject turning over the phone continuously in the palm of their hand. Note that since the task required them to flip multiple times, grasp stability is part of their control strategy, since otherwise it is likely that dexterous manipulation would not be possible after the first few turns.

PCA was performed on pressure data from separate trials. Approximately 450 sampled frames were used in the analysis, and each pressure frame consists of 725 pressure values from the taxel arrays on our glove. The plot on the right side of Figure 6-5 illustrates that the first 20 principal component vectors contain more than 90% of the variance in the pressure data. Also, when the pressure frames are projected onto the first PC vector, the cyclic nature of the motion becomes obvious; peaks of the plot clearly correspond to the sequence of five flips.

Similarly to how control of hand postures for grasping tasks involves a few postural synergies (see Santello et al., 1998), the modulation of contact forces during manipulation may be equally synergistic. Contact planning for grasping typically occurs in the space of contact wrenches at fingertips. For a given task, the general footprint of contact forces across the hand could be determined relatively quickly using a latent variable model derived from statistical data, with small scale adjustments used to refine the manipulation.

Applications other than simulated grasping may also be possible using this type of analysis. For example, the interaction synergies provide patterns that may be used for task or object identification using pressure sensor data.

6.4 Discussion

Although this chapter presents preliminary work on data capture and grasping control, the results are encouraging. Our novel system for collecting interaction data gives us a unique opportunity to make interesting progress in skilled motion synthesis for hands. Therefore, we plan to leverage this data as part of future work.

Our analysis of interaction forces in Section 6.3 indicates that basic force patterns are being used to perform stable grasping and manipulation of objects. This corroborates the observations of Santello and Soechting (2000), that synergies are used to generate a majority of the control forces

in similar grasping scenarios. We envision a plan of acquiring interaction data from many real-world manipulation tasks and using it to determine control strategies for online grasp synthesis. In Chapter 3, an appropriate control strategy was selected based on the phases used by the mid-level control framework. The plots shown in Figure 6-5 suggest that automatic segmentation of interaction forces into phases is also possible. Segmentation of the interaction trajectories we capture could be used to automatically design controllers from real-world manipulation experiments, for example, by splitting the trajectories into phases and tracking pertinent simulation features. Modifying existing methods for segmenting motion capture to perform phase-level segmentation warrants investigation, such as the work by Barbič et al. (2004).

The climbing experiments discussed in Section 6.2 use synergies built from a hand pose corpus. This was done using only the joint angle information about the hand. However, we intend to explore the idea of building interaction synergies which include both contact force and joint angle information. A full body motion capture system, like OptiTrack,[5] could be combined with our interaction capture system to collect a comprehensive dataset from human climbing trials. Knowing motion trajectories for the full skeleton will allow us to generate synergies that include the limbs and trunk, in addition to the contact forces and joint motion of the hand. Motion and contact planning could be performed for entire limbs using a low dimensional space of control parameters. Planning approaches for characters in contact with their environment, such as the work by Tonneau et al. (2014), could then be applied to real-time climbing tasks for characters in a multi-body dynamical simulation. This is an exciting idea, since the reduced coordinate space facilitates an online method.

[5] https://www.naturalpoint.com/optitrack/

Chapter 7

Conclusions and future work

This thesis has presented novel methods for animating hands in physics-based virtual environments. The methods run in real-time, or in the case of the FORK^{-1}S method, faster than real-time. The low computational overhead also makes them well suited for a variety of applications where physically based articulated characters are used.

7.1 Contributions

Synthesizing realistic motion for dexterous manipulation tasks is a difficult problem. A framework for generating single-handed grasping motions was introduced in Chapter 3. By building a policy of phase based controllers, motions are synthesized for a variety of manipulation tasks. Optimization by a forward dynamics simulation is used to tune their parameters. Not only are the motions plausible, but since the approach does not assume a predefined trajectory and the primary objective is to achieve a given goal state, the agent is capable of adapting to task changes in real-time. The multi-phase controller architecture also generates motion sequences that exhibit periodic finger gaiting, which is a characteristic of dexterous manipulation. The motions meet not only the task objectives, but stylistic ones too. Additionally, the agent is able to adapt to changes in the task specification in real-time, making the approach an excellent one for interactive applications. Although the control policies can be queried in real-time, building one is protracted by having to run many forward dynamics simulations. Accelerated models for physics simulation is therefore

an important stream of research. Within the context of our work, it allows for efficient evaluation of control parameters. The FORK^{-1}S method outlined in Chapter 4 provides a fast alternative to simulating virtual humans and robots. By focusing on the end effectors, the simulation only needs to solve a small dense system while the full state of the non-reduced mechanism can be computed in parallel. Using the exponential map to compute the state of the internal links produces good behaviour with little separation at joints for a good range of interaction forces. The method requires no special treatment for loops in the constraint topology, and permits different levels of physics fidelity by adjusting the number of end effectors included in the reduced model. FORK^{-1}S provide an important new approach among a large spectrum of techniques that are necessary for the creation of interactive and immersive virtual environments.

Synthesizing plausible motion for an articulated character is only one aspect for creating believable animated characters; rendering and displaying surface detail is also of concern. Chapter 5 describes a method for changing the appearance of fingertips due to contact forces. This provides important visual cues to show the force used when grasping an object, much like shadows give important information about the existence of contact. Although deformations and wrinkles are also important cues that give an indication of the interaction force, these attributes are not addressed since other techniques already exist to deal with them. Ultimately, any of the contact deformation techniques proposed in previous work could be combined with our work to improve the overall quality of hand simulations. Additionally, the construction of the data-driven model used by this work is straightforward, requiring a minimal storage and computational footprint. The inclusion of these appearance changes within a physics-based grasping simulation demonstrates how subtle visual cues can improve the richness of the overall interactive experience.

7.2 Future work

Incorporating linear feedback as part of our controller design will likely improve the robustness of the learned control policies. Feedback using domain specific features has been successfully applied to character locomotion tasks (Yin et al., 2007). The work by Ding (2011) presents a general approach for designing linear feedback controllers and preliminary efforts to integrate with our framework are promising. The use of this type of feedback control in the grasping domain is

interesting and deserves further investigation. For example, by tracking grasping features, such as pressure distribution at finger segments, it may be possible to improve the overall robustness and stability of the control policy. Also, it is likely that a sparser control policy could be learned if feedback were incorporated into the controllers.

A straightforward extension to the control work presented in this thesis is to use a motion capture corpus to compute a latent parameter space for the low-level controllers. Investigating the force-joint synergies of human grasping is key to building more complex control strategies that correspond to phase transitions that occur in actual manipulation tasks. Another potentially interesting avenue of research is in building control policies for coordinated manipulation with two or more hands. One possible approach is to treat control policies learned for single-handed tasks as abstract actions within a hierarchical reinforcement learning framework, as shown by Huber and Grupen (1998). Such strategies remain relatively unexplored in computer animation applications, but have been successfully used in robotics research applications.

In addition to the grasping postures captured during climbing tasks, interaction forces were also collected using the system described in Section 6.1. Work by Santello and Soechting (2000) noted that force synergies exist for multi-fingered grasping tasks. They determined that variations in force patterns during holding and lifting tasks were modulated as a function of physical properties, such as the relative location of the center of mass. It is likely that such synergies could be exploited to improve the grasping controller used by the climbing application. Preliminary work on this has been very successful, with the grip strength of the learned grasp modulated according to the force required to support the character. We also plan to investigate using postural synergies that include the limb joints as part of motion planning for climbing dynamic characters.

The FORK^{-1}S method currently does not consider the geometric limits of joints in modeling the compliant behaviour of an articulated mechanism. Some preliminary work has begun on non-linearly scaling the effective compliance once geometric joint limits are reached. This technique may be used to adjust the behaviour of dependent bodies in the chain to restrict their motion once a joint limit has been reached. Alternatively, multiple FORK^{-1}S models could be built by linearizing the behaviour at strategic points in the configuration space of the mechanism. Nearby models could then be blended to give a better approximation of the behaviour of the full system. Investigating

various methods for controlling end effector motion is a related topic. Rather than assuming a static joint configuration with PD servo control, it may be possible to linearize the compliant behaviour at multiple configurations across pose space. By blending across these models, the compliant behaviour of the mechanism could be simulated for the entire motion trajectory. Manipulation tasks for character animation and robotics simulation could benefit from such an extension to this work.

There is some manual effort required to pre-process the images used in building our fingertip appearance model. However, warping images to permit the construction of the PCA basis only needs to be done once, and the model is transferable across fingers and subjects. An automatic method is being developed to perform image warping and construct the PCA vectors, reducing the required manual effort. Also, it is natural to include the center of pressure in the model of hemoglobin changes. The center of pressure may be estimated from the current data set and this extension is currently being explored. Note that the variance of the pressure distribution may also be measured in the data collection process, and in turn create a model that can capture the difference between a sharp and flat contact surface (e.g., the corner of a cube in contrast to a face). Because the physics simulation uses rigid collision proxies, it would likewise need to be reworked to include soft elastic contact, or estimate the size of contact patches based on inter-penetration volume.

In a broader context, the methods presented in this thesis are pieces of a larger puzzle, with the overall mandate to create high-fidelity, interactive digital characters. One long-term challenge facing physics-based animation researchers is how to use the growing catalogue of techniques in commercial video games. Future research should focus on leveraging the parallel computing platforms which are ubiquitous for consumer gaming. Similarly, data harvested from human motion and real-world interactions provides important targets for underlying models used in control, simulation, and rendering of human characters. Strategically combining this data with physics-based models is a daunting task, but has thus far shown promising results. Although it is tempting to simply rehash trajectories, a vast amount of data is required to match the possible configurations of increasingly complex virtual environments. Rather, a more viable approach is to consider these trajectories as “sketches” for an underlying general framework.

Appendix A

Appendix A - Rigid Body Kinematics

Any rigid motion from one position to another may be described as a *screw* motion. That is, there exists a coordinate frame in which the motion consists of a translation along an axis combined with a rotation about the same axis. The time derivative of a screw motion is a *twist* consisting of the linear velocity $v \in \mathbb{R}^3$ and angular velocity $\omega \in \mathbb{R}^3$. Since much of Chapter 4 concerns statics, and because it is convenient to write rigid displacements (screws) in body frames, we abuse the term *twist* for these small displacements ξ. We use ϕ and the term *velocity* to write the equations of motion and specifically use the *body velocity* as defined by Murray et al. (1994a). Analogous to a twist, a *wrench* $w \in \mathbb{R}^6$ is a generalized force consisting of a linear force $f \in \mathbb{R}^3$ and a rotational torque $\tau \in \mathbb{R}^3$. Following Murray et al., we pack twist and wrench vectors with linear parts on top and angular parts on the bottom, i.e., $\phi = (v^T, \omega^T)^T$ and $w = (f^T, \tau^T)^T$.

Twists and wrenches transform to different coordinate frames using the adjoint matrix $\mathsf{Ad} \in \mathbb{R}^{6\times 6}$. To transform twists from coordinate frame a to coordinate frame b, we directly use

$$ {}^b_a\mathsf{Ad} = \begin{bmatrix} {}^b_aR & {}^b\hat{p}_a\, {}^b_aR \\ 0 & {}^b_aR \end{bmatrix}, \tag{A.1} $$

where ${}^b_aR \in SO(3)$ is the rotation matrix from frame a to b, the origin of coordinate frame a in coordinates of frame b is ${}^bp_a$, and $\hat{\cdot}$ is the cross product operator. That is, ${}^a\phi$ in frame b is computed as ${}^b\phi = {}^b_a\mathsf{Ad}{}^a\phi$. The inverse transpose of the adjoint is used to transform a wrench

between coordinate frames, ${}^{b}w = {}^{b}_{a}\mathsf{Ad}^{-T}{}^{a}w$. Finally, we use the exponential map $e^{\hat{\phi}} : \mathbb{R}^6 \to SE(3)$ on a twist to compute the relative rigid motion as a homogeneous transformation matrix. Note that here $\hat{\phi} \in se(3)$ and has the 4×4 matrix form

$$\hat{\phi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}.$$

For further details, see the formulas in Murray et al. (pp. 412-413).

Bibliography

Allard, J., Faure, F., Courtecuisse, H., Falipou, F., Duriez, C., and Kry, P. G. Volume contact constraints at arbitrary resolution. *ACM Trans. on Graphics*, 29(4):82, 2010.

Andrews, S. and Kry, P. G. Policies for goal directed multi-finger manipulation. In *The 9th Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, pages 137–145, 2012.

Andrews, S. and Kry, P. G. Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics*, 37(7):830–839, 2013.

Andrews, S., Jarvis, M., and Kry, P. G. Data-driven fingertip appearance for interactive hand simulation. *ACM SIGGRAPH conference on Motion in Games*, 2013.

Andrews, S., Teichmann, M., and Kry, P. G. FORK-1S: Interactive compliant mechanisms with parallel state computation. *ACM Symposium on Interactive 3D Graphics and Games*, 2014.

Arechavaleta, G., López-Damian, E., and Morales, J. L. On the use of iterative lcp solvers for dry frictional contacts in grasping. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6. IEEE, 2009.

Aristotle. *On the soul*. 350 B.C.E.

Ascher, U. and Lin, P. Sequential regularization methods for simulating mechanical systems with many closed loops. *SIAM Journal on Scientific Computing*, 21(4):1244–1262, 1999.

Ascher, U. M. and Petzold, L. R. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, PA, USA, 1st edition, 1998.

Baraff, D. Linear-time dynamics using Lagrange multipliers. In *Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, 1996.

Baraff, D. and Witkin, A. Large steps in cloth simulation. In *Proc. of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

Barbič, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J. K., and Pollard, N. S. Segmenting motion capture data into distinct behaviors. In *Proce. of the 2004 Graphics Interface Conference*, pages 185–194. Canadian Human-Computer Communications Society, 2004.

Barbič, J. and Zhao, Y. Real-time large-deformation substructuring. *ACM Trans. on Graphics*, 30 (4):91:1–91:8, July 2011.

Ben Amor, H., Kroemer, O., Hillenbrand, U., Neumann, G., and Peters, J. Generalization of human grasping for multi-fingered robot hands. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2043–2050, 2012.

Bicchi, A. Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Trans. on Robotics and Automation*, 16(6):652–662, 2000.

Bonneel, N., van de Panne, M., Paris, S., and Heidrich, W. Displacement interpolation using Lagrangian mass transport. *ACM Trans. on Graphics*, 30(6):158:1–158:12, 2011.

Borshukov, G., Montgomery, J., and Hable, J. *GPU Gems 3*, chapter 15. Playable Universal Capture. Addison-Wesley Professional, 2007.

Boukhalfi, T. Automatisation des expressions faciales liées à l'activité physique. Master's thesis, École de technologie supérieure, 2012.

Buss, S. R. and Kim, J.-S. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10, 2004.

Chiu, S. L. Control of redundant manipulators for task compatibility. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 4, pages 1718–1724. IEEE, 1987.

Ciocarlie, M., Goldfeder, C., and Allen, P. K. Dimensionality reduction for hand-independent dexterous robotic grasping. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems: San Diego, CA, 29 October-2 November 2007*, pages 3270–3275. IEEE, 2007.

Coros, S., Beaudoin, P., and van de Panne, M. Robust task-based control policies for physics-based characters. *ACM Trans. on Graphics*, 28(5):1–9, 2009.

Cutkosky, M. R. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Trans. on Robotics and Automation*, 5(3):269–279, 1989.

Ding, K. Learning reduced order linear feedback policies for motion skills. Master's thesis, University of British Columbia, 2011.

Donner, C., Weyrich, T., d'Eon, E., Ramamoorthi, R., and Rusinkiewicz, S. A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM Trans. on Graphics*, 27 (5):140:1–140:12, December 2008.

Duriez, C., Dubois, F., Kheddar, A., and Andriot, C. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Trans. on Visualization and Computer Graphics*, 12(1):36–47, 2006.

Dutreve, L., Meyer, A., and Bouakaz, S. Easy acquisition and real-time animation of facial wrinkles. *Comp. Anim. Virtual Worlds*, 22(2-3):169–176, 2011.

Erleben, K. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. on Graphics*, 26(2), 2007.

Erleben, K., Sporring, J., Henriksen, K., and Dohlmann, H. *Physics-based Animation*. Charles River Media, 2005.

Faure, F. Fast iterative refinement of articulated solid dynamics. *IEEE Trans. on Visualization and Computer Graphics*, 5(3):268–276, 1999.

Featherstone, R. *Rigid Body Dynamics Algorithms*. Springer, New York, 2008.

Featherstone, R. and Orin, D. Robot dynamics: equations and algorithms. *Proc. of IEEE International Conference on Robotics and Automation*, 1:826–834, 2000.

Ferrari, C. and Canny, J. Planning optimal grasps. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 2290–2295. IEEE, 1992.

Flanagan, J. R., Bowman, M. C., and Johansson, R. S. Control strategies in object manipulation tasks. *Current Opinion in Neurobiology*, 16:1–10, 2006.

Hajian, A. Z. and Howe, R. D. Identification of the mechanical impedance at the human finger tip. *Journal of biomechanical engineering*, 119(1):109–114, 1997.

Hämäläinen, P., Eriksson, S., Tanskanen, E., Kyrki, V., and Lehtinen, J. Online motion synthesis using sequential monte carlo. *ACM Trans. on Graphics*, pages 51:1–51:12, 2014.

Han, L. and Trinkle, J. C. Object reorientation with finger gaiting. In *Proc. of 2nd IMACS International Multiconference: Computational Engineering in Systems Applications (CESA'98)*, 1998a.

Han, L. and Trinkle, J. C. Dextrous manipulation by rolling and finger gaiting. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 1, pages 730–735. IEEE, 1998b.

Hansen, N. The CMA evolution strategy: A comparing review. *Towards a New Evolutionary Computation: Advanceson Estimation of Distribution Algorithms*, pages 75–102, 2006.

Hansen, P. Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank. *SIAM Journal on Scientific and Statistical Computing*, 11(3): 503–518, 1990.

Harmon, D. and Zorin, D. Subspace integration with local deformations. *ACM Trans. on Graphics*, 32(4):107:1–107:10, 2013.

Harrison, J., Rensink, R. A., and van de Panne, M. Obscuring length changes during animated motion. *ACM Trans. on Graphics*, 23(3):569–573, 2004.

Hasser, C. J. and Cutkosky, M. R. System identification of the human hand grasping a haptic knob. In *Proc. of the 10th Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, 2002.

Hauser, K., Ng-Thow-Hing, V., and Gonzalez-Banos, H. Multi-modal motion planning for a humanoid robot manipulation task. Proc. of International Symposium on Robotics Research (ISRR), 2007.

Hogarth, B. *Drawing dynamic hands*. Random House LLC, 1988.

Huang, H., Zhao, L., Yin, K., Qi, Y., Yu, Y., and Tong, X. Controllable hand deformation from sparse examples with rich details. In *Proc. of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 73–82, 2011.

Huber, M. and Grupen, R. Robust finger gaits from closed-loop controllers. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1578–1584, 2002.

Huber, M. and Grupen, R. A. Learning robot control – using control policies as abstract actions. NIPS Workshop on Abstraction and Hierarchy in Reinforcement Learning, 1998.

Jakovels, D., Spigulis, J., and Rogule, L. RGB mapping of hemoglobin distribution in skin. In *Clinical and Biomedical Spectroscopy and Imaging II*, page 80872B. Optical Society of America, 2011.

Jensen, H. W., Marschner, S. R., Levoy, M., and Hanrahan, P. A practical model for subsurface light transport. In *Proc. of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, 2001.

Jimenez, J., Scully, T., Barbosa, N., Donner, C., Alvarez, X., Vieira, T., Matts, P., Orvalho, V., Gutierrez, D., and Weyrich, T. A practical appearance model for dynamic facial color. *ACM Trans. on Graphics*, 29(6):141:1–141:10, 2010.

Johansson, R. S., Häger, C., and Riso, R. Somatosensory control of precision grip during unpredictable pulling loads. *Experimental brain research*, 89(1):192–203, 1992.

Khatib, O. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, 3(1):43–53, 1987.

Kider, J. T., Jr., Pollock, K., and Safonova, A. A data-driven appearance model for human fatigue. In *Proc. of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 119–128, 2011.

Kim, M.-J., Kim, M.-S., and Shin, S. Y. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proc. of the 22nd annual conference on Computer graphics and interactive techniques*, pages 369–376. ACM, 1995.

Kim, T. and James, D. Physics-based character skinning using multidomain subspace deformations. *IEEE Trans. on Visualization and Computer Graphics*, 18(8):1228–1240, 2012.

Klein, C. A. and Baho, B. E. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, June 1987.

Kovar, L., Schreiner, J., and Gleicher, M. Footskate cleanup for motion capture editing. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 97–104, 2002.

Kry, P. G., Reveret, L., Faure, F., and Cani, M. P. Modal locomotion: Animating virtual characters with natural vibrations. *Computer Graphics Forum*, 28(2):289–298, 2009.

Kry, P. G. and Pai, D. K. Interaction capture and synthesis. *ACM Trans. on Graphics*, 25(3): 872–880, 2006.

Kry, P. G., James, D. L., and Pai, D. K. Eigenskin: real time large deformation character skinning in hardware. In *Proc. of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 153–159, 2002.

Kumar, V., Tassa, Y., Erez, T., and Todorov, E. Real-time behaviour synthesis for dynamic hand-manipulation. In *Proc. of IEEE International Conference on Robotics and Automation*, 2014.

Kyota, F. and Saito, S. Fast grasp synthesis for various shaped objects. *Computer Graphics Forum*, 31(2):765–774, 2012.

Lewis, J. P., Cordner, M., and Fong, N. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, 2000.

Li, Y., Fu, J. L., and Pollard, N. S. Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Trans. on Visualization and Computer Graphics*, 13(4):732–747, July 2007.

Li, Z. and Sastry, S. S. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal of Robotics and Automation*, 4(1):32–44, 1988.

Liu, C. K. Dextrous manipulation from a grasping pose. *ACM Trans. on Graphics*, 28(3):1–6, 2009.

Liu, L., Yin, K., van de Panne, M., Shao, T., and Xu, W. Sampling-based contact-rich motion control. *ACM Transctions on Graphics*, 29(4):Article 128, 2010.

Magnenat-Thalmann, N., Laperriere, R., and Thalmann, D. Joint-dependent local deformations for hand animation and object grasping. In *Graphics Interface*, pages 26–33, June 1988.

Mascaro, S. and Asada, H. Measurement of finger posture and three-axis fingertip touch force using fingernail sensors. *Robotics and Automation, IEEE Trans. on*, 20(1):26–35, 2004.

Mordatch, I., Popovic, Z., and Todorov, E. Contact-invariant optimization for hand manipulation. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 137–144. Eurographics Association, 2012a.

Mordatch, I., Todorov, E., and Popović, Z. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. on Graphics*, 31(4):43, 2012b.

Mukherjee, R. M. and Anderson, K. S. A logarithmic complexity divide-and-conquer algorithm for multi-flexible articulated body dynamics. *Journal of Computational and Nonlinear Dynamics*, 2(1):10–21, 2006.

Murray, R., Li, Z., and Sastry, S. S. *A mathematical introduction to robotic manipulation*. CRC Press, 1st edition, 1994a.

Murray, R. M., Sastry, S. S., and Li, Z. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994b.

Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

Nesme, M., Kry, P. G., Jeřábková, L., and Faure, F. Preserving topology and elasticity for embedded deformable models. *ACM Trans. on Graphics*, 28(3):52, 2009.

Nishidate, I., Maeda, T., Niizeki, K., and Aizu, Y. Estimation of melanin and hemoglobin using spectral reflectance images reconstructed from a digital rgb image by the wiener estimation method. *Sensors*, 13(6):7902–7915, 2013.

Nunes, R. F., Cavalcante-Neto, J. B., Vidal, C. A., Kry, P. G., and Zordan, V. B. Using natural vibrations to guide control for locomotion. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 87–94. ACM, 2012.

Oat, C. Animated wrinkle maps. In *ACM SIGGRAPH 2007 courses*, pages 33–37, 2007.

Okamura, A. M., Smaby, N., and Cutkosky, M. R. An overview of dexterous manipulation. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 255–262. IEEE, 2000.

Olsen, T. G. K.-R., Andrews, S., and Kry, P. G. Computational climbing for physics-based characters. Poster presented at the 2014 ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2014.

O'Sullivan, C. and Dingliana, J. Collisions and perception. *ACM Trans. on Graphics*, 20(3): 151–168, 2001.

Parker, E. G. and O'Brien, J. F. Real-time deformation and fracture in a game environment. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 165–175, 2009.

Pollard, N. S. and Zordan, V. B. Physically based grasping control from example. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 311–318, 2005.

Pollard, N. S. Synthesizing grasps from generalized prototypes. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 3, pages 2124–2130. IEEE, 1996.

Redon, S., Galoppo, N., and Lin, M. C. Adaptive dynamics of articulated bodies. *ACM Trans. on Graphics*, 24(3):936–945, 2005.

Santello, M. and Soechting, J. F. Force synergies for multifingered grasping. *Experimental Brain Research*, 133(4):457–467, 2000.

Santello, M., Flanders, M., and Soechting, J. F. Postural hand synergies for tool use. *The Journal of Neuroscience*, 18(23):2123–2142, December 1998.

Sloan, P.-P. J., Rose, C. F., III, and Cohen, M. F. Shape by example. In *Proc. of the 2001 symposium on Interactive 3D graphics*, pages 135–143, 2001.

Spalding, M. C. *Characterizing the correlation between motor cortical neural firing and grasping kinematics*. PhD thesis, University of Pittsburgh, September 2010.

Sueda, S., Kaufman, A., and Pai, D. K. Musculotendon simulation for hand animation. *ACM Trans. on Graphics*, 27(3):83:1–83:8, Aug 2008.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning I: Introduction*. The MIT Press, 1998.

Tan, J., Liu, K., and Turk, G. Stable proportional-derivative controllers. *Computer Graphics and Applications, IEEE*, 31(4):34–44, July 2011.

Teichmann, M. A grasp metric invariant under rigid motions. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 3, pages 2143–2148. IEEE, 1996.

Tonneau, S., Pettré, J., and Multon, F. Task efficient contact configurations for arbitrary virtual creatures. In *Proc. of the 2014 Graphics Interface Conference*, pages 9–16. Canadian Information Processing Society, 2014.

Tsumura, N., Haneishi, H., and Miyake, Y. Independent component analysis of skin color image. *Journal of Optical Society of America A*, 16(9):2169–2176, 1999.

Tsumura, N., Ojima, N., Sato, K., Shiraishi, M., Shimizu, H., Nabeshima, H., Akazaki, S., Hori, K., and Miyake, Y. Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin. *ACM Trans. on Graphics*, 22(3):770–779, 2003.

Wang, J. M., Fleet, D. J., and Hertzmann, A. Optimizing walking controllers. *ACM Trans. on Graphics*, 28(5):1–8, 2009.

Yamane, K. and Nakamura, Y. Natural motion animation through constraining and deconstraining at will. *IEEE Trans. on Visualization and Computer Graphics*, 9(3):352–360, 2003.

Yamane, K. and Nakamura, Y. Stable penalty-based model of frictional contacts. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1904–1909, 2006.

Yasumuro, Y., Chen, Q., and Chihara, K. Three-dimensional modeling of the human hand with motion constraints. *Image and Vision Computing*, 17(2):149–156, 1999.

Ye, Y. and Liu, C. K. Synthesis of detailed hand manipulations using contact sampling. *ACM Trans. on Graphics*, 31(4), 2012.

Yin, K., Loken, K., and van de Panne, M. Simbicon: Simple biped locomotion control. *ACM Trans. on Graphics*, 26(3), 2007.