Summary of Directed Study Coursework in Sensorimotor Computation, Spring 2012

Ben Humberston*
University of British Columbia

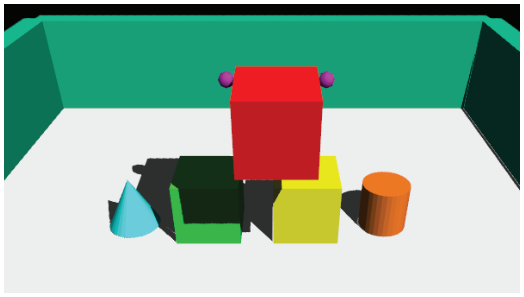


Figure 1: *Example of a virtual dynamic scene with dual haptic interaction. The red cube is lifted by the user by grasping each side of the block via a pair haptic proxies. Haptic devices track the position of the user's thumb and index finger in order to determine the location of the proxies, shown in purple.*

Abstract

A summary of the tasks and studies completed for an independent study course in sensorimotor computation are presented. Particular attention is given to a course project that explores the design of a system which lets a user interact with a dynamic virtual environment via haptic and graphical interfaces. The primary goal for the project is to enable grasping and manipulating three-dimensional virtual objects using multiple haptic effectors. Details are provided on the various possible approaches to this problem using both the HL and HD APIs of the OpenHaptics framework. The current solution tracks two finger points in the workspace and applies force feedback using a pair of Phantom 1.0 haptic devices. The work for this project is ongoing, as several issues related to haptic rendering must yet be resolved.

Keywords: virtual workspace, finger tracking, multihaptic interactions, haptic grasping

1 Introduction

The field of sensorimotor computation, speaking broadly, seeks to explain how the brains of living organisms coordinate the incoming flow of sensory information and the outgoing command signals which determine muscle activations. It integrates a number of discplines, including biomechanics, neuroscience, robotics, and control theory. Distinguishing it from related fields in biology is the focus on building constructive models of biological systems. Mathematically explicit models are created with the intent of portraying some component of a biological control mechanism (generally, the true natural system is abstracted or simplified to make the model construction tenable).

One direction in sensorimotor computation experiments investigates the nature of biological control behind voluntary motions. For example, [Krakauer 2009] quantitatively describes phenomena such as savings and learning saturation that occur when a subject adapts to a rotation between visual input and motor output. [d'Avella and Pai 2010] describes an experimental procedure to test the accuracy of the widely accepted group of theories that postulate the necessity of muscle synergies as a basic motor control unit.

In the spirit of these experiments, the main portion of time in this course was devoted to the implementation of a virtual environment that tracks one or more digit positions for use in sensorimotor experiments. The environment is designed to allow a user to grasp, move, and rotate objects in three dimensional space by directly converting motion of the finger tips into motion of a proxy in the virtual environment. Haptic feedback is applied to the user which communicates the solidity, weight, and frictional properties of objects in the scene. The highest goal of the environment is to simulate interactions on a small tabletop workspace with enough sensory realism to enable users to interact naturally with the scene objects.

For nominal convenience, the implementation of this virtual environment is entitled "Dihaptic" (an abbreviation for either "digit haptic" or "double haptic"); we use this term to refer to the project implementation in the following.

Section 2 summarizes the work completed during the past term for the directed studies. In Section 3, we provide a concise listing of the reviewed material related to sensorimotor computation. Section 4 introduces the notion of a haptic proxy and how it is employed in Dihaptic, while the physical configuration of the haptic devices is described in Section 5. Specific implementation details and issues encountered while building Dihaptic are provided in Section 6. Models considered for resolving contact and normal impulses in the dynamic virtual environment are reviewed in Section 7. Qualitative impressions of interactions with the current Dihaptic virtual environment are given in Section 8. The remaining work to resolve outstanding issues and possible experiments to build using Dihaptic are outlined in Section 9.

2 Timeline

We describe each work phase of the course project, roughly split by month. While there has been some parallel work on different directions for the project (eg: some experimentation with the HL version of the code continued while implementing the new HD version), the bulk of the work was split in time as described.

*e-mail: bhumbers@cs.ubc.ca

The first month was primarily devoted to exploring usage of the Phantom haptic devices and OpenHaptics API, as well as regaining familiarity with the C++ language itself. Using the HD calls, we implemented one of the classic introductory haptic programs, the infinite frictionless plane. This simple demonstration of haptic interaction applies a spring force to prevent a user from penetrating through a horizontally oriented plane; the feedback force increases proportionally to the penetration of the device endpoint in the direction perpendicular to the plane.

After establishing basic competence with haptic programming, approximately one additional month was spent developing a dynamic virtual environment with haptic feedback forces rendered using the HL API. A basic scene management framework was implemented in order to coordinate haptic feedback, graphical rendering, and dynamic simulation of a small virtual environment. In this implementation, the Bullet Physics Library [Coumans 2012] steps the dynamic simulation of rigid bodies with friction and contact constraints, while the environment is graphically rendered using basic OpenGL calls with ground plane shadows. For comparison, the dVC3d [Nguyen and Trinkle 2010] code was integrated as an alternative solver for contact and friction constraints. To provide haptic rendering, the HL API was selected for its simplicity and robustness. However, as described in Section 6.4, this method of rendering haptic forces proved difficult to integrate with the dynamic environment simulation.

Consequently, the remaining time in the term has been devoted to the implementation of a new haptic rendering scheme that utilizes low-level HD calls to determine force feedback. In contrast to the point-proxy scheme necessitated by the HL scheme, this new HD implementation allows arbitrary convex rigid bodies to be employed as haptic proxy objects. The visual shadow rendering has been improved in order to provide better depth and position information to the user. The switch to HD forces required modifications to the haptic rendering architecture so that forces may be smoothly rendered at a high frequency (the HL API, in contrast, abstracts away much of the internal high frequency rendering from the client code).The resolution of new issues associated with using the HD API is ongoing.

3 Related Readings

The following summarizes material covered during the past term for the directed studies. Included are both topics in sensorimotor computation as well as haptic literature relevant to the Dihaptic project.

In order to gain some exposure to the biological basis for sensorimotor studies, I attended in an informal capacity lectures for the “Sensory Systems”, “Motor Systems”, and part of the “Learning, Memory” modules of the introductory neuroscience course. Although much of the course was targeted to an audience interested in chemical or cellular aspects of neuroscience, several of the lectures covered material relevant to computational sensorimotor studies. Topics of particular interest included a description of the Jeffress delay line model for auditory localization, cell types and layering in the cerebellum, and the organization of spinal cord communication pathways.

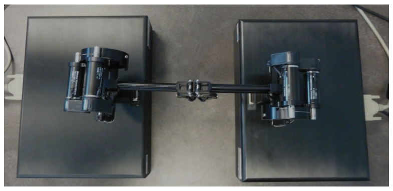
[Tweed 2003] provides a high-level motivational introduction to the area of sensorimotor studies, utilizing the vestibulo-ocular reflex to demonstrate how techniques of computational analysis may be applied to biological control systems. Chapter 5 of [Schwartz 1993] discusses the implications of and reasoning behind treating the brain as a type of computer in the field of computational neuroscience. [Tresch and Jarc 2009] briefly reviews evidence from recent experiments which support or conflict with the theory that the central nervous system uses muscle synergies to reduce the degree of complexity of motor control problems. A dynamic model of the extraocular muscles is presented by [Wei et al. 2010].

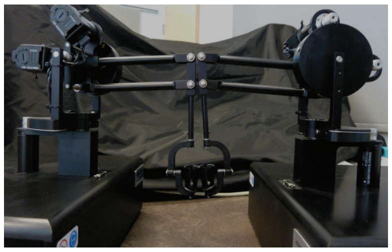
A number of topics discussed in the SMRT reading group this term covered the area of motor learning and adaptation. [Wolpert et al. 2011] provides an overview of motor learning process terms and theories. [Krakauer 2009] reviews the experimental evidence for the phenomena of adaptation, savings, interference, and training consolidation in the context of visuomotor rotations. Contrasting models of motor learning rates are given in [Ingram et al. 2011] and [Lee and Schweighofer 2009]. The former argues that only a single context-dependent learning rate is apparent when subjects interact with an object exhibiting simple dynamic behavior in the plane. The latter refines a two-process theory from [Smith et al. 2006] and suggests that experimental data necessitate a model with both a single fast and a context-dependent slow rate for visuomotor rotation adaptation. [Abdelghani et al. 2008] introduces the theory of implicit supervision, which describes how the sensitivity derivative matrix (ie: control Jacobian) may be formed and transmitted to a motor controller via synaptic firing; [Abdelghani and Tweed 2010] presents evidence that these sensitivity derivative values may in fact be numerically inverted as part of adaptation to visuomotor transformations. On the topic of voluntary grip force, [Johansson and Westling 1984] and [Westling and Johansson 1984] give an intrepretation of experiments that examined the grip force employed by subjects when lifting objects of varying mass and texture. Recent meetings have been devoted to the introductory chapters of [Shadmehr and Mussa-Ivaldi 2012], which describes computational representations of space and the possible biological implementation of such representations in the mammalian brain.

During the investigation of systems for tracking motion of the hand in three dimensions, the option of using hand-tracking solutions that impose minimal motion constraints was briefly considered. [Park and Yoon 2006] uses a glove with attached LEDs to capture hand motions that are re-interpreted into discrete gestures. The “colored glove” solution of [Wang and Popović 2009] was an attractive option for its simple equipment and low cost requirements. In that work, a user wears a glove with a distinctive color pattern; the gloved hand is captured using a consumer-grade camera. The camera’s image of the gloved hand is translated into a low-resolution, color-indexed query image; the current hand pose is determined by finding the nearest neighbor image in a precomputed database of virtual images with known hand pose. However, we decided against glove-based solutions such as the above due to the inherent estimation noise in vision-based techniques as well as the lack of force feedback when contacting virtual objects.

Using Phantom haptic devices adds the capability to render haptic feedback to complement the visual environment. [Harwin and Melder 2002] describes one of the standard methods for simulating friction when using a point-based representation of the haptic device. The creation of a two-device haptic system that uses point-based proxies is described in [de Pascale et al. 2006]. [Barbagli et al. 2004] specifies the number of contact points needed to ensure stable grip under various haptic models. [Ang et al. 2011] uses a physically constraining mechanism between two devices to eliminate the ambiguity of applied torques about the grip axis that occurs if the devices do not use orientation encoders.

By default, the Bullet Physics Library utilizes an iterative projected Gauss-Seidel solver to solve the linear system that arises in its dynamic rigid body simulations [Catto 2005]. Alternative friction and contact constraint formulations and solvers that were considered for this project are found in [Nguyen and Trinkle 2010] and [Kaufman et al. 2008].


(a) Top


(b) Side

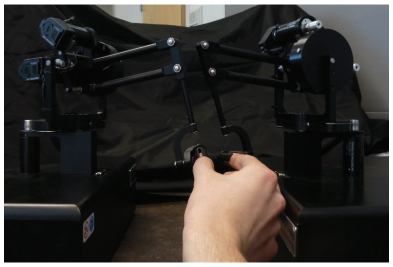

(c) Side (during active use)

Figure 2: *Images showing device setup for Dihaptic.*

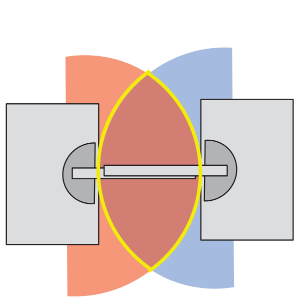
4 Haptic Proxies

Similar to many haptic applications, the user is represented in the virtual world by a "proxy" object (also known as "god object" in some sources [Harwin and Melder 2002]). In the HL variation of Dihaptic, the proxy for each device is a single point in the world that is coincident with the device position in free space but which is constrained to the surface of an object when the device position penetrates the object's surface. During penetration of objects, forces are applied to the device endpoint with the goal of restoring the endpoint to the surface of the object at the proxy position, with the applied force proportional to the penetration depth. The HD implmentation of Dihaptic uses a finite rigid body as proxy rather than an infinitesimal point. In this case, spring forces are applied to the device endpoint to restore it to the proxy object's local origin during contacts.

In the following discussion, "proxy" refers to the virtual point or object that specifies the virtual state of a device, while "device endpoint" or simply "endpoint" is used instead to denote the true physical location of the haptic device. The former must follow the constraints of the virtual world (eg: no penetration of the proxy into rigid bodies) while the latter moves freely around the world, though it is of course influenced by the forces applied to it both by the user and by the haptic feedback motors.

5 Device Configuration

Two Phantom 1.0 Premium devices provide both digit tracking and haptic feedback in Dihaptic. Each device is capable of tracking the position of its armature endpoint with submillimeter precision, and


(a) Two device setup

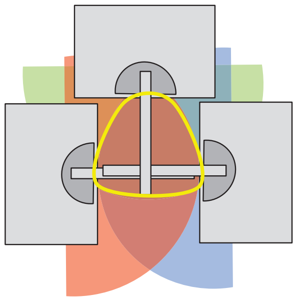

(b) Three device setup

Figure 3: *Top-down view of shared workspace for two and three device setups (outlined in yellow); the workspace for two devices (shown in red and blue) overlap to a large degree, but using three devices (third device workspace shown in green) significantly limits the usable workspace area.*

it may apply forces of a few Newtons in magnitude in any direction within its workspace. As shown in Figure 2, the devices are rotated in opposite directions by 90 degrees and positioned so that their endpoints are located at roughly the same point when they are set in the standard configuration (right angles at each link of the arm). In the virtual environment, the local device reference frames are also transformed by a 90 degree rotation so that physical motions of the two device endpoints are correctly aligned with the world frame of the environment.

In order to use the setup, the thumb and index finger are inserted in thimbals attached to the device endpoints, which provide passive rotational freedom of movement. During environmental simulation, the devices apply translational forces to the endpoint (and thus, the user's fingers) that simulate contact with the virtual environment.

Calibration defines the origin and direction of the x, y, and z axes in the device's local space; precise calibration requires that the calibration call occur when the device is exactly in the standard configuration. The Premium Phantom models in particular do not provide a method for easily manipulating the device into the standard configuration (other models may contain an "inkwell" where the endpoint is placed during calibration). During single device interactions, it is generally sufficient to place the device in approximately the correct configuration for calibration without noticable errors in the correspondence between the physical device position and the virtual proxy. However, when using two or more devices, imprecise calibration becomes apparent when identical translations of the endpoint of each device yields slightly different motions of the virtual proxy points. This error scales with distance from the virtual world's origin, as it is caused by a slight mismatch of the direction of the reference axes for each device. This issue is non-critical during development, but must be resolved before actual experiments are conducted by producing a calibration jig that allows each device to be placed in its standard configuration to high precision.

If desired, additional devices may be integrated into the setup with minimal modification of the code base. The primary impediment to increasing the number of devices is the associated reduction in the shared workspace volume between all devices. The physical workspace of each device is, roughly speaking, the hemisphere swept out by the device arm at full extension. To ensure that any point may be reached by any device proxy in the virtual environment, it is necessary to map the virtual world only to the volume defined by the intersection of the individual device workspaces. Due to spatial and mechanical constraints, this intersection volume shrinks appreciably when using more than two devices (see Figure 3). The use of additional devices thus trades workspace volume for added tracking capabilities; the decision of how many devices to integrate will depend on the experimental goals.

6 Implementation

Implementation details for the Dihaptic project are provided. All code was written in either C++ or GLSL (for shaders).

6.1 Application Structure

Updates to the virtual environment are handled by a scene management class. In response to a program timer that signals roughly every 10 milliseconds (yielding a $\sim$100 Hz update rate), the manager explicitly steps the rigid body physical simulation by the elapsed time since the previous update. After the virtual world is updated, the scene is optionally redrawn to the graphical window. If the HL API is being used, the scene is also rendered to each device separately at this time.

The application is currently embedded within a GUI constructed using the Qt framework, which handles window setup, user input, and timer management. The haptic proxies are optionally positioned relative to the graphical view, so that the effective virtual workspace changes as the user moves the camera around the scene.

6.2 Rigid Body Dynamics

The Bullet Physics Library [Coumans 2012] was selected as the rigid body dynamics simulator for its robustness, ease of use, and emphasis on real-time interactions. Bullet is responsible for both collision detection (contact generation) and contact resolution procedures. Currently, Dihaptic supports scene objects that are constructed using boxes, spheres, or convex collision meshes.

For the resolution of contact and friction forces, we use the built-in Bullet solver by default, though an implementation of the Stanley-Trinkle solver is available for comparison. Details on these solvers are given in Section 7.

6.3 Graphical View

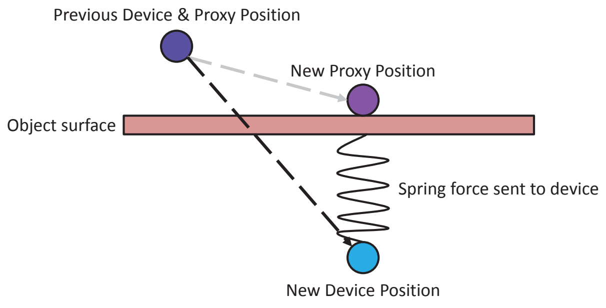
The virtual environment is rendered graphically using OpenGL at a rate of 30 - 100 Hz, depending on the host system's graphical hardware capabilities. Note that the graphical rendering rate generally runs at a lower frequency than the physical simulation update and the haptic force rendering loops; visual perception of motion is acceptably smooth at 30-60 Hz, whereas haptic forces must be refreshed at around 1000 Hz to ensure a smooth tactile perception.

For clarity, objects are currently rendered without texture mapping, but this may be implemented if desired.

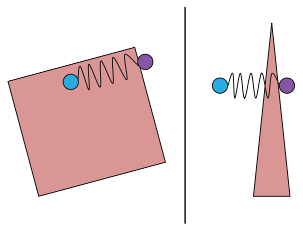
Particularly when moving through empty space, it may be difficult for the user to perceive the three-dimensional location of the haptic proxies with respect to other objects. Thus, shadows are rendered for all scene objects, including the haptic proxies, using variance shadow maps [Donnelly and Lauritzen 2006], which gives the user much-needed depth cues when interacting with the scene.

6.4 HL Implementation

OpenHaptic's HL API is modeled on the notion that haptic feedback may be "rendered" to the user in a manner analogous to graphical rendering systems. The current state of a scene is sent to the haptic device controller, specifying the surface topology of each



(a)



(b)

Figure 4: *HL contact forces. (a) Method for determining normal contact force when penetrating an object. Adapted from Figure 6-1 of [SensAble Technologies 2008]. (b) Situations where proxy usage correctly resolves desired force to simulate contact.*

object. HL programs utilize three threads of execution: client, collision, and servo. The client thread updates the current world state, stepping the physical simulation and sending the current world geometry to the collision thread. The collision thread (managed internally by HL) runs at 100 Hz; it culls and simplifies the geometry for the servo thread to accelerate force calculations and proxy position resolution. The servo thread, running at 1000 Hz, updates the device forces by applying a spring/damper force from each device endpoint toward its corresponding proxy position.

6.4.1 Contact Forces

Knowing the position of the device endpoint in both the previous and current updates, HL may determine whether the endpoint crossed the surface boundary of one of the scene objects. If this occurs, it applies a force in the direction from the device endpoint to the surface in order to simulate contact with the solid object. The specific location to which the endpoint is forced is defined by the current proxy position. HL employs a point-based proxy for maintaining the virtual position to which the device endpoint is drawn, as shown in Figure 4(a). When the device endpoint is in free space, the proxy and endpoint positions are the same. On the update when the device endpoint first penetrates an object, the proxy is positioned on the object at the orthogonal projection of the device endpoint onto its surface. While the device endpoint remains within the object, a normal contact force (effected using a spring/damper system) is applied to the endpoint which pulls the device endpoint toward the current proxy position on the surface.

The use of a proxy point resolves possible ambiguities regarding the correct object contact location, which are incorrectly resolved using earlier, non-proxy approaches, as described in [Zilles 1995]. Particular instances of this resolution are shown in Figure 4(b). On the left, the device endpoint has penetrated a box-shaped object on its side face near the corner, but is currently positioned closer to the to the top face of the object; the proxy, however, remembers the original point of contact and so the user feels a restoring contact force perpendicular to the side face. Similarly, on the right, the endpoint has in fact completely penetrated and exited a thin triangular object, but the user still feels the correct force which seeks to return the endpoint to the virtual object surface.

This form of normal contact force alone is sufficient to support push-type behaviors in a dynamic world. When a user presses into a dynamic rigid body, they feel a resisting force proportional to the depth of penetration into the object. An equal and opposite force is applied to the object on the next dynamics timestep, causing the body to accelerate away from the contact. Over a series of timesteps, this produces a reasonable sensation of pushing an object with mass about the world. In a similar interaction, the user feels the weight due to gravity when lifting an object from below; if the user holds the device endpoint at a steady vertical position, the downard gravity force on the object and the upward reaction force from the haptic device balance at a certain depth of device penetration, producing a constant downard weight force on the device endpoint.

In friction-free models, the proxy object moves along the object surface so that contact forces are always applied in the direction of the surface normal (ie: the proxy is set at the begining of each update to be the orthogonal projection of the device endpoint onto the object surface). This ensures that only normal contact forces are applied; however, frictional forces may be simulated by impeding the motion of the proxy object on the surface so that a tangential force component is felt by the user.

6.4.2 Friction Forces

HL provides methods for setting static and dynamic frictional coefficients that apply to the current proxy-object contact. Unfortunately, the nature of how HL calculates frictional forces is only lightly documented. Based on experimental investigation, the HL friction forces do produce slip-stick behavior, as expected, during relative motion of the contact point on the object surface. Additionally, the frictional force apparently scales in magnitude with increased depth of penetration of the device endpoint (thus, it uses some form of Coulomb friction). Specific details beyond these are not readily available. Noting HL's use of the point-proxy convention, we may suppose its friction model is similar to the god-object sliding model described in [Harwin and Melder 2002], but this cannot be verified.

Difficulties arose when combining HL's frictional forces with the rest of the dynamic environment simulation in Dihaptic. We hypothesize that this occurs because of the unintegrated nature of the frictional force. For example, if we are holding a rigid box at a constant position using a pinch grip of the sides of the box, it must be the case that the vertical load force (summed between the two contact points) equals the weight of the box and that the two horizontal grip forces sum to zero. The load forces are provided entirely by friction at the contacts in this instance; assuming no slipping of the contacts, we must therefore constrain the frictional forces to be equal to the object weight. However, by design, HL has no notion of gravity or other dynamic properties of the environment; so, it calculates frictional force in a manner that is unaffected by the described constraint. Any vertical motion of the box relative to the contacts will cause HL to produce a frictional force in the opposite direction, which may cause the lifted box to reverse the direction of its motion on the next simulation timestep. Over successive timesteps, this causes visible oscillations of the vertical position of the box along with varying frictional forces felt via the haptic interface. Of course, such instability runs counter to the expected state of a static grasp scenario.

Thus, though using the HL API yields high quality normal contact forces and reasonable frictional feedback when sliding along a static object (eg: walls, floor), we have been unable to successfully integrate its frictional reaction forces with the dynamic environment simulation. Since accurate frictional behavior is essential for gripping, lifting, and other object manipulation capabilities desired in Dihaptic, this issue eliminates the HL API as an acceptable haptic rendering method.

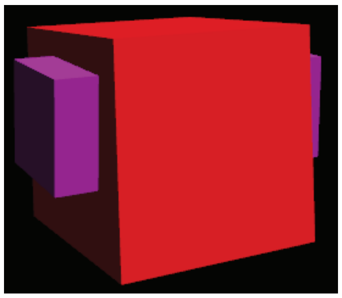


Figure 5: *Changing the haptic proxy from an infinitesimal point into boxes (shown in purple) or other rigid bodies allows us to restrict rotation of a lifted object (red).*

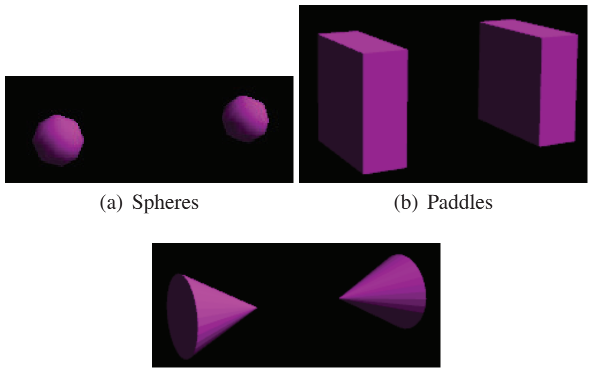

(a) Spheres (b) Paddles (c) Cones

Figure 6: *Examples of different proxy body types. The specific body type shapes how the user interacts with the dynamic environment.*

6.5 HD Implementation

The HD API is an alternative method for simulating haptic interaction using the OpenHaptics framework. It provides methods to access the current device endpoint position as well as to specify directly the current force applied to the endpoint by the device motors. In contrast to the HL API, there is no built-in notion of a virtual proxy object or scene geometry; the approach to determining feedback forces are left to the client.

After reaching a developmental impasse described above in Section 6.4, Dihaptic was reformulated using the HD API. Although implementation is ongoing, current testing shows improved results when attempting grip-and-lift actions in the virtual world.

6.5.1 Proxy Objects

One issue associated with point-based haptic interactions is that two rigid contact points do not apply frictional torque to the object and thus do not constrain rotation of the object about the grasp axis (the line connecting the two proxies on the surface of the object) [Barbagli et al. 2004]. This may be solved using a soft-point contact, used by [de Pascale et al. 2006] and others, which applies a torque based on the angular distance between the current stiction angle and the device angle about the grip axis. Alternatively, we may use a finite-sized rigid body to represent the haptic proxy in the virtual world in order to constraint rotations. For example, if we attach large, flat boxes to the device position and constrain their orientation to a canonical direction, we may grasp and lift other objects in the scene using the boxes as grippers, as shown in Figure 5. In contrast with frictional forces produced by HL, which are unrestricted by dynamic constraints, using a proxy body allows us to determine appropriately constrained friction within the dynamic system during the same timestep as gravity and other forces (see Section 7 for dynamic frictional models considered for Dihaptic). Rotation of the lifted objects is prevented by virtue of the multiple stabilizing points of contact with each of the box proxies.

An added benefit when using rigid proxy bodies is that we may choose the shape of the body according to the requirements of the experiment. Figure 6 shows but a few possible shapes for the proxy bodies. The body may be shaped as some arbitrary virtual tool (note that, currently, concave proxy bodies are approximated using simpler convex hulls).

The coupling of the virtual proxy object with the device endpoint is accomplished via a dual damped spring system, which provides a stable mechanism for exchanging forces between the dynamic virtual world and the haptic device [SensAble Technologies 2008]. A haptic spring force is applied to the device endpoint that pulls it toward the center point (local origin) of the proxy body; likewise, a virtual spring is applied to the proxy object that pulls it toward the current device endpoint. The endpoint force $\mathbf{F}_{device}$ is set according to:

$$\tilde{\mathbf{F}}_{device} = k_d(\mathbf{x}_{body} - \mathbf{x}_{device}) + c_d(\dot{\mathbf{x}}_{body} - \dot{\mathbf{x}}_{device})_{\parallel} \quad (1)$$

$$\mathbf{F}_{device} = \alpha\tilde{\mathbf{F}}_{device} + (1-\alpha)\mathbf{F}_{device,old} \quad (2)$$

Here, k_d and c_d are spring constant and damping parameters, respectively, and $\mathbf{x}_{device}$ and $\mathbf{x}_{body}$ are the world-space positions of the device and proxy bodies. $\mathbf{F}_{device,old}$ is the force sent to the device on the previous update, and $\alpha \in [0, 1]$ is a smoothing parameter which prevents sudden changes in device force that manifest as mechanical "buzzing"; currently, $\alpha = 0.2$. Note that we only apply velocity damping in the direction parallel to the line between the device endpoint and the proxy body. Naturally, $\mathbf{F}_{device}$ must be scaled and rotated into the device's local coordinates before being sent to the device.

The corresponding virtual force applied to the proxy body, $\mathbf{F}_{body}$, is defined as:

$$\mathbf{F}_{damping,\parallel} = c_{b,\parallel}(\dot{\mathbf{x}}_{device} - \dot{\mathbf{x}}_{body})_{\parallel} \quad (3)$$

$$\mathbf{F}_{damping,\perp} = c_{b,\perp}(\dot{\mathbf{x}}_{device} - \dot{\mathbf{x}}_{body})_{\perp} \quad (4)$$

$$\mathbf{F}_{body} = k_b(\mathbf{x}_{device} - \mathbf{x}_{body}) + \mathbf{F}_{damping,\parallel} + \mathbf{F}_{damping,\perp} \quad (5)$$

k_b is the spring constant, and $c_{b,\parallel}$ and $c_{b,\perp}$ are velocity damping parameters in the directions parallel and perpendicular to the line between the proxy body and and the device endpoint, respectively. $\mathbf{F}_{damping,\parallel}$, the perpendicular velocity damping force, is included in order to limit "orbiting" behavior of the proxy object about the device endpoint when the former acquires significant tangential velocity.

The spring constant k_b is increased significantly when the proxy body is not in contact with other objects, resulting in tight tracking of the device endpoint. This diminishes the haptic sensation of the proxy body's inertia during free motion, which is considered an undesired side effect in this project (for the same reason, we do not apply gravity to proxy bodies so that they have no weight).A more sophisticated solution may apply a velocity impulse to the proxy body that exactly moves it to the current device endpoint position, $\mathbf{x}_{device}$, though we must limit the impulse magnitude to prevent unstable dynamic behavior when the proxy body contacts another object in the scene.

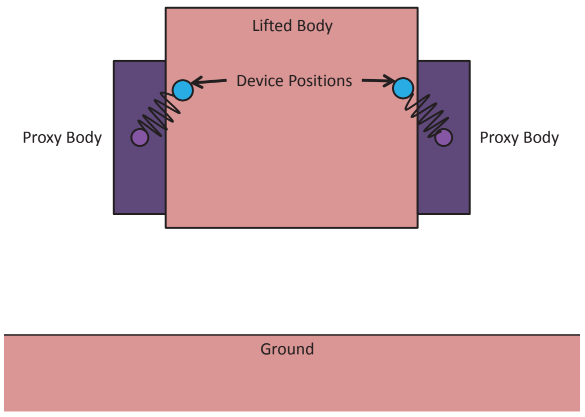
6.5.2 Grasp Stability

In order to illuminate the benefits of using finite proxy objects with dual spring interactions to track the device endpoint, we describe in detail the scenario of grasping a body using box-shaped proxy objects (with faces oriented parallel to the sides of the lifted body) and holding it statically at a position above the ground of the virtual environment. While this simple interaction example certainly does not generalize directly to all possible interactions (especially those with dynamic motions), it provides an easily analyzed setup that highlights the basic features which a haptic feedback system must correctly handle in order to be capable of simulating more complex interactions. As noted above in Section 6.4.2, this scenario is not handled well using HL friction forces; the lifted object and haptic feedback forces oscillate unpredictably.

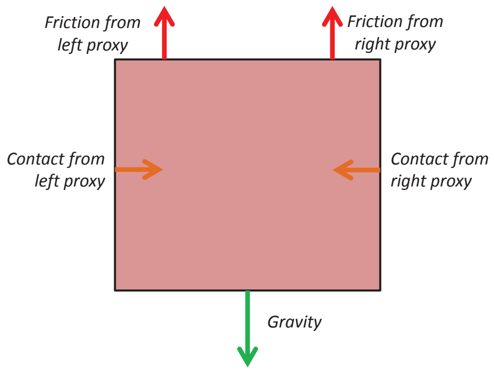
We propose two requirements for this scenario. First, it is expected that the proxy bodies as well as the lifted object will remain essentially stationary due to the balance of forces and torques; we assume the box was lifted from the ground and is now held statically in the air. Second, the user should feel haptic feedback forces that compel him/her to apply grip and load forces that counteract the weight of the lifted object (see [Johansson and Westling 1984] and [Westling and Johansson 1984] for analysis of these forces in live trials).

The scenario is depicted in Figure 7(a). The two proxy objects (in purple) are pinned to the sides of the lifted body (red) by the grip force applied by the user. The lifted body falls downward due to its weight, dragging the proxy bodies down as well due to the frictional force bewteen them. When the sum of upward frictional forces from the two proxy bodies matches the weight of the lifted body, the lifted body and proxies cease vertical motion. The lifted body has reached an equilibrium; it is horizontally balanced by the opposing grip of the proxy bodies and vertically balanced by its weight and the proxy body friction, as shown in Figure 7(b). The proxy bodies will be force-balanced as well if the virtual spring force $\mathbf{F}_{body}$ exactly counteracts the contact normal and frictional forces from the lifted body (Figure 7(c)).

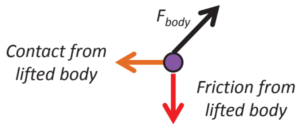
This will be the case if the user maintains the current position of the device endpoint as it was placed when the lifted body reached equilibrium (so that the spring force $\mathbf{F}_{body}$ remains constant). Because a corresponding $\mathbf{F}_{device}$ force is applied to the device endpoint in this state, the user must apply an equal and opposite $\mathbf{F}_{user}$ force in order to maintain this endpoint position (Figure 7(d)). Observing that this $\mathbf{F}_{user}$ is directed upward and inward toward the lifted object, we conclude that both scenario requirements are met. As expected, the lifted and proxy objects are only stable if the user supplies an inward grip force and an upward load force that match the weight and contact forces of the rest of the system.



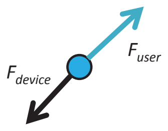
(a) Static grip of a lifted body. The dual virtual/haptic forces between device endpoints and proxy body centers are depicted as springs.



(b) Lifted Body Forces



(c) (Left) Proxy Body Forces



(d) Device Forces

Figure 7: *Illustration of forces involved when user grips a virtual object by its sides and holds it at a stationary location. At a specific separation distance of the proxy object centers and device endpoints, the forces on the lifted and proxy objects sum to zero, balancing these objects in a stationary position. Crucially, the user must apply an inward and upward force* $\mathbf{F}_{user}$ *to counteract the device endpoint force* $\mathbf{F}_{device}$ *which is active in this static configuration. Indeed, this is the expected force profile during static side grip (inward grip force, upward load force).*

7 Dynamics Models

We briefly describe the systems considered for resolving normal contact and friction impulses during the dynamic system timestep. Note that only the default Bullet and Stewart-Trinkle formulations are currently available in Dihaptic.

7.1 Bullet Default

The default contact resolution model in Bullet implements the model described in [Catto 2005]. At each timestep, it solves for the constraint force multiplier vector λ used in the constrained equations of motion:

$$M\dot{V} = J^T\lambda + F_{ext} \tag{6}$$

$$JV = \zeta \tag{7}$$

Here, M is the generalized mass matrix of the system, V is the generalized velocity, J is the constraint Jacobian, F_{ext} are external forces such as gravity, and ζ is a vector of bias values that allow constraints to do non-zero work if desired. The term $J^T\lambda$ represents the generalized constraint forces which must be determined.

Elements of λ may be clamped to an allowed range depending on the constraint type in order to limit constraint forces. For example contact normal constraint multipliers λ_n must be greater than or equal to 0 so that the applied force only applies a separating push to contacting bodies, and frictional constraints may use a force limited in magnitude to be at most the chosen coefficient of friction multiplied by the magnitude of the contact penetration.

By substituting in a first order approximation of the accelerations, $\dot{V} \approx \frac{V(t+1)-V(t)}{\Delta t}$, and eliminating $V(t+1)$, an explicit time stepping scheme is constructed [Catto 2005]:

$$JM^{-1}J^T\lambda = \eta = \frac{1}{\Delta t}\zeta - J(\frac{1}{\Delta t}V(t) + M^{-1}F_{ext}) \tag{8}$$

This is a linear system of the form $A\lambda = \eta$, so we may use standard solvers to find the values of λ, with the exception that each term λ_i must lie within its predetermined range of allowed values. Bullet uses a projected Gauss-Seidel (PGS) method to solve for λ, which simply applies Gauss-Seidel iterations that clamp λ_i values to their prescribed ranges at the end of each iteration. To accelerate convergence, the solver is "warm-started" using the solution values from the previous iteration. The number of iterations per simulation timestep is set by the client code to balance resolution accuracy with computational speed.

7.2 Stewart-Trinkle

Though it allows increasing PGS iterations per timestep, Bullet's default PGS solver is fundamentally built for predictable runtime and stability rather than physically accurate contact resolution. To evaluate whether an alternative contact/friction solver would improve grasp behavior in the HL version of Dihaptic, we integrated the Stewart-Trinkle formulation first described in [Stewart and Trinkle 1996]. An implementation derived from the existing Bullet framework is provided by [Nguyen and Trinkle 2010], which highlights its improved frictional accuracy using a simulated robotic grasping demonstration. Use of this formulation is provided as an option in the Dihaptic interface.

However, we currently still prefer the default Bullet model for several reasons. First, after implementing the new HD version of Dihaptic using finite proxy bodies, the default Bullet PGS solver now exhibits stable and reasonable friction forces during grip-and-lift scenarios. Second, the runtime for PGS is expected to be linear in the number of contacts and objects, whereas the available implementation of Stanley-Trinkle has a cubic expected runtime [Nguyen and Trinkle 2010]. Finally, in practice, the Stanley-Trinkle formulation has failed to converge entirely during the course of several scene interactions; for an experimental haptic environment, it is imperative that the world simulation remain stable and active in order to maintain smooth haptic force rendering.

7.3 Staggered Projections

Introduced in [Kaufman et al. 2008], the method of staggered projections computes a generalized total impulse applied at the current set of body contacts at each time step. It first finds a predictor velocity by integrating the dynamic system without contact or friction constraints. Then, it iteratively updates the estimated contact and friction impulse values until a desired residual precision is reached. At each iteration, it finds the estimated contact impluses by projecting the difference between the original prediction velocities and the previous friction impulses onto the space defined by the allowed contact impulse directions. The next estimate for the friction impulses are then found by projecting the difference between the original prediction velocities and the just-computed contact impulse estimates onto the space defined by a set of allowed friction directions. The final velocity for each body is the sum of the prediction velocity with the contact and friction impulses.

We were unable to produce a working implementation of staggered projections using contacts provided by Bullet within the current term. According to the authors, staggered projections shows noticably superior friction resolution to other methods when structural stability is highly dependent on frictional sticking, such as in a house of cards or interwoven flexible rods. As such, we may wish to implement a Bullet version of staggered projections (or switch Dihaptic to the ODE physical simulator, for which an existing staggered projections implementation exists) if we wish to perform experiments using environments that are more tightly constrained in a frictional sense than the current scenarios.

8 Results

The quality of the user experience when interacting with the virtual environment in Dihaptic varies in nature depending on whether we utilize the HL or HD implementations. If using the HL system, users feel accurate and smoothly rendered contact forces and friction with immovable objects such as the floor or walls. However, intuitive interactions are limited to pushing objects about the scene or lifting from below objects constrained in all but the vertical direction (eg: a bead on a vertical string). Manipulations that depend on frictional grip forces, such as lifting objects by grasping its sides or tossing objects, are not easily accomplished in the HL implementation.

Conversely, the HD version using finite proxy objects allows a user successfully to lift or toss objects using side grips; more complex interactions such as rotating a lifted object my be implemented by adding endpoint orientation encoders to the haptic devices. The haptic feedback forces when using the HD implementation, though, are still rough and unstable. If a user moves in free space too quickly, he will feel the inertial drag of the proxy object as it lags behind the device endpoint. When first contacting an object, the user may feel a “kick” away from the object surface as the device endpoint over-penetrates into the object, causing an overpowered haptic restoring force to be applied to the device. These and other haptic feedback issues must be resolved before the HD haptic experience matches the quality of the rest of the dynamic environment simulation.

9 Future Work

The current work agenda for Dihaptic focuses on increasing the haptic feedback quality for the HD implementation by integrating various practical mechanisms and modifications. These include methods for more smoothly increasing haptic feedback when first contacting an object, determining optimal spring force and damping parameters for the dual virtual/haptic spring system, and breaking dynamic system updates into a separate thread from other client code in order both to increase the simulation update rate and to make the time interval between updates more predictable. Once the HD haptic feedback is deemed acceptable, attention will be turned to providing better support for concave proxy bodies, determining a method for precisely calibrating the device axes, and fixing lower priority simulation issues.

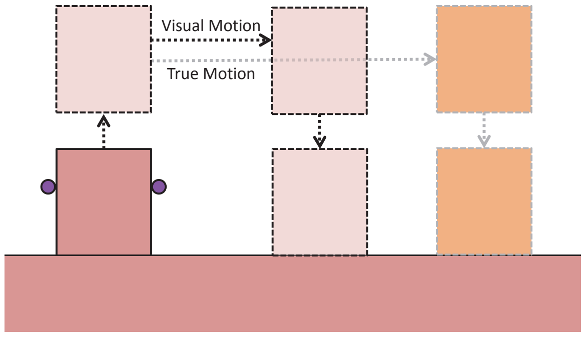
The Phantom devices provide force and positional readings with high spatial and temporal resolution, which facilitates recording of the forces applied by a user during interactions. Assuming Dihaptic succeeds in its primary goal of presenting a virtual tabletop workspace to the user that allows intuitive object manipulations, we may outline a number of sensorimotor experiments that utilize these recording capabilities.

Since we provide both visual and haptic sensations of the virtual environment, one branch of experiments may examine the relative contribution of these two senses to environmental perception. For example, suppose a subject is asked to grasp a block using a side grip, lift it to a specified height, carry it for some horizontal distance, and then lower it to the ground once more. After training in an unmodified environment, what is the threshold scaling of the visual translation of the block at which the subject may note a mismatch between visual and haptic cues (see Figure 8(a))? Does this threshold vary by dimension (ie: does the human sensitivity to visual scaling of motion depend on whether the motion is up/down, left/right, or forward/backward relative to the subject’s viewpoint)? What sort of visuomotor adaptation occurs when scaling the visual motion?

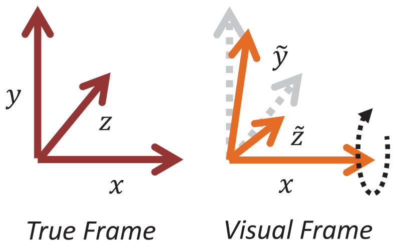
Suppose we rotate the visual axes relative to the motor axes instead. This is similar to the visuomotor rotations from [Krakauer 2009] and similar experiments, except we may examine how subjects adapt to rotations in a three-dimensional environment rather than only two dimensions (see Figure 8(b)). Does the adaptation rate to visuomotor rotations depend on whether we rotate the x, y, or z axes of the world? Do subjects adapt equally well to rotations about non-canonical axes as to rotations about the view frame axes?

Other experiments may explore the nature of sensory adaptation to changes in object mass. Imagine that we scale the apparent mass for a body, but only in some directions and not others (eg: load force is unchanged when lifting an object, but the force necessary to translate the object horizontally is increased/decreased). Do subjects adapt to the change in inertial properties? Is this rate of adaptation related in any notable manner to the rate of adaptation to visuomotor rotations?

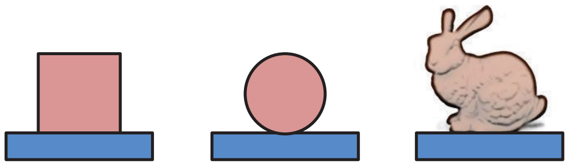
We may also envision experiments that investigate other aspects of perception and motor coordination besides adaptation. Suppose a subject is presented with a platform to grasp and lift. On each trial, a different object is placed on the platform, such as a cube, sphere, Stanford bunny, etc. (Figure 8(c)). If subjects are informed that all objects have the same density, what grip force magnitude will they



(a) If the visual motion of a grapsed block is scaled relative to the true motion, do subjects notice this scaling? Do they adapt to this change?



(b) How do subjects adapt to visuomotor rotations in a three-dimensional environment?



(c) Do subjects use visual information to accurately estimate object mass before gripping and lifting an object on a platform?

Figure 8: *Illustration of possible experiments using Dihaptic.*

employ during the various phases of lifting (see [Johansson and Westling 1984]). Are they able to accurately estimate volume (and hence weight) from the visual appearance of the object in order to employ a minimally sufficient grip force, or do they instead use a high-force grip to guarantee non-slippage instead? Does the behavior vary with the complexity of the object on the platform? Note that this sort of experiment may be conducted using a highly simplified virtual environment (point-based haptic proxy with platform motion constrained to vertical direction and canonical orientation), or it may be translated into a non-virtual experiment with physical objects and a platform device that encodes grip/load forces.

The above covers only a few of the possible experimental variations using the Dihaptic environment; many other combinations of visual and haptic transformations or scenario preparations may be considered.

References

ABDELGHANI, M., AND TWEED, D. 2010. Learning course adjustments during arm movements with reversed sensitivity derivatives. *BMC Neuroscience 11*, 1, 150.

ABDELGHANI, M. N., LILLICRAP, T. P., AND TWEED, D. B. 2008. Sensitivity derivatives for flexible sensorimotor learning. *Neural Comput. 20*, 8 (Aug.), 2085–2111.

ANG, Q.-Z., HORAN, B., NAJDOVSKI, Z., AND NAHAVANDI, S. 2011. Enabling multi-point haptic grasping in virtual environments. In *3D User Interfaces (3DUI), 2011 IEEE Symposium on*, 55 –58.

BARBAGLI, F., FRISOLI, A., SALISBURY, K., AND BERGAMASCO, M. 2004. Simulating human fingers: a soft finger proxy model and algorithm. In *Proceedings of the 12th international conference on Haptic interfaces for virtual environment and teleoperator systems*, IEEE Computer Society, Washington, DC, USA, HAPTICS'04, 9–17.

CATTO, E. 2005. Iterative dynamics with temporal coherence. *Technical Report* (February).

COUMANS, E., 2012. Bullet physics library. `http://bulletphysics.org`.

D'AVELLA, A., AND PAI, D. K. 2010. Modularity for sensorimotor control: Evidence and a new prediction. *Journal of Motor Behavior 42*, 6, 361–369.

DE PASCALE, M., FORMAGLIO, A., AND PRATTICHIZZO, D. 2006. A mobile platform for haptic grasping in large environments. *Virtual Reality 10*, 11–23. 10.1007/s10055-006-0026-6.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '06, 161–165.

HARWIN, W. S., AND MELDER, N. 2002. Improved haptic rendering for multi-finger manipulation using friction cone based god-objects. In *Proceedings of Eurohaptics Conference*.

INGRAM, J. N., HOWARD, I. S., FLANAGAN, J. R., AND WOLPERT, D. M. 2011. A single-rate context-dependent learning process underlies rapid adaptation to familiar object dynamics. *PLoS Comput Biol 7*, 9 (09), e1002196.

JOHANSSON, R., AND WESTLING, G. 1984. Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. *Experimental Brain Research 56*, 550–564.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers*, ACM, New York, NY, USA, SIGGRAPH Asia '08, 164:1–164:11.

KRAKAUER, J. W. 2009. Motor learning and consolidation: The case of visuomotor rotation. *Advances in Experimental Medicine and Biology 629*, 405–421.

LEE, J.-Y., AND SCHWEIGHOFER, N. 2009. Dual adaptation supports a parallel architecture of motor memory. *Journal of Neuroscience 29*, 33, 10396–10404.

NGUYEN, B., AND TRINKLE, J. 2010. dvc3d: a three dimensional physical simulation tool for rigid bodies with contacts and coulomb friction. *The 1st Joint International Conference on Multibody System Dynamics*.

PARK, J., AND YOON, Y.-L. 2006. Led-glove based interactions in multi-modal displays for teleconferencing. In *Proceedings of the 16th International Conference on Artificial Reality and Telexistence–Workshops*, IEEE Computer Society, Washington, DC, USA, ICAT '06, 395–399.

SCHWARTZ, E. L., Ed. 1993. *Computational neuroscience*. MIT Press, Cambridge, MA, USA.

SENSABLE TECHNOLOGIES, 2008. Openhaptics® toolkit version 3.0 programmer's guide. `http://dsc.sensable.com`.

SHADMEHR, R., AND MUSSA-IVALDI, S. 2012. *Biological Learning and Control: How the Brain Builds Representations, Predicts Events, and Makes Decisions*. MIT Press, Cambridge, MA, USA.

SMITH, M. A., GHAZIZADEH, A., AND SHADMEHR, R. 2006. Interacting adaptive processes with different timescales underlie short-term motor learning. *PLoS Biol 4*, 6 (05), e179.

STEWART, D. E., AND TRINKLE, J. C. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering 39*, 15, 2673–2691.

TRESCH, M. C., AND JARC, A. 2009. The case for and against muscle synergies. *Current Opinion in Neurobiology 19*, 6, 601 – 607.

TWEED, D. 2003. *Microcosms of the brain. What sensorimotor systems reveal about the mind*. University Press.

WANG, R. Y., AND POPOVIĆ, J. 2009. Real-time hand-tracking with a color glove. In *ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, SIGGRAPH '09, 63:1–63:8.

WEI, Q., SUEDA, S., AND PAI, D. K. 2010. Physically-based modeling and simulation of extraocular muscles. *Progress in Biophysics and Molecular Biology 103*, 2-3, 273–283.

WESTLING, G., AND JOHANSSON, R. S. 1984. Factors influencing the force control during precision grip. *Experimental Brain Research 53*, 277–284. 10.1007/BF00238156.

WOLPERT, D. M., DIEDRICHSEN, J. A., AND FLANAGAN, J. R. 2011. Principles of sensorimotor learning. *Nat Rev Neurosci 12*, 12 (Dec.), 739–751.

ZILLES, C. B. 1995. *Haptic Rendering with the Toolhandle Haptic Interface*. Master's thesis, MIT Department of Mechanical Engineering.