Improving Interactive Multisensory Simulation and Rendering Through Focus On Perceptual Processes

BY Timothy Edmunds

A dissertation submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
Graduate Program in Computer Science

Written under the direction of
Dinesh K. Pai

and approved by

New Brunswick, New Jersey

January, 2009
ABSTRACT OF THE DISSERTATION

Improving Interactive Multisensory Simulation and Rendering Through Focus On Perceptual Processes

by Timothy Edmunds

Dissertation Director: Dinesh K. Pai

In this dissertation we present a novel approach to the design of multisensory interactive applications. We develop methods that improve the effectiveness of this type of application by incorporating the existing understanding of the human perceptual system. We quantitatively demonstrate the validity of our techniques through user studies and laboratory measurement.

Explicitly including the human user as a part of the model for interactive application design, we propose a design approach for training simulators that augments the dynamical simulation of an interactive task (e.g., a surgical procedure) with feedback that highlights the aspects of the interaction that are perceptually pertinent for the purposes of training. We show how this type of augmentation can improve the training effectiveness of a simulator without necessitating more expensive rendering hardware.

To make our perceptually-based augmentation technique more useable by application designers, we propose a decomposition approach to simplify the
general process of developing the appropriate augmentation for a training simulator. We validated our approach by applying it to the design of a training simulator for a haptic search task; we conducted a user study that found a statistically significant improvement in the training effectiveness of the augmented simulator vs. an unaugmented simulator. We propose specific guidelines for how existing psychophysical experimental results can be used to build augmentations for training simulators.

We also developed a novel rendering architecture for distributed interactive applications that is suitable for the type of perceptually augmented simulator described above. Our architecture uses prediction of perceptually pertinent interaction events to achieve application latency and asynchrony bounds that can be constrained to within psychophysically established thresholds.
Preface

Portions of this dissertation are based on work previously published or submitted for publication by the author [Edmunds and Pai 2006, 2008, 2009].
Acknowledgements

There are many people who have contributed in making my journey possible, and for whose support I am grateful. Chief among these is my research advisor, Dinesh Pai, whose guidance has been invaluable throughout my research career. I also appreciate the advice given by the rest of my thesis committee; their comments have helped me improve this work and helped identify avenues for future research. I am indebted to Danny Kaufman, Warren Code and Amy Goldlist for their patient proof-reading; I look forward to returning the favour.

The members of the Multisensory Computation Laboratory have all helped make my experience at Rutgers successful and more enjoyable.

Chris Mansley has been both a great friend and an irreplaceable local arrangements facilitator, without whose efforts I could not have entered the country to defend my thesis, nor left the country and submitted my dissertation on time.

Finally, I am supremely grateful to my parents, to my sister, and to Bethany Leffler, whose support has been unwavering and unbounded.
Dedication

For my family, old and new.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Preface</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>v</td>
</tr>
<tr>
<td>Dedication</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Contributions</td>
<td>4</td>
</tr>
<tr>
<td>2. Interaction Feature Augmentation in Haptic Training</td>
<td>6</td>
</tr>
<tr>
<td>2.1. Introduction</td>
<td>6</td>
</tr>
<tr>
<td>2.1.1. Related Work</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2. Outline</td>
<td>10</td>
</tr>
<tr>
<td>2.2. Task Description</td>
<td>11</td>
</tr>
<tr>
<td>2.3. Simulation Description</td>
<td>13</td>
</tr>
<tr>
<td>2.3.1. Full Stiffness Simulator</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2. Degraded Stiffness Simulator</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3. Augmented Low Stiffness Simulator</td>
<td>18</td>
</tr>
<tr>
<td>2.4. Experiment</td>
<td>21</td>
</tr>
<tr>
<td>2.4.1. Baseline Block</td>
<td>22</td>
</tr>
<tr>
<td>2.4.2. Training Block</td>
<td>23</td>
</tr>
</tbody>
</table>
3. A Method for Automatically Designing Interaction Feature Augmentation

3.1. Introduction

3.2. Background and Approach

3.3. Haptic Search Task

3.3.1. Stimulus Design

3.3.2. Stimulus Interaction

3.4. Augmentation

3.4.1. Task Decomposition

3.4.2. Subtask Augmentation

3.5. User Study

3.5.1. Results and Analysis

3.5.2. Discussion
3.6. Conclusions ................................................. 61
  3.6.1. Simulator Design Guidelines ....................... 63

4. Event-Based Rendering ..................................... 65
  4.1. Introduction .............................................. 65
    4.1.1. Related Work ......................................... 68
  4.2. Analysis of Latency in Distributed Multisensory Environments 72
  4.3. Architecture ............................................. 75
    4.3.1. Programming Model ................................... 75
    4.3.2. Synchronization ....................................... 76
    4.3.3. Event Scheduling ..................................... 76
  4.4. Evaluation Experiment .................................. 77
  4.5. Example Application ..................................... 80
  4.6. Conclusions .............................................. 84

5. Conclusion .................................................. 87

Appendix A. User Study Protocol .............................. 90
  A.1. Instruction Session ...................................... 90

Appendix B. Individual Haptic Search Results .................. 105
  B.1. Task Success ............................................ 105
  B.2. Scan Subtask ............................................ 105
  B.3. Identification Subtask .................................. 106
  B.4. Shape Assessment Subtask ............................... 106

Bibliography ................................................... 123

Vita ............................................................. 131
List of Tables

3.1. Texture Parameters ........................................... 41
3.2. Trial Parameters ............................................. 54
List of Figures

2.1. Experimental task setup ................................. 11
2.2. Task procedure ........................................... 12
2.3. Pin insertion tools ....................................... 13
2.4. Force/motion profiles for the different simulators .... 19
2.5. Force profile for the augmentation pulse ................ 21
2.6. Experimental apparatus .................................. 22
2.7. Task improvement by group ............................. 24
3.1. Example stimulus height field ........................... 37
3.2. Environment geometry ................................... 38
3.3. Scene element textures .................................. 40
3.4. Example stimulus ........................................ 42
3.5. Example Gabor filter ..................................... 48
3.6. Example processed height field ......................... 49
3.7. Task improvement by group ............................. 55
3.8. Scan subtask results ..................................... 57
3.9. Identification subtask results ........................... 58
3.10. Shape assessment subtask results ...................... 59
4.1. The HAVEN .................................................. 66
4.2. Application data timeline ................................ 73
4.3. Generic architecture ...................................... 75
4.4. Audio recordings of the evaluation application .......... 78
4.5. Multisensory PONG! ...................................... 81
4.6. Distributed PONG application structure ........................................ 82
A.1. User Study Instructions .......................................................... 91
B.1. Individual Task Success - Unaugmented .............................. 107
B.2. Individual Task Success - Augmented ................................. 109
B.3. Individual Scan Performance - Unaugmented ........................ 111
B.4. Individual Scan Performance - Augmented ........................... 113
B.5. Individual Identification Performance - Unaugmented ........... 115
B.6. Individual Identification Performance - Augmented ............... 117
B.7. Individual Shape Assessment Performance - Unaugmented .... 119
B.8. Individual Shape Assessment Performance - Augmented ......... 121
Chapter 1

Introduction

Applications that allow users to interact with simulated virtual environments are used in a variety of domains, from video games to training systems for surgical procedures. These interactive simulations require both the responsiveness needed for real-time systems and the stability and fidelity needed for dynamical simulation, particularly for applications like flight simulators or surgical trainers. However, since these interactive simulations are intended for use by human operators, these requirements can be reevaluated in the context of the properties and capabilities of the human perceptual and motor-control systems. If we extend our model of an interactive application to include the user as well as the simulated system, then the parameters of the human user (along with the purpose of the application) determine the quantitative constraints of responsiveness and fidelity required for an effective interactive simulation. The thesis of this dissertation is that by leveraging an understanding of the human perceptual system to focus simulation and rendering efforts on pertinent interaction features, we can improve the effectiveness of interactive simulations.

A traditional approach to dynamical simulation is to evaluate a model that provides a high-fidelity representation of the behaviour of the objects in an environment, or at least some features of that behaviour (e.g., elasticity, friction, fluid dynamics). This approach attempts to maintain high fidelity (at least in the quantities of interest, such as object motion, surface annealing, or sounds
resulting from object interaction) throughout the simulation. Similarly, real-
time systems are designed to provide constant guarantees about particular 
temporal features, such as system latency, update frequency, or cross-thread 
asynchrony. A basic approach to the design of interactive simulations is to 
combine these two approaches, building systems that combine user input with 
the dynamical simulation of the environment and produce high-fidelity output 
that conforms to constant thresholds of latency and update rate. This approach 
can be prohibitively expensive in terms of the computational costs required to 
simulate complex environments at interactive rates, and the hardware costs 
of devices capable of rendering high-fidelity representations of the computed 
output to the user.

A specific topic that we address in support of our thesis is the use of per-
ceptually-informed augmentation of haptic simulations for the purposes of 
surgical training. Dynamical simulations are often evaluated based on their 
perceived realism, or (when possible) on how accurately they reproduce mea-
sured real-world phenomena. In the context of surgical training, however, the 
dynamical simulation is merely a means to an end — namely improved reli-
able performance of the surgical task in question. Therefore a more appropriate 
gauge of the effectiveness of the simulation is how well it imparts skill to the 
user. By decoupling simulation effectiveness from simulation fidelity, there is 
scope for a broader array of approaches to the design of dynamical simulations. 
The approach explored in this dissertation is to focus simulation and rendering 
resources on features of the interaction between the user and the virtual envi-
ronment that are most perceptually pertinent to the user (given the context of 
the application).

In Chapter 2 we show how the targeted application of haptic rendering 
capabilities to specific aspects of a surgical training type of simulation can im-
prove the training effectiveness of the simulation without requiring an increase
in the capabilities (and hence, cost) of the rendering hardware. We describe a training simulator of a surgical task that we designed using a combination of measurement-based dynamical modelling and perceptually pertinent augmentation. We evaluated the effectiveness of this simulator (as compared to a more standard simulation approach) by performing a user study to examine the relative effectiveness of different simulators for training naive subjects for the real task. Our results showed a statistically significant difference between the effectiveness of our augmented simulator and the control simulator.

Building on our findings that the training effectiveness of a dynamical simulation can be improved by augmenting perceptually pertinent aspects of the interaction, Chapter 3 addresses the problem of developing a general-purpose method for designing simulators that incorporate perceptually-driven augmentation. We explore how the decomposition of a simulated task (again guided by established knowledge about the human perceptual system) can make the problem of automatically generating this type of augmentation of interaction features more tractable. We show how domain-specific decomposition of a task into subtasks, combined with existing knowledge about the psychophysical characteristics of different interaction strategies, allows the design of a training simulator to be structured around the augmentation of perceptually pertinent aspects of the task. We evaluate this approach through another user study in which a more complex task is decomposed into automatically identifiable subtasks, and simulation during performance of these subtasks is augmented in accordance with the perceptual characteristics of the procedures used to perform the subtask. Our results indicate that this approach of task decomposition and leveraging of generalizable psychophysical findings can successfully be used to develop augmented simulations in a structured fashion.

The third major area of work described in this dissertation is the problem of
effectively rendering feedback to the user in interactive simulations. In many cases, this problem is primarily a hardware design problem; more capable output devices are able to improve the fidelity between the signal computed by the simulation and the user’s perception of the signal. In Chapter 4 we examine the more specific domain of distributed multisensory interactive systems, where multiple modalities of output are simulated and rendered to the user in a distributed-computing environment. In this context, perceptual issues of system latency and multi-modal asynchrony present significant obstacles to the design of effective interactive simulations. As with our approach to the design of interactive simulations for surgical training, we leverage an existing understanding of the human perceptual system to provide guidelines that inform the design of a rendering architecture. We develop a general-purpose rendering architecture that allows psychophysically established perceptual thresholds for latency and asynchrony to be incorporated into the distributed application design in order to ensure an effective user experience. We evaluate this architecture by using it to implement a multisensory test application whose latency and asynchrony we can measure accurately; we establish that the application is capable of limiting latency and asynchrony to within relevant perceptual thresholds. We also show how our architecture (along with psychophysically verified thresholds) can be used to design more complex distributed multisensory interactive applications by describing our implementation of an example application that combines auditory, visual, and haptic modalities distributed across a multi-system networked rendering environment.

1.1 Contributions

This dissertation describes a novel approach to the design of interactive simulation and rendering systems. We present an argument for the structuring
of simulator design around perceptually pertinent features of the interaction between user and simulator. We support this argument with new human-subject findings, interpreted with a focus on simulator effectiveness as gauged by task improvement. Based on our findings, we propose specific guidelines for the design of surgical simulators, suggesting how existing psychophysical experimental results can be incorporated into the simulation design. We also present a novel distributed rendering architecture that is compatible with the above focus on perceptually pertinent interaction events. This architecture illustrates how results from the psychophysical literature can be used to design interactive applications that achieve latency and asynchrony within perceptual thresholds.
Chapter 2
Interaction Feature Augmentation in Haptic Training

2.1 Introduction

One of the primary uses of interactive simulation is for training in tasks where repeated practice is necessary for competent execution of the real task, and real world training opportunities are costly, dangerous, or unethical (e.g., aircraft piloting, surgery, nuclear reactor monitoring). In this chapter, we look specifically at using interaction-feature-based augmentation to enhance the effectiveness of interactive simulation for the purposes of training. Specifically, we address the problem of creating simulations that are effective at developing skilled performance of tasks that require neuro-muscular control as well as cognitive awareness of appropriate situational response. An example of this class of task is surgical procedures, where a surgeon must develop both high-level decision making skills to be able to decide what actions to take over the course of a surgery, and also the low-level motor control necessary to perform those actions. This type of training most requires effective haptic simulation, since appropriate neuro-muscular control cannot easily be imposed by training that does not provide force-feedback to the neuro-motor system.

The thesis presented here is that by leveraging an understanding of the human perceptual system to focus simulation and rendering efforts on pertinent interaction features, we can improve the effectiveness of interactive simulations. In this chapter, we address this thesis by showing how the effectiveness
of haptic training simulations can be improved (in terms of the skill transfer effected by the simulation) by augmenting a basic haptic simulation to highlight the perceptually pertinent aspects of the interaction.

In this chapter, we address the issue of how an interaction-feature-based approach can help haptic skills be effectively learned from renderings on low-fidelity haptic devices. There has been considerable research on rendering specific haptic features or events (which we review below) but relatively little is known about rendering complex tasks that require skilled performance and training. Ideally we would like skilled performance of a real world task to improve after training on a virtual task with a haptic device.

One way to achieve effective skill transfer is to focus, not on the raw sensory data, but rather on the sequences of perceptual events that occur during task performance. There is some evidence that the raw sensory information is not experienced directly but is quickly integrated into perceptual features that separate small “action phases” [Johansson 1996]. Even in the simple skill of lifting an object, the task involves approach, making contact, pre-loading grip forces, and lift off. Transitions between phases in the sequence are made based on the sensory signals. In this case, the primary role of sensory signals is event detection, that is, marking the start of the next action phase and the corresponding changes in neural control.

A more complex skill on which we focus in this chapter is inspired by a surgical procedure: bone-pin placement. In this surgical procedure, the surgeon stabilizes a fractured long bone by screwing a sharpened metal pin through the bone. This involves driving the pin through the hard outer cortex of the bone, then through the spongy cancellous bone, then through the far cortical layer. Throughout these material transitions, the surgeon must maintain a controlled movement of the pin along its trajectory so as to avoid damage to the bone or soft tissue.
We hypothesize that over the course of such a procedure, there are specific events that are most perceptually significant when learning how to successfully perform the procedure. If these events can be identified, then the simulation can be designed to focus on rendering these events with high fidelity, without necessarily requiring that the entire simulation provide that level of fidelity. Such a focus could allow a simulation to be an effective trainer without requiring high-cost hardware.

A key issue in the design of training simulators is the selection of feedback to be provided to the trainee. One assumption that guides many simulator designs is that a high-fidelity simulator will generate a training effect; i.e., if the simulator is sufficiently similar to the real task, the trainee’s task performance will improve after training. However, there is also evidence that the skill transfer effectiveness of a simulator can be improved by controlled deviation from the real task. Our technique is compatible with both of these approaches to skill transfer, since it can be used to increase the fidelity of pertinent aspects of the interaction, or to exaggerate interaction features in a deliberate deviation from the real task.

2.1.1 Related Work

There has been considerable work on the perception and rendering of haptic features. We broadly classify these features into two categories: object features and interaction features. Object features include shape [Robles-De-La-Torre and Hayward, 2001], texture (which we take to include both friction and roughness [Lloyd and Pai, 2001; Lederman and Klatzky, 2004]) and elasticity [Srinivasan et al., 1996]. Interaction features include making and breaking contact, and relative motion of the contact surface (including sliding, rolling, and sticking). Even though much of the existing literature on haptic perception
(e.g., [Stein and Meredith, 1993; Calvert et al., 2004]) does not make a clear distinction between the two categories, there is evidence that these are used in very different ways in skilled human performance [Johansson, 1996].

Event-based haptic rendering is one approach that focuses on the simulation fidelity of particular events. [Salcudean and Vlaar, 1997] found that a braking force pulse increases the perceived stiffness of a virtual surface upon penetration. [Constantinescu et al., 2004] extended this braking force approach to create impulsive forces in response to multi-body collision events. A primary obstacle to haptic rendering focusing on discrete events is that traditional closed-loop controllers often do not operate at the high frequencies (up to 1 kHz) that mediate human perception of discrete events. By using brief open-loop high-frequency playback triggered by contact events, [Hwang et al., 2004] were able to reduce stopping distance and increase the effective stiffness of virtual surfaces.

The majority of research into event-based haptic feedback focuses on increasing the fidelity of stiff surface tapping. The effectiveness of event-based feedback has been gauged both by measurement of quantifiable properties (effective stiffness, rate-hardness, stopping distance) [Hwang et al., 2004], and by single-blind studies of user ratings of realism [Kuchenbecker et al., 2006; Okamura et al., 2001].

Haptic interaction almost invariably involves contact, which produces correlated sounds and visual deformation in addition to forces. Multisensory rendering of these contact events can contribute strongly to the correct perception of the interaction; see [Pai, 2005] for a review.

One approach to the problem of skill transfer in manual tasks that goes beyond the straightforward maximization of fidelity is the focus on differentiation of perceptual invariants [Lintern, 1991]. In this model, the emphasis is on developing the trainee’s sensitivity to changes in relationships between
variables in the environment, in particular, those relationships that are invariant throughout successful execution of the task (e.g., the ratio of projected runway length to projected breadth of the end of the runway during aircraft landing [Mertens 1981]). Our approach is similar, in that we are exploiting the fact that skill transfer can be improved by controlled deviation from task similarity [Wightman and Lintern 1985], and that we are seeking to identify features of a task that are most perceptually significant for skill transfer. The focus on perceptually pertinent events is also complementary to a focus on perceptual invariants; where the latter seeks to sensitize the trainee to perceptual phenomena that occur during each phase of a task, the former seeks to increase sensitivity to the events that signal transitions between phases.

2.1.2 Outline

We conducted a user study to investigate the effectiveness of event-based augmentation for simulator training. We developed a task that mimics the characteristics of the real-world surgical task. We compared the training effectiveness of quasi-static closed-loop and event-augmented haptic simulations of this surrogate task. Our results showed that perceptual augmentation of a low fidelity haptic rendering produced measurable improvements in skill transfer.

The remainder of this chapter is organized as follows. In Section 2.2 we describe the surrogate surgical task that we developed for our investigation. The dynamical simulation that we created for the surrogate task is presented in Section 2.3. In Section 2.4 we describe the methodology of our user study. The results of the study are presented in Section 2.5. In Section 2.6 we draw conclusions about the effectiveness of event-based simulator augmentation in training for surgical tasks.
2.2 Task Description

The task we developed for this experiment is a surrogate for the more complex task of bone-pin placement. Our experimental task focuses on the requirement that the pin must be inserted with sufficient control to prevent excessive motion of the pin when the resistance changes as a result of a transition from one material to another. The subjects’ task was to drive a bone-pin (3 mm in diameter with a sharpened tip) through a slab of polystyrene (a surrogate for cortical bone) until the pin’s tip emerged into an air gap (whose lower resistance parallels the low density and low strength of cancellous bone) on the far side of the slab (see Figure 2.1).

The polystyrene slab was 24 mm thick, and its far side was laminated with card stock to increase the force required to puncture through to the air gap behind the slab. The air gap was 13 mm thick; beyond it lay another slab of
Figure 2.2: Task procedure.  

(a) The task starts with the tip of the pin resting on the surface of the polystyrene.  
(b) The subject must drive the sharpened pin through the first polystyrene slab.  
(c) The subject must stop before the pin touches the second slab.  
(d) If the pin penetrates the second slab, the task is failed.

of polystyrene. Both slabs were mounted behind a layer of plywood; a guide hole was drilled through the plywood to govern the pin’s insertion location and direction of motion. The entire assembly was arranged at a $45^\circ$ angle 0.9 m above the ground.

Since our haptic rendering hardware did not support the generation of torque about the axis of insertion, we removed the screwing component of the pin insertion — the pin was pushed through the polystyrene without a screwing motion (this is possible because the polystyrene slab is weaker than cortical bone).

To successfully complete the task, the subject was required to push the bone pin through the first polystyrene slab, but stop before the tip of the pin reached the second slab (see Figure 2.2). To emphasize efficient performance of the procedure, a time limit was also imposed. The subject was required to complete this insertion within 3 seconds (starting with the pin inserted through the guide hole with its tip resting on the surface of the first slab). The subject held the bone pin in a T-handled pin vise (see Figure 2.3a).
Figure 2.3: Pin insertion tools (at \(~0.3\times\) scale). (a) The bone pin is held in a T-handled pin vise. (b) To measure the force characteristics of the task, a pin holder was built that incorporated a force sensor and motion-tracking markers.

2.3 Simulation Description

To create a virtual model of the task apparatus, the real task’s mechanics were measured by performing the task with an instrumented version of the bone-pin holder (see Figure 2.3b). The instrumented holder used a 6-axis force/torque sensor [ATI] and a motion tracking system [Vicon] to simultaneously record the position of the pin’s tip and the forces exerted on the pin. These recordings were used to guide the design of a haptic simulation whose force characteristics paralleled those of the real materials.

Analysis of the force/motion profile showed that the force required to penetrate the polystyrene rose approximately quadratically with penetration depth,
and that non-penetrating movement (i.e., movement that did not alter the structure of the material) was resisted by a force that was approximately linear with penetration depth.

We implemented our haptic simulation using a dynamic proxy whose behaviour is similar to that described by Mitra and Niemeyer [2004]. The tip of the bone-pin is represented by a proxy, whose position, $x_p$, is coupled to a user-controlled master, $x_m$. The proxy’s motion is constrained to one dimension (representing the movement of the bone-pin’s tip along the axis of insertion).

We model the slab of polystyrene as an interval ($slab_{top}$ to $slab_{bottom}$) along this single dimension; for convenience, and without loss of generality, we set $slab_{top} = 0$ and $slab_{bottom} = -slab_{thickness}$. As the user inserts the bone-pin into the polystyrene, the structure of the environment is changed — a channel has been carved down to the point of furthest penetration. To model this dynamic aspect of the environment, we define a variable, $slab_{top} \geq x_c \geq slab_{bottom}$, representing the maximum depth to which the proxy has carved.

The position of the master can be described by one of three cases:

**Non-Contact**: when $x_m > slab_{top}$, the master (and the proxy) is not in contact with the slab.

**Contact**: when $slab_{top} \geq x_m \geq x_c$, the master (along with the proxy) is inside the slab, but not penetrating beyond the maximum depth carved so far.

**Penetration**: when $x_m < x_c$, the master is penetrating the virtual surface. Note that when $x_c = slab_{bottom}$ (i.e., the pin has carved all the way through the slab) the proxy is no longer in the **Penetration** case — it is handled by the **Contact** case.

We will describe the dynamics of our simulation separately for each of these
cases. Note that we structure our dynamics so that the behaviour of our system is continuous across the case boundaries.

In the **Non-Contact** case, the proxy moves with the master, and no forces are generated:

\[
\begin{align*}
  x_p &= x_m \quad \text{(2.1)} \\
  f_m &= 0 \quad \text{(2.2)}
\end{align*}
\]

In the **Contact** case, to mimic the measured linear increase in resistance with pin depth, we attenuate the motion of the proxy with a factor that increases linearly with depth:

\[
\begin{align*}
  \Delta x_p &= (1 - \alpha)(x_m - x_p) \quad \text{(2.3)} \\
  \alpha &= \alpha_{\text{max}} \left( \min \left( 1, \frac{\text{slab}_{\text{top}} - x_p}{\text{slab}_{\text{thickness}}} \right) \right) \quad \text{(2.4)}
\end{align*}
\]

where \( \alpha_{\text{max}} \) is the maximum amount of damping (between 0 and 1) that occurs when \( x_p \leq \text{slab}_{\text{bottom}} \). \( \Delta x_p \) is the one-time-step change in the position of the proxy; to match our measured forces, we used the value \( \alpha_{\text{max}} = 0.9 \) for our update rate of 1 kHz. The force generated at the master is based on a spring coupling between the proxy and the master:

\[
\begin{align*}
  f_m &= k(x_p - x_m) \quad \text{(2.5)}
\end{align*}
\]

In the **Penetration** case, we build on the traditional dynamics for stiff surfaces, where the proxy stays on the virtual surface and exerts a force on the master:

\[
\begin{align*}
  f_m &= \begin{cases} 
  k(x_c - x_m), & x_m < x_c \\
  0, & x_m \geq x_c
\end{cases} \quad \text{(2.6)} \\
  x_p &= \begin{cases} 
  x_c, & x_m < x_c \\
  x_m, & x_m \geq x_c
\end{cases} \quad \text{(2.7)}
\end{align*}
\]
In this paradigm, the proxy can also be considered to be applying a force \( f_p = -f_m \) on the virtual surface. In our model, the position of the virtual surface is dynamic, so we compute the force applied to the virtual surface as:

\[
\mathbf{f}_p = \begin{cases} 
  k(x_m - x_c), & x_m < x_c \\
  0, & x_m \geq x_c 
\end{cases} 
\]  

(2.8)

Unlike the traditional stiff surface, our surface must yield to allow the proxy to be moved deeper and deeper into the slab. However, there is some minimum force that the surface is able to resist without its structure changing. Furthermore, our measurements showed that this minimum force should increase quadratically with depth. We define a quantity that specifies what force can be exerted on the surface without carving for a given value of \( x_c \):

\[
f_{\text{resist}} = a_0 + a_1 x_c + a_2 x_c^2
\]

(2.9)

(fitting to our measured forces yields \( a_0 = 0.9 \) N, \( a_1 = -9.6 \) N/m, \( a_2 = 184.9 \) N/m\(^2\) for our material). If this threshold is exceeded (i.e., \( \| \mathbf{f}_p \| > \| f_{\text{resist}} \| \)), then the proxy has carved the surface, and \( x_c \) must be adjusted. The carving should not move past the master position, and the amount of carving should be proportional to the force applied. We compute the new value of \( x_c \) to satisfy these conditions:

\[
\Delta x_c = \beta (x_m - x_c)
\]

(2.10)

\[
\beta = \min(1, \gamma (\| \mathbf{f}_p \| - \| f_{\text{resist}} \|))
\]

(2.11)

where \( \gamma \) is a tuneable parameter to control the rate of carving (all of our simulations used \( \gamma = 10 \) N\(^{-1}\)). As with \( \Delta x_p \) above, \( \Delta x_c \) is the one-time-step change in the value of \( x_c \).

Since the value of \( x_c \) has been adjusted, the proxy is no longer fixed at its old position. We move the proxy to the new position of the surface, and use
the new position to compute the master force:

\[ x_p = x_c \]
\[ f_m = k(x_p - x_m) \]  

(2.12)  
(2.13)

The simulation was implemented using a dual 2.0 GHz Xeon workstation with 1 GB of RAM, and a PHANTOM Premium 1.0 [SensAble] haptic device with 6 degrees of freedom in position input and 3 degrees of freedom in force output. The same T-handled pin vise used in the real task was attached to the PHANTOM stylus so that it could be grasped in the same way as when performing the real task. The position of the PHANTOM stylus tip was the master for the simulation dynamics.

In order to evaluate our approach by verifying that skill transfer of a simulator could be improved by augmentation without increasing the cost of the rendering hardware, we created three different versions of the simulator: one which artificially degrades the stiffness and force output of the PHANTOM (which emulates the common condition of hardware that cannot replicate the real world with high fidelity), an event-augmented version of this artificially degraded version to test our augmentation approach, and a control, full-stiffness version that has no augmentation or artificial degradation.

2.3.1 Full Stiffness Simulator

The baseline full stiffness simulator used the PHANTOM device’s maximum rated stiffness (600 N/m) to determine the stiffness, \( k \), of the virtual spring (between the master and the proxy), and the device’s maximum rated force output (8.5 N) to place a cap on the force rendered to the user. The force/motion profile for an execution of the task on the full stiffness simulator is shown in Figure 2.4a. With this simulator, the force output is sufficient to allow the user’s force to build up to \( f_{\text{resist}} \) before the movement of the virtual floor drops the
resistance. Repetitions of this stick-slip type of behaviour yield high frequency variation in the applied force as carving occurs; this variation mimics the characteristics of the real material as the internal structure of the polystyrene breaks in discrete steps. This simulator also produces a noticeable discontinuity in the velocity of the master at the point of emergence, resulting from the drop in resistance in transitioning from the Penetration case to the Contact case.

2.3.2 Degraded Stiffness Simulator

The degraded simulator artificially imposed lower stiffness (300 \( \frac{N}{m} \)) and force output ceilings (0.425 N) on the rendered force. The force was degraded only along the direction of penetration (the device’s full capabilities were used to constrain the user’s motion to the penetration channel). The force/motion profile for an execution of the task on the degraded stiffness simulator is shown in Figure 2.4b. The cap on this simulator’s force output results in saturation that eliminates the high-frequency force discontinuities during carving, and the degraded stiffness severely reduces the velocity discontinuity of the master as it emerges from the virtual slab.

2.3.3 Augmented Low Stiffness Simulator

The augmented simulator used the same artificially lowered force parameters as the degraded simulator, but overlayed an open-loop event-based force pulse to exaggerate the emergence of the probe tip from the material. Event-based haptic rendering has primarily focused on creating high-frequency accelerations on impact with a stiff virtual surface; in this context, researchers have used hand-tuned decaying sinusoids and fixed-magnitude or fixed-duration pulses [Constantinescu et al., 2004; Hwang et al., 2004; Salcudean and Vlaar, 2004].
Figure 2.4: The force/motion profiles for the different simulators.

(a) The full stiffness simulator reproduces both the high-frequency force discontinuities encountered during carving, and the sudden negative acceleration of the master upon emergence from the material.

(b) The degraded stiffness simulator saturates below the force levels at which high-frequency discontinuities occur and fails to generate significant master acceleration at the point of emergence.

(c) The open-loop force pulse applied in the augmented low stiffness simulator restores some of the master acceleration at the time of emergence from the material.
[1997], as well as analytical acceleration-matching transients based on measurements of real collisions [Kuchenbecker et al., 2005]. The event that we are attempting to augment, however, is more like a stick-slip transition than stiff contact. This event is characterized less by the high-frequency ringing transients that result from rigid collision, and more by a sudden drop in resistive force and a corresponding increase in acceleration.

Since we have degraded the stiffness and maximum force of the passive component of the simulation, the drop in force upon emergence from the virtual material is less severe (and less perceptually noticeable). However, we can exaggerate the force change by applying a negative pulse (which pulls the proxy further into the air gap); such a pulse has the effect of increasing the master’s acceleration, requiring user compensation similar to that required by a higher-stiffness transition.

Although we want our pulse to have a sudden onset (corresponding to the sudden emergence of the pin tip from the material), the offset should be smooth (as the user adjusts the force applied to the master to lower its velocity); therefore we use a decaying pulse rather than a fixed-magnitude pulse. Similarly, we chose not to use a decaying sinusoid because the interaction does not involve high-frequency vibration upon emergence.

The pulse is initiated as soon as the proxy point moves below the deepest level of the material, and decays exponentially in time (see Figure 2.5):

$$f_{pulse} = (-0.425 \text{ N})(0.99)^{1000t}$$

(2.14)

The rendered force is capped at $\pm 0.425$ N after summing the event-based pulse with the closed-loop spring force. The force/motion profile for an execution of the task on the augmented simulator is shown in Figure 2.4c. Although this
simulator’s force ceiling still eliminates the stick-slip behaviour during carving, the overlayed negative force pulse restores some of the velocity discontinuity at the transition into the air gap.

Figure 2.5: Force profile for the augmentation pulse. The pulse is negative (pulling the probe into the air gap) and decays exponentially to zero.

2.4 Experiment

Eighteen subjects (recruited from faculty, staff, students and visitors in the Rutgers Computer Science and Psychology departments) were included in the experiment. All subjects gave written consent and were compensated for their time (with money or course credit). Two of the subjects were left-handed (and performed the task and training with their left hands). The subjects were informed as to the purpose of the study (to gauge the effectiveness of different simulators on task performance), but were naïve as to the details of the simulation used. Each subject was randomly assigned to one of three groups corresponding to the three different simulators.
Prior to beginning the experiment, the subjects were told what the evaluation task was (including the material dimensions and the criteria for successful completion), but they were not able to see the arrangement of the materials (which were concealed behind the top layer of plywood).

### 2.4.1 Baseline Block

Before training, the subject performed multiple repetitions of the task (in most cases, 10). The pin was pre-positioned in the guide hole with the tip resting on the surface of the polystyrene slab. The subject was instructed to grasp the handle, push the pin through the polystyrene slab, and then release the handle without withdrawing the pin (see Figure 2.6a). After each repetition, the investigator informed the subject whether the pin was successfully inserted (fully penetrating the first slab without touching the second slab).

Figure 2.6: Experimental apparatus. (a) The user cannot see the material inside the box, and must rely on haptic cues to complete the task. (b) The task is simulated using a PHANTOM haptic device.
2.4.2 Training Block

After performing the real task, the subjects were instructed on the use of the PHANTOM device and operation of the simulator (see Figure 2.6b). Each subject was allowed to train on the simulator for a total of 10 minutes (in two 5 minute sessions interrupted by a 1 minute break). During training, the subject could re-initialize the simulator as many times as desired and experiment with the simulation’s dynamics in any fashion.

To provide high-level feedback to the user about successful completion of the task, the system emitted audible cues to signal whether the task was completed successfully or if the task was failed due to penetrating too far (past the air gap) or due to time-limit expiry.

The subjects were supervised during training, and the experimenter controlled the emergency motor shut-off switch for the PHANTOM.

2.4.3 Evaluation Block

After completing the training phase, each subject was re-evaluated on the real-world task. The task conditions and instructions were the same as in the baseline phase of the experiment.

In both the baseline block and the evaluation block, the subjects had the same timing mechanism available to them as was used in the training block (pressing the space-bar on a keyboard started the timer, and an audible cue was played at the expiry of the time limit). Some subjects chose not to activate the timing mechanism, but its use did not affect the final evaluation of the task completion time (which was measured from video recordings of the trials).
Figure 2.7: Baseline success, evaluation success, and change in success rate are plotted for each subject. On the left is the group of subjects that trained on the full stiffness simulator. In the centre is the group of subjects that trained on the degraded stiffness simulator. On the right is the group of subjects that trained on the augmented version of the degraded stiffness simulator.

2.5 Results

For each subject, we measured separately the rate of successful task execution before and after simulator training:

\[
\text{success rate} = \frac{\text{successful executions}}{\text{total executions}} \quad (2.15)
\]

\[
0 \leq \text{success rate} \leq 1 \quad (2.16)
\]

We compared the success rate before and after simulator training to determine the subject’s absolute improvement.

\[
\text{improvement} = \text{success rate}_{after} - \text{success rate}_{before} \quad (2.17)
\]

\[
-1 \leq \text{improvement} \leq 1 \quad (2.18)
\]

The success rates and improvement of each subject, grouped by training method, are shown in Figure 2.7. The group that trained on the full stiffness simulator had an average improvement of 0.35 ($\sigma = 0.23$), the group that trained on the
artificially degraded simulator had an average improvement of -0.01 ($\sigma = 0.10$), and the group that trained on our event-augmented version of the degraded simulator had an average improvement of 0.26 ($\sigma = 0.20$).

To interpret these results, we performed a two-sample Kolmogorov-Smirnov test on each pair of groups to test the null hypothesis in each case that the samples were drawn from the same underlying continuous distribution (i.e., that the type of simulator used did not differentiate the subjects with respect to task performance). The asymptotic $p$-values were 0.012 for the full stiffness group vs. the degraded stiffness group, 0.077 for the augmented group vs. the degraded stiffness group, and 0.81 for the full stiffness group vs. the augmented group$^1$. These results lead us to reject the null hypothesis in the full stiffness vs. degraded and augmented vs. degraded cases (i.e., we conclude that either full stiffness training or augmented training significantly differentiates task performance from that of the degraded training group).

### 2.5.1 Discussion

One issue when analyzing improvement after training in an experimental setup like ours is assessing possible non-training sources of improvement. In our experimental design, a specific concern is that there may be a training effect from the real executions of the task (in the baseline and evaluation blocks). Our experimental method controlled for this effect in two ways. Firstly, we attempted to minimize the relative training effect of real executions vs. simulated executions by limiting the number of real executions of the task. In their ten minutes of simulator training, subjects performed on the order of hundreds of executions of the simulated task, whereas they performed less than 25 total executions of the real task in the baseline and evaluation blocks. Secondly,

---

$^1$For a discussion of the accuracy of the K-S test’s asymptotic $p$-values for small sample sizes, see [Klotz, 1967]
since we compared only the relative improvement between subject groups, the
degraded-simulator group functioned as a control group, since the only differ-
ence between groups was the simulator on which they trained.

The results presented in Figure 2.7 support our hypothesis that the event-
augmented version of the degraded simulator is more effective at instilling
reproducible skill than the degraded simulator’s passive force-field alone.

As mentioned in Section 2.1, skill transfer can be achieved both by high-
fidelity training, and by training with controlled deviation from reality. In
our user study, we saw evidence of the fact that high-fidelity training can con-
tribute to skill transfer, in that the group that trained on the high-fidelity sim-
ulator showed significantly more improvement than the low-fidelity unaug-
mented training group. However, we showed that the augmented simulator
(which deviates from reality by applying the open-loop force pulse to empha-
size/exaggerate the pin emergence event) also generated significantly more
skill transfer than the unaugmented simulator. This indicates that a simulator
does not need to provide high-fidelity feedback throughout the entire interac-
tion in order to generate skill transfer; focussing on the simulation and render-
ing of key aspects of the interaction can also generate significant skill transfer.

2.6 Conclusions

The results of our user study support the claim that augmentation of events
that signal perceptual transitions in a task can improve the training effective-
ness of a simulator without requiring an improvement in the rendering capa-
bilities (or increase in cost) of the simulator hardware. This finding has im-
lications for the design of a broad class of simulation; for training simulators
whose purpose is to develop proficiency at tasks that require sensorimotor con-
trol, we have shown that augmenting specific aspects of a physical simulation
of the task can achieve improved skill transfer. Therefore, a valuable step in the design of this type of simulation is an analysis of how the task is perceived and performed by the sensorimotor system; in particular, determining which aspects of the interaction are perceptually pertinent for learning the task. Such an analysis will allow the simulation designer to build an implementation of the dynamics of the task that does not necessarily conform to the measured dynamics throughout the interaction, but rather, provides a user experience that faithfully reproduces (or even exaggerates) the aspects of the task most important for developing the necessary sensorimotor control.

It is also worth noting that our augmented training simulation was able to improve on the training effectiveness of the degraded simulation even though the augmentation applied only remedied one of the two fidelity disparities we identified (the saturation that eliminated the high frequency force discontinuities during carving, and the velocity discontinuity on emergence). This observation reinforces our assertion that overall simulation fidelity is not necessary to achieve effective training.

There are limitations to our approach as described so far. In the study we performed, we examined an interaction that was (almost) entirely in the haptic modality (there was also visual feedback in that the subjects could see the motion of the real bone-pin and the simulation’s bone-pin holder — they did not have to rely exclusively on proprioception for their motor control). Though not specifically addressed in this study, our augmentation approach is not inherently limited to the haptic modality; the same approach of augmenting perceptually pertinent interaction features can be applied in multimodal contexts. As well as haptic augmentation of the type used in our simulation, simulators that include other sensory modalities (such as graphical display) could be augmented with multimodal stimuli. For example, in a simulator for the bone-pin
insertion task in which the subject cannot see his or her own arm and hand, seeing instead a graphically rendered version of the apparatus, the pin-emergence event could be augmented by both applying a force pulse and artificially accelerating the visual motion of the pin to exaggerate the excessive acceleration of an un-controlled emergence. Also, in multisensory simulations, cross-modal stimuli (such as audio cues to reinforce the perception of haptic events) could be applied to achieve the same effect of emphasizing particular interaction features to improve training.

Another limitation of the user study we described here is the method by which we developed the augmentation for our simulation. Although we have shown that our proposed method of augmenting perceptually pertinent interaction features can improve the training effectiveness of a simulation, we have not provided a general-purpose recipe for how such augmentation can be developed by a simulation designer. We used force-motion measurement of the real task to build the base dynamics of our simulation, but the force pulse that we chose to augment the pin-emergence event was hand-built. Furthermore, we arbitrarily (though not randomly) identified the pin-emergence event as perceptually pertinent and deserving of augmentation. For the purposes of a simulation designer, more guidelines are necessary to indicate how to analyze a task (including the measurement of the physical properties of the task) to determine how it can be augmented to improve its effectiveness. In the next chapter, we will expand our approach to address the issue of constructing a more general method of augmenting simulations to improve their effectiveness.
Chapter 3

A Method for Automatically Designing Interaction Feature Augmentation

3.1 Introduction

Having established (as shown in Chapter 2) that the augmentation of perceptually salient interaction features can improve the training effectiveness of an interactive simulation, we now turn to the problem of how to design such augmented trainers.

As a user performs a task, various different interaction features are encountered. For example, when a mechanic inserts an engine part into a visually obscured location, the interaction will involve the shape and surface properties of the part and of the engine, as well as transient features such as making/breaking contact between the part and the engine, stick/slip as the part slides into place, or jamming if the part is inserted incorrectly. But the features of the interaction that are pertinent to the user depend on what aspect of the task is being performed. For the mechanic, contact between the part and the engine may be irrelevant when maneuvering the part towards the general area of insertion, but is critical to correctly insert the part precisely in place. The high-level task can be decomposed into subtasks that correspond to different contexts for interaction. In our example, these subtasks might be: manipulating the part to acquire a secure grasp and assess its shape; maneuvering into the general area; exploring the area of insertion to find the correct insertion point; positioning the part for insertion; and sliding the part into place. The
subtask being performed determines which interaction features are most perceptually pertinent and need to be effectively rendered by the simulation (e.g., when the user is sliding the part into place, effective rendering of stick/slip and jamming is critical).

When faced with the problem of trying to automatically generate augmentation for a haptic simulation of a task, we can make the problem more tractable through decomposition. If we can identify the subtask being performed by the subject at any given time, then we can selectively augment the interaction features that are deemed pertinent for that subtask. We have thus reduced the problem of holistically assessing an interaction in progress and generating appropriate augmentation to three subproblems: decomposing the overall task into subtasks; determining what augmentation is appropriate for the perceptual context of each subtask; and detecting throughout the interaction what type of subtask is being performed.

A challenge in investigating the broad problem of training simulator design is that experimentation requires a laboratory task that captures aspects of real-world tasks while being repeatable and allowing detailed analysis of the interaction. Towards this end, we designed an artificial haptic search task that mimics the activities found in an engine-part insertion task: the subject has to scan the haptic environment to find textured surface patches; identify the surface patch with the correct texture; and precisely locate its centre. We use this task as the basis for an evaluation of our design approach.

3.2 Background and Approach

A primary focus of our research into the design of training simulators is developing techniques suitable for surgical simulators. In this domain haptic simulation is of critical importance, since successful performance of the surgical
tasks relies heavily on correct management of the interaction forces between the surgical tools and the patient’s tissue. For this reason, we focus our investigation primarily on establishing methods for developing simulators that successfully impart training on haptic tasks, with the understanding that the methods should also be extensible to multimodal training simulators.

One approach to the problem of automatically generating augmentation for haptic simulations is to measure the haptic properties of the real task, find the differences between those properties and the properties rendered by the simulation, and augment the simulation with the “difference” between the two. Acceleration matching for impact augmentation [Kuchenbecker et al., 2005] is an example of this type of approach. However, while this approach might help achieve greater fidelity with a real environment, that criterion is not always the only or even the best one for judging the effectiveness of a haptic simulation; for example when the goal of the simulation is to improve transfer of training, controlled deviation from the real dynamics can improve the simulation’s effectiveness [Wightman and Lintern, 1985]. Instead of a purely fidelity-based evaluation criterion, we need to consider what augmentations will achieve the desired training effect in a haptic simulation. For example, to evaluate the effectiveness of a surgical trainer, the procedure success rate after training is more important than the amount of error in the forces rendered.

In contrast to other passive (or partly passive) sensory modalities (like audition or vision), haptic sensing is an active process; it involves not only the passive assimilation of input received from the somatosensory system, but also the observation of the consequences of actions. In the course of haptic interaction with an environment, a variety of different actions are performed (e.g., tapping, scraping, lifting, pressing), and a variety of haptic features of the environment are experienced (e.g., roughness, shape, impact, stick/slip). This both makes the haptic modality a rich source of sensory input and complicates
the problem of creating haptic rendering augmentations, since so many different dimensions of an environment are perceived through haptic interaction.

In their early work on haptic exploration (focusing primarily on haptic identification), Klatzky and Lederman [1990] identified a set of haptic dimensions that are directly sensed and aid in the haptic identification of objects. (The term *haptic dimension* [Lederman and Klatzky, 1997] refers to a domain of variation that is accessible to the perceptual system (e.g., roughness of a surface being scraped); the value that a particular situation has in that dimension (e.g., very smooth) is a *haptic property.*) The dimensions that they identified are texture (roughness), hardness, temperature, weight, global volume, exact shape, part motion, and specific function. They also identified a set of *exploratory procedures* that are typically used to assess an object’s value along each of these dimensions.

In the broader context of general tasks, we can define the set of *interactive procedures* as a parallel to (and superset of) the set of exploratory procedures described above. Where the goals of all exploratory procedures are to investigate and assess a haptic dimension, not all recognizable actions made while performing a task fall into this category. Some actions are taken to have an effect on the environment in order to accomplish the goal of the task (e.g., making an incision in a surgical simulation). Such actions are not exploratory, but they conform to the schema that an interactive procedure is performed with a specific intent, be it to explore a specific characteristic of the environment (like texture) or to change a specific property of the environment (as by cutting an object into parts).

One well studied category of interactive procedures is the exploratory procedures used in *haptic identification* which allows the interactor to identify the current haptic situation. Awareness of the haptic situation can involve such
factors as identifying an object being explored with the hand [Klatzky and Lederman, 1990], assessing the geometric properties of an object [Robles-De-La-Torre and Hayward, 2001], or determining the material properties of a surface being tapped or scraped with a tool [Kuchenbecker et al., 2005; Pai et al., 2001]. Identifying the haptic situation can involve the assessment of both object features (such as texture, shape, and compliance) and interaction features (such as the onset of impact, the transition from sticking to slipping, or object-part motion). The classifying characteristic of these interactive procedures is that they all involve performing an action intended to reveal a property of the environment.

Another identifiable class of interactive procedures involve haptic localization — i.e., actions whose goal is to spatially locate a haptic feature. This can involve determining which finger is touching a material with a specific haptic property [Purdy et al., 2004], noticing where amongst a line of distractors a haptic target lies [Overvliet et al., 2007], or finding where on an object a specific haptic feature has been placed [Lecreuse and Fragaszy, 1996]. Haptic localization involves both the construction of mental models of spatial location of features and the execution of exploratory motions to obtain the necessary sensory input.

Some interactive procedures involve not only the discovery of properties of the environment, but also the intent to change the environment. This class of interactive procedure is particularly pertinent in the context of simulators designed to train users at a particular task. Such tasks often require the user to explore an environment, and then modify it to conform to a goal configuration (e.g., make an incision in a surgical trainer, or arrange parts in a mechanical assembly simulation). This type of interactive procedure still involves assimilating sensory information, since accomplishing the desired action is often assisted by haptic interaction with the environment; the environment can
influence the trajectory of a planned motion, and the interactor can, in turn, make use of this influence to achieve a desired trajectory. This haptic guidance of the user by the environment is critical in performing many tasks, for example inserting a catheter into a vessel [Gobbetti et al., 2000] or inserting a peg into a hole [Unger et al., 2001].

A given task may involve the exercise of any combination of interactive procedures, either simultaneously or sequentially, in separate subtasks. By focusing on the set of interactive procedures used in performing a task, we can guide the automatic generation of augmentation to improve the performance of the task.

As well as determining which subtasks are performed in executing a task, we need to consider what augmentations are appropriate to improve the skill transfer for the subtasks. We can describe the domain of possible haptic augmentation as the addition of haptic features to the environment or the modification of features already present. For example, a smooth surface could be made rough, a shallow groove could be made deeper, or a light object could be made heavier. Simulated haptic environments allow for a broader range of features than real environments. Force pulses can be applied in response to certain events. Surfaces can be made to vibrate on command. Features like grooves can be made to move or disappear.

If we identify what sensory percepts are being generated in performing a subtask, we can attempt to generate augmentation that will target those percepts (for example exaggerating the roughness of a surface to make it easier to identify). By taking this approach, we decompose the problem of generating the augmentation into two problems: determining which interactive procedures are used in performing the subtask, and choosing haptic augmentations that appeal to those procedures to generate the desired effect (i.e., improved transfer of training). The first of these problems can in many cases be solved
wholly or in part by high level domain knowledge of the task, particularly
with a focus on recognizing known interactive procedures. To simplify the
second problem, we can leverage existing psychophysical findings that illumi-
nate how different haptic dimensions are perceived.

Lederman and Klatzky [1997] performed experiments to determine which
haptic dimensions are most pertinent to the task of surface identification. Based
on response time in a target/distractor search task, they found that dimensions
that are discernible without a spatial reference frame (termed intensive dimen-
sions) are available earlier in the neural pathway. Hence, variation in a sur-
face’s magnitude of roughness might make a better cue than changes in the
orientation of an anisotropic surface (like a grating) for appealing to the hap-
tic identification capability. For haptic localization, there is an inherent need
for a spatial reference frame. Even when the space across which localization
takes place is the set of fixed fingertip sites, rather than a continuous 2- or 3-
D domain, there is a processing cost incurred in determining the location of
a particular haptic property [Purdy et al., 2004]. Thus, for a localization task,
a cue that contributes to the assembly of a spatial reference frame should be
more effective than one that does not. For example, if the task is to locate a
small haptic feature that lies at the centre of a circle, then an inherently spatial
feature (such as a groove around the periphery of the circle) is more helpful
than an intensive feature (such as a circular patch of uniform roughness).

The remainder of this chapter is organized as follows. In Section 3.3 we
describe the haptic search task that we developed to facilitate a concrete inves-
tigation of our approach to simulator design. We give the details of how we
applied our decomposition-based approach to generate an augmented training
simulator for the haptic search task in Section 3.4. In Section 3.5 we describe
the user study we conducted to evaluate the training effectiveness of our aug-
mented simulator. In Section 3.6, we draw conclusions about the effectiveness
of our proposed decomposition-based approach, including specific guidelines for the design of perceptually-augmented haptic training simulators.

3.3 Haptic Search Task

To allow for a concrete investigation of the design of virtual simulators for real tasks, we need to test our approach on a specific task. We created a haptic search task that is structurally similar to the mechanic’s problem of inserting an engine part without visual feedback. The task parallels the standard visual search task commonly used in psychophysical experiments: the subject attempts to locate a target stimulus that is presented in the company of distractor stimuli that have similar (but distinguishable) characteristics. In our search task, the subject must: haptically scan the environment to search for a target (or distractor) texture patch; discriminate between the target and distractors based on texture properties; and finally locate the precise centre of the target patch.

This synthetic search task is useful in the laboratory setting because it captures key aspects of real-world tasks (such as including a sequence of actions that must be performed to allow later phases of the task to be completed) while allowing detailed recording of the subject’s interaction to support rigorous analysis of the task performance along multiple dimensions. This task is also easily repeatable for user studies because it can be implemented entirely in a virtual environment.

3.3.1 Stimulus Design

In order to apply and evaluate our decomposition approach to developing an augmented training simulator for the haptic search task, we created a virtual environment implementation of the search stimulus.
The environment for the haptic search task consists of a 3-D workspace with smooth flat walls around four sides of a 240 mm by 240 mm floor whose height and surface roughness are varied to create the target and distractor stimuli.

The floor is a height field that is uniformly zero everywhere outside a target or distractor (a scene element) — an example stimulus height field is shown in Figure 3.1. Each scene element consists of a groove surrounding a flat circular patch of roughly textured surface. At the centre of the patch, there is a small pit. See Figure 3.2 for a visualization of the cross-sectional geometry of the scene elements. The textured patch has a radius of 20 mm, and the surrounding groove is 10 mm wide (its maximum depth of 2 mm is at a radius of
Figure 3.2: Environment geometry at 5× scale. (a) The cross-section of the groove surrounding each scene element. (b) The cross-section of the pit at the centre of each scene element.
25 mm). The groove’s cross-section is smooth, with its bottom being a segment of a circle (of radius 2 mm), and each lip being segments of circles (of radius 3 mm). The pit at the centre is similarly smooth, with a total radius of 5 mm, a maximum depth of 1 mm at the centre, where the radial cross-section is a segment of a circle (of radius 2 mm); likewise, the cross-section of the lip is a segment of a circle of radius 3 mm. The only difference between targets and distractors is the texture of the patch (manifested as variation in the coefficient of friction, which is functionally equivalent to texture, but more robust [Pai et al., 2001]). We use the standard Coulomb friction model

$$f_f = -\mu ||f_n||u_m$$

(3.1)

where \(f_f\) is the frictional force, \(u_m\) is a unit vector in the direction of motion, and \(\mu\) is the (spatially varying) coefficient of friction. Both types of texture are generated by adding a baseline coefficient of friction (\(\mu_0\)) to the output of a noise-driven autoregressive AR\( (p)\) process that generates the randomness and periodicity typical of real surfaces [Pai et al., 2001]:

$$\mu(x, y) = \mu_0 + \tilde{\mu}(x, y)$$

(3.2)

$$\tilde{\mu}(x, y) = \tilde{\mu}(x\Delta_x) + \tilde{\mu}(y\Delta_y)$$

(3.3)

$$\tilde{\mu}(k) = \sum_{i=1}^{p} a_i \tilde{\mu}(k - i) + \sigma \epsilon(k)$$

(3.4)

where \(\Delta_x\) and \(\Delta_y\) are the spatial discretization resolutions in the \(x\) and \(y\) direction, \(k\) is the sample index along a dimension, \(\sigma\) is the standard deviation of the input noise, and \(\epsilon(k)\) is a zero mean noise input with a standard deviation of one. For both the target and distractor textures, we used an AR\( (2)\) model, and a spatial discretization resolution of 10 samples/mm (along both axes). See Table 3.1 for the parameters for the AR\( (2)\) functions used to generate the target and distractor textures. Examples of the texture patches generated by these parameters are shown in Figure 3.3.
Figure 3.3: Example (a) distractor and (b) target texture patches at $2 \times$ scale. The texture is represented visually by lightness corresponding to the coefficient of friction, $\mu$. The background surrounding both texture patches corresponds to the uniform coefficient of friction, $\mu_{\text{background}} = 0.2$. Note that although horizontal/vertical structure of the texture is readily apparent to the visual system, it is not perceived by haptic exploration.
Table 3.1: Parameters for the AR(2) functions that generate the (a) target textures (see an example texture generated with these parameters in Figure 3.3a) and (b) the distractor textures (see an example texture generated with these parameters in Figure 3.3b).

The surface outside the texture patches has a uniform coefficient of friction $\mu_{\text{background}} = 0.2$ (including the grooves surrounding the scene elements).

The full stimulus for one episode of the search task consists of two distractors and one target (see Figure 3.4 for an example of the stimulus haptic environment). In a visual search task, scene elements are often (though not always) scattered about the visual field, rather than according to some fixed pattern. In those experiments, however, the viewer’s peripheral vision is being used to locate scene elements in order to make saccades to inspect them foveally. In the case of haptic search, there is no source of peripheral information regarding the location of scene elements, so we provide a fixed structure for their placement. The scene elements are equally spaced around a circle of radius $\frac{200}{3}$ mm; the only variation between episodes is the orientation of the triangle described by the three scene elements (and the noise driving the autoregressive texture of the elements).
Figure 3.4: A visual rendering (at 0.5× scale) of a stimulus environment presented to the user. The texture is represented visually by lightness corresponding to the coefficient of friction, $\mu$, and the height map of the surface is overlayed in blue.
3.3.2 Stimulus Interaction

To realize a repeatable haptic search task, we implemented a virtual environment that allows the subject to interact with the stimulus through a PHANTOM Premium 1.0 [SensAble] haptic device with 6 degrees of freedom in position input and 3 degrees of freedom in force output.

Interaction with the environment is simulated by a quasi-static system where the stylus tip of the PHANTOM represents the master position that is spring-coupled to a proxy point that is constrained to lie within the workspace and above the surface of the floor. When the master is inside the walls and above the floor, the proxy moves with the master and no forces are generated:

\[ x_p = x_m \]  \hspace{1cm} (3.5)

\[ f_m = 0 \]  \hspace{1cm} (3.6)

When the master is outside the walls or below the floor, the proxy is placed at the closest permitted point, and a spring force acts on the master to pull it toward the proxy:

\[ f_n = k(x_p - x_m) \]  \hspace{1cm} (3.7)

where \( k \) is the stiffness of the virtual spring, and \( f_n \) is the normal force applied to push the master toward the surface. Note that we apply no damping to the motion of the master (other than the inherent damping introduced by the inertia of the haptic device), as we are attempting to convey the impression of interacting with the environment through a low inertia stylus.

The texture of the scene elements (and the friction of the background) is implemented by a stick-slip Coulomb model [Salisbury et al., 1995]; when the proxy comes in contact with the surface, a draggable stiction point is established (this stiction point serves as an anchor to which the proxy is connected by a virtual spring). At each update step of the haptic rendering cycle until the
proxy breaks contact with the surface the radius of the Coulomb friction cone is determined according to the current normal force and friction coefficient:

\[ r = \mu \| f_n \| \] (3.8)

This radius, combined with the spring stiffness \( k \), gives a spring length \( l \) that is the maximum distance between the proxy and the stiction point before the stiction point is dragged toward the proxy; this maximal spring length determines the maximum tangential force that can be generated:

\[ l = \frac{r}{k} \] (3.9)

\[ x_s' = u_t \min(l, \| x_s - x_p \|) \] (3.10)

\[ f_t = k(x_s' - x_p) \] (3.11)

where \( x_s' \) is the position of the stiction point after it is dragged toward the proxy, \( u_t \) is a unit vector from the proxy toward the stiction point, and \( f_t \) is the tangential force that is applied to the master (in conjunction with the normal force):

\[ f_m = f_n + f_t \] (3.12)

Each episode of the search task begins with the master held in place (by a stiff spring force) in the centre of the workspace, 20 mm above the surface. Once the episode begins the spring force is released, and the subject is free to explore the environment. The goal of the task is for the subject to locate the centre of the target scene element and hold the stylus tip there for 0.5 seconds.

**Visual Stimulus**

In addition to the haptic feedback described above, the subject is presented with a few visual cues: the outline of the boundaries of the workspace, and the position of the proxy. This sparse visual rendering of the environment is displayed on a vertical screen in front of the subject. This allows the subject
to use visual cues to construct a spatial representation of the location of haptic features (which are not displayed) as they are felt.

### 3.4 Augmentation

Having defined the stimulus for our haptic search task, we created a basic training simulator that is simply the same rendering algorithm as the real task but with artificially degraded stiffness (corresponding to the general design condition in which the rendering hardware cannot trivially reproduce a real interaction with high fidelity). We then applied our approach to develop an augmentation scheme for this simulator. In our approach to automatic simulation augmentation, a complete task is decomposed into subtasks for which different augmentation is applied in accordance with the perceptual features involved in executing the subtask.

#### 3.4.1 Task Decomposition

For most real-world tasks, domain knowledge will be critical to successfully decompose the task into appropriate subtasks (e.g., identifying the sequence of low-level actions used to perform a surgical procedure). However, as discussed in Section 3.2, the decomposition can also be assisted by focussing on interactive procedures that are known to be performed in a somewhat atomic manner. For our haptic search task, we were able to use this assistance to create a subtask decomposition with limited domain knowledge.

The first subtask that the subject must execute is to locate a scene element; we call this the `scan` subtask. In this subtask, the subject typically scans the surface with large scale, high-speed motions, until he or she detects the high-temporal-frequency force discontinuity event that signals that the stylus has encountered (the rising slope of) a groove around a scene element.
The second subtask is assessing the shape (and thus the extent of the texture patch) of the scene element. In the shape assessment subtask, the subject traces part or all of the groove around the scene element to generate a spatial representation of where the texture patch (and its centre) lies.

The other subtask the subject performs is the identification subtask; having located a scene element, the subject must explore it (with a scrubbing exploratory procedure) to gauge the roughness of the surface in order to identify the scene element as a distractor or target.

Although there is only a small set of subtasks in this decomposition, a single execution of the overall task can include multiple instances of each subtask in different orders. For example, while the scan subtask is by necessity the first subtask performed, it may be performed again if the user decides to examine a second scene element after performing the shape assessment and identification subtasks on the first scene element. Likewise, after performing the identification subtask and deciding that the scene element is the target, the subject will likely perform the shape assessment subtask again (or for the first time if the subject went immediately from scanning to identification) to strengthen the spatial awareness of the shape of the scene element in order to locate its centre.

3.4.2 Subtask Augmentation

Having identified the different subtasks that make up our haptic search task, we need to assign augmentations for each subtask.

Scan Augmentation

In the scan subtask, the pertinent perceptual features of the interaction are the force discontinuities experienced when the stylus tip passes over areas of
changing height. However, since the scanning exploratory procedure is performed with relatively high frequency motions (reducing the spatial accuracy of the proprioceptive system), the temporal coherence of these events is of greater significance than the precise spatial alignment of the force discontinuity with the surface feature. Since the simulator has low stiffness haptic feedback, these high-temporal-frequency force discontinuities are lost. These perceptual features can be restored through the use of open-loop augmentation generated by automated techniques similar to those used in computer vision.

A common technique for processing images to extract or highlight pertinent features is to convolve the image with a filter (either in the spatial domain or in the frequency domain). An example of this is edge detection by convolution with gradient-approximating kernels. Since we want to identify places in the environment where force discontinuities are experienced during scanning, our problem is similar to that of edge detection. Rather than edge detection in the 2-dimensional \((x,y)\) space though, we are performing edge detection in the 4-dimensional \((x,y,v_x,v_y)\) space of the interaction between the height of the surface at \((x,y)\) and the velocity of the stylus.

We can think of the height map of the surface as an image whose edges we want to find, where for a particular stylus velocity we are only interested in edges of a certain orientation and spatial frequency (i.e., at higher speeds, we want to detect edges with lower spatial frequency). The 2-dimensional anti-symmetric Gabor filter is a widely used convolution kernel for oriented edge detection at configurable spatial resolution:

\[
g(x, y, \lambda, \theta, \sigma, \gamma) = \exp \left( -\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2} \right) \cos \left( 2\pi \frac{x'}{\lambda} + \frac{\pi}{2} \right)
\]

\[
x' = x \cos \theta + y \sin \theta
\]

\[
y' = -x \sin \theta + y \cos \theta
\]

Here \(\lambda\) is the wavelength of the cosine factor, \(\theta\) is the orientation of the filter.
Figure 3.5: One example of the 2-dimensional anti-symmetric Gabor filter used to pre-process the height maps of the stimuli. This filter (shown in false-colour at 2× scale) corresponds to $\sigma = 2^4$ pixels, $\theta = 0.589$ radians, and $\gamma = 0.5$.

(direction perpendicular to the parallel stripes), $\gamma$ is the aspect ratio of the filter, and $\sigma$ is the standard deviation of the Gaussian envelope that (together with $\lambda$) determines the spatial resolution of the filter. See Figure 3.5 for an example of the type of filter used.

By pre-computing the convolution results of the surface’s height map with Gabor filters of various orientations and spatial resolutions, we can create a 4-dimensional lookup table that indicates which surface locations (at a given stylus velocity) should trigger a haptic pulse to signal an edge-crossing.

For the augmented training simulator, we pre-computed the convolution of each stimulus with Gabor filters at 32 different (equally spaced) orientations and 5 different scales ($\sigma = 2^0, 2^1, \ldots, 2^4$, $\lambda = 4\sigma$, where $\sigma$ is in units of pixels, and the height map of the stimulus is represented as a 1024x1024 image). An example of the result of the height-field pre-processing is shown in Figure 3.6. The aspect ratio of the filter ($\gamma$) was uniformly 0.5. During the scan subtask, the stylus tip location and velocity are used as indices into a lookup table formed
Figure 3.6: The result of convolving the example height field from Figure 3.1 with the Gabor filter shown in Figure 3.5. This image is one slice of a $32 \times 5 \times 1024 \times 1024$ lookup table for the scan augmentation.

by all 160 pre-processed images for the current stimulus; if the lookup value exceeds a threshold, an open-loop fixed-width force pulse is initiated (upwards).

Shape Assessment Augmentation

In the shape assessment subtask, the subject follows the groove around a scene element to determine the spatial extent of the element (and the location of its centre). This is an example of an exploratory procedure that uses the environment to constrain and guide the exploratory motion (as the walls of the groove create an anisotropic resistance to motion that channels the stylus tip
longitudinally along the groove). Since this exploratory procedure leverages the curvature of the surface (which produces the constraints on motion), we augment the simulation for this subtask by applying local force-fields based on surface curvature.

The motion constraints imposed by curved surfaces channel motion towards points (or paths) that are local minima of surface curvature (i.e., points of maximum concavity). By constructing force fields that attract the proxy towards these loci of minimal curvature, the guidance used by the shape-exploration procedure can be replicated in the low-stiffness simulator.

Provided the force field is sufficiently smooth, it will not introduce the instability that results when a simulation’s stiffness exceeds the capabilities of the rendering device. For example, a notch-shaped force field (where the magnitude of the force increases linearly with distance from the locus) is likely to create instability near the locus, but a cosine function (having zero slope at the locus) will be more stable.

In order to simplify matters computationally (and to match the local effect of curvature-induced constraints), we want force-fields that have bounded extent. We choose a single-cycle cosine function:

$$f_{shape} = \begin{cases} 
  f_{max} \left(1 - \cos\left(2\pi \frac{d}{d_{max}}\right) \right) \frac{1}{r_{curv}} \hat{n} & \text{if } d \leq d_{max} \\
  0 & \text{if } d > d_{max} 
\end{cases}$$

(3.16)

where $f_{max}$ is a parameter controlling the overall scale of the augmentation force (we used $f_{max} = 0.75$ N), $d$ is the distance to the nearest local minimum of curvature, $d_{max}$ is the distance threshold imposed to make the force-fields local in extent (we used $d_{max} = 5$ mm), $\hat{n}$ is a unit vector towards the attracting point, and $r_{curv}$ is the radius of curvature (along the principal direction in which the attracting point is a local minimum of curvature).
Identification Augmentation

In the identification subtask, the subject uses the lateral motion exploratory procedure to assess the roughness of the surface. Klatzky and Lederman [2002] found that when perceiving roughness through a probe (as when perceiving roughness from direct skin contact), humans are able to achieve some measure of speed constancy in their perception of the vibratory phenomena induced by surface roughness (i.e., roughness is judged not by vibratory frequency alone, but by speed-normalized vibratory frequency).

Since the subject’s perception of the surface roughness is affected by the speed of the subject-controlled motion, it is insufficient to simply augment the identification subtask by applying open-loop vibration at a fixed frequency. Instead, we wish to produce vibratory effects that mimic those of high-stiffness texture interaction, independent of speed. To achieve this, we can work in the speed-independent space of the original texture.

Real surface texture can be well modelled for human perception by the autoregressive friction variation process that we are using directly (see Section 3.3.1). This texture model provides a Coulomb friction coefficient $\mu$ at any point on the surface (down to the resolution of the spatial discretization $\Delta$). In the friction coefficient variation model of texture, the vibration experienced during lateral motion over texture is due to changes in the coefficient of friction; therefore, we augment the identification task by applying vertical forces proportional to the change in the coefficient of friction.

When the proxy is in contact with the surface, we look up the coefficient of friction, but instead of using it to generate tangential forces (which, in the low stiffness simulator fail to convey the surface texture), we compare it to the previous coefficient (i.e., the coefficient at the previous time-step of the rendering cycle), and generate a vertical force proportional to the change in
coefficient (independent of the normal force). Although this generates vertical forces, rather than lateral friction forces, the vibratory signal experienced through the stylus conveys the same frequencies of variation.

3.4.3 Subtask Identification

In our approach to automatic augmentation of interactive simulation, we call for the use of domain knowledge to assist in identifying the subtasks performed in the course of completing a high-level task, and the use of pre-selected augmentations that appeal to the exploratory procedure used in each subtask. In this decomposition scheme, the remaining problem is to identify during the interaction which subtask is being performed, and which augmentation(s) should thus be active.

Here we once again leverage the coupling between subtask and interactive procedure; since different procedures are used to accomplish different subtasks, we can identify the subtask that the subject is attempting to perform by identifying the interactive procedure being used. In our case, we can distinguish between the scanning, tracing, and scrubbing procedures used respectively in the scan, shape assessment, and identification subtasks, based solely on position in the environment and velocity thresholds.

Since scanning is a relatively high-speed motion, the scan augmentation is only activated if the stylus speed is at least 468.75 mm/s (the speed corresponding to the smallest size of Gabor filter used to preprocess the height map).

Since the shape assessment procedure is executed using finer-controlled (slower) actions than the scan procedure, the shape assessment augmentation (the local force fields around local minima of curvature) is activated when the stylus speed is below the scanning augmentation threshold. Of course, since
the force fields are local in extent and are located around the curvature minima, the shape assessment augmentation is only truly active when the subject is exploring pertinent surface geometry.

The identification subtask is characterized by the use of lateral motion to investigate an area of surface texture. The identification augmentation is thus only activated when the stylus tip is in contact with a textured surface, and only when the stylus is “moving laterally.” In the context of a discrete time-step rendering loop this lateral motion criterion is deemed to be satisfied when the stylus’s tangential speed is at least enough to move it from one texel (at the spatial discretization $\Delta$) to the next (i.e., $v_{x,y} \geq \sqrt{2}\Delta$).

### 3.5 User Study

To test the effectiveness of the augmented simulator design generated by our approach, we conducted a user study comparing the augmented simulator against the basic unaugmented simulator. As with the experiment described in [Chapter 2], our experimental design was structured around the assessment of simulator effectiveness by comparing the task performance improvement across simulator designs. In this experiment however, we simplified the experimental process by using a virtual environment as the reference task. Thus the baseline and evaluation blocks of the reference task as well as the training were all performed using the same haptic device. Twelve subjects (recruited from faculty and students in the Rutgers Computer Science and Psychology departments) were included in the experiment. All subjects gave written consent and were compensated for their time. All subjects were right-handed and used the PHANTOM with their right hands.

A subject’s participation consisted of three blocks of trials taking place in two sessions on different days (see Table 3.2). In the first session, each subject
<table>
<thead>
<tr>
<th>Block</th>
<th>Group 1</th>
<th>Group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>Full Stiffness</td>
<td>Full Stiffness</td>
</tr>
<tr>
<td>Training</td>
<td>Low Stiffness</td>
<td>Augmented Low Stiffness</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Full Stiffness</td>
<td>Full Stiffness</td>
</tr>
</tbody>
</table>

Table 3.2: The simulations used by the subjects in each of the blocks of the experiment. The baseline block took place on the first day of the subject’s participation; the training and evaluation blocks both took place on the second day.

was familiarized with the capabilities of the PHANTOM device and the task to be performed; the subject then performed a baseline block of 50 episodes of the search task. See Appendix A for a complete description of the instructions given to explain the search task to the subjects.

In the second session, each subject performed 50 training episodes on one of two simulations of the search task (with subjects randomly assigned to a simulation group), followed by 50 episodes of evaluation on the “real” search task.

The first simulation was simply a degraded version of the rendering of the real search task (the stiffness coefficient $k$ was set to 60.0 $\frac{N}{m}$ in contrast to the rendering of the real task that used the device’s nominal max stiffness of 600 $\frac{N}{m}$). The second simulation was an augmented version of the degraded simulation. The low stiffness was still used for the quasi-static force output, but active augmentation was applied according to the subtask being performed.

### 3.5.1 Results and Analysis

Since all blocks of trials in this experiment (including the baseline and evaluation blocks) were performed on a simulated virtual environment, complete captures of the stylus position and forces were recorded for all trials, allowing multiple dimensions of the subjects’ performances to be analyzed. As well
as comparing overall task improvement after training, we measured and compared indicators of proficiency for each of the identified subtasks. While the success rate on the task doesn’t significantly differ between the augmented and unaugmented training groups, analysis of the subtask proficiency metrics indicates that the effort required to achieve post-training success is significantly larger for the group trained on the unaugmented simulator.

**Task Success**

A primary metric for evaluating the effectiveness of the proposed augmentation technique is a comparison of the improvement in rate of successful task completion after training on the augmented vs. the unaugmented training simulation. For each subject, we measured separately the rate of successful task execution (i.e., finishing the trial by selecting the target scene element rather
than one of the distractors) before and after simulator training:

\[
\text{success rate} = \frac{\text{successful executions}}{\text{total executions}} \tag{3.17}
\]

\[0 \leq \text{success rate} \leq 1 \tag{3.18}\]

We compared the success rate before and after simulator training to determine the subject’s absolute improvement.

\[
\text{improvement} = \text{success rate}_{after} - \text{success rate}_{before} \tag{3.19}
\]

\[-1 \leq \text{improvement} \leq 1 \tag{3.20}\]

Figure 3.7 shows the success rates and improvement of each subject, grouped by whether the subject was trained on the augmented or unaugmented training simulation.

The group that trained on the unaugmented simulation had an average improvement of \(-0.03 (\sigma = 0.26)\), and the group that trained on the augmented simulation improved by an average of \(0.05 (\sigma = 0.12)\). While this indicates that the subjects who trained on the unaugmented simulation improved more than the other subjects, a two-sample Kolmogorov-Smirnov (K-S) test (with the null hypothesis that unaugmented group’s improvement cumulative distribution function is larger than that of the augmented group) yields an asymptotic \(p\)-value of 0.4425, so these results alone are not necessarily statistically significant (as indicated by the relatively large standard deviations of the samples).

**Scan Subtask**

To evaluate the isolated effectiveness of the scan subtask augmentation, we looked at the time between the start of the trial and the subject’s first encounter with one of the scene elements, which we take to be indicative of both the subject’s proficiency at the scan subtask and the subject’s confidence in his or her sensitivity to the haptic stimulus (since faster exploratory movement indicates
Figure 3.8: Change in average time to first encounter with a scene element before and after training. On the left is the group of subjects that trained on the unaugmented simulation. On the right is the group of subjects that trained on the augmented simulation.

that the subject expects to be able to detect higher-frequency changes in the haptic stimulus).

For each subject, we measured the average time between the start of a trial and the first encounter with a scene element for both the baseline block of trials and the evaluation block. We then computed the absolute change in that average from before to after training. The change for each subject (sorted by training type) is shown in Figure 3.8.

The analysis of this data indicates a statistically significant difference between the two groups of subjects. The K-S test (with the null hypothesis that the unaugmented training group’s change is lower than the augmented training group) gives an asymptotic $p$-value of 0.0383.

**Identification Subtask**

Although a subject’s proficiency at the identification subtask is strongly indicated by the overall task success rate (since over-all task failure indicates that
Figure 3.9: Change in average number of scene elements explored after first exploring the target element. On the left is the group of subjects that trained on the unaugmented simulation. On the right is the group of subjects that trained on the augmented simulation.

the subject failed to identify the selected scene element as a distractor), we can also gauge the subject’s self-assessment of his or her skill at the identification subtask by examining the number of scene elements that the subject explores.

Since the subject does not know apriori the location of the target scene element, whether the first element explored is the target or a distractor is up to chance (as is the second element explored if the first is a distractor). However, once the subject has inspected the target element, the number of subsequent element inspections performed is a good indicator of the subject’s confidence in his or her ability to discriminate between target and distractor textures.

We computed the average number of scene element explorations that a subject performed after first visiting the target scene element in the baseline trial block and the evaluation block. This number can exceed the total number of scene elements (three), since we count the number of explorations (including repetitions), not the number of scene elements visited. If the subject completed a trial without exploring the target element (by selecting a distractor element),
the number of explorations was 0. This condition was rare; omission of such trials from the analysis did not significantly affect the results, and we justified their inclusion on the basis that these trials were also indicative of the subject’s confidence in his or her discrimination ability. If the subject concluded the trial at the first exploration of the target element, the number of explorations was 1. The absolute change in average number of scene element visits is plotted in Figure 3.9.

The K-S test (with the null hypothesis that the unaugmented training group’s change is lower than the augmented training group) gives an asymptotic $p$-value of 0.0061. This indicates a statistically significant difference between the changes in exploration behaviour of the subjects that trained on the augmented simulation and the subjects that trained on the unaugmented simulation.
Shape Assessment Subtask

Since the shape assessment subtask is used to provide the subject with a spatial representation of the location of the centre of the scene element, a good indirect indicator of the subject’s skill at performing this subtask is the number of attempts needed to locate the centre after the subject has decided to select a scene element. Once a subject has selected the scene element as being the target, he or she attempts to position the stylus tip in the small pit at the centre of the element and hold it there for at least half a second. Failure to accomplish this positioning usually manifests as the subject holding the stylus tip still in a position that he or she thinks is the centre (even though it is not in the small pit), then (once the trial fails to end) returning to the groove around the scene element to re-perform the shape assessment and make another attempt.

We computed each subject’s average number of failed approaches preceding the final successful approach for the baseline trial block and the evaluation trial block. The absolute change in average number of failed approaches is plotted in Figure 3.10.

For these results, the K-S test (with the null hypothesis that the unaugmented training group’s change is lower than the augmented training group) gives an asymptotic \( p \)-value of 0.1597.

3.5.2 Discussion

As with the user study in Chapter 2, for analysis of the results of this user study we considered the possibility of non-training causes of performance improvement. In the previous study, we attempted to minimize the relative training effects of the “real” executions of the task by limiting the number of baseline and evaluation executions compared to the number of simulated executions.
In this study, however, the subjects performed the same number of training executions as they did of baseline or evaluation executions. That is why, for this study, we had the subjects perform their baseline block of trials on an earlier day than the training and evaluation blocks; by doing so, we hoped to increase the relative skill transfer impact of the training block vs. the baseline block. As before, we also controlled for non-training causes of performance improvement by limiting our analysis to inter-group comparison; since all subjects were exposed to the same non-training stimuli, the degraded training group acted as a control when compared with the augmented training group. To check the effectiveness of our attempts to limit non-training influences on performance, we inspected the per-trial performance of the subjects on each of the metrics described above; we found no clear indication that subject performance was affected by the number of baseline and evaluation trials performed. Complete plots of each subject’s performance are included in Appendix B.

3.6 Conclusions

The results of our haptic search study gives evidence in support of the effectiveness of our decomposition approach. Although our user study did not reveal a statistically significant difference between the task performance improvement of subjects who trained on the augmented simulation, the availability of full recordings of the task performance allowed analysis of other metrics that are indicative of the effectiveness of the different augmentations applied. We have argued that for dynamical simulations intended to train users to perform tasks, the simulator is best assessed by evaluating its effectiveness at developing the requisite skill. That said, the performance of surgical tasks can typically be evaluated along multiple dimensions (such as smoothness of motion, accuracy of tool placement, and time taken to perform a procedure). In
the task studied in Chapter 2, multiple dimensions were explicitly defined for the task; successful completion criteria included both the placement of the bone pin and the time taken to insert the pin. (Even these explicit dimensions are not necessarily a complete characterization of successful performance of the bone-pin insertion procedure; for example, smooth insertion of the pin through the cortical bone surrogate was not explicitly included in the criteria for success.) Incorporating both of the explicit criteria allowed the subjects’ performance to be more easily evaluated on a pass/fail basis, but this simplicity compromises the ability to judge the subjects’ performances along the different dimensions. In the earlier study, no distinction could fairly be drawn between subjects who correctly inserted the pin but were just over the time limit and subjects who failed to insert the pin correctly (since they were instructed that both the correct placement and the time constraint were critical to the success of the procedure).

In the study described in this chapter, only one dimension of task performance (correct discrimination between target and distractor patches) was made explicit to the subjects, but we were also able to analyze subject performance (and improvement) along other dimensions. This analysis indicated that the augmentations that were incorporated on the basis of existing psychophysical findings were effective at improving the ability of subjects to locate the scene elements, at speeding the discrimination decision (in terms of number of redundant visits to scene elements), and at developing more control in the subjects’ approach to the scene element centre once the discrimination choice was made.

The support we found for our decomposition approach to augmented simulator design reinforces the thesis of this dissertation. The decomposition approach uses existing information about the human perceptual system, both to help identify the subtasks involved in a complex task and to guide the selection of augmentations for the identified subtasks. We have shown that the resulting
designs can increase the skill-transfer effectiveness of training simulators.

Inspection of the study results also raises some questions for future work. One interesting finding of our experiments is that simply repeating low fidelity training does not result in the improvement that one might expect. In fact, several subjects showed no improvement or worsened in their second block of executions of the real task; in contrast, a larger fraction improved with the augmented training. It is possible that this phenomenon is due to loss of motivation or attention due to fatigue. Another interesting result is the apparent effect of the degraded training on subject behaviour indicated by the identification subtask results; the subjects who trained on the unaugmented simulator all became more cautious (in terms of the amount of exploration before making a decision). Here the paucity of the simulation seems to have had a strong training effect (though the increased caution did not translate into increased average success rate).

3.6.1 Simulator Design Guidelines

We can extract some specific guidelines for interactive simulation design from our results. The first step of incorporating our proposed method of augmentation into a simulation design is the identification of the subtasks involved. This decomposition needs to be based on domain knowledge of the task in question, but it can be guided by analyzing the task in terms of the sequence of interactive procedures used (such as the exploratory procedures identified by [Klatzky and Lederman][1990]). This approach to the task decomposition simplifies the next steps of designing a method to identify the relevant subtask at run-time and selecting augmentations for each subtask.

The fact that different subtasks tend to correspond to different interactive procedures makes it possible to identify the subtask being performed by analyzing the action being performed. The interactive procedure used may be
temporally complex (e.g., the back-and-forth scraping in roughness examination), but identifying the procedure being applied may be simpler (particularly when the expected set of procedures are known). For instance, a simple velocity threshold can be effective for identifying the scraping procedure to indicate that the user is performing a roughness assessment subtask.

When designing the augmentation tailored for a particular interactive procedure (and, hence the performance of a particular subtask), it is important to note that the augmentation is not necessarily aimed at reproducing the real-world interaction experience, even for the targeted exploratory procedure; rather, the purpose is to reproduce or enhance the aspects of the interaction that are perceptually pertinent for the subtask being performed. For example, in the case of texture comparison for the purpose of surface identification, the application of a varying normal force corresponding to the varying coefficient of friction (rather than the traditionally computed tangential force) reproduces the spatially-linked vibratory phenomenon encountered during the back and forth scraping exploratory procedure used to assess roughness. However, since tangential force is not applied, it does not actually fully mimic the experience of the interaction. For interactive procedures that target the inspection of tangential forces (such as slow dragging), we expect that this augmentation would be ineffective.
Chapter 4
Event-Based Rendering

4.1 Introduction

In Chapter 2 and Chapter 3 we discussed an approach for designing the dynamics of an interactive simulation. The other side of the design of an interactive application is the rendering system — the way that the simulated dynamics are presented to the user. Given our emphasis on prioritizing perceptually pertinent aspects of an interaction, we need a rendering architecture that allows rendering resources to be dedicated to the simulation’s output in accordance with the perceptual priority. Also, since the goal of the interactive simulation is to create a specific user experience (for instance reliable training on a surgical simulator), it is important that the rendering architecture faithfully reproduces the simulation’s experience as a whole without introducing artifacts that detract from the intended experience.

When using an interactive simulator, the user’s productivity and training experience can be destroyed by latency between the user’s input and the system’s response, and by asynchrony between the system’s different output sensory modalities. This is particularly important in multichannel or multisensory rendering environments in which it is necessary to control and synchronize a large number of specialized sensors and display devices, each controlled by different computers over a local area network (LAN). An example of this type of distributed interaction environment is the Rutgers HAVEN (Haptic, Auditory, and Visual Environment), shown in Figure 4.1.
Figure 4.1: The Rutgers HAVEN.

One avenue to decreased latency and asynchrony is the development of increasingly fast sensors, powerful processing hardware, and low-latency output renderers; while this approach brings improvement, it is also burdened by the inevitability of diminishing returns.

A well-known approach to reducing latency and asynchrony is to make predictions about the future interactions between the user and the system. The limitations of the human body make it possible to perform two distinct types of prediction: continuous and discrete. Prediction of continuous future state is possible because inertia and muscle dynamics cause human movement to be relatively smooth; once a body part is in motion, it tends to continue that motion. For example, a virtual golf simulation might predict the trajectory of a
golf swing in order to graphically render the motion of the club.

The second type of prediction that can be used to improve latency and synchronization is the prediction of discrete interaction events. Many virtual environment (VE) and augmented reality (AR) applications involve discrete decisions by the user; after the decision is in the user’s mind, the body takes time to physically manifest the decision. Following the virtual golf example above, the application might not be concerned with the precise trajectory of the golf club throughout the user’s swing; instead, the important aspect of the interaction is the moment when the club makes contact with the ball, triggering the rendering (e.g., graphically, acoustically, and haptically) of the ball’s launch. Correctly predicting this event, rather than the entire club trajectory, is likely to be both more important (given that the motion of the club is a perceptual blur), and simpler (since the speed of a golf swing would require a very high frequency tracking sensor to yield enough readings to accurately predict the entire trajectory).

The ability to generate advanced warning of user actions prompts us to develop an event-based architecture for interactive applications.

When considering discrete events in VE, other aspects of the human user affect the design decisions. Specifically, characteristics of the user’s perception affect the interpretation of output stimuli as cohesive events. Psychophysical studies yield intermodal and intramodal thresholds beyond which stimuli are perceived as separate events (see Section 4.1.1). These thresholds impose design requirements for the synchronization of output rendering across sensory modalities.

Our contribution in this chapter is a flexible, easy-to-implement distributed architecture for rendering multisensory spatio-temporal events with low latency and good synchronization. Our approach decouples the prediction of system state from synchronization of multisensory output, and can be run on
commodity consumer-level computers and networks. We show that structur- 
ing multisensory interactive applications within an event-based architecture 
simplifies the problem of reducing both the relative latency between the ren- 
dering of multisensory aspects of the same event, and the end-to-end latency 
between the occurrence of the event and its perception by the user.

The rest of this chapter is organized as follows: in Section 4.1.1 we discuss 
related work, including some psychophysical results that influence our design 
choices. Section 4.2 contains an analysis of latency and asynchrony in an in- 
teractive application pipeline. We present our general-purpose architecture in 
Section 4.3. In Section 4.4 we give a proof-of-concept evaluation of how our 
architecture can be used to achieve latency bounds imposed by human per- 
ception. Finally, in Section 4.5 we describe a concrete example implementa- 
tion that illustrates how our architecture can be used to develop multi-modal 
spatio-temporal event-based interactive applications.

4.1.1 Related Work

Many papers proposing architectures for distributed interactive applications 
present specific codebases, APIs, or communications protocols [Taylor II et al., 
2001; Blach et al., 1998; Macedonia et al., 1994; Shirmohammadi and Geor-
ganas, 2001; Shen et al., 2004; Pettifer et al., 2000; Frécon and Stenius, 1998; 
Mauve et al., 2004]. Our intent is to provide a design pattern for distributed 
multisensory interactive applications that can be flexibly applied in a man- 
ner independent of the underlying application development environment, and 
that can be easily incorporated in mixed model architectures that could allow 
a broader class of applications [Macedonia and Zyda, 1997].

The purpose of our architecture is to address two key issues: system latency 
(the delay between a user signalling a decision and the system rendering the
appropriate response), and cross-modal synchronization (the temporal separation between different aspects of a single event being rendered). The impact of these two issues on interactive applications is well-recognized [Garrett, 2002; Mania et al., 2004; Macedonia and Zyda, 1997; Mauve et al., 2004; Blach et al., 1998], and a number of methods to address them have been proposed in the context of VE and AR applications.

Most of the work on distributed interactive applications (such as shared virtual worlds [Mauve, 2000; Macedonia et al., 1994; Mauve et al., 2004; Shir-mohammadi and Georganas, 2001; Pettifer et al., 2000] and cooperative workspaces [Buttolo et al., 1997; Shen et al., 2004]) focusses on accurately replicating state between different sites, where a different user (or set of users) experiences each “subjective” [Pettifer and West, 1999] interpretation of the shared state. In multimodal applications, we must be concerned with replicating a single subjective interpretation of the state across different sensory modalities.

The coordination of different sensory modalities at a single site is usually left to be handled by a single powerful machine [Pettifer et al., 2000], likely making use of multiprocessor hardware [Blach et al., 1998]. However, building multimodal systems often requires (or is simplified by) separate machines to control independent hardware for some or all of the sensors or rendering systems, and a desirable architectural feature is the ability to connect different local components of the application over a network.

Distributing replicated state over a network is the goal of many of the approaches mentioned above, but the techniques used to address replication between different users are often unsuitable for replication between different aspects of a single user’s experience. For instance, exploiting the trade-off between local responsiveness and the frequency of replication inconsistencies [Mauve et al., 2004] addresses the need for replication correctness, but comes at the cost of increased end-to-end latency. When dealing with AR, this
cost is particularly problematic, since the real world cannot be time-shifted and always provides a baseline against which other latencies are measured. Relaxing the synchronization by allowing temporary or slight inconsistencies [Frécon and Stenius, 1998] can result in conflicting cues in the user’s perception; the applicability of such constraint relaxation approaches is bounded by human perceptual characteristics.

Previous work in the AR field has recognized the impact of latency on the user’s experience [Mine, 1993], and the various contributions to latency found in typical AR applications have been categorized [Mine, 1993; Wloka, 1995; Jacob et al., 1997]. These works have also established the importance of latency measurement as the precursor to latency elimination. However, the analysis and solutions presented in those papers was set in the context of single machine, single output modality AR applications, whereas we are also concerned with multisensory output applications (particularly the case where different machines are producing different components of the output). The incorporation of multiple output modalities adds new sources of lag to the latency analysis (and new psychophysical constraints to the latency requirements). The desire to host rich multisensory virtual environment applications (like surgical trainers) on locally distributed systems also calls for further techniques for latency reduction.

When rendering multiple simultaneous signals that are intended to be perceived as a single stimulus, we must quantify what temporal range qualifies as “simultaneous”. Psychophysical studies have shown that the modality of the signals affects the temporal threshold within which simultaneity is perceived. Whereas two sounds must occur within 2 ms of each other to be perceived as a single sound [Levitin et al., 1999], the sight/sound asynchrony threshold has been measured at values ranging from as high as 175 ms [Miner and Caudell, 1998] to as low as around 75 ms [Levitin et al., 1999; Dixon and Spitz,
The touch/sound threshold is approximately 24 ms [Adelstein et al., 2003; DiFilippo and Pai, 2000], and the touch/sight threshold is approximately 45 ms [Vogels, 2004] (but has been seen to vary with attention). These perceptual thresholds dictate the synchronization that must be maintained between output renderers to provide a consistent experience by the user.

Even as LAN speeds and clock synchronization algorithms [Ramanathan et al., 1990] advance to approach the behaviour of a multiprocessor machine, the sensors and rendering devices used impose a minimum end-to-end latency from direct signal detection to system response. Advances in sensor and rendering hardware can lower this minimum [Regan et al., 1999], but with current and foreseeable systems, there is an unavoidable latency floor that establishes a need for the prediction of future state.

A major application for the prediction of future state is head tracking for use in immersive virtual reality (VR) [Garrett, 2002] and head-mounted AR [Azuma, 1997]. When determining user head orientation for AR, latency results in registration errors during movement [Azuma, 1997; Azuma and Bishop, 1994]; in VE, the visual artifacts due to latency can cause performance degradation, loss of immersion, and nausea [Stanney et al., 1998; Mania et al., 2004]. Predictive compensation (constantly estimating real current head orientation from old measurements) is a widely explored [Garrett et al., 2002; Jung et al., 2000; Liang et al., 1991; Marins et al., 2001; Azuma and Bishop, 1994] remedy to this problem.

This type of continuous state prediction is the one most commonly seen in the literature. In contrast, our approach is to focus on the prediction of when and where discrete system events (including user input) will occur (this approach was introduced by Pai, 2005). The focus on discrete events rather than on continuous state highlights different issues. For example, continuous prediction is robust with respect to the occasional spurious prediction (which
will only elicit a momentary blip in the user’s perception) whereas with discrete prediction, it is more important that every prediction be (close to) correct. Slight errors in the time/location at which an event is predicted will be likely to go unnoticed by the user (whereas even small persistent errors in continuous head tracking will cause perceptible registration problems).

4.2 Analysis of Latency in Distributed Multisensory Environments

The goal of this event prediction architecture is to eliminate the latency between real-world events and the synthesized response to them, and the asynchrony between different modalities in responding to the same event. To understand our approach to achieving this goal, we must examine the factors that contribute to the existing latency.

Our model of the data-flow within the system is that state data flows from sensors to controllers, where events are detected/predicted and dispatched to renderers. This model can be illustrated by charting the time-line of one piece of data as it flows through a naïve implementation of the system (see Figure 4.2b).

Let $t_{\text{event}}$ be the time when a renderable event occurs, at which point the world is in state $s_{\text{event}}$. Let $t_{\text{in}} = t_{\text{event}} + \delta_{\text{in}}$ be the time at which the sensor begins processing the $s_{\text{event}}$ data ($\delta_{\text{in}}$ includes both the time it takes for real world phenomena to reach the sensor and the processing overhead involved in such activities as sampling and buffer reading). The sensor then processes the $s_{\text{event}}$ data (for example, an optical tracking sensor might perform 3-D reconstruction, temporal filtering, etc.), and produces a sensor reading, $r_{\text{event}}$ at $t_{\text{sensed}} = t_{\text{in}} + \delta_{\text{sense}}$; for the purposes of this discussion, the $t_{\text{sensed}}$ is the value timestamped onto the sensor reading, $r_{\text{event}}$ (and timestamp) is sent (e.g., over
Figure 4.2: In general, an interactive application has a dataflow from the real world, through a sensor, through the application logic, then through a renderer, and finally back through the real world to the user. Each component of the latency in the naïve event-based application contributes directly to the end-to-end latency. In the predictive event-based application, the timeline begins at an earlier state from which the event can be predicted; predicting the correct time to initiate rendering results in an output perception time that coincides with the real world pertinent state.
the network) to the controller, arriving at time $t_{\text{notified}} = t_{\text{sensed}} + \delta_{\text{notify}}$. In the naïve system, the controller checks to see whether $r_{\text{event}}$ implies that a renderable event has occurred; if it has, an event is dispatched (at time $t_{\text{detected}} = t_{\text{notified}} + \delta_{\text{detect}}$) to the renderer. This overhead $\delta_{\text{detect}}$ could include the computation necessary to estimate hand configuration from pressure sensor data in order to perform gesture detection. The event arrives (over the network) at time $t_{\text{alerted}} = t_{\text{detected}} + \delta_{\text{alert}}$, and (in the naïve system) the renderer “immediately” (at time $t_{\text{rendered}} = t_{\text{alerted}} + \delta_{\text{render}}$) initiates rendering of an appropriate stimulus, which is perceived by the user at time $t_{\text{perceived}} = t_{\text{rendered}} + \delta_{\text{out}}$. In addition to real-world transmission from the output device to the user’s sensory organs, $\delta_{\text{out}}$ includes overhead added by the operating system, the device drivers, and the hardware itself. The end-to-end latency for the naïve system is thus $\delta_{\text{total}} = \delta_{\text{sense}} + \delta_{\text{notify}} + \delta_{\text{detect}} + \delta_{\text{alert}} + \delta_{\text{render}} + \delta_{\text{out}}$. This means that the rendered stimulus will be perceived $\delta_{\text{total}}$ too late.

As developers of an application using a given set of sensors and rendering devices, we only have control over certain parts of this time-line. Even if the controller and render applications are implemented with no computational or communication overhead (i.e., $\delta_{\text{detect}} = \delta_{\text{alert}} = \delta_{\text{render}} = 0$), there will still be some latency ($\delta_{\text{in}} + \delta_{\text{sense}} + \delta_{\text{notify}} + \delta_{\text{out}}$) between an event occurring and the user perceiving the system’s response to the event. This shows that rendering of an event must be initiated before the controller is notified of the sensor reading corresponding to the event; i.e., the event must be predicted from previous states reported by the sensor.

We classify the identified sources of latency into two categories: those which can be circumvented, and those which must be compensated for. Recall that we defined $t_{\text{sensed}}$ to be the time with which the sensor reading is stamped. If all the application’s machines are working to a common clock, $\delta_{\text{notify}}, \delta_{\text{detect}}, \delta_{\text{alert}},$ and $\delta_{\text{render}}$ can all be circumvented by prediction; if (from $r_{\text{event}}$) we can predict
the event at least $\delta_{\text{circumvent}} = \delta_{\text{notify}} + \delta_{\text{detect}} + \delta_{\text{alert}} + \delta_{\text{render}}$ in advance, they will not affect the perceived end-to-end latency. However, since $\delta_{\text{in}}, \delta_{\text{sense}},$ and $\delta_{\text{out}}$ represent latencies before time-stamping and after rendering, they will be manifest no matter how far in advance the event is predicted. These latencies must be compensated for by offsetting the rendering time.

### 4.3 Architecture

#### 4.3.1 Programming Model

![Figure 4.3: Generic architecture of an interactive multisensory virtual environment](image)

Our model of a multisensory interactive system incorporates three entities: sensors, controllers, and renderers (see Figure 4.3). The sensors continually send *time-stamped* measurements to the controllers. The controllers decide what spatio-temporal events are renderable and *predict* when and where these events will occur. The event predictions are sent to renderers that *schedule* and *manifest* appropriate output for the event. The important point is that this decouples the latency (which depends on how prediction is accomplished) from synchronization (which depends on the accuracy with which the clocks on different machines are synchronized and on the variance of only $\delta_{\text{out}},$ the last link
4.3.2  Synchronization

Our architecture relies on accurate clock synchronization to compensate and circumvent timing problems. Fortunately, robust distributed clock synchronization is a well studied problem [Ramanathan et al., 1990]. We will assume that the clocks on each node in our network have been synchronized as closely as desired. For most human interactions, clock synchronization to within a millisecond is sufficient and easy to achieve even on consumer operating systems. We used a simple multicast protocol to synchronize the clocks in our network.

4.3.3  Event Scheduling

Suppose now that we can forecast (perfectly), $\theta_{\text{forecast}}$ time units in advance, the time and location at which the event will occur. This means that we will be able to dispatch the event at time $t'_{\text{predicted}} = t_{\text{detected}} - \theta_{\text{forecast}}$ to the renderers, who will be able to initiate rendering at any $t'_{\text{rendered}} \geq t_{\text{rendered}} - \theta_{\text{forecast}}$. To correctly synchronize the output, we want $t'_{\text{rendered}} = t_{\text{rendered}} - \delta_{\text{total}}$; we will thus be satisfied provided $\theta_{\text{forecast}} \geq \delta_{\text{total}}$ (see Figure 4.2c).

The above reasoning provides us with an estimate of how far into the future we must be able to forecast, but it leaves unresolved the question of exactly when to initiate rendering. The analysis in Section 4.2 assumes that the latencies quoted are consistent; compensation using a fixed offset will result in the transmission of any latency variation through the the end-to-end perceived latency. In the naïve system, since the playback scheduling is “as soon as possible”, the variability in every component of the latency contributes directly to the variability of the end-to-end latency. Analyzing the proposed architecture, it is important to reemphasize the two categories of latency; variation in
\( \delta_{\text{circumvent}} \) (provided the variation is bounded) will not be transmitted, but variation in \( \delta_{\text{in}}, \delta_{\text{sense}}, \) and \( \delta_{\text{out}} \) will. Thus it is the temporal consistency of the sensors and renderers (not of data and event transmission) that is most important in maintaining the quality of the output in this architecture. Only the input and output latencies (\( \delta_{\text{in}}, \delta_{\text{sense}}, \) and \( \delta_{\text{out}} \)) need to be compensated for when choosing the time at which to initiate rendering \( (t'_{\text{rendered}}) \). Since the prediction horizon builds in sufficient time to dispatch the event, a scheduled wait is introduced into the final phase of the architecture; this wait time acts as a buffer that absorbs the variation in \( \delta_{\text{circumvent}} \).

While we assumed above that we can forecast arbitrarily far into the future, our analysis establishes a bound for the minimum forecast horizon necessary to compensate for a given set of sensors and renderers and to circumvent the latency introduced by the application’s logic.

### 4.4 Evaluation Experiment

To test the feasibility of our spatio-temporal event architecture, we implemented a simple application that allows for precise measurement of the end-to-end latency of the entire sensing/prediction/rendering system.

The sensor input to our application consists of motion-tracking measurements of a probe’s position. The measurements are obtained from a six-camera motion capture system [Vicon] that tracks the location of optical markers on the probe and reconstructs the probe’s position and orientation at approximately 120 Hz.

The event that this test detects/predicts is the impact of the tip of the probe with the surface onto which it is dropped from a height of about 40 cm. The system responds to the event by playing a collision sound over a loudspeaker and/or by rendering a force pulse on a haptic mouse [Logitech]. The test was
Figure 4.4: Audio recordings of the evaluation application.

(a) The waveforms of the real impact sound, as well as isolated playback of the audio and haptic responses were all recorded with the same microphone.

(b) A typical simultaneous recording of the real stimulus and the system’s response under the naïve solution show significant latency. Note that the microphone placement for the haptic recording is such that the real impact sound is more muffled.

(c) When our event prediction model is applied, the latency between the real stimulus and the system’s response is greatly reduced — the two signals (three in the final plot) coincide.
implemented on a collection of 4 commodity PCs (ranging from a 2 GHz dual processor Xeon to a 3.2 GHz Pentium 4) connected by a gigabit network. The distributed components are Java applications running under Windows XP (except the sound renderer, which is a Java application running under Fedora Core 2 controlling an RME Hammerfall DSP).

For this application, the pertinent psychophysical thresholds are for auditory intramodal (2 ms) and audio-haptic intermodal (24 ms). To time the actual outputs, we exploit the fact that haptic devices cannot avoid making a sound when they apply a force, and the probe also makes a sound when it hits the surface. By recording the output sounds from all devices (loudspeaker, haptic mouse, actual collision) with a single microphone, we can achieve accurate timing (better than 0.1 ms) and perfect stimulus synchronization.

The recordings of the naïve solution (Figure 4.4b) show significant delay (greater than the perceptual thresholds) between the onset of the real event and the system’s response.

Having measured the mean end-to-end latency observed in the naïve implementation, a simple kinetic predictor with a forecast horizon of 90 ms was implemented; tuning of the compensation offset yielded successful latency correction (Figure 4.4c shows that the asynchrony between the real sound and the synthesized response is within the desired thresholds) with an 82 ms offset for the audio renderer and a 50 ms offset for the haptic renderer.

As well as showing the capability of the spatio-temporal event architecture to synchronize synthetic rendering with real phenomena, this evaluation application also reveals some of the boundaries of a system’s effectiveness. Although the prediction architecture circumvents the internal latency, the variation in the compensated latency is transmitted (see Section 4.3.3). This can be seen by examining the final two plots in Figure 4.4: the location of the distinctive haptic onset varies within the ringing of the real impact sound, due largely
to the maximum temporal resolution at which haptic playback can be initiated.

The sensor is the other aspect of this application’s data chain that cannot be fully controlled. In the case of the Vicon sensor, the temporal variation arises from the preprocessing done by the motion tracker coupled with the fact that the data frames are sequentially numbered, rather than actually time-stamped.

### 4.5 Example Application

As well as our feasibility evaluation, we implemented an example multisensory interactive application to illustrate how the spatio-temporal event prediction architecture can be scaled to meaningful application domains. Our application is a PONG-derivative table-top game incorporating multimodal user interaction — see Figure 4.5.

This application was implemented using the sensor and rendering capabilities of the Rutgers HAVEN, depicted in Figure 4.1.

The playing surface and the virtual PONG ball are displayed on the table’s surface using overhead video-projection. The user grasps and manipulates an iFeel haptic mouse, which is tracked by the Vicon motion tracking system. When a collision occurs with the virtual paddle represented by the mouse, the virtual ball is redirected and a haptic pulse and collision sound are rendered. Audio feedback is also used to perceptually reinforce ball bounces off the side walls and the ball dropping off the table when the player fails to intercept it.

To highlight the distributed flexibility of our architecture, the example application is structured with each system component running on a different commodity PC. Figure 4.6 illustrates the different components and the communication between them.
Figure 4.5: Multisensory PONG! This simple game demonstrates the issues arising in many distributed interactive virtual environments. The user input is measured (using an optical motion capture markers), and the game logic has to render the virtual world in time, with graphics from a projection display, sound from loudspeakers, and haptic forces from a force feedback mouse.
The Clock Source machine provides the synchronization clock (see Section 4.3.2) that gives the other system components a common basis for communicating temporal events. Since the Vicon Tracker is running off-the-shelf software, its time is not synchronized to the common clock; instead, the Event Predictor collects a set of data frames from the tracker and performs a linear least squares regression to find an approximate mapping from frame number to acquisition time (using the data arrival time as the acquisition time). This approach assumes that the sensor time-stamps ($t_{sensed}$) can be well approximated by a linear function of arrival time ($t_{notified}$) — i.e., that $\delta_{notify}$ is consistent.

The Event Predictor uses the Vicon tracking information to determine when and where the user intercepts the PONG ball with the virtual paddle. In order to ensure that notification of changes to the ball’s trajectory arrive at the renderers in time to allow coherent rendering, the interceptions must be predicted in advance. The time and location (and bounce angle) of the next interception
is predicted by extrapolating the mouse’s trajectory (assuming constant acceleration) from the most recent tracking information. If an interception is found within the forecast horizon (a tuneable parameter), an interception event is dispatched to the renderers. Since the renderers need to know in a timely manner about other events that affect the ball’s trajectory (like wall bounces and miss-line crossings), if no interception is found within the forecast horizon, but another system event will occur before the end of the forecast period, that event is dispatched to the renderers.

Three renderers are used in this application to create the user environment. The simplest is the Haptic Renderer that drives the iFeel’s vibration motor in response to interception events. The Audio Renderer is similar to the Haptic Renderer in that it receives events relating to changes in the PONG ball’s trajectory and schedules playback to contribute to the user’s perception of impact (with different audio cues for paddle hits, wall bounces, and misses). The Audio Renderer’s playback scheduling is more precisely controlled than in the Haptic Renderer; instead of waiting until the appointed time to begin playback, the Audio Renderer is constantly writing (silence) to the rendering hardware, and the response sounds are inserted into the written stream at the appropriate location to achieve the desired timing. The lower-level control provided by the hardware interface allows the application programmer to reduce the variation in $\delta_{out}$ for this modality (relative to the Haptic Renderer, where $\delta_{out}$ is subject to the timing vagaries of the iFeel API and USB communication).

The final output device is the Graphical Renderer, which uses structured-light reality augmentation to display the virtual PONG ball and the field of play on a blank table top. Since the unperturbed behaviour of the ball is deterministic, the Graphical Renderer needs to be informed only of the spatio-temporal events corresponding to changes in the ball’s trajectory (launches, wall bounces, paddle hits, misses) or to other changes in the game state (e.g.,
score increases when the ball crosses the centre-line).

The table-top display is accomplished in a similar manner to that used in other dynamic projector applications [Raskar et al., 2003, 1998], except that instead of using either a two-pass render-to-texture or computing a projector-display homography, we use modern graphics hardware to perform a prerasterization transformation of the displayed geometry into the appropriate frame. The geometry — as desired in the display frame — is flattened (by zeroing the depth component) onto the rectangle corresponding to the display; the coordinates are then transformed from display coordinates to projector coordinates (i.e., by the model-view matrix that treats the projector as a camera being positioned to view the table). Once the geometry has been moved into the projector’s coordinate frame, the depth values are restored to yield the correct pixels after rasterization.

This example is intended to show how our architecture is used in a meaningful (if simple) real-world application. The use of spatio-temporal event prediction allows for responsiveness in the interaction between real and virtual elements of the environment, and also facilitates the synchronization of the different modalities (visual, auditory, and haptic). By ensuring that the forecast horizon is sufficiently large to allow in-time rendering by the different renderers, the different modalities combine to reinforce the perception of seamlessness between real and virtual.

4.6 Conclusions

We described a simple — yet effective — architecture for distributed rendering of multisensory events. The demonstrated utility of this architecture provides further support for the thesis of this dissertation. The architecture makes use of established psychophysical findings to provide design targets for latency
and asynchrony thresholds; we showed that by using prediction of interaction events, the resulting interactive simulations can achieve these latency and asynchrony bounds. Based on our experiments, we also make the following observations.

Our implementations highlight the importance of separation of the abstraction of the event-driven architecture from the particulars of the sensors and renderers involved in a particular application. The two applications also illustrate the decoupling of event prediction (i.e., synchronization between the real world and the virtual world) from rendering (i.e., synchronization between elements of the virtual world). This decoupling allows for modular replacement of either component. For our examples, we used a Vicon optical motion tracker as the sensor driving our prediction, but a different sensor/predictor could be used without affecting the application logic or rendering infrastructure.

Separation of latency from synchronization is easy in our architecture, and significantly reduces asynchrony. For further improvements, it is sufficient to focus on the renderer latency and clock synchronization.

The effect of latency variation on synchronization makes accurate time-stamping of sensor data a worthwhile area of focus. For example, in using Vicon’s real-time output as our sensor, we observed that Vicon’s “sensor readings” include temporal filtering, and can arrive at a variable rate, but are stamped with a frame-number which is not consistently correlated with the time a given state occurred.

Although not all interactive applications can be structured in a purely event-based architecture like that we have described (e.g., those where tracking is tightly coupled with output on a frame-by-frame basis), hybrid architectures could be implemented that combine an event-based structure with additional data-paths where necessary for continuous phenomena.

The calibration of the forecast horizon and the latency compensation offset
should be performed sequentially in two phases, since the introduction of forecasting changes the latency introduced by the application logic. The presence of a sufficient forecast horizon allows the pertinent compensatable latencies to be isolated.

Simple predictive models are often sufficient to achieve the necessary forecast horizon. In our applications, simple 1-D and 2-D kinetic predictors were sufficient to obtain adequate results. While domain knowledge of the mechanics of an interaction could facilitate more sophisticated prediction models, it is worth noting the effectiveness of easy-to-implement solutions.

In our architecture as currently described, predicted events are “fired and forgotten.” However, the architecture could be extended to allow for interruption/recall communication between the controllers and the renderers. Such an extension would allow the controllers to dispatch event predictions with greater forecast horizons (and lower confidence); if the controller later determines that the prediction was erroneous, a recall command can be sent to the affected renderers. This approach would take advantage of unused capacity on the rendering machines, where output could be prepared for multiple possible events and then discarded or modified according to later commands.
In this dissertation we explored how simulation and rendering for interactive virtual environments can be improved by structuring their design around specific aspects of the human perceptual system.

In Chapter 2 we presented our argument for focusing simulation and rendering efforts on perceptually pertinent interaction features in order to improve the effectiveness of training simulators. The results of our user study lent support to our approach and indicated that training simulations can be improved (without necessitating more costly hardware), by concentrating on the aspects of the simulation most perceptually pertinent to training. It is important to note that this concentration does not have to take the form of striving for higher fidelity with the real task; controlled deviation from the dynamics of the real task (such as exaggeration) can be more effective for training.

Having established the feasibility of an approach that seeks to improve a training simulation by augmenting perceptually pertinent features of the interaction, we attempted in Chapter 3 to make this result more useful for the general simulation designer by addressing the problem of how such augmentation can be generated for a particular task. We did so by proposing a decomposition-based approach in which the simulated task is broken up into
subtasks that correspond to different interaction procedures. Based on this de-
composition, the simulation designer can select augmentations for each sub-
task that reproduce or enhance the aspects of the interaction that are perceptu-
ally pertinent for the particular subtask. Here we prescribe leveraging existing
psychophysical results that indicate how users perceive the world through var-
ious interactive procedures. Such a recommendation also motivates possible
avenues of future research to more exhaustively catalog the set of general in-
teractive procedures used in manual (and tool mediated) manipulation tasks,
and to identify the perceptual characteristics of those procedures.

As well as investigating how knowledge of the human perceptual system
can be used to improve interactive simulation, we also proposed a method for
leveraging the known limits on perception to allow interactive application de-
signers to tailor their rendering to satisfy latency and asynchrony bounds (see
Chapter 4). The rendering architecture we developed uses interaction-event
prediction to allow multiple renderers distributed over a local network to co-
ordinate their output to compensate for system latency and maintain cross-
modal synchronicity. The event-prediction threshold acts as a tuneable pa-
rameter that allows the simulation designer to measure the system’s end-to-
end latency and compensate for it in order to bring the resulting latency and
asynchrony to within tolerance thresholds that have been established by psy-
chophysical experiments.

The central theme of this dissertation is that just as the capabilities of the
human perceptual system have guided the design of rendering hardware (e.g.,
the colour-gamut of a graphical display device or the update frequency of a
haptic rendering device), examination of how humans perceive particular as-
psects of an interaction allows designers to tailor the simulation and rendering
of their interactive applications to achieve the desired user-experience (e.g., effective simulator training) within the constraints of existing simulation methods and rendering hardware.
Appendix A

User Study Protocol

The experimental setup for the user study (see Section 3.5) had the subjects sit facing a 21” CRT monitor at eye level at a distance of \( \sim 70 \) cm. The PHANTOM haptic device was positioned so that the subject could operate it while resting his or her arm on a desk.

A.1 Instruction Session

Before executing the first of the experiment’s blocks of trials (the baseline block), the subject was introduced to the experimental apparatus and the task. This introduction was presented as a series of instructions displayed on the screen, along with accompanying haptic and visual stimuli. The complete series of these screen instructions is shown in Figure A.1.
The purpose of this experiment is to evaluate the effectiveness of different training simulators.

Press the space bar to continue

(a) The background/text colours have been inverted for clarity in this medium.

Today, you will be familiarizing yourself with the PHANTOM device and performing a search task so that we can establish a baseline.

Press the space bar to continue

(b)

Figure A.1: Instructions presented to the subject (continued below).
The PHANTOM is a force-feedback device that uses motors to apply forces to the tip of the stylus as you move it around the workspace.

Press the space bar to continue

Right now the motors are turned off, so you can move the stylus freely. Hold the stylus like a pen, with your fingers just above the switch, and try it out.

Press the space bar to continue

(d) The position of the haptic master within the workspace is indicated by the yellow sphere.
Now, keep holding the stylus, and I'll activate the motors to pull you to the middle of the workspace.

Press the space bar to continue

Right now, the PHANTOM is generating forces that act as though there is a spring pulling the stylus tip to a certain point.

Press the space bar to continue
Let the spring pull you to that point so that I can safely deactivate the force.

Press the space bar to continue

The PHANTOM can also generate forces that act like a solid surface. Try tapping on the virtual surface below the stylus.

Press the space bar to continue
I've also created some virtual walls around the workspace. See if you can feel those.

Press the space bar to continue

I can also modify the surface to create some texture. Move the stylus around on the surface to find the rough patch.

Press the space bar to continue
(k) It's hard to tell the shape of the rough patch, but if I add a groove around the edge, it's easier to feel the shape. Try it.

Press the space bar to continue

(l) Now that you know that the rough patch is circular, see if you can find the small dent at the centre of it. Once you do, hold the tip of the stylus there.
Well done! We moved the patch (and the surrounding groove) somewhere else on the surface. See if you can find its centre again.

Good! Now I'll show you where I'm putting two patches with different roughnesses. Scrub the stylus back and forth on each of them to feel the difference.

Press the space bar to continue.
In the search task you'll be doing, you'll be trying to find patches that feel like the blue patch. Scrub the patches again to make sure you can recognize the blue patch's texture.

Press the space bar to continue.

Now I'll explain the search task. Each episode of the task will start with the stylus held (by the spring force) in the centre of the workspace, just above the surface.

Press the space bar to continue.
When the episode begins, the spring force will be released and three texture patches will be placed on the surface.

Press the space bar to continue

Two of the patches will have the roughness of the red patch you felt earlier, and one (the target) will have the roughness of the blue patch.

Press the space bar to continue
The three patches will always have the same size and separation, and be arranged somewhere on the circumference of this circle.

Press the space bar to continue

The first thing to do is to find a patch by sliding the stylus over the surface until you feel the groove around a patch (you can do this quite quickly).

Press the space bar to continue
Once you find a patch, scrub it a few times to check whether it feels like the blue patch or the red patch.

Press the space bar to continue.

If it feels like the red patch, find another patch and repeat the scrubbing test.

Press the space bar to continue.
If it feels like the blue patch, find the small dent at the middle of the patch (remember, it is easier to find the middle if you slide the stylus around the groove to feel the shape).

Press the space bar to continue.

Once you hold the stylus tip in the dent at the centre of a patch for half a second, the episode is complete. If it was the dent at the centre of a "blue" patch, you earn 2 points, but if it was a "red" patch, you lose 1 point.

Press the space bar to continue.
If the subject selects the wrong (red) patch, the next screen is (z); otherwise the next screen is (aa).

After this screen, the subject is taken back to (y).

Whoops! That was a “red” patch. Try it again!
Got it! That was the correct patch!

Press the space bar to continue

Now we'll begin the search task. Once you are ready, I'll start the first episode. After each episode, there will be a 5 second pause, and then the next episode will begin. There will be a total of 50 episodes.

Press the space bar to continue
Appendix B

Individual Haptic Search Results

B.1 Task Success

Figure B.1 shows the task success/failure for each trial for each subject in the unaugmented training group. Along with the performance data, a linear least-squares fit is plotted for each of the baseline, training, and evaluation blocks. Figure B.2 shows the same quantities for each subject in the augmented training group.

In these individual results, it is not clear whether the linear fit provides a meaningful indicator of change in performance over time. Since each trial is pass/fail, change in performance is a function of pass frequency; in the case of these results, it seems unlikely that a significant frequency trend can be extracted, particularly for the subjects with very low failure rates. To investigate the change in success rate over time, a further study would require an experimental method that includes more evaluation trials (possibly over the course of several sessions).

B.2 Scan Subtask

Figure B.1 shows the first scan time for each trial for each subject in the unaugmented training group. Along with the performance data, a linear least-squares fit is plotted for each of the baseline, training, and evaluation blocks. Figure B.2 shows the same quantities for each subject in the augmented training group.
B.3 Identification Subtask

Figure B.1 shows the number of scene element visits after the first target visit for each trial for each subject in the unaugmented training group. Along with the performance data, a linear least-squares fit is plotted for each of the baseline, training, and evaluation blocks. Figure B.2 shows the same quantities for each subject in the augmented training group. Note that in both cases, zero feature visit trials (corresponding to not visiting the target before selecting a distractor) are relatively scarce.

B.4 Shape Assessment Subtask

Figure B.1 shows the number of failed final approaches for each trial for each subject in the unaugmented training group. Along with the performance data, a linear least-squares fit is plotted for each of the baseline, training, and evaluation blocks. Figure B.2 shows the same quantities for each subject in the augmented training group.
Figure B.1: Per-trial task success of the unaugmented training group subjects. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.1: Per-trial task success of the unaugmented training group subjects. (Continued from above)
Figure B.2: Per-trial task success of the augmented training group subjects. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.2: Per-trial task success of the augmented training group subjects. (Continued from above)
Figure B.3: Per-trial performance of the unaugmented training group subjects on the scan subtask. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.3: Per-trial performance of the unaugmented training group subjects on the scan subtask. (Continued from above)
Figure B.4: Per-trial performance of the augmented training group subjects on the scan subtask. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.4: Per-trial performance of the augmented training group subjects on the scan subtask. (Continued from above)
Figure B.5: Per-trial performance of the unaugmented training group subjects on the identification subtask. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.5: Per-trial performance of the unaugmented training group subjects on the identification subtask. (Continued from above)
Figure B.6: Per-trial performance of the augmented training group subjects on the identification subtask. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.6: Per-trial performance of the augmented training group subjects on the identification subtask. (Continued from above)
Figure B.7: Per-trial performance of the unaugmented training group subjects on the shape assessment subtask. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.7: Per-trial performance of the unaugmented training group subjects on the shape assessment subtask. (Continued from above)
Figure B.8: Per-trial performance of the augmented training group subjects on the shape assessment subtask. Trials 1–50 are the Baseline block, trials 51–100 are the training block, and trials 101–150 are the evaluation block. A linear least-squares fit is plotted individually for each block. (Continued below)
Figure B.8: Per-trial performance of the augmented training group subjects on the shape assessment subtask. (Continued from above)
Bibliography


Vita

Timothy Edmunds

1996–2001 B. Sc. from the University of British Columbia

2007–08 Bevier Fellow, Department of Computer Science, Rutgers University

2002–07 Graduate/Teaching assistant, Department of Computer Science, Rutgers University

2001–02 Project Manager, MDX Medical Inc.

Publications


