uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Sheldon Andrews
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Measurement-Based Modeling of Contact Forces and Textures for Haptic Rendering

TITRE DE LA THÈSE / TITLE OF THESIS

Professor J. Lang
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Professor E. Petriu
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor A. El Saddik

Professor P. Liu

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

MEASUREMENT-BASED MODELING OF CONTACT FORCES AND TEXTURES FOR HAPTIC RENDERING

Sheldon Andrews

A Thesis
submitted to the Faculty of Graduate
and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of
Master of Science in Electrical and Computer Engineering

September 2007

Ottawa-Carleton Institute for
Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-41617-4
Our file Notre référence
ISBN: 978-0-494-41617-4

Abstract

Haptics is a topic which, in recent years, has gained significant attention from several research fields, including computer graphics, robotics, human-computer interfaces, and virtual environments. Through the use of computer controlled forces, a user is given tactile and kinesthetic cues about the underlying application. These cues enhance the overall quality of the application, or possibly add a new dimension, by allowing the user to experience first-hand some previously unavailable phenomena (e.g., hardness, softness, mass, inertia).

In this thesis, we introduce an interactive and mobile system for the acquisition and synthesis of haptic textures, which includes the surface profile and compliance. Texture represents the attributes at the surface of an object and is related to physical characteristics such as roughness and friction. By texturing objects in a 3D virtual application, the realism of the simulation is greatly increased. Our system is cost effective, simple, and adaptable to a number of workspace environments. We demonstrate how surface features may be acquired by scanning an object using a hand-held touch probe. Sensor data is processed and used to generate haptic textures of sufficient quality to allow for haptic rendering. We also introduce a method to assign 2D texture coordinates based on registration of scanning trajectory with 3D geometry.

Acknowledgements

I would like to thank Dr. Jochen Lang for his guidance and support throughout my research endeavours. I also thank Dr. Emil Petriu for his continuing support of my project. Finally, thank you to Jilin Zhou for offering his advice and allowing me to use his haptic texturing application as a testbed for my system.

I gratefully acknowledge the financial support from the Ontario Research Network for Electronic Commerce (ORNEC) and from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Contents

List of Figures

List of Tables

Glossary of Terms

2D	Two Dimensional
3D	Three Dimensional
COM	Communications port
DOF	Degrees Of Freedom
HIP	Haptic Interface Point
HLA	High Level Architecture
ICP	Iterative Closest Point
JND	Just Noticeable Difference
LERP	Linear Interpolation
PC	Personal Computer
RMS	Root Mean Square
SLERP	Spherical Linear Interpolation
SVD	Singular Value Decomposition
VR	Virtual Reality
WHaT	Wireless Haptic Texture Sensor

Chapter 1

Introduction

1.1 Haptics

Haptics refers to technology which stimulates a users sense of touch. Through computer generated forces, users are able to touch and manipulate virtual objects. Such interaction exposes characteristics about the object which cannot be easily purveyed using graphical or acoustic displays. Unlike these other displays, haptic devices offer a bi-directional human-machine interface, providing both input and output. As outlined by Hayward [1], this is a distinguishing feature of haptic interfaces, that they respond to human gestures by generating tactile and kinesthetic cues creating a simultaneous exchange of information between the machine and user.

The areas of application for haptic technology are widespread and include medicine (tele-remote surgery, surgical simulation), entertainment (video games, 3D painting and sculpting, haptic multimedia playback), engineering design (augmented CAD, virtual product assembly), and e-commerce. For example, in combination with a visual display, haptic technology may be used to train students for tasks requiring hand-eye coordination, such as virtual surgery. This type of application allows would-be surgeons to *feel* realistic touch sensation when training to perform laparoscopic surgery [2]. The Laparoscopy VR^{TM} [3] is a hapto-visual display designed specifically for this task. Incorporating haptics as part of surgical simulation is an application of notable importance because it has been shown to increase the overall success rate of

surgeons during training [4]. The touch models used by these applications need to be able to accurately purvey tactile and kinesthetic information to the surgeon about the medical probe as it comes in contact with virtual human organs (e.g., during slicing, cutting, suturing, puncturing, and poking). Otherwise, the surgeon may apply too much force, causing tissue trauma, or too little force, failing to perform a procedure correctly. This requires accurately modeling the physical properties of human organs.

Another specific application for haptics technology is video games [5, 6, 7]. *HaptiCast* [7], a haptic game developed by this author and colleagues, investigated the use of haptic metaphors in 3D first-person shooter style video games. Players are given a series of haptically enabled wands which offer various methods of interacting with the game world. The *lift* wand allows a player to pick up any object in the game world. Forces affecting the object (e.g., gravity, collisions, inertia) are displayed to the player at their haptic device. Users can even sense friction as two object slide against each other, since this is implicitly rendered by the physics engine. Objects may also be used as a shield to defend against others players, or as a weapon to crush an opponent! An alternate mode for this wand projects some of the force feedback onto the player's avatar, giving them the capability to maneuver using force guidance. The *blast* wand shoots a fiery blast from the end of the wand. This wand may be used to damage an opponent or remove obstacles from players' paths, and is accompanied by a trembling in the haptic device followed by a recoil. The *bolt* wand also shoots projectiles from the wand. However, projectiles are fired quickly and are accompanied by short, transient impulses which represent recoil. The *lob* wand is like a slingshot in which the player pulls on a virtual elastic band in order to shoot a projectile from the wand. The more the elastic band is stretched, the larger the force that is used to launch the projectile. A proportional force is rendered to the player at the haptic interface. All wands take advantage of the high degrees-of-freedom (DOF) input capabilities of the haptic device. HaptiCast was demoed at the Future Play 2006 conference in London, Canada. It received the runner up prize for the category of "Future Game Technology and Design". Response to the game was extremely positive, with many conference attendees expressing a desire to see haptic technology integrated into future games. The audience commented on how the combination of physical modeling and force

feedback made for realistic interaction with the game world. Players enjoyed feeling the physical characteristics of in-game objects and we determined that this added a dimension of interaction that that traditional interface devices do not offer.

This is becoming a trend in the video game industry, where highly detailed physical models are being coupled with novel human-computer interfaces, such as haptic devices. These interface devices increase the realism of the game simulation, and thus the overall entertainment value of the game is also increased. Haptic technology will certainly become increasingly significant in the future development of video games and other entertainment applications.

1.2 Physical Modeling

Given the importance of haptic technology to future applications, how are tactile and kinesthetic cues displayed to the user? Indeed, this is partially dependent on the intended application. The main focus of our work will be on realistic and physically-based touch models, particularly in the context of 3D virtual environments. However, our work is not restricted to this domain and many applications are possible for physically-based haptics.

The algorithms used to model and synthesize haptic cues are collectively called *haptic rendering.* Similarly to how graphic rendering is used to display visual aspects of a computer generated object or environment, haptic rendering displays aspects which are touch-based. Many phenomena, physical or otherwise, may be rendered using haptic displays. For example, users of the technology may use force-feedback to explore the shape of an object in a 3D virtual environment; a graphical user interface (GUI) application may use haptic cues to purvey information about interaction with on-screen components, such as when a button is pushed or a radio box is checked; or the sensation felt as a needle punctures skin may be simulated by a medical application. Rendering of these phenomena is often done based on models which represent physical attributes of objects in the application environment, and several notable attributes are *stiffness*, *damping*, and *roughness.*

Stiffness defines the elastic resistance of an object to deformation by some force,

modeled as being proportional to a displacement value (e.g., $\vec{f} = -k_s\vec{x}$). This property is often used to control how hard or soft an object feels, and stiffness coefficients are common among many haptic rendering algorithms. Damping controls the rate of deformation due to some force and is often modeled as being proportional to velocity (e.g., $\vec{f} = -k_d\vec{v}$). This property is related to the viscosity felt due to interaction with an object or environment. Roughness is a measure of the small-scale variations in a surface and is related to friction and texture. There are multiple models for representing roughness, which may be loosely classified as stochastic, image-based, procedural, or geometrically-based. Unlike stiffness and damping, roughness is a property valid only at the surface of an object. Assigning values to these and other characteristics can sometimes be done trivially or through artistic efforts. However, it often requires careful inspection of the real-world object being rendered.

Estimation of physical properties based on real-world interaction behaviour can be difficult and cumbersome. Equipment for performing these tasks can be quite expensive or inaccessible, and in some cases it is unclear what property is being measured. Technical knowledge is also required. Multiple projects [8, 9, 10] have attempted to acquire estimates of physical properties using various scanning techniques and have developed efficient processes for doing so. Most notably, Pai et al. [8] developed a "one stop shop" scanning facility capable of measuring several of the physical properties discussed above. However, this system consists of an expensive scanning setup and careful planning must be done to perform measurement. The facility is not mobile and has a constrained workspace, allowing only small- to medium-sized objects to be scanned.

1.3 Major Contributions

We have developed a scanning system capable of scanning tactile surface features–particularly texture and compliance. One advantage of our system is that we do not restrict the scanning process with the need for a robotic linkage, and therefore our system is mobile and does not face related workspace restrictions. Also, our system is implemented using inexpensive, "off-the-shelf" components, which makes

it an attractive alternative to other tactile scanning systems. Considerable efforts were made such that the scanning process requires little technical skill on part of the operator and our system may be adapted to fit a variety of scanning environments and situations. We also show that our system may be integrated as part of the standard 3D scanning pipeline by registering scanning trajectory with object geometry (e.g., a polygonal mesh). Hapto-visual models of the virtual objects are rendered using a haptic rendering algorithm which is capable of displaying rigid body contraints, texture, and compliance to the user.

In summary, the key contributions arising from our work are:

- a mobile, cost-effective system for scanning surface texture and compliance,
- integration of our system as part of the standard 3D scanning pipeline [11], and
- development of a haptic rendering algorithm which incorporates texture and surface compliance data.

Publications based on the work from this thesis will appear in the proceedings of the *6th International Conference on 3D Digital Imaging and Modeling (3DIM 2007)* [12] and the *6th International Workshop on Haptic Audio Visual Environments (HAVE 2007)*.

1.4 Terminology

Here we will briefly discuss some of the terminology used in this thesis. Specifically, we'll attempt to clarify ambiguity about certain terms which we anticipate could cause confusion.

The term *surface profile* refers to the height-displacement relationship at the surface of an object. The surface profile function represents the height of the small-scale variations in an object's surface at a given location. Typically, this is a 1D or 2D function. *Texture synthesis* refers to constructing the surface profile function from sensor data, which optionally includes estimating the surface compliance. The term *scan path* refers to the path (a series of discrete point in 3D space) of the probe tip as

it is dragged across the surface of an object, which does not include the small-scale motion of the probe due to variations in the objects surface. Note, we use the terms scan path and *scanning trajectory* interchangeably. *Frequency* is a term which has two meanings, depending on the context in which it is used. When used in discussion about the surface profile, it refers to the *spatial frequency* components of surface variations and has units samples per millimetre. Otherwise, it takes on the traditional meaning of frequency, which is time dependent and has units of samples per second.

1.5 Outline and System Overview

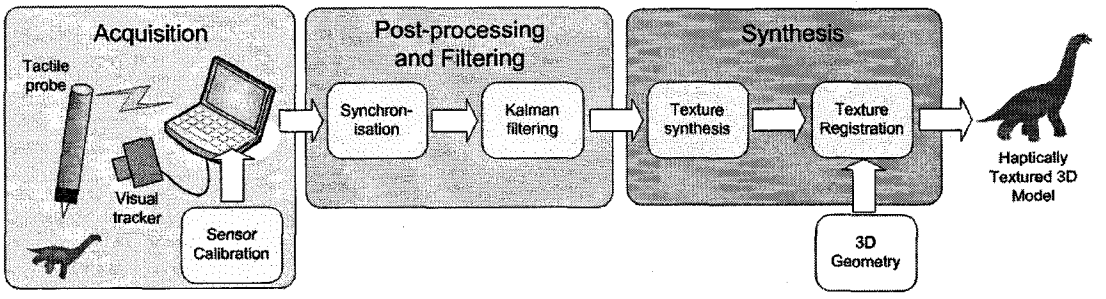


Figure 1: Scanning flow chart.

Figure 1 shows the high level architecture (HLA) for our system. It displays the three stages of our system and illustrates the data flow for generating textured 3D objects from raw sensor data. These stages are categorized as: *acquisition*, *signal processing and filtering*, and *synthesis*. Detailed discussion of the stages and their components occurs in the ensuing chapters.

A background on related work in the areas of haptic texturing is given in Chapter 2. That chapter also includes some of the work to date on scanning techniques for various physical properties, including texture and surface roughness. Chapter 3 gives an overview of the acquisition process for our system, including the hardware and software components. That chapter also discusses the calibration techniques that we used to calibrate the sensors in our system. In Chapter 4 we give details on the steps

taken to condition and prepare sensor data for the synthesis of textures generated by our system.

In Chapter 5, a procedure is presented for generating haptic textures using the conditioned sensor data collected during the acquisition and signal processing stages. Registration of haptic textures with 3D geometry is also discussed, which is an effort to show that our system may be incorporated into the standard 3D scanning pipeline [11] or as part of a 3D modeling tool. Results and concluding remarks are presented in Chapter 6 and Chapter 7, respectively.

Chapter 2

Related Work

In this chapter, we discuss previous work on the acquisition and synthesis of haptic textures. Section 2.1 introduces some of the surface properties which have been modeled using texturing techniques, followed by a discussion of the human perception of surface texture and roughness. Acquisition of physical surface properties are presented in Section 2.2. The discussion focuses mainly on the acquisition of texture, but mentions efforts which acquire other object features (e.g., geometry). Section 2.3 gives an in-depth discussion of force models used for haptically rendering texture, as well as some of their advantages and disadvantages. Finally, we summarize this chapter in Section 2.4.

2.1 Surface Properties and Texture Perception

Describing textures and reproducing them in a virtual context has proven to be one of the most problematic tasks in haptic research. This is due, in part, to that fact that it is difficult to say which properties best characterize the surface of an object. Rough-smooth, hard-soft, slippery-sticky, and rigid-elastic are all examples of descriptors for surface properties. We begin this discussion with some of the work done on tactile perception and psychophysics, as well as classifying the tactile dimensions pertaining to texture.

A study by Hollins et al. [13] attempted to identify the dimensionality of surface

texture by asking subjects to categorize 17 different texture samples based on the perceived similarity when the samples were stroked with a finger. Their study found that we perceive texture in the dimensions of rough-smooth and hard-soft, additionally identifying compressional elasticity as a weak third dimension. Bergmann Tiest and Kappers [14] conducted a similar experiment, with 124 sample materials, for which the physical parameters of compression force related to the material's bulk-modulus and roughness were measured. They confirmed that the perceptual dimensions of rough-smooth and hard-soft correspond to the respective physical parameters of the surface. Therefore, our system aims to scan roughness and compliance in order to acquire pertinent surface texture features.

Perception of roughness has been studied extensively by Klatzky and Lederman [15, 16]. They found that when an intermediary object is interposed between a surface and skin (e.g., a rigid link or glove), vibrotactile coding determines the perceived roughness of the surface. However, this was not the case for direct skin contact, where deformation of the skin surface is highly correlated to the surface gradient of the texture and roughness is perceived according to *spatial intensity* of the texture. Levesque and Hayward [17] confirmed that cutaneous deformation occurs by performing feature extraction on images of fingertips as they were used to explore textured surfaces.

For the case of texture perception via a rigid link, probe size and shape were also found to have an affect on the perceived texture. In their experiments [15, 16], Klatzky and Lederman asked users to explore bumpy surfaces, of varied inter-element spacing, using a pair of rigid probes: one with a small tip diameter and the other with a large tip diameter. When using the small-tipped probe, users experienced a higher degree of roughness than when using the larger-tipped probe. The perceived roughness of the surfaces peaked at an inter-element spacing which was smaller than the peak of the large-tipped probe or the glove. They found the spacing distance at which roughness perception peaked was relatively close to the contact diameter of the tip. For the small probe, these values were 1.7 mm and 2.0 mm, respectively; and 5.0 mm and 4.0mm for the large probe. Also, the roughness perceived when using the large probe and bare finger and glove were very similar, and the perceived roughness

when using the small probe was distant from these others. This is relevant because it suggests that point-based models, which are popular in haptic rendering algorithms, are not ideal for rendering haptic textures since the perceived roughness of a real and virtual texture will be dissimilar. Nevertheless, their work indicates that it is possible to perceive roughness, and surface texture, using stylus-based haptic device.

More recently, a study by Pongrac [18] showed that frequency and amplitude are key factors in perception of texture when vibrotactile coding is used. Her experiments investigated the identification of a series of sinusoidal vibrations and found that they were primarily identified by their frequency and amplitude, not acceleration. The *just noticeable difference* (JND), which is the threshold at which two separate vibrations are distinguishable, was independent of the acceleration. At frequencies $> 300Hz$ the JND was approximately 18% for a given reference frequency. Other researchers have reported JND values of 3% at frequencies less than 250 Hz [19]. These values are consistent with studies which have shown the sensitivity of mechanoreceptors found in the skin to peak at around 300 Hz [20]. Evidently, vibrotactile information in the low frequency range is important to the perception of texture in haptic rendering.

Contrary to Pongrac's study, work by Costa and Cutosky [21] supports the theory that amplitude is most important to roughness perception, and not so much frequency. They measured the roughness perceived by users when the root mean square (RMS) amplitude and *fractal dimension* was varied for stochastic surfaces and rendered using a stylus-based haptic interface. The fractal dimension, D, is a statistical quantification which indicates how much space a curve occupies. In this case, the curve is a fractal which characterizes the surface profile of a rock. The fractal dimension therefore is analogous to the spatial frequency of the surface. Users reported that perception of roughness increased with the amplitude and inter-element spacing of grooves in the surface profiles, much as determined by Lederman and Klatzky. For surfaces with D in the range 1.2 to 1.35, a lower fractal dimension did contribute to perception of roughness. However, for values of $D > 1.35$, the dimension was insignificant to the perceived roughness. Costa and Cutosky's results indicate that frequency is not relevant to roughness perception beyond a point, but is band-limited. This is in agreement with the other work [20, 18, 19].

The geometry and physical parameters of a surface are not the only factors affecting perception when considering computer-based haptic texturing. Campion and Hayward [22] show that perception is also affected by the haptic device being used to render a texture. They compared the acceleration response of two haptic interfaces, the Pantograph [23] and PHANTOM [24], for sinusoidal textures synthesized at various frequencies. The results showed that textures displayed by the devices were highly dependent on the bandwidth and electromechanical transfer function of the device. For example, the Pantograph performed quite reliably at rendering vibrotactile information up to a frequency of approximately 400 Hz; the Phantom performed poorer, with anti-resonance occuring at $30Hz$ and $100Hz$. This suggests that without proper precautions, haptic rendering of a texture may result in something quite different than what was originally intended. Consequently, Campion and Hayward suggest that some perception studies which use virtual haptic textures have been "tainted" due to poor performance of the device, and even present formulae for determine the texture rendering limits of a haptic device. One suggestion they offer is to use a reconstruction filter which alters the texture to better suit the bandwidth of the haptic interface. Our work is primarily concerned with capturing and synthesizing texture detail based on interaction with real-world objects, but Campion and Hayward's work is noteworthy when implementing a rendering application to test our textures.

2.2 Acquisition and Scanning

Scanning physical interaction behaviour has previously required a complex robotic environment [25, 8]. While specialized service providers may opt to offer scanning services based on this kind of facility, the cost and expertise required to operate such a facility is a major obstacle in its acceptance. A robotic measurement facility is also inherently inflexible due to workspace limitations, robotic control and the direct contact between a robot and an object to be scanned. A human-held touch probe can be far more flexible because of the superior control a human can exert on the probe. A human-in-the-loop system is able to adapt to fragile objects, to workspaces of great variety and to objects differently mounted on their support surface (e.g., by holding

the object in hand). Additionally, a human-held scanning device can be made quite inexpensively at a small fraction of the price of a professional 3D scanner. We believe that interactive scanning of physical object properties can eventually make scanning physical interaction behaviour a routine part of the 3D scanning pipeline.

Despite the availability of the 3D scanning pipeline [11], scanning requires great user expertise due to the often fragile acquisition process. However, there are still many shortcomings of this pipeline, specifically that it does not include any method to scan physical interaction properties of objects. These physical interaction properties are important for the animation of 3D objects, such as in physics-based animation involving surface-to-surface interaction. For realistic animation of deformable objects, the behaviour of real-world soft objects has to be measured as a response to applied force fields. The need to scan physical interaction behaviour is also highlighted by the emergence of haptic rendering as a viable display technology for computer animation (e.g., [26]).

Acquisition planning is still a major challenge in automated acquisition. The acquisition may fail to obtain salient object features, or may contain errors due to unfavourable object properties despite the large amount of data produced by a typical scanning process. In industry, 3D scanners based on human operated touch probes (e.g., Immersion Microscribe^{TM} [27]), have had remarkable success and numerous applications. Human control of the scanning could easily prevent some of these problems by directing the data acquisition to relevant detail and dynamically adjusting the parameters of the process. Recently, some "human-in-the-loop" systems have been proposed for the acquisition of geometry [28, 29, 30, 31]. In spirit, our interactive scanning system is similar to these other systems, but we focus on capturing the texture properties and not geometry. We do show how a image-based digitizer [32] may be used to scan the geometry of some objects, this step is to verify that our system is compatible as part of the 3D scanning pipeline and the focus remains on texture acquisition and synthesis.

Scanning of interaction behaviour has been reported by Pai and co-workers in [25, 8]. They estimate friction based on scanning the surface of an object with a force-controlled robotic arm. A frictional coefficient, μ, is estimated by measuring the

tangential forces as a 6 DOF force sensor is dragged over the object's surface, and autoregressive parameter estimation is used to fit a model of surface friction based on the force profile. However, this approach does not capture many of the salient features which help to identify a material (e.g., patterned texture). Later, Kry and Pai [33] recorded the interaction force exerted by a human during stroking of an object with a finger, but did not measure the force variations due to surface texture. Related work to our interactive acquisition approach of surface texture can be found in the 3D imaging literature as well as the robotics or haptics literature. Interactive scanning of the shape of objects has been described by Rusinkiewicz et al. [28] and others [29, 30].

Many approaches exist to measure surface profiles of samples, but most commonly a surface profilometer is employed. Costa and Cutkosky [21] used an optical profilometer to scan rock surfaces, and synthesized similar textures using a fractal algorithm. Wall and Harwin [34] used a linear variable differential transformer to measure displacement of a probe as it is moved (at near constant speed) over a surface. This system generates plausible profiles, however a fixed assembly is required (linear track, motors) and scanning is limited to planar surfaces. One of the disadvantages of using the setup of Wall and Harwin, or a commercial surface profilometer, is the lack of force measurement. Such devices can only measure roughness and not hardness of the surface. The WHaT [35] tactile probe in our work measures force and acceleration and we show how these measurements can be used to estimate roughness and compliance.

Other approaches which do not directly measure surface profile also exist. Okamura et al. [9] augmented a 6 DOF haptic device with an accelerometer in order to acquire vibrotactile data during several styles of interactions– tapping, scraping, and puncture. Again, Okamura's acquisition technique requires a fixed assembly and the modification of a stylus-based haptic device. The instrumentation may be adapted to scan arbitrary objects, but interaction is limited to the workspace of the haptic device.

Surface texture could also be measured with an array of pressure sensors. These kind of sensors have been used for tactile sensing in robotics and haptics [36] and

the resolution of recent experimental devices reportedly approach the resolution of a human fingertip [37].

2.3 Haptic Texture Rendering

Both vibration and force feedback models may be used to render haptic textures. Our discussion focuses primarily on models which are suitable for stylus-based haptic interfaces (e.g., [24]), although display on other devices is possible (e.g., [38]). The textures generated by our system are compatible with several rendering algorithms, as we later show in Chapter 6.

We ask that the reader keep in mind the work of Campion and Hayward [22] and Choi and Tan [39] when considering the texture rendering systems, since these works discuss the reliability and stability of such systems. When generating our textures, we do not take into account the electromechanical limits of the hardware used to render our textures, or their related capabilities. We leave it up to the application developer to ensure that any vibrotactile and force cues rendered by texture interaction are compatible with the limits of the haptic device.

Rendering and modeling of textures is an area which is receiving increasing attention in the field of haptics. Much of the work has been to create perceptually plausible and distinct textures which mimic the roughness perceived when an object it explored either by direct human contact or through some intermediary object (e.g., a rigid probe). These rendering approaches can be coarsely classified as correlated or uncorrelated stochastic approaches, basis function approaches, image-based approaches, and approaches based on digital half-toning [40].

The ideal method for displaying texture is to use a full-resolution model (e.g., with microscopic detail) of an object and render contact forces based on geometric techniques. However, this is not computationally feasible since the complexity of such a model would not permit the fast update rate required for stable haptic rendering, which is widely accepted as being approximately $1000Hz$ [41].

Haptic rendering algorithms for force-based displays are variations on a common theme. Quite often, a penalty-based approach is taken to render both rigid-body

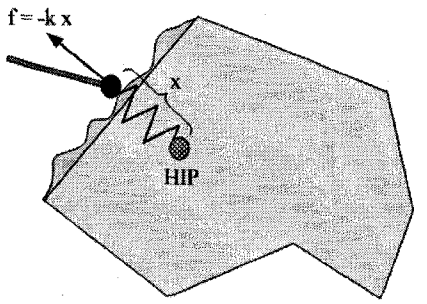


Figure 2: Penalty-based haptic rendering of a textured object.

constraints and texture detail, and is so for many of the models discussed in this section. This approach to haptic rendering uses a spring constant, k, and a displacement value, x, to generate a spring force which is proportional and opposite in direction to the penetration of the *haptic interface point* (HIP) beneath the surface of a rigid object, such that force values, f, are calculated by

$$\vec{f} = -k\vec{x}.$$

For textured surfaces, the penetration depth is calculated by using the boundaries of the object as well as the height of the texture at some point along its surface. An example of rendering a textured object using spring forces is shown in Figure 2. The value of the spring constant k controls how hard or soft an object "feels" when it is displayed to the user. Its value also affects the stability of the haptic simulation. Recall the work by Choi and Tan [39], in which they suggest that a stable value for k when displaying haptic textures is typically quite small (e.g., $< 0.45N/mm$) [1]. Otherwise, the tactile cues displayed to the user may be due to instability in the simulation and not interaction with the texture. However, this restriction on the value of k for texture rendering is dependent on the algorithm employed by Choi and Tan and is not necessarily applicable to the other algorithms discussed in this section.

A pioneering attempt to haptically render texture was that by Minsky [42]. She simulated textures on a 2 DOF haptic interface using lateral forces, which were based on the gradient of image depth maps generated using Perlin noise [43] and fractal

[1] Unfortunately, textures rendered using such a small constant tend to feel very "spongy".

techniques. This "sandpaper system", produced bumpy textures which were perceptually similar to sandpaper. The textures consisted of a series of peaks and valleys. During exploration of a surface, user motion was opposed by rendered spring forces which were directly proportional to the gradient of the the texture in the local area, such that

$$f_{lateral} = -k\frac{\Delta h}{\Delta x},$$

where k is a spring constant which determines the stiffness of the texture, Δh is the height of a bump, and Δx is the lateral translational direction of exploration.

Fritz and Barner [44] used a stochastic approach to increase the realism of surfaces. Random and pseudo-random texture vectors were generated across a uniformly spaced 2D lattice, with vectors at points inside the lattice cells being calculated as a weighted average of vectors in neighbouring cells. Textures are modeled by combining multiplicative and additive Gaussian noise with an implicit deterministic function. This function was built using spectral synthesis techniques, wherein texture characteristics could be defined in the frequency domain. A benefit of their method is that it is easily extended to a 3D lattice. Such lattices would be free of the warping which occurs when a 2D texture is mapped onto a 3D object, making them independent of object size and shape, not unlike methods for solid texturing (e.g., [43]).

Siira and Pai [45] also generate stochastically-based textures, with parameters sampled from a Gaussian distribution. Their approach was capable of creating a dynamic range of realistic textures using only a few parameters. They make the assumption that surface roughness is characterised by a combination of vibrotactile texture forces which are generated in a direction normal to contact, and lateral frictional forces which are proportional to the normal texture forces. A unified rendering equation is introduced which is composed of forces due to texture, friction, and rigid body constraints, e.g.,

$$f_{contact} = f_{constraint} + f_{friction} + f_{texture}.$$

Texture forces were generated randomly from a Gaussian distribution with mean

$\mu = 0$ and variance σ modified to produce perceptually different textures, e.g., a large σ produces very rough surfaces. Normal forces were calculated as:

$$f_{texture} \sim e^{-\frac{(x-\mu)}{2\sigma^2}},$$

where x is a random value, and tangential frictional forces as:

$$f_{friction} = \rho f_{texture}$$

where ρ is a scaling factor, approximating Coulomb friction. A rendering application displayed these forces to users on a 2 DOF haptic interface [23]. They found that texture forces could be generated directly in the device joint-space, without mapping to world-space, and still maintain perceptually distinct textures.

Wall and Harwin [34] developed a procedural model which incorporates the spatial frequency and amplitude information of a surface, obtained by sampling the discrete Fourier Transform of an object's surface profile. Real textures, which were scanned using custom apparatus, are reproduced using Fourier series coefficients. This allows textures to be synthesized using just a small number of parameters, depending on the complexity of the surface profile. Also, relevant characteristics of a surface are easily identified by analysis of its *power spectral density.* Coefficients which correspond to low-power frequencies, or those which are out-of-range of the haptic device rendering capabilities, may be dropped from the profile. Okamura et al. [9] also reproduce texture detail using basis functions generated based on real-world interaction. The acceleration profile for a probe is captured as it is dragged over a textured surface, which is then used to determine parameters for a basis function (in this case, an exponentially decaying sinusoid). The frequency of the basis function is related to the spacing of the pattern and velocity of stroking. Okamura et. al calculate the amplitude, A, on force and scanning velocity as:

$$A(v, f) = av + bf$$

where a and b were parameters determined by the least squares method, and v and f are the velocity and force, respectively, measured at the tip of the stylus. Textures

were synthesized by replaying the basis functions as vibrotactile coding to users via a haptic interface.

Basdogan et al. [46] use an image-based approach for haptic texturing, wherein a surface height field is calculated from grayscale image data using static, procedural, and fractal synthesis techniques. However, unlike Minsky's image-based rendering [47], Basdogan's rendering algorithm combines uses the image gradient to perturb the surface normals of the textured object being displayed. This approach is based on a graphical rendering technique called "bump mapping", introduced by Max and Becker [48]. Force values are calculated as:

$$\vec{f} = kx\vec{n} + kh\vec{m}$$

where k is the stiffness constant, x is the rigid body penetration depth, h is the penetration depth into the texture height field, $\vec{n}$ is the original surface normal, and $\vec{m}$ is the perturbed surface normal which is calculated using the gradient of the height field. Note that, again, the contact force has been decomposed into smaller, individual forces. Also, Basdogan's technique allows for interaction with just the texture of the object, requiring no penetration by the haptic interface point into the rigid body portion of the object (e.g., the case where $x = 0$ and $h > 0$ from above). Their work goes on to comment about the *stability* of rendering haptic textures: for textures with a sharp gradient, the k value should be small, and best simulation results are achieved when the height and wavelength of the textures are of the same order.

A technique which departs from point-based haptic texture rendering is the work of Otaduy and Lin [49, 50]. They compute texture forces using probe geometry and surface detail. Their model is inspired by perceptual studies carried out by Lederman and Klatzky. In their experiments, Otaduy and Lin allowed users to explore textured surfaces using a virtual probe which had been textured using depth images. The perceived texture could change drastically depending on the size, shape, and texture of the probe. The normal and tangential texture forces of their rendering equation are defined as:

$$f_{normal} = -k\delta_n$$

$$f_{tangent} = f_{normal}\frac{d\delta_n}{du}$$

where k is the stiffness (spring) constant, and δ_n is the object inter-penetration depth. This model addressed some of the geometric and dynamic issues with the perception of virtual textures. However, it is also computationally challenging, especially for complex geometry. To speed up their simulation, Otaduy and Lin implemented their algorithm for calculating penetration depth on a graphical processor.

Shopf and Olano [51] have created a *haptic shading framework* which builds on the techniques discussed above and allows user-defined procedures to define texture characteristics of an object. Their goal was to provide a shading framework for 3D objects similar to graphical shaders, such as the Renderman Shading Language [52]. They note that the graphical and haptic shaders used to texture an object must be identical (not just statistically similar) if consistency is to be maintained, and they hint that a unified shading language would be one possible method of doing so. Their implementation uses a C++ library with a variety of procedural methods for generating textures. A rendering equation which incorporates several of the techniques discussed above is used to display haptically textured objects.

2.4 Summary

In this chapter, we discussed some of the properties and features which represent surface texture. Roughness and compliance were identified as being especially pertinent in the context of haptic rendering. An analysis of the psychophysics of these properties was presented, and a consensus reached that spatial intensity and frequency both contribute to the tactile perception of roughness.

Some of the scanning techniques for acquiring surface features were listed in Section 2.2. Many approaches have opted for high accuracy techniques, but also require sophisticated robot linkage or expensive scanning equipment. Section 2.3 presented some of the models to date which have been used to haptically render texture and other pertinent surface features. We will show in this thesis that the textures generated by our system are compatible with these models.

Chapter 3

Acquisition System

In this chapter we present the acquisition component of our haptic texturing system. In Section 3.1 and Section 3.2 we discuss the hardware components employed by our system, mainly consisting of a tactile probe and digital camera (visual tracker). A standard PC or laptop is employed as a data host and is used to coordinate acquisition of sensor data. Section 3.3 outlines the procedure for scanning real-world objects and surfaces. We also give a detailed discussion of the techniques for calibrating the touch probe and tracker.

3.1 Tactile Probe

A tactile probe, called the WHaT [35] (see Figure 3), senses small-scale motion and force data as it is moved across the surface of an object. It consists of a force sensor, aligned with the major axis of the device, and a pair of bi-axial accelerometers which provide motion data about the probe tip in 3D space. The force sensor is capable of measuring forces in the range of 0 to $4.9N$, and acceleration in the range of $-2g$ to $+2g$, where g is gravitational acceleration on the surface of the Earth[1].

Force and acceleration sensor data is streamed to a host PC at a rate of $500Hz$. This is done over a wireless interface using a 115,200 bps serial port communication

[1] approximately $9.81m/s^2$

link[2]. Each frame contains a start byte (FFh) followed by 4 data bytes (3 accelerometer values, 1 force value) for a total of 5 bytes per frame. Each sensor uses 8-bits to represent its data, giving a resolution of approximately $0.015g$ for accelerometers and $0.019N$ for the force sensor. The WHaT may be powered using an internal power source, or by a specially constructed external battery pack which is attached via a pair of wire leads.

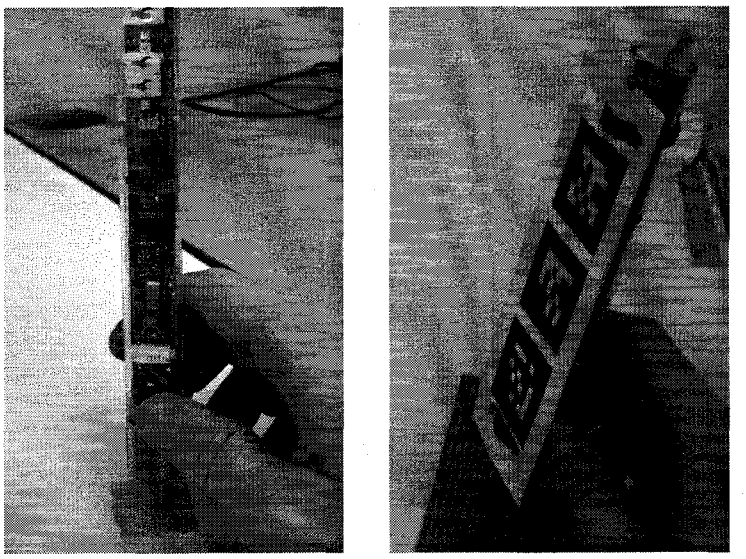


Figure 3: The WHaT.

Figure 4 shows acceleration and force profiles obtained from the WHaT as it is moved over a bumpy surface. Notice that there is a strong correlation among each of the acceleration profiles. Spikes in the profiles indicate locations where a discontinuity in the object's surface was encountered by the probe tip. Given a sampling rate of $400Hz$ and scanning speed of $50mm/s$, the probe is capable of detecting variations in the surface of an object as small at $0.125mm$. The resolution will vary depending on the scanning speed (e.g., $0.1mm$ at $40mm/s$).

[2]Note: there is no error correction or detection present in the communication protocol, but steps described in Section 4.1 remove some spurious data from the sensor profiles.

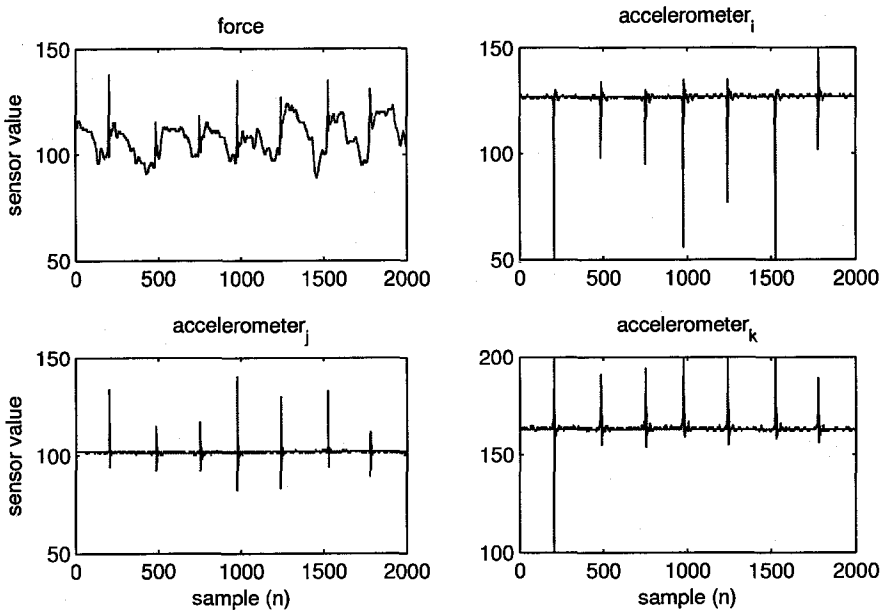
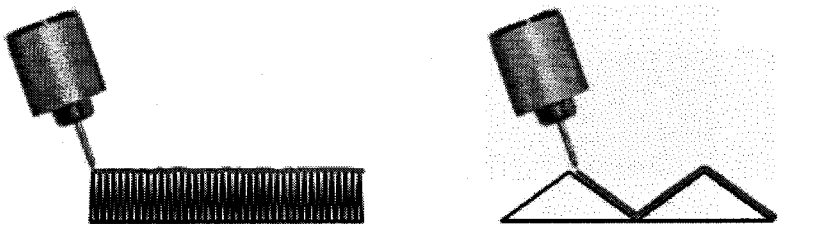


Figure 4: WHaT sensor data.

However, the probe's ability to detect variations in an object's surface cannot increase without limit, even when moving the probe at very low scanning speed. The spatial sampling capabilities of the WHaT, or any tactile probe, are limited by the size and shape of the probe tip. As experiments by Unger [53] and Lederman et al. [16] have shown, the size, shape, and compliance of a tactile probe have an affect on the psychophysical perception of texture (roughness). This carries over to our touch probe in that the probe tip affects the level of detail it is able to measure about a surface. We illustrate this concept in Figure 3.1, where the size and shape of the probe tip affects the spatial frequency contents of the surface which can be measured. Effectively, the probe tip acts like a low-pass filter whose cut-off frequency is determined by its size and shape.

For a texture detail with a spatial resolution finer than the tip diameter, the probe is unable to measure texture detail. For the WHaT, the tip is conically shaped with a diameter of approximately $1.57mm$ at the base and $0.64mm$ near the tip. There, we estimate the actual spatial scanning capability of the WHaT to be about $0.65mm$ and features with a wavelength smaller than this threshold will be difficult to detect.



(a) Probe tip scanning a fine surface.

(b) Probe tip scanning a coarse surface.

Figure 5: Illustrations showing the effect of probe tip size on spatial resolution.

3.2 Visual Tracker

For estimation of large-scale motion, we affix a set of fiducial markers to the probe. ARTag [54], an augmented reality system, is used to locate the marker within images obtained from the digital camera. ARTag boasts low false-positive and false-negative marker recognition rates and is capable of accurately recognizing markers even when they are partially occluded. ARTag is capable of recognizing over 2000 individual markers, however only a few are used to track the tactile probe. Figure 3 shows a picture of the WHaT with markers attached. The size and orientation of the marker in the camera image allows the probe to be tracked accurately at a scanning distance of approximately 250 to $600mm$, but also allows the probe to be comfortably manipulated by the user. Figure 6 shows the marker we used for our experiments, with the printed image dimensions given in the caption. It is orientated such that the length of the marker is coincident with the k sensor axis of the WHaT and the image plane is orthogonal to the j axis.

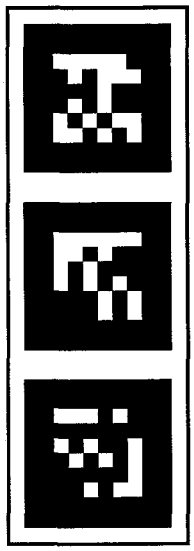


Figure 6: The marker used to track the touch probe. It is affixed to the front of the probe, with its length being coincident with the major axis of the WHaT and with dimensions of $46mm$ (width) by $98mm$ (height).

We employ a Point Grey Dragonfly Express camera for tracking which is capable of capturing 640x480 full-frame colour images at $200Hz$. Images from the camera are sent to the visual tracking software. The camera communicates with the host PC by a IEEE 1394b "firewire" channel. The camera is mounted on a tripod for easy height adjustment (when scanning objects of various sizes). Our system requires that the camera remains stationary during the scanning procedure. From our trials, the visual tracker component is able to estimate the global position and orientation of the touch probe at a rate of approximately $45Hz$ using this setup.

Obviously, a synchronization step is required in order to match data from the tracker with data obtained from tactile probe due to the different sampling rates of the sensors. For this reason, all sensor data is timestamped using a high resolution timer, with $1ms$ accuracy, on the host PC. More on the synchronization of sensor data in Section 4.1.

Other options were considered for tracking the WHaT, but were rejected due mainly to the mobility and cost criteria of our system. Ruffaldi et al. [55] tracked

a touch probe using a six-camera Vicon motion-capture system[3]. They were able to track the probe position to within $0.5mm$ accuracy. However, such a system restricts the mobility of the scanning apparatus, since multiple cameras must be placed strategically in the environment to track the probe. Also, their system is much more costly than our current setup and therefore is not in line with the goal of having a low-cost system.

Magnetic tracking systems, such as the Ascension Flock of Birds real-time motion tracking system[4], offers occlusion-free tracking within a 3D environment. However, these systems require additional cableage to power the markers. Also, technical specifications quote that they are accurate to within $\approx 2mm$, which is poorer than most visual tracking systems. The large and heavy sensors used by this class of tracker also make them near impossible to attach to a tactile probe such as the WHaT, thus affecting their mobility. Magnetic trackers are also known to be affected by metal in the environment, which would affect our ability to scan metal objects. Again, these systems are more costly than the solution we have adopted.

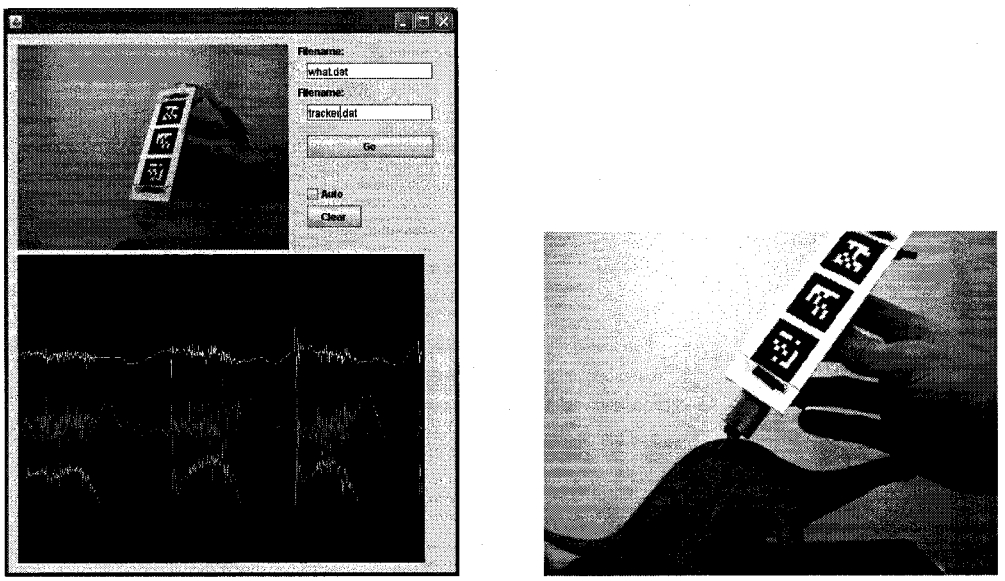
3.3 Scanning

A suite of Java and MATLAB-based software was developed to acquire and process sensor data from the WHaT and ARTag tracker. The components of this suite, analogous to blocks found in the flow chart of Figure 1, are:

- low-level libraries to retrieve sensor data from the WHaT and digital camera;
- an OpenGL-based application to display probe path, orientation, and tactile sensor data in real-time;
- MATLAB programs for synchronizing, filtering, and interpolating sensor data.

Figure 7(a) shows a screen shot of the application used to acquire texture information. A real-time display of the position and orientation of the probe tip, as well

[3]http://www.vicon.com/
[4]http://www.ascension-tech.com/products/flockofbirds.php



(a) Java-based scanning application.

(b) Using the tactile probe to scan a toy dinosaur.

Figure 7: Our scanning system in action.

as the acceleration and force profiles, is shown to the operator. Immediate feedback of sensor data is an advantage (and a necessity) for our interactive scanning system since the operator can judge the quality of the scan based on this information. A check box on the application window, labelled "Auto", allows the operator to select if all sensor data is stored (off), or if sensor data is only stored when the probe is in contact with the object's surface (on). The application automatically opens the COM port on the host PC connected to the WHaT. A worker thread continuously polls the COM port for data packets from the WHaT and stores them in a buffer for later processing. Similarly, another thread polls the digital camera for images and processes them using the ARTag software. Position and orientation information about the touch probe are also stored in a buffer for later processing.

The object being scanned is placed approximately $250mm$ to $600mm$ in front of the camera. Objects of various shape and size may be scanned by our system, however it is a requirement that markers must remain within the boundaries of the camera image in order for the path of the probe tip to be reconstructed. Large or

partially occluded objects may be scanned in segments. ARTag is capable of detecting partially occluded markers, but at the cost of decreased accuracy. For best results, it is recommended to use the probe with as little or no occlusion of the markers.

We assume that the object does not move relative to the camera during a single scan, and vice-versa. The operator places the tip in contact with the object and signals the host PC that it should to begin storing sensor data from the camera and probe (this may also be done automatically using the data acquisition software), scans a desired section of the surface, and signals the host PC to stop storing sensor data. If the scan is unsatisfactory (e.g, noisy sensor data, poor scanning trajectory), the section of the surface may simply be scanned again, which is one of the great advantages of a "human-in-the-loop" system. Figure 7(b) shows an example of scanning a toy dinosaur. The operator is able to identify and regions of an object's surface which contain different or similar texture information and select scan paths accordingly.

3.4 Coordinate Frames

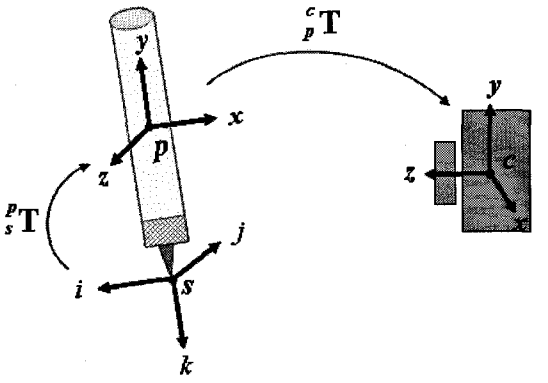


Figure 8: Coordinate frames.

Before continuing, we'll briefly discuss the coordinate frames used by our system since we reference them often in upcoming sections. Figure 8 shows the various coordinate frames used by our system: the camera frame c, the frame defined by the visual tracker markers on the WHaT probe p, and the probe sensor frame s. We

use the camera frame as a global reference frame; position and orientation from the tracker is relative to this frame. The transformation, ${}^{c}T_{p}$, from the marker coordinate frame is performed using position and orientation information obtained from the visual tracker. The force and accelerometer sensors are coincident with frame s and force-accelerometer vectors must undergo an additional transformation, ${}^{p}T_{s}$, to the marker frame before finally being transformed to the global frame.

3.5 Calibration

We determine the intrinsic properties of our camera using Bouguet's camera calibration toolkit [56]. The toolkit determines an estimate of the camera focal length, principal point (image center), skew and distortion coefficients using images of a checkered, planar surface. A set of 25 images is used to calibrate the Dragonfly Express camera for our experiments.

Since ARTag uses a basic pin-hole camera model which doesn't correct for radial or tangential distortion, only the image center and focal length parameters are used in the camera calibration. We obtain a black-box estimate of the tracker observation noise using the following method: the probe marker is held steady by the operator in front of the camera and the variance of position and orientation vectors is measured. This accounts for observation error in tracker position and orientation due to illumination flicker, hand jitter, or other environmental conditions.

Figure 9 shows the distribution of position measurement over several hundred recorded samples. These histograms suggest that a Gaussian distribution is appropriate for modeling tracker noise. We estimate the variance of position and orientation measurement and use the values to create a covariance matrix for Kalman filtering of the probe position and orientation data (see Section 4.2). However, it should be noted that the measurement error of the tracker is dependent on: (a) the distance from the marker to the camera lens, and (b) the orientation of the marker *away* from the image plane. That is, the tracker is able to more accurately estimate the probe's position and orientation if it is held close to the camera and oriented in such a way that the marker plane is parallel to the projection plane (assuming a pin-hole camera

model). The conditions used to generate the plots in Figure 9 used typical scanning distances and orientations for the probe.

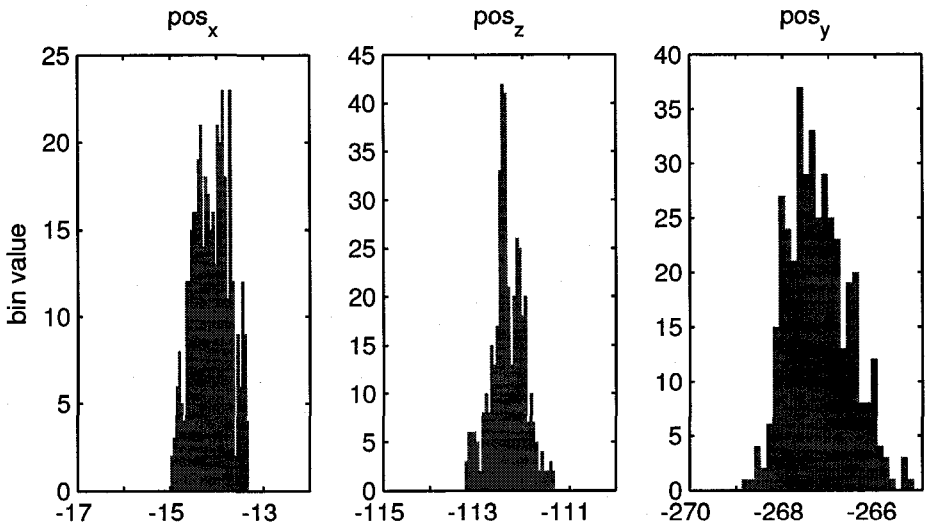
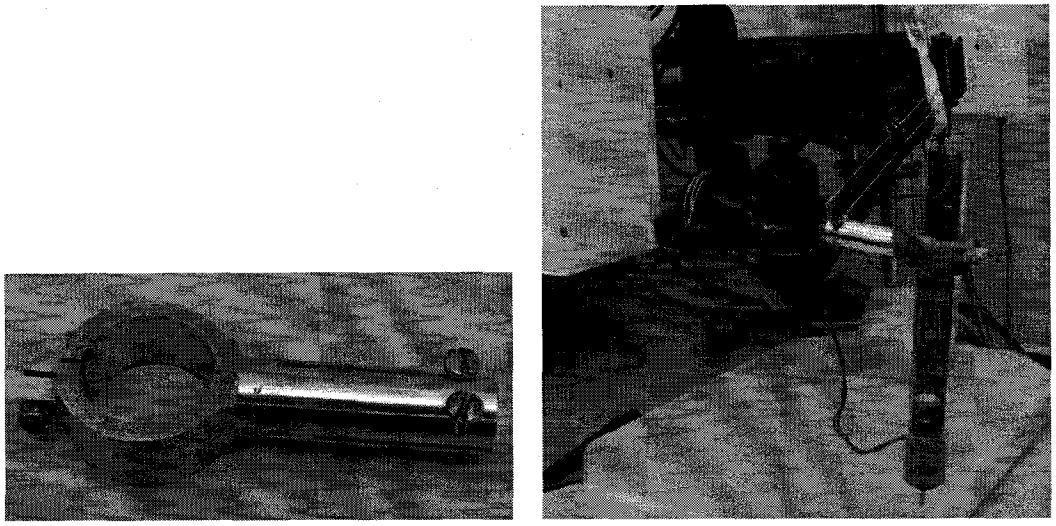


Figure 9: Histograms showing the distribution of tracker noise for position estimates. The probe was held a distance of $280mm$ from the camera lens, with the marker plane oriented parallel to the image plane. Variance for each dimension: $\sigma_x \approx 0.15mm$; $\sigma_y \approx 0.13mm$; $\sigma_x \approx 0.54mm$.

The force sensor of the WHaT is calibrated using a MPB Freedom 6S force feedback controller. Figure 10(a) shows a custom gripping mechanism specially designed for calibration of the touch probe. The forward kinematics of the Freedom 6S is adjusted so that the coordinate frame of the Freedom 6S handle is roughly aligned with the sensor of the WHaT, as shown in Figure 10(b).



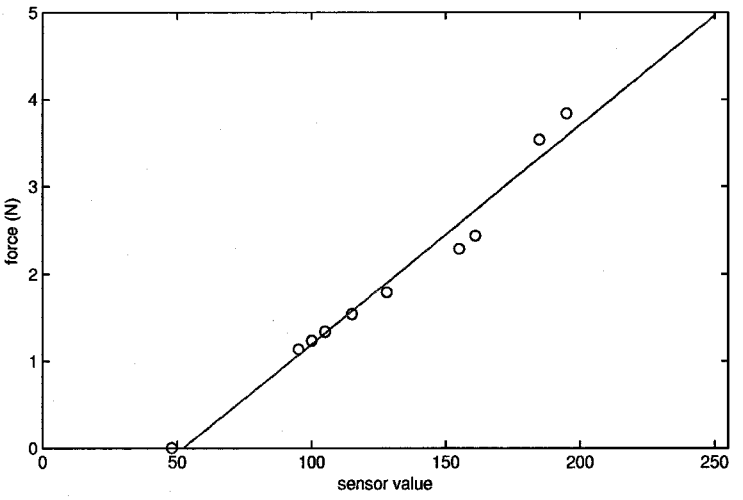
(a) The gripping mechanism for the WHaT. (b) The WHaT attached to the Freedom 6S robotic linkage.

Figure 10: Calibration apparatus.

The probe is placed in the gripping mechanism and positioned directly above a hard, flat surface and orientated so that the probe tip is perpendicular to the surface. A known downward force is applied using the controller software. Samples (see Figure 11(a))of the probe force sensor and Freedom 6S sensor are recorded for various force values over its estimated range.

sensor	force (N)
48	0.00
95	1.14
100	1.24
105	1.34
115	1.54
128	1.79
155	2.29
161	2.44
185	3.54
195	3.84

(a) Force sensor values used to determine force offset and scaling.



(b) Plot of force vs. sensor values, showing samples and straight line fit.

Figure 11: Calibration values for the WHaT force sensor.

A straight-line fit[5] of this data (see Figure 11(b)) provides a scaling and offset value for the probe force sensor, and allows us to determine a real-world force value from raw sensor data.

Originally, calibration for the accelerometers used a simple in-field method [12], in which a scalar and offset value were determined for each sensor axis (e.g., ${}^{s}a = m^{p}a + b$). This was done by aligning the axes with the gravity vector in the negative and positive directions and computing a straight line fit to the data. The resulting calibration model was very similar to the one used for the force sensor. However, there proved to be large inaccuracies in this model for certain orientations of the touch probe. This is illustrated in Figure 12, where actual sensor values are compared to the values predicted by this model.

[5]Technical specifications for the LPM562 force sensor quote a linearity of ±2% over the range of the component.

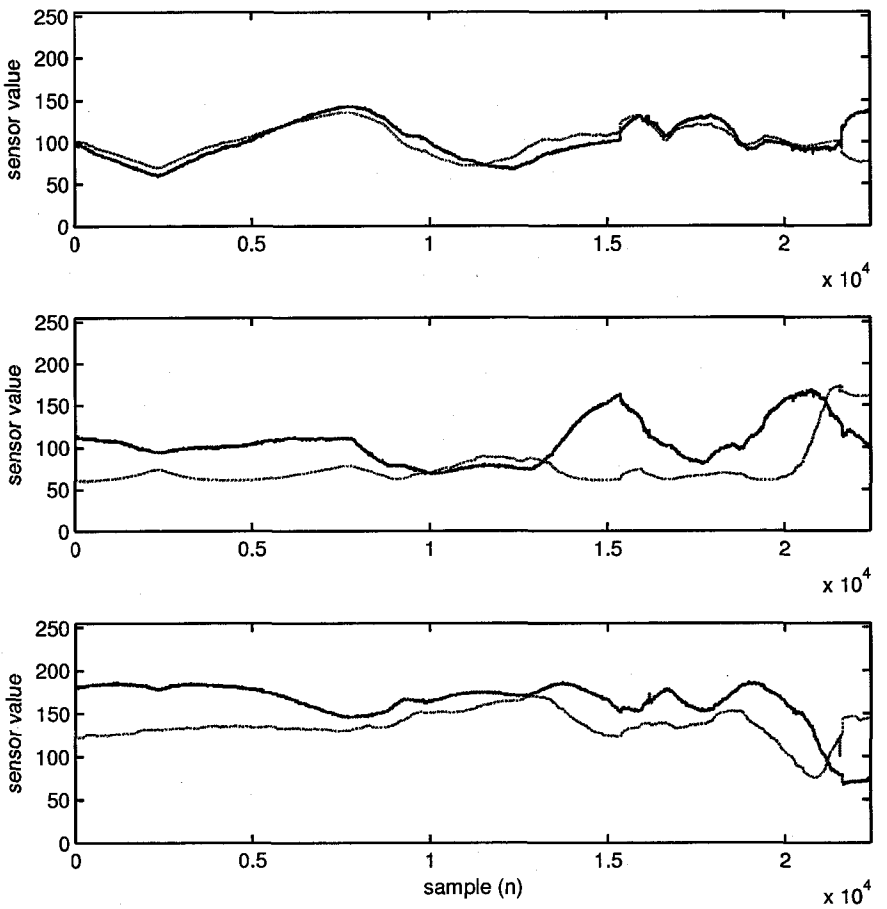


Figure 12: The scalar-offset model, showing the actual (solid) and predicted (dashed) sensor values for the i (top), j (middle), and k (bottom) sensor axis. There is large error for some orientations of the probe, which we attribute to non-orthogonal alignment of the accelerometers

The Freedom 6S was again used to determine calibration parameters, this time for a better-suited model. The orientation of the probe (obtained from the forward kinematic matrix of the Freedom 6S) is used to estimate the gravity vector in the sensor coordinate frame. The probe was placed in the gripper and moved slowly (in order to avoid contamination of gravity vector data) to various orientations and positions in the Freedom 6S workspace. We constructed a model, B, which maps an acceleration vector in the marker coordinate frame, ${}^{p}a$, to accelerometer values in the

sensor coordinate frame, ${}^{s}a$ (e.g., a raw sensor value). This linear model incorporates the scaling, rotation, and translation in order to map between the two coordinate frames. We define our model as:

$$ {}^{s}a = Bg + C $$

where the size of B and C is dependent on the number of parameters used in the vector g. Parameterization was performed a linear and a quadric model. For the linear model, B is a 3-by-3 matrix and C is a 3-by-1 vector containing offset constants. For the quadric model, B is a 9-by-9 matrix. The three parameters used by the linear model are the acceleration components of each axis of the p coordinate frame (e.g., $g =^{p} \vec{a}$). Hence, the linear model is composed of the system of linear equations:

$$ \begin{aligned} {}^{s}a_i &= b_{11}{}^{p}a_x + b_{12}{}^{p}a_y + b_{13}{}^{p}a_z + c_1 \\ {}^{s}a_j &= b_{21}{}^{p}a_x + b_{22}{}^{p}a_y + b_{23}{}^{p}a_z + c_2 \\ {}^{s}a_k &= b_{31}{}^{p}a_x + b_{32}{}^{p}a_y + b_{33}{}^{p}a_z + c_3 \end{aligned} $$

and for the quadric model, a 2nd order version of these equations is used. We obtain N measurements from the Freedom 6S and WHaT for orientation and acceleration data (e.g., $N \approx 20,000$ for our experiments), and define our measurement arrays as:

$$ D = \begin{bmatrix} {}^{s}a_{i_1} & {}^{s}a_{j_1} & {}^{s}a_{k_1} \\ {}^{s}a_{i_2} & {}^{s}a_{j_2} & {}^{s}a_{k_2} \\ \vdots & \vdots & \vdots \\ {}^{s}a_{i_N} & {}^{s}a_{j_N} & {}^{s}a_{k_N} \end{bmatrix} $$

and

$$ G = \begin{bmatrix} {}^{p}a_{x_1} & {}^{p}a_{y_1} & {}^{p}a_{z_1} & 1 \\ {}^{p}a_{x_2} & {}^{p}a_{y_2} & {}^{p}a_{z_2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ {}^{p}a_{x_N} & {}^{p}a_{y_N} & {}^{p}a_{z_N} & 1 \end{bmatrix} $$

where D is the measurement array obtained from the WHaT, and G is the parameterized measurement array obtained from the orientation of the Freedom 6S. Array D is related to G by B and C. We calculate B and C by using a least squares method which minimizes the length of the vector $Bg+C-{}^{s}a$. A benefit of performing this type of calibration is that it is possible to account for errors in the alignment of the accelerometer sensors. Due to manufacturing imprecision, these sensors have not been mounted orthogonally and it is necessary to account for this error in order to obtain a true acceleration value of the probe tip.

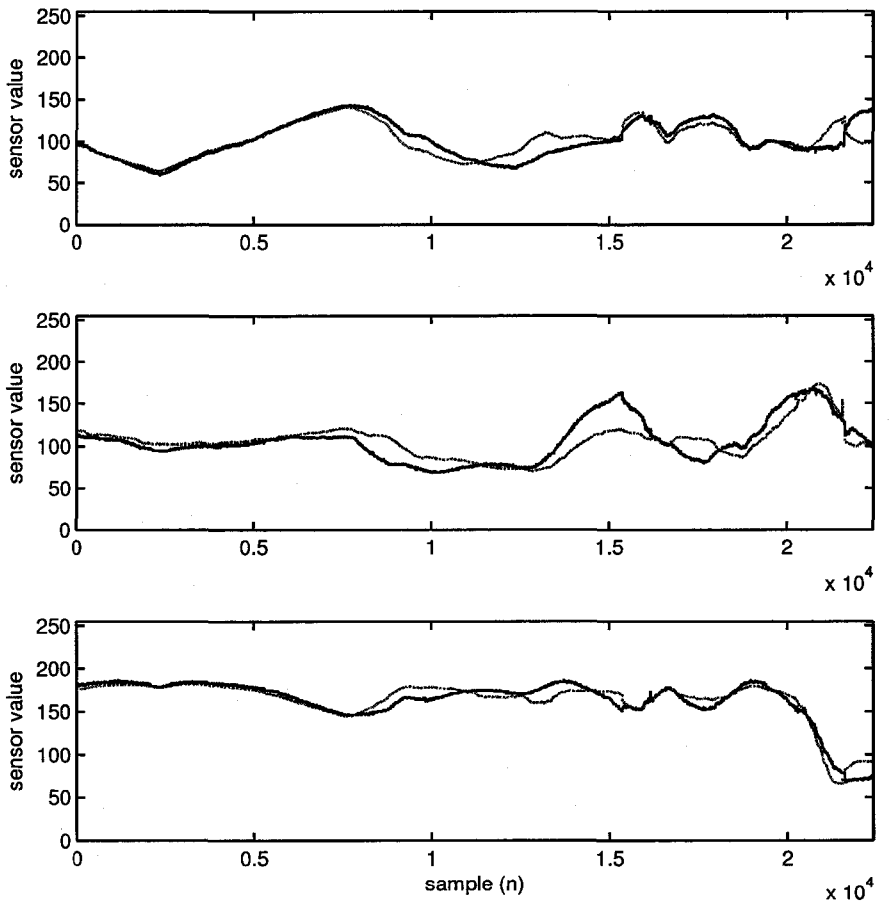


Figure 13: The linear model, showing the actual (solid) and predicted (dashed) sensor values for the i (top), j (middle), and k (bottom) sensor axis. This model shows significant improvement over the scalar-offset model.

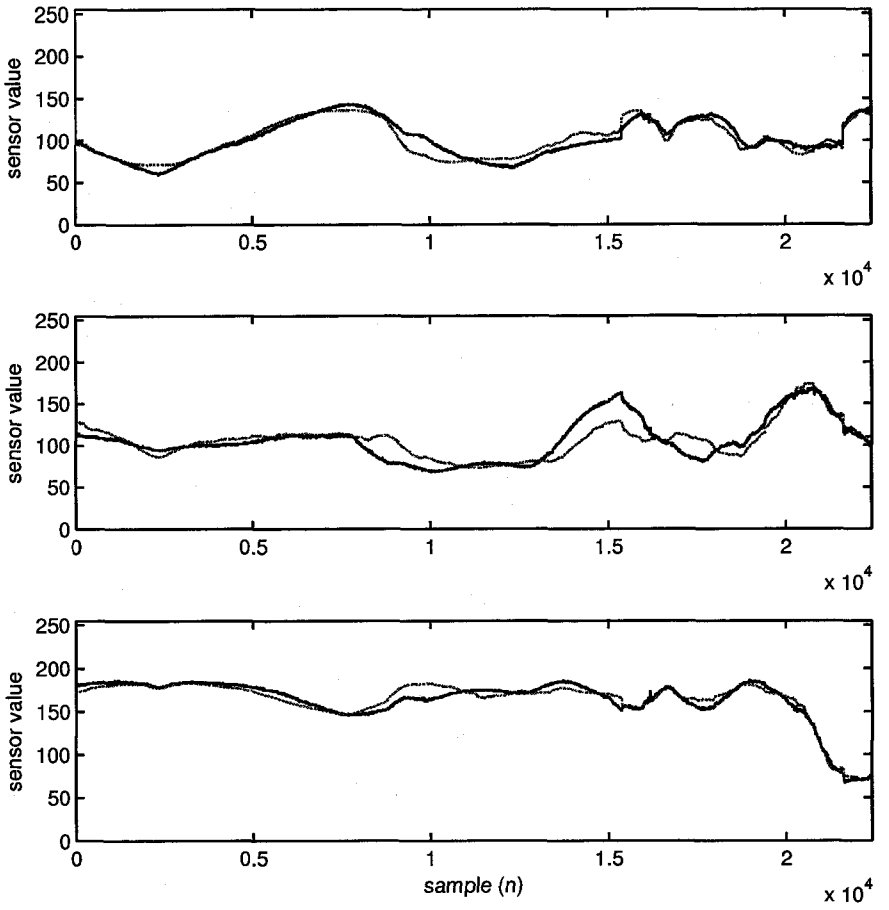


Figure 14: The quadric model, showing the actual (solid) and predicted (dashed) sensor values for the i (top), j (middle), and k (bottom) sensor axis.

Figure 13 shows a plot of actual acceleration vector and the resulting acceleration vector estimated using the linear model. Using a quadric model with additional parameters results in little improvement of the overall error of acceleration estimation. This is illustrated by comparing the results of the linear model (see Figure 13) to the quadric model (see Figure 14).

The B matrix and C vector used by our system are:

$$B = \begin{bmatrix} 53.4307 & 15.5323 & -7.0840 \\ 11.8876 & 6.9135 & 53.1222 \\ -21.1769 & -55.6179 & -8.5817 \end{bmatrix},$$

$$C = \begin{bmatrix} 110.6883 \\ 120.6481 \\ 122.0551 \end{bmatrix}.$$

We determine the suitability of the linear model by performing *analysis of variance* of the model fit and parameterization. The sensor measurement data and the predicted sensor values are used to calculate the variance of the model fit, σ_{fit}, such that:

$$\chi^2 = (D - (BG + C))^T (D - (BG + C)$$
$$\sigma_{fit} = \sqrt{\frac{\chi^2}{N}}.$$

From this step, the variance of the linear model fit for the i,j, and k axis were found to be $\sigma_i \approx 11.24$, $\sigma_j \approx 14.98$, and $\sigma_i \approx 8.16$. Performing singular value decomposition (SVD) on G, we obtain a covariance estimate of the parameter values, which we can use to estimate the variance, $\sigma_{parameter}$, of each parameter in the model (e.g., $b_{11}, b_{12}, ..., c_1$) :

$$[U, S, V^T] = svd(G)$$
$$\psi^{-1} = V(S^{-1})^2 V^T$$
$$\sigma_{parameter} = \sigma_{fit}.$$

where U,V,S are the two orthonormal matrices and the gain matrix, respectively, obtained from SVD factorization, and ψ^{-1} is the covariance matrix for the parameter measurement array G. A variance estimate, $\sigma_{parameter}$, was found for each parameter in the linear model. Note, we have simplified our analysis by assuming (i) that the measurements from the Freedom 6S and probe sensors have a Gaussian error distribution, and (ii) by excluding the sensor measurement array D from the analysis.

The variance analysis allowed us to determine that the linear model was well-suited to remove gravitational acceleration from our acceleration profile based on the

probe orientation gathered from the tracker. The inverse of model (e.g., B^{-1}) is also used to map accelerometer sensor data from the s frame to the p frame, and from there into global acceleration vectors.

Chapter 4

Data Acquisition and Signal Processing

Having well-conditioned sensor data is important for synthesizing accurate versions of the textures we scan. Therefore, some filtering and digital processing is performed on the raw data we acquire from the sensors. First, data from the WHaT and visual tracker is merged using a synchronization step. Then, noise is removed from sensor profiles using a filtering technique. Finally, we attempt to compensate for anomalies which arise due to the "human-in-the-loop" aspect of our system.

4.1 Sensor Acquisition and Synchronization

Sensor data is collected from the WHaT and visual tracker at uncoordinated, non-uniform sampling intervals. The tracker collects position and orientation information about the probe at a sampling rate of approximately $45Hz$; the WHaT probe collects force and accelerometer data about the probe tip at a rate of approximately $400Hz$. Later steps in our system require that data from each of the sensors has a common, uniform sampling period. Therefore, a time stamp is attached to samples from the WHaT and tracker so that they may be synchronized in a post-processing step.

Preliminary to synchronization, *outlier rejection* is performed on the force and

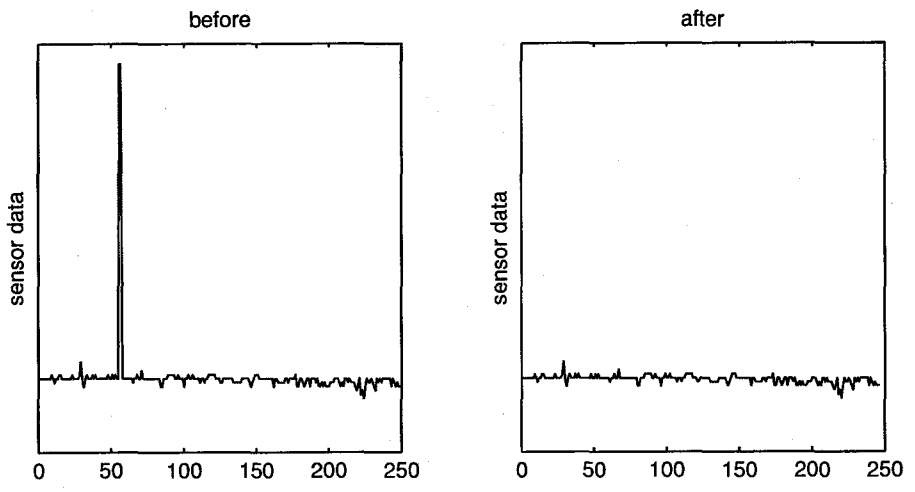


Figure 15: Removal of "spikes" from an acceleration profile, which are detected using a sample-to-sample threshold. Using a more reliable communication link may alleviate this problem.

acceleration profiles obtained from the touch probe. This is necessary since the wireless communication link of the WHaT is prone to periodic dropout errors. Here, we consider outliers to be samples which are numerically distant from neighbouring measurements. These are easily detected by "spiking" in the sensor data, usually across all fields of the communication frames. Figure 15 shows an example where a communication error is detected and successfully removed from the sensor profile by thresholding on the rate of change in the sensor data. The proceeding synchronization step interpolates over the gaps left by outlier rejection.

Sensor data is synchronized using a uniformly spaced time vector with a period of $5ms$. This is larger than the mean sampling period of the touch probe ($2.5ms$), but much smaller than the mean sampling period of the tracker ($22ms$). First-order linear interpolation is used to re-sample the timestamped sensor data using the new time vector. This synchronizes data from the two sensors, giving them a uniform sampling period, and is important for later steps involving digital signal processing techniques on the sensor data and surface profile.

Since the orientation of the probe is stored as a quaternion, spherical linear interpolation (SLERP) [57] is used to determine intermediary values. For more details on

SLERP we direct the reader to Appendix A.

4.2 Kalman Filtering

A *Kalman filter* is a recursive filter which estimates the state of a dynamic system when incomplete or noisy measurements are being used. It is popular due to its simplicity, optimality, and robustness. This is useful for estimating the position and orientation of the probe, since estimates from the ARTag tracker can sometimes be noisy due to hand jitter, illumination variation, or other environmental factors.

We use a linear Kalman filter [58] with noise modeled as a zero mean normal distribution. An observation noise covariance matrix is built using measurement variation estimated during a calibration step (see Section 3.5); a process covariance matrix is estimated based on average scanning velocities.

The algorithm for the filter is composed of two steps: predicting and updating. The system state, x, is predicted using previous measurements and the process model. For tracking the probe position p, $x = [p\_\dot{p}]^T$; for the probe orientation q, $x = [q]^T$. Using these parameters, the filter generates a current estimate of the system state. Then, the filter updates this estimate by using a current measurement of the system state, taking into account the observation noise. Note, since the orientation of the probe is stored as a quaternion it must be normalized after the Kalman filter is applied. For estimating the position of the probe, we define the filter parameters as

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

$$Q = \begin{bmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\frac{dx}{dt}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\frac{dy}{dt}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\frac{dz}{dt}} \end{bmatrix},$$

$$R = \begin{bmatrix} \varsigma_x & 0 & 0 \\ 0 & \varsigma_y & 0 \\ 0 & 0 & \varsigma_z \end{bmatrix}$$

where F is the state transition matrix, H is the observation model, Q is the process noise, and R is the observation noise. σ and ς represent the covariance estimates for the state transition and observation models, respectively, and Δt is the re-sampling period introduced during the synchronization and interpolation step (e.g., $5ms$ for our experiments). Notice that the state includes a velocity term, e.g., $\frac{dx}{dt}$, which is estimated along with position. Adding this parameter gave an improved estimate of the probe position, even though the velocity is not measured by the tracker. Similar matrices are used for estimating probe orientation. During the *predict* stage of the filter, the system state is updated according to:

$$\hat{x}_k = F_k x_{k-1} + w_k$$

where $\hat{x}_k$ is the predicted system state at time interval k, and w_k is the process noise, computed according to Q. The system state is reinforced using the sensor measurements from the tracker during the *update* stage of the filter, such that:

$$y_k = z_k - H_k \hat{x}_k + v_k$$

where z_k is the measurement at time k, y_k is called the *measurement residual*, and v_k is the measurement noise, computed according to R. The measurement residual is a measure of the error between the state estimated by the process model, $\hat{x}_k$ and the observed state, z_k. The final state estimate is determined according to:

$$x_k = \hat{x}_k + K_k y_k$$

where K_k is called the *optimal Kalman gain.* As can be seen, the final state estimate, x_k, incorporates both the measurement estimate from sensor and the estimate from the process model, which defines the state transition from one time interval to the next. In our experiments, the F, H, Q, and R matrices were static for all time intervals.

By using a standard Kalman filter, we assume the state transition and observation models are linear functions of the system state. For many non-trivial systems, this is not the case since there is usually some non-linearity associated with the process and observation model. For simplicity sake, we have opted not to use the *extended Kalman filter* (EKF) [59], which is capable of using more complex, non-linear models.

Estimating the probe state by using techniques which reconstruct a Markov chain (MC) were also considered, such as *particle filters.* Particle filters [60] are known to be more accurate than the EKF, approaching an optimal *Bayesian estimate* of the system state when given enough samples. Like the Kalman filter, it is a recursive filter which updates its estimation of the system state by predicting and observing. However, the benefit of this filter is that it can use non-linear transition and observation models and non-Gaussian distributions for noise. It is also notably more complex than the standard or extended Kalam filter and requires a large number of samples to obtain a near-optimal estimate.

We have seen no evidence to support use of the EKF or particle filters for our application. Based on our assumptions that: a) the dynamics of the probe position and orientation are linear, and b) the observation noise from the tracker is of Gaussian distribution, the standard Kalaman filter should provide optimal results. However, a formal comparison which benchmarks the results of other filters at correctly estimating the probe position and orientation may be considered for future work.

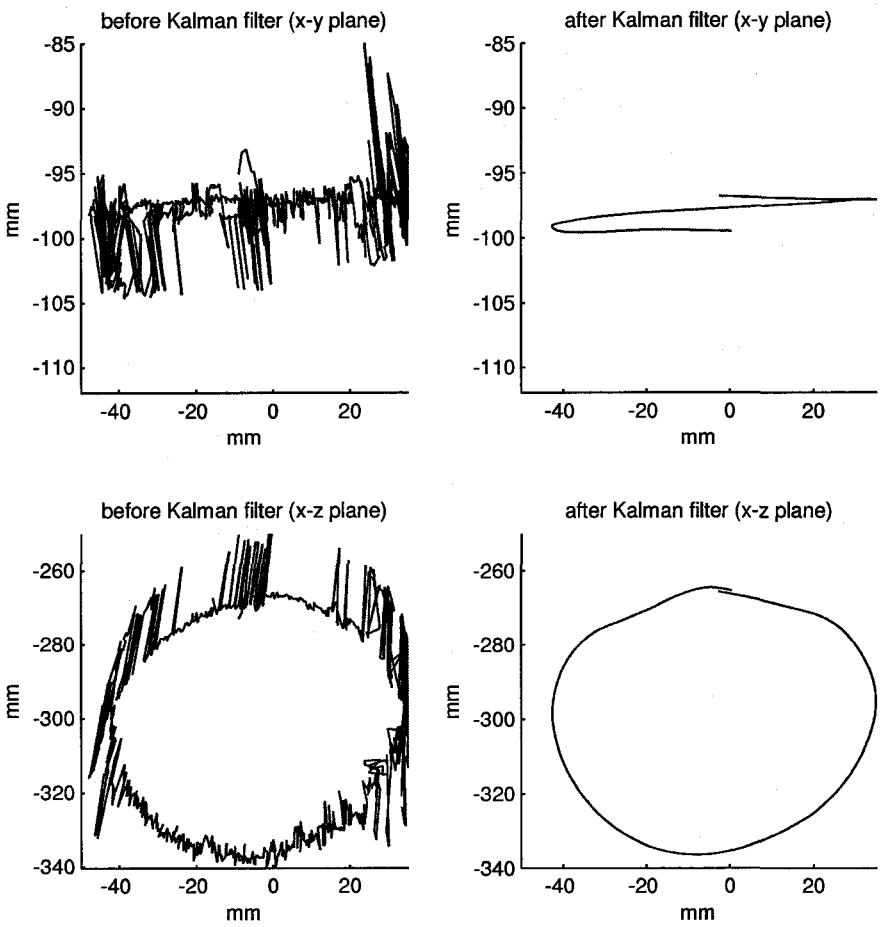


Figure 16: The outline of a circle was traced on a planar surface using the probe, showing raw tracker position (left side) and Kalman filtered position (right side). The diameter of the circle was approximately $78mm$ (circumference $245mm$). The x-y planar view of the probe position (top) and x-z planar view (bottom). Note that the nodal position of the camera is $(0, 0, 0)mm$.

Figure 16 shows a trajectory of the probe before and after Kalman filtering. The outline of a circle was traced, by hand, over a planar surface using the touch probe. The circle diameter is approximately $78mm$, which is reflected in the Kalman filtered plot. Much of the tracker noise has been removed by the filter, resulting in a smoothly transitioning path along the perimeter of the shape. Some error due to the inaccuracies of tracing a shape by hand are inevitable, but the probe trajectory

has been been produced with a fair degree of accuracy. Note that there is a certain amount of skewing in the vertical direction (y-axis) of the reconstructed trajectory, but this is relatively small when considering the length of the scanning trajectory (e.g., approximately a $3mm$ offset over a $254mm$ circumference, or 1.2%). Note that a discussion on the possible improvements to the Kalman filtering step is given in Section 7.2.

4.3 Acceleration Blending

Since the operator is free to use the probe in arbitrary orientations, the pose of the probe relative to the scanned surface must be recovered in order to determine the acceleration of the probe tip $\vec{a}_n$ in a direction normal to the surface. The surface profile is generated based on the probe's motion in a normal direction relative to the surface being scanned. The normal vector $\vec{n}_i$ may be obtained by registering the path of the probe tip with a 3D mesh (which is constructed at another stage in the 3D scanning pipeline). For simple surfaces, e.g., planar surface or bodies of revolution, estimates may be obtained directly from the trajectory of the probe.

Once the normal vectors along the probe path have been determined, the orientation of the device may be used to estimate *blending weights*, w, for each accelerometer in order to obtain the acceleration of the probe-tip normal to the surface. We calculate the weights using the dot product of the scanpath normal vectors and each of the i, j, and k-axis vectors from the sensor coordinate frame (see Figure 8). The dot-product determines the blending weight for each accelerometer, since they are aligned with the i, j, and k axes the weight vector may be written as:

$$\vec{w} = \begin{bmatrix} {}^c\vec{i}^T \\ {}^c\vec{j}^T \\ {}^c\vec{k}^T \end{bmatrix} \vec{n}.$$

In order to express the sensor axes in world coordinates, we transform them along the kinematic chain from sensor frame to marker frame and into the camera frame (see Figure 8), such that:

$$\begin{aligned} {}^{p}\vec{i} &= {}^{p}_{s}R\,{}^{s}\vec{i} \\ {}^{p}\vec{j} &= {}^{p}_{s}R\,{}^{s}\vec{j} \\ {}^{p}\vec{k} &= {}^{p}_{s}R\,{}^{s}\vec{k} \end{aligned} .$$

followed by the transformation into the camera frame based on the quaternion q obtained from the tracker:

$$\begin{aligned} {}^{c}\vec{i} &= {}^{c}_{p}q\,{}^{p}\vec{i}\,{}^{c}_{p}\overline{q} \\ {}^{c}\vec{j} &= {}^{c}_{p}q\,{}^{p}\vec{j}\,{}^{c}_{p}\overline{q} \\ {}^{c}\vec{k} &= {}^{c}_{p}q\,{}^{p}\vec{k}\,{}^{c}_{p}\overline{q} \end{aligned} .$$

The acceleration is projected into the direction of the unit normal, $\vec{n_i}$, of the surface. This results in each point i on the filtered path of the probe to have an associated acceleration vector:

$$\vec{a}_{n,i} = |\vec{w}_i\vec{a}_i|\vec{n}_i.$$

At this stage we also remove the gravitational acceleration since we know the global orientation of the probe. The linear calibration model obtained in Section 3.5 is used to map the gravity vector to accelerometer sensor values and the gravitational acceleration is subtracted directly from the acceleration profile.

4.4 Motion Compensation

For some sections of the scanning trajectory, the sensor data does not give a good representation of the surface profile of the object. This is due to variations in the motion of the probe tip as it moves over the surface, which are bound to occur with a human operator. Namely, *stick and slip* effect and *force variation* are two source of this error.

The stick and slip effect occurs when the probe tip becomes stuck on the surface due to friction, and continues moving only after the frictional forces are overcome by the operator. Sections affected by stick and slip are identified and removed by generating a velocity profile for the probe tip, using trajectory data from the tracker.

For sections of the scan path where the magnitude of velocity drops below a certain threshold, corresponding sensor values are dropped. This is done after the Kalman filtering step since measurement noise may falsely indicate that the probe tip is moving when it is not (e.g., variations in the position measurement between consecutive sample intervals be interpreted as velocity).

Since it is nearly impossible for a human operator to apply a constant force as they scan the surface, variations in the force profile will occur. Hence, some sections of the acceleration profile become amplified and others are attenuated. For example, if the operator presses the probe firmly against the surface, the acceleration profile, and correspondingly the resulting texture, will contain amplified features. Conversely, if the operator presses lightly, the texture features will be attenuated. This phenomena can be explained by Newton's second law.

Consider from Figure 17 that the acceleration of the probe tip due to the operator's pushing force $\vec{f_u}$ is (1). Therefore, the acceleration $\vec{a}_u$ along the major axis of the WHaT is given by (2) and $\vec{a}_u \propto |\vec{f_u}|$.

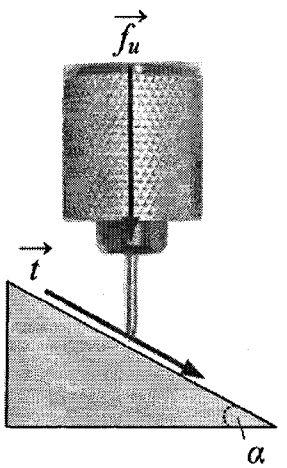


Figure 17: The motion of the probe tip due to a force applied by the operator. Scale has been exaggerated to show detail.

$$\vec{a} = \frac{\sin\alpha - \mu_c \cos\alpha}{m} |\vec{f_u}| \vec{t} \tag{1}$$

$$\vec{a}_u = (\vec{u} \cdot \vec{t}) \frac{\sin\alpha - \mu_c \cos\alpha}{m} \tag{2}$$

Therefore, we compensate for the influence of the applied force by a scaling step which adjusts for variations in the force applied by the operator. A scaling factor, s, is calculated for each entry in the acceleration profile a_n and is determined by:

$$s_i = 1.0 - \left(\frac{f_i - f_{mean}}{f_{max} - f_{min}} \right).$$

The adjusted acceleration value, $\vec{a_n}'$, is calculated as

$$\vec{a_{n,i}}' = s_i \vec{a_{n,i}}.$$

The force f_i is the i^{th} value obtained from the touch probe and f_{mean}, f_{max}, and f_{min} are the mean, maximum, and minimum values, respectively, of the force profile. We also apply a *moving average* smoothing filter to the force profile in order to avoid amplification of the high-frequency noise in the force measurement. This linear scaling factor does not correct any effects due to variation in the surface compliance.

4.5 Summary

In this chapter, we have presented the steps taken in order to condition sensor data for generating textures. This included identifying sources of error, removing noise from sensor data, and maintaining coherency among sensor data. The next chapter shows how this data is used to generate surface profiles and compliance values, which are the core of our haptic textures.

Chapter 5

Texture Synthesis

In the previous chapter, we discussed techniques used to process sensor data from the visual tracker and touch probe. These techniques were used to remove noise from the sensor data and to give better estimates of the probe trajectory, force and acceleration profiles. In this chapter, we synthesize textures using the processed sensor data. Also, we show how the probe scan path may be registered with a 3D polygonal mesh and used to generate texture coordinates for use in haptic rendering of the textures in a haptic application.

5.1 Generating Surface Profile

The surface profile represents the height-displacement relationship of the scanned surface. We generate the surface profile using the filtered acceleration data and scan path. The acceleration values, $\vec{a}_{n,i}$, are integrated using the interpolated time vector, $\vec{t}$. We use a *Verlet* integration scheme [61] to generate the height profile, h, from the acceleration data such that:

$$h_{i+1} = 2h_i - h_{i-1} + |s_i \vec{a}_{n,i}| \Delta t$$
$$\Delta t = (t_i - t_{i-1})$$

To establish a spatial relationship for the height data, we use the positions of the filtered scan path to determine a distance estimate along the surface for each

entry in our height profile. This is done by breaking the path into linear segments and calculating the lengths of each individual segment using Euclidean distance. The distance value at a point along the probe path, x_i, is equal to the summation of the lengths of all previous $i-1$ segments. The resulting surface profile function, $h(x_i)$, is a discrete function with intermediary height values found by interpolating over the range $[i, i+1]$. Also, by registering the path of the probe tip with a 3D mesh it is possible to mark regions of the mesh using different texture functions, as we will discuss in Section 5.4.

5.2 High-pass Filtering

After compensating for various sources of sensor error (using the techniques discussed in Chapter 4), surface profiles generated using the integration step have no significant high frequency noise; the remaining noise is predominantly low frequency. We use a high-pass filter to remove low frequency noise from the surface profiles which arises due to quantization error, synchronization errors, and *drift* in the conditioned sensor data. Figure 18 (top) shows an example of a surface profile generated by the integration of the acceleration profile, which evidently has large-scale and low-frequency errors. This type of error is common when a time-step integration scheme is used, such as the discrete time integration which is used to synthesize the surface profiles. To compensate, we follow the example set by Wall and Harwin [34] and remove the low frequency data from our surface profiles.

A 2nd order *Butterworth filter* [62] is used to remove low frequency data from our surface profiles. The Butterworth filter is a linear digital filter with an exceptionally flat frequency response in the pass band. Other linear filters, such as the Chebyshev filter, have a steeper roll-off in the transition band, but the Butterworth filter contains little to no ripple in the passband; it will not warp the frequency characteristics of the surface profile by introducing unwanted features due to ripple.

Wall and Harwin removed low frequency artifacts from their surface profiles by excluding the Fourier coefficients from their rendering equation which correspond to spatial frequencies below the bandwidth threshold of the commercial rendering

device. They concluded that the bandwidth of their commercial haptic device was not suited to render texture detail [63], particularly high frequency vibrotactile cues. Frequencies below $10Hz$[1], based on an average exploration velocity of approximately $100mm/s$, are removed from their profiles, and texture detail is rendered using a composite force-feedback haptic interface and frequency probe. While this method removed drift from their surface profiles, pertinent texture detail was also lost. Given that exploration velocity may vary significantly among users and that future haptic interfaces may be capable of rendering a larger bandwidth of vibrotactile signals, we opt not to use their approach.

Our approach is to use a cut off frequency, f_c, which best preserves the identifying features of the surface profile, but removes low frequency noise. Here, we use frequency to describe a spatial frequency (e.g., the unit for Hz becomes samples per mm). By removing frequencies in the spatial domain, there is no dependency on the scanning velocity. For our experiments, a cut-off frequency of $4Hz$ gave good results. Surface profiles for this step used height data spaced equidistant at $0.025mm$.

The cut off frequency was determined using trial and error on a sample of several surface profiles scanned from a variety of materials (see Figure 27). Figure 18 (middle and bottom) shows a surface profile after being processed by the filter. For $f_c = 2Hz$, some low frequency noise remained in the profile. A cut-off frequency of $f_c = 4Hz$ removed most of the low frequency drift and many of the texture features remained intact. Because this balance was found for many of the surface profiles we tested, $4Hz$ was chosen as a suitable cut off frequency. Features of spatial frequency $f < 4Hz$, which are removed by the filter, may be recovered using other scanning methods (e.g., by a 3D laser scanner [32]).

[1]Note that this is a temporal frequency, not a spatial one.

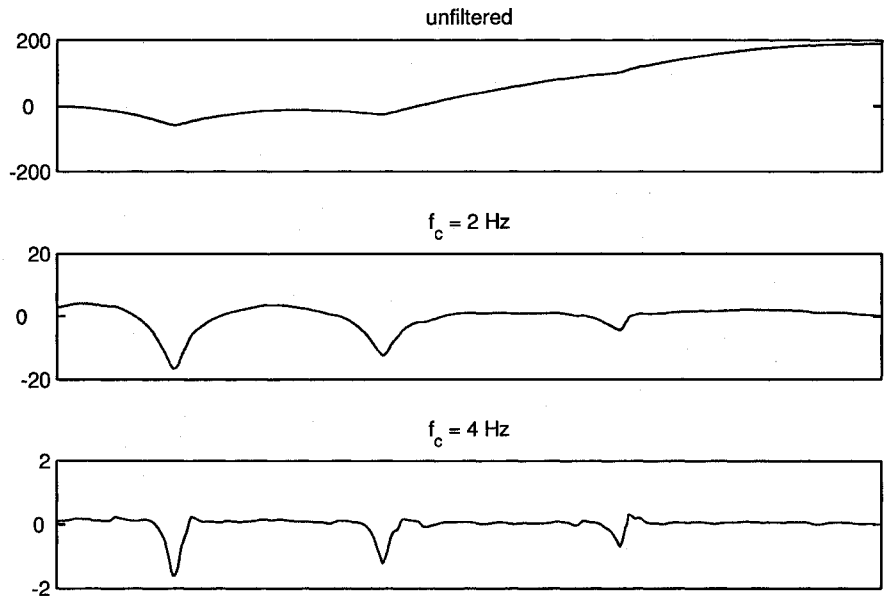


Figure 18: Surface profiles showing the height-to-displacement relationship of variations in a test surface. Shown are: unfiltered (top), high-pass filter $f_c = 2Hz$ (middle), high-pass filter $f_c = 4Hz$ (bottom). The high-pass filter removes low-frequency errors from the profiles.

5.3 Compliance

Compliance represents the force-displacement relationship of a deformable surface and is inversely proportional to stiffness. A surface's compliance value may be computed by deforming a surface (e.g., by pushing against it) and directly measuring the force generated at the contact point and the resulting displacement. For example, the apparatus used by Wall et al. [34] could be modified with a force sensor to measure the force generated by displacement as the probe scanned an object, and several displacement measurements could be used to generate a compliance estimate.

Other measurement equipment, such as *extensometers* and *universal testing machines*, are capable of measuring the stress-strain curve for a given sample of material, and a compliance estimate may be extracted from the elastic region of this curve. Extensometers measure changes in the length and diameter of a material specimen as

it undergoes tensile testing by a known force. These devices have some drawbacks which make them unattractive for integration into our system. For example, the measurement process typically results in the destruction of the material specimen. This is unacceptable, since if other measurement are to be performed on the objects we scan the object must remain intact. Also, this type of equipment is usually affixed to a bench-top or workshop floor and would hinder our goal of providing a mobile and interactive scanning system. Mobile *durometers* do exist, but these devices measure the *hardness* of a material, which represents an entirely different portion of the stress-strain curve.

It should be noted that while it is theoretically possible to directly measure the force-displacement based on sensor data from the WHaT, it is not a practical option given the limitations of the hardware. The force and displacement data required to determine a compliance estimate may be retrieved directly from the WHaT sensors (e.g., displacement may be recovered by discrete time integration of the accelerometer data). However, due to quantization error and drift (as discussed in Section 5.2), this results in a highly inaccurate estimate. Also, to perform this type of measurement the probe would ideally be moved very slowly as it deforms the surface of an object and the accelerometers in the WHaT are incapable of measuring changes in displacement at such low motion. In addition to problems with the resolution and range of the WHaT sensors, it is unlikely that the visual tracker would be capable of measuring displacement of the probe tip with a degree of accuracy for which a reasonable force-displacement relationship may be determined.

Instead, we opt to estimate the compliance of a surface by analyzing the force and acceleration profiles obtained from the WHaT. Our hypothesis is that stiff surfaces will have a force-acceleration ratio that is approximately constant; surfaces with a high compliance will have a force-acceleration ratio that varies depending on the applied force.

As previously discussed in Chapter 4 we expect the accelerometer readings to be linearly scaled by the user applied force. This is implied by Newton's second law,

$$\vec{f} = m\vec{a}.$$

Thus, for a rigid body of constant mass, m, the ratio of force, $\vec{f}$, to acceleration, $\vec{a}$, will also be constant. This is a fundamental of rigid body kinematics and holds true for the force and acceleration profiles of the touch probe if it scans an infinitely stiff surface. But compliant surfaces will deform due to applied force and the deformation will cause the surface profile to appear smoother with increasing force. As illustrated in Figure 19, the probe tip deforms the surface being measured. How much the surface is deformed is proportional to its compliance. Effectively, texture features are being "pushed" out of the way by the probe tip.

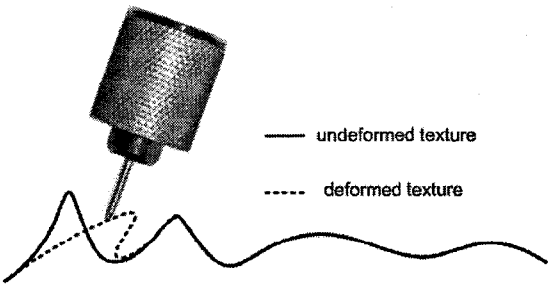


Figure 19: Deformation of the surface due to interaction with the probe tip.

We justify this hypothesis by examining the acceleration profile of two different surfaces, scanned once by pressing lightly and again by pressing firmly. Figure 20 and Figure 21 show the magnitude of acceleration profiles for scans of sandpaper and soft rubber, respectively. Based on our assumption, the acceleration profile of sandpaper will scale proportionally to force applied by the operator; the rubber will not.

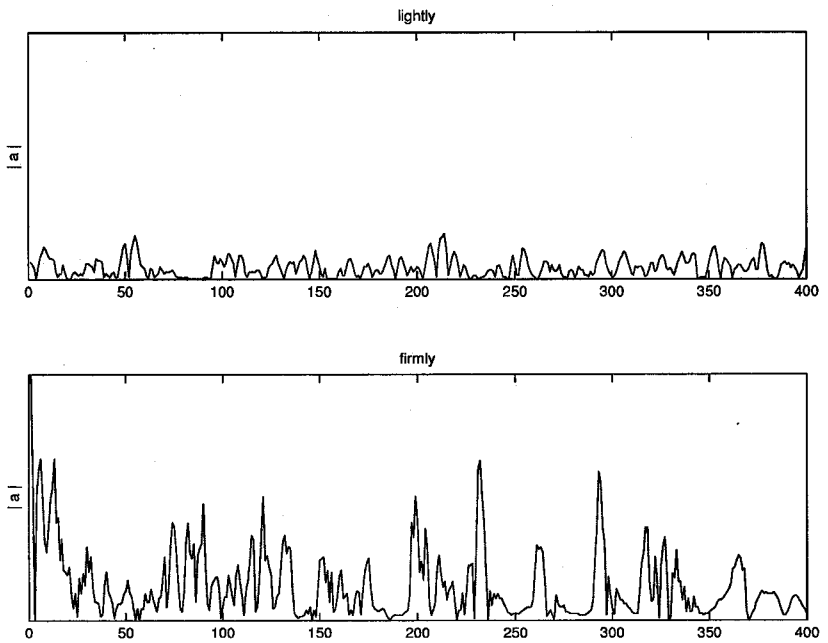


Figure 20: Acceleration amplitude profiles for sandpaper, scanned by pressing lightly (top) and firmly (bottom).

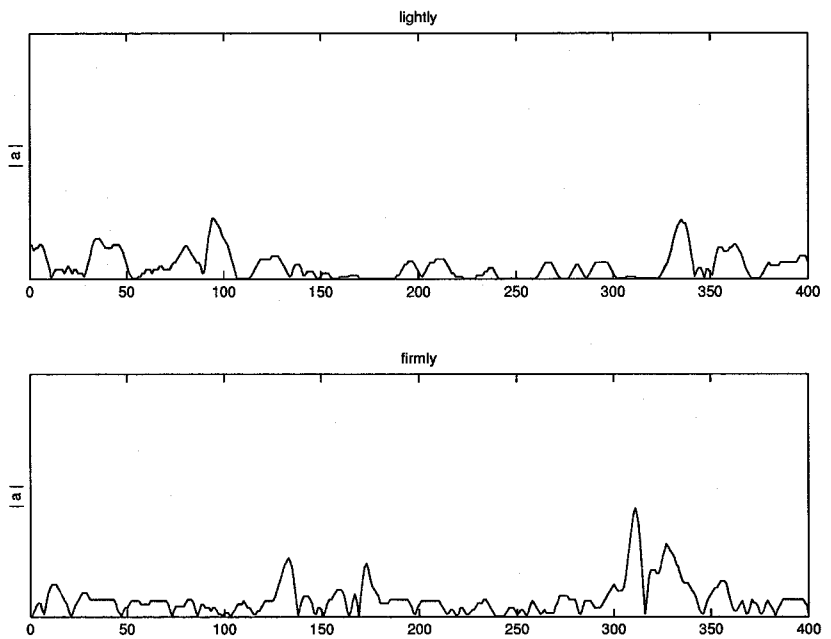


Figure 21: Acceleration amplitude profiles for rubber, scanned by pressing lightly (top) and firmly (bottom).

A *heuristic* compliance correction coefficient, k_c, which is based on the variance σ_r^2 of the force-acceleration profile, is computed as

$$k_c = \frac{1}{e^{\sigma_r^2}}.$$

The coefficient k_c is used by the haptic rendering algorithm for textures synthesized by our system. Note that $k_c \approx 1$ for stiff surfaces, and $k_c \to 0$ for very soft surfaces. An implementation example is presented and discussed in Chapter 6, where the compliance coefficient estimates accompany the surface profile function as part of a haptic rendering algorithm.

We should note that for compliance estimation, variations in the force profile are *useful* and thus compliance estimation is performed using the unscaled acceleration profile, not the scaled profile from Section 4.4. Otherwise, the force-to-acceleration ratio will not be a constant for stiff surfaces. We found that pressing lightly with the probe and then gradually increasing the applied force gave good results when scanning for compliance estimation. The results presented in Chapter 6 used this technique.

5.4 Texture Painting

Texture painting is the process of applying our textures to a virtual 3D object. This involves two steps: (i) registering the scanning trajectory with a 3D geometry, and (ii) generating texture coordinates on the surface of the 3D object. Registration involves transforming the scanning trajectory from one or more scans into a single coordinate system (e.g., the coordinate frame used by a polygonal mesh). Points from the scanning trajectory of the WHaT are matched with point defining a path over the surface of a 3D object. Geometry for the object is typically obtained using other scanning methods [11]. However, our system does not require that 3D geometry be obtained from a real-world object or even the same object that was used to generate the textures. Instead, artists may specify a virtual scanning trajectory over the surface of an arbitrary 3D object and generate texture coordinates based on this trajectory.

We register some of our textures with 3D geometry obtained using a Konica Minolta VIVID 910 3D Digitizer [32] (shown in Figure 22). The scanner uses a laser range finder and image-based techniques to estimate depth of points on an object's surface. A complete 3D model is obtained by rotating the object 360° on a turntable and acquiring depth information for multiple viewpoints. Scans are merged using a commercial software package, *Geomagic Studio* [64].

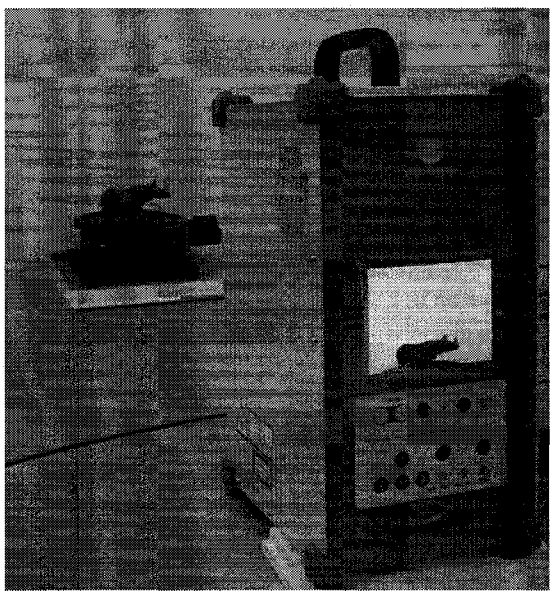


Figure 22: The Konica Minolta VI-910 3D Digitizer being used to scan the geometry of a toy rhinoceros.

We use an iterative closest point (ICP) algorithm [65] to match points from a scan path to points from a 3D mesh. ICP estimates the transformation of a path obtained from the ARTag visual tracker to a path over the surface of a 3D mesh. The ICP algorithm requires a good initial estimate of the transformation (rotation and translation) from the scan path coordinate frame to the mesh coordinate frame. This step is done by hand. The ICP algorithm begins by associating points from the scanning trajectory to points on the surface of the 3D mesh. Then, the algorithm attempts to find a homogeneous transformation matrix which minimizes some error value, in our case *distance*, between the scan path points and the associated mesh

points. An optimization algorithm is typically employed for this step. The ICP algorithm updates the scan points using the new transformation matrix and re-associates surfaces points, and repeats. Further details of the ICP algorithm are discussed in Appendix A.

For manual alignment of the scan path, six parameters are used– three translational values (x, y, z) and three rotation angles (α,β,γ), which correspond to translation and rotation about the x-, y-, and z-axis, respectively. Figure 23 shows a screen shot of the MATLAB application used to perform manual alignment of the scan path with the 3D mesh. The user may change the translation and rotation values of the transformation matrix by using the slider bars near the bottom of application window. The user is able to view the mesh and scan path from a top, bottom, front, or back perspective. The application accepts a polygonal mesh and scan path as input, and returns a homogeneous transformation matrix, which is built from the translation and rotation parameters, and returned to the calling MATLAB script. The ICP algorithm performs optimization directly on the homogeneous transformation matrix and not the six parameters used for hand alignment of the path.

The scan path is projected onto the 3D mesh, using a minimum distance criteria, as illustrated in Figure 24. The minimum distance between mesh polygons and each point in the scanning trajectory is found and these pairs are stored in an array. The minimum point on the surface of each polygon is found using *barycentric coordinates*, and since for our implementation we use triangular meshes to represent the 3D geometry, the problem may be stated as: compute the minimum distance, D between point $\vec{P}$ and triangle $\vec{T}(s,t) = \vec{A}+s(\vec{B}-\vec{A})+t(\vec{C}-\vec{A})$ such that $0 \leq s \leq 1, 0 \leq t \leq 1$, and $s+t \leq 1$, where $\vec{A}$,$\vec{B}$, and $\vec{C}$ are the three vertices of triangle $\vec{T}$ and s and t are the barycentric coordinates of a point on the surface of the triangle. A closed-form solution for D is computed for each triangle in the mesh and the lowest value is chosen as the associated mesh point.

The problem of finding an optimal transformation matrix which describes the transformation from the mesh point coordinates to the scanning trajectory coordinates is that of *absolute orientation.* For solving the relative transformation, we use a technique developed by Horn [66] which is used to solve the robot-world hand-eye

calibration problem. A least sum of squares algorithm is used to optimize a 4-by-4 homogeneous transformation matrix which maps the scanning trajectory onto the mesh. Each element of the matrix is considered to be linearly independent and optimization is performed on individual elements. This is not quite the case, since the 12 elements (a 3-by-3 rotation matrix and 3-by-1 translation vector) being optimized are actually composed of only 6 independent parameters– 3 rotation and 3 translational values. Therefore, it is possible that scaling errors are introduced in the resulting transformation matrix since coherency of the rotation portion of the matrix is not maintained. (e.g., all elements of a rotation matrix must be in the range [0,1]). In order to remove these errors, we finish by computing the singular value decomposition (SVD) of the rotation matrix, such that

$$H = \begin{bmatrix} \mathbf{R}\, t \\ \mathbf{0}\, 1 \end{bmatrix},$$

$$R = USV^T$$

where H is the homogeneous transformation matrix, R is the 3-by-3 rotation matrix, t is the translation vector, and the matrices U, S,and V represent the SVD factorization of R. The matrix S is important here since it controls the singular values, or gain, for the rotation matrix. Scaling is fixed by normalizing the diagonal values of this matrix and rebuilding the rotation matrix.

The ICP algorithm repeats until there is no noticeable change in the computed error value between iterations. The error is calculated using the average over all points of the distance from points in the scan path to their associated mesh points. Figure 24 shows an illustration of how the scanning trajectory projected onto a surface mesh for each iteration of the ICP algorithm. A version of the scan path with uniform, equidistant sampling was used to perform ICP alignment. Previous work [67] suggests some techniques (e.g., non-uniform sampling, outlier rejection, normal-space mapping) to improve or speed up the resulting transformation, but we have not yet implemented these for our system.

Convergence and uniqueness of the ICP algorithm is dependent on the initial

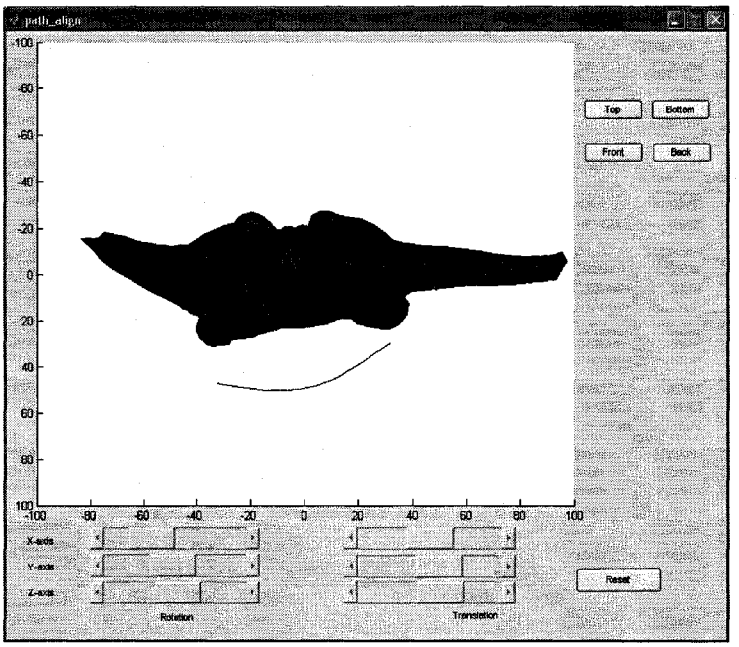


Figure 23: Manual alignment of the scan path in order to obtain an initial estimate for the homogeneous transformation matrix.

transform given by the user during the manual alignment step. If this "guess" is not close to the actual relative transform, the ICP algorithm may converge to a result which is locally optimal, but not globally. For example, in Figure 25 the ICP algorithm provides two different solutions for the relative transform. The error values for both solutions were similar ($< 0.1mm$ per point), but which is correct? This illustrates the sensitivity of the ICP algorithm to the initial user alignment and the importance of providing a good first estimate.

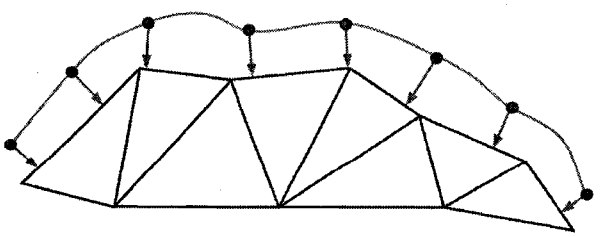


Figure 24: Association of mesh points with the scan path. Points from the scanning trajectory are projected onto the mesh using a minimum distance criteria.

Also, for meshes without much detail, the ICP algorithm may immediately converge on a solution, using just the manual alignment provided by the user. For example, consider a mesh representing a perfectly smooth plane. Scanning this surface would, theoretically, produce a curve which lies in the plane, and there are an infinite number of transformation matrices which would produce an optimal result for registration with the mesh. Therefore, the initial estimate provided by the user would be considered to be the "best" solution.

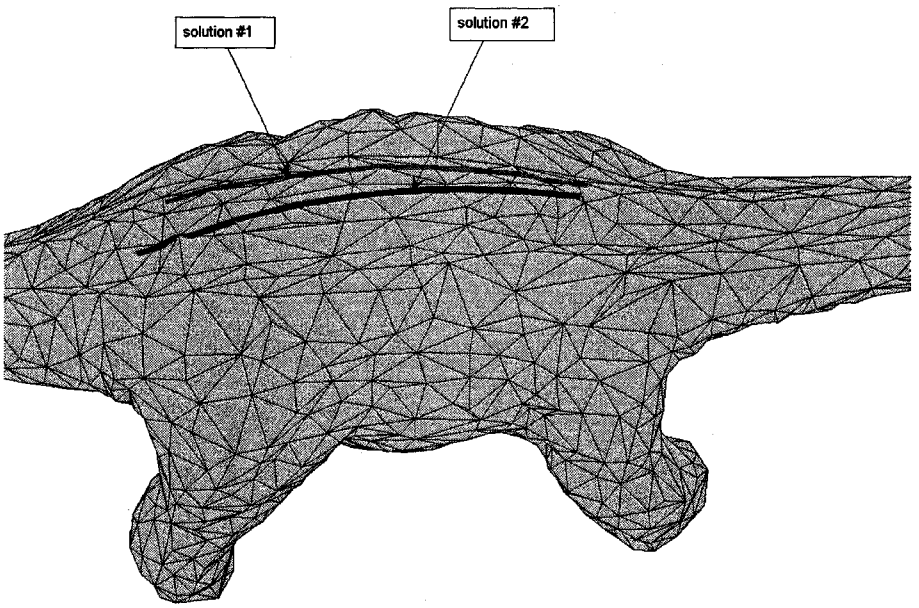


Figure 25: Convergence and uniqueness of ICP is affected by initial manual alignment of the scanning trajectory. Here, the ICP algorithm provided two unique solutions when given two different initial estimates for the relative transformation.

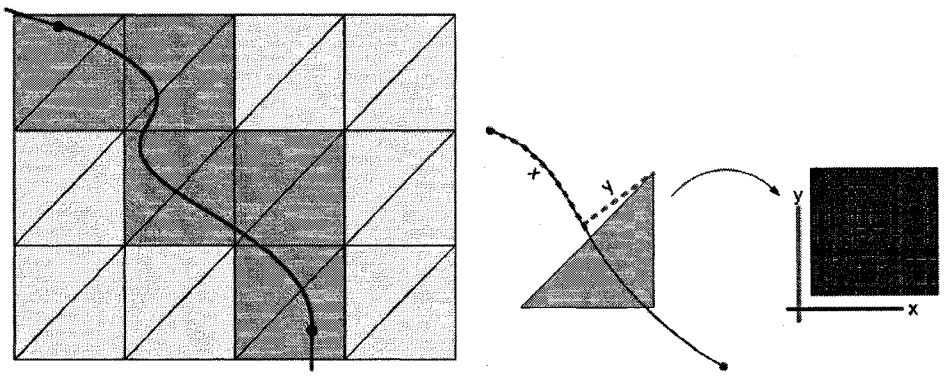
Texture coordinates may be assigned to elements of a 3D mesh directly from the set of mesh points resulting from scan path registration by the ICP algorithm. To better understand how texture coordinates may be generated, consider the texture function, $f(x, y)$ which represents the surface profile of the scanned surface. The function uses a 2D coordinate pair, or a single parameter for 1D textures, to generate a height value which corresponds to the texture height of that point on the object's

surface.

Therefore, we assign a 2D texture coordinate to each vertex in polygons from the 3D geometry mesh that are associated with the registered scan path. Polygons are selected by testing if points in the registered scan path intersect with the polygon boundaries. This has already been done in the distance minimization step of Section 5.4, since polygons may be selected when a point on their surface is the minimum distance to a point in the scan path. The polygons, their vertices, and minimum distance points are stored for each iteration of the ICP algorithm and we use the values obtained in the final iteration. In Figure 26(a), relevant polygons have been shaded and texture coordinates will be generated for each vertex of each polygon in this set. The polygon set may be extended to neighbouring polygons by using a breadth-first search of the mesh topology, since our algorithm for generating texture coordinates does not require that polygons intersect with the scan path or its path of associated mesh points.

Texture coordinates are generated for each vertex of a polygon by calculating the minimum distance to each line segment in the set of associated mesh points. Linear interpolation is used to find the position of the exact position along each segment in the scanning trajectory. We consider the scan path to be one of the parameter dimensions of the texture function, which corresponds to the signed distance *along* the scan path. For the other parameter, we estimate signed distance *to* the scan path, with positive and negative direction being arbitrarily chosen. This scheme of assigning texture coordinates is better understood by considering Figure 26(b). The y parameter is obtained by finding the minimum distance from a vertex in the polygon to a point on the path. Since the texture function accepts spatial values as parameters, this distance value may be used as one of the texture coordinates. The x parameter is also obtained using the path point for y. The distance along the scan path at that point is used for the other texture coordinate. For 1D textures, this value can be used as the single texture function parameter.

The texture coordinate generation method used by our system results in a texture mapping scheme that is independent of the texture which is being applied to the object. A texture may be transferred from one object to another simply by using



(a) Scan path superimposed on a polygonal mesh. Relevant polygons have been shaded.

(b) Generating texture coordinates from a scan path. For 2D textures, the x and y value are used; for 1D textures, only x is used.

Figure 26: Texture coordinates are generated from the associated scan path on the 3D mesh.

the same scanning trajectory, or by specifying a virtual scan path over the surface of a mesh. For example, many 3D modeling packages allow users to draw graphical textures and colour directly onto the mesh surface. This feature could be extended to allow drawing (e.g. "painting") a scanning trajectory from which texture coordinates are generated and a texture mapped onto the surface of the object.

We note that it is possible that spatial warping may occur in textures using the above technique. For example, if there is "pinching" or overlapping in the scanning trajectory, the texture coordinates may distort the texture at certain areas. Best results are achieved when well-formed scan paths are used to acquire and register a texture (e.g., the probe is moved laterally over the object, with no looping or "zig-zag" motion). Also, we do not expect discontinuities (e.g., Mach banding) to arise in the rendering of textures based on our technique for coordinate assignment. Provided the surface profile of a texture is continuous at its boundaries, it will feel seamless when a user explores it after being applied to a 3D object. One possible strategy here is only using sections surface profiles which are approximately C^1 continuous at their edges. In most cases, this can be confirmed by visual inspection. For polygonal meshes, another requirement is that neighbouring polygons have been textured using

the same scanning trajectory. Obviously, crossing between two differently textures sections of a mesh may result in noticeable changes at the boundaries.

5.5 Summary

The steps outlined in this chapter enable us to generate a texture based on the surface profile function and using the processed sensor data from Chapter 4 We also show how geometry from a 3D polygonal mesh may be textured using an ICP algorithm to match mesh points with scanning trajectory, and 2D texture coordinates may be assigned based on minimum distance estimates from the mesh vertices to positions along the scanning path.

If the reader is interested in knowing more details about our texture synthesis and texture coordinate implementation, the pseudo-code for these processes is presented in Appendix B.

Chapter 6

Results

6.1 Overview

In this chapter we present the "fruits of labour" for our haptic texturing system. Section 6.2 presents some surface profiles we have generated by scanning a selection of objects. We also comment on the accuracy and repeatability of our system by using a surface profile with clearly measurable features as a reference. In Section 6.4 the compliance coefficients for several materials are obtained, perceptually ranging from hard to soft. Finally, we show that the textures generated by our system may be incorporated as part of haptic texture models introduced in Chapter 2. Textured objects are rendered as part of hapto-visual applications, including a 3D polygonal mesh for which multiple scanning trajectories have been registered and 2D texture coordinates assigned.

6.2 Surface Profiles

Using the methods discussed in Chapter 4 and 5, we generate for the objects shown in Figure 27. The surface profiles generated from scans of these objects are shown in Figure 28. Each surface is clearly identifiable from the features displayed in their surface profiles and demonstrates the ability of our system to generate textures for stochastic and patterned surfaces.

As expected, the sandpaper and clay pot profiles show bumpy, stochastically varying surfaces. The sandpaper has a slightly larger amplitude and the clay pot a higher spatial frequency. The protoboard and cable tie each have identifiable valleys and peaks spaced at approximately $2.5mm$ and $1.2mm$, respectively.

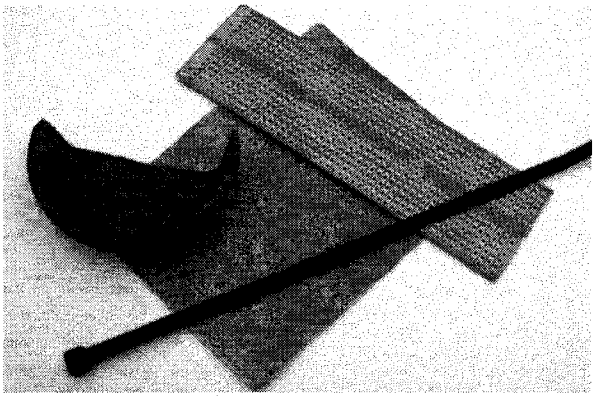


Figure 27: Scanned objects: a clay pot, cable tie, circuit protoboard, and sandpaper.

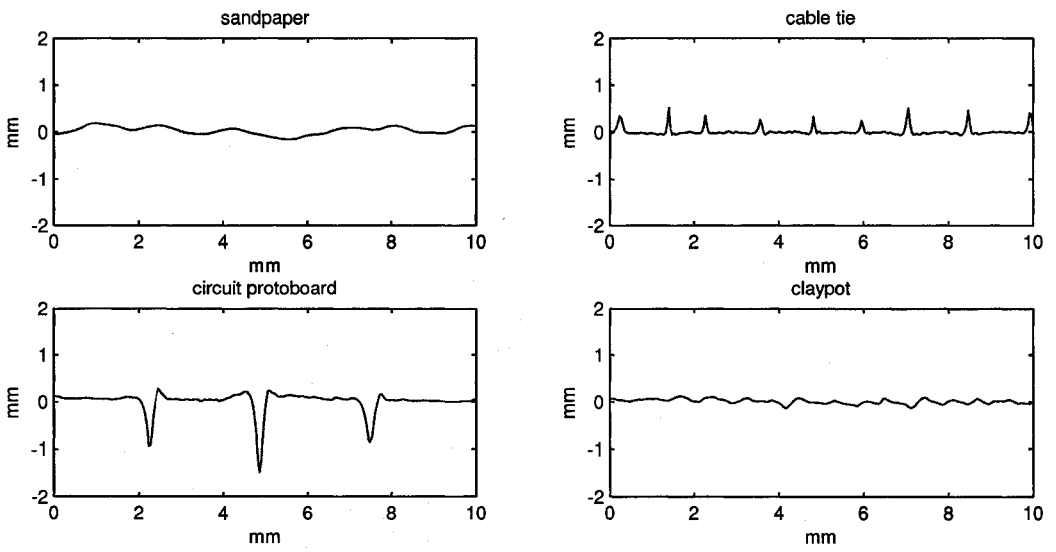


Figure 28: Surface profiles generated by our system.

Figure 29 shows several different surface profiles superimposed onto its scanning trajectory. This demonstrates the ability of our system to scan non-planar objects,

as well as its ability to apply textures to other geometries (e.g., "texture swapping"). The scan path may also be registered with the geometry of the clay pot, resulting in a textured 3D object.

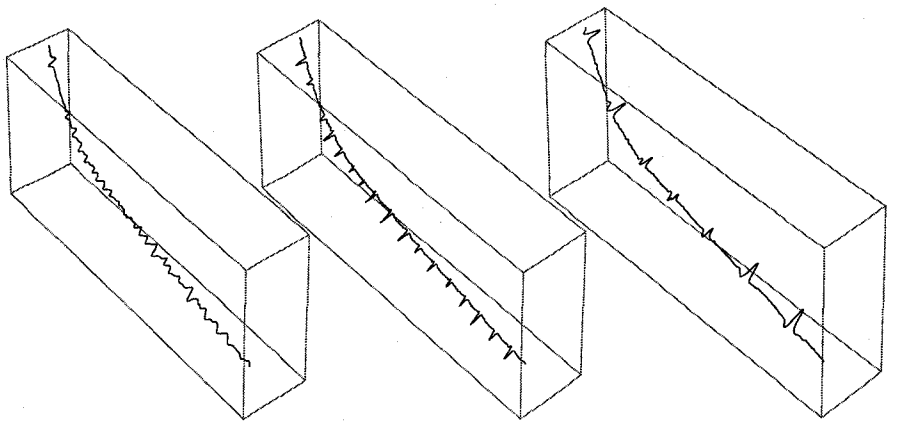


Figure 29: Several surface profiles superimposed onto the scan path for the clay pot. Scale has been increased to show detail. Textures may be "painted" onto a virtual 3D trajectory for different effects.

Now we will discuss the *accuracy* and *repeatability* of our system by scanning a texture with visibly measurable features and making some comparisons. For our purposes, a circuit protoboard is used since its features, such as hole spacing and depth, are measurable using conventional tools (e.g., callipers). The protoboard used here has a horizontal and vertical hole spacing of $2.5mm$, and hole depth of $1.5mm$.

Figure 30 shows the surface profile generated by our system after scanning the same object five times, separately. For each scan, we deliberately varied the scanning speed and orientation of the probe. This was an effort to test the robustness of our system not only for the acquisition of sensor data under various conditions, but the ability to reproduce similar surface profiles. The profiles in Figure 30 show an average hole spacing of $2.81mm$ with standard deviation of $0.11mm$. The average hole depth is $0.94mm$ with standard deviation of $0.14mm$, but there are some outliers. We believe this comparison this shows that our system is sufficiently precise to generate similar textures given various scanning conditions.

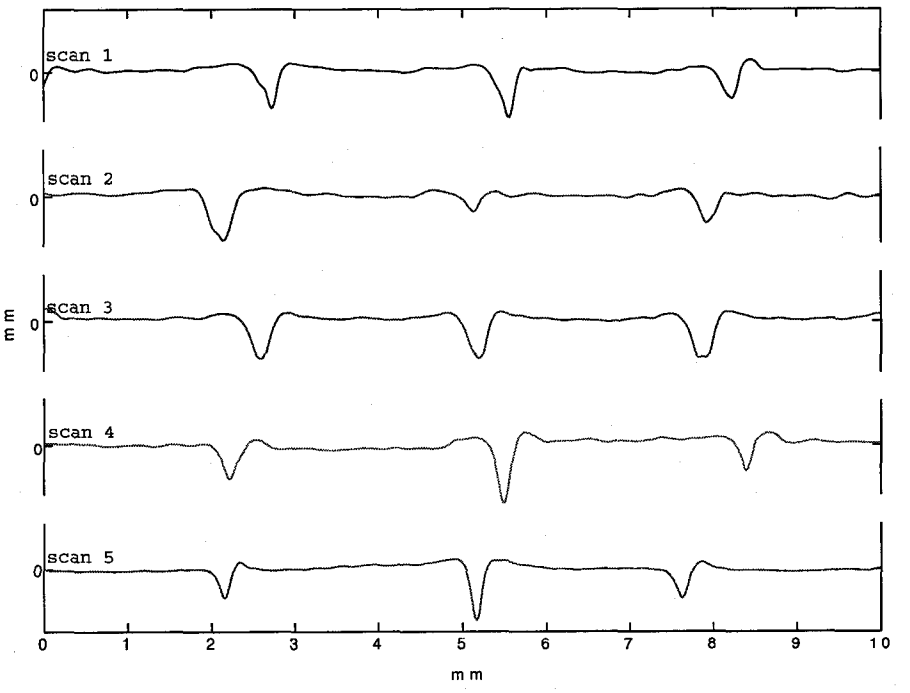


Figure 30: Repeatability: comparing surface profiles generated from multiple scans. The same surface was scanned multiple times without restrictions on the scanning speed, probe orientation, or force applied by the operator. Each of the scans exhibits similar characteristics.

Figure 31 shows a surface profile of the protoboard plotted versus a procedurally generated surface profile based on measurements of the protoboard. The hole spacing for the texture generated by our system is approximately $2.65mm$ and hence the alignment of the two profiles begins to drift after a few millimetres. However this drift is only about 6% over the entire texture sample, which we feel is acceptable. The average hole depth for our texture is approximately $0.81mm$, which is about half of the actual depth. Our system fares poorly here, but since the depth of the hole is consistent for the entire texture this error is easily fixed by scaling the height values of the surface profile.

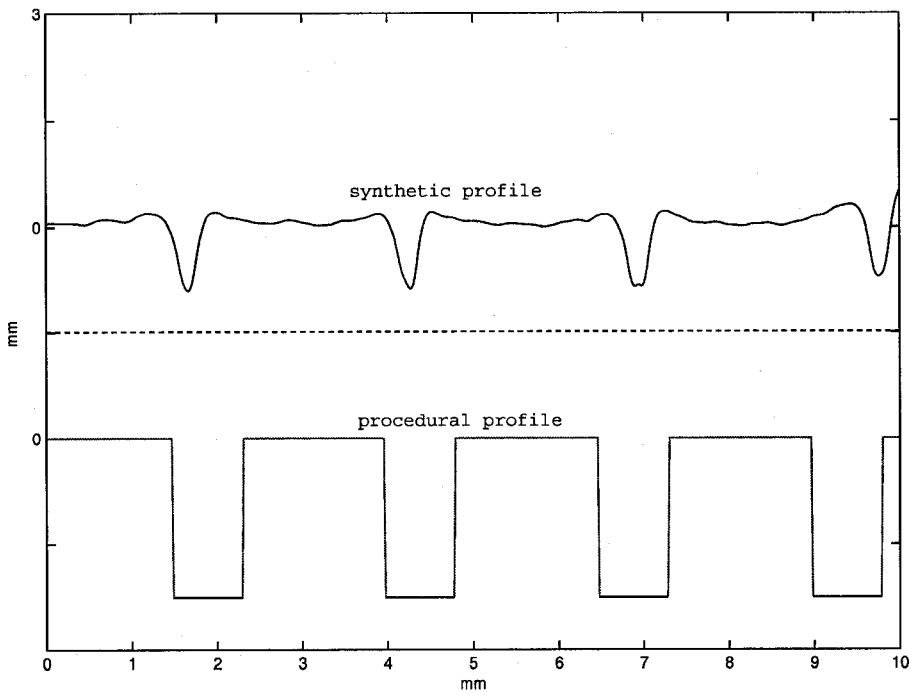


Figure 31: Accuracy: comparing the real and virtual texture features of a circuit protoboard. The texture created by our system (top) exhibits many similarities to the procedurally generated one (bottom).

One problem which is evident from the plots in Figure 30 and Figure 31 is the spatial drift which becomes noticeable after just a few millimetres. We believe this is the fault of the Kalman filter which is used to remove noise from the visual tracker data. As mentioned in Section 4.2, the filter does a good job at removing noise, but it isn't perfect. Also, the process and observation model parameters remain static for samples processed by the filter. Tracker measurement error is dependent on the distance and angle of the marker away from the camera lens, and experimentation has shown us that small changes to the parameters of the Kalman filter can improve the resulting texture profile.

We estimate the normal vectors for scans by fitting a plane to the scanning trajectory. This requires some careful scanning by the operator. For planar objects (e.g., the sandpaper, protoboard and cable tie), we make sure to scan a curved path and

simply fit a plane to the path. For the clay pot, which is a (broken) body of revolution, we scan normal to the rotation axis and again fit a plane. Since the normal vectors are used by the acceleration blending step outlined in Chapter 4, reconstruction of the correct scan path also makes height detail sensitive to the parameters of the Kalman filter. This provides additional motivation to use a dynamic process and observation model. Another method for estimating normal vectors is to use 3D geometry obtained when an object is scanned (e.g., using the Minolta digitizer), and registering the scanning trajectory with the polygonal mesh.

6.3 Texturing 3D Objects

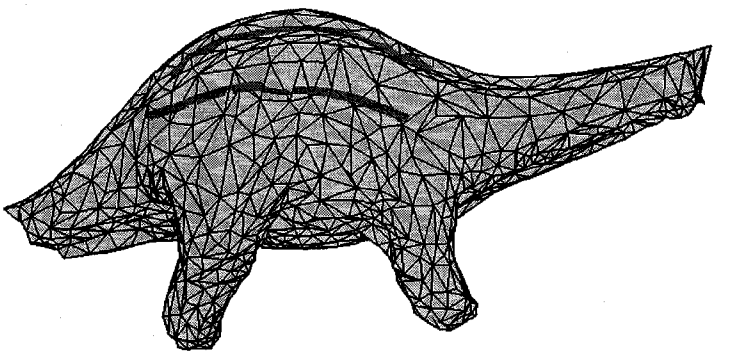


Figure 32: Scan registration using the geometry of a toy dinosaur.

In this section we register the scanning trajectory from multiple scans with 3D object geometry. Based on the associated path generated by registration process, we are able to automatically generate 2D texture coordinates using the technique outlined in Section 5.4. Being able to merge our textures with geometry obtained using the Konica Minolta 3D digitizer is a necessary step in order to integrate tactile scanning as part of the 3D scanning pipeline [11].

Textures generated from scans of a toy dinosaur were registered with a 3D polygonal mesh and displayed in a hapto-visual application. Shown in Figure 32, two scanning trajectories have been registered with the mesh topology. Paths over the

mesh surface are used to select which polygons are textured, and is extended to the surrounding 1-neighbourhood polygons.

We store object geometry and texture coordinates together using an augmented Wavefront *.obj* file, as shown in Figure 33. Texture coordinates are assigned per face, and coordinate values are denoted by a "h" appearing as the first character on a line. This character is followed by an index value and three texture coordinate pairs, which represent the 2D coordinates used to generate a height value from the texture function. The index is used to identify which texture should be used when more than one texture has been applied to an object. An index value of -1 indicates that no texture should be used for that face.

```
## vertices
v -40 0 40
v -40 0 -40
v 40 0 40
v 40 0 -40
## faces
f 1 3 2
f 2 3 4
## haptic texture coordinates
h 1 0.0 0.0 0.0 50.0 50.0 0.0
h 1 50.0 0.0 0.0 50.0 50.0 50.0
```

Figure 33: Example of an augmented Wavefront .obj file which contains geometry and texture information. The file shown here is for an 80x80 textured 3D plane.

6.4 Compliance

The compliance coefficients and variance of the force-acceleration ratio for several materials is listed in Table 1, which also correspond to the plots shown in Figure 34. Intuitively, these values are as expected: Sandpaper has a very rigid surface with $k_c \approx 1.0$; cork is slightly less rigid with $k_c \approx 0.8$; soft rubber is the least rigid of the three surfaces with $k_c \approx 0.02$.

We use the compliance estimates obtained here in a rendering application, which

Material	σ_r^2	k_c
sandpaper	0.0454	0.9556
cork	0.2581	0.7725
soft rubber	3.7127	0.0244

Table 1: Variance and compliance estimates of several materials: sandpaper, cork, and soft rubber.

we discuss in Section 6.5. The compliance coefficients are used to modify the stiffness (one is the mathematical inverse of the other) of textures when they are displayed to users at a haptic interface using a hapto-visual application. When using a very rigid compliance value, such as the sandpaper, the textures felt pronounced. The salient features of textures were easily detectable. For soft compliance values, texture features were attenuated, with the soft rubber coefficient being mostly characterized by rigid body contact forces.

Despite good results perceived during informal evaluation of our system, the mapping of the stress-strain relationship of a material to its compliance coefficient is something that requires investigation, and future work will require that the physical relevance of the compliance coefficient be explored more carefully. This may require "tweaking" of the compliance coefficient equation from Section 5.3. However, we can conclude that our novel technique for compliance estimation generates a good heuristic regarding the stiffness of a material.

6.5 Haptic Rendering

To test our textures, we implemented a C++ hapto-visual application on a 2.4GHz Pentium PC with an attached Phantom^{TM}Omni, a device capable of 6 DOF input and 3 DOF force display. The test environment generated by our application allows users to load textured 3D objects into a virtual workspace. Users can interact the 3D objects using a virtual proxy, which is a "single point of contact" interface metaphor for the haptic device.

We model our textures using a the Fourier series reconstruction of the surface

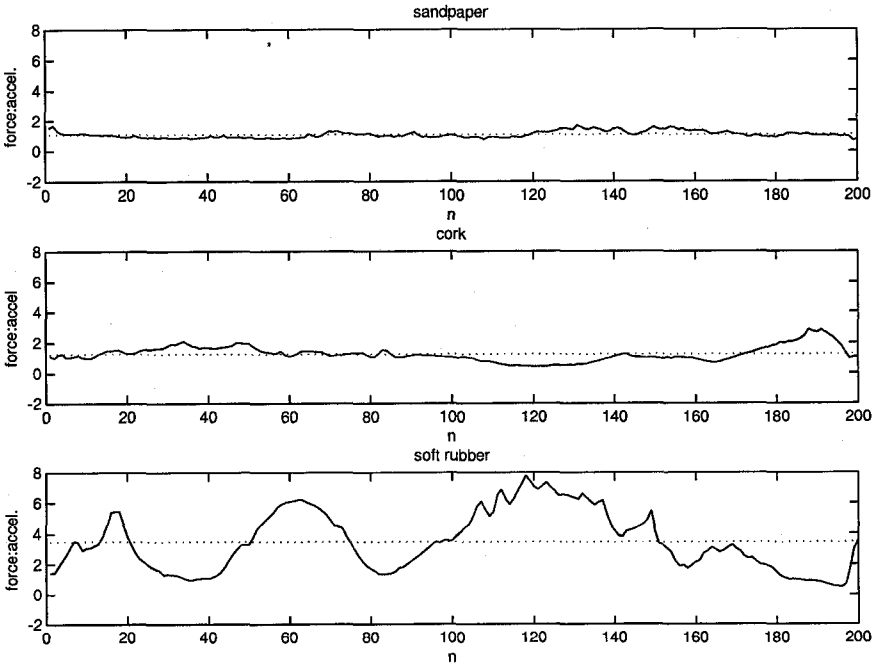


Figure 34: Force-acceleration ratios for profiles obtained by scanning various material: sandpaper, soft rubber, and cork, with mean ratio value indicated by a dashed line.

profiles generated by scans. Coefficients of the sinusoidal basis functions are obtained by sampling the discrete Fourier transform (DFT) of each surface profile. For objects such as the cable tie, sandpaper, clay pot, and protoboard, we use a sample length of $L = 10mm$ and extract $N = 128$ coefficients from the DFT of the surface profile, resulting in a texture function

$$g(x) = \sum_{j=1}^{N} a_j \cos(j\omega x) + b_j \sin(j\omega x),$$

where x is a spatial coordinate, ω is the fundamental frequency of the texture sample calculated as

$$\omega = 2\pi \frac{L}{N},$$

and a and b are the real and imaginary DFT coefficients, respectively. For 1D textures, a and b are 1-by-N vectors; for 2D textures, they are N-by-N matrices which

are built by cross correlating their 1D counterparts. Consequently, this is the same model used by Wall and Harwin [34] to reconstruct their surface profiles. Figure 35 shows a surface profile generated by our system and the corresponding surface profile generated by Fourier series reconstruction. This model works well with the surface profiles generated by our system since they are uniformly sampled, spatially, which allows the DFT operation to be performed without any re-sampling.

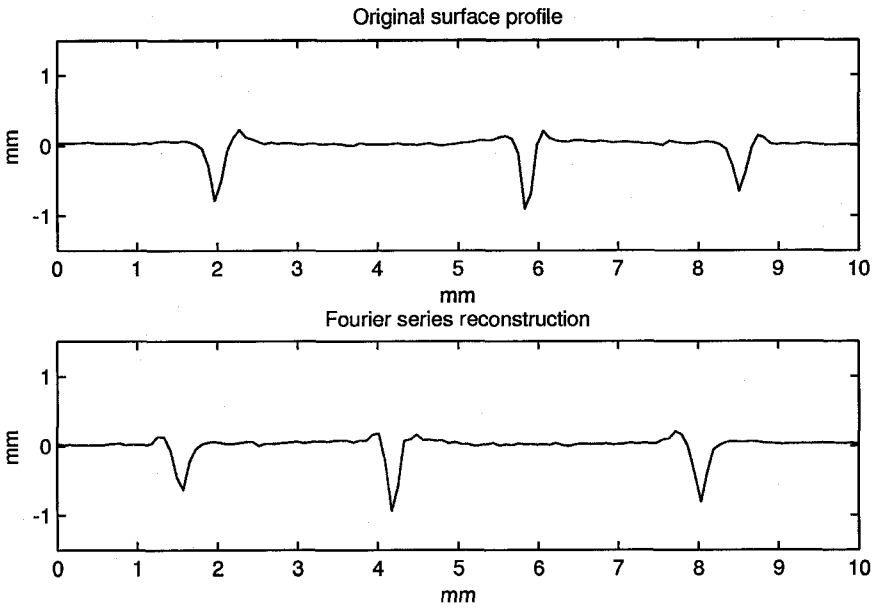


Figure 35: Surface profile of a protoboard from scanning (top) and the surface profile synthesized using Fourier series model (bottom).

We adopt a force model similar to that used by Siira and Pai [45], whereby the force vector, $\vec{f}$ due to contact and texture interaction is calculated as:

$$\vec{f} = \vec{f_c} + \vec{f_t} + \vec{f_f}. \tag{3}$$

Forces due to rigid body constraints, f_c, and texture, f_t are rendered using a penalty-based method, whereby spring forces are generated proportional to the penetration depth of the proxy into the surface, or texture, of a 3D object (as similarly described in Section 2.3). Components f_c and f_t are normal forces, and $\vec{f_f}$ is a lateral

frictional force which is proportional to $f_c + f_t$. We decompose (3) into its individual force components, calculated by:

$$\vec{f_c} = -k_s \, \Delta x \vec{n}, \tag{4}$$

$$\vec{f_t} = -k_c \, k_s \, \Delta h \vec{n}, \tag{5}$$

$$\vec{f_f} = u_s \, |\vec{f_c} + \vec{f_t}| \vec{t} \tag{6}$$

where Δx is the penetration depth of the proxy into the planar surface, Δh is the penetration depth into the texture profile, and u_s is a preselected frictional coefficient. $\vec{n}$ and $\vec{t}$ are the normal and tangential vectors, respectively, to a point on the surface of the 3D object where the haptic proxy makes contact. A visual depiction of the force rendering parameters for (4),(5), and (6) is given in Figure 36.

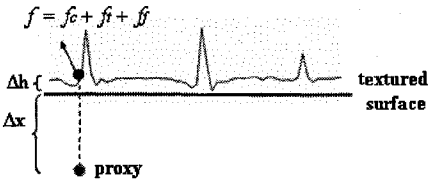


Figure 36: Haptic rendering force model. The texture surface profile is shown superimposed onto a planar surface. Contact, texture, and frictional forces are displayed at the haptic interface.

Figure 37 shows a hapto-visual application which is displaying a textured 3D dinosaur. Sections of the mesh which have been textured are highlighted and, when the user brings the proxy in contact with these areas of the mesh, the texture is displayed according to our haptic rendering algorithm. We use the Fourier model discussed above to calculate the height of a texture on the surface at the collision point of the haptic proxy and a textured polygon.

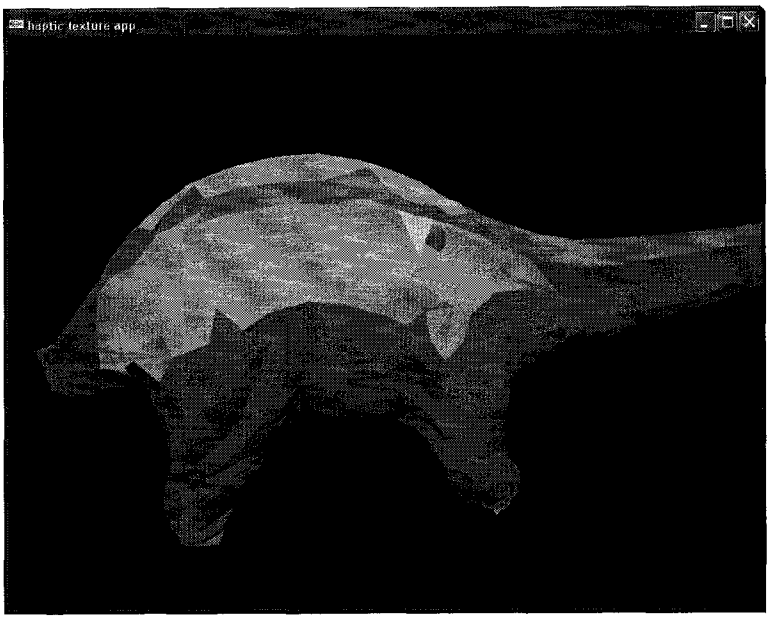


Figure 37: Rendering a textured 3D model. Areas of the mesh are textured using scan registration, and are highlighted.

We calculate the texture coordinates at the collision point by finding the barycentric coordinates for the location of the haptic proxy within the boundaries of the colliding polygon. The exact texture coordinate value at the haptic proxy is calculated as:

$$(x,y)_{proxy} = a(x,y)_{v_1} + b(x,y)_{v_2} + c(x,y)_{v_3}$$

where (a,b,c) are the barycentric coordinates of the haptic proxy determined by collision with the polygon, $(x,y)_{proxy}$ is the texture coordinate at the haptic proxy, and $(x,y)_{v_{1,2,3}}$ are the texture coordinate values assigned to each vertex of the polygon (e.g, as in Figure 33). Given that a collision only occurs inside the boundaries of a polygon, we know that $a,b,c \in [0,1]$ and $a+b+c=1$, thus giving a texture coordinate pair within a range determined by the texture coordinate values at the vertices.

An extra step in our collision detection algorithm determines if the proxy is touching only the surface profile of the texture, and if so will only render texture forces. Otherwise, if the proxy has penetrated far enough into the object, both rigid-body

and texture forces are rendered.

The rendering application is able to load and display any polygonal mesh in the augmented Wavefront file format. Textures are loaded separately, by specifying a file which contains the Fourier coefficients of the sampled surface profile, fundamental spatial frequency, and compliance estimate. This allowed us to test a variety of artistically generated and scanned 3D meshes, and to swap the textures independent of the mesh. If no texture data is available, or if untextured areas of a mesh are being rendered, only contact forces due to rigid-body constraints are rendered.

Figure 38 shows a screen shot of another haptic application which renders texture using image-based height maps, which we have also generated using surface profiles obtained by scanning. The rendering algorithm used by this application is similar to that of Basdogan et al. [46], which we discussed in Chapter 2. Each pixel of the images represents a height (or depth) value at a specific location on the surface. In the example shown, the height data is represented using 256x256 grayscale images. The surface profile functions of the cable tie, protoboard, sandpaper, and clay pot were sampled at 256 points over the range of $[0mm, 10mm]$ and a spatial cross-correlation method was used to generate a 2D profile. The intensity value of each pixel determines a height in the range of 0.0 to $2.5mm$, giving a resolution of about $0.01mm$ (e.g., $\frac{2.5mm}{256} \approx 0.01mm$).

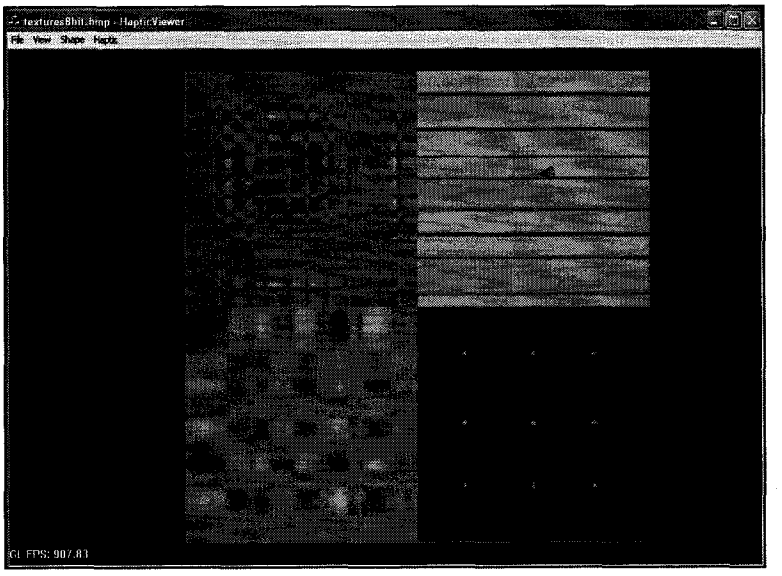


Figure 38: The haptic texture viewer. The application displays textures from image-based depth maps. Shown height maps (from upper-left, clockwise): clay pot, cable tie, protoboard, and sandpaper.

The textures generated by our system may also be used with other rendering models. For example, the parameters for the stochastic texturing model proposed by Siira and Pai [45] may be extracted from our surface profiles by statistical analysis. The stochastic texturing parameters may be chosen as mean $\mu = 0$ and variance σ^2 equal to the variation in height of the texture function. For patterned textures (e.g., the cable tie), this model is poorly suited, but for stochastic textures (e.g., the sandpaper) this model should reproduce a perceptually similar surface.

6.6 Summary

The results presented in this chapter show that our system is a viable solution for creating textures for use in 3D hapto-visual application. The textures, when rendered, exhibited the expected perceptual characteristics: the sandpaper was rough and granular; the clay pot bumpy; and the divots and peaks of the cable tie and protoboard could be felt as the proxy scanned across a textured surface. Modifying the

value of k_c in the rendering equation also made the textures "feel" as if the material had been changed, and salient features were less prominent when k_c was small.

There are accuracy and precision issues in our system which also need to be addressed. Even though the textures were perceptually similar to the real thing, quantitative analysis of features revealed that spatial warping and scaling was present in the constructed surface profiles. We suspect that efforts to increase the accuracy of the Kalman filter will alleviate some of these errors.

For interested readers, the details of our haptic rendering and pseudo-code are presented in Appendix B.

Chapter 7

Conclusions

7.1 Executive Summary

In this thesis, we have presented an interactive modeling system which uses a low-cost tactile probe and a visual tracker to generate realistic surface textures and compliance estimates. We have outlined the steps for generating haptic textures from raw sensor data and addressed several of the problems which arose from an interactive scanning system such as ours (e.g., sensor noise, merging sensor data sets, variability due to a human operator).

Our technique may be integrated as part of the 3D scanning pipeline, which we demonstrated by merging some of our textures with an existing 3D model (e.g., a polygonal mesh). Also, we have shown that both stochastic and patterned textures may be measured using our system and utilized by force models for haptic rendering. The force models used in our implementation incorporate not only the spatial components synthesized by our system, but the compliance estimates too. We tested our system using a simple hapto-visual application, capable of rendering haptic textures for arbitrary 3D objects.

Some of the drawbacks of our system, such as spatial sampling resolution, are related to sensor hardware. This could be improved by revising our hardware implementation based on some of the lessons we learned during our experiments, as we will discuss in the next section. Also, by improving the software models used by our

system (e.g., "tweaking" the Kalman filter), we should be able to increase the overall quality of synthesized textures at no additional cost to our system. While there is definitely room for improvement, we believe our system provides a solid foundation for the extraction of physical properties by a mobile and interactive scanning system. By the results presented in Chapter 6, we deem that our system is a good option for producing haptic textures which are perceptually identifiable and which represent the physical characteristics of a scanned 3D object.

7.2 Future Work

We would like to improve our scanning system while maintaining its design goals of a cost effectiveness and mobility. With this mindset, the wireless communications between the touch probe and the host PC can be improved with a Bluetooth^{TM}module, which are low-cost radio devices with a transmission range of up to $100m$. This would add error detection to the communications channel, as well as increase the bit rate of communications from 115.2 to 768 kbps. This would improve the sampling rate of the touch probe and the resolution of our textures. A Bluetooth^{TM}communications module would also have the added benefit of making the probe more compatible with newer host platforms, since such modules are commonplace in modern day laptops and PCs.

Tracker targets could be modified to better fit the probe shape and improve handling of the probe. Currently, the probe may be comfortably held by the operator but there is room for improvement. For example, the markers may be arranged such that the probe can be gripped over its entire circumference. Also, by placing markers over 360 deg, the probe may be tracked from all viewpoints. This would remove the constraint of having the probe marker facing the camera and increase the robustness of our system.

The performance of sensor measurement could be increased by an upgrade to the sensor hardware. As mentioned in Chapter 3, the accelerometer and force sensors have a range of $[-2g, +2g]$ and $[0, 4.9N]$, respectively. Low-cost sensors with a much larger range are available commercially. For example, the Analog Devices ADXL210E sensor

is a dual-axis accelerometer with a range of $[-10g, +10g]$ and is currently available for under $20 CDN. A replaceable tip would also improve scanning of objects, since the current tip can sometimes penetrate or cut the surface. This would allow a direct analysis of the affect of probe tip on the resulting texture, such as the spatial power spectral density. A tri-axial force sensor could also be used to increase the accuracy of the force scaling step and compliance estimation. Currently, these sensors are expensive and thus are not consistent with our design goal of a low-cost system.

The operator must rely on the real-time force and acceleration profile, displayed by the scanning software, to detect if pertinent features were scanned by the probe. A planned improvement to the scanning software will be to integrate the digital filtering and texture synthesis algorithms from Chapter 4 and Chapter 5 as part of the application, such that surface profiles are immediately generated and displayed to the operator. This would further exploit the "human-in-the-loop" advantage of our system and would allow interactive tuning of the process parameters (e.g., spatial cut-off frequency, force scaling, velocity profile threshold).

As mentioned in Section 6.2, the reconstruction of the scanning trajectory by the Kalman filter affects the resulting surface profile, mainly due to residuals in the distance and normal vector calculations. We attributed the sensitivity in path reconstruction due to the static process and observation model which is used to remove noise from the tracker data. We plan to improve the filter by using a dynamic model which incorporates state information as part of its estimate (e.g., tip velocity, angular velocity, distance from camera lens).

The compliance coefficients k_c determined by are system are a heuristic estimate which are intended to be used as part of the unified haptic rendering equation (3). However, the mapping between the k_c and the actual stress-strain curve of a material is not clear. Future work in this area will be to compare the coefficients determined using our system with "ground truth" data (e.g., obtained using a universal testing machine). This will shed more light on how k_c may be used as part of our haptic rendering algorithm.

Appendix A

Miscellaneous Algorithms

A.1 SLERP

Spherical linear interpolation (SLERP) is an interpolation scheme introduced by Ken Shoemake for use in computer animation. A 3D rotation is represented by a quaternion vector, $\vec{q}$, with components w,x,y,z giving a location on the surface of a 4D hyper-sphere.

Traditional interpolation schemes, such as linear interpolation (LERP), may be used to interpolate between two quaternions by:

$$lerp(\vec{q}_0, \vec{q}_1, t) = (1-t)\vec{q}_0 + t\vec{q}_1, t \in [0,1].$$

This method is similar to drawing a straight line between two points on the 4D hyper-sphere and interpolating along this line. However, this introduces scaling and warping errors. Since unit quaternions are used to represent a 3D rotation, a normalization step must be performed after LERP is performed. The resulting quaternions also appear to be non-uniformly spaced over the range of the interpolation parameter, t.

A better way to interpolate between two quaternions is to move along the surface of the hyper-sphere. This is achieved by using a sinusoid function, such that intermediary quaternion values are determined by:

$$slerp(\vec{q}_0, \vec{q}_1, t) = sin\left(\frac{(1-t)\pi}{2}\right)\vec{q}_0 + sin\left(\frac{t\pi}{2}\right)\vec{q}_1.$$

Figure 39 illustrates the differences between these two methods.

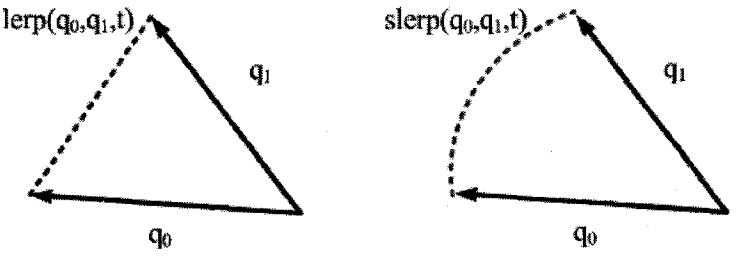


Figure 39: A comparison of LERP and SLERP for interpolation between two quaternions.

A.2 Iterative Closest Point

The iterative closest point (ICP) algorithm is used to align two or more sets of 3D geometry, given an initial estimate of their relative transformation. The algorithm works by finding corresponding points in each set, computing a transform between these points, updating each set with the transform, and iterating until the point sets are sufficiently aligned. Pseudo-code for the ICP algorithm is shown in Figure 40.

while $error > minError$ **do**
  associate points by nearest neighbour;
  estimate transform using minimum cost function;
  update points using the estimated transform;
  $error = |points_0 - points_1|^2$;
end while

Figure 40: Pseudo-code for the ICP algorithm.

For many ICP implementations, the cost function is a distance metric (e.g., Euclidean distance) between each pair of corresponding points. Corresponding points may be determined by assuming the closest points in each set are pairs. A good review of the ICP algorithm and its variants may be found in work by Rusinkiewicz et al. [67].

A.3 Kalman Filtering

Kalman filters are a type of recursive filter which is useful for estimating the true state of a linear dynamical system which is perturbed by Gaussian noise. Given a set of observed data $z_1, z_2, ..., z_k$, the filter finds for each $k \geq 1$ the optimal estimate of the true state, x_k. For $i = k$, this problem is called *filtering*, for $i > k$ it is called *prediction*, and for $1 \leq i < k$ it is called *smoothing*.

The Kalman filter algorithm has two stages: *predict* and *update*. During the predict stage, the filter obtains a predicted state for the system using a model of the dynamics which describes the system state transition from interval $k-1$ to k. In the update stage, the filter adjusts its estimate of the true system state by considering real observations of the system. At any interval of the Kalman filter's algorithm, it begins by estimating the current state of the system:

$$x_k = F_k x_{k-1} + w_k$$
$$w_k \sim N(0, Q_k)$$

Here, x_k is the predicted system state based on the previous state x_{k-1} which is evolved using the state transition model, F_k. The addition of a noise vector, w_k, to the predicted state indicates that we are certain about it's value only to within a margin of error. The noise is modeled as a Gaussian distribution with $\mu = 0$ and variance determined by the matrix Q_k. Next, the Kalman filter improves its estimation of the system state using a measurement:

$$z_k = H_k x_k + v_k$$
$$v_k \sim N(0, R_k)$$

The observation vector, z_k, is modeled as the value of the predicted system state, x_k plus some measurement noise vector, v_k. Similar to the the process noise, measurement noise is also modeled as a Gaussian distribution, but with variance determined by the matrix R_k. The predicted state is related to the observation vector by the observation model matrix, H_k. Note that the measurement vector and state vector do not necessarily contain the same parameters.

We can expand the predict-and-update cycle of the Kalman filter into more precise steps:

1. Predict the next state using F:

$$\hat{x}_k = F_k x_{k-1}$$

2. Project the error covariance ahead to the current interval:

$$\hat{P}_k = F_k Pk - 1F_k^T + Q_k$$

3. Compute the Kalman gain using H_k:

$$S_k = H_k \hat{P}_k H_k^T + R_k$$
$$K_k = \hat{P}_k H_k^T S_k^{-1}$$

4. Update using measurement z_k:

$$y_k = z_k - H_k \hat{x}_k$$
$$x_k = \hat{x}_k + K_k y_k$$

5. Update the error covariance matrix using the gain and H_k matrix:

$$P_k = (I - K_k H_k)\hat{P}_k$$

6. Repeat for $k + 1$

For a good analysis on the use of Kalman filters in visual tracking, we recommend reviewing the report on this topic by Cuevas et al. [68].

Appendix B

Pseudo-code

This appendix contains the pseudo-code for a selected set of processes of our haptic texturing system Section B.1 contains the algorithm for generating the surface profile of an object from sensor data acquired by the scanning software. Section B.2 outlines the algorithm for scan path registration, a process which encompasses the ICP algorithm and assignment of haptic texture coordinates. Finally, Section B.3 contains the pseudo-code for the haptic rendering algorithm used by the hapto-visual software application from Chapter 6

B.1 Texture Synthesis

Below is the pseudo-code for generating a surface profile from raw sensor data:

$t \Leftarrow$ time data
$P \Leftarrow$ position data
$Q \Leftarrow$ orientation data
$F \Leftarrow$ force profile
$A \Leftarrow$ acceleration profile
$B, C \Leftarrow$ linear calibration model
use Kalman filter to remove noise from P and Q
${}^{p}a_{gravity} = Q < 0, -9.81, 0 > \overline{Q}$

${}^{s}a_{gravity} = B^{p}g_{gravity} + C$
$A \Leftarrow A - {}^{s}a_{gravity}$
$X = Q < 1,0,0 > \overline{Q}$
$Y = Q < 0,1,0 > \overline{Q}$
$Z = Q < 0,0,1 > \overline{Q}$
$N \Leftarrow$ compute unit normal vectors of $path$
$w_X = X \cdot N$
$w_Y = Y \cdot N$
$w_Z = Z \cdot N$
$A \Leftarrow B^{-1}(A - C)$ converts to probe coordinate frame
$A_{tip} = w_X A_X + w_Y A_Y + w_Z A_Z$
scale A_{tip} according to force profile F
$h \Leftarrow$ Verlet integration of A_{tip} over t
$d \Leftarrow$ distance profile of P
generate velocity profile V from d and t
for all v in V such that $|v| < v_{min}$ **do**
  $i \Leftarrow$ index of v
  remove i^{th} element of h and d
end for
use Butterworth high-pass filter to remove drift from h

B.2 Scan Path Registration

Below is the pseudo-code for the registration of a scanning trajectory with a 3D polygonal mesh and texture coordinate assignment based on path-mesh alignment:

$S \Leftarrow$ scanning trajectories
$P \Leftarrow$ polygons from mesh
for all s in S **do**
  manual alignment of s with P

$M \Leftarrow$ closest points in P to s
while $|s - P| > error_{min}$ **do**
project s onto P using minimum distance criteria
$H \Leftarrow$ optimal relative transform of s to P
update s using H
$M \Leftarrow$ closest points in P to s
end while
$D \Leftarrow$ distance profile of s
extend D and M by a small amount
for all p in P such that p contains a point in M **do**
$V \Leftarrow$ vertex of p
for all v in V **do**
$c \Leftarrow$ closest point in M to v
$y \Leftarrow$ signed distance from c to v
$x \Leftarrow$ distance value from M corresponding to c
assign vertex v of polygon p the texture coordinate (x, y)
end forgain using H_k:
end for
end for

B.3 Haptic Rendering

Below is the pseudo-code for the haptic rendering algorithm used by our hapto-visual application:

$h = 0$
$d = 0$
if collision with texture **then**
$p \Leftarrow$ find colliding polygon
$i \Leftarrow$ texture index of p

if $i > -1$ **then**
  $(a, b, c) \Leftarrow$ barycentric coordinates of the collision point
  $(x, y) \Leftarrow$ texture coordinates computer using (a, b, c) and p
  $h \Leftarrow$ height of i^{th} texture function at (x, y)
end if
$nor \Leftarrow$ unit normal vector of p
$tan \Leftarrow$ unit tangential vector of proxy velocity
$f_t = K_c \times K_s \times h \times nor$
if collision with mesh **then**
  $d \Leftarrow$ penetration depth of proxy
  $f_c = K_s \times d \times nor$
else
  $f_c = 0$
end if
$f_u = U_s \times (f_c + f_t) \times tan$
$f = f_c + f_t + f_u$
render force f
end if

Bibliography

[1] V. Hayward, O.R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre, "Haptic interfaces and devices," *Sensor Review*, pp. 16–29, February 2004.

[2] S. Srikanth R.K. Mishra, "GENIE – an haptic interface for simulation of laparoscopic surgery," *Proc. of International Conference on Intelligent Robots and Systems*, pp. 714–719, 2000.

[3] Immersion Corporation, "Surgical simulator, the laparoscopy VR virtual reality system," http://www.immersion.com/medical/products/laparoscopy/, July 2007.

[4] R.D. Howe C.R. Wagner, N. Stylopoulos, "The role of force feedback in surgery: analysis of blunt dissection," *Proc. of 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 68–74, 2002.

[5] V. Gourishankar, G. Srimathveeravalli, and T. Kesavadas, "Hapstick: A high fidelity haptic simulation for billiards," *Proc. of 2nd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 494–500, 2007.

[6] D. Morris, N. Joshi, and K. Salisbury, "Haptic battle pong: a networked haptic game," *Proc. of Future Play 2006*, 2006.

[7] S. Andrews, J. Mora, J. Lang, and W.S. Lee, "Hapticast: a physically-based game with haptic feedback," *Proc. of FuturePlay 2006*, 2006.

[8] D. Pai, K. van den Doel, D. James, J. Lang, J. Lloyd, J. Richmond, and S. Yau, "Scanning physical interaction behavior of 3D objects," in *Computer Graphics, Annual Conference Series*, Los Angeles, USA, Aug 2001, ACM SIGGRAPH, pp. 87–96.

[9] A. Okamura, J. Dennerlein, and M. Cutkosky, "Reality-based models for vibration feedback in virtual environments," *IEEE/ASME Transactions on Mechatronics*, vol. 6, no. 3, pp. 245–252, 2001.

[10] H.Y Yao, V. Hayward, and R.E. Ellis, "A tactile enhancement instrument for minimally invasive surgery," *Computer Aided Surgery*, pp. 233–239, May 2005.

[11] F. Bernardini and H. Rushmeier, "The 3d model acquisition pipeline," *Computer Graphics Forum*, vol. 21, no. 2, pp. 149–172, 2002.

[12] S. Andrews and J. Lang, "Interactive scanning of haptic textures and surface compliance," *Proc. of 6th Intl. Conference on 3D Digital Imaging and Modeling (3DIM)*, 2007.

[13] M. Hollins, R. Faldowski, S. Rao, and F. Young, "Perceptual dimensions of tactile surface texture: a multidimensional scaling analysis," *Perception and Psychophysics*, vol. 54, no. 6, pp. 697–705, 1993.

[14] W.M. Bergmann Tiest and A.M.L. Kappers, "Analysis of haptic perception of materials by multidimensional scaling and physical measurements of roughness and compressibility," *Acta Psychologica*, vol. 121, no. 1, pp. 1–20, 2006.

[15] R.L. Klatzky and S.J. Lederman, "Tactile roughness perception with a rigid link interposed between skin and surface," *Perception & Psychophysics*, vol. 61, no. 4, pp. 591–607, 1999.

[16] S.J. Lederman, R.L. Klatzky, C.L. Hamilton, and G.I. Ramsay, "Perceiving roughness via a rigid probe: Psychophysical effects of exploration speed and mode of touch," *Haptic e-journal*, vol. 1, no. 1, 1999.

[17] V. Levesque and V. Hayward, "Experimental evidence of lateral skin strain during tactile exploration," in *Proc. of Eurohaptics 2003*, July 2003.

[18] H. Pongrac, "Vibrotactile perception: Differential effect of frequency, amplitude, and acceleration," in *Proc. of IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2006, pp. 72–77.

[19] O. Franzen and J. Nordmark, "Vibrotactile frequency discrimination," *Perception & Psychophysics*, vol. 17, no. 5, pp. 480–484, 1975.

[20] K. Johnson, "Neural basis of haptic perception," in *Stevens' handbook of experimental psychology, Vol. 1: Sensation and perception.* 2002, pp. 537–583, John Wiley.

[21] M.A. Costa and M.R. Cutkosky, "Roughness perception of haptically displayed fractal surfaces," *Proc. of the ASME Dynamic Systems and Control Division*, vol. 69, no. 2, pp. 1073–1079, 2000.

[22] G. Campion and V. Hayward, "Fundamental limits in the rendering of virtual haptic textures," in *Proc. of 1st Joint Eurohaptics Conference and Symp. on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, 2005.

[23] C. Ramstein and V. Hayward, "The pantograph: A large workspace haptic device for a multi-modal human-computer interaction," in *Conference on Human Factors in Computing Systems ACM/SIGCHI*, 1994.

[24] SensAble Technologies Inc., "Phantom omni haptic device," http://www.sensable.com/haptic-phantom-omni.htm, July 2007.

[25] D. Pai, J. Lang, J. Lloyd, and R. Woodham, "ACME: a telerobotic active measurement facility," *Experimental Robotics VI*, vol. 250, pp. 391–400, 2000.

[26] N. Galoppo, S. Tekin, M.A. Otaduy, M. Gross, and M.C. Lin, "Interactive haptic rendering of high-resolution deformable objects," *Proc. of Human-Computer Interaction (HCI) International*, 2007.

[27] Immersion Corporation, "Microscribe G2 3D digitizers," http://www.immersion.com/digitizer/products/microscribe_g2.php, July 2007.

[28] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," in *ACM Trans. on Graphics*, San Antonio, USA, Jul 2002, ACM SIGGRAPH, 21(3), pp. 438–446.

[29] V. Popescu, E. Sacks, and G. Bahmutov, "The modelcamera: a hand-held device for interactive modeling," in *Proc. of the Fourth International Conference on 3D Imaging and Modeling (3DIM)*, 2003.

[30] J.D. Deschênes, P. Hébert, P. Lambert, J.N. Ouellet, and D. Tubic, "Multiresolution interactive modeling with efficient visualization," in *3DIM05*, Ottawa, Canada, June 2005, IEEE, pp. 39–46.

[31] J.-Y. Bouguet and P. Perona, "3D Photography on your desk," in *International Conference on Computer Vision*, Bombay, India, Jan 1998, IEEE, pp. 43–50.

[32] Konica Minolta Holdings, "Non-contact 3d digitizer vivid 910/vi-910," http://konicaminolta.com/products/instruments/vivid/vivid910.html, June 2007.

[33] P. Kry and D. Pai, "Interaction capture and synthesis," in *ACM Trans. on Graphics*, Boston, USA, July 2006, ACM SIGGRAPH, 25(3), pp. 872–880.

[34] S. Wall and W. Harwin, "Modelling of surface identifying characteristics using fourier series," in *Proc. ASME Dynamic Systems and Control Division (Symposium on Haptic Interfaces for Virtual Environments and Teleoperators)*, 1999, pp. 65–71.

[35] D. Pai and P. Rizun, "The WHaT: A wireless haptic texture sensor," in *Proc. 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Los Angeles, USA, 2003.

[36] R.S. Fearing, "Tactile sensing mechanisms," *International Journal of Robotics Research*, vol. 9, no. 3, pp. 3–23, 1990.

[37] V. Maheshwari and R.F. Saraf, "High-resolution thin-film device to sense texture by touch," *Science*, vol. 312, pp. 1501–1504, June 2006.

[38] J. Pasquero and V. Hayward, "Stress: A practical tactile display system with one millimeter spatial resolution and 700 hz refresh rate," in *Proc. of Eurohaptics 2003*, July 2003.

[39] S. Choi and H.Z. Tan, "An analysis of perceptual instability during haptic texture rendering," in *Int. Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. ASME, March 2002, pp. 129–136.

[40] T. Aysal and K. Barner, "Stochastic and deterministic models for haptic pseudo-textures," in *Int. Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Arlington, VA, USA, March 2006, IEEE, p. 71.

[41] M. Srinivasan and C. Basdogan, "Haptics in virtual environments: Taxonomy, research status, and challenges," *Computer Graphics*, vol. 21, no. 4, pp. 393–404, 1997.

[42] M. Minsky, *Computational haptics: the Sandpaper system for synthesizing texture for a force-feedback display*, Ph.D. thesis, MIT, 1995.

[43] K. Perlin, "An image synthesizer," *Computer Graphics*, vol. 19, no. 3, pp. 287–296, July 1985.

[44] J. Fritz and K. Barner, "Stochastic models for haptic texture," in *SPIE Intl. Symp. on Intelligent Systems and Adv. Manufacturing – Telemanipulator and Telepresence Technologies III*, Boston, MA., 1996.

[45] J. Siira and D. Pai, "Haptic rendering: A stochastic approach," in *Proc. 1996 IEEE Intl. Conference Robotics and Automation*, 1996, pp. 557–562.

[46] C. Basdogan, C. Ho, and M.A. Srnivasan, "A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environment," in *Proc. Of the ASME Dynamic Systems and Control Division*, 1997, pp. 77–84.

[47] M. Minsky, O. Ming, O. Steele, F. Brooks, and M. Behensky, "Feeling and seeing: issues in force display," in *Proc. Symposium on Interactive 3D Graphics.* 1990, pp. 235–241, ACM Press.

[48] N. Max and B. Becker, "Bump shading for volume textures," *IEEE Computer Graphics and App.*, pp. 18–20, July 1994.

[49] M.A. Otaduy and M.C. Lin, "A perceptually-inspired force model for haptic texture rendering," in *Proc. of Symposium on Applied Perception in Graphics and Visualization*, 2004, pp. 123–126.

[50] M.A. Otaduy, N. Jain, A. Sud, and M.C. Lin, "Haptic display of interaction between textured models," in *Proc. of IEEE Visualization Conference*, 2004, pp. 297–304.

[51] J. Shopf and M. Olano, "Procedural haptic texture," in *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, 2006, pp. 179–186.

[52] R. Cortes and S. Raghavachary, *The RenderMan Shading Language Guide*, Course Technology PTR, March 2007.

[53] B. Unger, "Texture perception with realistic probes in virtual haptic environments," *Carnegie Mellon University thesis proposal*, 2005.

[54] M. Fiala, "ARTag revision 1, a fiducial marker system using digital techniques," in *NRC/ERB1117 Technical Report. National Research Council of Canada*, 2004.

[55] E. Ruffaldi, D. Morris, T. Edmunds, F. Barbagli, and D. Pai, "Standardized evaluation of haptic rendering systems," *Proc. of the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'06)*, 2006.

[56] J. Bouguet, "Camera calibration toolkit for MATLAB," http://www.vision.caltech.edu/bouguetj/calib_doc/.

[57] K. Shoemake, "Animating rotation with quaternion curves," in *Computer Graphics, Annual Conference Series*, San Francisco, USA, Jul 1985, ACM SIGGRAPH, pp. 245–254.

[58] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[59] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[60] J.S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[61] L. Verlet, "Computer experiments on classical fluids," *Phys. Rev.*, vol. 159, no. 1, pp. 98, July 1967.

[62] S. Butterworth, "On the theory of filter amplifiers," *Wireless Engineer*, vol. 7, pp. 536–541, 1930.

[63] S. Wall and W. Harwin, "Mechatronic design of a high frequency probe for haptic interaction," in *Proc. 6th Intl. Conference on Mechatronics and Machine Vision in Practice*, July 1999.

[64] Geomagic, "Geomagic studio," http://www.geomagic.com/en/products/studio/, June 2007.

[65] N. McKay P. Besl, "A method for registration of 3d shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, 1992.

[66] Berthold K. Horn, *Robot Vision*, McGraw-Hill Higher Education, 1986.

[67] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3rd International Conference on 3-D Digital Imaging and Modeling*, Québec City, Canada, 2001, IEEE.

[68] E. Cuevas, D. Zaldivar, and R. Rojas, "Kalman filter for vision tracking," Tech. Rep., Freie Universitat Berlin, 2005.