Refined Methods for Creating Realistic Haptic Virtual Textures from Tool-Mediated Contact Acceleration Data

Heather Culbertson*, Joseph M. Romano*, Pablo Castillo*, Max Mintz§ and Katherine J. Kuchenbecker*

*Haptics Group, GRASP Laboratory
Mechanical Engineering and Applied Mechanics
University of Pennsylvania, USA
hculb, jrom, pabloc, kuchenbe@seas.upenn.edu

§GRASP Laboratory
Computer and Information Science
University of Pennsylvania, USA
mintz@cis.upenn.edu

ABSTRACT

Dragging a tool across a textured object creates rich high-frequency vibrations that distinctly convey the physical interaction between the tool tip and the object surface. Varying one's scanning speed and normal force alters these vibrations, but it does not change the perceived identity of the tool or the surface. Previous research developed a promising data-driven approach to embedding this natural complexity in a haptic virtual environment: the approach centers on recording and modeling the tool contact accelerations that occur during real texture interactions at a limited set of force-speed combinations. This paper aims to optimize these prior methods of texture modeling and rendering to improve system performance and enable potentially higher levels of haptic realism. The key elements of our approach are drawn from time series analysis, speech processing, and discrete-time control. We represent each recorded texture vibration with a low-order auto-regressive moving-average (ARMA) model, and we optimize this set of models for a specific tool-surface pairing (plastic stylus and textured ABS plastic) using metrics that depend on spectral match, final prediction error, and model order. For rendering, we stably resample the texture models at the desired output rate, and we derive a new texture model at each time step using bilinear interpolation on the line spectral frequencies of the resampled models adjacent to the user's current force and speed. These refined processes enable our TexturePad system to generate a stable and spectrally accurate vibration waveform in real time, moving us closer to the goal of virtual textures that are indistinguishable from their real counterparts.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Haptic I/O

1 INTRODUCTION

Imagine picking up a screwdriver and gently dragging its tip across a swatch of finely woven wool, a convoluted seashell, a wicker basket, and a burnished bronze sculpture. You can feel both striking and subtle variations in texture through a tool, even though your skin is not directly touching the surface. High-frequency vibrations are a rich and valuable source of information during these tool-mediated contacts with real objects [10]. The human sense of touch excels at sensing and interpreting these vibrations to gather information about the physical contacts taking place.

Richly textured surfaces are essential for creating an immersive and realistic experience for users in virtual environments. However, most modern haptic devices and algorithms cannot output high-fidelity reproductions of the vibrations that occur during tool-mediated texture exploration [6]. Increasing haptic realism is especially crucial for medical simulators where doctors train to learn a new skill before performing it on a human patient. Unrealistic haptic feedback is considered by many to hinder the widespread adoption of this technology into the medical curriculum [8].

Figure 1: Our texture recording and rendering apparatus. The ABS plastic sample is shown on the left. An accelerometer mounted on the stylus records the vibrations that occur as it is dragged across the real surface. A force sensor in the stylus measures the applied normal force, and sensors embedded in the Wacom tablet measure the tool's position. The virtual texture is rendered on the right. The computer generates a synthetic texture vibration in real time, and a voice-coil actuator mounted on the stylus transmits these vibrations to the user's hand.

Although many methods have been proposed for representing textures, no consensus exists concerning the best solution to this high-dimensional problem [20]. Given their success in other haptic rendering areas, such as tapping and cutting [19], measurement-based models offer a promising new approach to creating realistic virtual textures. The research presented in this paper uses the measurement-based approach of haptography to model textures felt through a tool. Haptography draws its name from a comparison to photography in the way that it allows an individual to record the feel of an interesting interaction and then reproduce the feel of that surface at a later time [12]. Our group recently made the TexturePad (Fig. 1), the first demonstration of a full system for capturing and rendering haptic textures [23]. This paper's goal is to refine and understand the methods put forward in prior work to enable more realistic haptographic textures in the future.

2 BACKGROUND

Traditional impedance-type haptic devices are capable of measuring the user's position and outputting low-frequency forces. These forces are commonly calculated using a Hooke's law relationship with the penetration depth between the device's position and the object's surface [27]. Although this approach allows for an accurate representation of the overall shape of an object, the objects tend to feel unrealistically soft and smooth. Much of the interaction is lost through the virtual model's focus on low-frequency forces and

the device's inability to output high-frequency vibrations. Without these vibrations, virtual surfaces lack the rich textural information available during tool-mediated exploration of real objects.

Many attempts have been made to improve the realism of haptic virtual interactions, but most have been unable to completely match the richness and usefulness of the haptic feedback experienced in natural interactions with the physical world. The Sandpaper system was the first project to attempt to recreate virtual textures using an impedance-type haptic device [16]. In this work, Minsky et al. modeled the surface roughness as a series of lateral springs that would repel or attract the user's hand. This system allowed for roughness perception, but it required extensive interactive tuning by the experimenters for each new desired sensation.

Recently, many researchers have diverged from using a priori model designs and are exploring the creation of data-driven haptic textures. A seemingly obvious approach is to record and play back texture acceleration signals. However, there are many drawbacks to this approach [23]. Because the power and frequency content of the vibration must change as a function of the user's force and speed, one needs a method for interpolating between signals recorded under different conditions. One could simply always play the recording closest to the user's current force and speed, but switching between recordings when force and speed change will create perceptually noticeable artifacts in the new acceleration signal. Another possible method could be to output a weighted average of recorded acceleration signals as a function of force and speed. However, this summation in the time domain causes significant constructive and destructive interference between the signals and does not preserve frequency content. Furthermore, recordings require a large amount of storage space, and any pecularities that happened to be captured would likely be perceptually noticeable when repeated in a loop. These pitfalls in directly playing recorded acceleration signals has motivated a range of work in creating data-driven texture models.

For example, Okamura et al. [18] represented patterned textures as a set of data-driven decaying sinusoids that depend on the user's speed and applied force. The vibrations used to represent the textures were superimposed on forces used to represent the stiffness of the surface for output to a force-feedback joystick. Guruswamy et al. [7] took a similar approach and created texture models based on a spatial distribution of infinite-impulse-response filters that are fit with decaying sinusoids.

Several other researchers have explored the use of data captured from real interactions to create virtual haptic texture models in order to increase realism. Pai et al. [21] modeled measured friction variations as an autoregressive process under the assumption that the roughness of a surface is isotropic and people are sensitive only to the statistical features of varying friction. Vasudevan and Manivannan [26] used a SensAble PHANToM to drag across a textured surface and modeled the resulting haptic texture from the frequency spectrum of the tooltip's vertical displacements. Although normal force was kept constant, they allowed for variations in scanning speed by interpolating between two texture datasets at different speeds.

This paper builds on the methods by which Romano and Kuchenbecker created and rendered haptic texture models [23, 24]. A major benefit of this data-driven approach is that it does not require tedious and inaccurate hand-tuning of the output model. Rather, models are generated automatically from a recorded dataset. This previous work was an initial implementation of data-driven haptic textures and resulted in haptic textures with an average realism rating of 65.4/100 when compared to real materials. In this paper, we explore this topic in further detail to provide additional modeling and rendering techniques capable of reducing the size of stored models while attempting to maintain or improve realism.

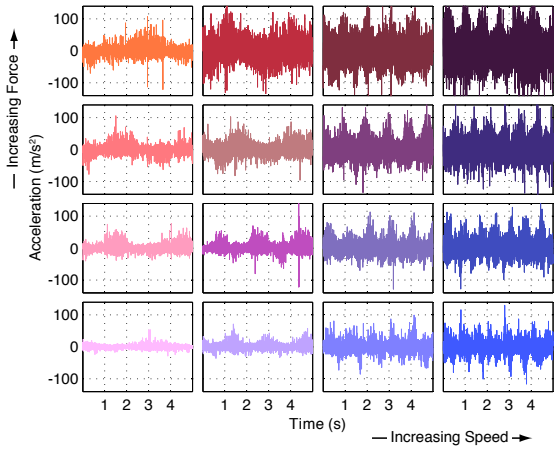

Figure 2: Sixteen time domain signals for rough ABS plastic. The speed varies in the horizontal direction from 0.05 to 0.20 m/s. The force varies in the vertical direction from 0.3 to 1.2 N.

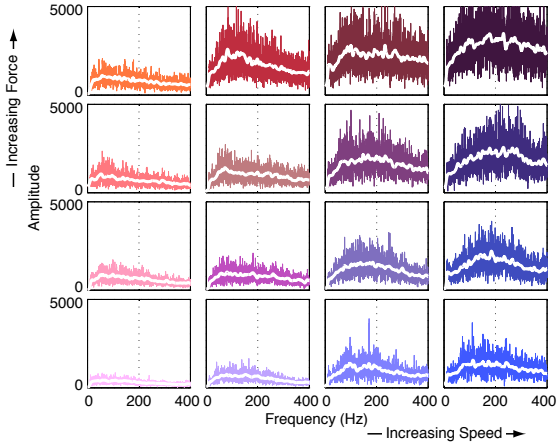

Figure 3: DFTs of the recorded accelerations shown in Fig. 2. The smoothed DFT is also shown in white to help visualize the central tendency. The power of the DFT increases with increased force and/or speed, and the spectrum shifts.

3 Texture Modeling

By critically examining the methods of Romano and Kuchenbecker, this project aimed to understand and optimize the steps needed to transform acceleration recordings into a rendered texture that is as realistic as possible. This section details our improved texture modeling process, and the following section covers the texture rendering improvements. Below, we first present the method for acquiring and processing texture acceleration data, which is largely common to this paper and [23]. Then we discuss the mathematical model we chose to represent the texture data. Finally, we describe the metrics we use to determine an appropriate model order based on error and spectral match compared to the original recordings.

3.1 Data Collection and Preprocessing

The datasets used to create our texture models were recorded using a Wacom tablet and stylus equipped with a three-axis high-bandwidth accelerometer. The apparatus for data collection is shown in Fig. 1; the experimenter places the surface on top of the tablet and uses the stylus to explore the texture. The tablet, a Cintiq 12WX interactive pen display by Wacom Co., Ltd., measures the tool's scanning speed and normal force during data collection. The x-position and y-position measurements have a resolution of 4.934 μm and 4.861 μm respectively, and the force has a resolution of 0.0013 N. These three variables are accessed through a modified

version of the Linux Wacom Project code base [2]. An Analog Devices ADXL345 accelerometer is firmly attached to the stylus. This digital accelerometer was configured into ± 78.4 m/s^2 (± 8 g) mode with a resolution of ± 0.153 m/s^2 and was polled at a rate of 800 Hz; faster rates were not possible due to the 1000 Hz limit of the Sub-20 SPI-to-USB converter used to communicate with the accelerometer. For each surface, five seconds of data (position, force, and acceleration) were recorded for each combination of four speeds (0.05, 0.10, 0.15, and 0.20 m/s) and four forces (0.3, 0.6, 0.9, and 1.2 N), yielding a total of sixteen datasets.

Data was recorded while the experimenter moved the stylus in a circle on the surface. For each five-second recording, the experimenter held the stylus vertical and kept the scanning speed and normal force approximately constant using visual indicators. Accelerations were recorded along three axes and high-pass filtered at 10 Hz to remove gravity and the effects of the circular hand movement. These accelerations were then mapped onto a single axis using the DFT321 method described in [13], which preserves the spectral and temporal properties of the three-axis signal. This conversion is motivated by the fact that humans cannot discriminate the direction of high-frequency vibrations [4]. Fig. 2 shows the recorded acceleration data for the sixteen tested combinations of speed and force on rough ABS plastic, and Fig. 3 shows the corresponding DFTs. Many other isotropic surfaces have also been characterized; this paper uses ABS plastic as a representative sample because it was the highest rated virtual texture in [23].

3.2 Model Type

To provide a more efficient and robust method of building haptic texture models from tool-surface interaction data, we applied the methods of time series analysis to the recordings of high-frequency accelerations described in the previous section.

Romano and Kuchenbecker [23, 24] used linear predictive coding (LPC) with 400 coefficients to represent the time-domain patterns in the data, which they recorded at 800 Hz and upsampled to 5000 Hz. In LPC, the system's next output is modeled as a linear combination of the last p outputs; this arrangement is also known as an auto-regressive (AR) model or an all-pole model. We expanded this approach and modeled the acceleration waveforms using auto-regressive moving-average (ARMA) models, which include both poles and zeros. The output of an ARMA model is a combination of the past outputs (AR) and a weighted moving average of the past inputs of white Gaussian disturbance noise (MA). The ARMA model structure is the difference equation:

$$A(p)\,y(t) = C(q)\,e(t) + u(t) \tag{1}$$

where $A(p)$ is the array of AR coefficients, $y(t)$ is the output at time t, $C(q)$ is the array of MA coefficients, $e(t)$ is the white-noise disturbance value at time t, and $u(t)$ is the residual at time t [15]. This difference equation corresponds to the following discrete-time transfer function:

$$H(z) = \frac{\sum_{k=0}^{q} c_k z^{-k}}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{2}$$

where p is the AR model order and q is the MA model order.

The coefficients are estimated using the Levinson-Durbin algorithm, which minimizes the final prediction error (FPE) using a least squares approach [15] with the equation:

$$\begin{aligned} \text{FPE} &= \sum_{n=1}^{N} (u_n)^2 \\ &= \sum_{n=1}^{N} \{ y_n - a[1]y_{n-1} - \cdots a[p+1]y_{n-n_a} \\ &\qquad - e_n - c[1]e_{n-1} - \cdots c[q]e_{n-n_c} \}^2 \end{aligned} \tag{3}$$

where N is the length of the dataset being modeled, u_n is the nth residual, y_n is the nth output, e_n is the nth disturbance value. The numerical procedure for estimating the model coefficients is available in Matlab using the function `armax(accel,[p,q])`.

Typically, increasing the model order will decrease the residuals and the FPE of the model. However, raising the model order too high causes overfitting. Although the model may fit the recorded data points better than a lower-order model, it will be poor at prediction, as an overfit model will tend to fit noise and other random effects that are present in the data. Therefore, it is necessary to determine the appropriate model order for each texture.

3.3 Model Order Selection

When selecting the appropriate model order, we follow the principle of parsimony, which seeks to balance adequately representing the data and minimizing the number of model parameters. This tradeoff is traditionally expressed via the Bayesian Information Criterion (BIC):

$$\text{BIC} = N\,\ln(\hat{\sigma}_a^2) + (p+q)\,\ln(N) \tag{4}$$

where N is the number of data points used to make the model and $\hat{\sigma}_a^2$ is the maximum likelihood estimate of the residual variance σ_a^2 [5]. The theory behind this cost function is that a smaller BIC will result in a more parsimonious model due to the addition of the second term $(p+q)\,\ln(N)$, which penalizes larger model orders. Fig. 4 shows a color matrix plot of the BIC computed for a wide variety of ARMA model orders for the ABS plastic dataset. The number of AR coefficients was varied from zero to fifteen (16 options), and the number of MA coefficients was also varied from zero to fifteen (16 options); a model needs at least one coefficient, so there is no model with zero AR and zero MA coefficients, leaving $(16 \times 16) - 1 = 255$ possible model orders. Each datapoint that appears in the plot is the BIC for that model order averaged across the models fit for the sixteen recordings in the dataset. The minimum BIC occurred for a model with ten AR and seven MA coefficients. The models are poor for either no AR or no MA coefficients: a large decrease in the BIC value is observed if the model has at least one AR and one MA coefficient, a trend that supports our choice of ARMA over AR models.

The BIC focuses on minimizing the FPE, but small residuals are not the only condition that must be met to ensure accurate haptic texture models. These models must also feel correct to the user. In order to accomplish this objective of haptic realism, we believe the model must be able to match the frequency content of the data in both amplitude and spectral shape, which is supported in [22].

In order to compare data and model spectra, we calculated the discrete Fourier transform (DFT) of the raw data using the Matlab function `fft(accel,N)` where `accel` is the raw acceleration data and `N` is the number of datapoints. This DFT was then smoothed with a 50 point Bartlett-Hann window and normalized by dividing by $\sqrt{N}$. The frequency content of the model was found using the Matlab function `bode`.

The Hernandez-Andres Goodness-of-Fit Criterion (GFC) was used to compare the spectra of the recorded vibrations and the model. This criterion was first proposed as a metric for testing reconstructed daylight spectra [25]. It is based on the Cauchy-Schwarz Inequality, and it is calculated as:

$$\text{GFC} = \frac{\left|\sum_i A_d(f_i) A_m(f_i)\right|}{\sqrt{\left|\sum_j \left[A_d(f_j)\right]^2\right|}\sqrt{\left|\sum_k \left[A_m(f_k)\right]^2\right|}} \tag{5}$$

where $A_d(f_i)$ is the amplitude of the DFT of the data at frequency f_i, and $A_m(f_i)$ is the amplitude of the model's frequency response at frequency f_i. Romero et al. [25] calculated that GFC $= 0.90$ corresponds to a reconstructed spectrum with 19% less energy than the

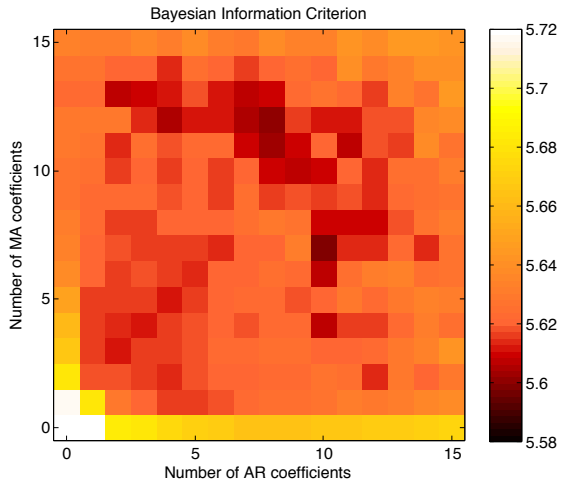

Figure 4: Bayesian Information Criterion for all considered ARMA model orders averaged across sixteen datasets for ABS plastic. The number of AR coefficients varied from zero to fifteen, and the number of MA coefficients also varied from zero to fifteen. The minimum BIC occurred for a model with ten AR and seven MA coefficients.

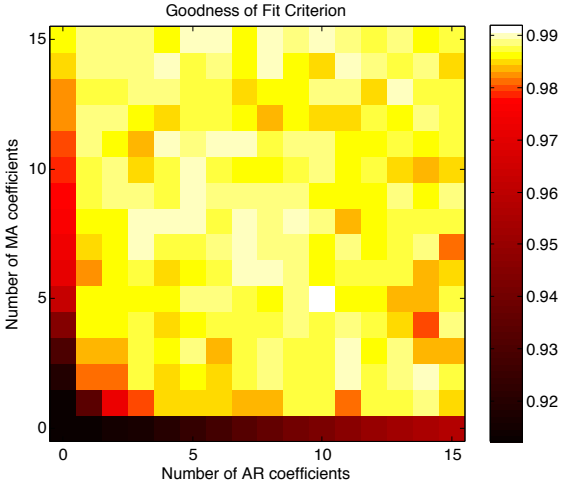

Figure 5: Goodness of Fit Criterion for all considered ARMA models averaged across sixteen datasets for ABS plastic. The maximum GFC occurred for a model with ten AR and five MA coefficients.

original data. Therefore, since the just noticeable difference (JND) for vibrations greater than 150 Hz has been experimentally determined to be 17% [22], it is reasonable to expect that GFC > 0.90 will result in a model with a good frequency match. The exact threshold would need to be determined through extensive psychophysical testing. Fig. 5 shows a color matrix plot of the GFC for all 255 possible models averaged across the sixteen datasets. The maximum GFC occurred for a model with ten AR and five MA coefficients. Similar to the BIC, the GFC shows that models are poor for either no AR coefficients or no MA coefficients.

As the metrics described above seek to optimize separate aspects of the model, they rarely agree on the appropriate model order. In order to integrate the different criteria, some further constraints must be implemented. To simplify computations during texture rendering, the same number of coefficients should be implemented for each model used to characterize a material. This work does not consider blending between two distinct textures, so it is not necessary to keep the number of coefficients consistent across materials.

The information provided by both the BIC and the GFC are important for determining the appropriateness of the model; therefore

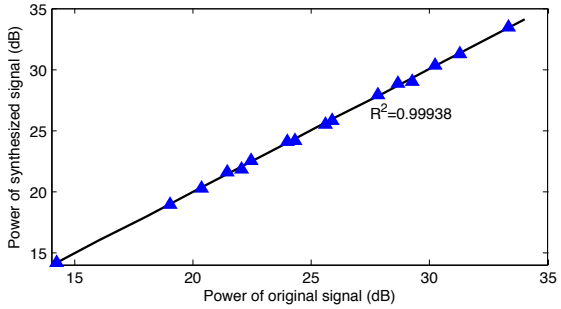

Figure 6: Comparison of power of sixteen recorded and synthesized acceleration signals. The powers exhibit a high degree of correlation.

both should be taken into account when selecting the final model order. We added the GFC to the inverse of the BIC (GFC+1/BIC) and found its maximum to determine the appropriate model order. We chose this weighting scheme to place more emphasis on spectral match over error in increasing realism of texture models. This combination of the metrics showed that models with ten AR coefficients and six MA coefficients would be best for ABS plastic. Several nearby ARMA model orders would produce similar results.

Once the appropriate-order models were found, synthetic texture accelerations were generated by driving the model transfer function with white Gaussian noise with a power equal to the FPE. Fig. 6 compares the power of the recorded and synthesized accelerations for all sixteen force-speed combinations; as expected, the R^2 value is very close to unity, showing a high degree of correlation.

4 Texture Rendering

After creating data-driven models of each material's vibration response, we use these models to synthesize textures in real time. This section discusses the method by which the texture models were implemented for haptic rendering on the TexturePad. As shown in Fig. 1, the stylus was augmented with a Haptuator (Tactile Labs) oriented parallel to the surface. This voice-coil actuator is driven by the computer's sound card at 5000 Hz via a linear current amplifier. Below, we present a method for resampling the 800 Hz models to allow for playback at 5000 Hz. Next we discuss the method for interpolating between models in real time. Finally, we detail the software and hardware used to output the accelerations to the user.

4.1 Model Upsampling

The sixteen texture acceleration datasets were all recorded at 800 Hz. Therefore, the models identified in the previous section are discrete-time transfer functions with a sampling rate of 800 Hz. This means that the output of these models can only be updated every 1.25 ms. However, the sound card of the computer requires outputs to be specified at a minimum rate of 5000 Hz. Though this sampling rate mismatch could have been resolved through the use of a general-purpose digital to analog device, this choice would limit the identified texture models to 800 Hz playback. Sharing models across platforms and between devices will eventually require sampling rate conversion, so we decided to address this issue.

The problem of recording and rendering vibrations at different rates is best handled by resampling the discrete transfer function of each model. The method proposed in [9] was found to satisfy the needs of our application. This deterministic algorithm requires that the autocorrelation function (ACF) of the original model is equal to the ACF of the upsampled model at the lags that are common to both ACFs. The method neglects the stochastic nature of the system, which is then compensated by adjusting the noise-covariance matrix of the resampled model. Fig. 7 verifies that the step response and DFT of representative original and upsampled models are equivalent.

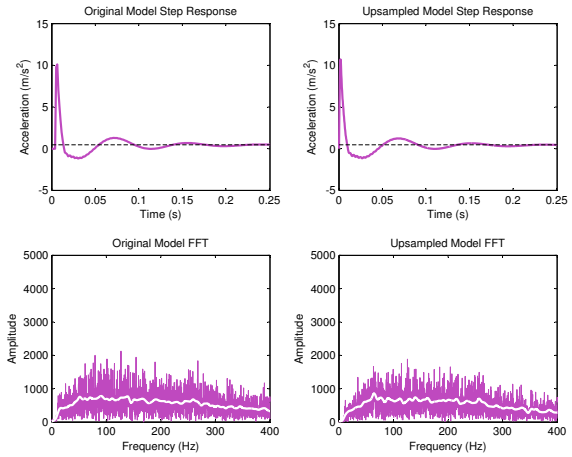

Figure 7: Step response and DFT of original and upsampled models. The step responses share the same damped frequency and settling time. The DFTs share the same frequency content.

Another approach that we tested was to run the texture model at its original sampling rate (800 Hz in our case) and use a zero-order hold to fill in outputs between model updates. It can be shown that the zero-order hold somewhat attenuates the high frequency content of the signal, reducing the spectral match, but it is a decent option that can be used to address sampling rate mismatches in certain cases. It is not an option when the rendering rate is slower than the recording rate, nor when the rendering rate is a non-integer multiple of the recording rate. A final option that was used in the previous work [23] is to upsample the original data to the rendering frequency before fitting models. We moved away from this approach because it inherently fabricates data and depends on the exact interpolation method used.

4.2 Interpolating Models

In prior work, the sixteen models for each texture were saved in a custom C++ lookup table as their coefficients and FPE, labeled by tool speed and normal force [24]. During rendering, the TexturePad measures the user's speed and force at a rate of 125 Hz. To select between available models, speed is rounded to the two closest speeds that exist within the lookup table, one lower and one higher than the actual speed, and the same procedure is used for the force. These four values are used to retrieve the four closest models within the lookup table. The user's speed and force are capped at 0.2 m/s and 1.2 N respectively; if the user's speed or force are above these thresholds, we saturate the user's speed or force to the maximum modeled value. Additionally, models are created to handle the case when the user's speed or force are zero. The coefficients for these models are equal to the coefficients of the nearest model, and the FPE is set to zero. This ensures that no vibrations occur if the user stops moving the stylus or lifts the stylus off the tablet surface.

The previous system used the measured speed and force values to perform a bilinear interpolation to calculate the new coefficients and FPE. However, it is not appropriate to simply interpolate the AR and MA coefficients of the four models because this procedure does not ensure stability. Fig. 8(a) shows the pole loci that result from interpolating between the coefficients of four models. The poles travel outside of the unit circle for some values of speed and force, thereby resulting in an unstable model. This instability must be avoided at all times in order to simulate realistic textures. Therefore it is necessary to represent the coefficients in a different manner that ensures stability and is more robust. Note that [23, 24] used the naive approach of interpolating coefficients.

It is common practice in digital speech coding to transform the coefficients into Line Spectral Frequencies (LSF). LSFs have many

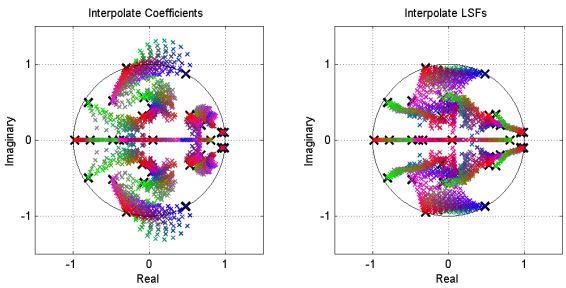

Figure 8: Loci of poles for interpolating between four models using (a) coefficients and (b) line spectral frequencies. When interpolating with the coefficients of the models, some poles travel outside the unit circle, resulting in a new unstable model. The line spectral frequencies ensure the stability of the model by requiring all poles to lie within the unit circle.

benefits over coefficients including greater robustness and less sensitivity. They encode spectral information in the frequency domain by mapping the poles and zeros in the discrete plane onto the unit circle, as described in [11]. The LSFs are defined as the angles the complex poles and zeros make with the real axis. These LSFs are interpolated between the four models selected earlier. After interpolation, the resulting LSFs are converted to the new coefficients by first mapping the poles and zeros within the unit circle. Fig. 8(b) shows the pole loci that result from interpolating between the LSFs of four models. The stability of the interpolation is ensured because the LSFs require that all poles lie within the unit circle.

4.3 Filtering and Calculating Coefficients

Although the stability of individual models is ensured when interpolating the LSFs, this is not sufficient to ensure stable behavior of the system. The user's position and force change continuously and are measured at 125 Hz. Additionally, the system must calculate a new acceleration output value at 5000 Hz. With a naive approach, the transfer function of the system changes often and quickly. This high-frequency model switching between stable models has been shown to have the potential to result in an unstable system [14], which we observed in our simulated and implemented systems.

We solve this problem by judiciously filtering the scanning speed, normal force, and LSFs. A first-order linear low-pass filter with a cut-off frequency of 8 Hz is applied to the user's speed and force to smooth quantization effects. The cut-off frequency is sufficiently above the normal hand motion bandwidth of 2 Hz [3] to avoid significant delay or attenuation of deliberate human motions.

A third-order linear low-pass filter a with cut-off frequency of 8 Hz was applied to the AR LSFs, MA LSFs, and FPE. The LSFs and FPE, which are a direct result of the user's force and speed, should not change faster than 2 Hz. Therefore, this filter serves to further attenuate additive noise from the force and speed calculations and to handle rounding errors in calculations during interpolation. After filtering, the LSFs are converted to their corresponding AR and MA coefficients using the method outlined in [11].

First- and second-order filters on LSFs and FPE were also found to help stabilize the system, but they occasionally permitted strong transients in acceleration. These transients occurred when speed or force changed quickly, and were never observed with the described third-order filter. The delay introduced by the third-order filter is approximately 50 ms. The threshold of time delay for users to experience changes in perceived textures has been experimentally determined to be 40 ms [17]. Therefore, the delay introduced by filtering the LSFs may have a small noticeable effect on the perception of our synthesized textures. Future work will explore alternative filters and other approaches to avoid this type of instability in texture synthesis without incurring a 50 ms time delay.

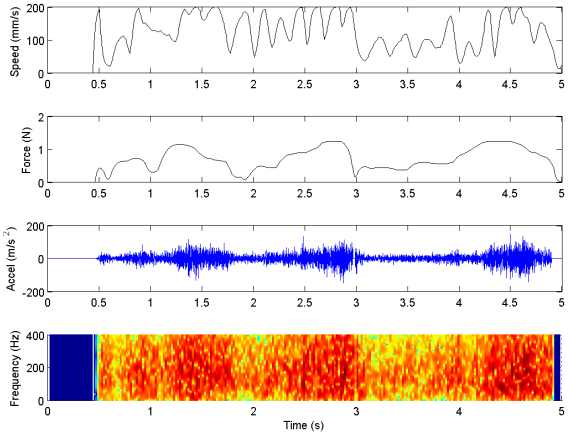

Figure 9: Data from implementation of new rough ABS plastic models on TexturePad. The power spectrum of the acceleration varies significantly with force and speed, and no unstable transients occur.

Table 1: Calculated Model Order

Material	AR Coefficients	MA Coefficients
ABS plastic	10	6
Brushed plastic	10	8
Canvas	9	9
Cardboard	10	10
Denim	15	11
Paper	5	3
Vinyl	10	7
Wood	9	5

4.4 Signal Synthesis

After the new coefficients are calculated, they are used to generate an appropriate acceleration waveform. Our software generates a white Gaussian noise excitation signal to drive the ARMA model. The power of the excitation signal is equivalent to the bilinearly interpolated FPE value. The software uses the excitation signal's history and a history of previous acceleration output values to calculate the new acceleration value at a rate of 5000 Hz.

4.5 Hardware

After the next acceleration value is synthesized, it must be played to the user. First the magnitude of the acceleration is converted to volts in order to be played back through the soundcard of the computer using the PortAudio C library [1]. The authors direct the reader to [23] for a full description of this conversion. This output voltage is passed through a linear current amplifier with a gain of 1 A/V. This output drives a Haptuator vibrotactile transducer (TactileLabs, model no. TL002-14-A) that is firmly attached to the stylus via a custom 3D-printed plastic bracket. The Haptuator shakes the stylus, transmitting the synthesized vibrations to the user. The previous implementation described in [24] used a pair of custom voice coil actuators; we now prefer the Haptuator for its very low static friction and commercial availability.

5 Results

We applied the methods described in the previous two sections to create a full haptographic texture model set for ABS plastic and render it on the TexturePad. This section discusses the results of this implementation and quantitatively compares them to the results achieved via the previous method for creating texture models described in [23].

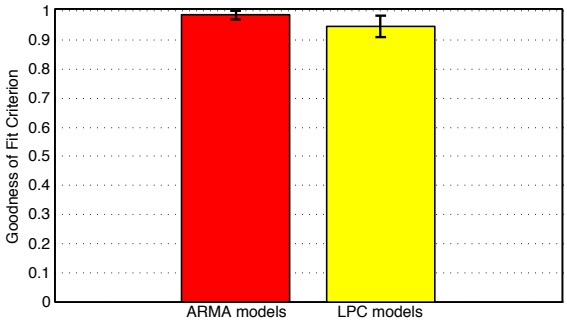

Figure 10: Goodness of Fit Criterion comparing the bode spectral content to the original data for all sixteen ARMA and LPC models. All ARMA models had a GFC value above the JND threshold of 0.9. Three of the LPC models had GFC values below this threshold.

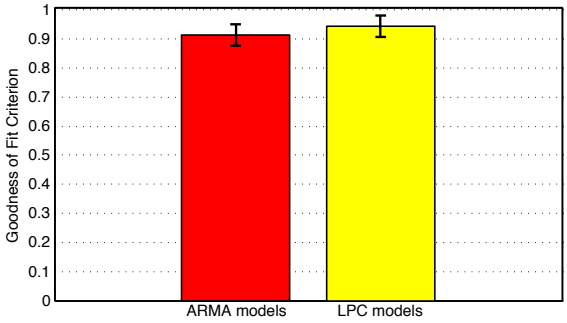

Figure 11: Goodness of Fit Criterion comparing DFT of data synthesized using the ARMA and LPC models. The GFC was calculated by comparing the DFTs of the recorded and synthesized accelerations for 80-ms-long segments of data. The average GFC for the ARMA models was 0.91 and the average GFC for the LPC models was 0.94.

5.1 Implementation Results

Sample results of implementation on the TexturePad are shown in Fig. 9. The user moved the tool across the virtual surface using natural exploratory motions, varying scanning speed and normal force as seen in the top two panels of the figure. The resulting accelerations vary in both amplitude and frequency content as a function of the speed and force. The spectrum of the signal also varies significantly over time, as we would expect from real recordings of similar exploratory movements.

5.2 Comparisons to Prior Approach

Model Order The texture modeling methods from Section 3.3 were completed for rough ABS plastic and seven other materials. The resulting model orders are shown in Table 1. All of the models needed significantly fewer coefficients than in the previous implementation of this system, which used 400 AR coefficients for every material [23]. This change reduced the model storage space requirements by more than 90%. This is a significant improvement that will be important if large databases of texture models are to be created and shared. Furthermore, using fewer coefficients lowers the computational complexity of real-time texture synthesis.

Spectral Match of Individual Models We compared the frequency responses of the ARMA and LPC models to determine if the new modeling methods affected the spectral match, since we believe this metric is central to the realism of the resulting texture renderings. First we compared the frequency content of all sixteen models with the DFT of the acceleration data used to make the model. A bar graph of the resulting GFC values is shown in Fig. 10. All of the ARMA models had a GFC value above the JND threshold of 0.9 established in Section 3.3. However, three of the LPC models had GFC values below this threshold, which may result in

perceptually noticeable differences in the frequency content of the model.

Spectral Match of Free Exploration We also compared the GFC for accelerations synthesized using the ARMA coefficients and the LPC coefficients separately. We recorded five seconds of tool speed and tool force data from free exploration of the real ABS texture sample, along with the high-frequency accelerations experienced by the tool during this interaction. We used the recorded speed and force to synthesize two acceleration output signals offline, one using the new method and the other using the prior method with LPC models. The real and synthesized acceleration signals were then divided into 80-ms-long segments, and the DFT was computed for each segment. We calculated the GFC between the real and synthesized signals for each time segment. A bar graph of the resulting GFC values is shown in Fig. 11.

For the depicted data, the average GFC values for the signals synthesized using the ARMA coefficients was 0.91, and it was 0.94 for LPC coefficients. Although the average GFC was slightly larger when using the LPC coefficients, we do not believe that the resulting differences in the spectrum of the signal would be perceptually distinguishable to the human user; more extensive analysis and human subject testing are needed to truly investigate this question. We attribute some of the differences in the GFC between the synthesized signals to the delay that occurs when filtering the LSFs, rather than deficiencies in the models, so we intend to continue looking for ways to improve this aspect of the system.

6 Conclusion and Future Work

This paper builds on previous work in the area of haptography, which involves the creation of texture models from measured acceleration, force, and speed data. The focus of the research presented here was to optimize the process of model building to create more realistic texture models and to increase efficiency of model storage and texture synthesis. Metrics for determining an appropriate model order were presented, along with new methods for upsampling models and interpolating between models to ensure stability. This approach was implemented on the TexturePad system. There was a large decrease in the space required to store the models, and the spectral match of the individual models was improved overall, though the spectral match of free exploration declined slightly.

In future work, we hope to implement a new data capture system with hardware that is capable of recording data at higher sampling rates to avoid the need to upsample the model. We also intend to explore a more efficient method of data capture. Rather than collecting controlled datasets, we envision a system that will allow the haptographer to explore the entire surface using natural motions with varied force and speed. We will then parse the data into segments of approximately constant force and speed that will be used to create the set of models for the explored surface.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0845670. The first author was supported by research fellowships from the Department of Mechanical Engineering and Applied Mechanics of the University of Pennsylvania and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-0822.

References

[1] PortAudio - Cross-Platform Audio API, April 2010.

[2] The Linux Wacom Project, April 2010.

[3] E. R. Bachmann. Inertial and magnetic tracking of limb segment orienation for inserting humans into synthetic environments. In *Ph.D. Dissertation, Naval Postgraduate School*, 2000.

[4] J. Bell, S. Bolanowski, and M. Holmes. The structure and function of pacinian corpuscles: A review. *Progress in Neurobiology*, 42(1):79–128, 1994.

[5] S. Bisgaard and M. Kulahci. Quality quandaries: time series model selection and parsimony. *Quality Engineering*, 21:341–353, 2009.

[6] G. Campion and V. Hayward. Fundamental limits in the rendering of virtual haptic textures. In *Proc. of IEEE World Haptics*, pages 263–270, 2005.

[7] V. L. Guruswamy, J. Lang, and W. S. Lee. Modeling of haptic vibration textures with infinite-impulse-response filters. In *Proc. of IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE)*, pages 105–110, Nov. 2009.

[8] M. Harders. Haptics in medical applications. In M. Lin and M. Otaduy, editors, *Haptic Rendering: Foundations, Algorithms, and Applications*, chapter 24, pages 501–515. AK Peters, 2008.

[9] A. Horch and A. J. Isaksson. Assessment of the sampling rate in control systems. *Control Engineering Practice*, 9:533–544, 2001.

[10] R. L. Klatsky, S. J. Lederman, C. Hamilton, M. Grindley, and R. H. Swendsen. Feeling textures through a probe: Effects of probe and surface geometry and exploratory factors. *Attention, Perception and Psychophysics*, 65(4):613–631, 2003.

[11] A. M. Kondoz. *Digital Speech: Coding for Low Bit Rate Communication Systems*. John Wiley and Sons, Ltd, 2004.

[12] K. J. Kuchenbecker, J. M. Romano, and W. McMahan. Haptography: Capturing and recreating the rich feel of real surfaces. In *Proc. of the 14th International Symposium of Robotics Research (ISRR)*, pages 245–260, 2009.

[13] N. Landin, J. M. Romano, W. McMahan, and K. J. Kuchenbecker. Dimensional reduction of high-frequency accelerations for haptic rendering. In *Proc. of EuroHaptics 2010*, pages 79–86, 2010.

[14] D. Liberzon. *Switching in Systems and Control*. Birkhauser, 2003.

[15] L. Ljung. *System Indentification: Theory for the User*. Prentice Hall PTR, 1999.

[16] M. Minsky, O. Y. Ming, O. Steele, J. F. P. Brooks, and M. Behensky. Feeling and seeing: issues in force display. In *Proc. of the 1990 Symposium on Interactive 3D Graphics*, volume 24, pages 235–241, 1990.

[17] S. Okamoto, M. Konyo, S. Saga, and S. Tadokoro. Detectability and perceptual consequences of delayed feedback in a vibrotactile texture display. *IEEE Transactions on Haptics*, 2(2):73–84, April-June 2009.

[18] A. M. Okamura, J. T. Dennerlein, and R. D. Howe. Vibration feedback models for virtual environments. In *Proc. of IEEE International Conf. on Robotics and Automation*, pages 674–679, May 1998.

[19] A. M. Okamura, K. J. Kuchenbecker, and M. Mahvash. Measurement-based modeling for haptic display. In M. Lin and M. Otaduy, editors, *Haptic Rendering: Foundations, Algorithms, and Applications*, chapter 21, pages 443–467. AK Peters, 2008.

[20] M. A. Otaduy and M. C. Lin. Rendering of textured objects. In M. Lin and M. Otaduy, editors, *Haptic Rendering: Foundations, Algorithms, and Applications*, chapter 18, pages 371–393. AK Peters, 2008.

[21] D. K. Pai, K. v. d. Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau. Scanning physical interaction behavior of 3d objects. In *Proc. of the 28th Annual Conf. on Computer Graphics and Interactive Techniques*, pages 87–96, 2001.

[22] H. Pongrac. Vibrotactile perception: examining the coding of vibrations and the just noticeable difference under various conditions. *Multimedia Systems*, 13:297–307, 2008.

[23] J. M. Romano and K. J. Kuchenbecker. Creating realistic virtual textures from contact acceleration data. *IEEE Transactions on Haptics*, PP(99):1, 2011.

[24] J. M. Romano, T. Yoshioka, and K. J. Kuchenbecker. Automatic filter design for synthesis of haptic textures from recorded acceleration data. In *Proc. of IEEE International Conf. on Robotics and Automation*, pages 1815–1821, May 2010.

[25] J. Romero, A. García-Beltrán, and J. Hernández-Andrés. Linear bases for representation of natural and artificial illuminants. *Journal of the Optimcal Society of America*, 14(5):1007–1014, 1997.

[26] H. Vasudevan and M. Manivannan. Recordable haptic textures. In *Proc. of IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE)*, pages 130–133, 2006.

[27] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems*, volume 3, pages 146–151, Aug. 1995.