Automatic Filter Design for Synthesis of Haptic Textures from Recorded Acceleration Data

Joseph M. Romano*, Takashi Yoshioka†, and Katherine J. Kuchenbecker*

*Haptics Group, GRASP Laboratory
University of Pennsylvania, USA
{jrom,kuchenbe}@seas.upenn.edu

†Zanvyl Krieger Mind/Brain Institute
Johns Hopkins University, USA
takashi@jhu.edu

***Abstract*— Sliding a probe over a textured surface generates a rich collection of vibrations that one can easily use to create a mental model of the surface. Haptic virtual environments attempt to mimic these real interactions, but common haptic rendering techniques typically fail to reproduce the sensations that are encountered during texture exploration. Past approaches have focused on building a representation of textures using a priori ideas about surface properties. Instead, this paper describes a process of synthesizing probe-surface interactions from data recorded from real interactions. We explain how to apply the mathematical principles of Linear Predictive Coding (LPC) to develop a discrete transfer function that represents the acceleration response under specific probe-surface interaction conditions. We then use this predictive transfer function to generate unique acceleration signals of arbitrary length. In order to move between transfer functions from different probe-surface interaction conditions, we develop a method for interpolating the variables involved in the texture synthesis process. Finally, we compare the results of this process with real recorded acceleration signals, and we show that the two correlate strongly in the frequency domain.**

I. Introduction

Modern haptic devices are very successful in creating the low frequency forces experienced during interactions with virtual environments. Often the designer will program the device to generate interaction forces by defining a spring contact between the haptic device tip and the simulated object surfaces. When the device tip intersects a region occupied by a virtual object, a restoring spring force is calculated based on the penetration depth between the haptic device and the encountered object [19]. While this approach succeeds in defining the coarse shape of virtual objects, their surfaces typically feel too smooth to be real.

This odd sensation stems from the fact that the virtual environment is often modeled as a rough approximation consisting of parametric surfaces, such as planes, stitched together to form a three-dimensional mesh. While these meshes succeed in giving the user rough **macro-scale** information such as the size and shape of the object, they fail to provide the user with any **micro-scale** features such as surface texture. Geometrically modeling these micro-scale properties is very challenging for several reasons. First, the computing power and memory necessary to store and perform collision detection algorithms on such detailed models is well beyond the capabilities of today's computers.

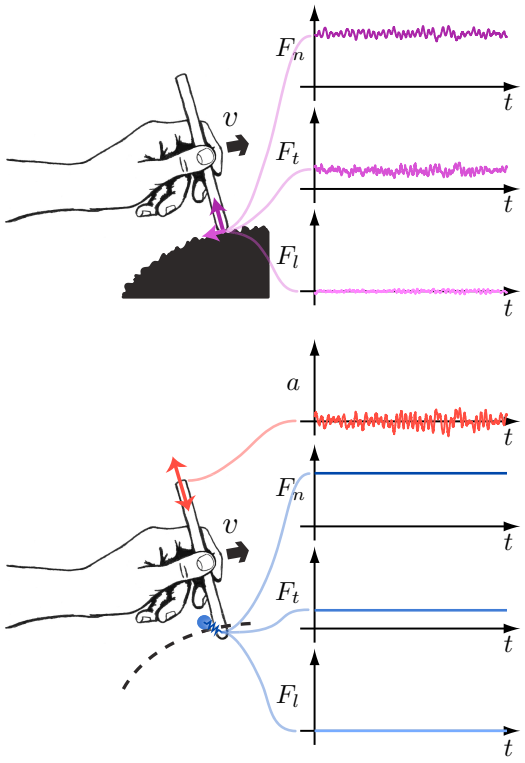

Fig. 1. **Top:** The normal, tangential, and lateral forces generated by a user holding a probe and dragging it over a real textured surface. Notice the high-frequency force components that are a result of the probe interacting with the micro-surface geometry. **Bottom:** Our proposed method for simulating contact with a textured surface. The low-frequency components of the normal, tangential, and lateral forces are rendered via a virtual proxy using standard haptic rendering techniques and an impedance-type haptic device. A high-frequency acceleration signal is generated along a single degree of freedom to convey the texture response.

Second, virtual environment mesh surfaces are often defined manually by the programmer, or from a three-dimensional dataset collected from scanned objects. In both cases the resolution of the data is not on the micro-scale necessary to discriminate texture features. Lastly, traditional haptic device controllers are not stiff enough to realistically render such fine surface details.

In the following sections we present a new method for synthesizing realistic textures. First we develop transfer

function models from real data that represent the acceleration response of a probe-surface interaction. As shown in Fig. 1, we aim to use these models to augment traditional haptic virtual environments with a high frequency vibration in order to increase the realism of the perceived interaction. We explain how to distill these models from real data, and how to generate acceleration signals in an efficient and robust way so that they are suitable for implementation in a real-time system.

A. Background

A long standing problem in computer graphics has been the efficient display of visual textures. It is often desirable to display a complex visual surface texture, but modeling the correct three-dimensional shape and computing all the appropriate lighting interactions is computationally impractical. In order to circumvent this complicated fine-detail modeling, several different solutions have been created, such as mapping images of texture onto flat surfaces to create the illusion of volumetric texture [9]. It comes as no surprise then that the earliest attempts to create virtual textures for haptics were inspired by these graphical approaches.

The early work of Basdogan et al. [1] describes several methods for creating geometric height fields based on "bump map" images from the computer graphics community. By taking a greyscale image of the texture they wish to represent, and extruding the pixels of this image to different heights based on their shading value, they are able to create a detailed surface geometry. Unfortunately, there is no reliable real-world relationship between surface height and pixel color, and this approach is also mired in the problems that go along with collision detection on large complex geometry. One can use these maps to create very detailed surfaces, but the generated surface is not guaranteed to match the real surface that the bump map is attempting to approximate. Parametric models, such as sinusoidal waveforms [4], can greatly reduce the necessary mathematical computations, but it is difficult to use these functions to generate textures with the complexity of those found in the real world.

Significant research effort has also been put into using force maps overlaid on the flat macro-scale surface of coarse three-dimensional models to reduce the computational complexity associated with generating micro-scale surface geometry. Early work by Minsky et al. [15], [16] showed that surface roughness could be simulated by applying various repelling and attracting forces to the haptic probe as it slid across a flat surface. Rather than model the complex interactions of the haptic device and the micro-surface geometry, one needs only the position of contact between the haptic device and a single flat surface patch. Once this information is known, a simple data lookup can be performed to decide what additional disturbance force should be applied to the user. These force lookup maps were generated from spatially periodic patterns or specialized noise generation functions. This approach was later advanced by using three-dimensional Gaussian force fields in uniform [20], and non-uniform distributions [6]. While this approach was computationally tractable, the authors make no attempt to correlate the generated textures with those encountered in the real world.

In order to provide sensations that closely follow haptic feedback encountered in the real world, several researchers have explored data-driven approaches [8], [18], [17], [13]. By recording data from the real world, such as the interaction force between a tool and its environment, and later playing back samples of this data during haptic simulation, it is possible to generate more realistic sensations. For example, Okamura et al. recorded cutting forces for surgical scissors during a real tissue-cutting experiment, and then developed a simulation system that used a lookup table to display these recorded forces to the user [17]. A related approach is taken by MacLean in developing the "Haptic Camera" [13], a tool that captures data during probe interactions with the real world. The recordings enable the creation of parametric linear dynamic models that represent this real world interaction. While this approach worked well for the toggle switch modeled in [13], where an a priori model structure was assumed, we will show that the dynamic vibrations occurring from probe-surface interaction do not have such intuitive analogs.

The major advantage of data-driven approaches is that they are able to generate sensations that correspond closely to real-world signals. However, these signals vary as a function of many different parameters. In the case of surface texture interaction, two such variables are scanning velocity over the surface, and contact force between the haptic device and the surface [10]. In order to be able to provide sensations over a broad range of variables, data must be taken and stored over an equally large range.

Data-driven approaches that rely on the concept of playing back recorded signals to the user must overcome a variety of additional challenges. A recorded signal inherently has a predetermined length, but it is often desirable to display a signal of differing length to the user. A naive approach to solving this problem involves looping the same signal, but humans have an uncanny ability to detect repetitive patterns, and such looped signals can often be detected. Stitching these signals together with other similar signals is also a viable option, but this can create undesirable artifacts, as smooth transitions between the various signals are difficult to achieve. Additionally, when no data exists for certain conditions, it is unclear what signal to present, and interpolating between these pre-recorded time-domain signals can produce results that feel distorted.

B. Our Approach

Many position-position haptic teleoperation environments also have the goal of realistically relaying interaction events to the master device that occur at the slave end. Unfortunately, the controller stiffness with which the two devices can be joined is severely limited by stability requirements. Furthermore, the compliant linkage systems that connect the motors and sensors of most haptic devices to the hand-held tool do a poor job of transmitting position vibrations that

occur at high-frequencies and small amplitudes [3], [14]. When the probe tip undergoes such vibrations, very little of this information is recorded by the slave device's sensors (often digital encoders mounted at the base of each joint). These high-frequency small amplitude vibrations are the same type of motion that is induced when a hand-held probe is dragged over a textured surface. Several researchers [14], [11] have shown that these high-frequency sensations can be restored by attaching an accelerometer to the slave tool, and a small dedicated high-frequency actuator to the master tool. These high-frequency actuators are able to provide the user with a better feel for the interaction by matching the second time-derivative of these position vibrations (acceleration) between the slave and master tools.

We believe that this same paradigm can be used to solve the problem of representing virtual textures. In place of the teleoperated slave, we propose a virtual model capable of generating appropriate transient acceleration signals. We hypothesize that if we can generate streams of acceleration data that match those felt in real probe-surface interaction, and then display these signals to the user with a system similar to the master device used in the teleoperation research discussed above, the resulting interactions will feel more realistic than prior approaches in virtual texture modeling. The following sections describe our methodology for creating these virtual models from recorded data.

II. Acceleration Data

This section discusses the hardware we use for collecting acceleration data of probe-surface interactions. We then explain several post-processing steps taken with the recorded data to put it in a form more appropriate for use in the generation of synthetic accelerations.

A. Data Collection Apparatus

The data collection apparatus consists of two distinct parts: a hand-held tool and a rotating drum (Fig. 2). The tool is made of a low-friction Delrin plastic with a 3 mm hemispherical diameter tip. A three-axis Kistler 8692C5M1 accelerometer is mounted on the end of the tool. This tool was specifically used in our study for its mechanical simplicity and stiffness, properties that help to eliminate any undesirable effects such as mechanical resonance or tool deformation. The drum is a cylinder capable of rotating at various servo-controlled speeds, and it has textured surfaces mounted along its exterior. Four textures were studied in this project: paper, organza, vinyl, and denim. The drum is also actuated in two additional degrees of freedom. The first is along the axis of drum rotation, so that it is possible to reposition the various texture samples on the drum surface for experimentation. The second degree of freedom is in the vertical direction. It is actuated by a DC motor in an open-loop control arrangement where the current through the motor directly determines the contact force between the drum and the probe tip.

During our data collection trials, the tool was always held by a user, as opposed to other possibilities such as a mechanical mount. We purposely chose to have a human grip the tool so that we can capture the acceleration response of the entire hand-tool system, since these are the accelerations we are later interested in recreating. Unless carefully designed and validated, a tool held by any other apparatus would experience vibrations that are different from what a human feels. The user holds the tool with their forearm resting on the arm rest, as seen in Fig. 2. A custom LabView program on a Windows PC controls the drum to maintain the desired constant velocity and upward force, and acceleration data is recorded to the PC along all three acceleration axes of the tool at a rate of 5000 Hz. Data for this study was taken for all four surfaces at tangential speeds from 20 to 300 mm/s at intervals of 20 mm/s. At each speed the force was varied from 0 to 1.4 Newtons (N) at intervals of 0.2 N. A total of fifteen different velocities were recorded at eight different force levels for the four different textures, resulting in 480 unique trials.

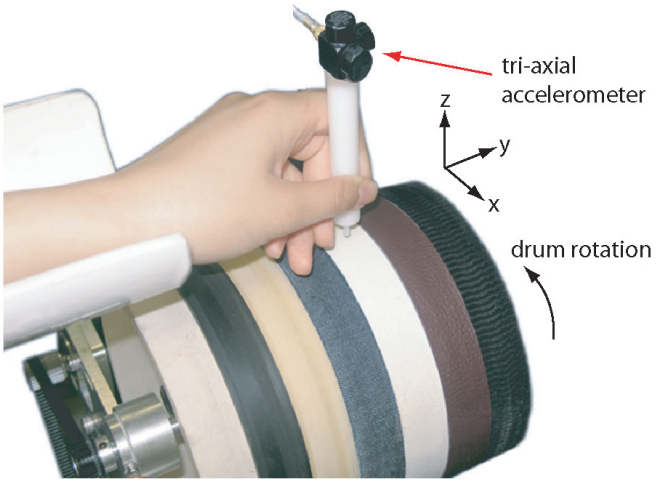

Fig. 2. The data collection apparatus used to collect acceleration transience signals. Pressure is applied in the positive z direction by the drum as it rotates about the -y axis. The drum can be automatically repositioned so that the different material strips are in contact with the probe tip.

B. Data Processing

After acceleration data is collected, we manually parse the data into two-second samples. We then convert each sample from voltage to m/s^2 using calibration constants. Since the user holds the tool in a fixed position during data collection, we know that no net motion occurs, and therefore any small steady state offset in our voltage signal is assumed to be due to sensor error. To eliminate this error, we subtract out the mean of each two-second sample.

Our next goal is to reduce the acceleration information from our three distinct axes into one single signal. We perform this reduction because it has been shown that hand's primary vibration mechanoreceptors, the Pacinian corpuscles, are not sensitive to the direction of vibration [7]. We use principal component analysis (PCA) to find the axis of vibration containing the most energy, and we project our three-dimensional acceleration onto this axis (Fig. 3). In

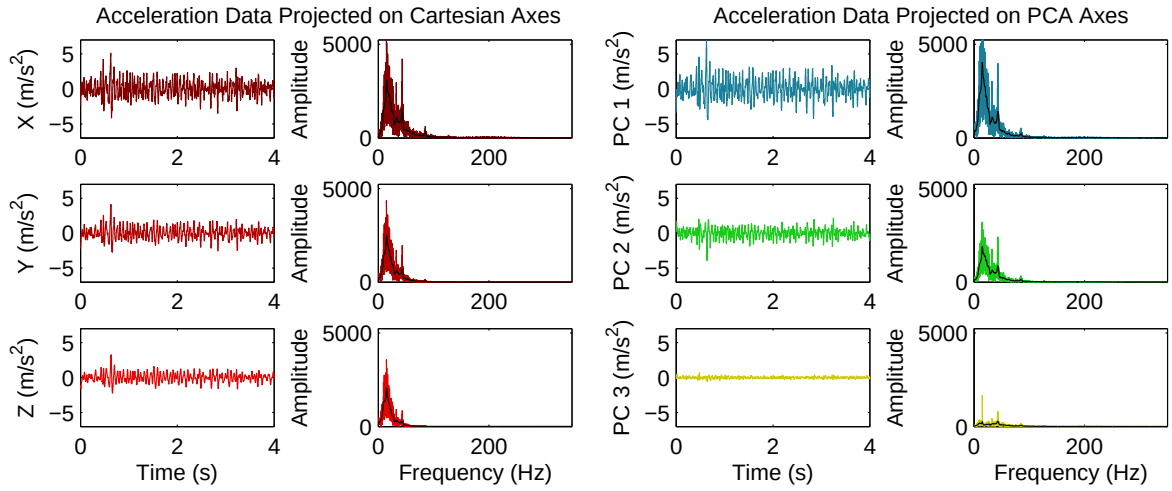

Fig. 3. Data recorded from a probe interacting with denim at a scanning velocity of 200 mm/s and a normal force of 1.0 N. The FFT of the first principal component (right) contains energy from all three of the cartesian components (left) and is one method of obtaining a single acceleration signal from three cartesian components.

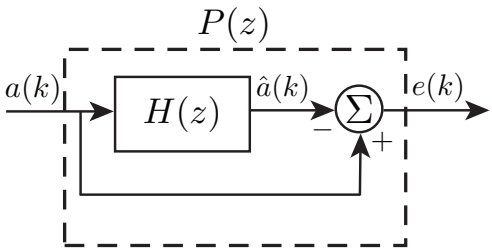

Fig. 4. Block diagram for prediction of the next contact acceleration $\hat{a}(k)$ given the recorded series $a(k)$ with residual $e(k)$.

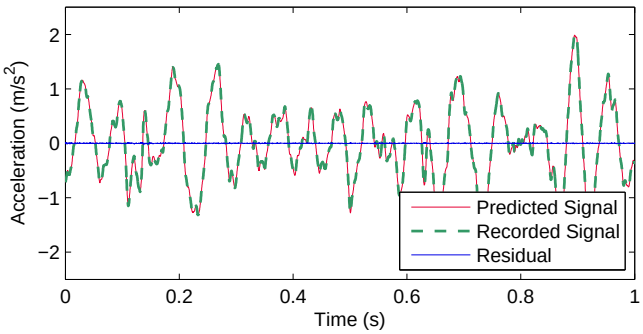

Fig. 5. Forward prediction signal generation and residual error. The data shown are from a real sample of organza fabric moving at a tangential velocity of 200.0 mm/s and a normal force of 0.8 N. The linear prediction filter $H(z)$ includes 400 coefficients.

practice we have found that projecting onto the first principal component captures 80% $\pm$ 10% of the total signal energy, depending largely on the surface type.

III. Texture Model

The previous section discussed the procedure necessary to capture surface-probe interaction data and distill it into a single meaningful time-domain acceleration signal. In this section we show how to make a transfer function model of this signal to capture its important time-domain and frequency-domain components. Once we have created this texture model, we show how to synthesize unique virtual texture signals that have these same components. We first proposed this idea in [12], and here we develop and test it further. The next section briefly summarizes the mathematics of linear prediction and acceleration synthesis below, using the notation conventions of [2].

A. Linear Prediction

Our first goal is to develop a discrete time transfer function that can predict the next sample in a texture acceleration stream based on the previous acceleration values. Fig. 4 shows the block diagram used for this system identification process. Let our acceleration data vector from PCA be called $\vec{a}(k)$, the prediction of our filter be called $\vec{\hat{a}}(k)$, and the residual of these two signals $\vec{e}(k)$. $H(z)$ is defined as an IIR filter of length n of the form $H(z) = [-h_1 z^{-1} - h_2 z^{-2} ... -h_n z^{-n}]$. We can write the transfer function for $P(z)$ as:

$$\frac{\vec{e}(k)}{\vec{a}(k)} = 1 - H(z) = P(z) \tag{1}$$

We can then define the vector of filter coefficients $\vec{h} = [h_1 \ h_2 \ h_3 \ ... \ h_n]^T$, and write out the residual at each step in time with the following difference equation:

$$e(k) = a(k) - \hat{a}(k) = a(k) - \vec{h}^T \vec{a}(k-1) \tag{2}$$

Our goal is to find an optimal filter vector, $\vec{h}_o$, that corresponds to a minimum value of $e(k)$. If we select a cost function based on mean-square error, the problem can be reduced to the Wiener-Hopf equation. In practice we apply the Levinson-Durbin algorithm [5], to solve the Wiener-Hopf equation and obtain $\vec{h}_o$. For demonstration, Fig. 5 shows a sample plot of $\vec{a}(k)$, $\vec{\hat{a}}(k)$, and $\vec{e}(k)$ for the optimal filter $H(z)$ of order $n = 400$.

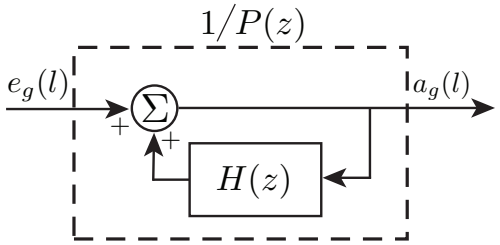

Fig. 6. Block diagram for the synthesis of an acceleration signal $a_g(l)$ from the appropriately scaled white noise input $e_g(l)$.

B. Synthesizing Accelerations

Now that we have developed the predictive filter $H(z)$, we can invert the $P(z)$ filter to synthesize new acceleration signals, as seen in Fig. 6. A white noise input signal $\vec{e}_g(l)$ is passed into $1/P(z)$ in order to excite the system and generate our desired acceleration response, $\vec{a}_g(l)$. By rewriting (1), we can formulate this new transfer function as follows:

$$\frac{\vec{a}_g(l)}{\vec{e}_g(l)} = \frac{1}{1 - H(z)} = \frac{1}{P(z)} \tag{3}$$

We now observe that the difference equation for the synthesized acceleration is:

$$a_g(l) = e_g(l) + \vec{h}^T \vec{a}_g(l-1) \tag{4}$$

During texture synthesis, we generate $e_g(l)$, a white noise signal with a Gaussian distribution of amplitudes. The average signal power of the white noise excitation, $P\{\vec{e}_g(l)\}$, is of critical importance for controlling the magnitude of the desired acceleration signal $\vec{a}_g(l)$. We use the definition of power as:

$$P\{\vec{a}(l)\} = \frac{1}{N} \sum_{n=0}^{N-1} |a(n)|^2 \tag{5}$$

Note that this definition of power is equivalent to signal variance (σ^2) because we are dealing with zero-mean signals. To synthesize a texture at a specific normal force and translational speed, the power of the generated noise signal $P\{\vec{e}_g(l)\}$ must be equivalent to that of the average signal power remaining in the residual, $P\{\vec{e}(k)\}$, after filter optimization. Fig. 7 shows one such sample in both the time and frequency domains.

C. Coefficient Order

The order of our predictive filter, $H(z)$, has a large impact on our ability to accurately predict future acceleration values. Higher order filters are desirable for their ability to reduce the residual $e(k)$, but they require more calculations when implemented in a real-time system. In order to evaluate the quality of our synthetic result, we use a cost function $C\{\vec{\hat{a}}_g(l)\}$ defined as the RMS error between the smoothed frequency-domain amplitudes of the recorded and synthesized signals, normalized by the RMS of the recorded signal's smoothed frequency-domain amplitude:

$$C\{\vec{\hat{a}}_g(l)\} = \frac{RMS\left(DFT_s\{\vec{a}(l)\} - DFT_s\{\vec{\hat{a}}_g(l)\}\right)}{RMS\left(DFT_s\{\vec{a}(l)\}\right)} \tag{6}$$

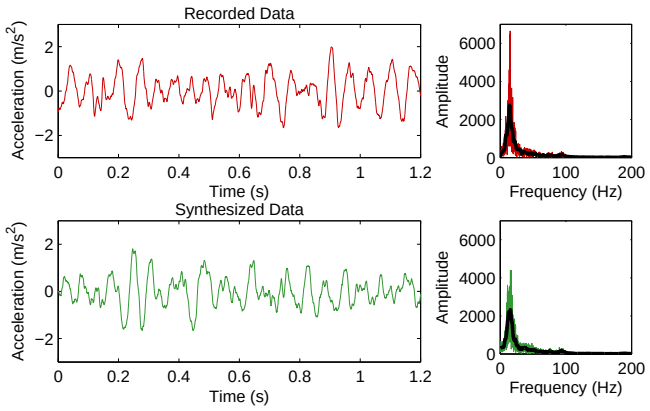

Fig. 7. Time- and frequency-domain views of a recorded acceleration and a signal synthesized to emulate that interaction using our novel texture-modeling techniques. The real setup and the synthesis filter are the same as those used in Fig. 5. While it is visually obvious that the two time-domain signals are different from one another (left), they maintain the same spectral characteristics (right), so they will feel the same to a user.

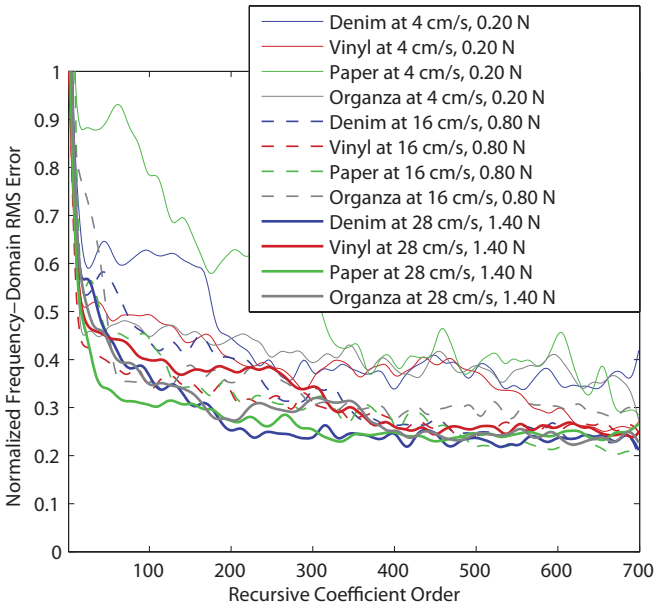

Fig. 8. Predictive filter order n versus our cost function (6) for several synthesized accelerations. For the surfaces tested in this paper, a filter with 400 coefficients was found to be the threshold where additional coefficients had minimal benefit for most tested samples.

Where $DFT_s\{\vec{x}\}$ represents the smoothed vector of amplitudes obtained from the discrete Fourier transform of vector $\vec{x}$. Using this metric we choose the highest order coefficient number after which $C\{\vec{\hat{a}}_g(l)\}$ shows little or no decrease ($n = 400$), as seen in Fig. 8.

D. Interpolation Between Models

We have now developed all the mathematical tools necessary to synthesize an acceleration signal at discrete values of normal force and tangential velocity. However, real user interactions span a continuous and constantly changing range of these parameters. Within a short sample of several seconds, a user might decide to speed up or slow down their

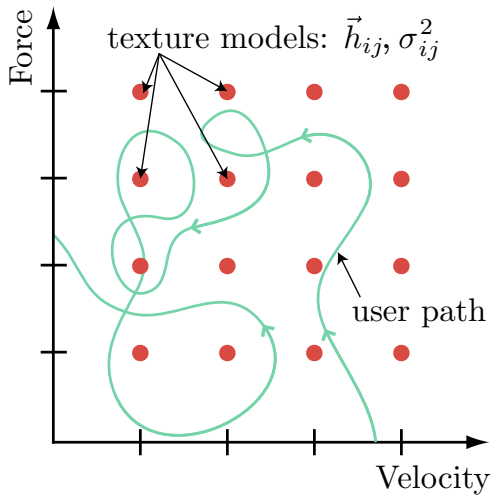

Fig. 9. A graphical example of our velocity-force grid. Model data is contained at each node in the form of a recursive coefficient vector $\vec{h}$, and the white noise signal variance σ^2. As the user moves about this two-dimensional space, we generate continuous values of $\vec{h}$ and σ^2 using bilinear interpolation.

motion, and also transition from pushing hard to pushing soft. In order to make realistic acceleration signals, we need a system that is capable of varying the synthesis parameters according to these user-driven changes.

Looking back at our synthesis equation, (4), notice there are two fixed variables that are unique for any given texture under constant force and velocity conditions: $e_g(l)$ and $\vec{h}$. The vector $\vec{h}$ has length n and contains recursive filter coefficients obtained from Linear Prediction. The value $e_g(l)$ is a sample of Gaussian white noise. While $e_g(l)$ is a randomly generated value, note that its variance (σ^2) is scaled to make the vector over time $\vec{e}_g(l)$ have the same power $P\{e_g(l)\}$ as the original residual signal.

As shown in Fig. 9, we use our data collected at standard velocity and force intervals to generate a regularly spaced grid where each node stores the recursive coefficients $\vec{h}$ and variance level σ^2 of a particular model. By applying bilinear interpolation, we are able to move smoothly between models in this two-dimensional space. As the user varies their force and velocity, we calculate a proportional value of the variance, and each individual recursion coefficient, based on their distance from the four closest grid nodes. Fig. 9 shows a schematic of our regular grid and what a user trajectory might look like.

IV. Results

Separate interpolation grids were made for each of the four test surfaces using the 120 recorded data samples per surface. We used these grids to generate test signals by purposely removing a single node of the grid, and then interpolating the data at this node from the surrounding nodes. Using this interpolated data, we generate synthetic acceleration signals using (4). A sample synthesis result is shown in Fig. 10.

Quantitative observations of the correlation between our real and synthetic interpolated signals are contained in

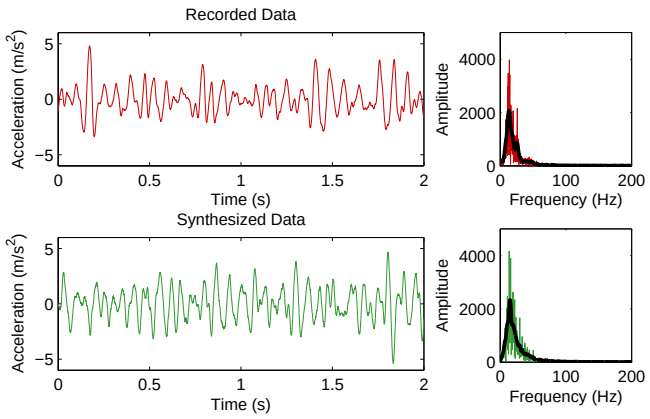

Fig. 10. Time- and frequency-domain views of both a recorded acceleration and a signal synthesized from interpolated noise variance and recursive coefficients. The recorded and synthesized data shown is of denim at a velocity of 120.0 mm/s and normal force of 1.0 N. The node containing data recorded at 120.0 mm/s and 1.0 N was purposely removed from the interpolation table so that it would be interpolated from the surrounding nodes. By removing this data from the interpolation table, we are later able to compare the results of our synthesized signal with this real data. Using our cost function (6) we calculate a 38% error in the frequency-domain correlation between the recorded and synthesized signals. While it is visually obvious that the two time-domain signals are unique (left), they have similar spectral characteristics (right).

TABLE I
Synthesized Data Correlation Table

	Organza	Paper	Denim	Vinyl
$C\{\vec{a}_g(l)\}$	29%	31%	36%	42%
$St.Dev.(C\{\vec{a}_g(l)\})$	13%	12%	9%	20%
$Avg(\sigma^2_{\vec{a}_g(l)} - \sigma^2_{\vec{a}(l)})$	$1.5e^{-5}$	$1.7e^{-4}$	$2.2e^{-4}$	$2.4e^{-4}$

Table I. The entry $C\{\vec{a}_g(l)\}$ represents the average result of our cost function (6), where lower numbers represent a better correlation between the synthetic and real DFT results. $St.Dev.(C\{\vec{a}_g(l)\})$ is the standard deviation of the above data set. The average normalized error across all surfaces is $34.5\% \pm 13.5\%$. We do not yet know whether these correlations are strong enough to fool the human sense of touch, and we plan to investigate this topic in future work. One should note that the metric we are using (6) is quite strict, and frequency domain descrepancies will likely be much lower when taking human sensory abilities into consideration.

Also in Table I, $Avg(\sigma^2_{\vec{a}_g(l)} - \sigma^2_{\vec{a}(l)})$ is the average difference between the variance of the recorded and synthesized white noise inputs. This variance discrepancy is a direct indication of how much error our bilinear interpolation introduces, and as expected the cost function increases as this error rises. Sources of error include trial-to-trial data recording variability, and any nonlinear behavior in the variance that would violate the assumptions inherent in bilinear interpolation. In addition to discrepancies between the recorded and interpolated variance, it is also possible that deviations exist between the real and interpolated recursive coefficient $\hat{h}$ vectors, though we do not examine that here.

V. Conclusion

This paper describes a novel method for generating synthetic texture signals via automated analysis of real recorded data. This method is capable of modulating the synthetic signal based on changes in two critical probe-surface interaction parameters, translational velocity and normal force, by using bilinear interpolation. We have shown that these synthetic signals are both simple and fast to compute, as well as strongly matched to their real counterparts in the time- and frequency-domains.

In future work we hope to improve our synthetic interpolated signals by interpolating alternative representations of the frequency-domain model, such as cepstral coefficients, in place of the recursion coefficients $\vec{h}$. We are also interested in comparing and perhaps combining our texture models with more event-based methods similar to those presented in [8]. We also intend to explore more natural means of data collection. One drawback of the data collection methods used in this paper is that they require precise control of normal force and translational velocity of the probe-surface interaction. We hope that by investigating more advanced segmentation algorithms we will be able to record and segment natural human probe-surface interactions at a continuous range of forces and speeds. Finally, work is currently underway in our laboratory to develop haptic devices capable of detecting the user's normal force and translational velocity and outputting the appropriate computed acceleration signal in real time.

VI. Acknowledgments

This work was supported by the National Science Foundation (grant IIS-0845670), the National Institutes of Health (grant NS054180), and the University of Pennsylvania's GAANN and Ashton graduate research fellowships. We thank our colleague Austin Chen for his assistance with the data collection system, and our colleague Will McMahan for his insights and assistance in the mathematical concepts developed in this work.

References

[1] C. Basdogan, C.-H. Ho, and M. A. Srinivasan. A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. In *Proceedings of the ASME Dynamic Systems and Control Division*, pages 77–84, 1997.

[2] J. Benesty, M. M. Sondhi, and Y. Huang, editors. *Springer Handbook of Speech Processing*. Springer Berlin / Heidelberg, 2008.

[3] G. Campion and V. Hayward. Fundamental limits in the rendering of virtual haptic textures. In *Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 263–270, Washington, DC, USA, 2005. IEEE Computer Society.

[4] S. Choi and H. Z. Tan. Toward realistic haptic rendering of surface textures. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, pages 125–132, New York, NY, USA, 2005. ACM.

[5] J. Durbin. The fitting of time series models. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, 28(3):233–244, 1960.

[6] J. P. Fritz and K. E. Barner. Stochastic models for haptic texture. In M. R. Stein, editor, *Telemanipulator and Telepresence Technologies III*, volume 2901(1), pages 34–44. SPIE, 1996.

[7] E. P. Gardner and C. I. Palmer. Simulation of motion on the skin. i. receptive fields and temporal frequency coding by cutaneous mechanoreceptors of optacon pulses delivered to the hand. *Journal of Neurophysiology*, 62(6):1410–1436, 1989.

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

[9] P. Heckbert. Survey of texture mapping. *Computer Graphics and Applications*, 6(11):56–67, November 1986.

[10] R. L. Klatzky, 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. *Perception & Psychophysics*, 65:613–631, 2003.

[11] D. A. Kontarinis and R. D. Howe. Tactile display of vibratory information in teleoperation and virtual environments. *Presence: Teleoperators and Virtual Environments*, 4:387–402, 1995.

[12] K. J. Kuchenbecker, W. McMahan, and J. M. Romano. Haptography: Capturing and recreating the rich feel of real surfaces. To appear In Proceedings of the 14th International Symposium of Robotics Research (ISRR 09), Lucerne, Switzerland, Aug. 31-Sept. 3 2009.

[13] K. E. MacLean. The "Haptic Camera": A technique for characterizing and playing back haptic properties of real environments. In *Proceedings of the ASME Dynamic Systems and Control Division 5th Annual Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, volume DSC-58, pages 459–467, 1996.

[14] W. McMahan and K. J. Kuchenbecker. Haptic display of realistic tool contact via dynamically compensated control of a dedicated actuator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent RObots and Systems,*, 2009.

[15] M. Minsky and S. J. Lederman. Simulated haptic textures: Roughness. In *Proceedings of the ASME Dynamics Systems and Control Division*, volume 58, pages 421–426. ASME, 1996.

[16] M. Minsky, O.-y. Ming, O. Steele, F. P. Brooks, Jr., and M. Behensky. Feeling and seeing: issues in force display. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, volume 24, pages 235–241. ACM, 1990.

[17] A. M. Okamura, R. J. Webster, J. Nolin, K. W. Johnson, and H. Jafry. The haptic scissors: Cutting in virtual environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 828–833, September 2003.

[18] 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 *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 87–96. ACM.

[19] K. Salisbury, F. Conti, and F. Barbagli. Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications*, 24:24–32, 2004.

[20] J. Siira and D. K. Pai. Haptic texturing - a stochastic approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 557–562, April 1996.