Data-Driven Modeling of Anisotropic Haptic Textures: Data Segmentation and Interpolation

Arsen Abdulali and Seokhee Jeon

Department of Computer Science and Engineering, Kyung Hee University, Republic of Korea,
{abdulali, jeon}@khu.ac.kr

Abstract. This paper presents a new data-driven approach for modeling haptic responses of textured surfaces with homogeneous anisotropic grain. The approach assumes unconstrained tool-surface interaction with a rigid tool for collecting data during modeling. The directionality of the texture is incorporated in modeling by including 2 dimentional velocity vector of user's movement as an input for the data interpolation model. In order to handle increased dimentionality of the input, improved input-data-space-based segmentation algorithm is introduced, which ensures evenly distributed and correctly segmented samples for interpolation model building. In addition, new Radial Basis Function Network is employed as interpolation model, allowing more general and flexible data-driven modeling framework. The estimation accuracy of the approach is evaluated through cross-validation in spectral domain using 8 real surfaces with anisotropic texture.

Keywords: Data-Driven Modeling, Anisotropic texture, RBF Networks

1 Introduction

Surface haptic texture is one of the essential information for human to discriminate objects. While small-scale geometry variation is one of the main causes of haptic texture, human can effectively perceive fine details of the variation through not only a bare-hand interaction, but also tool-mediated stroking as high frequency vibrations [10]. Sometimes, these small-scale geometry variations can be anisotropic: the characteristic of the vibration varies depending on the stroking direction. This direction-dependent haptic texture sometimes plays as a crucial cue for haptically identifying surfaces, e.g., identifying wooden surface based on directional grain, judging the quality of fabric using thread grain, etc.

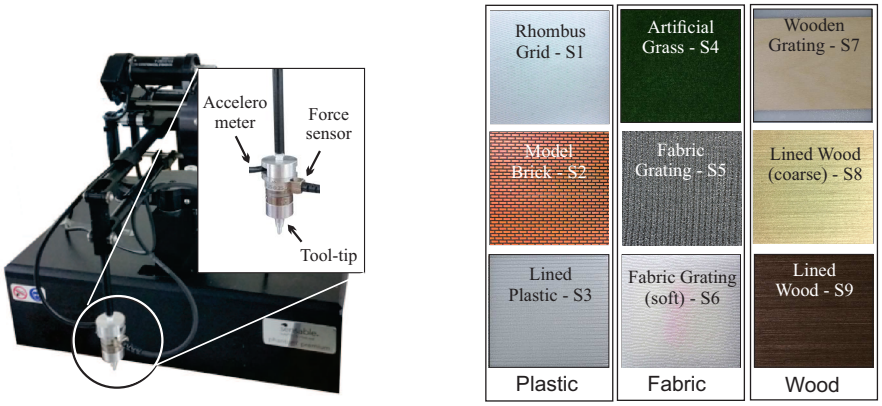
Realistic reproduction of haptic texture feedback for virtual reality has been constantly investigated in the haptics community, e.g., [2, 13, 6, 12]. Among them, pure data-driven texture modeling and rendering is one of the emerging techniques [18, 4, 1, 19]. This approach records necessary signals for texture responses, e.g., high frequency vibrations through active stroking of a target surface and uses them in rendering for approximating the responses of the target surface under given user's interaction. It can effectively handle complex high frequency behaviors with less computational load without knowledge about the surface and system. Consequently, it is one of the most relevant approaches for applications requiring high haptic realism [14].

The most relevantly related work to this paper is a series of researches done by Kuchenbecker's research group. Their approach assumes that the frequency responses due to texture is dependent mostly on normal force and tangential velocity of the interaction. Their modeling process collects data pairs for this relationship and stores them in an appropriate interpolation model. The rendering algorithm then interpolates the stored data. In their earliest work, they used linear predictive coding (LPC) for modeling the vibration signal [16], where 400 buffered output were used for predicting next results in order to achieve reasonable accuracy. This work was improved so that it was allowed to build the model from the 2-dimensional surface data using interactive texture display [15]. Another improvement was done for the efficiency of the prediction model: 400 order LPC model was replaced with ARMA(Auto Regressive Moving Average) model in [3]. Their recent work focused on the usability of modeling: unconstrained interaction is allowed for data collection [4]. This was done by segmenting captured data so that signals in each segment was kept stationary for stable prediction. They performed segmentation along output acceleration signals using AutoPARM algorithm. Although their work demonstrates the competence of their approach, non of them considered directional dependency of the texture yet.

Only a few efforts has been done for modeling anisotropic texture. Guruswamy et al. tried to incorporate directionality by mapping the texture models onto scanning path on the object surface [7]. Recently, a frequency-decomposed neural network model was applied for data-driven modeling of isotropic surface textures, which can be also extended to directional texture modeling [17]. The main idea was the decomposition of acceleration signals based on frequency level. Their work, however, was limited to only use constant force and velocity inputs in the modeling. This requires dedicated sampling hardware, which significantly decrease usability of the system.

In this paper, we introduce a new method allowing modeling of both isotropic and anisotropic textures through unconstrained tool-based interaction. In order to incorporate the directionality of the texture, we newly take tool's two dimensional velocity vector along with a scalar normal force as an input for the interpolation. For unconstrained interaction, we follow approach in [4]: the whole data are segmented into a limited number of data set each of which contains relatively stationary data. Representative input-output pairs from the segments become samples used for building interpolation model.

However, due to increased dimentionality, original algorithms used in [4] cannot be applied. For instance, due to directional data in input (2D velocity vector), the original acceleration-based (output-based) segmentation algorithm now does not guarantee stationary inputs, which breaks assumption of the algorithm. In addition, output-based segmentation does not generate evenly scattered input samples in the input space, which may deteriorate prediction. To cope with this, we introduced new segmentation algorithm where the modeling data is segmented based on input data vector (see Section 3.1). In addition, we revised the interpolation algorithm to handle the increased input dimension: a Radial Basis Function Network (RBFN) is used for storing and interpolating acceleration segments of vibrations (see Section 3.2). Our algorithm is tested with 9 real sample surfaces (see Section 4).



(a) Recording hardware. (b) Sample set for evaluation.

Fig. 1. Hardware for recording and sample set.

2 Data Acquisition

2.1 Recording Hardware and Sample Set

Our recording setup consists of a haptic interface and two sensors attached to it (see Fig. 2 (a)). The interaction tool of the haptic interface (Phantom Premium 1.0; Geomagic Inc.) is instrumented with an accelerometer (ADXL335; Analog Devices) and a force/torque sensor (Nano17; ATI Technologies), allowing us to measure the position, acceleration, and applied force of the interaction tool tip. The force/torque sensor is connected to a NI DAQ acquisition board (PCI-6009; National Instrument), while the accelerometer is plugged to the portable data acquisition device (USB-6220; National Instrument). For data recording and preprocessing, a desktop PC with Intel Core i5-3570 CPU and 24 Gb DDR-3 RAM was used.

In order to evaluate our algorithm, 9 samples in three different groups are prepared as shown in Fig. 2 (b). The first group consists of hard plastic materials (S1-S3), the second group is wooden samples (S7-S9), and the last group has samples made of fabric (S4-S6). Eight samples have anisotropic texture, while only S4 has isotropic texture.

2.2 Data Capturing and Preprocessing

Data for modeling are recorded through an unconstrained stroking with the interaction tool. In our implementation, 3D position data are captured in 1 kHz sampling rate. From the position data, 3D velocity is also estimated and then is projected onto a tangential plane of the contacting surface, generating a 2D velocity vector. 3D force data are captured in 10 kHz and downsampled to 1 kHz for synchronization with position data. Then, the force measurements are projected onto the normal direction of the target surface, yielding a scalar normal force data. Acceleration data are also sampled in 10 kHz and downsampled to 1 kHz.

In order to remove noise, the position, velocity, and force signals are lowpass-filtered using a Bessel filter with 25 Hz cutoff frequency. Acceleration signals are band-pass filtered within the range of 10 Hz and 1000 Hz in order to remove a gravity component and a noise.

For the segmentation procedure in the next section, we normalize the position data in a range from zero to one. The normal force and the two velocity components are also normalized to the zero mean and the unit variance, becoming inputs for interpolation while acceleration is the output for the interpolation. Normalization coefficients are preserved for the further use in rendering.

3 Modeling Approach

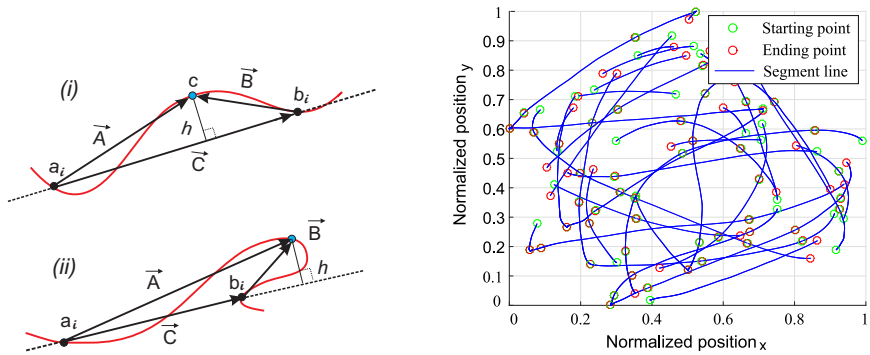
After input-output data pairs are captured through the recording hardware, we follow the prodecedure revised from [4]. We first segment the data pairs in a way that data in a segment are nearly stationary. Since the output of the interpolation is a high frequency acceleration signal, modeling it in frequency domain is more beneficial, and consequently, data pairs should be analyzed in a group, rather than individually. It is also beneficial to have a stationary data-pair in a group, which implies that a sophisticated data pair segmentation algorithm is needed (see Section 3.1). After the segmentation, representative (or averaged) values in each segment becomes a sample input-output pair used for model building and the interpolation in rendering. Using these representative samples, an interpolation model is built (see Section 3.2).

3.1 Data Segmentation

In [4], the stationary data pair requirement was fulfilled using an assumption that stationary output (acceleration signal) guarantees the input also in stationary. Their implementation watched the output acceleration signals and segmented the data based on the variation in them. However, this assumption may not hold for anisotropic texture modeling where the directional velocity is used as input. For example, acceleration signals from one directional stroking may be very similar to that from stroking toward the other direction. Completely different input data might be grouped together in their algorithm.

To remedy this, we decided to perform segmentation based on the input data space. Adjacent data-pairs in time, which has little variation in direction of movement, velocity, and normal force are grouped together. We define two thresholds for determining variations, one for testing variation in direction (τ_1) and another for limiting the velocity and normal force variation (τ_2).

Among several linear segmentation techniques, we choose a bottom-up algorithm considering a trade-off between segmentation quality and time complexity [11]. A bottom-up algorithm breaks the initial set of time series data into the finest segments (two points) and merges a pair of neighboring segments that have the lowest merging cost. Merging costs of neighboring pairs are stored apart and recalculated for each neighboring segment iteratively. The segment merging cycle stops when the minimal cost from entire set is greater than a predefined threshold value.



(a) Segment constraints. (i) - Score estimation; (ii) - Loop-out prevention.

(b) Segmentation results of 10 seconds data.

Fig. 2. Segmentation algorithm and results.

In our implementation, the segmentation is complete in two steps. In the first phase, the algorithm splits the initial set into multiple straight lines based on its position data. In the second phase, the straight lines are segmented in a way that each segment has nearly constant normal force and velocity magnitudes. More details are below.

Phase 1 Suppose that we have a multivariate time series X that should be broken down into several piecewise linear segments $S_i = [a_i, b_i] \in S$. The goal of this phase is to find the minimal number of segments, where each sample point from S_i satisfies

$$\frac{\|A \times B\|}{\|C\|} \leq \tau_1, \tag{1}$$

where A and B are adjacent vectors in the position data (see Fig. 2(a)-i). This equation ensures that the distance h between each sample point of every segment and the line segment $a_i b_i$ should not surpass the threshold τ_1 (See Fig. 2(a)-i). In addition, in order to prevent loops and ensure that particular point lays on the line segment S_i (see Fig. 2(a)-ii),

$$\frac{\sqrt{\|A\|^2 - h^2} + \sqrt{\|B\|^2 - h^2}}{\|C\|^2} = 1. \tag{2}$$

In our implementation τ_1 was set to 2 mm, which was decided in accordance with the tooltip and the textured plate sizes.

Phase 2 In the second phase, we divide each segment S_i into subsegments containing similar magnitude values of velocities and normal forces. The segmentation process stops when the minimal merging cost of (3) exceeds the threshold τ_2.

$$\frac{1}{2}\left(\frac{\sigma_v}{\mathbb{E}(\mathrm{V}_j)} + \frac{\sigma_f}{\mathbb{E}(\mathrm{F}_j)}\right) \leq \tau_2, \quad \text{for } \mathbb{E}(\mathrm{V}_j),\, \mathbb{E}(\mathrm{F}_j) > \varepsilon \tag{3}$$

where $\mathbb{E}[\cdot]$ denotes the expected value, σ_v and σ_f are standard deviations of velocity

and normal force magnitudes V_j and F_j, respectively. The expected value below ε is considered as zero.

The overall segmentation procedure is culminated with elimination of segments that are shorter than 75 samples, which is assumed to be data in transition period with very high curvature. Note also that we skip merging of neighboring segments if a mutual mean of velocity or normal force magnitudes is equal to near zero. Number of data in these subsegments is normally very small, and they are removed from the set S_i. It is reasonable since subsegments with near zero mean velocity or normal force can be assumed to be data from very beginning or very end of the contact, which normally has very small vibration. The values of τ_2 and ε were set to 0.08 and 0.001 respectively, which are found from our empirical tests.

The segmentation result of 10 second data is shown in Fig. 2(b).

3.2 Interpolation Model

The goal of the interpolation model is to estimate the vibration output under a given input data sample based on interpolating captured data. We denote the input data sample as a 3D vector, $\mathbf{u} = \langle v_x, v_y, f_n \rangle$, where v_x and v_y are 2D tool velocity vector, and f_n is a normal response force. Since output of the interpolation model is a time-series high frequency vibration, it is more convenient to express it using a time-varying parametric model. For this, auto regression model (AR) are commonly used in data-driven haptic texture rendering, which we also use.

However, the coefficients of the AR model cannot be directly interpolated due to stability problem, which happens when poles of the transfer function H in Fig.3 are not within the unit circle in the complex plane [5]. Therfore we convert AR coefficients into line spectrum frequency (LSF) coefficients for storing in the interpolation model as introduced in [4]. For rendering, we restore the AR coefficient from LSF model.

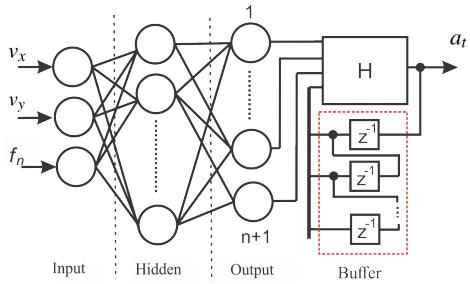


Fig. 3. Model storage and interpolation RBFN architecture.

Another contribution of this paper is the use of a radial basis function network (RBFN) as an interpolation model for texture modeling. The RBFN interpolation model outperforms simplex based in two aspects. First, the output is computed using basic mathematical operations that makes it fast whilst the interpolation result remains good.

Second, the input space can be easily increased. For example it is possible to store several different models inside of the network, switch them or even interpolate using additional input during rendering.

RBFN architecture that we used in this work consists of three layers (see Fig. 3). The input of the network is a vector $\mathbf{u} \in \mathbf{R}^n$ of the n-dimensional Euclidean vector space. The nodes of the hidden one are non-linear RBF activation functions. The output of RBFN is a scalar function of the input vector, $\phi : \mathbf{R}^n \to \mathbf{R}$, which is described as

$$f(u) = \sum_{j=1}^{N} w_j \phi(\|u - q_j\|) + \sum_{k=1}^{L} d_k g_k(u), \qquad \mathbf{u} \in \mathbf{R}^n \tag{4}$$

where w_j is the weight constant and q_j is the center of the radial basis function. The functions $g_k(u)$ $(k = 1, ..., L)$ form a basis of the space $\mathbf{P}_m^n$ of polynomials with degree at most m of n variables. Since we use the first order polyharmonic splines $\phi(r) = r$ as the kernel for RBF, the polynomial term is necessary. Otherwise, the interpolation results might be not as accurate as we want [9]. Using equation (4), a linear system can be obtained to estimate the weight constant vector $\mathbf{w}$ of radial basis functions as well as the polynomial coefficient vector $\mathbf{d}$, such that

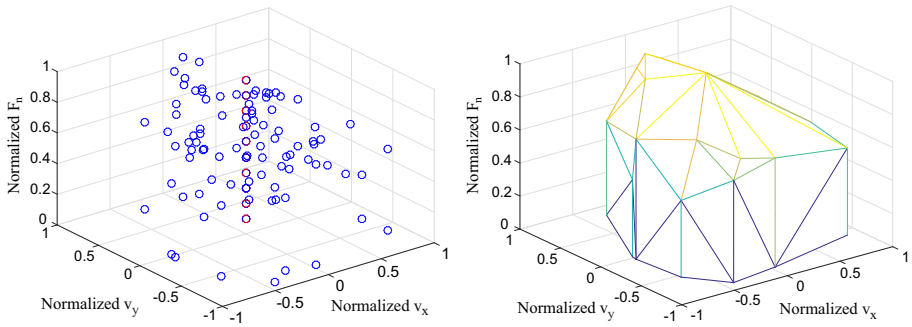
$$\begin{pmatrix} \Phi & G \\ G^T & 0 \end{pmatrix} \begin{pmatrix} w \\ d \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{5}$$

where $\Phi_{ij} = \phi(\|u_i - u_j\|)$, and $G_{ik} = g_k(u_i)$ of the range $i, j = \{1, ..., N\}$, $k = \{1, ..., L\}$.

The input vector $\mathbf{u}$ is fed into three nodes of input layer. There are n outputs each of which corresponds to each LSF coefficient. One extra output is for interpolation of the variance provided by the Yule-Walker algorithm [8].

Once a set of LSF coefficients is obtained, the output vibration can be estimated in two steps. First, the estimated coefficients are converted to AR. Second, the vibration value is calculated applying Direct Form II digital filter along with n previous outputs. It is common way to use digital filter for AR signal estimation. The digital filter we used in this work can be replaced by any of a kind.

The RBFN is trained as follows. First, the representative input points are calculated from each segment by averaging data points in a segments after the zero-mean/unit variance normalization along each axis of $\mathbf{u}$ (see Fig. 4(a) as an example). Second, in order to cover zero normal force area, we select points that lies on the convex hull of existing points and are facing the $\langle v_x, v_y \rangle$ plane. Then we project them onto the $\langle v_x, v_y \rangle$ plane (see Fig. 4(b)). For the new points at the zero normal force, the LSF coefficients is copied from that of the original point, while the variances are set to zero. In case of zero velocity, new model points are uniformly created and scattered along f_n axis, whose variance is set to zero, and the LSF coefficients are copied from the closest model points. Lastly, using the model points, we trained RBFN applying a SpaRSA algorithm [20] that identifies sparse approximate solutions to the undetermined system (5) using an extended set of features from the previous step. We found SpaRSA as the most suitable method for our work in terms of processing time and accuracy [21].



(a) Model points.

(b) A convex hull that limits input feature space during rendering.

Fig. 4. Input feature space of the model.

4 Evaluation

This section evaluates our approach in a cross-validation.

4.1 Error Metric

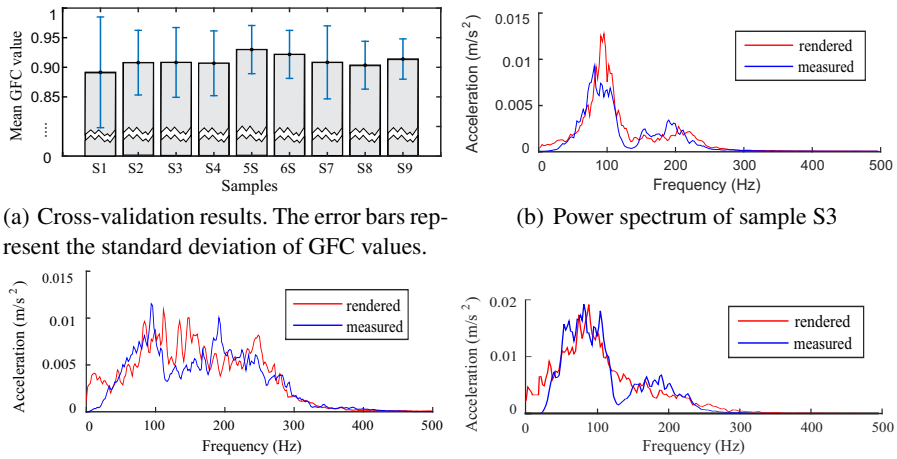
For testing the vibration signal, a frequency spectural comparison was conducted. As a comparison criterion, a Hernandez-Andres Goodness-of-Fit Criterion (GFC) was used to measure the error between the power spectrum of rendered and measured acceleration signals. The GFC is based on the Schwartz inequality and the value is between 0 and 1: 1 means perfect reconstruction. However there is no linear relationship between reconstruction quality and the GFC value. According to [3], it is reasonable to expect that GFC > 0.90 can be considered as a good frequency match of measured and synthesized acceleration signals of haptic texture. GFC can be derived as

$$GFC = \frac{\|\sum_i A_d(f_i)A_m(f_i)\|}{\sqrt{\|\sum_j [A_d(f_j)]^2\|}\sqrt{\|\sum_k [A_m(f_k)]^2\|}}\ , \tag{6}$$

where $A_d(f_i)$ and $A_m(f_i)$ are the DFT amplitudes of the frequency response at a frequency f_i of the measured and modeled signals, respectively.

For cross validation, we collected data for each sample surface twice; one for the training and the other for the cross-validation. The training data was recorded for 20 seconds so that it covered the input space while the cross-validation data was taken for 10 seconds. The cross-validation data was preprocessed in the similar manner with training data. The input cross-validating parameters was put on the same scale space with normalized modeling ones.

The captured and synthesized signals were divided into the windows of 500 data points with 50% overlap, so each window shares half data points left neighboring window and rest of data points with the right one. Each window from captured signal is



(a) Cross-validation results. The error bars represent the standard deviation of GFC values.

(b) Power spectrum of sample S3

(c) Power spectrum of sample S1.

(d) Power spectrum of sample S7

Fig. 5. Cross-validation results.

compared with the estimated counterpart using (6). The average GFC value for all windows shows the overall performance of algorithm. Also, in order to prevent a spectral leakage, we smoothed DFT components using the Whelch's power spectral density estimate with the 50 points Hamming window.

4.2 Results and Discussion

Fig. 5 shows the overall GFC values for each sample and examples of power spectrum comparison. As shown in Fig. 5(a), all samples showed averaged GFC values higher then 0.9 except for S1, which indicates that our estimation algorithm effectively captures the frequency resposnes of the vibration. One reason for lower GFC score and large standard deviation of S1 may be the wider frequency range of S1 as shown in Fig. 5(c) compared with other samples. The relatively small mean GFC with high deviation can be caused by several considerable drops in GFC values across the windows. It means that the data used in modeling did not cover the whole input space of validating data set. This shortcoming is common in data-driven modeling, and can be solved by careful data collection for modeling. Another possible reason for the relatively bad performance of S1 can be the nature of the sample. The rhombus grid is made of tiny plastic threads that were losing its initial properties during capturing the data for modeling. The frequency content of the signal during the second recording session could differ. Nevertheless, we consider the result of the model S1 still satisfactory, as the score is still very close to 0.9. Another aspect that we noticed from the power spectrum graphs is that the synthesized signals tend to exhibit considerably higher amplitude at very low frequencies (see the discrepancies at 0-20 Hz frequency). Note that this range of frequencies was filtered out in both the acceleration for model building and that for the cross-validation. One of the possible sources of these discrepancies can be the latency of the buffer update. As it can be seen in Fig. 3, the vibration values are queued

in a buffer and used for the next output prediction. In general, latency due to buffered output plays a role of low pass filter, and our implementation of buffered coefficient may generate undesired and arbitrary low frequency signals. For example, our implementation uses 16-sample long buffer, which may contain 1 to 8 Hz changes depending on user's movement, which can make these discrepancies.

5 Conclusion and Future Work

The main goal of this work is to model and render vibrations due to directional textures. Towards this goal, we develop a model architecture where a radial basis function network is used to estimate LSF coefficients that will be converted to an AR polynomial to predict an acceleration output. In order to train the model from data captured during free-hand movement, we also develop a new segmentation algorithm that utilizes input data space. The performance of the algorithm is validated using 9 real texture samples. The results show the effectiveness of the proposed algorithm.

As our tangible future work, actual rendering hardware will be incorporated with the algorithm, and the realism of the output vibrations will be assessed through a psychophysical testing.

Acknowledgments

This research was supported by Basic Science Research Program through the NRF of Korea (NRF-2014R1A1A2057100), by Global Frontier Program through NTF of Korea (NRF-2012M3A6A3056074), and by ERC program through NRF of Korea (2011-0030075).

References

1. Andrews, S., Lang, J.: Haptic texturing based on real-world samples. In: Haptic, Audio and Visual Environments and Games, 2007. HAVE 2007. IEEE International Workshop on. pp. 142–147. IEEE (2007)
2. Basdogan, C., Ho, C., Srinivasan, M.A.: A ray based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. In: ASME Winter Annual Meeting. vol. 61, pp. 77–84 (1997)
3. Culbertson, H., Romano, J., Castillo, P., Mintz, M., Kuchenbecker, K.: Refined methods for creating realistic haptic virtual textures from tool-mediated contact acceleration data. In: Haptics Symposium (HAPTICS), 2012 IEEE. pp. 385–391 (March 2012)
4. Culbertson, H., Unwin, J., Kuchenbecker, K.: Modeling and rendering realistic textures from unconstrained tool-surface interactions. Haptics, IEEE Transactions on 7(3), 381–393 (July 2014)
5. Erkelens, J.S.: Autoregressive modeling for speech coding: Estimation, interpolation and quantization. Chapter 4: Spectral Interpolation. TU Delft, Delft University of Technology (1996)
6. Fritz, J.P., Barner, K.E.: Stochastic models for haptic texture. In: Photonics East'96. pp. 34–44. International Society for Optics and Photonics (1996)

7. Guruswamy, V.L., Lang, J., Lee, W.S.: Modelling of haptic vibration textures with infinite-impulse-response filters. In: Haptic Audio visual Environments and Games, 2009. HAVE 2009. IEEE International Workshop on. pp. 105–110. IEEE (2009)
8. Hayes, M.H.: Statistical digital signal processing and modeling. John Wiley & Sons (2009)
9. Iske, A.: Multiresolution methods in scattered data modelling, vol. 37. Springer Science & Business Media (2004)
10. Katz, D.: The world of touch (le krueger, trans.). Mahwah. NJ: Rrlbaum.(Original work published 1925) (1989)
11. Keogh, E., Chu, S., Hart, D., Pazzani, M.: Segmenting time series: A survey and novel approach. Data mining in time series databases 57, 1–22 (2004)
12. Kim, L., Kyrikou, A., Sukhatme, G.S., Desbrun, M.: An implicit-based haptic rendering technique. In: Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. vol. 3, pp. 2943–2948. IEEE (2002)
13. Okamura, A.M., Dennerlein, J.T., Howe, R.D.: Vibration feedback models for virtual environments. In: Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on. vol. 1, pp. 674–679. IEEE (1998)
14. Okamura, A.M., Kuchenbecker, K.J., Mahvash, M.: Measurement-based modeling for haptic rendering. Haptic Rendering: Foundations, Algorithms, and Applications pp. 443–467 (2008)
15. Romano, J.M., Kuchenbecker, K.J.: Creating realistic virtual textures from contact acceleration data. Haptics, IEEE Transactions on 5(2), 109–119 (2012)
16. Romano, J.M., Yoshioka, T., Kuchenbecker, K.J.: Automatic filter design for synthesis of haptic textures from recorded acceleration data. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. pp. 1815–1821. IEEE (2010)
17. Shin, S., Osgouei, R., Kim, K.D., Choi, S.: Data-driven modeling of isotropic haptic textures using frequency-decomposed neural networks. In: World Haptics Conference (WHC), 2015 IEEE. pp. 131–138 (June 2015)
18. Vasudevan, H., Manivannan, M.: Recordable haptic textures. In: Haptic Audio Visual Environments and their Applications, 2006. HAVE 2006. IEEE International Workshop on. pp. 130–133. IEEE (2006)
19. Wall, S.S., Harwin, W.S.: Modelling of surface identifying characteristics using fourier series (2008)
20. Wright, S.J., Nowak, R.D., Figueiredo, M.A.: Sparse reconstruction by separable approximation. Signal Processing, IEEE Transactions on 57(7), 2479–2493 (2009)
21. Zhang, Z., Xu, Y., Yang, J., Li, X., Zhang, D.: A survey of sparse representation: algorithms and applications. Access, IEEE 3, 490–530 (2015)