HAVE 2007 - IEEE International Workshop on
Haptic Audio Visual Environments and their Applications
Ottawa, Canada, 12-14 October 2007

Haptic Texturing based on Real-World Samples

S. Andrews and J. Lang

SITE, University of Ottawa
Ottawa, Canada
E–mail: {$sandr071|jlang$}@site.uottawa.ca

Abstract – *Haptic texturing allows for small scale surface features to be represented in haptic rendering applications. Medical, entertainment, multimedia, and e-commerce applications all stand to benefit from the display of realistic haptic textures. Therefore, acquiring texture from real-world objects is of particular importance for the realistic modeling of objects in virtual environment and other hapto-visual applications. In previous work, we presented a cost-effective system for acquiring haptic surface properties (texture and compliance) using a mobile touch probe and visual tracker. In this paper, we discuss how textures generated by our system may be registered with object geometry and integrated as part of the standard 3D model acquisition pipeline.*

Keywords – *haptics, texture, measurement-based modeling*

I. INTRODUCTION

Haptic textures [1] can play a similar role to textures in graphics by bridging the gap between geometry and microscale point contacts. It has been shown that material details have a significant effect on the object identification [2], indicating that the realism of textured virtual objects is increased. Many application areas of haptic technology can profit from haptic textures including medicine (tele-remote surgery, surgical simulation), entertainment (video games, 3D painting and sculpting, haptic multimedia playback), engineering design (augmented CAD, virtual product assembly), and e-commerce. The haptic texture models used by some of these applications, e.g., medicine, should accurately purvey tactile and kinesthetic information to the user. Even in the video game industry there is a trend to couple highly detailed physical models with novel human-computer interfaces. Haptic textures and other tactile cues help to increase the realism of the game simulation, and thus increase the overall entertainment value of the game [3].

Estimation of physical properties based on real-world interaction behaviour can be difficult and cumbersome. In [4] we introduced a system for scanning and synthesizing haptic textures based on sensor data acquired by a tactile probe and a visual tracker (see Figure 1). We implied that textures generated by our system could be registered with 3D geometry, which is necessary in order to integrate our system with the standard 3D scanning pipeline [5]. In the current paper, we present a technique for registering scans from our haptic texturing system with a 3D geometry. We use an ICP-based algorithm to register scanning trajectories with object geometry and we introduce a novel technique for assigning texture coordinate values which may be used to texture sections of the object, based on scans from our system.

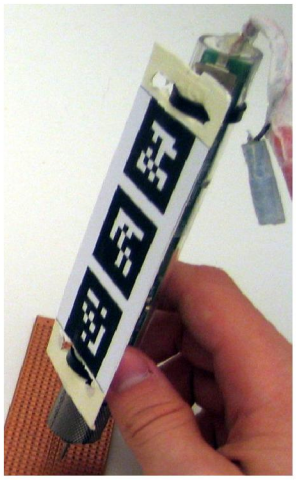

Fig. 1. Texture scanning with the *WHaT* probe

A. System Overview

The diagram in Figure 2 shows a flowchart for scanning haptic textures using our system. It illustrates the stages for generating a textured 3D object from raw sensor data. During acquisition, data is collected from sensors and stored for off-line processing. This stage also includes some on-line processing to remove noise and other errors from the sensor profiles. In the signal processing and filtering stage, sensor data obtained in the first step is conditioned in order to increase the quality of textures produced by our system. In the final stage, texture synthesis, we generate surface profiles and compliance estimates from filtered sensor data, and register a texture scan with the object geometry.

At the heart of our system is a tactile probe called the *WHaT* [6] which senses the small-scale motion and contact forces as the probe tip is moved across the surface of an object. The WHaT consists of a force sensor aligned with the major axis of the device, and a pair of bi-axial accelerometers which provide motion data about the probe tip in 3D space. Force and acceleration sensor data is streamed from the probe

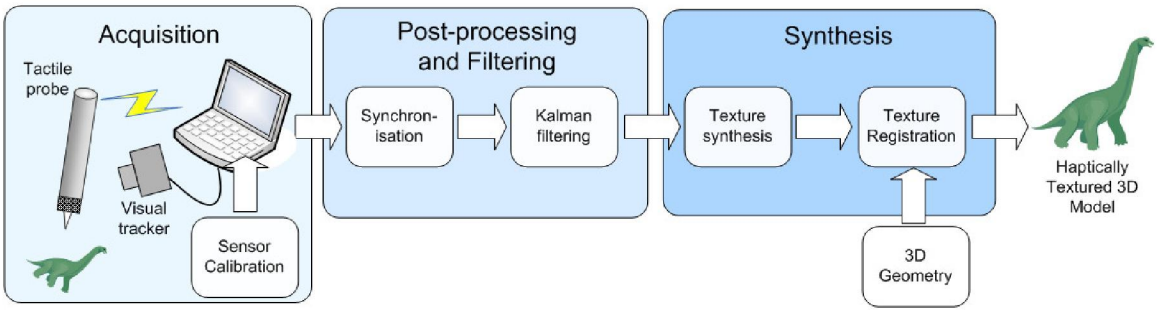

Fig. 2. Scanning flow chart.

to a host PC at a rate of approximately $500Hz$, providing a sampling rate capable of reconstructing most surface features. The accelerometers and the force sensor. have a resolution of approximately $0.015g$ and $0.02N$, respectively.

For estimation of large-scale motion, we affix a set of fiducial markers to the probe and track it using *ARTag* [7], an augmented reality system. ARTag locates the marker within images obtained from a digital camera and estimates the position and orientation of the probe relative to the camera, which is coincident with the global reference frame.

As described in [4], calibration and signal processing techniques were used to increase the quality of sensor data obtained from the probe and to give more accurate reconstruction of the probe's path. Most notably, a Kalman filter [8] allowed us to remove most of the noise from visual tracker measurements and the resulting scan trajectory is suitable for registration with 3D geometry, such as a polygonal mesh.

B. Outline

In Section II we review related work to our haptic texturing approach. We detail our technique for registering scans from our system with 3D object geometry in Section III. Section III also discusses our novel approach for assigning texture coordinates to elements in a polygonal mesh. Section IV describes our rendering technique. Results of this technique and an application are presented in Section V, and we conclude in Section VI.

II. RELATED WORK

Despite the availability of a 3D scanning pipeline [5], object scanning requires great user expertise due to the often fragile acquisition process. Typically, scanning equipment consists of a complex robotic environment that is expensive and difficult to operate. Such facilities are inflexible due to workspace limitations, robotic control, and immobility. Hand-held systems can be far more flexible because of the superior control a human can exert on the probe and their ability to adapt to most environments; a *human-in-the-loop* system [9] is able to adapt to both the object and the workspace.

Pai et al. [10], [11] introduced a general robotic system for scanning physical interaction behaviour. Lloyd and Pai [12] also demonstrated an interactive scanning interface to this robot for texture acquisition. However, only visual texture was captured using their system, and a frictional coefficient was used to model the tactile properties of an object. Their approach does not capture many of the salient features which help to identify a material (e.g., roughness, patterned texture).

Other approaches measure surface features, such as texture, but without registering them to object geometry. Okamura et al. [13] augmented a haptic device with an accelerometer in order to acquire parameters for basis functions for several styles of interactions– tapping, scraping, and puncture. Wall and Harwin [14] used a linear variable differential transformer to measure displacement of a probe as it is moved (at near constant speed) over a surface. Their system generates plausible profiles, but a fixed assembly is required (linear track and motors) and scanning is limited to planar surfaces.

The modeling and rendering of texture is an area which is receiving increased attention in the field of haptics. Much of the work has been to create perceptually plausible and distinct textures which mimic the roughness perceived when an object it explored either by direct human contact or through some intermediary object A pioneering attempt to render haptic texture was that by Minsky [15]. She simulated textures on a 2 DOF haptic interface using lateral forces, which were calculated from the gradient of image-based depth maps. Fritz and Barner [16] used a stochastic approach to render texture wherein pseudo-random texture vectors were generated across a uniformly spaced 2D lattice and textures were modeled by combining Gaussian noise with a deterministic texture function. Siira and Pai [17] also generated stochastically-based textures, wherein parameters for texture synthesis were sampled from a Gaussian distribution. Their approach was capable of creating a dynamic range of realistic textures using only a few parameters. Wall and Harwin [14] developed a procedural model which reconstructs the height-displacement profile of a surface using the discrete Fourier series coefficients. Basdogan et al. [18] used an image-based approach for haptic texturing, wherein a surface height field is calculated from grayscale image data using static, procedural, and fractal synthesis techniques. Unlike Minsky's image-based rendering [1], their rendering algorithm uses the image gradient to perturb the surface normals of the textured object being displayed.

III. TEXTURE PAINTING

In this section we present our method for registering scans from our haptic texturing system with 3D geometry. We have nicknamed this process *texture painting*, since it is analogous to painting or drawing texture over sections of the 3D object. One advantage of our method is that the scanning trajectory may be real or virtual. That is, users may manually specify a path over the surface of a 3D object (e.g., using a modeling tool) in order to determine which sections are textured. We show how 1D and 2D texture coordinates may be automatically generated from the scanning trajectory and a polygonal mesh. Texture painting involves two steps: (i) registering a scanning trajectory with 3D geometry, and (ii) generating texture coordinates for sections on the surface of a 3D object. If a synthetic scanning trajectory is being used (e.g., hand drawn using a modeling tool), the first step is redundant since the path is already registered with the object's surface.

A. Registration

Registration of textures involves transforming the scanning trajectory from one or more scans into a single coordinate system– usually the coordinate frame used by a polygonal mesh. The scanning trajectory consists of points sampled by the visual tracker, indicating the position of the probe tip in 3D space at a given time. These are matched with points from the corresponding object geometry, thus defining a path over the object's surface. We obtain geometry for real-world objects using a Konica Minolta VIVID 910 3D Digitizer and create a polygonal mesh from a series of point clouds with Geomagic Studio.

We use an iterative closest point (ICP) algorithm [19] to match points from the scanning trajectory to points on the surface defined by a polygonal mesh. ICP estimates the transformation of a path obtained from the visual tracker to a path over the surface of a 3D mesh. The algorithm works using an *associate-and-update* cycle, whereby points from the scanning trajectory are associated with points on the surface of the mesh and an optimal homogeneous transformation matrix is calculated for the two sets of points. Pseudo-code for the algorithm is given in Figure 3.

while $\Delta error > threshold$ **do**
  project scan path points onto mesh geometry
  estimate transform using minimum cost function
  update points using the estimated transform
  $error = |points_0 - points_1|^2$;
end while

Fig. 3. Pseudo-code for the ICP algorithm.

The ICP algorithm requires a good initial estimate of the relative transformation (rotation and translation) from the scan path coordinate frame to the mesh coordinate frame. This step is done by hand using a software application, shown in Figure 4. Aside from this step, no other interaction is required on the part of the user. The scanning trajectory is manually

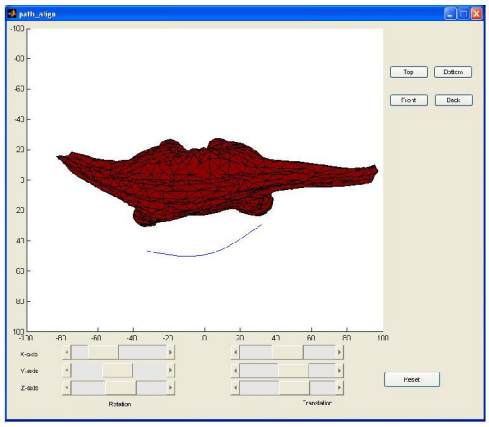

Fig. 4. Manual alignment of the scan path in order to obtain an initial estimate for the homogeneous transformation matrix. Users may specify six translational and rotational parameters.

aligned by adjusting the three translational values along the axes and three rotation angles (α, β, γ) around the fixed coordinate axes. The resulting alignment is displayed to the user from a top, bottom, front, or back perspective.

A known problem in using ICP to register sets of points is that the convergence and uniqueness of the solution is highly dependent on the initial transform provided by the user. That is, the algorithm is sensitive to the "guess" from the manual alignment step. For meshes without much detail, the ICP algorithm may immediately converge on a solution which has not been optimized from the manual alignment provided by the user. Future work may introduce variants of the ICP algorithm (e.g., [20]) to make the registration process less sensitive to mesh geometry and errors in the manual alignment.

The scan path is projected onto the 3D mesh using a minimum distance criteria. The minimum distance between the mesh and points in the scanning trajectory are found using barycentric coordinates and stored in an array. Next, a least sum of squares algorithm is used to optimize a 4-by-4 homogeneous transformation matrix which maps the scanning trajectory onto the mesh. The ICP algorithm performs optimization directly on the homogeneous transformation matrix and not the six parameters used for manual alignment. We set the eigenvalues of the rotation matrix to unity in order to arrive at a proper homogeneous transformation matrix as suggested by Arun et al. [21].

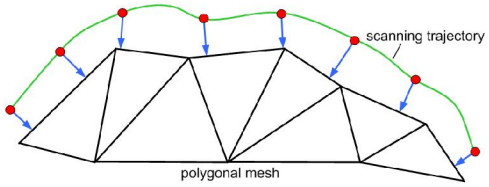

Fig. 5. Association of mesh points with the scan path. Points from the scanning trajectory are projected onto the mesh using a minimum distance criteria.

The algorithm given in Figure 3 repeats until there is no noticeable change in the computed error value between iterations.

The error is calculated using the average distance from points in the scan path to their associated mesh points. Figure 5 illustrates how the scanning trajectory is projected onto the surface mesh for each iteration of the ICP algorithm.

B. Coordinates

Texture coordinates may be assigned to elements of a polygonal mesh from the path arising from registration of the scanning trajectory. To understand how texture coordinates are generated, consider a 2D texture function, $f(x, y)$ which represents the height profile of a scanned surface. The function uses a coordinate pair x, y (or a single coordinate for 1D textures) to generate a height value which corresponds to the texture height of a point on an object's surface. Texture coordinates are assigned to each vertex of a polygon using the position relative to the path of associated mesh points found during the registration stage.

Polygons are selected for texturing by testing if a point in the associate scan path intersects the polygon boundaries. This has already been done in the distance minimization step during registration, since polygons are selected when a point on their surface contains the minimum distance to a point in the scan path. The polygons, vertices, and barycentric coordinates at minimum distance are obtained from the final iteration of the ICP algorithm.

In Figure 6, relevant polygons have been shaded and texture coordinates will be generated for each vertex of polygons in this set. The set may be extended to neighbouring polygons by using a breadth-first search of the mesh, since our algorithm for generating texture coordinates does not require that polygons intersect with the original scan path or associated mesh points.

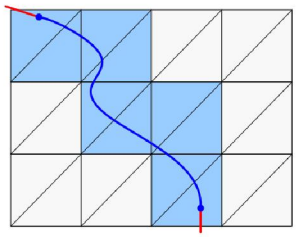

Fig. 6. Scan path superimposed on a polygonal mesh. Relevant polygons have been shaded

Texture coordinates are generated for each vertex by calculating the minimum distance to the associated mesh points. We assign the signed distance *along* the path to parameter x and the signed distance *to* the scan path to parameter y to create the texture function $f(x, y)$ This scheme of assigning texture coordinates is better understood by considering Figure 7. The y parameter value is obtained by finding the minimum distance from a vertex to a point on the path; the x parameter value is obtained by finding distance along the scan path where the minimum distance to the vertex is located.

Note that it is possible that spatial warping may occur in textures using the above technique. For example, if there is "pinching" or overlapping in the scanning trajectory, the texture coordinates may distort the texture at certain areas. Best

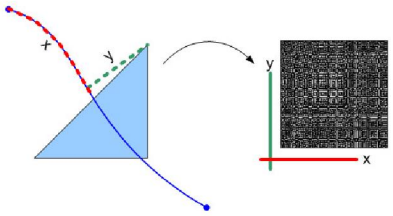

Fig. 7. Generating texture coordinates from a scan path. For 2D textures, the x and y value are generated. For 1D textures, only the x value is generated.

results are achieved when well-formed scan paths are used to acquire and register a texture (e.g., no looping or "zig-zag" motion). Also, provided the surface profile of a texture is continuous at its boundaries, it will feel seamless when a user explores a textured object. One possible strategy here is to use a post-processing filter which reshapes the surface profiles to be C^1 continuous at their edges. Obviously, crossing between two differently textured sections of a mesh will result in noticeable changes at the boundaries.

IV. HAPTIC RENDERING

We model textures using a Fourier series reconstruction of their surface profile [4], similar to the model used by Wall and Harwin [14]. Coefficients of the sinusoidal basis functions are obtained by sampling the discrete Fourier transform (DFT) of each surface profile. We extract $N = 128$ coefficients from the DFT by sampling a surface profile of length L, which is synthesized by our haptic texturing system. The resulting texture function $f(x)$ is defined as:

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

where x is a spatial coordinate, ω is the fundamental frequency of the texture sample calculated as $\omega = 2\pi L/N$ and a and b are the real and imaginary DFT coefficients, respectively. For 1D textures, a and b are 1-by-N vectors; for 2D textures, they are N-by-N matrices which are built by cross-correlation of their 1D counterparts in the spatial domain.

Force decomposition is done similarly to that by Siira and Pai [17] where the force vector $\vec{f}$ due to contact and texture interaction is calculated as:

$$\vec{f} = \vec{f_c} + \vec{f_t} + \vec{f_f}. \quad (1)$$

The unified force equation (1) is composed of several other forces, mainly: due to rigid body constraints (f_c), due to interaction with texture (f_t), and frictional forces (f_f). Components f_c and f_t are normal forces which are rendered using a penalty-based method using spring forces generated proportional to the penetration depth of a haptic proxy into a 3D object. Component f_f is a lateral frictional force which is proportional to $(f_c + f_t)$. Individual force components are calculated by the set of equations:

$$\vec{f_c} = -k_s\ \Delta x \vec{n}, \tag{2}$$

$$\vec{f_t} = -k_c\ k_s\ \Delta h \vec{n}, \tag{3}$$

$$\vec{f_f} = \mu_s\ |\vec{f_c} + \vec{f_t}|\vec{t} \tag{4}$$

where Δx is the penetration depth of the proxy into the planar surface, Δh is the penetration depth into the texture profile, and μ_s is a preselected frictional coefficient. $\vec{n}$ and $\vec{t}$ are the normal and tangential vectors, respectively, to a point on the surface of the 3D object where the haptic proxy makes contact. Figure 8 shows a visual depiction of the forces and rendering parameters of Equations (2),(3), and (4)

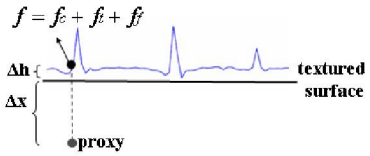

Fig. 8. Haptic rendering force model. The texture surface profile is shown superimposed onto a planar surface. Contact, texture, and frictional forces are displayed at the haptic interface.

The force rendering algorithm needs to determine the texture coordinates of the haptic proxy point with respect to the surface mesh. Exact values of the texture coordinates at a collision point are calculated by interpolation using the barycentric coordinates for the location of the haptic proxy within the boundaries of the colliding polygon, and are calculated as:

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

where (a,b,c) are the barycentric coordinates of the haptic proxy determined by collision with the polygon, $(x,y)_{proxy}$ is the texture coordinate at the haptic proxy, and $(x,y)_{v_1}$ to $(x,y)_{v_3}$ are the texture coordinate values assigned to each vertex v_1 to v_3 of the polygon (e.g, as in Figure 10). Given that a collision only occurs inside the boundaries of a polygon, we know that $a,b,c \in [0,1]$ and $a+b+c=1$, thus giving a texture coordinate pair within a range determined by the texture coordinate values at the vertices.

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

V. RESULTS

Textures generated from scans of a toy dinosaur were registered with a 3D polygonal mesh and displayed in a hapto-visual application. Figure 9 shows two scanning trajectories registered with the mesh. Paths over the mesh surface are used to select which polygons are textured and is extended to the surrounding 1-neighbourhood polygons.

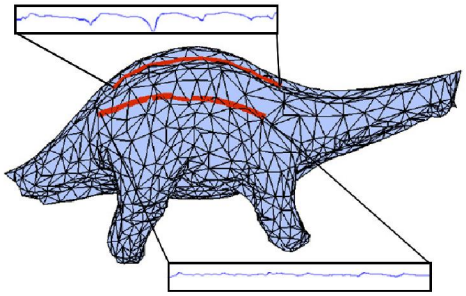

Fig. 9. Scan registration using the geometry of a toy dinosaur.

We store object geometry and texture coordinates together using an augmented 3D file format file. Figure 10 shows an example of a file for a 3D object which has been textured using our method. Only haptic texture coordinates are included with the geometry, and texture data is stored in a separate file. For polygonal meshes, texture coordinates are assigned per face and coordinate values are denoted by a "h" appearing as the first character on a line. This identifying character is followed by a texture index and three texture coordinate pairs, which represent the 2D coordinates used to generate a height value from the texture function. The index is used to identify which texture should be used when more than one texture has been applied to an object. An index value of -1 indicates that the face is not textured.

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

Fig. 10. Example of an augmented Wavefront .obj file which contains geometry and texture information. The file shown here is for a 80x80 plane in 3D textured with a 50x50 2D texture.

We implemented a C++ hapto-visual application on a 2.4GHz Pentium PC with an attached Phantom™Omni, a device capable of 6 DOF input and 3 DOF force display. The test environment generated by our application allows users to load and explore textured 3D objects. Figure 11 shows this application as it displays the textured 3D dinosaur. Sections of the mesh which have been textured are highlighted and, when the user brings the proxy in contact with these areas of the mesh, the texture is displayed according to our haptic rendering algorithm.

The hapto-visual application is able to load and display any polygonal mesh using the augmented 3D file format. Textures are loaded separately, by specifying a file which contains the

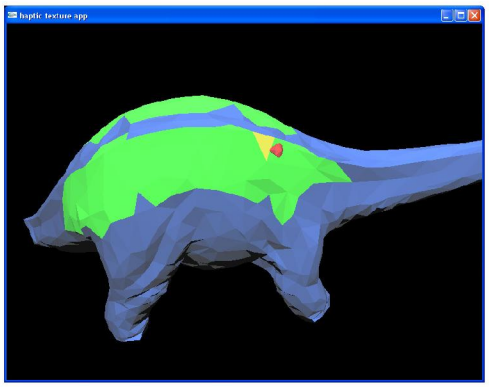

Fig. 11. Rendering a textured 3D model. Areas of the mesh are textured using scan registration, and are highlighted.

Fourier coefficients of the sampled surface profile, fundamental spatial frequency, and compliance estimate. An important aspect of our system is that texture mapping occurs independently from the texture data. This allows us to use our texture with a variety of artistically generated and mechanically obtained 3D meshes, and to swap textures among meshes.

If no texture data is available, or if untextured areas of a mesh are being rendered, only contact forces due to rigid-body constraints are rendered.

VI. CONCLUSIONS

We have shown how textures generated using our haptic texturing system may be registered with 3D object geometry, thus integrating our system into the standard 3D scanning pipeline. A novel technique was introduced for assigning texture coordinates from a scanning trajectory. Texture coordinates were assigned to faces of a polygonal mesh and stored using an augmented 3D file format. Virtual objects were rendered in a hapto-visual application which displayed both visual and tactile aspects of the objects to user.

A texture may be transferred from one object to another by specifying a virtual scan path across the mesh. Many 3D modeling packages allow a user to paint texture and colour directly onto the mesh surface. This feature could be extended to also allow drawing a scanning trajectory. Future work may involve integrating our texturing technique as part of a modeling software, such as Blender.

ACKNOWLEDGEMENTS

We would like to thank D.K. Pai and P. Rizun for the loan of the WHaT and we gratefully acknowledge the financial support from the Ontario Research Network for Electronic Commerce (ORNEC), Point Grey Research Inc., and from the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

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

[2] R.L. Klatzky, J.M. Loomis, S.J. Lederman, H. Wake, and N. Fujita, "Haptic identification of objects and their depictions," *Perception & Psychophysics*, vol. 54, no. 2, pp. 170–178, 1993.

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

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

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

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

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

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

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

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

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

[12] J.E. Lloyd and D.K. Pai, "Interactive exploration of remote objects using a haptic-VR interface," in *Proc. of the International Symposium on Experimental Robotics*, Italy, July 2002, vol. 5 of *Springer Tracts on Advanced Robotics*, pp. 560–569.

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

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

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

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

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

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

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

[20] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[21] K.S. Arun, T.S. Huang, and S.D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.