Standardized Evaluation of Haptic Rendering Systems

Emanuele Ruffaldi[1], Dan Morris[2], Timothy Edmunds[3], Federico Barbagli[2], Dinesh K.Pai[3]
[1]*PERCRO, Scuola Superiore S. Anna*
[2]*Computer Science Department, Stanford University*
[3]*Computer Science Department, Rutgers University*
pit@sssup.it, {dmorris,barbagli}@robotics.stanford.edu, {tedmunds,dpai}@cs.rutgers.edu

ABSTRACT

The development and evaluation of haptic rendering algorithms presents two unique challenges. Firstly, the haptic information channel is fundamentally bidirectional, so the output of a haptic environment is fundamentally dependent on user input, which is difficult to reliably reproduce. Additionally, it is difficult to compare haptic results to real-world, "gold standard" results, since such a comparison requires applying identical inputs to real and virtual objects and measuring the resulting forces, which requires hardware that is not widely available. We have addressed these challenges by building and releasing several sets of position and force information, collected by physically scanning a set of real-world objects, along with virtual models of those objects. We demonstrate novel applications of this data set for the development, debugging, optimization, evaluation, and comparison of haptic rendering algorithms.

CR Categories: H.5.2 [User Interfaces]: Haptic I/O

Keywords: haptics, ground truth, evaluation

1. INTRODUCTION AND RELATED WORK

Haptic rendering systems are increasingly oriented toward representing realistic interactions with the physical world. Particularly for simulation and training applications, intended to develop mechanical skills that will ultimately be applied in the real world, fidelity and realism are crucial.

A parallel trend in haptics is the increasing availability of general-purpose haptic rendering libraries [1,2,3], providing core rendering algorithms that can be re-used for numerous applications. Given these two trends, developers and users would benefit significantly from standard verification and validation of haptic rendering algorithms.

In other fields, published results often "speak for themselves" – the correctness of mathematical systems or the realism of images can be validated by reviewers and peers. Haptics presents a unique challenge in that the vast majority of results are fundamentally interactive, preventing consistent repeatability of results. Furthermore, it is difficult at present to distribute haptic systems with publications, although several projects have attempted to provide deployable haptic presentation systems [1,4].

Despite the need for algorithm validation and the lack of available approaches to validation, little work has been done in providing a general-purpose system for validating the physical fidelity of haptic rendering systems. Kirkpatrick and Douglas [5] present a taxonomy of haptic interactions and propose the evaluation of complete haptic systems based on these interaction modes, and Guerraz et al [6] propose the use of physical data collected from a haptic device to evaluate a user's behavior and the suitability of a device for a particular task. Neither of these projects addresses realism or algorithm validation. Raymaekers et al [7] describe an objective system for comparing haptic algorithms, but do not correlate their results to real-world data and thus do not address realism. Hayward and Astley [8] present standard metrics for evaluating and comparing haptic devices, but address only the physical devices and do not discuss the software components of haptic rendering systems. Similarly, Colgate and Brown [9] present an impedance-based metric for evaluating haptic devices. Numerous projects (e.g. [10,11]) have evaluated the efficacy of specific haptic systems for particular motor training tasks, but do not provide general-purpose metrics and do not address realism of specific algorithms. Along the same lines, Lawrence et al [12] present a perception-based metric for evaluating the maximum stiffness that can be rendered by a haptic system.

This paper addresses the need for objective, deterministic haptic algorithm verification and comparison by presenting a publicly available data set that provides forces collected from physical scans of real objects, along with polygonal models of those objects, and several analyses that compare and/or assess haptic rendering systems. We present several applications of this data repository and these analysis techniques:

- *Evaluation of rendering realism:* comparing the forces generated from a physical data set with the forces generated by a haptic rendering algorithm allows an evaluation of the physical fidelity of the algorithm.

Symposium on Haptic Interfaces for
Virtual Environment and Teleoperator Systems 2006
March 25 - 26, Alexandria, Virginia, USA
1-4244-0226-3/06/$20.00 ©2006 IEEE

- *Comparison of haptic algorithms:* Running identical inputs through multiple rendering algorithms allows identification of the numeric strengths and weaknesses of each.

- *Debugging of haptic algorithms:* identifying specific geometric cases in which a haptic rendering technique diverges from the correct results allows the isolation of implementation bugs or scenarios not handled by a particular approach, independent of overall accuracy.

- *Performance evaluation:* Comparing the computation time required for the processing of a standard set of inputs allows objective comparison of the performance of specific implementations of haptic rendering algorithms.

The data and analyses presented here assume an impedance-based haptic rendering system and a single point of contact between the haptic probe and the object of interested. This work thus does not attempt to address the full range of possible contact types or probe shapes. Similarly, this work does not attempt to validate the realism of an entire haptic rendering pipeline, which would require a consideration of device and user behavior and perceptual psychophysics. Rather, we present a data set and several analyses that apply to a large (but not universal) class of haptic rendering systems. We leave the extension of this approach to a wider variety of inputs and to more sophisticated metrics as future work.

The remainder of this paper is structured as follows: Section 2 will describe our system for physical data acquisition, Section 3 will describe the process by which we simulate a contact trajectory for evaluation of a haptic rendering algorithm, Section 4 will describe some example results we have obtained through this process, and Section 5 will discuss the limitations of our method and several scenarios in which our data and methods may be useful to others in the haptics community. We conclude with a description of our public data repository and a discussion of future extensions to this work.

2. DATA ACQUISITION

Haptic rendering algorithms typically have two sources of input: a geometric model of an object of interest and real-time positional data collected from a haptic interface. The output of this class of algorithms is typically a stream of forces that is supplied to a haptic interface. A key goal of our data and analyses is to compare this class of algorithms to real-world data, which requires: (a) collecting or creating a geometric model of a real-world object and (b) collecting a series of correlated forces and positions on the surface of that object.

Figure 1. The sensor used to acquire force and torque information, alongside a coin to indicate scale.

We have constructed a sensor apparatus that allows the collection of this data. Our specific goal is to acquire data for haptic interaction with realistic objects using a hand-held stylus or pen-like device (henceforth called "the probe"). We use the HAVEN, an integrated multisensory measurement and display environment at Rutgers, for acquiring measurements interactively, with a human in the loop.

In previous work [13,14], we acquired such measurements using a robotic system called ACME (the UBC Active Measurement facility). This robotic approach has many advantages, including the ability to acquire repeatable and repetitive measurements for a long period of time, and the ability to acquire measurements from remote locations on the Internet. However, our current goals are different, and a hand-held probe offers a different set of advantages that are important for evaluating interaction with a haptic device.

First, it measures how a real probe behaves during natural human interaction, and therefore provides more meaningful data for comparison. This is important, because contact forces depend in part on the passive, task-dependent impedance of the hand holding the probe, which is difficult to measure or to emulate with a robot arm. Second, the dexterity of robot manipulators available today is very poor in comparison with the human hand. Furthermore, acquiring measurements in concave regions or near obstacles using a robot is very difficult, but is easy for a human.

We acquired three types of measurements for each object in our data repository:

1. The object's 3D shape
2. Motion of the probe tip relative to the object
3. The force on the probe tip during contact

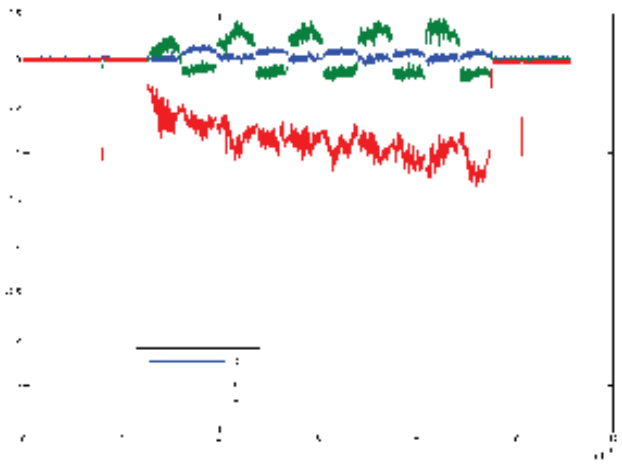

Figure 2. Data collected from our scanning apparatus. Normal (z) forces are indicated in red, tangential (x,y) forces are indicated in green and blue. The data presented here represent a scanning motion, primarily on the y axis, on a flat plane. Brief initial and final taps were added to aid registration of force and motion data; they are visible in the normal force.

We describe these measurements in the remainder of this section, in reverse order.

Force data are acquired using a custom-designed hand-held probe built around a Nano17 6-axis force/torque sensor (Figure 1) (ATI Industrial Automation, Apex, NC, USA). The reported spatial resolution of the force sensor is as follows (the z-axis is aligned with the axis of the probe): Fx,Fy 1/320 N; Fz 1/640 N; Tx,Ty 1/128 N·mm; Tz 1/128 N·mm.

A replaceable sphere-tipped Coordinate Measuring Machine (CMM) stylus is attached to the front face of the force sensor, and a handle to the rear, allowing a user to drag the probe tip over the surface being measured. The interchangability of the probe tip is important, since the curvature of the contact area kinematically filters the probe motion and thus impacts the acquired data.

As the surface is being probed, the force/torque measurements from the Nano17 are sampled at 5kHz using a 16-bit A/D converter (National Instruments, Austin, Texas, USA). The static gravitational load due to the probe tip is compensated for based on the measured orientation of the probe. The force and torque measured at the force sensor are transformed to the center of the probe tip to compute the contact force on the tip.

In addition to measuring force and torque, the probe's motion is tracked to provide simultaneous position data. The probe is tracked using a six-camera motion-capture system (Vicon Peak, Lake Forest, CA, USA). Several small retroreflective optical markers are attached to the probe, allowing the camera system to record and reconstruct the probe's position and orientation at 60Hz. The reconstructed position is accurate to less than 0.5mm.

The object being measured is also augmented with optical tracking markers, so the configuration of the probe

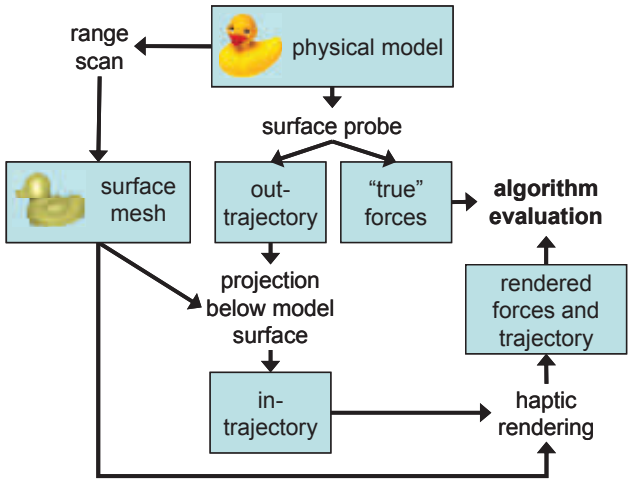

Figure 3. An overview of our data processing and algorithm evaluation pipeline. An object is scanned, producing a 3D geometric model and an out-trajectory. An in-trajectory is synthesized from this out-trajectory and is fed as input to a haptic rendering system, which produces force and trajectory information. This information can be compared to the physically-scanned forces and the original trajectory.

with respect to the object is known even when the user moves the object to access different locations on the surface. The object is scanned with a Polhemus FastScan laser scanner (Polhemus, Colchester, VT, USA) to generate a mesh representation of the object's surface. The manufacturer reports an accuracy of 1mm for the surface. A water-tight triangular mesh is extracted from the scans using a fast RBF method. The location of the optical tracking markers are included in the scan to allow registration of the surface geometry with the motion capture data acquired during contact measurement. Figure 2 shows an example data series acquired with our setup. The full data set is available in the public repository (see Section 7).

Our initial scanning effort has focused on rigid objects, to constrain the analysis to static geometry.

3. Data Processing

Given a set of scanned trajectories, we evaluate a haptic rendering algorithm by feeding a sequence of scanned probe positions into the algorithm and comparing the computed forces to the physically-scanned forces. For penalty-based haptic rendering algorithms, this requires a pre-processing step to create a virtual trajectory that is inside the virtual representation of the scanned object.

This section will describe this process, which can be summarized in three stages:

1. Pre-processing of a scanned trajectory to allow direct comparison to rendered trajectories.
2. Computation of rendered forces and a surface contact

point trajectory by the haptic rendering algorithm that is being evaluated, using the pre-processed input positions.

3. Computation of performance metrics from the output of the haptic rendering system.

Figure 3 summarizes this process.

3.1 Data pre-processing

The haptic rendering algorithms on which we have performed initial analyses are penalty-based: the virtual haptic probe is allowed to penetrate the surface of a simulated object, and a force is applied to expel the haptic probe from the object. A physical (real-world) probe scanning the surface of a physical object never penetrates the surface of the object. Therefore a virtual scanning trajectory is not expected to be identical to a physical trajectory, even if a user intends to perform the same probe motions on the real and virtual objects. We therefore perform a pre-processing step that – given a physical scanning trajectory – generates a sub-surface trajectory that (under ideal conditions) produces a surface contact trajectory that is equivalent to the scanned trajectory. This allows a direct comparison of a trajectory collected from a haptic simulation with the ideal behavior that should be expected from that simulation.

We refer to an ideal trajectory (one in which the probe never penetrates the surface of the object) as an "out-trajectory", and a trajectory that allows the probe to travel inside the object as an "in-trajectory". Figure 4 demonstrates this distinction.

The penetration depth (the distance between the in- and out-trajectories) of a virtual haptic probe into a surface is generally dependent on an adjustable spring constant, which is an input to the algorithm and should be considered part of the system that is under evaluation; this constant is reported along with all results in our online repository. The spring constant is assumed to be homogeneous for purposes of the present analysis.

Typically, penetration depth and the resulting penalty force are related to this spring constant according to Hooke's Law:

$$\mathbf{f_p} = -k\mathbf{x} \qquad (1)$$

Here $\mathbf{f_p}$ is the penalty force vector, k is the scalar stiffness constant, and $\mathbf{x}$ is the penetration vector (the vector between the haptic probe position and a surface contact point computed by the haptic rendering algorithm). We use this relationship to compute a corresponding in-trajectory for a physically-scanned out-trajectory.

Surface normals are computed at each point in the out-trajectory, using the scanned geometric model of the object. These surface normals are then used to extract the normal component of the recorded force at each point. Each point

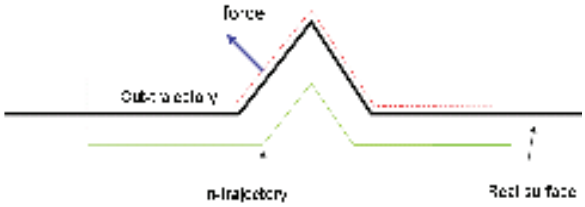

Figure 4. An "out-trajectory" represents the path taken by a physical probe over the surface of an object; a haptic rendering algorithm typically approximates this trajectory with an "in-trajectory" that allows the probe to enter the virtual object.

in the sampled out-trajectory is then converted to a corresponding point in the in-trajectory by projecting the surface point into the object along the surface normal, by a distance inversely proportional to the chosen stiffness and directly proportional to the recorded normal force (for a given normal force, higher stiffnesses should result in lower penetration depths):

$$\mathbf{p_{in}} = \mathbf{p_{out}} - \mathbf{F_n} / k \qquad (2)$$

Here $\mathbf{p_{in}}$ and $\mathbf{p_{out}}$ are corresponding in- and out-trajectory points, $\mathbf{F_n}$ is the recorded normal force at each point, and k is the selected stiffness constant. This relationship is illustrated in Figure 5. Each in-trajectory point is assigned a timestamp that is equal to the corresponding out-trajectory point's timestamp.

Following this computation, the in-trajectory corresponding to a physical out-trajectory is the path that a haptic probe would need to take in a virtual environment so that the *surface contact point* corresponding to that haptic probe path precisely follows the sampled out-trajectory.

3.2 Trajectory processing

The input to a haptic rendering algorithm is typically a geometric model of an object of interest and a series of positions obtained from a haptic interface. For the present analysis, we obtain a geometric model from the laser-scanning system described in Section 1, and we present a stream of positions – collected from our position-tracking system – through a "virtual haptic interface". From the perspective of a rendering algorithm implementation, this interface plays the role of a haptic device that is able to report its position in Cartesian space.

Given an in-trajectory computed from a physical out-trajectory, we can thus simulate a virtual haptic interaction with an object, which will produce a stream of forces and – in the case of many common haptic rendering algorithms – a new out-trajectory (which we refer to as a "rendered trajectory"), representing the path that a virtual contact point traveled on the surface of the virtual object.

The computational complexity of this simulation is identical to the case in which a haptic interface is used

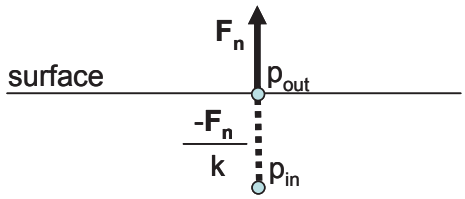

Figure 5. Computation of an in-trajectory point from a sampled out-trajectory point.

interactively, allowing assessment of computational performance in addition to algorithm output.

3.3 Metric extraction

Each time an in-trajectory is fed through a haptic rendering algorithm, producing a stream of forces and surface contact point locations, we collect the following evaluation metrics:

- *Output force error:* the difference between the forces produced by the haptic rendering algorithm and the forces collected by the force sensor. This is summarized as a root-mean-squared Euclidean distance, i.e.:

$$e = \frac{1}{N}\sum_{i=1}^{N}\left\|\vec{F}p_i - \vec{F}r_i\right\| \qquad (3)$$

  Here N is the number of samples in the out-trajectory, $\mathbf{Fp_i}$ is the physically-scanned force at sample *i* and $\mathbf{Fr_i}$ is the rendered force at sample *i*. This metric is referred to as "RMS Force Error" in Section 4. The physically-scanned forces have been resampled to align in time with the position samples.

- *Output position error:* the difference between the surface contact point position produced by the haptic rendering algorithm and the physically sampled out-trajectory. This can also be summarized as a root-mean-squared Euclidean distance, although we have found that it is more valuable to collect the cases that exceed a threshold instantaneous error, representing "problematic" geometric cases.

- *Computational cost:* the mean, median, and maximum numbers of floating-point operations required to a compute a surface contact point and/or penalty force and the floating-point operation count for the complete trajectory. While this is not a truly platform-independent measure of computational complexity, it scales well among CPU speeds and is roughly proportional to computation time on a particular CPU.

We do not present these metrics as a comprehensive representation of haptic rendering performance, rather we present them as examples of immediately-useful data that can be extracted using our data collection system, data repository, and offline processing approach. We anticipate that future work and future contributions by the haptics community will expand the set of available metrics and assess their correlations to the perceptual quality of haptic environments.

4. Experiments and Results

We used the analyses discussed in Section 3 to conduct four experiments that attempt to quantify and compare haptic rendering algorithms. Specifically, we explored:

1. The relative accuracy and computational cost of a haptic proxy algorithm and a rendering scheme based on voxel sampling.
2. The impact of simulated friction on the accuracy of haptic rendering and the use of ground truth data for friction identification.
3. The impact of mesh resolution on the accuracy of haptic rendering.
4. The impact of force shading on the accuracy of haptic rendering.

For consistency, these analyses have all been performed using the same model (a scanned plastic duck) and input trajectory (see Figure 6), which is available in the online repository.

These results are presented as examples of analyses that can be derived from our data sets, and their generalization to a wider variety of rendering algorithms, models, and trajectories is left for future work and is the primary goal of our online repository.

4.1 Proxy-based vs. voxel-based rendering

Our approach was used to compare the computational cost and force errors for a public-domain implementation [1] of the haptic proxy (god-object) algorithm [15] and a voxel-based rendering scheme [16], and to assess the impact of voxel resolution on rendering accuracy. This analysis does not include any cases in which the proxy provides geometric correctness that the voxel-based rendering could not; i.e. the virtual haptic probe never "pops through" the model.

Figure 6. The model and scanned trajectory used for the experiments presented in section 4.

Algorithm	Voxel resolution	RMS force error (N)	Floating-point ops	Init time (s)	Memory (MB)
voxel	32^3	.136	484K	0.27	1.0
voxel	64^3	.130	486K	2.15	8.0
proxy	N/A	.129	10.38M	0.00	0.0

Table 1. Accuracy and cost of haptic rendering using proxy- and voxel-based rendering schemes.

Voxel-based rendering was performed by creating a fixed voxel grid and computing the nearest triangle to each voxel center. The stored triangle positions and surface normals are used to render forces for each voxel through which the probe passes.

Results for the proxy algorithm and for the voxel-based algorithm (at two resolutions) are summarized in Table 1, including the computational cost in floating-point operations, the initialization time in seconds (on a 1.5GHz Pentium), and the memory overhead. We observe that the voxel-based approach offers comparable force error and a significant reduction in floating-point computation, at the cost of significant preprocessing time and memory overhead, relative to the proxy (god-object) approach. It should be noted that analysis of this particular trajectory does not capture the fact that the proxy-based approach offers *geometric* correctness in many cases where the voxel-based approach would break down. We will discuss this further in section 5.

4.2 Friction identification and evaluation

Our approach was used to evaluate the impact of simulated friction on the accuracy of haptic rendering, using a public-domain implementation [1] of the friction-cone algorithm [17]. This analysis also demonstrates the applicability of our approach for identifying rendering parameters – in this case a friction radius – from ground-truth data.

This analysis uses the friction cone algorithm available in CHAI 3D (version 1.31). The in-trajectory derived from the physical-scanned (raw) trajectory is fed to CHAI for rendering, and the resulting forces are compared to the physically-scanned forces. The coefficient of dynamic friction is iteratively adjusted until a minimum error between the physical and rendered forces is achieved. Static (stick-slip) friction was not considered for this analysis.

Results for the no-friction and optimized-friction cases are presented in Table 2, including the relative computational cost in floating-point operations. We observe that the trajectory computed with friction enabled contains significantly lower force-vector-error than the no-friction trajectory, indicating a more realistic rendering, with only a slightly higher computational cost.

Friction radius (mm)	RMS force error (N)	Flops
0.0000 (disabled)	0.132	10.4M
0.3008	0.067	10.8M

Table 2. Rendering accuracy with and without simulated dynamic friction.

4.3 Impact of mesh resolution

Our approach was used to assess the impact of varying mesh resolution on the accuracy of haptic rendering. This is a potentially valuable application of our data, since mesh resolution is often varied to trade off performance for accuracy for specific applications, and the use of ground truth data will allow application developers to select minimal models that meet application-specific accuracy bounds.

The haptic proxy algorithm was provided with an in-trajectory and with eight versions of the duck model, each at a different tessellation level. The results for each resolution are presented in Table 3 and Figure 7. We observe that the error is fairly stable for a large range of resolutions between 1000 and 140000 triangles, and increases sharply for lower resolutions.

4.4 Impact of force shading

The analysis presented in Section 4.3 was repeated with force shading [18] enabled, to quantify the impact of force shading on the accuracy of rendering this trajectory. Force shading uses interpolated surface normals to determine the direction of feedback within a surface primitive, and is the haptic equivalent of Gouraud shading.

Results are presented in Figure 7, along with the results assessing the impact of model size on rendering accuracy. We observe that for a large range of model sizes – between 1k and 10k triangles, a typical range for object sizes used in virtual environments – force shading significantly reduces the RMS force error for rendering our duck model. Note that the impact of force shading is related to the curvature

Model size (kTri)	Flops	RMS force error (N)	Relative error
0.2	9.7136M	0.085	9.92
0.5	10.361M	0.031	3.55
1	9.7921M	0.031	3.61
3	10.380M	0.022	2.61
6	10.560M	0.022	2.61
9	10.644M	0.015	1.80
64	10.064M	0.013	1.51
140	9.2452M	0.009	1.00

Table 3. Rendering accuracy of the duck model at various mesh resolutions, computed using the proxy algorithm. "Relative error" is computed as a fraction of the error obtained using the maximum-resolution model.

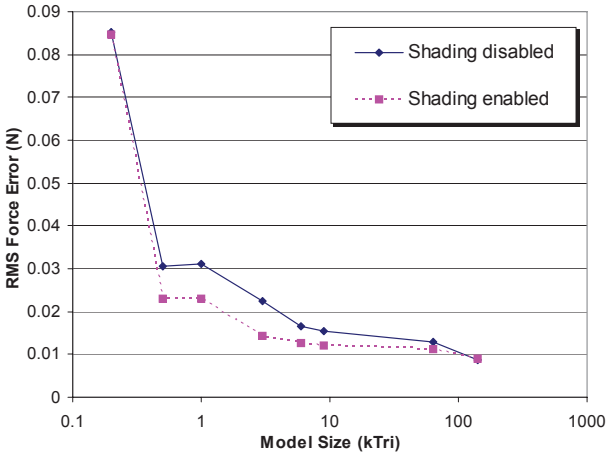

Figure 7. Impact of mesh size (logarithmic on the x-axis) and force shading on RMS Force Error (y-axis) for our duck model, rendered with the proxy algorithm.

of the object being rendered, and an object with smoothly-varying curvature (like our duck model) is expected to benefit significantly from force shading.

5. DISCUSSION

We have provided a series of "ground truth" data sets for haptic rendering, acquired with a novel scanning paradigm that allows force and position data to be acquired during a natural, human-driven scanning motion. We have also presented an approach for preprocessing this data to make it suitable as input for a variety of haptic rendering algorithms, and we have provided a series of example analyses that demonstrate our approach's ability to quantitatively assess haptic rendering systems.

A key application of these data and analyses is to assess the accuracy of a particular haptic rendering system and to approximately bound the difference between the forces experienced by a user through a haptic interface and the forces the user would experience performing the same interactions with a real object. This analysis can also be used to compare haptic rendering algorithms more objectively: if one algorithm consistently produces a lower force error relative to a real data set than another algorithm, it is objectively "more realistic" by our metrics. In this context, our ground truth data set and preliminary analysis techniques may play a role in haptics similar to the role played by [19] in stereo computer vision.

This approach has an application not only in evaluating published rendering systems, but also in debugging individual implementations. Debugging haptic rendering systems is often difficult relative to debugging other computer systems, due to the hard-real-time constraints, the nondeterminism introduced by physical devices, and the difficulty of reliably replicating manual input. Our approaches and our data sets allow a developer to periodically test a haptic rendering system via a series of objective evaluations, and thus rapidly identify problems and isolate the changes that caused them.

We have also provided an objective series of input data that can be used to evaluate the *computational performance* of an algorithm. In this context, our data sets and analyses provide a "haptic benchmark", analogous to the rendering benchmarks available to the graphics community, e.g. 3DMark (Futuremark Corp). Computational performance of a haptic rendering system can vary significantly with input, but it is difficult to describe and distribute the input stream used to generate a performance analysis result. By providing a standard data series and a set of reference results, we present a performance benchmark that authors can use to describe algorithmic performance. This is particularly relevant for objectively presenting the value of optimization strategies for rendering and collision detection whose primary value may lie in performance improvements. Performance results are still dependent on the platform used to generate the results, but this information can be reported concisely along with results.

The analyses presented here have focused primarily on "force correctness", with the ultimate metric of algorithmic correctness being the accuracy of output forces relative to ground truth forces. However, the use of standardized, pre-recorded haptic input data is also suited to assessing the *geometric* correctness of rendering algorithms, and for identifying anomalous cases that cause incorrect behavior in haptic rendering systems.

For example, figure 8 illustrates a problematic geometry that can be captured by our analysis approach. In this case, for certain stiffness values and angles of extrusion (i.e. "bump sharpness"), the surface contact point produced by the proxy algorithm becomes "stuck" on the bump, producing an incorrect trajectory that misrepresents object geometry. Our approach allows a rapid evaluation of this geometry using a variety of synthetic models and a variety of algorithmic parameters (friction values, stiffnesses), allowing quantification of such problematic cases for particular renderer implementations. These cases are very difficult to reliably isolate when a user and physical device are in the debugging loop.

Our current approach and available data sets, however, suffer from significant limitations. While a direct comparison of an algorithm's output forces to ground truth forces is expected to correlate to some degree with perceptual realism, it is not nearly a comprehensive metric. Furthermore, algorithmic performance and even results are expected to vary somewhat when collected with a user and a physical device in the loop, and no set of reference data can completely capture all possible cases that may have particular impacts on various rendering algorithms. Despite these limitations, we propose that a standard approach to haptic rendering analysis and standard data series will significantly enhance the quality and objectivity of haptic rendering system evaluation. In the following section, we will discuss future work and planned

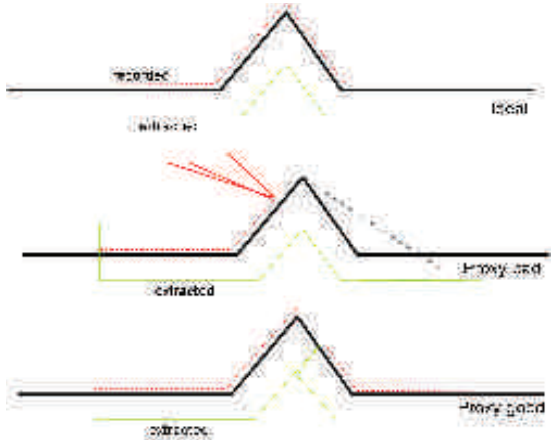

Figure 8. This failure case for the Proxy algorithm is an example of a geometric anomaly that can be captured and quantified using pre-recorded trajectories.

improvements to our online repository that will broaden the applicability of our data and methods.

6. Future Work

To address the limitations discussed in the previous section, future work will add both data and additional analyses to our repository. In particular, we hope to capture a wide variety of geometries, material types, contact pressures, and contact trajectories. Subsequent acquisitions will focus on adding more complex contact shapes (our current probe approximates a single point of contact).

Furthermore, the simple RMS force error metric used in this paper is not expected to be an optimal representation of *perceptual* accuracy of haptic rendering. Future work will include the development and psychophysical evaluation of more appropriate metrics for "haptic correctness".

Given a sufficient variety of data, our approach also may have value in the automated optimization of various parameters used in haptic rendering; the identification of a dynamic friction coefficient in section 4.2 is a preliminary example of this application. Future work will include the generalization of this optimization scheme to a wider variety of parameters, e.g. static friction, local compliance, roughness, and haptic texture.

7. Data Repository

To provide a standard reference that can be used by the community for evaluation of haptic rendering systems, the data, methods, and results discussed in this paper are publicly available at:

http://jks-folks.stanford.edu/haptic_data/

Acknowledgements

Support for this work was provided by NIH grant LM07295, the AO Foundation, and NSF grants IIS-0308157, EIA-0215887, ACI-0205671, and EIA-0321057. We also thank our reviewers for detailed and helpful feedback.

References

[1] F. Conti, F. Barbagli, D. Morris, and C. Sewell, "CHAI: An Open-Source Library for the Rapid Development of Haptic Scenes", *IEEE World Haptics*, Pisa, Italy, March 2005.
[2] SenseGraphics AB, "H3D API", http://www.h3d.org/
[3] SensAble Technologies, Inc., "OpenHaptics toolkit", http://www.sensable.com
[4] U.O. Gretarsdottir, F. Barbagli, and J.K. Salisbury, "Phantom-X", EuroHaptics 2003, Dublin, Ireland.
[5] A.E. Kirkpatrick and S.A. Douglas, "Application-based Evaluation of Haptic Interfaces", 10th IEEE Haptics Symposium, 2002, Orlando, USA.
[6] A. Guerraz, C. Loscos, and H.R. Widenfeld, "How to use physical parameters coming from the haptic device itself to enhance the evaluation of haptic benefits in user interface?", *EuroHaptics 2003*, Dublin, Ireland.
[7] C. Raymaekers, J. De Boeck, and K. Coninx, "An Empirical Approach for the Evaluation of Haptic Algorithms", IEEE World Haptics 2005, Pisa, Italy.
[8] V. Hayward and O.R. Astley, "Performance measures for haptic interfaces", Proc Robotics Research: 7th Intl Symp. 1996.
[9] J.E. Colgate and J.M. Brown, "Factors Affecting the Z-Width of a Haptic Display", Proc IEEE Conf on Robotics and Automation, San Diego, CA, USA, May 1994.
[10] D. Feygin, M. Keehner, and F. Tendick, "Haptic Guidance: Experimental Evaluation of a Haptic Training Method for a Perceptual Motor Skill", 10th IEEE Haptics Symposium, 2002.
[11] H.Z. Tan, "Identification of sphere size using the PHANToM: Towards a set of building blocks for rendering haptic environments", Proc ASME Annual Meeting, Vol. 61, Nov 1997.
[12] D. A. Lawrence, L. Y. Pao, A. M. Dougherty, M. A. Salada, and Y. Pavlou. "Rate-Hardness: a New Performance Metric for Haptic Interfaces", IEEE Transactions on Robotics and Automation, 16(4): 357-371, Aug. 2000.
[13] D. K. Pai, J. Lang, J. E. Lloyd, and R. J. Woodham. "ACME, A Telerobotic Active Measurement Facility". Proceedings of the Sixth International Symposium on Experimental Robotics, Sydney, Australia, March 1999.
[14] D. K. Pai, K. van den Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau, "Scanning Physical Interaction Behavior of 3D Objects," in Computer Graphics (ACM SIGGRAPH 2001 Conference Proceedings), August 2001.
[15] C. B. Zilles and J.K. Salisbury, "A Constraint-based God-object Method for Haptic Display", Intl Conference on Intelligent Robots and Systems, 1995.
[16] W.A. McNeely, K.D. Puterbaugh, and J.J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling", Proceedings of ACM SIGGRAPH 1999, pages 401-408.
[17] W.S. Harwin and N. Melder, "Improved Haptic Rendering for Multi-Finger Manipulation using Friction Cone based God-Objects", Proceedings of EuroHaptics 2002, Edinburgh, UK.
[18] H. B. Morgenbesser and M. A. Srinivasan, "Force shading for haptic shape perception", Proceedings of ASME Dynamic Systems and Control Division, (DSC-Vol.58): 407-412, 1996.
[19] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms", International Journal of Computer Vision. 47(1/2/3):7-42, April-June 2002.