Haptic Interaction with Multiresolution Image Curves

Dinesh K. Pai L.-M. Reissell

Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, Canada
Email: `pai|reissell@cs.ubc.ca`

We describe a system for interacting with shapes in two dimensional images with force feedback, using haptic devices. We construct multiresolution models of boundary curves in images, using wavelet multiresolution with good approximation properties. The boundary curves are treated as solid objects that will produce a force when pushed with a mouse-like haptic interface. This interaction force is then be rendered through a 2D haptic interface, providing kinesthetic feedback to the user.

1 Introduction

We describe a system for interacting with two dimensional images with force feedback, using haptic devices. Such a system has several potential applications: it could make image editing software more natural to use by providing kinesthetic feedback; it could also allow the visually impaired to access visual information more easily.

Specifically, we treat the boundaries of shapes in images as solid objects that will produce a force when pushed with a mouse-like haptic interface. This interaction force can then be "rendered" (i.e., felt by the user) through the haptic interface.

The boundary curves are obtained from edge detection and linking, and can have hundreds of edges on each curve. For such models the collision detection

[⋆] Portions of this paper were presented at the 1996 ASME Symposium on Haptic Interfaces, Atlanta, GA. This work was supported in part by IRIS, BC Advanced Systems Institute, and NSERC.

and contact geometry computation are critical issues. They have to be performed at high servo rates (for example, 100Hz – 1 KHz) on relatively small embedded computers.

We develop a wavelet multiresolution representation of curves which is well suited for such haptic interaction. Contact computations are performed at a desired tolerance level, and hierarchically refined. The resolution level can be selected to accommodate different haptic scales, as well as to account for limitations of the hardware. The haptic scales can be adapted dynamically to the task as well. For instance, we can choose to "render" finer scales only when the hand is moving slowly along the boundary.

Based on this multiresolution hierarchy, we have developed a fast, anytime algorithm to determine if a contact has occurred with the model, and if so, to compute the reaction force due to contact using the appropriate resolution. We maintain a partially expanded binary tree called a *collision tree*, which augments the multiresolution tree with error boxes to bound the maximum deviation of the curve from the approximation. The use of these selectively expanded trees corresponds to the "spatial coherence" used by other authors. We use the penetration of the object (at the given level of resolution) as a hybrid force and position command for computing the reaction force and the motion of the virtual object.

In §2 below, we describe our wavelet multiresolution representation of curves; §3 we describe the hierarchical collision detection method based on the multiresolution representation. §4 describes how contact forces and transitions are computed. §5 describes the current implementation and results using the Pantograph haptic interface.

1.1 Related Work

There has been considerable recent interest in the development of high performance haptic interface devices, and in controlling the devices to emulate contact with stiff walls (see, for example, [10] for a survey). We are not aware of any other work which addresses direct interaction with 2D images. In related work, [16] describes a system for reconstructing 3D shape from a stereo image pair and haptic interaction with the 3D shape at a single resolution.

Methods of hierarchical curve and surface representation based on subdivision (e.g. strip trees, quadtrees, spline subdivision) have been used extensively in geometric modeling (see, e.g.,[5]). Our hierarchical representation and its use is perhaps closest to the sphere trees of [6] and box trees of [18]. The important difference with our wavelet-based method is that it allows better control of the least-squares error between the approximation and the original curve; this means that the approximate, low resolution curve feels much more like the original and and we generally get tight error boxes.

In other related work, there has been considerable recent interest in fast incremental computation of the distance between polyhedra (e.g., [3], [7], [12]). Unlike the present work, these consider the geometry at a fixed resolution; hence it is not easy to adapt the performance of the algorithms dynamically by trading resolution for speed. The OBBTree [4] also uses a similar hierarchy of bounding boxes, but does not provide lower resolution approximations to the geometric object; therefore the hierarchy can not be directly used for interaction at different resolutions or denoising.

2 Multiresolution Curve Representation with Wavelets

We use wavelet multiresolution to represent the boundary of an object hierarchically. These methods will provide us with approximations of the object at different resolution levels, as well as a sequence of bounding boxes for collision detection.

There are other methods of related hierarchical data representation, including different filtering schemes. However, wavelet multiresolution has advantages over these methods: the approximation properties of the multiresolution hierarchy are good and can be carefully controlled by selecting suitable wavelet bases. Choosing good approximations leads to tighter bounding boxes and efficient collision checking. Wavelets also naturally support related applications such as compression of data for transmission in telerobotics tasks and denoising of data.

In this section, we provide a brief introduction to the use of wavelets in curve representation. The techniques can be directly extended to gridded surfaces. Wavelet methods are available for other types of surfaces as well. For more details, we refer the reader to, for example, [17], [15], and [2].

Generally, the wavelet transform converts a function into an alternate, but equivalent, representation in terms of scale and position. This can be compared with the Fourier transform which represents the data in terms of frequency.

More precisely, the function is represented using its coefficents in a basis of *wavelets*, $\tilde{\psi}_{lj}$, analogous to Fourier coefficients in a basis of sinusoids. In contrast to the globally supported sinusoids, wavelets are localized in position; they are also well localized in the frequency domain, so a single wavelet basis function picks out a given frequency band in a given spatial location.

There are many choices in selecting the wavelet basis. In a standard construction, a wavelet basis, $\tilde{\psi}_{lj}$, is obtained by dilations and translations of a basic wavelet, where the subscript l indexes the scale (or "level") of the wavelet and j indexes the position. The discrete wavelet transform of a function f is then a set of coefficents w_{lj} such that $f = \sum_{l,j} w_{lj}\tilde{\psi}_{lj}$. The transform can be computed using linear filtering operations.

A simple example of a discrete wavelet transform of a scalar function is shown in Fig. 1 (see [11] for more details). In the picture, the original data is shown on top, and below it is its partial wavelet transform. The wavelet coefficients on different resolution levels are arranged on top of each other, with the finest level coefficients at the top.

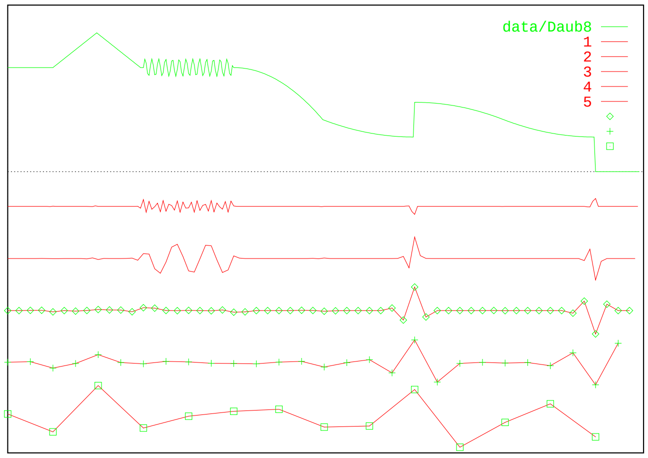


Fig. 1. Data and its wavelet coefficient levels, arranged from fine (top) to coarse (bottom). Going from coarse to fine, wavelet coefficients shrink rapidly to 0 in smooth areas, but stay large in rough areas. Note that the number of wavelet coefficients is doubled at each refinement step.

The wavelet transform procedure also yields a sequence of smoothed approximations to the original data. Smoothed approximations of the previous example, at selected levels, are shown in Fig. 2. The size of the data needed to determine the approximation on each level drops by 1/2 as the levels get coarser.

2.1 Wavelet filters

The discrete wavelet transform can be conveniently viewed as a fast ($O(n)$ where n is the number of data points) filtering operation, which simultaneously

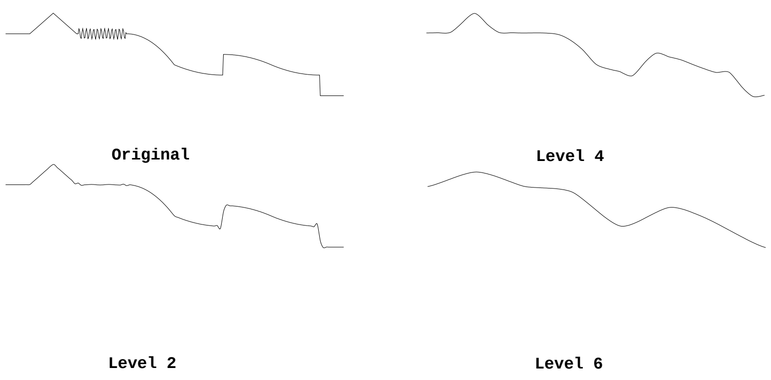


Fig. 2. Multiresolution approximation of data. The original data (level 0) and levels 2, 4 and 6 are shown. Note the disappearance of high frequency noise and the gradual smoothing effect.

builds a sequence of increasingly smoothed versions of the data and a sequence storing the omitted details. The latter sequence forms the wavelet transform.

In more detail, we begin with four finite filters H, G, $\tilde{H}$ and $\tilde{G}$, which satisfy the conditions required for a *biorthogonal* [2] wavelet scheme. These conditions are necessary to make the wavelet filtering and subsequent theoretical properties work. There is a variety of filters to choose from.

The discrete *wavelet transform* (or wavelet decomposition) of a one-dimensional data sequence $\mathbf{s}$ is the result of a repeated application of the two filters, the *scaling filter* H and the *wavelet filter* G:

$$\begin{tikzcd}
\mathbf{s} \arrow[r, "H"] \arrow[dr, "G"'] & \mathbf{s}_1 \arrow[r, "H"] \arrow[dr, "G"'] & \mathbf{s}_2 \arrow[r, "H"] \arrow[dr, "G"'] & \dots \\
& \mathbf{w}_1 & \mathbf{w}_2 & \dots
\end{tikzcd} \qquad (1)$$

- The sequences $\mathbf{s}_l$, are the *scaling coefficients* of the data at resolution level l.
- The sequences $\mathbf{w}_l$ are the wavelet coefficients at level l (depicted in Fig. 1).
- All the wavelet coefficients together constitute the *wavelet transform* (w_{lj}) of the original data. This is a sequence indexed by both resolution level l and position j.

The decomposition can be stopped at any level. Note that the number of scaling and wavelet coefficients is halved at each level. The full wavelet transform of data of length 2^L has L levels and the same length as the original data. Reconstruction is performed in the opposite direction with the other two filters $\tilde{H}$ and $\tilde{G}$. The algorithms and filters can be generalized to higher dimensions.

2.2 Wavelet decomposition and multiresolution approximation of curves

We can obtain a wavelet transform for a plane curve $f = (x(t), y(t))$ as the transforms of the separate coordinate functions $x(t)$ and $y(t)$. This generalizes to higher dimensions.

The wavelet transform of a curve yields not only the wavelet coefficients but also the *scaling coefficients* for each level and coordinate. The scaling coefficients completely determine the *multiresolution approximation* curves on each level; this is analogous to the control points of a spline. The scaling coefficients can also be used to give good piecewise linear approximations of the curve, provided the wavelets are chosen suitably.

A technical quantity, the *number of vanishing moments* of the wavelet, determines how fast the L^2 error of the multiresolution approximation decays towards finer scales [17]. Rapid error decay is very desirable behavior: it means that the curve approximations remain close to the original in the least squares sense even as we decrease the resolution level. The number of vanishing moments allows us to control the approximation obtained with wavelets.

Choosing good approximations leads to tighter bounding boxes and efficient collision checking. In this paper, we use the interpolating biorthogonal wavelets called *pseudocoiflets* [15] which provide scaling coefficients that approximate well. The wavelets have a relatively large number of vanishing moments (four) and we thus obtain multiresolution approximation curves with small errors to the original.

The example in Fig. 3 shows the scaling coefficients of an image boundary curve at the finest resolution of the original data, denoted level 0 (512 boundary points), as well as smoothed data at resolution level 2 (128 points) and level 4 (only 32 boundary points). The quality of approximation is good; for instance the level 0 and level 2 curves look almost identical at this resolution even though the level 2 curve has 25% of the data. Even the level 4 curve, with only 6.25% of the data, maintains the overall shape well.

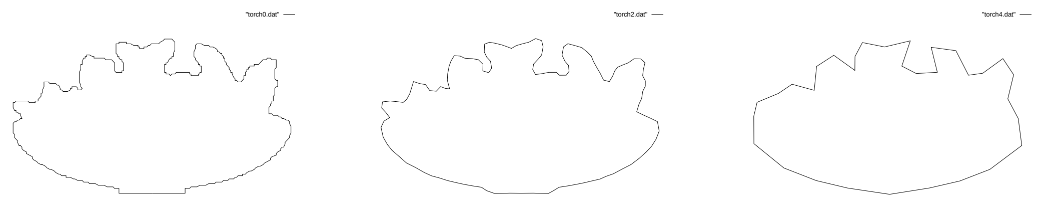


Fig. 3. Piecewise linear multiresolution approximation of curve data, using scaling coefficients. The original data (level 0) and levels 2 and 4 are shown.

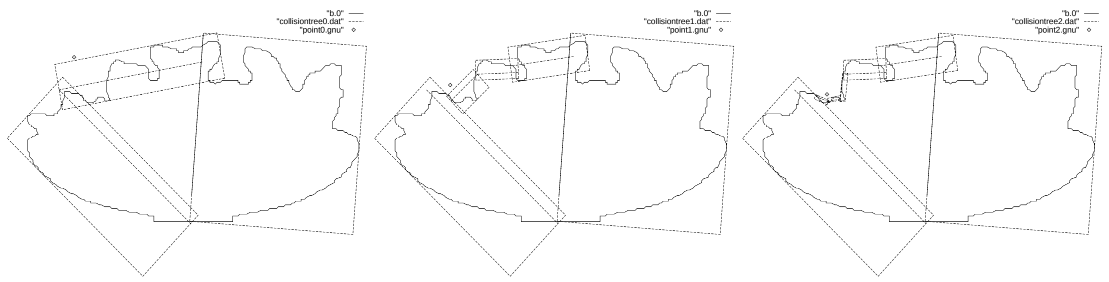


Fig. 4. Sections of collision tree for testing collision with approaching point. Shown in the figure are the error bounding boxes for the section and the linear segment at the given resolution level.

3 The Collision Tree

The collision tree is a hierarchical representation of each curve which consists of the approximating curve at each level. Each segment of the approximation also stores an error bounding box for the line segment which is aligned with the segment; these boxes bound the error between the original data and the approximating curve, and are precomputed offline. Thus we obtain a binary tree hierarchy of bounding boxes that is tightly fitted on the curve, matching its multiresolution structure.

The *virtual handle* of the haptic interface is the configuration of the virtual object being emulated by the haptic device; it is represented as a point in the configuration space of the object (see also [19]). For the purpose of this paper, we can assume that the object being emulated *is* a point in a two dimensional Euclidean space. A *boundary node* of the collision tree corresponding to a handle configuration is a node whose bounding box does not intersect the handle, but its parent's bounding box does. The *boundary section* of the collision tree is the list of boundary nodes. The boundary section therefore provides a representation of proximity to various parts of the curve. Fig. 4 shows the boundary sections at different times as the handle approaches a particular point on the top left side of the image boundary curve.

The configuration of the virtual handle, v, is sampled at discrete times; we denote the sample at time k as $v^{(k)}$. To detect collision as the handle moves from $v^{(k)}$ to $v^{(k+1)}$, we intersect the straight line segment connecting the two points with the boxes of the boundary section. This intersection test is similar to that in other hierarchical curve subdivision methods, such as strip trees [1] and arc trees [5]. The line segment is intersected with each box using parametric line clipping, and candidate intersections are sorted by proximity to the starting point $v^{(k)}$. If an intersection is detected with a box at level l, the box is refined to the next finer resolution level $l-1$ and the process is repeated. At the desired finest resolution level, or if the deadline for collision isolation is passed, the segment is intersected with the approximation to the curve at that level.

We use the fact that at the high sampling rates relative to the motion of the handle, the boundary section will require only a small number of changes from one time instance to the next. The change is typically either splitting a single box or merging two, by descending or ascending the tree. The distances to the boxes are easily computed and can be used to schedule box updates as in [8], though we have not implemented this.

4 Contact Force and Transitions

The reaction force is now computed as follows. We interpret the incremental motion of the real handle of the haptic device as a nominal motion of the virtual handle from $v^{(k)}$ to $v^{(k+1)}$. The penetration of the virtual handle into the boundary curve is interpreted as a hybrid force and position command. We model the contact as frictionless, with specified stiffness; the normal component of the penetration gives the reaction force while the tangential component gives the sliding motion along the boundary. This results in a net change in the applied force on the handle and the nominal position $v^{(k+1)}$. The nominal position is then checked for collision with adjacent regions of the curve. Note that the virtual handle never penetrates the boundary, even though the real handle may do so due to limits on the virtual stiffness — the visual display of the handle on the boundary greatly enhances the perception of hard contact (see also [19]).

In principle, contact transitions (e.g., transitions between single edge contact and vertex contact) could be implemented in the same way as collision detection. However, this is difficult using finite precision arithmetic. Instead, we organize contact computations as follows. We consider the types of contact to be states of a finite state machine with 3 states, labeled *0* (free space, no contact), *1* (edge contact), and *2* (vertex contact). Only the following transitions are allowed.

$$\begin{tikzcd} 0 \arrow[r, shift left] \arrow[loop below] & 1 \arrow[l, shift left] \arrow[r, shift left] \arrow[loop below] & 2 \arrow[l, shift left] \arrow[loop below] \end{tikzcd}$$

Note that transitions between contact states *0* and *2* occur, without loss of generality, via state *1*.

In state *0* (free space), the hierarchical collision test is performed as described in §3. If a collision is detected, the commanded motion is clipped to the boundary and a state transition occurs to state *1*.

In state *1* (single edge contact), the motion of the handle is interpreted as a hybrid position and force command; the component tangential to the edge is considered to be the desired motion, while the component perpendicular to the edge is considered a desired force. Note that we need a model of the contact impedance to convert handle motion to force. This should ideally be based on a material models of the curve and the handle. In the current implementation, we assume that the impedance is associated with the handle only, and is an isotropic stiffness (spring). This is a reasonable approximation of touching a hard curve using a soft, compliant finger.

Based on this model, we look for the first contact transition during the commanded motion. The next time for transition to state *0* corresponds to the time at which the contact force becomes zero. Similarly, the time for transition to state *2* corresponds to the time of intersection with a neighboring edge.

In state *2* (vertex contact), the motion of the handle is interpreted as a force command, with the handle impedance providing the conversion between handle motion and force. This makes sense since the virtual handle position is already completely constrained while the handle remains in this state. However, we must check if the resulting force can be supported by the contact.

Since the edges are assumed to be frictionless, each edge can only support a force directed along the inward normal to the edge; thus a necessary condition for remaining in vertex contact is that the total force is positively spanned by the two edge normals. However, this condition treats convex and concave corners identically. As a result, convex corners can sustain forces through a finite range of angles; while this is theoretically possible, it is unlikely and makes the corner feel "sticky." Therefore we additionally impose a stable contact requirement: the component of the handle force along an incident edge should be directed towards the vertex; if not, a state transition occurs to state *1* (in contact with the edge which violated this requirement[1]).

[1] If both edges fail the requirement, we pick one arbitrarily. It is possible to impose additional criteria, for instance to select the edge that produces the greatest rate of

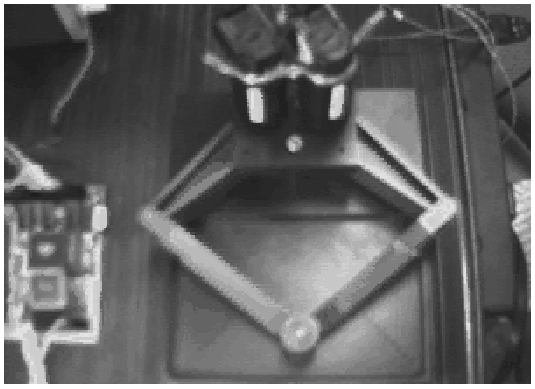


Fig. 5. Pantograph Haptic Interface

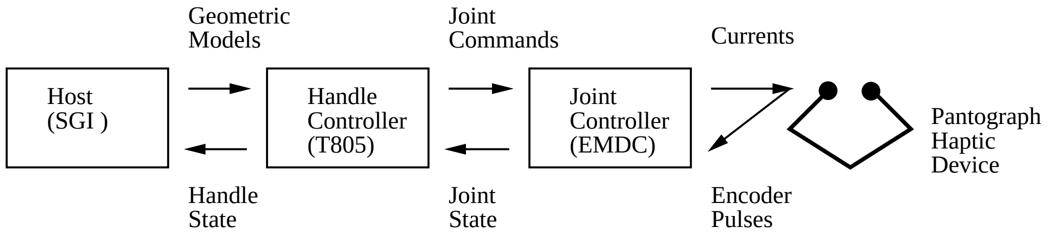


Fig. 6. System Architecture

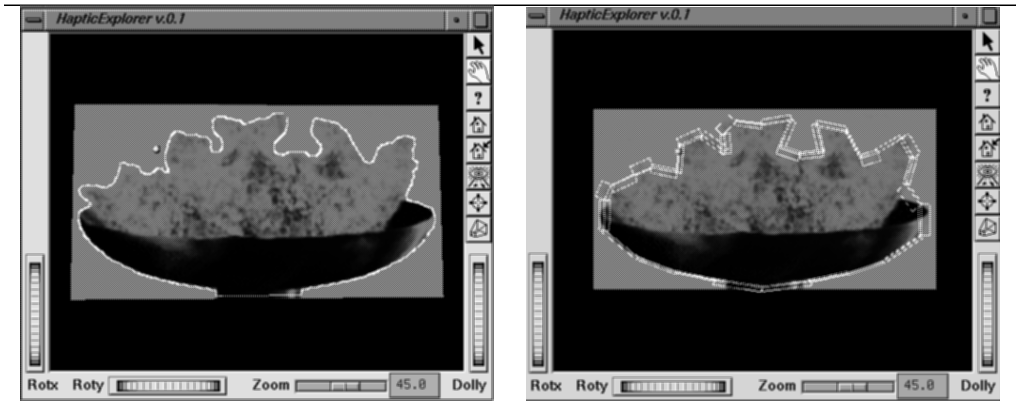
5 Results

The collision detection and response algorithms have been implemented on an embedded controller connected to the Pantograph haptic device designed by V. Hayward, et. al. [14]. See Fig. 5. It is a two degree of freedom, planar device; the user grasps the handle and moves it in a rectangle approximately 16cm wide and 10cm long, much as one moves a mouse. Forces are imposed on the handle using DC motors shown at the top of the figure.

The system architecture is shown in Fig. 6. The device is controlled at a 500 Hz servo rate by an "Embedded Module for Distributed Control" developed in our lab. The geometry computations as well as the forward kinematics of the closed loop linkage are performed by an embedded "Handle Controller," a T805 30MHz 32bit transputer with on-chip floating point hardware. The host is an SGI Indy XZ which boots the system, loads geometry onto the handle controller, as well as displays the virtual environment and handle at 60Hz using the Open Inventor graphics library.

Fig. 7 shows the display of the Haptic Explorer program, interacting with an image of the Olympic flame. The virtual handle is shown as a small white sphere. The boundary curve was obtained from the image by edge detection and linking. The Vista computer vision software developed by Pope and Lowe

decrease in potential energy; we feel this is unnecessary for the current application.



Level 2 (25% of original curve)   Level 4 (6.25% of original curve)

Fig. 7. Example of haptic exploration of Olympic flame image

[13] was used for this purpose. Specifically, an implementation of the Canny edge detector is used to extract edges, which are then linked using hysteresis thresholding. The figure also shows the boundary curve and bounding boxes at different resolution levels overlaid on the image. Note that bounding boxes for even the level 4 curve are quite small, due to the good approximation properties of our choice of wavelet basis.

The system performed well; users were able to interact with curves having several hundred edges – at a servo rate of 500 Hz – using simple embedded computers for collision detection and contact force computation. The ability to select resolution levels was also useful, since this allows denoising and compression.

For future work, an important area is the improvement of the contact model. Frictionless surfaces feel somewhat less realistic, and cause the handle to easily slip off the curve being explored – incorporating a friction model might improve the interaction. Another area that requires further development is automatic selection of appropriate resolution levels, based on limitations of the hardware and limitations of human perception. Finally, the addition of a rotational degree of freedom to the haptic interface promises to be useful since this will allow the user to probe the image "holding" any two dimensional shape, and to feel the resulting forces and torque.

6 Conclusions

We have presented a method for interacting with two dimensional images using haptic interfaces and force feedback. Our method extracts boundary curves using edge detection and linking. We use a wavelet multiresolution representation of the boundary curves, and a hierarchy of error bounding boxes at each resolution level for fast collision detection. We use wavelets with four vanishing moments; this yields approximations that fit the original data well in the least-squares sense and produce small error boxes. We have also described the computation of contact forces and contact transitions, based on interpreting the input motion of the handle as a hybrid force/position command. A prototype of the system has been implemented.

References

[1] D. Ballard, Strip trees: a hierarchical representation for curves, *ACM Communications*, 24, pp. 310–321, 1981.

[2] I. Daubechies. *Ten Lectures on Wavelets. CBMS-NSF Regional Conference Series in Applied Mathematics* 61, SIAM, 1992.

[3] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, April 1988.

[4] S. Gottschalk, M. C. Lin, and D. Manocha, OBBTree: A hierarchical structure for rapid intersection detection. *SIGGRAPH 96 – Computer Graphics Proceedings*, August 1996, 171–180.

[5] O. Gunther and S. Dominguez, Hierarchical schemes for curve representation, *IEEE Computer Graphics and Applications* 13(3), May 1993, 55–63.

[6] P. M. Hubbard. Real-time collision detection and time-critical computing. In *First Workshop on Simulation and Interaction in Virtual Environments*, pages 92–96, 1995.

[7] M. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991.

[8] B. Mirtich and J. F. Canny. Impulse-based dynamic simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, 1995.

[9] S. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Trans. PAMI* **11**, 1989, pp. 674–693.

[10] National Research Council. *Virtual Reality – Scientific and Technological Challenges*. National Academy Press, 1995. Chapter 4: Haptic Interfaces.

[11] D. K. Pai and L.-M. Reissell, "Multiresolution Rough Terrain Motion Planning," to appear in *IEEE Transactions on Robotics and Automation*, 1997.

[12] M. Ponamgi, J. Cohen, M. Lin, and D. Manocha. Incremental algorithms for collision detection between polyhedral models. In *First Workshop on Simulation and Interaction in Virtual Environments*, pages 84–91, 1995.

[13] A. Pope and D. Lowe. Vista: A Software Environment for Computer Vision Research. *CVPR* 1994. URL: http://www.cs.ubc.ca/nest/lci/vista/vista.html.

[14] C. Ramstein and V. Hayward. The pantograph: A large workspace haptic device for a multi-modal human-computer interaction. *Conference on Human Factors in Computing Systems ACM/SIGCHI*, 1994.

[15] L.-M. Reissell. Wavelet Multiresolution Representation of Curves and Surfaces. *Graphic Models and Image Processing*, Vol.58, No.3, pp.198–217, 1996.

[16] Y. Shi and D. K. Pai, "Haptic Display of Visual Images," to appear in *Proceedings of IEEE Virtual Reality Annual International Symposium* (VRAIS '97), Albuquerque, NM, March 1997.

[17] G. Strang. Wavelets and dilation equations: a brief introduction. *SIAM Review* 31(4), 1989, pp. 614–627.

[18] G. Zachmann and W. Felger. The boxtree: Enabling real-time and exact collision detection of arbitrary polyhedra. In *First Workshop on Simulation and Interaction in Virtual Environments*, pages 104–113, 1995.

[19] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 146–151, 1995.