Active Voodoo Dolls: A Vision Based Input Device for Nonrigid Control

John Isidoro and Stan Sclaroff
Computer Science Department
Boston University
jisidoro@bu.edu, sclaroff@bu.edu

Abstract

A vision based technique for nonrigid control is presented that can be used for animation and video game applications. The user grasps a soft, squishable object in front of a camera that can be moved and deformed in order to specify motion. Active Blobs, a nonrigid tracking technique is used to recover the position, rotation and nonrigid deformations of the object. The resulting transformations can be applied to a texture mapped mesh, thus allowing the user to control it interactively. Our use of texture mapping hardware in tracking makes the system responsive enough for interactive animation and video game character control.

1 Introduction

A wide variety of input devices have been invented for human-computer interaction. One category of devices can transform intentional human motion into a measurable analog quantity. Not only must such devices be accurate; they must also be intuitive in such a way that motion of the device corresponds directly with the motion that the human is controlling.

Basic analog joysticks and mice have two degrees of freedom, which correspond to the screen's x and y axes. The spaceorb and polhemus provide six rigid degrees of freedom. Haptic feedback pens, data gloves, and body suits can provide even more than six degrees of freedom. However, all of these devices have the limitation that they can only sense rigid, or in the best case, articulated motion. Although it is theoretically possible to build a device to measure higher degrees of freedom in motion (bending, twisting, *etc.*), vision based techniques offer an inexpensive alternative, without the wiring and cumbersome devices.

In this paper, we present a vision based technique for nonrigid control. The user grasps a soft object in front of a camera that can be moved and deformed in order to specify motion. Active Blobs, a nonrigid tracking technique is used to recover the object position, rotation, and deformations. The resulting transformations can be applied to a graphics model, thereby allowing the user to control motion interactively. The use of texture mapping hardware in tracking makes our system responsive enough for interactive animation and video game character control.

2 Related Work

Sensing motion and transforming it into meaningful data using a camera is most easily done using some form of tracking. Most previous systems that use vision to control graphics have tracked face, hand, or body gestures.

Some previous approaches are based on tracking feature points across a sequence of video frames. Williams [26] tracked iridescent points placed on the face of the user. The points were used to pull regions of a triangular mesh of a human head. This approach has the disadvantage that the dots must be painted or pasted onto the user's face. In contrast, Azarbayejani, *et al.* [1] used a feature finding algorithm to track facial points and converted them into three-dimensional estimates using an extended Kalman Filter. Motion estimates were then used to drive a rigid graphics model of the user's head.

Contour-based approaches have also been proposed. For instance, Terzopoulos and Waters [25] used snakes to follow lines drawn on the user's face. The snakes drive an intricate face model where muscle and skin are physically simulated. In a similar approach, Blake and Isard employed contours in gesture tracking[4], allowing the user to use hand motion as a three-dimensional mouse. Rigid motion supplies the position of the mouse while nonrigid motion supplies button pressing and releasing actions.

Other researchers [8, 10] have used optic flow to drive the motion of an anatomically-motivated polyhedral model of the face. Optic flow has also been used to capture motion of more general nonrigid models like deformable superquadrics [17, 18]. In a different approach, histograms of optic flow direction can be used directly in hand gesture recognition [11], allowing television viewers to freely use their hand like a mouse to make menu selections.

Other researchers have used simple low-level image processing techniques such as normalized image correlation [7], or background subtraction and thresholding [5, 15, 16, 27] for tracking moving body parts, facial features, and hand gestures. The resulting tracking output can be used to drive the animation of a face model [9], to drive manipulation of virtual objects [15], or to interact with animated creatures in a virtual world [27, 16, 5]. Such low-level methods are surprisingly useful for qualitative mea-
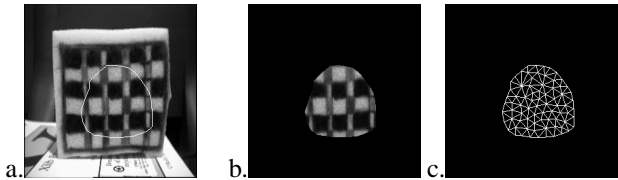
Figure 1: Example voodoo doll object used in our experiments (a), and a deformable color region on the object used for tracking (b). The deformable image region is modeled using a texture mapped triangular mesh. The underlying triangular model is shown in (c).
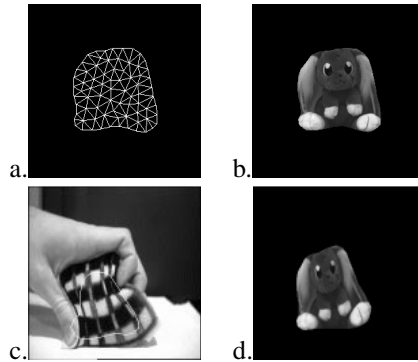

Figure 2: Example animation of a two-dimensional character via deformation of the voodoo doll object as constructed in Fig. 1. The character is defined as a polygonal model (a), with a texture map applied (b). By directly deforming the voodoo doll (c), the user can control the deformation of the character. The resulting deformation of the character is shown in (d).

sures (*e.g.,* person is standing, object is bending) but are not exact enough to allow quantitative measures of fine-grain deformation (*e.g.,* how much bending or stretching). Furthermore, correlation-based methods are not well-suited to tracking moving objects that undergo scaling, rotation, or nonrigid deformation, since tracking is accomplished via correlation with a translating template.

Larger deformations can be accomodated if the problem is posed in terms of deformable image registration [2, 23]. In particular, La Cascia, *et al.* [6], use this approach to register a texture-mapped polygonal model of a person's head with an incoming video sequence. The approach allows tracking of a wider range of free motion of the head than was possible using flow or simple correlation methods.

3 Approach

In our system, the user grabs a "voodoo doll" that can be as simple as a decorated piece of foam or as complex as a toy stuffed animal. The user moves, rotates, and squishes the voodoo doll in the same way that he wants to manipulate the corresponding graphics object on the screen. This allows the intuitiveness of a joystick or spaceorb, while providing more degrees of freedom. The only limitation of the voodoo doll's movement is that it must remain visible to the camera at all times in order to follow its deformations.

An example voodoo doll object is shown in Fig. 1(a). In this case, the object is simply a piece of foam rubber painted with a colored grid pattern.

For tracking purposes, we build a two-dimensional deformable texture mapped model of a region on the squishy object. We model the region using the *active blob* formulation [23]. Tracking of the voodoo doll's deformation is modeled in terms of a deformable, triangular mesh that captures object shape *plus* a color texture map that captures object appearance.

As shown in Fig. 1(b), a two-dimensional active blob model is constructed for our example using a modified Delaunay triangulation algorithm. The blob's appearance is then captured as a color texture map and applied directly to the triangulated model, as shown in Fig. 1(c).

Nonrigid motion tracking is achieved by warping this texture-mapped blob to register it with each incoming video frame. A blob warp is defined as a deformation of the mesh and then a bilinear resampling of the texture mapped triangles. By defining image warping in this way, it is possible to harness hardware accelerated triangle texture mapping capabilities becoming prevalent in mid-end workstations, and PC's.

An example of using the voodoo doll to control deformation of a two-dimensional animated character is shown in Fig. 2. In this case, the character is defined as a polygonal model Fig. 2(a), with a texture map applied Fig. 2(b). By directly deforming the voodoo doll Fig. 2(c), the user can control the deformation of the character. The resulting deformation of the character is shown in Fig. 2(d).

The same deformations used to warp the tracking blob can be applied to a completely separate graphics object. As was shown in this example, the approach can be used to warp a 2D model. As will be seen later in this paper, the approach can also be used to control the deformation of a 3D polyhedral model.

By deforming the voodoo doll, the user can control deformation of the graphics model with the benefit of haptic feedback. Furthermore, since it is a visual interface, it is an interface that is unencumbered by wires.

The voodoo doll input device not only allows for translation, scaling, and rotation, but also for bending, shearing, tapering, and other nonrigid deformations. It is also possible for the user to select only a subset of deformation parameters to be applied to the character. For instance, it is possible to use the voodoo doll to control bending of the character, while leaving rotation, translation, or other deformation parameters fixed.

4 Tracking Formulation: Active Blobs

Tracking is accomplished via the active blobs formulation. To gain better robustness, active blobs incorporate information about not only shape, but also color image appearance. Active blobs also provide some robustness to photometric variations, including specular highlights and shadows. By taking advantage of texture mapping hardware, active blobs can track nonrigidly deforming shapes at speeds approaching video frame rate. We will now give an overview of the active blobs formulation. For a more detailed formulation, readers are refered to [23].

4.1 Blob Warping

In the active blobs formulation, nonrigid deformation is controlled by parametric functions. These functions are applied to the vertices that define the active blob's two-dimensional trianglular mesh:

$$\mathbf{X}' = f(\mathbf{X}, \mathbf{u}) \qquad (1)$$

where $\mathbf{u}$ is a vector containing deformation parameters, $\mathbf{X}$ contains the undeformed mesh vertex locations, and $\mathbf{X}'$ contains the resulting displaced vertices.

Image warping and interpolation are accomplished by displacing the mesh vertices and then resampling using bilinear interpolation. Thus we define a warping function for an input image, $\mathbf{I}$:

$$\mathbf{I}' = W(\mathbf{I}, \mathbf{u}) \qquad (2)$$

where $\mathbf{u}$ is a vector containing warping parameters, and $\mathbf{I}'$ is the resulting warped image.

Perhaps the simplest warping functions to be used are those of a 2D affine model or an eight parameter projective model [24]. Unfortunately, these functions are only suitable for approximating the rigid motion of a planar patch. The functions can be extended to include linear and quadratic polynomials [2]; however, the extended formulation cannot model general nonrigid motion.

A more general parameterization of nonrigid motion can be obtained via the use of the modal representation [19]. In the modal representation, deformation is represented in terms of eigenvectors of a finite element (FE) model. Taken together, modes form an orthogonal basis set for describing nonrigid shape deformation. Deformation of a triangle mesh can be expressed as the linear combination of orthogonal modal displacements:

$$\mathbf{X}' = \mathbf{X} + \sum_{j=1}^{m} \phi_j \tilde{u}_j \qquad (3)$$

where $\tilde{u}_j$ is the jth mode's parameter value, and the eigenvector ϕ_j defines the displacement function for the jth modal deformation.

The lowest order modes correspond with rigid degrees of freedom. The higher-order modes correspond qualitatively with human notions of nonrigid deformation: scaling, stretching, skewing, bending, etc. Such an ordering of shape deformation allows us to select which types of deformations are to be observed. For instance, it may be desirable to make tracking rotation, position, and/or scale independent. To do this, we ignore displacements in the appropriate low-order modes.

4.2 Including Illumination in the Model

Surface illumination may vary as the user moves and manipulates the active voodoo doll. In order to account for this we can derive a parameterization for modeling brightness and contrast variations:

$$\mathbf{I}' = cW(\mathbf{I}, \mathbf{u}) + b, \qquad (4)$$

where b and c are brightness and contrast terms that are implemented using the workstation's image blending functionality.

Blob deformation and photometric parameters can now be combined in generic parameter vector $\mathbf{a}$. The image warping function becomes:

$$\mathbf{I}' = \mathcal{W}(\mathbf{I}, \mathbf{a}). \qquad (5)$$

5 Tracking

The first goal of our system is nonrigid shape tracking. To achieve this, the system recovers warping parameters that register a template image $\mathbf{I}_0$ with a stream of incoming video images. The maximum likelihood solution to this two image registration problem consists of minimizing the squared error for all the pixels within the blob:

$$E_{image} = \frac{1}{n} \sum_{i=1}^{n} e_i^2 \qquad (6)$$

$$e_i = \|\mathbf{I}'(x_i, y_i) - \mathbf{I}(x_i, y_i)\| \qquad (7)$$

where $\mathbf{I}'(x_i, y_i)$ is a pixel in the warped template image as prescribed in Eq. 5, and $\mathbf{I}(x_i, y_i)$ is the pixel at the same location in the input. The above equation is formulated for comparing two color images; thus, it incorporates the sum of squared difference over all channels at each pixel.

5.1 Robust Registration

Image registration can be corrupted by outliers. The process can be made less sensitive to outliers if we replace the quadratic error norm with a robust error norm [13]:

$$E_{image} = \frac{1}{n} \sum_{i=1}^{n} \rho(e_i, \sigma), \qquad (8)$$

where σ is a scale parameter that is determined based on expected image noise. If it is assumed that noise is Gaussian distributed, then the optimal error norm is simply the quadratic norm $\rho(e_i, \sigma) = e_i^2/2\sigma^2$. However, robustness to outliers can be further improved via the use of a Lorentzian influence function:

$$\rho(e_i, \sigma) = \log(1 + \frac{e_i^2}{2\sigma^2}). \qquad (9)$$

This norm replaces the traditional quadratic norm found in least squares, and is equivalent to the incorporation of an analog outlier process in our objective function [3]. The formulation results in better robustness to specular highlights and occlusions. For efficiency, the $\log$ function can be implemented via table look-up.

Equation 8 includes a data term only; thus it only enforces the recovered model's fidelity to the image measurements. The formulation can be extended to include a regularizing term that enforces the priors on the model parameters $\mathbf{a}$:

$$E \quad = \quad \frac{1}{n} \sum_{i=1}^{n} \rho(e_i, \sigma) + \gamma \sum_{j=1}^{m} a_j^2 \psi_j^2 \qquad (10)$$

where where ψ_j^2 are the penalties associated with changing each parameter, and γ is a constant that controls the relative importance of the regularization term. The penalties can be derived directly from the modal stiffness obtained in modal analysis [19].

5.2 Difference Decomposition

To minimize the energy (Equation 10) a *difference decomposition* approach[12] is used. The approach offers the benefit that it requires the equivalent $O(1)$ image gradient calculations and $O(N)$ image products per iteration.

In the difference decomposition, we define a basis of difference images generated by adding small changes to each of the blob parameters. Each difference image takes the form:

$$\mathbf{b}_k = \mathbf{I}_0 - \mathcal{W}(\mathbf{I}_0, \mathbf{n}_k), \qquad (11)$$

where $\mathbf{I}_0$ is the template image, and $\mathbf{n}_k$ is the parameter displacement vector for the kth basis image, $\mathbf{b}_k$. Each resultant difference image becomes a column in a difference decomposition basis matrix $\mathbf{B}$. This basis matrix can be determined as a precomputation.

During tracking, an incoming image $\mathbf{I}$ is inverse warped into the blob's coordinate system using the most recent estimate of the warping parameters $\mathbf{a}$. We then compute the difference between the inverse-warped image and template:

$$\mathbf{D} = \mathbf{I}_0 - \mathcal{W}^{-1}(\mathbf{I}, \mathbf{a}). \qquad (12)$$

This difference image $\mathbf{D}$ can then be approximated in terms of a linear combination of the difference decomposition's basis vectors:

$$\mathbf{D} \approx \mathbf{B}\mathbf{q}, \qquad (13)$$

where $\mathbf{q}$ is a vector of basis coefficients.

Thus, the maximum likelihood estimate of $\mathbf{q}$ can be obtained via least squares:

$$\mathbf{q} = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \mathbf{D}. \qquad (14)$$

The change in the image warping parameters is obtained via matrix multiplication:

$$\Delta \mathbf{a} = \mathbf{N}\mathbf{q}, \qquad (15)$$

where $\mathbf{N}$ has columns formed by the parameter displacement vectors $\mathbf{n}_k$ used in generating the difference basis. The basis and inverse matrices can be precomputed; this is the key to the difference decomposition's speed. If needed, this minimization procedure can be iterated at each frame until the percentage change in the error residual is below a threshold, or the number of iterations exceeds some maximum.

The difference decomposition can be extended to incorporate the robust error norm of Eq. 9:

$$\bar{\mathbf{b}}_k(x_i, y_i) = \operatorname{sign}(\mathbf{b}_k(x_i, y_i)) \sqrt{\rho(\mathbf{b}_k(x_i, y_i), \sigma)}, \qquad (16)$$

where $\mathbf{b}_k(x_i, y_i)$ is the basis value at the ith pixel. The difference template $\mathbf{D}$ is computed using the same formula. Finally, the formulation can be extended to include a and a regularizing term that enforces the priors on the model parameters as described in [23].

6 Using Observed Deformations to Animate Objects

As the user manipulates the voodoo doll input device, we want corresponding deformations to be applied to the graphics model. The appropriate amounts of deformations to be applied are determined through tracking the deforming active blob as describe above.

We could apply the observed deformations directly to the computer model. However, due to possible differences in scale, orientation, and position, we need to first allow the user to define the appropriate object-centered coordinate frame and bounding box for the animated object. This helps to establish a mapping between the coordinate spaces of the voodoo doll input device and the animated model.

The voodoo doll and the graphics model may not have the same overall shape or extent, and may have different polygonal sampling. Therefore, we must employ an interpolation scheme for mapping deformations from the

voodoo doll input device onto the graphics model. To begin with, we position, rotate, and scale the model such that its bounding box is the same as the voodoo doll's. Next we use the doll's finite element interpolants to obtain the modal displacement vectors of Eq. 3:

$$\phi_j' = \mathbf{H}(\mathbf{X}_{model})\phi_j \qquad (17)$$

where ϕ_j' is the interpolated modal displacement vector describing how deformations of the voodoo doll are applied at the graphics model's triangle mesh vertices $\mathbf{X}_{model}$. The FE interpolation matrix, $\mathbf{H}$, is formulated using Gaussian interpolants as specified in [22].

To give the further control over deformation, we allow the user to specify that only a subset of the observed deformation parameters is to be applied to the graphics model. For instance, it is possible to use the voodoo doll to control bending of the graphics model, while leaving rotation, translation, and/or other deformation parameters fixed.

7 Implementation Details

Blob construction starts with the determination of a support region for the object of interest. The bounding contour(s) for a support region can be extracted via a standard 4-connected contour following algorithm [20]. Alternatively, the user can define a bounding contour for a region via a sketch interface. In general, the number of contour segments must be reduced. We utilize the tolerance band approach, where the merging stage can be iteratively alternated with recursive subdivision [14]. In practice, a single merging pass is sufficient for a user-sketched boundary.

The triangles are then generated using an adaptation of Ruppert's Delaunay refinement algorithm [21]. The algorithm accepts two parameters that control angle and triangle size constraints. To satisfy these constraints, additional interior vertices may be added to the original polygon during mesh generation. The source code is available from http://www.netlib.org/voronoi/.

Once a triangle mesh has been generated, a RGB color texture map is extracted from the example image. Each triangle mesh vertex is given an index into the texture map that corresponds to its pixel coordinate in the undeformed example image $\mathbf{I}_0$. To improve convergence and noise immunity in tracking, the texture map is blurred using a Gaussian filter. Texture map interpolation and rendering were accomplished using OpenGL.

Given a triangle mesh, the FE model can be initialized using Gaussian interpolants with finite support. Due to space limitations, readers are directed to [22] for the formulation. The generalized eigenvectors and eigenvalues are computed using code from the EISPACK library: http://www.netlib.org/eispack/.

Finally, difference decomposition basis vectors are precomputed. In practice, four basis vectors per model parameter are sufficient. For each parameter a_i, these four basis images correspond with the difference patterns that result by tweaking that parameter by $\pm\delta_i$ and $\pm2\delta_i$. The factor $2\delta_i$ corresponds to the maximum anticipated change in that parameter per video frame.

8 Experimental Results

Our system was implemented on an Indigo2 Impact with a 195Mhz R10K processor, 192MB RAM, and hardware texture mapping. The code is written in C++ and all timings are reported for unoptimized code. In our interface, both the tracking and the graphics object can be seen at once in 2 separate 256x256 true color windows.

Fig. 3 shows an example of manipulation of a two-dimensional object. The user grabs a decorated piece of foam rubber that is moved and deformed. The region of the foam that was circled by the user is used to track the foam's movements.

The system observes the user manipulation of a deformable spongy object via the camera. Representative frames from such a video sequence are shown in Fig. 3(a). The system then tracks the motion of the object by tracking a deformable region as shown in Fig. 3(b).

The recovered deformation parameters are then applied to the computer model for animation as shown in Fig. 3(c). Note that because the voodoo doll device and the graphics model have different object coordinate systems, interpolation and/or extrapolation of the deformations was used (as described in Sec. 6).

It can be seen that the bunny character is following the motion of the foam patch. This example uses 14 deformation modes. The blob used for tracking contains 8503 pixels and consists of 72 points with 112 triangles. The animal object blob contains 10432 pixels and consists of 79 points with 118 triangles. At 256x256 resolution tracking and deformation occurred at interactive framerates, 6.6 frames per second (fps). Thus there was no delay in the haptic and visual feedback loop to the user controlling the deformation. At 128x128 resolution the same example runs at 10.2 fps with no loss in tracking quality.

In Fig. 4 a 3D object is deformed. In our interface, the object can be rotated so it and the voodoo doll object being manipulated by the user have the same orientation from the users point of view. Doing this gives the user the illusion of actually touching the object on the screen. The graphics object moves just like the real object, and in this way, it inherits the real object's physical properties. What was once a rigid cardboard animal cracker box, now seems to be made of soft spongy material.

In this example, 20 modes are used to perform the tracking. The tracking blob contains 17278 pixels and consists of 61 points with 87 triangles. The 3D object is produced by extruding the contour of the polygonal mesh of the ani-
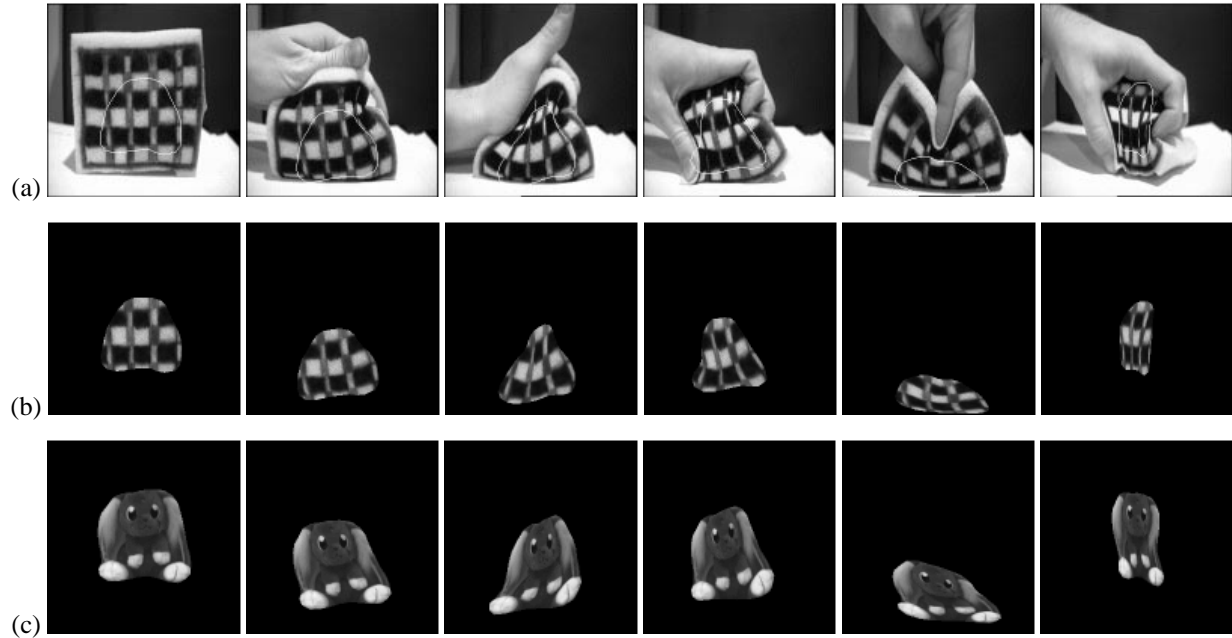
Figure 3: Animating a bunny character via direct manipulation of a spongy object. The system observes the user manipulation of a deformable spongy object via a camera; representative frames from such a video sequence are shown in (a). The system then tracks the motion of the object by tracking a deformable region as shown in (b). The recovered deformation parameters are then applied to the graphics model for animation as shown in (c).

mal cracker blob. It consists of 116 vertices with 226 triangles. At 256x256 resolution the system runs at 3.3 fps. The larger size of the tracking blob, the 3D rendering, and the extra 6 modes make the system slower than in the first example. However, at 128x128 resolution the same 20 mode example runs at 8.2 fps using the same blob at half size (4335 pixels) for tracking.

9   Conclusion

The Active Blobs formulation allows a level of control rarely seen in an input device. Due to the increasing popularity of personal computer video cameras and hardware texture mapping, Active Voodoo Dolls could perhaps be used as a game controller for home use in a few years. Games using this system could have characters capable of bending around obstacles, squishing under tables, and slithering around corners, with a level of control unavailable using conventional input devices. The main character in the game could be controlled by a real life stuffed animal likeness of it being moved in front of the camera.

Another possibility is to use the system as an computer animation tool. A animator can bend a real object the way he wants his synthesized object to bend. Because the physical object is actually being bent, there is an innate tactile response of the object. Using the system this way provides for a simple and inexpensive form of haptics.

One of the limitations of this system is that the track-

ing is two- dimensional, and all depth foreshortening effects in tracking must be overcome by the parametric warping functions used. Complex topologies extending along the axis of the camera can provide difficulty in the form of non-homogenous shading effects as well as occlusion. However, the graphics object is not limited to be two-dimensional, and can be three-dimensional with the deformations applied to two dimensions. This gives the user the feeling of manipulating a solid object on the screen.

References

[1] A. Azarbayejani, T. Starner, B. Horowitz, and A.P. Pentland. Visually controlled graphics. *PAMI*, 15(6):602–605, June 1993.

[2] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. *Proc. ICCV*, 1995.

[3] M.J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 19(1):57–91, 1996.

[4] A. Blake and M. Isard. 3D position, attitude and shape input using video tracking of hands and lips. *Proc. SIGGRAPH*, 1994.

[5] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. In *Proc. Royal Soc.: Special Issue on Knowledge-based Vision in Man and Machine*, 1997.
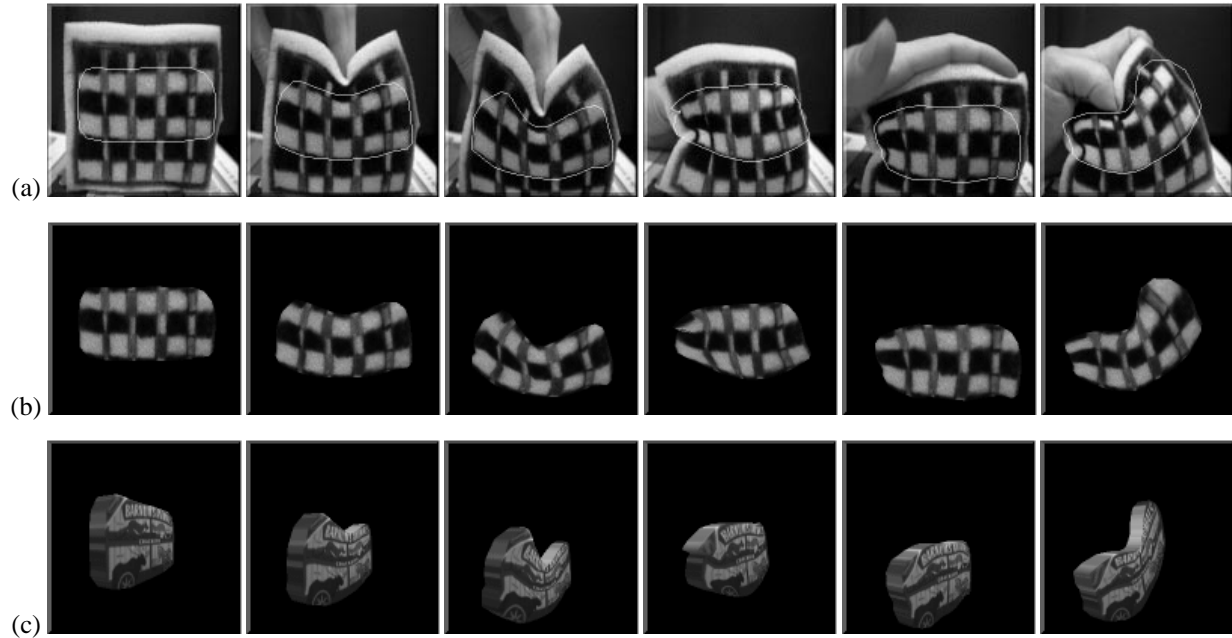
Figure 4: Animating a three-dimensional animal cracker box via direct manipulation of a spongy object. The video sequence is shown in row (a). The deformable region used for tracking is in row (b). In row (c) the deformation parameters are applied to two dimensions of a three-dimensional animal cracker box.

[6] M. La Cascia, J. Isidoro, and S. Sclaroff. Head tracking via robust registration in texture map images. *Proc. CVPR*, 1998 (to appear).

[7] T. Darrell and A. Pentland. Space-time gestures. *Proc. CVPR*, 1993.

[8] D. DeCarlo and D. Metaxas. The integration of optical flow and deformable models: Applications to human face shape and motion estimation. *Proc. CVPR*, 1996.

[9] I. Essa, T. Darrell, and A. Pentland. Tracking facial motion. *MIT Media Lab, Vision and Modeling TR 272*, 1994.

[10] I. Essa and A. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *PAMI*, 19(7):757–763, 1997.

[11] W. Freeman and C. Weissman. Television control by hand gestures. In *Proc. Intl. Workshop on Automatic Face- and Gesture- Recognition*, 1995.

[12] M. Gleicher. Projective registration with difference decomposition. *Proc. CVPR*, 1997.

[13] F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stehel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley and Sons, 1986.

[14] R. Jain, R. Kasturi, and B. Shunck. *Machine Vision*. McGraw-Hill, 1995.

[15] M. Krueger. *Virtual Reality II*. Addison Wesley, 1990.

[16] P. Maes, T. Darrell, B. Blumberg, and A. Pentland. The alive system: Wireless, full-body interaction with autonomous agents. *Multimedia Systems*, 5(2):105–112, 1997.

[17] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *PAMI*, 15(6):580 – 591, 1993.

[18] A. Pentland and B. Horowitz. Recovery of non-rigid motion and structure. *PAMI*, 13(7):730–742, 1991.

[19] A. Pentland and J. Williams. Good vibrations : Modal dynamics for graphics and animation. *Proc. SIGGRAPH*, 23(4):215–222, 1989.

[20] A. Rosenfeld and A. Kak. *Digital Picture Processing*. Academic Press, 1976.

[21] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. of Algorithms*, 18(3):548–585, 1995.

[22] S. Sclaroff. *Modal Matching: A Method for Describing, Comparing, and Manipulating Digital Signals*. PhD thesis, MIT Media Lab, 1995.

[23] S. Sclaroff and J. Isidoro. Active blobs. *Proc. ICCV*, 1998.

[24] R. Szeliski. Video mosaics for virtual environments. *IEEE CG&A*, 16(2):22–30, 1996.

[25] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *PAMI*, 15(6), 1993.

[26] L. Williams. Performance-Driven Facial Animation. *Proc. SIGGRAPH*, 24(4):235–242, 1990.

[27] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 19(7):780–785, 1997.