Stylizing 2.5-D Video

Noah Snavely
University of Washington

C. Lawrence Zitnick
Microsoft Research

Sing Bing Kang
Microsoft Research

Michael Cohen
Microsoft Research

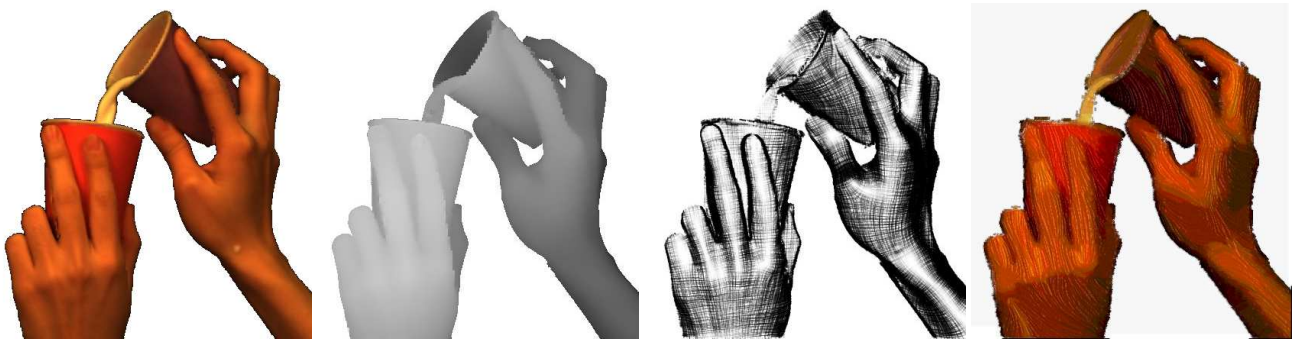

Figure 1: *Non-photorealistic rendering of a 2.5-D video (only one frame shown here).* From left to right: color input, depth input, hatching style, and painterly rendering.

Abstract

In recent years considerable interest has been given to non-photorealistic rendering of photographs, video, and 3D models for illustrative or artistic purposes. Conventional 2D inputs such as photographs and video are easy to create and capture, while 3D models allow for a wider variety of stylization techniques, such as cross-hatching. In this paper, we propose using video with depth information (2.5-D video) to combine the advantages of 2D and 3D input. 2.5-D video is becoming increasingly easy to capture, and with the additional depth information, stylization techniques that require shape information can be applied. However, because 2.5-D video contains only limited shape information and 3D correspondence over time is unknown, it is difficult to create temporally coherent stylized animations directly from raw 2.5-D video. In this paper, we present techniques for processing 2.5-D video to overcome these drawbacks, and demonstrate several styles that can be created using these techniques.

Keywords: non-photorealistic rendering, video, depth maps, computer vision.

1 Introduction

While computer graphics research has traditionally focused on methods for generating realistic images, non-photorealistic rendering (NPR) has recently attracted a lot of attention. NPR can be used to enhance an image in an illustrative yet concise way, or to give an image greater impact through an artistic style. Gooch and Gooch's book [2001] has an excellent survey of NPR techniques.

Automatic NPR techniques have generally been classified based on the type of input they operate on. One class of techniques uses 2D images as input, either stills or videos. These methods operate in image space, placing primitives such as paint strokes on a virtual canvas. A second class uses 3D models as input. They may apply primitives to a virtual canvas or directly to the surface of the 3D model. The placement of the primitives is often based on shape information, such as curvature.

Both 2D and 3D inputs have their advantages and disadvantages. Capturing images or video of complex scenes is relatively easy. However, these inputs are impoverished in that they usually lack strong depth cues. Some illustrative styles, such as cross-hatching, depend heavily on knowledge of the object's shape for guiding the placement of rendering primitives. Such styles require additional 3D information for effective rendering.

On the other hand, 3D models contain perfect shape information, so a wider variety of NPR techniques can be applied to them. Furthermore, correspondence between points on an animated model is often known, making temporal coherence easier to achieve. However, creating and animating 3D models of complex objects is difficult and time-consuming.

In this work, we combine the benefits of 2D and 3D inputs for stylization by using a different kind of input: video with depth information at each pixel, or *2.5-D video*. (The term 2.5-D video is used because only the 3D locations of the parts of the scene visible to the camera are known.) Capturing 2.5-D video is potentially easier than constructing complex 3D models, while at the same time containing enough information about shape to allow for a wider variety of artistic styles. High quality depth information can be acquired in several ways. ZCams[1] for capturing depths are commercially available, and recent systems have been developed for capturing high spatial and temporal resolution shape using off-the-shelf components (e.g., the space-time stereo system of Zhang *et al.* [2003]).

Unfortunately, in raw 2.5-D videos of complex scenes, objects are not segmented, depths are typically noisy, and 3D correspondence over time is unknown. As a result, techniques for rendering 3D models cannot be directly applied to raw 2.5-D videos. In this paper, we present methods for converting raw 2.5-D video to a stylization-friendly form. This work has several technical contributions. First, we present a method for finding correspondence information appropriate for NPR from a noisy sequence of depth maps. Second, we describe a technique for defining a smoothly varying directional field that is meaningful for stylization over a time-varying surface, which builds on existing techniques for static meshes. Third, we show how we can apply two styles, cross-hatching and painterly rendering, to the processed 2.5-D video.

[1] http://www.3dvsystems.com/products/zcam.html

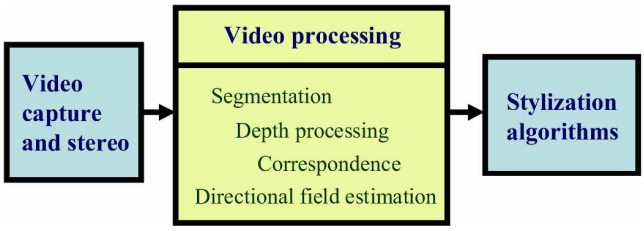

Figure 2: *System overview.* First, a set of input videos is captured, and a 2.5-D video is created by using stereo to estimate depth. The video is then processed to prepare it for stylization. Finally, a stylized video is created.

2 Prior work

Many automatic algorithms for NPR fall into two categories: those that operate on images or video, and those that operate on 3D models. There is a history of work on stylizing images and video for an artistic or illustrative effect. One area of research has focused on automatically processing images with parameterized style filters. Hertzmann [1998] uses curved brush strokes applied at several different scales to render still images in styles such as impressionism and pontillism. This work was extended to video by Hertzmann and Perlin [2000], which uses optical flow to update brush strokes over time. Hays and Essa [2004] present a system which also uses optical flow, but which improves temporal coherence by constraining brush stroke properties over time.

While these systems rely on image gradients to guide stroke placement, Salisbury *et al.* [1997] present a different technique in which a directional field is supplied by the user. Given a reference image, this can be used to create pen-and-ink illustrations from images. In contrast, in our work, the directional field is automatically generated from the input depth maps.

Wang *et al.* [2004] transform a video into a spatio-temporally coherent animation. Their system is semi-automatic, requiring a user interaction to extract semantically meaningful objects before an NPR style can be applied. Agarwala *et al.* [2004] combine user-specified contours with edge and shape preserving functions to facilitate rotoscoping (after which the video can be stylized).

In [Raskar et al. 2004], a camera with multiple flashes is used to locate depth discontinuities in images. These depth discontinuities could then be used for different stylization operations, such as enhancing silhouettes. In our work, we assume that depths are known for each pixel, giving us information which can be used to create an even greater range of effects.

There has been a separate body of work on creating stylized renderings from 3D objects. Hertzmann and Zorin [2000] presented a techniques for creating line-art illustrations from 3D models, and Praun *et al.* [2001] demonstrate a hatching style that can be applied in real time. We draw on these techniques for our stylization algorithms, extending them to work for surfaces where the complete geometry is unknown and where the shape is deforming over time.

There has also been work on constructing models of static and animated objects from range data, both from sparse scans and from video. An effort that stood out involves constructing 3D models of many large statues from range data [Levoy et al. 2000]. In [Zhang et al. 2003], an animated face model was created by fitting a template to range data. In our work, we do not construct an explicit model of objects from the 2.5-D video; instead, we intend our technique to be used for scenes that are difficult to model, and only estimate correspondences between points in neighboring frames.

3 Outline of approach

Our system for processing and rendering 2.5-D video consists of three main stages, shown in Figure 2. We first capture multiple videos of a scene using the stereo rig of Zhang *et al.* [2004]. We then use spacetime stereo [Zhang et al. 2004] to estimate depths from the videos. This results in a sequence of color video frames with depth information for each pixel (except for holes due to errors in the stereo estimation and where points were occluded in one view). We assume that the video was captured with calibrated cameras, so that given an image location (x, y), and a depth z, we can compute the 3D location of the corresponding point in space.

Second, we process the video to prepare for stylization. There are four parts to this processing stage. We start by segmenting the 2.5-D video into foreground and background layers. Next, we process the depth maps by filling in holes and smoothing to reduce noise (for our results on synthetic scenes, we omit the segmentation and processing steps). We then estimate a dense correspondence between the foreground layers of each pair of successive video frames, and simultaneously define a spatially and temporally smooth directional field over the video.

Finally, a stylization algorithm is selected, and a stylized animation is created using the processed video.

4 2.5-D video processing

In this section, we describe the processing stage of our system, which consists of four parts: segmenting the video into objects, processing the depth maps to get them into a more useful form, estimating correspondences between the shape of the objects in neighboring frames, and creating a time-varying directional field to guide stroke placement.

4.1 Video segmentation

Video segmentation can be a very difficult task. Wang *et al.* [2005] and Li *et al.* [2005] have developed effective semi-automatic systems for extracting semantically meaningful objects from video. In our work, we simplify the segmentation problem by filming objects in front of a black background, and using color and depth thresholding to segment the foreground layer.

4.2 Processing the depth maps

The depth maps that are produced by active stereo often suffer from holes and noise. We process the depth maps to make them more useful for stylization, and also compute differential information that is useful for the following stages of our system. First, we fill holes in the foreground layer by simply interpolating information from the hole boundary into the hole. Second, we apply a bilateral filter to each depth map independently to reduce noise while preserving depth discontinuities. Finally, we estimate surface normals and the directions of principle curvatures and their magnitudes at every pixel in the foreground layer, using the technique of Hameiri and Shimshoni [2002].

Figure 3: *Result of depth processing.* From left to right: raw input depth from the *hand* video, thresholded and smoothed depth, needle map (computed surface normal distribution). Note that both depth maps are rendered with a light source for clarity.

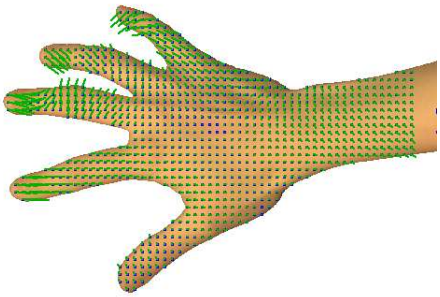

Figure 4: *Result of estimating correspondence.* The computed 3D flow between the two successive frames of the *hand* sequence (superimposed on the color image), shown as a needle map.

After processing, the representation of the 2.5-D video that is visible to the rest of the system is a set of video frames, each with a foreground mask. In addition to color, each pixel in the foreground layer stores a depth, a surface normal, and the principle curvatures and their directions.

4.3 Estimating shape correspondence

The next step is to compute a dense correspondence between successive video frames, which is necessary for creating temporally coherent animations. For 2D video input, this correspondence is known as optical flow. For 2.5-D or 3D input, this correspondence is between 3D points, and is called *scene flow* [Vedula et al. 1999].

A number of algorithms have been proposed for computing correspondences between 3D shapes. Some, such as those proposed in [Allen et al. 2003] and [Sumner and Popović 2004], are useful for estimating correspondences between shapes which have similar structure but which are otherwise quite different. These methods typically require a set of manually chosen correspondences. Our method is more similar to that of [Zhang et al. 2004], except that we compute correspondences between video points themselves without the use of a base or template mesh, and we do not explicitly use optical flow to constrain the scene flow—instead, color consistency is an implicit component of our optimization.

To compute the scene flow between a pair of successive depth maps, we consider them as two 3D points sets, S_i and S_{i+1}. First, we estimate a rigid transform between S_i and S_{i+1} using the iterated closest point algorithm [Besl and McKay 1992], obtaining a rotation R and a translation $\mathbf{t}$. Second, for each point $\mathbf{v}_j \in S_i$ we estimate a displacement vector $\mathbf{d}_j$ that minimizes an energy function $E = E_{\text{prox}} + \lambda E_{\text{smooth}}$ (we set λ to 16 in all our experiments). E_{prox} measures the proximity in *distance, normal, and color* between the displaced points in S_i and the points in S_{i+1}:

$$E_{\text{prox}} = \sum_{\mathbf{v}_j \in S_i} \rho\left(\min_{\mathbf{v}_k \in S_{i+1}} \{D(\mathbf{v}_j, \mathbf{v}_k)\}\right)$$

$$D(\mathbf{v}_j, \mathbf{v}_k) = ||\mathbf{v}_k - \mathbf{d}_j(\mathbf{v}_j)||^2 + \gamma||\hat{\mathbf{n}}_k - R\hat{\mathbf{n}}_j||^2 + \zeta||\mathbf{c}_k - \mathbf{c}_j||^2$$

where $||\cdot||$ is the L_2 norm, $\mathbf{d}_j(\mathbf{v}_j) = R\mathbf{v}_j + \mathbf{t} + \mathbf{d}_j$, $\hat{\mathbf{n}}_j$ and $\hat{\mathbf{n}}_k$ are the original unit normals associated with $\mathbf{v}_j$ and $\mathbf{v}_k$, and $\mathbf{c}_j$ and $\mathbf{c}_k$ are vertex colors represented as RGB triplets (where $r, g, b \in [0, 1]$). The weights γ and ζ control the influence of the normal and color distances; in our experiments, we set γ to 4 and ζ to 10 for the *face* videos (described in Section 6), and to zero for the other videos. The normal and color terms in D help to constrain the flow and reduce the "swimming" of points on the surface.

To improve the robustness of the correspondence estimation to occlusions, we use Tukey's robust estimator ($\rho(x)$ in the equation for E_{prox}. In addition, we treat points of S_i that, upon displacement, are occluded by a point of S_{i+1} as special cases; these points contribute a small constant to E_{prox}. This modification helps keep occluding objects (such as the arms of the dancer in the *samba* video) from "pushing" occluded points around.

To evaluate E_{prox} efficiently, we use the approximate nearest neighbor package of Arya *et al.* [1998].

E_{smooth} expresses our assumption that the objects we capture deform in a continuous way, i.e., that the displacement field varies smoothly over the object, while respecting depth discontinuities:

$$E_{\text{smooth}} = \sum_{(\mathbf{v}_j, \mathbf{v}_k)} \frac{||\mathbf{d}_j - \mathbf{d}_k||^2}{||\mathbf{v}_j - \mathbf{v}_k||^2}$$

with $\mathbf{v}_j$ and $\mathbf{v}_k$ being 4-connected neighbors in S_i. To find a minimum of E, we initialize the displacements to zero vectors, and use the LBFGS-B gradient descent algorithm described in [Zhu et al. 1997].

4.4 Defining a directional field

Most automatic NPR systems that apply brush strokes as a rendering primitive must define a directional field on their input domain in order to guide the orientation of the strokes. A directional field over a domain D is a function f from D to the set of directions, and is different from a vector field in that additive inverses $\mathbf{v}$ and $-\mathbf{v}$ represent equivalent directions. In cases where it is desirable to avoid singularities in directional fields, directional fields with 90 degree

symmetry are used. In this section, we describe our method for defining a directional field over the time-varying extracted objects.

For image stylization, the directional field is often defined in terms of the image intensity gradient. In Hertzmann's work on painterly rendering [Hertzmann 1998], the directional field is defined to be orthogonal to the image gradient, while in the work of Hays and Essa [2004], a smooth field is produced using radial basis functions to interpolate the field from strong gradients. In the system proposed by Salibury *et al.* [1997], the user supplies a direction field to guide the stylization of the input image.

In our case, we define the directional field over the foreground pixels in the 2.5-D video, but derive the field automatically in terms of shape, rather than color, following the work of Hertzmann and Zorin [2000]. In that work, it was argued that for pen-and-ink style illustrations, it is desirable that the directional field follow the directions of principle curvature where the curvature is high in one direction, and be approximately geodesic elsewhere. These guidelines were also used in Praun *et al.* [2001]. We adopt similar principles in using shape to constrain the field direction; however, we extend this work to *deformable scenes*.

4.4.1 Constraining the direction of the field

To address the issue of creating a directional field over a single 2.5-D video frame, we use a similar optimization framework as in [Hertzmann and Zorin 2000]. At each pixel $\mathbf{p}$ of the depth map, we define a *shape strength* $w_s(\mathbf{p})$ and *shape direction* $\mathbf{v}_s(\mathbf{p})$. The shape strength $w_s(\mathbf{p})$ is set to the difference between the principle curvatures κ_1 and κ_2 of the surface at $\mathbf{p}$: $w_s(\mathbf{p}) = |\kappa_1(\mathbf{p}) - \kappa_2(\mathbf{p})|$. The shape direction $\mathbf{v}_s(\mathbf{p})$ is set to the tangent vector where the magnitude of the curvature at $\mathbf{p}$ is largest.

To find the desired field, we solve for an angle $\theta(\mathbf{p})$ at each foreground pixel $\mathbf{p}$ by minimizing an energy function with two terms: $E = E_{\text{field}} + \alpha E_{\text{shape}}$ (α is set to 4 in our work). In [Hertzmann and Zorin 2000], an energy term for smooth fields was defined:

$$E_{\text{field}} = -\sum_{\text{all edges } (\mathbf{p},\mathbf{q})} \cos n((\theta(\mathbf{p}) - \phi_{pq}) - (\theta(\mathbf{q}) - \phi_{qp})),$$

where ϕ_{pq} is the direction of the projection of the edge $(\mathbf{p},\mathbf{q})$ onto the tangent plane at $\mathbf{p}$ (the angles θ and ϕ are all with respect to an arbitrary reference direction in the tangent plane fixed for each point), and n is set to 4 to solve for a field with 90 degree symmetry. We use the same expression for our E_{field} term, using $n = 2$ or $n = 4$, depending on whether 180 or 90 degree symmetry is desired.

The E_{shape} term assign a cost to differing from the field $\mathbf{v}_s$ weighted by the shape strength, i.e.,

$$E_{\text{shape}} = -\sum_{\text{all points } \mathbf{p}} w_s(\mathbf{p}) \cos n(\theta(\mathbf{p}) - \psi_s(\mathbf{p})),$$

where the angle $\psi_s(\mathbf{p})$ is the direction of $\mathbf{v}_s(\mathbf{p})$ with respect to the reference direction at $\mathbf{p}$, and n is set as with E_{field}.

4.4.2 Creating a temporally coherent field

If we were to apply the above optimization independently to each frame in the 2.5-D video, the result would likely lack temporal coherence, because the directional field is very weakly constrained at points that are far from high-curvature areas of the surface. To achieve temporal coherence, for all frames $i > 1$, we add a third term E_{time} to the energy function E described in Section 4.4.1, which acts to smooth the field with that of the previous frame:

$$E_{\text{time}} = -\sum_{\mathbf{p}^{(t)}} \cos n(f(\theta(\mathbf{p}^{(t-1)})) - \theta(\mathbf{p}^{(t)})).$$

$\mathbf{p}^{(t-1)}$ is the point corresponding to $\mathbf{p}^t$ in the previous time step, and $f(\theta(\mathbf{p}^{(t-1)}))$ is the new direction of $\theta(\mathbf{p}^{(t-1)})$ after the warp from frame $t-1$ to t. The new energy function becomes $E = E_{\text{field}} + \alpha E_{\text{shape}} + \beta E_{\text{time}}$; β is set to 1 in our work.

5 Rendering algorithms

The final stage of our system uses the information derived in the processing stage to produce a stylized rendering. Currently, we support two styles: cross-hatching and painterly rendering.

5.1 Hatching

Hatches can be used in pen-and-ink illustrations to suggest tone and form, by controlling hatch density, intensity, and direction. Hatching has been explored for static 3D models (e.g., by Hertzmann and Zorin [2000] and Praun *et al.* [2001]). We extend previous work to 2.5-D video.

For this style, we make the hatch the fundamental rendering unit; we represent each individual hatch as a "particle" in 3D, and track each hatch through the depth map sequence. As in [Praun et al. 2001], we create a set of discrete levels of hatches, each with a different density, and render the appropriate level at each surface patch based on lighting intensity. Each level has a base hatch spacing which defines the approximate distance in image space between neighboring hatches in that level.

One desired property of a hatch animation algorithm is that approximately the same image space density of hatches in each level is maintained throughout the animation. If we were to simply create a set of hatches in the first frame and track them through the video, this property would not hold, since regions of surface might expand, contract, or become foreshortened, and previously occluded regions might appear. Alternatively, if we created an entirely new set of hatches for each frame, this property would hold, but temporal coherence would be lost.

Instead, for each level, we create two sets of hatches in each frame: a new set H_{new} with the desired density, and the set of hatches tracked from the previous frame (H_{track}). We then merge H_{new} and H_{track} into a single set of hatches. The idea behind the merging step is that we want to keep as many of the hatches from H_{track} as possible to maintain temporal coherence, while removing enough and filling in the gaps with new hatches to maintain a uniform density.

To do the merging, we assign each hatch in H_{new} to zero or one hatches in H_{track} based on proximity. Each assigned hatch of H_{new} is then moved to the location of its corresponding hatch in H_{track}, and each unassigned hatch of H_{new} that is too close to one of the assigned hatches (where the distance threshold is based on the density of hatches at the current level) is removed. The assignment is computed greedily: the closest pair of hatches are added to the assignment (and removed from the candidate pool) until the distance between the closest pair reaches a threshold (in order to keep hatches in H_{new} from straying too far from their original locations). This approach results in small local variations in the hatch density, and in the appearance and disappearance of hatches, but we observe that the results look fairly natural.

To render the hatches, the user sets the desired hatch width and length, as well as the lighting direction. Then, for each hatch, we use Euler's method to integrate the directional field at that point and "grow" a curve. This resulting set of points is rendered as a piecewise linear line segment. The intensity of a hatch depends on its level and the dot product between the surface normal at the hatch point and the light direction. As in [Praun et al. 2001], the intensity is computed in such a way that the densest level of hatches are rendered with full intensity in the darkest surface regions, and that the sparsest level in the brightest regions, and transitions between each hatch level are smooth.

5.2 Painterly rendering

Painterly rendering has been explored largely for stylizing images and video. We have created a painterly rendering technique that closely follows that of [Hertzmann 1998], using curved brush strokes applied at different resolutions, but we use the shape, rather than color, to guide the placement of brush strokes. The rendering primitive we use for this style is a *stroke*, which contains a set of control points, a radius, a color, and an alpha value.

To create a painterly rendering of a video, the user first defines a style by setting the number of brushes to be applied and the radius and maximum length of each brush. The canvas is then painted in layers, from coarsest to finest, with curved brush strokes, as in the technique of [Hertzmann 1998]. We use the color image blurred with a Gaussian kernel whose standard deviation is proportional to the stroke radius as the reference image for each layer, and finer strokes are only used to fill in detailed regions that were not accurately rendered in the coarser layers.

Rather than creating a new set of strokes for each new image in the video, we achieve more temporal coherence (as in [Hertzmann and Perlin 2000] and [Hays and Essa 2004]) by using the correspondence information to track the control points of each stroke from one frame to the next. Next, we search for strokes that can be removed, either because they have become two small or too stretched over the course of the tracking, or because the coarser levels of the canvas now accurately render the area of that stroke. Last, we add strokes to each layer where the canvas and the blurred reference image for that layer disagree. We follow the technique of Hays and Essa [2004] in improving temporally coherence by gradually fading strokes in and out. For rendering, we also use their technique of treating a texture as a height map with which lighting computations can be performed.

6 Results

We have tested our algorithm on several data sets, both synthetic and real. The synthetic data sets demonstrate the accuracy of our correspondence and directional field estimation, while the real data sets show the reliability of our system given noisy data.

We created two synthetic 2.5-D videos, *samba*, a video of a samba dancer and *face*, a video of an animated face, by generating a set of depth maps from two animated 3D models viewed from a fixed viewpoint. Color information was unavailable for the *samba* sequence, so we did not use the color channel of the 2.5-D *samba* video. We also captured two real 2.5-D videos, *hand*, a video of a flexing hand, and *milk*, a video of milk being poured into a cup. Figure 1 shows an example frame from the *milk* video.

Figure 5 shows sample frames from animations created from the *samba*, *hand*, and *milk* videos, created using the hatching rendering style. Figure 6 show frames from the *hand* and *face* videos rendered in a painterly style. The full animations can be seen in the accompanying video.

7 Discussion

Our system has a few limitations. First, it currently does not distinguish between different objects in the scene, so the correspondence and directional fields of two objects can become correlated in unnatural ways. It would be more desirable to allow the user to segment the video into multiple objects and treat each independently. Second, unlike with a 3D model, which can be viewed from any direction, the viewpoint of a stylized rendering of 2.5-D video must be the same as the one that was captured. Similarly, it is difficult to edit the captured motion. These limitations are shared by normal video. Finally, 2.5-D video is currently somewhat difficult to capture, requiring either expensive equipment or a complicated setup. However, we predict that advances in stereo algorithms and capture technology will reduce this difficulty in the future.

8 Summary

In this paper, we have described our system for taking a raw 2.5-D video, processing it to create a more useful representation, and creating a stylized output video. In addition to the system itself, our contributions include a method for deriving a directional field over a time-varying depth map and algorithms for applying several NPR effects to 2.5-D video. Our system illustrates that even though 2.5-D video lacks complete shape information and correspondence, 2.5-D video can be used to create stylized animations depicting shape without the use of explicit 3D models.

We showed that a combination of depth map processing, temporal correspondence, and temporally coherent directional field is key to making seamless stylization of 2.5-D video possible. As part of future work, we would like to use both color and shape to estimate the directional field and provide tools to the user for editing the fields. In addition, we plan to apply more NPR styles, such as line drawing, to 2.5-D video.

References

AGARWALA, A., HERTZMANN, A., SALESIN, D., AND SEITZ, S. 2004. Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH Conference Proceedings*, 584–591.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of all body shapes: Reconstruction and parameterization from range scans. In *SIGGRAPH Conference Proceedings*, 587–594.

ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., AND WU, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM 45*, 6, 891–923.

BESL, P. J., AND MCKAY, N. D. 1992. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence 14*, 2, 239–256.

GOOCH, B., AND GOOCH, A. 2001. *Non-Photorealistic Rendering*. A K Peters, Ltd.

HAMEIRI, E., AND SHIMSHONI, I. 2002. Estimating the principal curvatures and the Darboux frame from real 3D range data. In *IEEE Inter. Symp. on 3D Data Proc. Visual. Trans.*, 258–267.

HAYS, J. H., AND ESSA, I. 2004. Image and video-based painterly animation. In *Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 113–120.

HERTZMANN, A., AND PERLIN, K. 2000. Painterly rendering for video and interaction. In *Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 7–12.

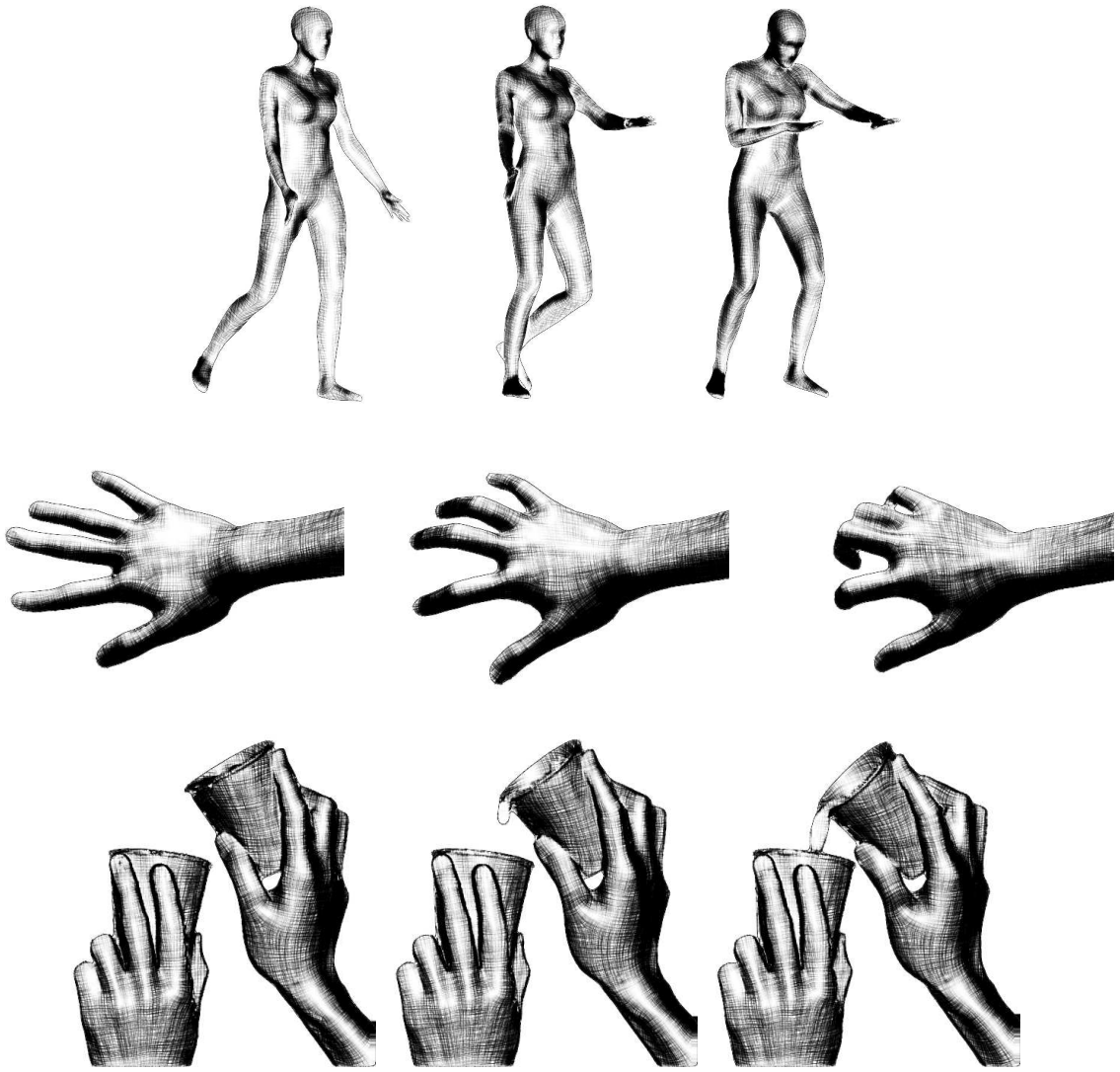

Figure 5: *Sample frames from hatched renderings of the* samba*,* hand*, and* milk *videos.*

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *SIGGRAPH Conference Proceedings*, 517–526.

HERTZMANN, A. 1998. Painterly rendering with curved brush strokes of multiple sizes. *Computer Graphics 32*, Annual Conference Series, 453–460.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The Digital Michelangelo Project: 3D scanning of large statues. In *SIGGRAPH Conference Proceedings*, 131–144.

LI, Y., SUN, J., AND SHUM, H.-Y. 2005. Video object cut and paste. *SIGGRAPH Conference Proceedings 24*, 3, 595–600.

PRAUN, E., HOPPE, H., WEBB, M., AND FINKELSTEIN, A. 2001. Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001*, 579–584.

RASKAR, R., TAN, K., FERIS, R., YU, J., AND TURK, M. 2004. Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging. In *SIGGRAPH Conference Proceedings*, 679–688.

SALISBURY, M. P., WONG, M. T., HUGHES, J. F., AND SALESIN, D. H. 1997. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH Conference Proceedings*, 401–406.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3, 399–405.

VEDULA, S., BAKER, S., RANDER, P., COLLINS, R., AND KANADE, T. 1999. Three-dimensional scene flow. In *Int'l Conf. on Computer Vision (ICCV)*, vol. 2, 722–729.

WANG, J., XU, Y., SHUM, H.-Y., AND COHEN, M. F. 2004. Video tooning. In *SIGGRAPH Conference Proceedings*, 574–583.

WANG, J., BHAT, P., COLBURN, R. A., AGRAWALA, M., AND COHEN, M. F. 2005. Interactive video cutout. *SIGGRAPH Conference Proceedings 24*, 3, 585–594.

ZHANG, L., CURLESS, B., AND SEITZ, S. M. 2003. Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 367–374.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: High-resolution capture for modeling and animation. In *SIGGRAPH Conference Proceedings*, 548–558.

ZHU, C., BYRD, R. H., AND NOCEDAL, J. 1997. Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software 23*, 4, 550–560.

Figure 6: *Sample frames from painterly renderings of the* hand *and* face *videos.*