Video Cubism

Allison W. Klein, Peter-Pike J. Sloan, R. Alex Colburn, Adam Finkelstein, Michael F. Cohen
Microsoft Research and Princeton University

Microsoft Research #MSR-TR-2001-45 — Princeton University, Dept. of Computer Science #TR-637-01

Abstract

We present new non-photorealistic (NPR) rendering tools for video. Inspired by the Cubist and Futurist art movements that questioned previous notions of space and time within paintings, we view an input video as a space-time cube of data, rather than a series of static frames. Our tools process the video as a whole to produce a set of *stroke-solids*, units for rendering that appear over multiple frames in the resulting NPR video. Depending on stylistic considerations and video content, each stroke solid is encoded with parameters such as location, size, curvature, orientation, color values, or other relevant information. A non-photorealistic video is constructed interactively by compositing slices of the stroke solids. These slices, or *strokes*, are rendered as multi-textured sprites. The textures for each stroke may be derived from a pre-defined texture atlas and/or drawn from the underlying video itself. Many of the parameters that define the appearance of a stroke are set at runtime. This provides the artist with a wealth of interactive aesthetic controls for modifying the final result. Benefits of our work include interactive stylistic flexibility and aesthetic control, and methods for exploiting the full temporal information present in a video sequence when designing the stroke solids. Finally, this work extends painterly rendering of video beyond the impressionist styles previously explored to Cubist, Futurist, and Abstract styles.

Keywords: Animation, Paint Systems, Temporal Aliasing, Video, Non-realistic Rendering

1 Introduction

Researchers have developed a variety of tools to modify two-dimensional images, giving the results a "painterly" or hand-created look in such diverse styles as impressionism [7, 8], pen and ink [15], watercolor [2], and engraving [12]. Existing NPR rendering methods are essentially filters that take in an image (and possibly some additional processing parameters) and return an altered, NPR version of the original. Because these methods were developed specifically for static images, when they are applied to video sequences on a frame-by-frame basis the results generally contain undesirable temporal aliasing artifacts. To overcome these artifacts, Litwinowicz [11] used optical flow information to push paint strokes from one frame to the next in the direction of pixel movement. Hertzmann *et al.* [9] created each successive frame of the video by first warping the previous frame to account for optical flow changes and then painting over areas of the new frame that differ significantly from the frame before. Nevertheless, these approaches are really local optimizations.

We present a new set of non-photorealistic (NPR) rendering tools for processing video. We begin by treating the video as a space-time volume, or *cube* of image data. Rather than defining two-dimensional strokes successively on each video frame, we create a set of parameterized *stroke-solids*. Stroke-solids are units for rendering that appear over multiple frames in the resulting NPR video. Stroke-solids may be specified automatically from the underlying video, interactively with authoring tools, or through a combination of both interactive and automatic techniques. Frames in the resulting NPR video are generated at run-time by successively slicing the stroke-solids with a plane as it passes through the video volume. Each stroke-solid "slice" provides a set of parameters sufficient to define a textured sprite that is composited onto the output frame. The texture itself may be derived either from a pre-defined texture atlas or from a local region in the underlying input video.

Treating the video as a whole offers advantages over previous techniques. With our approach, one can make local choices based on global knowledge from the full video sequence. For example, because stroke-solids exist over a stretch of time, they can be constructed to maximize certain properties, such as smoothness in color or orientation, over their lifetime. In addition, stroke-solids enable anticipatory effects such as automatically fading strokes in or out.

Interactivity is another benefit of our approach. Stroke-solids separate a stroke's relationship to the information contained in the original video sequence from the stroke's final rendered appearance. This allows interactive modification of the mapping between stroke-solid parameters and the individually rendered strokes. Thus, we are able to provide the artist with a wide array of interactive controls over the final look of the video.

We demonstrate the benefits of our approach to impressionist styles to compare our methods with previous NPR tools. In addition, the final contribution of our work is that it extends non-photorealistic rendering of video beyond impressionist styles to Cubist, Futurist, and Abstract imagery. In fact, the new NPR techniques for video presented here are inspired by the Cubist and Futurist art movements. Just as the early twentieth-century Cubist painters sought to decompose a scene into an arrangement of geometric figures, we seek to decompose a video stream into a set of geometric entities. In addition, the Cubist fascination with capturing a scene from different viewpoints or times motivates us to render one or several moments from the video sequence simultaneously, which is made possible by our volumetric approach.

The remainder of this paper is organized as follows. First, we discuss some general principles of processing the video as a space-time volume. We then demonstrate how our approach can be used to implement Impressionist, Cubist, and Abstract painting styles, as well as the photo mosaics of David Hockney. However, since the work presented here is fundamentally about moving images, the results are best seen on the accompanying video. After presenting some key performance and implementation details, we conclude with proposed areas of future work.

2 Considering Video as 3D Data

In this section, we present the idea of a video as a space-time volume and then discuss, at a high level, the relationship between this volume and stroke-solids.

2.1 Video Cube

A video is a sequence of images that has been discretized in two-dimensional space (pixels) and in time (frames). Because there is a large body of work in computer graphics devoted to discretized 2D image processing techniques, it seems only natural that most of the non-photorealistic video processing work thus far has consisted of running these techniques on each individual frame. The only concession to the third dimension, time, has been the use of optical flow [11, 9]. One of our main goals is to treat time more consistently with the spatial dimensions.

We do this by considering the video as a three-dimensional cube of data. One could, for example, view slices of the cube that are not necessarily orthogonal to the time axis (Figure 1). For instance, if we looked at slices that are parallel to the time axis, we would see motion (either of scene elements or the camera) as changes across each scanline. Each scanline represents a single pixel's trace over time within the original video. In computer vision, such images are sometimes referred to as *epipolar diagrams* [3]. Visualizations of time-sequenced imagery from viewpoints other than on the time axis are not unique to this project. Fels and Mase [4] presented an interactive system for passing arbitrary cut planes through the video data volume. *Multiple-center-of-projection* images [13] and *multiperspective panoramas* [18] may also be considered two-dimensional (though non-planar) slices of a video in which the camera is moving. One of our contributions is to apply the underlying ideas of a video volume and non-standard cutting planes towards non-photorealistic rendering of video, and more specifically towards Cubist and Futurist styles. Just as the Cubist and Futurist art movements of the early 20^{th} century mixed space and time within single images, we leverage the ability to slice the input video in non-standard ways to achieve similar effects.

2.2 Stroke-Solids

We define a *stroke-solid* as a three-dimensional unit for rendering that exists over a sequence of frames in the output NPR video. Stroke-solids are generated through a combination of automatic processing of the input video and input provided by the user. The specifics of the automated process and the parameters available to the user differ depending on the desired artistic style. In the next section we describe four example styles that we have implemented.

The stroke-solid is designed to be the basis for an interactive rendering system. As such, it has user-specified parameters that are not known until render time. Thus, the stroke-solid is not fully defined until render time. The stroke-solid's volume is defined by the pixels it generates over the course of the output NPR video (Figure 2).

Because of computational costs, a stroke-solid may be partially defined during a preprocessing stage. For example, a 3D curve representing the central skeleton of the stroke-solid over time, may be defined before run-time based on optical flow detected in the input video. Similarly, other aspects of the video, such as color or color gradient, may be recorded along the stroke-solid skeleton.

As previously stated, many decisions about a stroke-solid's final appearance are left until runtime to allow for experimentation by the artist. In general, these runtime decisions determine the mapping from the values determined during the preprocess to the different appearance choices provided by each style. In our implementations, described in more detail in the following sections and accompanying video, each style renders the slices of the stroke-solids as solid, textured, or multi-textured primitives. The final scale, orientation, position, choice of textures, and texture coordinates can all be interactively modified to produce a variety of visual effects.

Thus, at render-time, the stroke-solids provide flexibility in three ways. First, the values recorded in the preprocess can be modified with simple UI elements. Second, the stroke-solid forms an abstraction separating the structure of the input video from its final, non-photorealistic rendering. This means that the set of rendered strokes to which we map the stroke-solid can be changed dynamically. Third, since the stroke-solids are themselves continuous, they can be sampled at any arbitrary point during runtime. For example, the output frame rate can vary independent of the input video's frame rate.

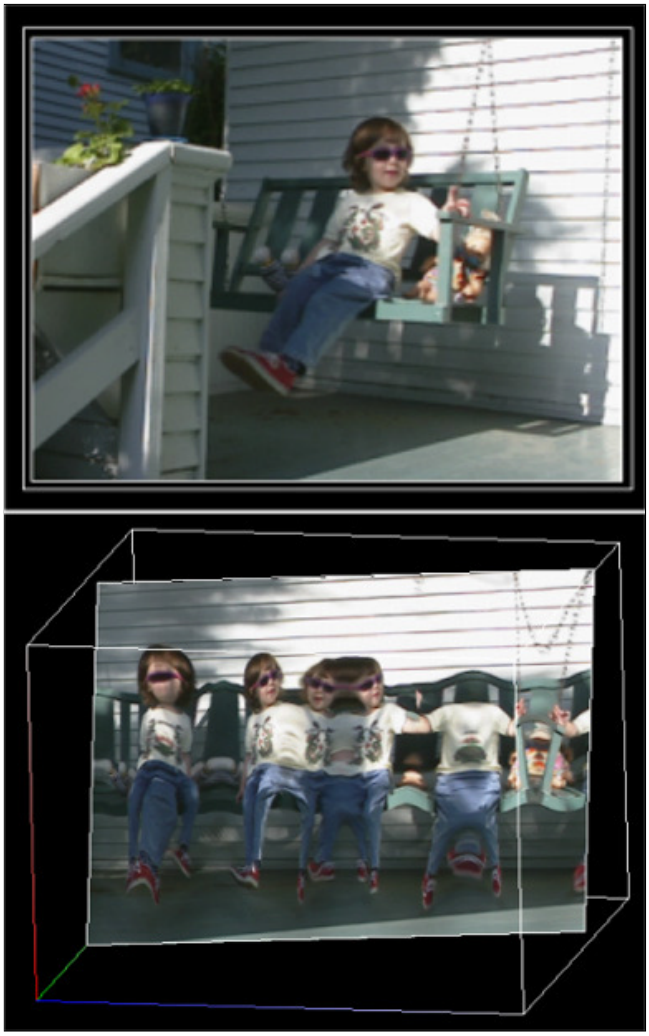

Figure 1: Video Cube

3 Stylized Rendering

In this section, we discuss a number of styles we used to explore the concept of stroke-solids. We chose these examples based on the 19^{th} and 20^{th} century paintings and styles that inspired this work, paintings by Monet (Impressionism), Picasso (Cubism), Duchamp (Futurism), Mondrian (Abstract), and Hockney (Photo Mosaics). For each style we will describe the preprocess required,

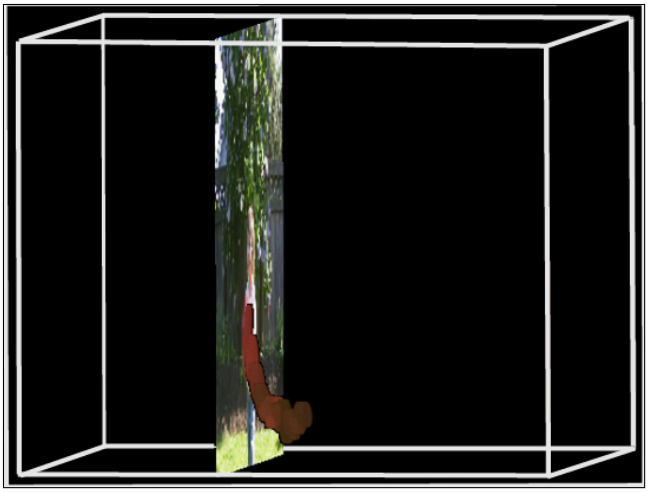

Figure 2: A stroke-solid that tracks the red ball

the interactive parameters provided to the user, and show some results.

It should be noted that the styles described below are only point samples in a very wide space of possibilities that open up by considering the video as a data volume. Therefore, these styles and their particular implementation details are presented more as examples of what one can do within this space rather than as a comprehensive coverage of the possibilities.

3.1 Impressionism

With the advent of the camera in the mid-19^{th} century, painters were liberated from simply portraying the physical world. By the 1880's, artists such as Monet and van Gogh began to use their brush strokes as visual objects in and of themselves. Rather than creating a realistic depiction of a scene, the resulting images provided the artist's *impression* of the scene, and thus the Impressionist movement in art was born.

In our impressionist implementation, we first perform a preprocess defining a set parameterized stroke-solids that are further manipulated at runtime. The preprocess begins by calculating additional information at each pixel of the input video: an optical flow vector (by searching a small window around the pixel for a matching region in the next frame), an orientation normal to the image gradient (a combination of horizontal and vertical 3×3 Sobel filters), and curvature (the change in gradient across a small window around the pixel). Because optical flow algorithms are not very robust, particularly when confronted with quickly-moving objects, we also manually adjust the optical flow information to track large, fast moving objects.

Given the color values, flow, orientation, and curvature at each pixel, we generate each stroke-solid's path. We randomly seed the volume with a set of initial positions (in the case of a $320 \times 240 \times 240$ video we used $40,000$ initial seeds). Next, for each seed, we use the optical flow data to grow the path of the stroke-solid both forward and backward in time from the seed position. At each growth step, we record the position, orientation, and curvature, and color, using bi-linear interpolation for sub-pixel accuracy. The size parameter is set to one. Whenever the color difference between neighboring samples exceeds a small threshold (a distance of 50 in the 255^3 RGB cube), the stroke is broken into two distinct pieces at that point. In some cases, after the stroke-solid has been generated, we may also taper the size parameter from 1.0 down to 0.0 at both ends of the stroke-solid to provide gradual stroke introduction and removal at run-time.

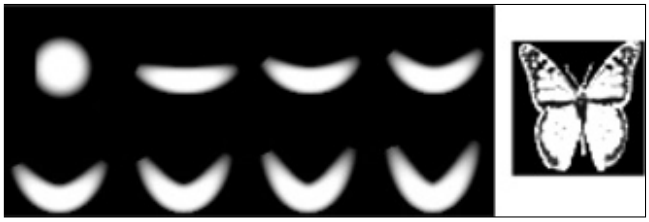

Figure 3: Two of the texture atlases used to render the impressionist style.

A final aspect of the preprocess involves creating a set of texture atlases by hand. These atlases, used at runtime to render the stroke-solids, contain both grayscale and alpha images of the strokes (figure 3); the colors associated with the individual stroke-solids will be used to apply color to these textures. In our examples, we have experimented with atlas images that look like jellybeans, leaves, and butterflies. In the case of the butterflies, we can modulate the width over time to create dynamic wing-flapping at runtime.

At runtime, a plane is passed through the video cube. Each frame is constructed on the fly by compositing one sprite for each stroke-solid intersecting the plane. An artist has a variety of interactive controls over the final look of the impressionist video:

- Choosing the texture atlas
- Drawing fewer stroke-solids, either for performance reasons or to make the result less busy looking
- Modifying a stroke-solid parameter's curve by smoothing (e.g., smoothing the variation in orientation will make the strokes rotate less during their lifetime) or adding noise (the opposite effect of smoothing)
- Scaling a stroke-solid parameter's curves, either by adding or multiplying a constant factor
- Gamma correcting the red, green, or blue color channels

Figure 4 shows a variety of results from a single preprocess run on a short video of a girl catching and throwing a ball. By varying both the interactive settings and the texture atlas, an artist can quickly modify the resulting video as it plays. Figure 5 depicts a similar style based on underlying videos of a school a fish and of a woman talking. Here, you can see additional effects of our pre-process, such as how the stroke-orientations lie normal to image gradients.

3.2 Cubism

Cubism, pioneered by Pablo Picasso and Georges Braque in the early twentieth century, created a new artistic vocabulary of multiple perspectives, interlocking planes, and fractured, flattened masses capable of "articulat[ing] the complex, fragmentary experiences of a new era[14]." The cubist decomposition of the image into geometric shapes was a precursor for much of what followed; painting became more and more abstract as the century progressed. In addition, time and space became twisted and stretched as images began to depict multiple viewpoints in both time and space. Duchamp's "Nude Descending a Staircase" is probably the best known work of this style, which is sometimes referred to as Futurism. We have created two rendering styles to capture some of the essence of the Cubist and Futurist movements. The first style automatically decomposes the video volume into 3D Voronoi tiles. The second style allows the artist to define swept sheets and generalized cylinders, which then subdivide the video volume. We will describe each in turn.

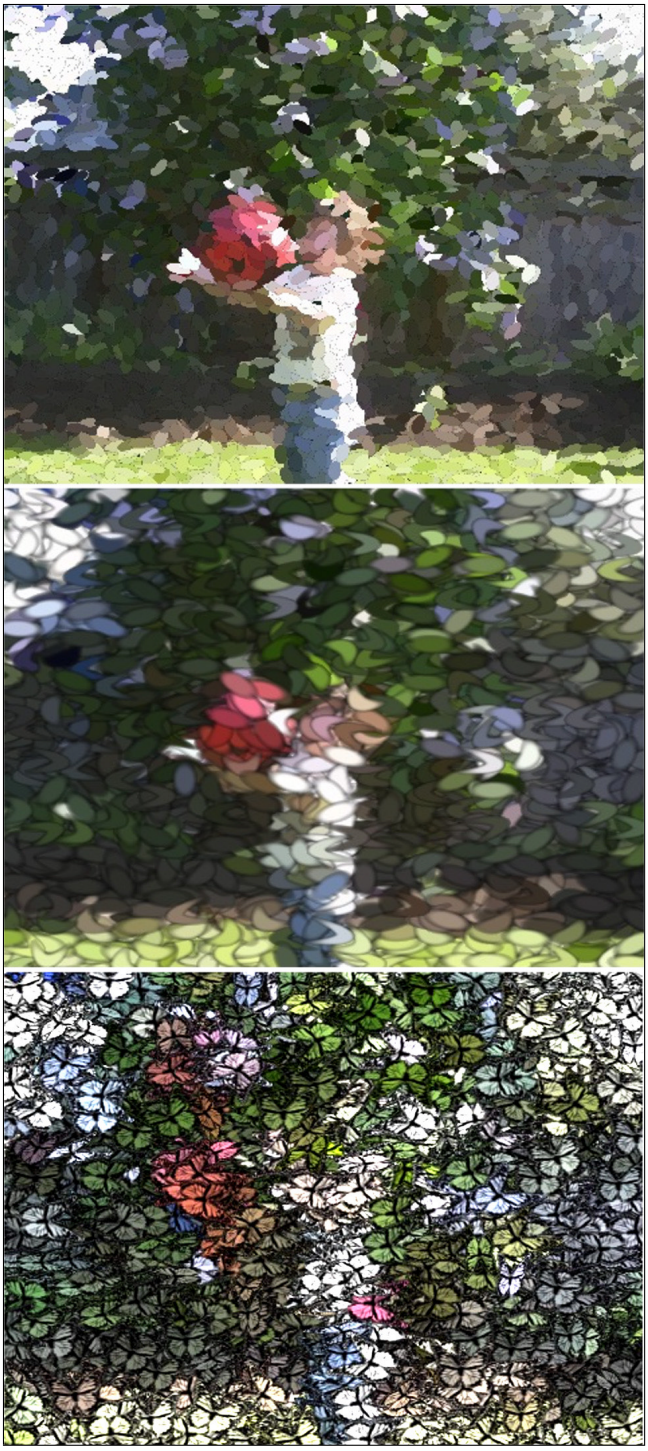

Figure 4: Impressionist-style images from a single preprocess.

Figure 5: Results from other impressionist-style preprocesses. Note the stroke-orientation information present in this style.

3.2.1 Diamonds as 3D Voronoi Cells

Many image manipulation programs provide the ability to generate tiled images from input images. The underlying algorithm is based on work by Haeberli [7] in which tiles are actually 2D Voronoi tiles, and the tile colors are sampled from the Voronoi seed point locations.

We take the same idea and generalize it to 3D: a video volume can be decomposed into a 3D Voronoi diagram. Then, each slice of the volume produces a tiled image. A hardware-based solution for rendering these successive slices consists of drawing, for each point, a hyperboloid representing the distance function between the point and the image plane. (This hyperboloid degenerates to cones when the point lies on the plane) The z-buffer then leaves intact just the region of the plane closest to the point. Hoff *et al.* [10] describe the algorithm in detail.

In addition to providing a means for decomposing our volume into geometric shapes, a significant benefit to this approach is that 3D Voronoi cells implicitly provide a high degree of temporal coherence. This coherence occurs because each cell will (in general) begin with a small cross section, grow, and then shrink smoothly before vanishing as the plane passes through it.

With Voronoi cells as our foundation, we must now address several implementation issues:

- How many seed points to use
- Where to place these seeds to best create 3D tiles representing the video source
- How to color the tiles
- What is the optimal hyperboloid tessellation for rendering
- How to manage video access
- How to achieve the above points in real time so that we can give the artist interactive control.

We discuss video access in Section 4, and the number of points (or 3D Voronoi tiles) is an aesthetic decision and is therefore left to the artist at run-time. Placing the seeds within the volume is more problematic. One might want to have them scattered evenly throughout the volume, or more likely, one might want more points located in regions of high *importance*. One measure of importance is the local variance of the color values in the video volume. We allow the artist to choose the final importance of a region as a blend between a desire for a uniform distribution and one determined by local variation. To create an importance-based distribution of points we perform an importance weighted stratified sampling strategy [16]. This results in a well spread set of points with a higher density in either spatial or temporal regions of change. Point selection is run whenever the artist changes the number of desired points or changes the importance function (i.e., whenever the artist changes the blend between a desire for uniformity and a distribution based on local variation). The color for each Voronoi cell is taken as the local mean color in the video. The entire point selection process takes 2 to 3 seconds for a 10 second video at 320×240 resolution.

We now have all that is needed to render slices of the 3D Voronoi diagram. However, since the Voronoi diagram is never explicitly represented, given a slicing plane, there are a number of unkowns we must solve at real-time. For example, which points will contribute to coloring the slice? Also, how large a section of the associated hyperboloid needs to be drawn to not leave holes yet avoid redundancy? (If all the points are chosen and the associated hyperboloid is drawn until it covers the frame, the system will grind almost to a halt.) Unfortunately, the optimal answers are dependent on the number of Voronoi cells and their distribution. We thus provide sliders to be set at runtime determining (a) the thickness of a *slab* of time surrounding the current slice for which to render the Voronoi cells, and (b) the radius of the portion of the hyperboloid to render. The user simply moves the sliders upwards until there are no holes. With a small bit of practice this is very intuitive.

Figure 6 shows some results from rendering Voronoi cells. In the first image of the woman's face and the third image of the girl swinging on a branch, the importance function was set to emphasize local detail. Note the high density of points around the face edge and near the girl's back, and the lower density in the ground area. The second image has a more even distribution of points. We have also used a second bubble-like texture (shown in the corner of the middle image) to modify each cell in the bottom two images.

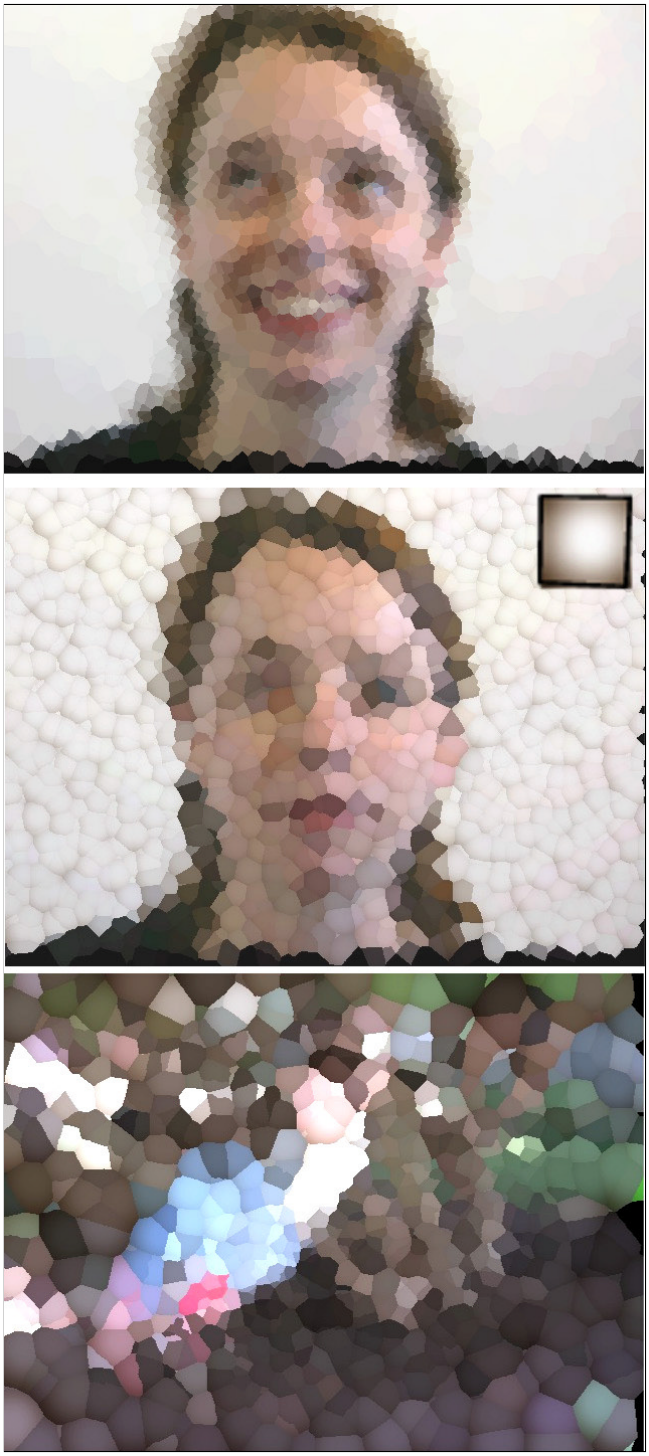

Figure 6: Voronoi rendering results

3.2.2 Shards: swept surfaces and generalized cylinders

A second style, also inspired by early Cubist works, enables the artist to subdivide the screen into discrete areas or *shards* at each frame.

We have implemented a small authoring tool in which the artist can interactively define a series of swept surfaces, each dividing the video volume into two half-spaces. The intersections of the many half-spaces form a set of "shards" tiling the volume. The interface

involves specifying a few lines contained in the swept surface. For example, the artist might start at the first frame in the video and specify a line that divides the screen into two half-planes. The intent might be to have this line follow the edge of a moving object in the scene. Then moving forward to a new keyframe, say, half way through the video, the artist is presented with the line specified for the first frame. The artist can move this line to a new position to follow the feature. Finally, moving to the last frame of the video, the line is repositioned again. A swept surface is generated by linearly interpolating these three lines through time. Thus the artist will see the line move from frame to frame, passing through the positions that were set at the keyframes. In a similar fashion, the artist can specify any number of swept surfaces. At each frame of the video, the interpolated lines subdivide the screen into a number of shards that smoothly vary through time. We also provide an equivalent tool for laying down ellipses to define a swept elliptical cylinder.

At runtime, as the cutting (image) plane passes through the swept surfaces, the individual shards are discovered on the fly in the following manner: For each frame, shard vertices are found at line crossings, which break the cutting lines into line segments. The segments are in turn linked together to form convex polygons, by tracing around each polygon in counter-clockwise order (always turning left at intersections). We also compute the centroid and area of each shard to be used as rendering parameters. Finally, any swept ellipses are superimposed as a new polygon on top of the shards.

At this point, there are many possibilities of how to process the shards for the final output. For example, we could pick up a color at the centroid of the shard and use that to color the whole shard as in the Voronoi style. Instead, we will use the original video to texture each shard. We provide the artist with a number of ways to modify each shard, including:

- Zooming each shard
- Modifying its time association with the input video
- Using two video streams as input
- Modifying the video texture by multiplying it with a second texture, as in the Hockney style.

One can also set the zoom factor and time variation to be a function of the distance from one of the swept surfaces, allowing these quantities to vary across each shard.

Figure 7 shows some results of using swept surfaces to decompose the image. Each shard has been scaled and shifted in time as a function of the shard size and proximity to main dividing line. Note that in the traffic scene, a second pre-blurred video stream is used as the texture for those shards to the left of one line. Each line moves in time (e.g., one follows the wiper blade), thus the polygons change at each frame time as does the zoom factor and/or time shift.

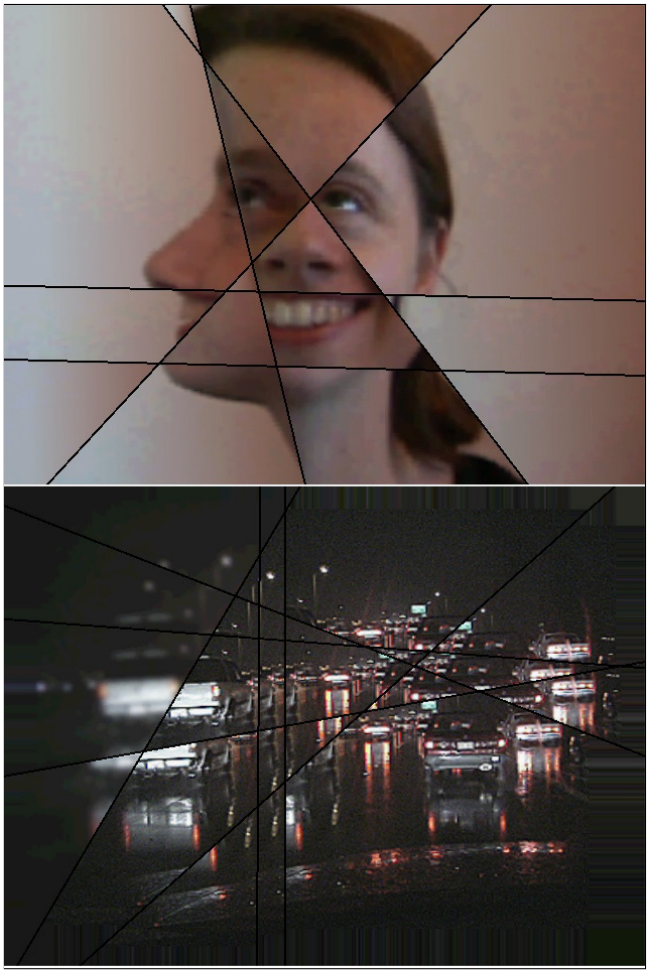

Figure 7: Shards rendering results

3.3 Abstraction

If the Cubist movement began the process of breaking down the image, the Abstract movement completed the process. Representation was completely abandoned in favor pure compositional balance, color, and design. One of the earliest pioneers and intellectuals in this field was Piet Mondrian. Mondrian is best known for his series of abstract paintings beginning in the 1920's depicting vividly colored rectangles offset by thick black lines. The starkness and simplicity of his work was a major influence within the art and design world, inspiring everyone from fashing designers(see Yves St. Laurent's Mondrian dress) to architects. The simple, almost algorithmic nature of his paintings has also led to "Mondrian machines," available as Java applets on the internet. But Mondrian's form of abstraction did not arise *ex nihilo*. Mondrian spent many years painting solitary trees, gradually adopting a more Cubist approach, and eventually discarding the representation itself in pursuit of a sense of pure balance [5, 17].

Motivated by Mondrian's work, we seek to turn video into a mobile series of colorful, rectuangular compositions. In addition, just as Mondrian was inspired by how trees decompose space, Mondrian has inspired us to decompose video volume into a kd-tree.

Kd-trees typically used to hierarchically decompose space into a small enough number of cells such that no cell contains too much complexity. This decomposition occurs by recursively slicing through space with half-planes, and these half-planes are usually axis-aligned. While kd-trees provide a fast way to access input objects by location, we use them to subdivide the video volume into rectangular sub-volumes where each sub-volume contains approximately the same amount of importance while also maintaining an even aspect ratio (if desired). As mentioned above in relation to the Voronoi style, one measure of importance is the local color variance in the video volume. We first construct a 3D summed-area-table[1] of importance. This provides the input for dynamically constructing the kd-tree at runtime.

As the video cube is traversed at runtime, each kd-tree cell is colored with a constant color drawn from a low pass filtered version of the video. This color is optionally remapped to increase the saturation. As new cells are added to (or removed from) the output, they are smoothly transitioned in (or out) to avoid popping.

We provide an artist with a number of interactive controls at

runtime:

- Number of cells: defines the desired number kd-tree leaf nodes
- Aspect ratio: how cubical each cell should try to be
- Saturation: a color remapping from video to output
- Jitter: randomly re-chooses higher level split points in the kd-tree. (The leaf nodes are still created according to the importance function.)
- Line thickness

Figure 8 shows some results of applying our Mondrian style to the video of a talking woman. We feel the top image with few cells is reminiscent of a Mondrian painting. The video shows the smooth variation from the top image to the bottom one by sequentially growing the number of leaf nodes in the kd-tree. This depicts more detail from the underlying video stream. Although our initial inspiration came from the complete abstraction of Mondrian, we found the transformation from abstract images to more representational ones visually intriguing.

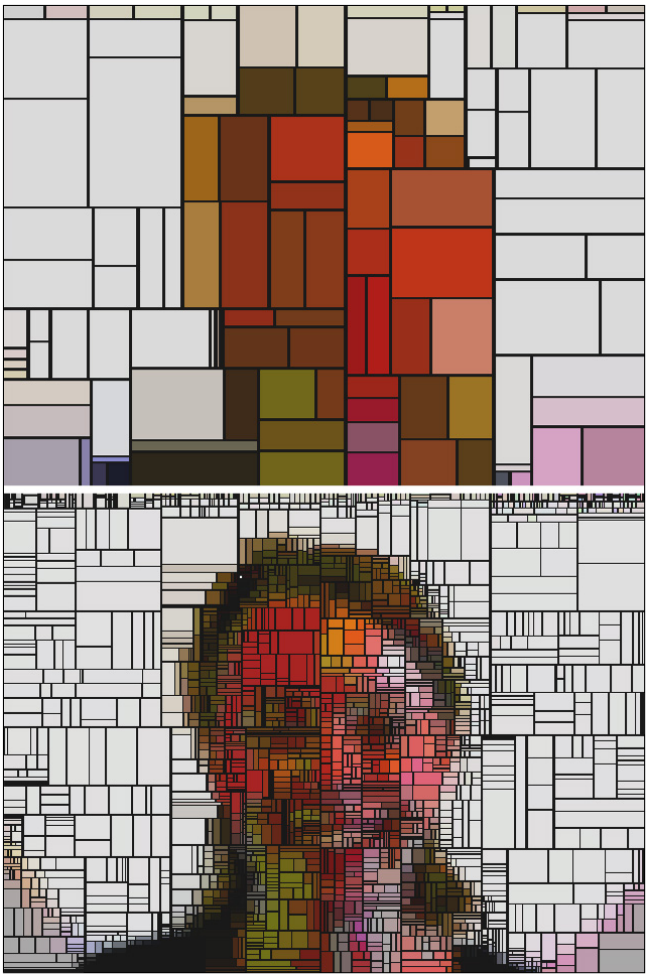

Figure 8: Mondrian rendering results

3.4 Photo Mosaics

In the 1980's, pop-artist David Hockney turned his camera towards common scenes and objects, snapping multiple images of scene

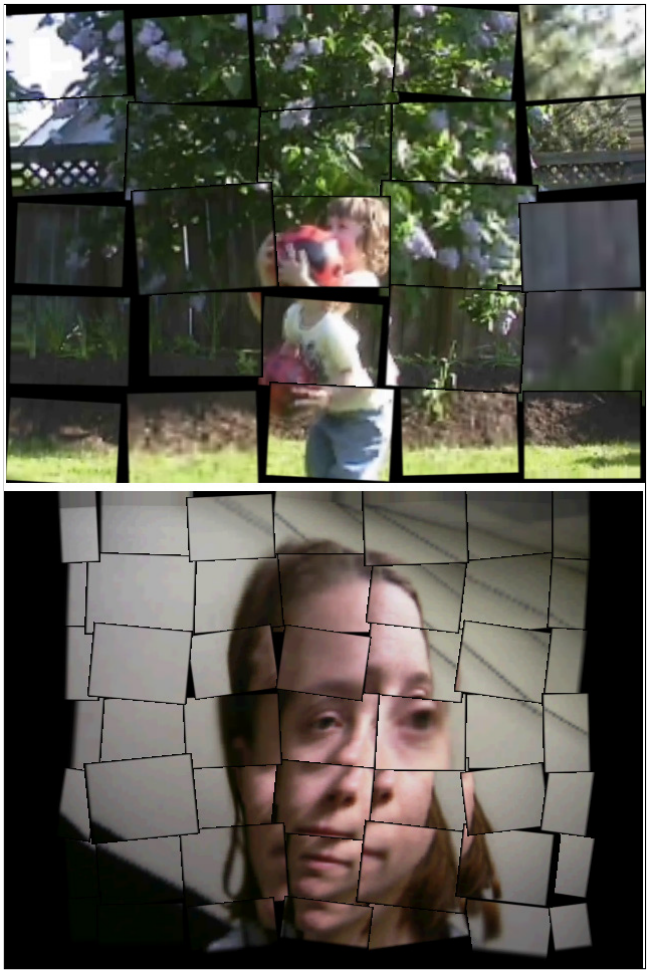

Figure 9: Photo mosaic results

details. He then composited numerous fragments of these images, creating an image mosaic. Inspired by these photo montages, we provide a tool enabling the artist to subdivide the video into a series of small, textured tiles where each tile draws its texture directly from a local region of the video.

Because the stroke-solids are drawn from a regular grid in space, and thus are implicitly defined, there is no preprocess required for this style. The interactive controls we provide to the artist at runtime include:

- Grid size (x and y): defines the number of output stroke-solids or tiles. For example, a 5 by 5 grid results in 25 tiles.
- Scale: defines the size of the output tile relative to the input grid rectangle. Tiles scaled to size less than one will leave gaps between it and its neighbors. These gaps are filled with black or optionally with the original video sequence.
- Offset (x and y): defines how the tile will be shifted in space relative to its grid position.
- Rotate: defines the tile's rotation.
- Zoom: defines the size of the source texture, specifically, the ratio of the size of the source texture from the video to the output tile. A small source texture mapped to a larger output tile will have an effect like a magnifying glass.

- Time offset: defines where in time the input texture is drawn from. This allows different tiles to depict somewhat different points in time in any single output frame.

For each of the above parameters(except for grid size), we actually enable the artist to set both a mean value and a variance for a random perturbation. Each tile can also be multi-textured (multiplied) with a second texture to create a border and/or an overall painterly or textured look.

Figure 9 shows some results from the photo-mosaic style. Each tile is manipulated at runtime to shift, rotate, scale, zoom, and shift its position in time. Note the multiple points in time depicted in the image of the girl throwing the ball. In the case of the woman's face, we first processed the underlying video to show multiple points of view at a single time. The original video stream was captured by a camera moving relative to the woman's head.

4 Notes on performance and implementation

Each of the images in this paper and all the examples in the accompanying video were rendered in real time on a 733MHz PIII PC with an Nvidia GeForce2 GTS graphics card. Depending on the specific style and particular settings chosen, the frames of the NPR video rendered at between 15 and 200 frames per second. That said, the timings on many of the styles can be seriously degraded by some of the choices possible at runtime. Clearly, by including too many stroke solids in any of the styles, the system can become polygon bound. More commonly, the system slows down due to limitations in fill rate. For example, by setting the radius too large for the hyperboloids associated with each seed point of the Voronoi style creates a high depth complexity and thus a fill rate too high for interactive speeds. Also, some operations, such as reseeding the Voronoi patterns or computing a summed-area-table to guide the kd-tree divisions in the Mondrian style, take a few seconds, but these are run infrequently during a design session.

The preprocess for the impressionist style is expensive, on the order of a few hours depending on video length. In particular, this is due to the optical flow determination. However, once this precomputation has been carried out, a wide array of looks can be explored interactively.

Many of the implementations require careful engineering to be able to achieve interactive speeds. Some of the important implementation steps include:

- **Evaluating the parameters** for the impressionist style. Each parameter's function over time is fit to a quadratic B-spline with identical knot spacings. Since all stroke solids will be fit to this one B-spline basis set, the fit matrix can be inverted once (using SVD) and used to fit all the stroke solids. At runtime, given the single knot spacing, the system only has to compute 3 B-spline coefficients per frame and reuse these for every parameter and every stroke solid. If the user requests a parameter's function to be smoothed, it is fit to a B-spline basis set with fewer knots and then reprojected back to the original higher number of control points. Optional noise functions that can be added to any parameter have the same structure.

- **Efficiently performing time offsets** for each stroke solid as in the Hockney and shards styles. In order to draw texture from more than one frame of video, we have implemented a *ring buffer* to handle the data access. Given a maximum of N frames for the time perturbation, holding the current frame plus the N frames before and after the current frame in texture memory provides all the source textures for any out frame. As each frame falls more than N frames in the past, its texture in the ring buffer is replaced with a new frame drawn from N frames in the future. In addition, to maintain better memory locality, each frame in the ring buffer is looped through and used to texture any stroke solids that touch it, rather than looping through the stroke solids and finding the appropriate frame to draw the texture from.

- **Deciding when and where to subdivide the kd-tree** for the Mondrian style. We first compute a 3D summed-area-table [1] of importance. This provides the values needed to quickly evaluate the integral importance of any subregion of the video very quickly. The same table also provides the average color for any node in the kd-tree.

- **Tessellating and rendering the hyperboloids** in the Voronoi style. We rely on the algorithm described by Hoff *et al.* [10] with some added engineering. In particular we find optimal radii for a series of rings with which to tessellate the hyperboloid. We recursively determine the radius with highest error between the current approximation (starting with a cone) and the true hyperboloid and subdivide the hyperboloid at this radius and repeat until an error threshold is met. This is done in advance for twenty point-to-plane distances. At runtime we use the one closest to each points distance to the plane.

- **Finding the Voronoi seeds** within a given time distance from the current plane. We developed a 3D binned spatial data structure to hold each Voronoi seed. This provides a very fast way to find and access just those Voronoi cells that need to be rendered at each frame.

- **Determining the polygons by intersecting the swept surfaces** in the shards style. At each frame we need to find all intersections, and then connect these into individual polygons. An efficient edge based structure can be computed very quickly [6]. Robustness problems are endemic to these types of operations when more than two lines cross. We detect such problems and offset the surfaces slightly to avoid zero sized polygons at runtime.

5 Conclusion and Future Work

The central idea presented in this paper is to consider video as a whole when designing non-photorealistic rendering methods for video. From this basic idea, we develop the concept of a stroke-solid, a 3D region of the output video used as a rendering unit. We demonstrated a wide variety of styles that can be created within this framework. We chose our inspiration for the artistic styles from a series of artistic movements ranging from Impressionism to Cubism to Abstract Art.

For each artistic style, we demonstrated a runtime environment that provides the artist with a large degree of flexibility for experimentation. To support this flexibility, we perform a preprocess where necessary, while leaving as much computation to runtime as possible to allow the widest latitude for interactive manipulation. We have described a number of the underlying engineering details to make this possible on current graphics hardware.

We are currently working on several enhancements to the current system. These include:

- Adding new stroke atlases that mimic more traditional paint strokes in the impressionist style.

- Adding new interactive tools for shard definition. In addition to the cutting surfaces we want to add new generalized cylinder shapes such as triangles and rectangles. These can also be sequenced to give the artist more control over the final compositing order of the stroke-solids.

- Adding the ability to control individual stroke-solids. Currently, given the large number of stroke-solids, we perform the same or randomly determined operations to all strokes.

- Migrating Voronoi seed points along paths in time. Initially we would like to augment the simple seed points with line segments that express the seed's path through time. Individual Voronoi cells will survive longer and thus improve temporal coherence. If not exactly parallel to the time axis they can accentuate optical flow. Alternately, we could use the stroke-solid paths described in Section 3.1, thereby unifying the rendering logic for our Impressionist and Cubist styles.

Clearly, there are many more styles one could choose to address. Just as clearly, we have only begun to define the set of tools one would want to present to the artist for each of the styles demonstrated. It will only be after an artist uses with the system for some time that we will better understand the limitations of the technology. This will also help us design an effective user interface for the tools. We hope the reader shares our excitement about exploring the variety of new possibilities in the context of NPR video.

References

[1] F. C. Crow. Summed-area tables for texture mapping. *Computer Graphics (Proceedings of SIGGRAPH 84)*, 18(3):207–212, July 1984. Held in Minneapolis, Minnesota.

[2] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. *Computer Graphics (Computer Graphics (Proceedings of SIGGRAPH 97))*, pages 421–430.

[3] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[4] S. Fels and K. Mase. Interactive video cubism. In *Proceedings of the Workshop on New Paradigms in Information Visualization and Manipulation (NPIVM-99)*, pages 78–82, N.Y., Nov. 6 1999. ACM Press.

[5] V. A. Grauer. Mondrian and the dialectic of essence. *Critical Review*, 11(1), 1996. http://www.creview.com/artcrit/ac1gra.htm.

[6] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[7] P. E. Haeberli. Paint by numbers: Abstract image representations. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):207–214.

[8] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. *Computer Graphics (Proceedings of SIGGRAPH 98)*, pages 453–460.

[9] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. *Computer Graphics (Proceedings of NPAR 2000)*, pages 7–12.

[10] K. H. III, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. *Proceedings of SIGGRAPH 99*, pages 277–286, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[11] P. Litwinowicz. Processing images and video for an impressionist effect. *Computer Graphics (Proceedings of SIGGRAPH 97)*, pages 407–414.

[12] V. Ostromoukhov. Digital facial engraving. *Computer Graphics (Proceedings of SIGGRAPH 99)*, pages 417–424.

[13] P. Rademacher and G. Bishop. Multiple-center-of-projection images. *Proceedings of SIGGRAPH 98*, pages 199–206, July 1998.

[14] R. Rosenblum. *Cubism and Twentieth Century Art*. Harry N. Abrams Publishers, New York, NY, 1966.

[15] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin. Orientable textures for image-based pen-and-ink illustration. *Computer Graphics (Proceedings of SIGGRAPH 97)*, pages 401–406.

[16] P. Shirley, C. Wang, and K. Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, January 1996. ISSN 0730-0301.

[17] D. Sylvester. *About Modern Art : Critical Essays, 1948-1997*. Henry Holt & Company, Inc, New York, New York, 1997.

[18] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin. Multiperspective panoramas for cel animation. *Computer Graphics (Proceedings of SIGGRAPH 97)*, pages 243–250.