Multiperspective Panoramas for Cel Animation

Daniel N. Wood[1] *Adam Finkelstein*[1,2] *John F. Hughes*[3] *Craig E. Thayer*[4] *David H. Salesin*[1]

[1]University of Washington [2]Princeton University [3]GVSTC [4]Walt Disney Feature Animation

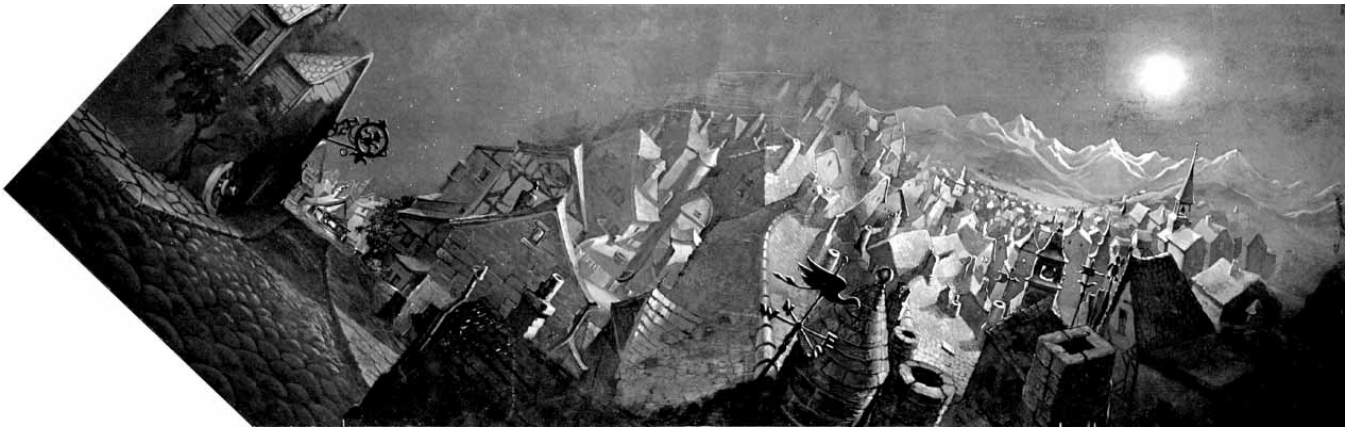

Figure 1 A multiperspective panorama from Disney's 1940 film *Pinocchio*. (Used with permission.)

Abstract

We describe a new approach for simulating apparent camera motion through a 3D environment. The approach is motivated by a traditional technique used in 2D cel animation, in which a single background image, which we call a *multiperspective panorama*, is used to incorporate multiple views of a 3D environment as seen from along a given camera path. When viewed through a small moving window, the panorama produces the illusion of 3D motion. In this paper, we explore how such panoramas can be designed by computer, and we examine their application to cel animation in particular. Multiperspective panoramas should also be useful for any application in which predefined camera moves are applied to 3D scenes, including virtual reality fly-throughs, computer games, and architectural walk-throughs.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation.

Additional Keywords: CGI production, compositing, illustration, image-based rendering, mosaics, multiplaning, non-photorealistic rendering.

1 Introduction

Walt Disney's 1940 feature animation, *Pinocchio* [14], opens with a long, continuous shot, in which the camera appears to fly over the rooftops of a small village and gradually descend into an alley facing Gepetto's cottage. This simulated 3D fly-through was actually accomplished via a stunning 2D effects shot. Instead of modeling a 3D scene, a single backdrop was painted that incorporated a kind of "warped perspective" (Figure 1). The backdrop was then revealed just a little at a time though a small moving window. The resulting animation provides a surprisingly compelling 3D effect.

In this paper, we explore how such backdrops, which we call *multiperspective panoramas*, can be created from 3D models and camera paths. As a driving application, we examine in particular how such computer-generated panoramas can be used to aid in the creation of "Disney-style" 2D cel animation. To this end, we envision using the following four-step process (Figure 2):

1. A 3D modeling program is used to create a crude 3D scene and camera path. (Since only rough geometry is required, a modeler like SKETCH [24] might provide an ideal interface.)
2. Our program takes the 3D scene and camera path as input, and outputs one or more panoramas, each with a 2D *moving window* for viewing the panorama during each frame of the animation. When viewed as a whole, the panoramas may appear strangely warped. However, when taken together, the panoramas and moving windows should produce the illusion of 3D motion along the camera path. In the rest of this paper, we will use the term *layout* to refer to the panoramas taken together with their moving windows.
3. An illustrator then uses each computer-generated panorama as a guide to produce a high-quality artistic rendering of the distorted scene, called an *illustrated panorama*. The illustrated panorama may be created with any traditional media and scanned back into the computer. Alternatively, the illustrated panorama may be created with a digital paint system directly on the computer.
4. For each frame in the scene, images are extracted from the panoramas according to the moving windows. These images are composited (together with any additional foreground or computer-animated elements) to produce the final frames of the animation.

[3]NSF STC for Computer Graphics and Scientific Visualization, Brown University Site

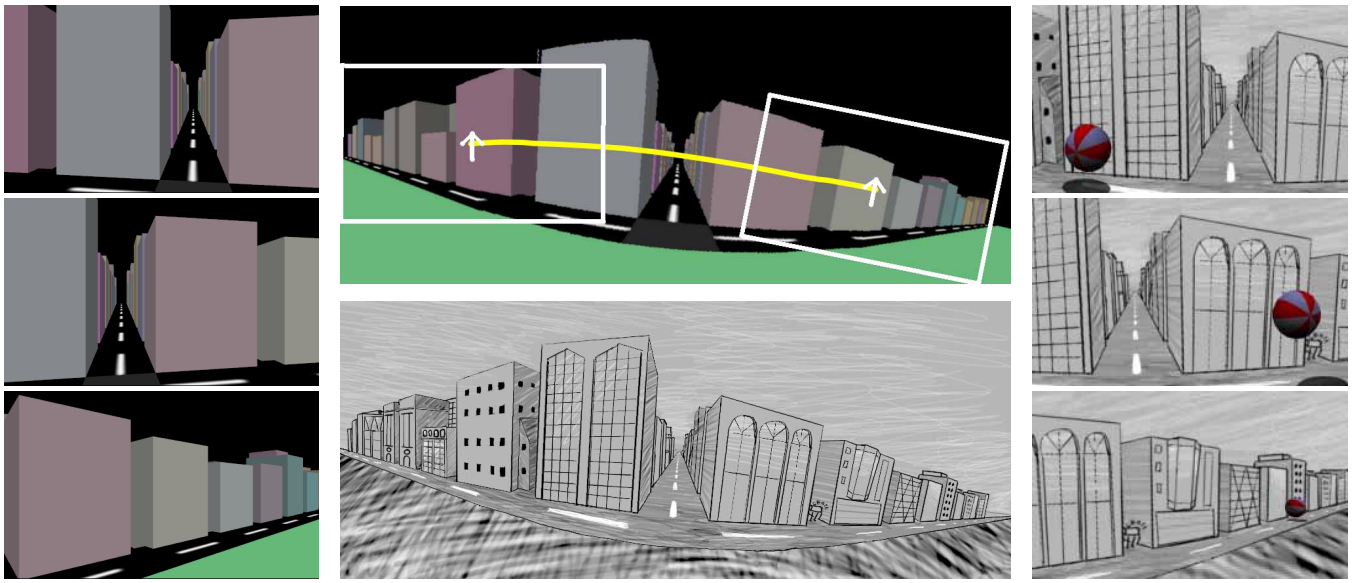

Figure 2 Pan. (a) Views from a 3D camera path. (b) Computer-generated layout. (c) Illustrated panorama. (d) Frames from the illustrated panorama with a computer-animated bouncing ball.

This process leverages the strengths of both the computer and the artist. The computer permits the use of much more complex camera paths than can be created by hand; in addition, it allows easier experimentation in designing them. The artist, on the other hand, is free to create the panorama in any artistic style, and is not limited by the availability of any particular computer rendering technique. Moreover, because the computer-generated layouts are created algorithmically from 3D models, they can be integrated with live-action or with conventional 3D computer-generated effects—something that it is extremely difficult to do with hand-drawn layouts, which often do not accurately correspond to any physical 3D scene or camera path. In addition, an automated process for creating such layouts should allow layout artists to work more efficiently and employ layouts more widely.

In addition to cel animation, the multiperspective panoramas described here should have applications to any situation in which "canned" camera moves are applied to 3D scenes, including virtual-reality fly-throughs, computer games like *Myst* [16], and architectural walk-throughs. In many internet-based applications, they may also be significantly faster to download and interact with than true 3D models, such as VRML.

1.1 Related work

The work described in this paper is related to several different threads of research in computer graphics.

First, our work is related to previous efforts on creating panoramas from multiple views [8, 20]. Our problem is in one sense simpler than that of these previous works, in that our source images are computer-generated. We therefore avoid solving the point-correspondence problem on images of real-world scenes. On the other hand, we allow for large changes in camera position and orientation across the panorama, and so we must accommodate a much greater sort of perspective distortion. Our problem is also related to Zorin and Barr's work on correcting distortion in perspective renderings [25, 26], although in our case we are concerned with the problem of making the distortions appear *locally* correct, rather than globally correct. (Much of Escher's art [6] also includes locally correct but globally distorted perspective.)

Our work also fits into the general framework of image-based rendering [3, 4, 5, 12, 15, 19], in which new views of a 3D scene are created from one or more source images. Our process differs from these previous image-based rendering approaches in that it generates a warped view (or set of views) that is optimized for the very simplest of extraction operations—that of selecting out a single rectangle for each frame. An advantage of using such multiperspective views is that the panorama created by the artist appears in the final frames in exactly the same way as it was painted—with no distortion in the shapes of the brush strokes, for example. In trade for this, however, our panoramas only allow this nice type of reconstruction along a single pre-specified path.

Another thread of research related to this paper is that of non-photorealistic rendering [10, 13, 17, 22, 23], in which 3D geometry is rendered in an artistically stylized form. The work in the paper is motivated by this same desire to create artistically rendered images of 3D geometry. However, the approach we take here is to output a flat design that can be rendered in any style, by hand, using traditional media. Alternatively, the panoramas that the program constructs can be given as input to an image-based non-photorealistic renderer, as described in Section 5.

This work is also related to previous results in automating the process of cel animation and digital compositing [7, 21], although we look at only a particular aspect of this process here—that of creating multiperspective panoramas—which has not been previously investigated.

1.2 Overview

In the next section, we approach the problem of generating arbitrary panoramas by first examining some simpler special cases. We then formulate a general solution in Section 3. We discuss our implementation in Section 4. Finally we conclude in Section 5 with some discussion and directions for future work.

2 An introduction to layouts

In this section, we describe panoramas produced in the simple cases of some basic camera moves: pan, tilt-pan, zoom, and truck [11].

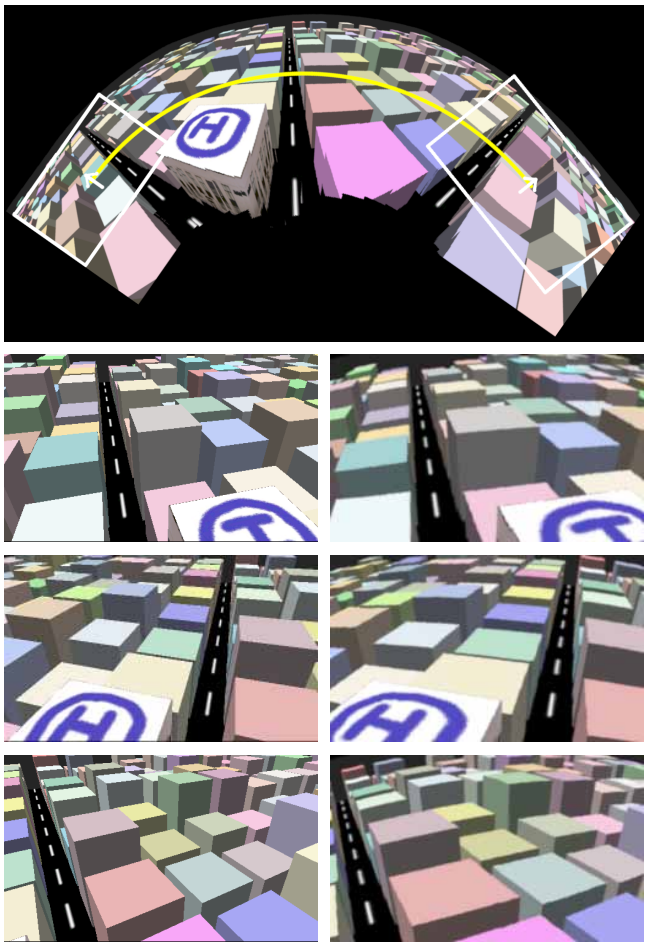

Figure 3 Tilt-pan. Computer-generated layout and frames (3D views on left, extracted frames on right).

Making layouts even for these relatively simple moves can be nontrivial.

2.1 Pan

Suppose that we wish to make a movie of a scene, taken by a camera rotating from left to right with its gaze always horizontal (a *pan*). We can ensure that the vertical centerline of each image is correct by unrolling a cylindrical projection for our panorama (as in QuicktimeVR [3]). Then the center vertical lines of extracted rectangles are perfect, but the vertical line segments on the left and right of the image appear too short. If the scene contains parallel horizontal lines, they become curved lines with vanishing points both to the left and to the right. Figure 2b demonstrates a pan panorama created by our application. With tight framing (Figure 2d) the bowed lines are not too objectionable.

2.2 Tilt-pan

As a more complex example, imagine that we are looking down and out across a city from the roof of a tall building, and wish to rotate our view from left to right. Our tripod is level, but the camera is tilted down as we pan. This *tilt-pan* requires a more complex layout.

If we simply use the cylindrical projection again, extracted rectangles from the lower portion of the cylinder will be deeply

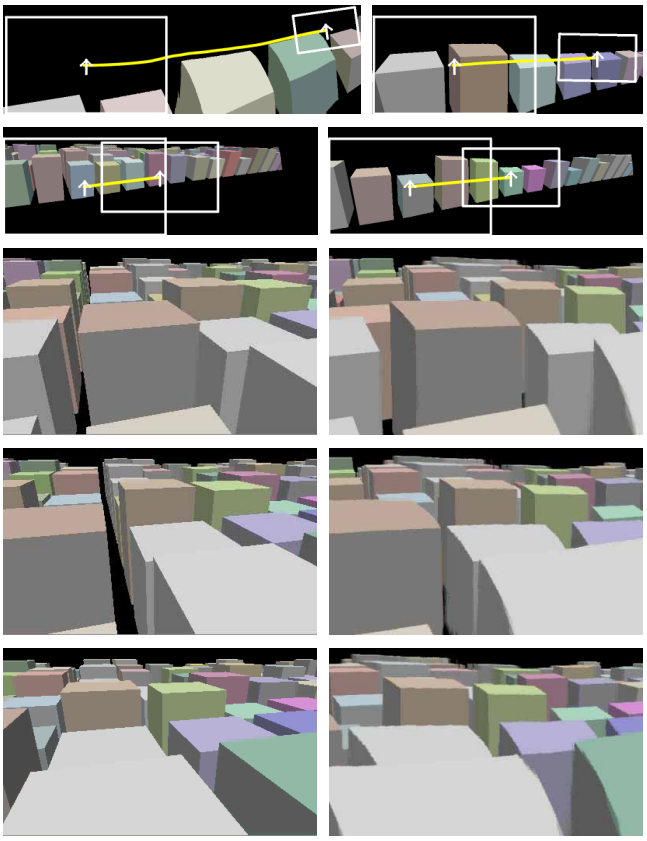

Figure 4 Truck. Computer-generated layout (top two rows) and frames (3D views on left, extracted frames on right). The four panoramas at top are composited from front to back, starting with the upper left panorama, and proceeding in clockwise order.

unsatisfactory: not only will horizontal lines become bowed so that they all sag in the middle, but the extracted images will differ from the original ones by a "keystoning" transformation. A far better approximation is a conical projection [2].

Figure 3 shows a panorama created by our application for a tilt-pan. Notice that the vertical direction is not mapped to a single consistent direction in the panorama, and that the eventual sequence of extracted rectangles rotates about the cone-point of the flattened cone.

2.3 Zoom

Now suppose that instead of panning, we want to *zoom* (change the focal length of a stationary camera). Our panorama is simply a normal image from which we extract smaller and smaller windows. In a painting the brush strokes will be enlarged as we zoom in, and it may become necessary to cross-fade to a more detailed image. For example, the opening sequence of *Pinocchio* ends with a zoom toward the window of Gepetto's cottage followed by a crossfade to a detailed closeup of the cottage. (Of course, if the final panorama is generated digitally, tools like multiresolution paint [1] can help address this problem.)

2.4 Truck

If the camera's center of projection moves, occlusions may change. Cel animation has taken two approaches to changing occlusion. One approach is to ignore it—small errors in realism may well go unnoticed. This approach works for relatively small occlusion

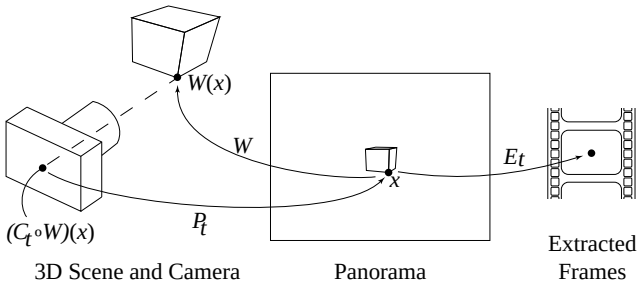

Figure 5 The coordinate systems and mappings between them.

changes (a hill in the far distance always occludes the same part of the mountain in the very far distance, for example). Alternatively, limited changes in occlusion can be suggested using *multiplaning*, in which objects in the scene are grouped roughly according to their depths, and each group appears on a separate panorama with its own moving window. The extracted frames are composited, back to front, to produce the final animation. This effect has also been widely used in computer games (*e.g.*, the backdrops of driving simulators).

In a scene with significant variation in depth, moving the camera perpendicular to the gaze direction (called *trucking*), creates the impression of objects at different depths moving at different relative speeds (called *parallax*). Figure 4 shows multiple panoramas created for a trucking scene, as well as some frames after compositing. If several objects are at the same depth, then an orthographic projection of the plane containing them can be used as the panorama with no distortion. To the extent that the scene has depth, objects in the same panorama will become distorted, as seen in the figure.

3 General formulation

In this section we describe a general solution for arbitrary camera paths, which specializes to the layouts we have already described for simpler camera moves like pan, tilt-pan, zoom, or truck. As previously stated, the goal for our layout is for the frames extracted from the panorama to match exactly the rendered images of the original camera path, which we will call *views* for contrast.

Historically, cel animators were limited to extracting rectangular sub-images from panoramas by their camera apparatus. We preserve this constraint for four reasons: it is simple; it meshes well mechanically with the existing animation process, and seamlessly with existing animations; it forces the resulting panorama to look (locally) like a traditional perspective view, making it easier for an artist to draw; and it preserves the artistic texture and composition of the panorama. Limiting extraction to rectangular windows with a fixed aspect ratio also tells us something about the relationships between neighboring frames in the resulting animation: any two will be related by a *similarity transform*—a transform involving a translation, rotation and uniform scale.

To create an image incorporating many perspectives, we begin with views of the scene taken from different perspectives and try to merge them. Regardless of how we merge them, each point $x \in \mathbb{R}^2$ of the resulting panorama corresponds to a world-space point $W(x) \in \mathbb{R}^3$ (Figure 5). For each time t, let C_t denote the map from world-space points to view points, and P_t denote the map from view points at time t onto the panorama. Let E_t be the map from a subset of the panorama to an image that is the extracted frame for time t. Finally, each point x of the panorama gets its color from a view at some particular time $S(x)$. From these definitions it follows that if $t = S(x)$, then $(P_t \circ C_t \circ W)(x) = x$.

If the extraction process produces the same picture as the original camera view, then $E_t(x) = (C_t \circ W)(x)$ for all points x in the domain of E_t. Hence $(P_t \circ E_t)(x) = (P_t \circ C_t \circ W)(x)$, which simplifies to $(P_t \circ E_t)(x) = x$ when $S(x) = t$. For points y in the domain of E_t for which $S(y) = s \neq t$, we have that $(E_t \circ P_s \circ C_s \circ W)(y) = E_t(y) = (C_t \circ W)(y)$.

If $(P_s \circ C_s \circ W)(y)$ differs much from $(P_t \circ C_t \circ W)(y)$, then the panorama will be distorted badly within the domain of E_t, and so the frame extracted at time t will look bad. In a perfect panorama (one in which the extracted frames look like the original views), $(P_t \circ C_t \circ W)(y) = (P_s \circ C_s \circ W)(y)$ for all points y in the domain of E_t such that $S(y) = s$. In this case $(E_t \circ P_t \circ C_t \circ W)(y) = (C_t \circ W)(y)$ for *all* y in the domain of E_t implying that in any perfect layout the (linear) extraction map E_t is the inverse of the linear placement map P_t. In our panorama, we therefore always choose E_t to be the inverse of P_t, since it is a necessary condition for a panorama to be perfect. For camera paths and scenes where no perfect panorama is possible, if the distortion is small the same sort of argument implies that E_t is *approximately* the inverse of P_t. Thus, if we can find a suitable rule for placing views into the panorama, then we will know how to extract frames from the panorama.

We now take our characterization of ideal panoramas and convert it from a descriptive one to a prescriptive one—one that tells us what the placement map P_t should be. We continue to be guided by the necessary condition for a perfect layout: for world points w visible at time t, $(P_t \circ C_t)(w) = (P_s \circ C_s)(w)$ for values of s near t. If we write $s = t + \epsilon$ and then expand both P and C in a first-order Taylor series about t, we get $(P_t \circ C_t)(w) \simeq ((P_t + \epsilon P_t') \circ (C_t + \epsilon C_t'))(w)$, which can be simplified to $(P_t \circ C_t')(w) + (P_t' \circ (C_t + \epsilon C_t'))(w) \simeq 0$, using the linearity of P_t and C_t. Taking the limit as $\epsilon \to 0$ gives $(P_t \circ C_t')(w) = (-P_t' \circ C_t)(w)$. In words, the rate at which w is moving across the film plane of the camera at the instant t is the negative of the rate at which the film-plane point $C_t(w)$ is being translated across the panorama. We want this true for every w, but because P_t must be a linear map, we compromise by requiring it only on average: we compute the similarity transform that is the least-squares approximation of the *optical flow*, the flow of world-space points across the image plane of the camera, and use its negative as the rate of change of the camera-placement map P_t. Choosing P_0 arbitrarily, we now find P_t by numerical integration. Since E_t is the inverse of P_t, all that is left for us to define is $S(x)$. In Section 4.2, we will describe a simple rule for $S(x)$ that works reasonably well in practice.

For sufficiently simple scenes, this formulation specializes to the layouts described in Section 2. In fact, the layouts for Figures 2 through 4 were created using the general formulation described in this section.

4 Implementation

In this section we describe our implementation of the general principles just described. We discretize the 3D camera path and render frames at finely spaced intervals. Then we find the placement transform for each of these views; finally, we select from among them in regions where they overlap.

4.1 Placement

The first view is arbitrarily placed at the origin of the panorama's coordinate system. We find a transform that places each subsequent view relative to its predecessor. The relative transform from any view $i + 1$ to i would ideally be the inverse of the optical flow between them. These transforms are composed to place each view on the panorama.

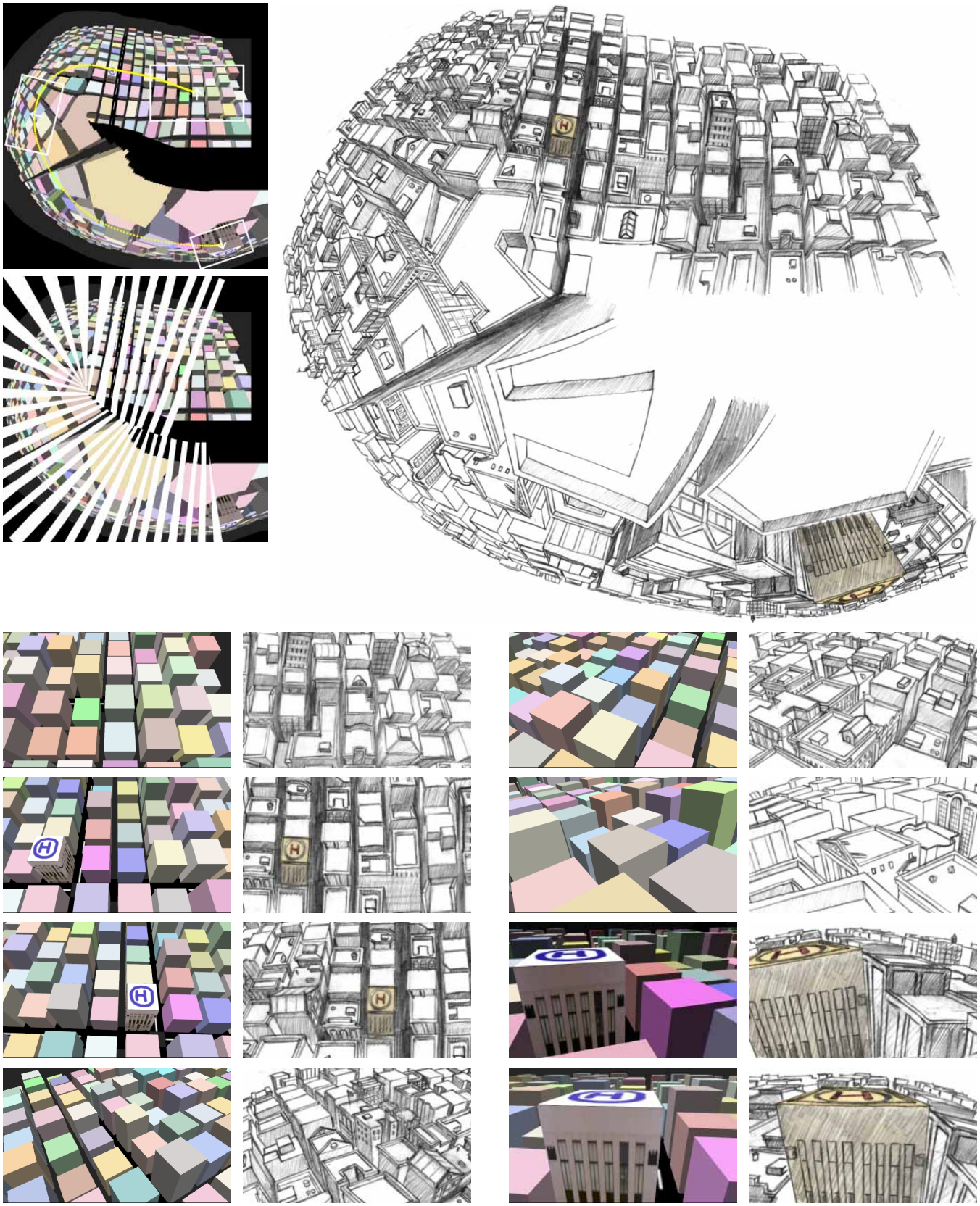

Figure 6 Helicopter scene. Top left: Computer-generated layout above Voronoi diagram. Top Right: Illustrated panorama by Ed Ghertner at Walt Disney Feature Animation. Bottom: Frames (3D views to the left of extracted frames.)

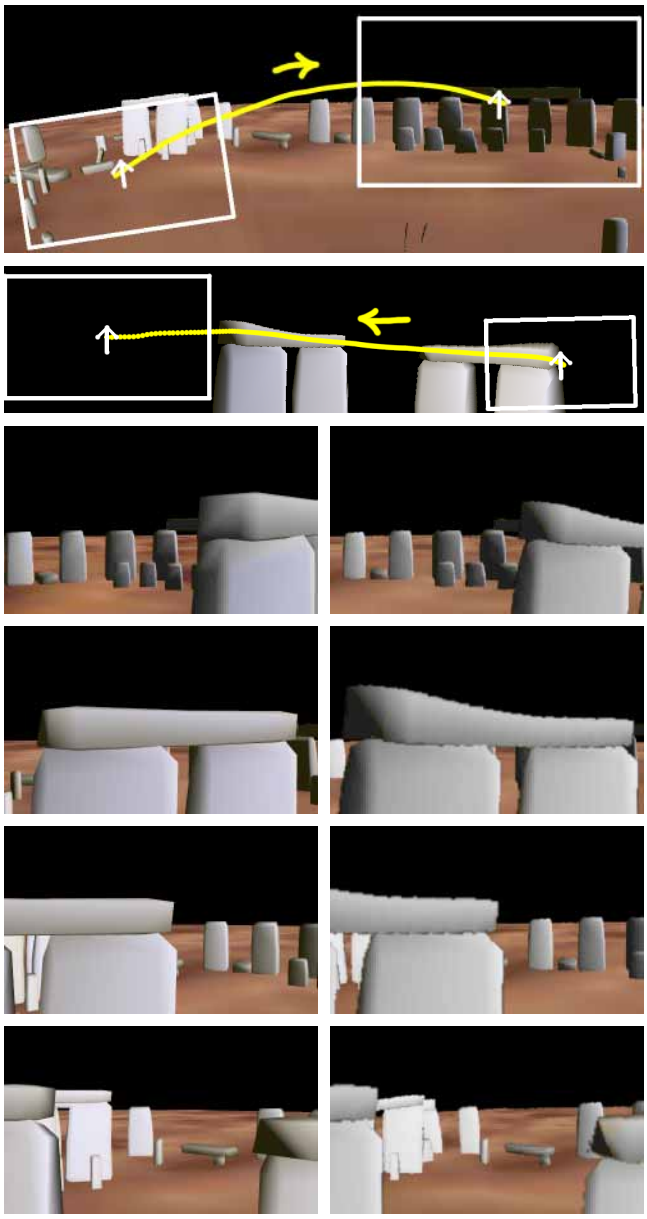

Figure 7 Frames from the Stonehenge movie (3D views on left, extracted frames on right).

The optical flow function can be approximated by discrete sampling. We take an evenly spaced grid of nine points (*source points*) on view $i+1$ and find corresponding points (*target points*) on view i. We reject any source points that see no objects in the scene; if fewer than four points remain, we subdivide the source grid until we have at least four points. The rejection of some source points leads to variations in our approximation of optical flow, which is why Figure 2 and Figure 3 are not exact cylindrical and conical projections.

To find the target point x' corresponding to source point x, we simply fire a ray from camera $i+1$ through point x into the 3D scene. As mentioned above, if the ray hits nothing it is rejected. If the ray intersects an object at $W(x)$, then we let x' be the projection of $W(x)$ into the image plane of camera i. This correspondence assumes no change in occlusion; that is, we assume that if camera $i+1$ can see point $W(x)$ then camera i can see it as well. Frame-to-frame coherence ensures that this assumption is reasonable.

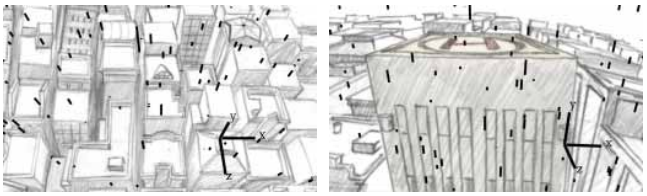

Figure 8 Two frames extracted from the Helicopter layout, showing overlaid 3D axes and computer-animated rain.

Finally, we find the least-squares-best similarity transform that matches the source points to their corresponding target points. Horn [9] describes a closed-form solution to this least-squares problem. We sometimes clamp the scale-change of the resulting transform in order to restrict the amount by which portions of the panorama are magnified.

4.2 Selection

Having placed all the views in a single coordinate system, we must choose, for each point, which of the many overlapping views at that point should be included in the final panorama. We project the center of each view into the panorama (using the transformations of the previous section). This collection of points, shown in yellow in our panoramas, we call the *spine* of the camera path. Then, at every point, we simply select the view corresponding to the closest spine point. The selected portions of each view constitute the Voronoi diagram of the spine.

Figure 6 shows a layout in which a helicopter flies across town, descends while spinning around, and then approaches a helicopter pad to land. (Note that the building with the helicopter pad, in yellow, appears twice in the panorama from two different perspectives.) The spine for the camera path and Voronoi diagram appear are also shown. Observe that the first and last camera position contribute large areas of the panorama. Along the spine, the separations between the Voronoi regions are roughly perpendicular to the motion of the camera center. Each camera position contributes a narrow wedge to the overall panorama. (For illustrative purposes, the figure shows the Voronoi diagram for a spine sampled very coarsely; the actual spine was sampled much more finely.)

4.3 Multiplaning

As mentioned in Section 2.4, multiplaning uses several layers of panoramas in concert to create an impression of parallax. In our process, the 3D scene is partitioned into separate planes by hand, although a useful area for future work would be automatic partitioning of the scene. Our application generates panoramas for each plane using the same camera path. The panorama for each plane contains an alpha channel, and the final animation is produced by simply compositing the extracted frames from back to front.

The Stonehenge example shown in Figure 7 illustrates an important use of multiplaning. The two windows move in opposite directions giving the impression of a camera circling around the ring of stones, looking inward. This effect is difficult to achieve with a single panorama. (We ignored the ground when firing rays during the placement stage, on the assumption that its lack of features would make a poor fit unimportant.)

4.4 Integrating computer-animated elements

Figure 8 shows the Helicopter movie, overlaid with 3D axes and a computer-animated rain element whose motion is appropriate for

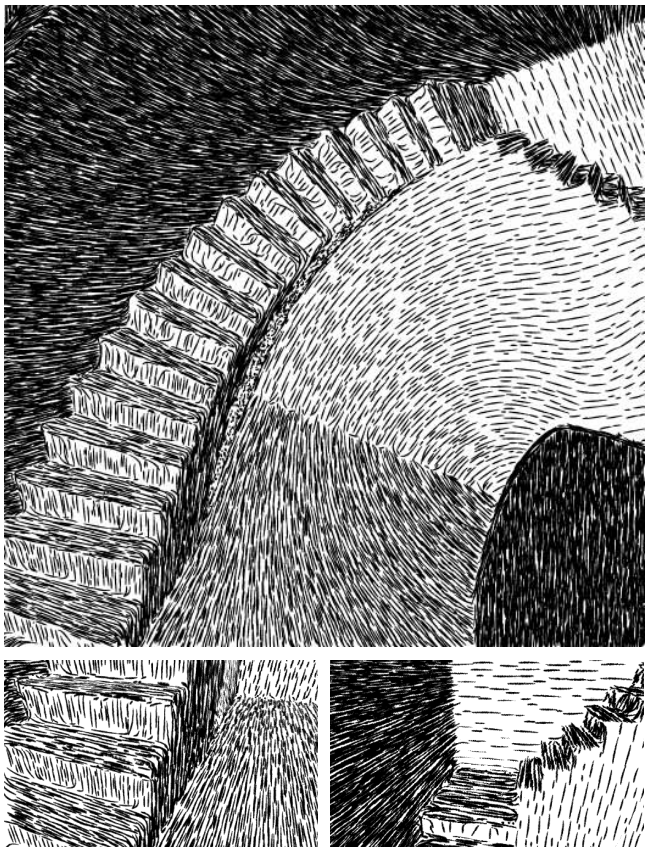

Figure 9 Pen-and-ink illustration of a panorama with two frames from from the resulting animation using warping.

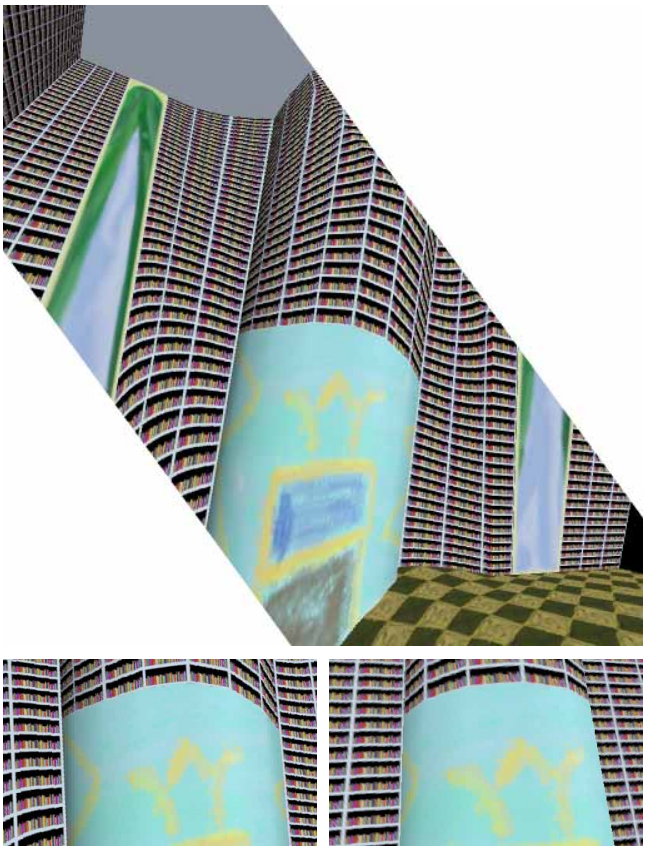

Figure 10 Library with curtains and fireplace. Excessive distortion in computer-generated panorama and unwarped frame (on left) is corrected in warped frame (on right).

the 3D model and camera. This alignment would have been difficult to achieve if the illustrated panorama had been designed by hand without a corresponding 3D model.

We have implemented two different techniques for integrating computer-animated foreground elements into our extracted frames. In both cases, the frames of the computer-animated elements are composited onto the layout movie. To generate the rain in the helicopter scene we simply make a standard 3D animation of rain using the 3D camera path. However, an element that interacts more closely with the background, like the bouncing ball (Figure 2d), requires a tighter match with the extracted frames. To generate the frame at time t, we take the world-space position w of the ball and find the corresponding location x on the panorama. To find x we look at the positions $P_s(w)$ on the panorama where w could have been placed at any time s. We solve $S(P_s(w)) = s$ for s, and let $x = P_s(w)$, which makes $w = W(x)$. We then render the ball using camera C_s. To create the final frame, we transform the rendered image of the ball with $E_t \circ P_s = E_t \circ E_s^{-1}$.

Incorporating a computer-animated element that interacts with the background and spans a significant number of views (*e.g.*, a flood of molasses) would require that the element share all of the multiple perspectives of the panorama.

5 Discussion and future work

Directly generating final panoramas. As our focus is on cel animation, the panoramas our program creates are drafts with rough transitions at the boundaries between views. Some applications, including integrating computer-animated elements for cel animation, require high-quality computer generated panoramas. Our staircase panorama, Figure 9, generated using a computer pen-and-ink illustration system [18] demonstrates one way to automatically create a final panorama, but a program that renders ready-to-use panoramas would be useful.

Panorama-creation problems. There are situations where successful panorama-creation is impossible. When occlusions on a single object change drastically without the object leaving the camera frame (e.g., if a camera makes a 360-degree pass around a dinner table, always showing the centerpiece and the surrounding place-settings), each extracted frame will show not only the part of the object that ought to be visible, but "adjacent" parts that ought to be hidden in the current view. Even for objects with smooth boundaries, this problem can occur if the visible silhouette of the object changes radically with different views. This problem is also closely related to another difficult situation: radical perspective changes. An example is a bug's-eye view of the floor as the bug flies upwards from the floor to a table. In each case, the difficulty is that the aggregate optical flow is not well-approximated by a similarity transform. This can be addressed, in part, by warping, which is discussed below.

There are also situations where *our* algorithm does not produce successful layouts, even though such layouts can be hand-generated. Figure 10 shows an example: because of the strong linear elements of the scene, it is essential that certain large-scale geometric features—the horizontal shelves and the vertical dividers—be preserved. Our algorithm, whose selection scheme is purely local, cannot handle this type of constraint. Once again,

warping can help address the problem, but an *ab initio* solution would be preferable.

Finally, there are two global issues not addressed by our method: first, the panorama can overlap itself as images from frames at widely separated times are placed (this almost happens in Figure 6); second, if the cumulative "scale" component of successive inter-frame transforms becomes too large or small, the artist may be compelled to render similar parts of the panorama at widely-differing scales, making a coherent artistic texture difficult. Allowing the extraction and placement maps to be adjusted by an arbitrary projective transformation might alleviate this; choosing the best correction would require global knowledge however.

Warping. If we store the world-space locations of points in the panorama, we can warp the panorama to produce a distortion-free animation. At time t, point x is warped to $(C_t \circ W)(x)$. This technique can be used with panoramas that would not produce a reasonable movie using the traditional technique, as demonstrated in the lower-right frame of Figure 10. (Warping is also used in the pen-and-ink example of Figure 9.) Unfortunately, using warping also has serious shortcomings. In particular, in some cases a warped view of the panorama may reveal world-space points that were not captured in the scene, leaving undesirable holes.

Fully exploring the potential of warping will surely expose a number of problems. Without the rectangular extraction constraint our general formulation will lead to a different algorithm for constructing the panorama. A good solution to the problems of occlusion and hole-filling for multiperspective panoramas should borrow from and extend related work in image-based rendering.

Acknowledgements

We would like to thank Scott Johnston for exposing us to the traditional use of panoramas in cel animation, and Ronen Barzel for suggesting the idea of generating such panoramas algorithmically. We also thank Tom Baker, Ed Ghertner, Dan Hansen, Kiran Joshi, Dan St. Pierre, Ann Tucker, and M.J. Turner from Disney for educating us further about cel animation and layouts. Particular thanks to Ed Ghertner for the illustrated panorama. Thanks to Brad deGraf and Protozoa for their VRML Stonehenge. Thanks to Michael Wong and Cassidy Curtis for helping to create the pen-and-ink staircase, and to Eric Stollnitz for creating the diagram and helping with the paper.

This work was supported by an Alfred P. Sloan Research Fellowship (BR-3495), an NSF Presidential Faculty Fellow award (CCR-9553199), an ONR Young Investigator award (N00014-95-1-0728) and Augmentation award (N00014-90-J-P00002), an NSF graduate fellowship, and an industrial gift from Microsoft.

References

[1] Deborah F. Berman, Jason T. Bartell, and David H. Salesin. Multiresolution painting and compositing. In *Proceedings of SIGGRAPH '94*, pages 85–90, New York, 1994. ACM.

[2] Nathaniel Bowditch. *Bowditch for Yachtsmen: Piloting; Selected from The American Practical Navigator*. David McKay Company, Inc., New York, 1976.

[3] Shenchang Eric Chen. Quicktime VR: An image-based approach to virtual environment navigation. In *Proceedings of SIGGRAPH '95*, pages 29–38, New York, 1995. ACM.

[4] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of SIGGRAPH '93*, pages 279–288, New York, 1993. ACM.

[5] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, Addison Wesley, August 1996.

[6] Maurits C. Escher et. al. *M.C. Escher : His Life and Complete Graphic Work*. Harry N. Abrams, New York, 1992.

[7] Jean-Daniel Fekete, Érick Bizouarn, Éric Cournarie, Thierry Galas, and Frédéric Taillefer. TicTacToon: A paperless system for professional 2-D animation. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 79–90. ACM SIGGRAPH, Addison Wesley, August 1995.

[8] Paul Haeberli. Grafica obscura web site. http://www.sgi.com/grafica/, 1997.

[9] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4), April 1987.

[10] John Lansdown and Simon Schofield. Expressive rendering: A review of nonphotorealistic techniques. *IEEE Computer Graphics and Applications*, 15(3):29–37, May 1995.

[11] Lenny Lipton. *Independent Filmmaking*. Straight Arrow Books, San Francisco, 1972.

[12] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of SIGGRAPH '95*, pages 39–46, New York, 1995. ACM.

[13] Barbara J. Meier. Painterly rendering for animation. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 477–484. ACM SIGGRAPH, Addison Wesley, August 1996.

[14] Walt Disney Productions. Pinocchio. Movie, 1940.

[15] Matthew Regan and Ronald Post. Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of SIGGRAPH '94*, Computer Graphics Proceedings, Annual Conference Series, pages 155–162. ACM SIGGRAPH, ACM Press, July 1994.

[16] Robyn and Rand Miller. Myst. Computer game, Cyan, Inc., 1993.

[17] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 197–206, August 1990.

[18] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH 97 Conference Proceedings*. ACM SIGGRAPH, Addison Wesley, August 1997.

[19] Steven M. Seitz and Charles R. Dyer. View morphing: Synthesizing 3D metamorphoses using image transforms. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 21–30. ACM SIGGRAPH, Addison Wesley, August 1996.

[20] Richard Szeliski. Video mosaics for virtual environments. In *IEEE Computer Graphics and Applications*, pages 22–30, March 1996.

[21] Bruce A. Wallace. Merging and transformation of raster images for cartoon animation. In *Computer Graphics (SIGGRAPH '81 Proceedings)*, volume 15, pages 253–262, August 1981.

[22] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of SIGGRAPH '94*, pages 91–100, July 1994.

[23] Georges Winkenbach and David H. Salesin. Rendering free-form surfaces in pen and ink. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 469–476. ACM SIGGRAPH, Addison Wesley, August 1996.

[24] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An interface for sketching 3D scenes. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 163–170. ACM SIGGRAPH, Addison Wesley, August 1996.

[25] Denis Zorin. *Correction of Geometric Perceptual Distortions in Pictures*. California Institute of Technology, Pasadena, CA, 1995.

[26] Denis Zorin and Alan H. Barr. Correction of geometric perceptual distortion in pictures. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 257–264. ACM SIGGRAPH, Addison Wesley, August 1995.