Computational Aesthetics in Graphics, Visualization and Imaging (2005)
L. Neumann, M. Sbert, B. Gooch, W. Purgathofer (Editors)

A Generative Model For Dynamic Canvas Motion

Matthew Kaplan and Elaine Cohen

Abstract

We present techniques for constructing realistic canvas and paper models and for enabling interactive dynamic canvas motion. Dynamic canvas motion means that there is a correspondence between the motion of canvas features and the motion of the models in the scene. Our artificial paper is created by simulating the physical process of creating paper with many individual fibers. To enable canvas motion, fibers are associated with each of the models in the scene. At runtime, the fibers associated with visible portions of the models and background fibers are used to construct a 2D canvas. Because fibers are "tied" to the models, the motion of canvas features corresponds to the motion of each model. This allows us to match the motion field of our dynamic 2D canvas to that of the the 3D scene exactly.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Non-Photorealistic Rendering, Canvas Synthesis, Dynamic Canvas Motion, Artistic Rendering

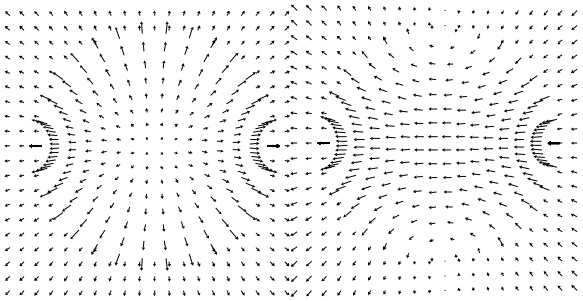
1. Introduction

Non-photorealistic rendering has expanded the bounds of traditional media by providing the ability to animate media that were traditionally static, such as pen and ink and pencil. This has introduced a number of artifacts peculiar to animated media, especially those pertaining to *temporal coherence*. Temporal coherence artifacts typically occur because marks that are meant to appear two dimensional and static(as on a typical flat drawing surface) are being matched to objects being animated in three dimensions. Difficulties often appear when attempting to ensure that the motion, shape and relevant media properties of marks appear consistent from frame to frame. As pointed out in [CTP*03], there are two solutions that are typically applied, neither one of which is ideal. First, marks are made in the two dimensional space of the image and warped to the 3D motion of the objects. This often results in the "shower door" effect where objects slide over the background. Second, marks may be applied to the surface of the 3D object and moved with the surface of the object. In this case the marks can appear plastered onto the surface of the object and often seem unrealistic in motion.

While much attention is paid to fixing the temporal coherence of foreground strokes, temporal coherence of the background canvas has been largely overlooked (except by [CTP*03]). When motion cues between the canvas and the foreground models is disjoint the sense of immersion in the environment is broken and the foreground and background seem to lie in separate planes as objects "slide" over the canvas. Motion in a 3D environment results in a complex optical flow that the flat quality of the 2D background cannot match. A *motion field* represents the time derivative of the 2D projection of a 3D point. This has been shown to be a critical motion cue for observers. Figure 1 shows examples of motion fields of a sphere in front of a plane under translation and rotation. This problem of attempting to match a 3D motion field with a 2D transformation is difficult because of the severe discontinuities at silhouette edges of objects and because objects under animation may have opposing directions. The non-uniform quality of the flows cannot be simply expressed as a 2D transformation. Because of this motion parallax, a fundamental tension exists between any mapping from a 2D surface to a 3D environment.

Two specific goals motivated this work when considering the canvas in such a context. First, we would like to be able to produce realistic paper and canvas models. Second, we would like to be able to achieve accurate dynamic motion that establishes a correspondence between the canvas model and the foreground objects.

Typically, such background canvases are created by either scanning existing paper or canvas models or by using Perlin noise functions [Per85, CAS*97] to create varying height fields. While these methods work well, it would be optimal for the user to be able to obtain arbitrary canvases with appropriate level of detail in grain, tooth, roughness, color, etc.. To this end we propose a simple model that simulates the



(a) (b)

Figure 1: *a)Motion field of a forward translation of a sphere in front of a plane. b)Motion field of the same scene under rotation.*

way paper and canvas are constructed and gives a realistic approximation of canvas and paper.

When animating scenes with both a canvas background and foreground objects, we would like the motion of the canvas to correspond with the all visible objects so as to maintain the immersive illusion of our NPR environment. Such a dynamic canvas should have the following properties: First, snapshots of the canvas model should appear to be a plausible canvas simulation across the entire visual field. Second, features of the canvas model should follow the objects to which they appear to correspond on the image. Third, the model should be variable in resolution since, as pointed out by [CTP*03], we may need to zoom infinitely. Fourth, the canvas texture in any position should be reproducible (i.e. if we rotate away from a view point and then rotate back, an identical canvas model should appear). Finally, the distortion of canvas elements should be kept to a minimum so as to avoid distracting the viewer. We propose an interactive solution that attempts to meet these five criteria by generating a new canvas image from any given viewpoint based on the position of paper fibers which are tied to the models in the scene.

Our paper offers the following contributions:

- Realistic Canvas Synthesis - We demonstrate a technique for producing realistic canvas and paper models based on simulations of canvas and paper construction using layers of fibers.
- Dynamic Canvas Motion - We show a method for mapping fibers to models in the scene. At runtime we construct a plausible canvas from all the fibers in all visible polygons. Because fibers are "tied" to the polygons, the movement of canvas features correspond to the movement of the objects in the scene. Finally, we give a method for enabling an infinite zoom.

2. Related Work

Most canvas models in computer graphics work have been created by one of two methods. Most commonly, a scanned image of a canvas is read in and a height field is created based on the intensity of each pixel [SB99]. Alternatively, Perlin noise is often used to build random height fields [Per85, CAS*97, CTP*03]. Both of these models can work well but both have problems. The first model is limited to images that the user has scanned. The second allows the user to obtain canvases of arbitary resolution and quality but it really only works well on the limited subset of paper and canvas models that Perlin noise represents well. Furthermore, it has not been shown how to embed structured elements in a canvas generated with Perlin noise. We propose a solution to the canvas generation problem by simulating the physical process of canvas construction to allow multiple canvas and paper models to be generated in a single framework as a function of several user-defined parameters. A good reference on the subject of papermaking can be found in [Toa83].

There has been significant work in the area of mapping the motion of strokes of various traditional media such as paint, pen and ink, pencil and charcoal. [Mei96, Dan99, SB99, KGC00, PHWF01]. The methods presented in this paper can be viewed as a projection into 2D from 3D elements that are attached to the surface of objects thus obtaining temporal coherence in a manner similar to [Mei96, KGC00]. However, our fibers are used in combination to form a cohesive structure rather than represent individual strokes.

We have found only one significant piece of work dealing with dynamic canvas motion [CTP*03]. They demonstrate a variety of techniques for mapping the 2D motion of a background canvas as closely as possible to the 3D motion of the animated scene. Unfortunately, these mappings are limited by the fundamental tension between 2D and 3D motion. In translation, the differential in motion between objects at different distances from the camera is significant and the mapping does not hold. In rotation, their spherical warp does a reasonable job when objects are at a distance but approximately 1/2 of each object in the foreground has an opposite motion field to that of the background. A static 2D model simply cannot accurately match the motion of a 3D environment. For this reason, we propose a model that moves canvas elements in 3D and reconstructs the canvas in 2D. This maps the motion field from 3D to 2D exactly for any camera motion. In addition, these authors give a model for an infinite zoom of the canvas background in order to approximate forward translations. Their method works well but suffers from significant blending artifacts between different levels of resolution. Our model for enabling infinite zoom actually has a different function and is entirely different in methodology.

3. Realistic Canvas Synthesis

3.1. Generating Canvas Models

When describing background models, the words canvas and paper are generally used interchangeably in NPR. While their purpose and appearance are often similar, the physical processes used to create them are very different. Paper is a surface created by pressing paper fibers in a binding medium. The fibers are generally made from cellulose which is found in plant fibers. The plant fibers are ground into wood pulp and dumped in acid to generate the cellulose. We ignore issues of what kind of binding medium and/or sizing is used. Those elements usually affect absorption rates which this model does not consider. The orientation of the resultant cellulose fibers is usually random and it is the size and quality of the fibers that gives each paper its own tooth, grain and feel. Canvas is a surface created of vertical and horizontal interwoven threads. While some weaving processes are very consistent, most interesting woven canvas surfaces contain many variations in the prominence and color of individual elements. The prominence of each thread is local because of the interaction with other threads. Moreover, many of the patterns are too small to be viewed as a consistent thread running the entire course of the canvas. This leads to the observation that many interesting canvas models can be simulated with small, local threads.

We define a *fiber* as the base element of our canvas model. A paper fiber is typically a randomly oriented small, rectangular or oblong piece of wood pulp. Paper is built up through the composition of many fibers. In our system, a fiber is modeled as a small, randomly positioned and oriented rectangular shape. Since we typically use fiber lengths between 20-40 pixels and widths between 1 and 3 pixels this shape can also be viewed as a line of variable thickness. To build a paper model, each rectangular shape is drawn into a height field which represents the canvas. The height is incremented for every location within the fiber shape. A color texture for the canvas is computed after the height field is created with the minimum height being mapped to black (or some muted dark color) and maximum height being mapped to white. Mutiplying the color texture by a user defined scale in rgb allows us to produce differently colored canvases. A basic algorithm to create a canvas is shown in Algorithm 1.

Algorithm 1: *CreateCanvas()*

```
for count = 0 to count < MaxFibers do
   GenerateRandomFiber()
end for
for i = 0 to i < CanvasWidth do
   for j = 0 to j < CanvasHeight do
      color[i][j] = (height[i][j] - HeightMin)/(HeightMax -
      HeightMin)
      color[i][j] *= κ
   end for
end for
```

Algorithm 2: *GenerateRandomFiber()*

```
fiber.position.x = unitRand() * CanvasWidth
fiber.position.y = unitRand() * CanvasHeight
fiber.direction.x = unitRand()
fiber.direction.y = unitRand()
fiber.direction.makeUnitVector()
fiber.direction *= FiberLength
fiber.width = FiberWidth
DrawFiberIntoHeightField(fiber)
```

Here, the *height* is the height field of the canvas and *color* is the color map for the canvas. κ is the user defined base color of the canvas. *MaxFibers* is the total number of fibers in our canvas simulation, which we typically set between 200,000 and 400,000 for a 512x512 canvas. $GenerateRandomFiber()$ is a function which produces a random position and direction for the fiber, as shown in Algorithm 2. Here, $unitRand()$ is a function that returns a random number between 0 and 1 and $makeUnitVector()$ is a function that normalizes a vector. *FiberWidth* and *FiberLength* are values set by the user. Altering these values allows the user to vary the tooth and grain of the paper, with wide fibers being useful for producing simulations of heavy papers such as those used in watercolor and charcoal and narrow fibers being useful for more delicate, even, fine grained papers. After the fiber is generated, the height field is incremented in the area occupied by the new fiber with the $DrawFiberIntoHeightField()$ function. Results of this algorithm are shown in Figure 2.

We can simulate canvas with this model as well, using the observation made above that local elements are sufficient to emulate long threads due to the artifacts inherent in weaving and the small quality of the characteristic canvas texture patterns. The only change in Algorithm 2 is that we orient the direction of the fibers manually. Because canvas models are typically woven in only horizontal and vertical directions, this simplifies our direction choice to the four cardinal directions which we can choose from randomly at runtime or allow the user to choose various options for weighting vertical versus horizontal threads. As shown in Figure 2, these produce fairly realistic results for a variety of canvas models.

The canvas is displayed as a texture mapped quadrilateral. Canvas and paper models produced in this fashion are suitable for most media simulation techniques or simply for background textures.

3.2. Canvas impurities and additions

Often, artifacts will occur on canvas or paper models due either to impurities in the paper production or to the inclusion of additional materials the paper maker may use dur-

(a) Paper - Fine Grain (b) Paper - Rough Grain

(c) Canvas - Horizontal - Fine (d) Canvas - Horizontal - Rough

(e) Canvas - Vertical - Fine (f) Canvas - Vertical - Rough

(g) Canvas - Interwoven - Fine (h) Canvas - Interwoven - Rough

Figure 2: *Canvas examples: (color scale = 1,1,.7) Various canvas models produced with our system. Line widths for fine grains are 1 and for rough grains are 3.*

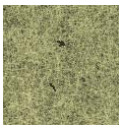
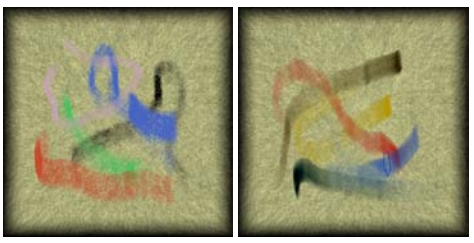


Figure 3: *An example of impurities added into our paper model*



(a) Pencil (b) Paint

Figure 4: *Examples of media simulations using canvas models generated with our system. a) shows pencil on paper and b) shows paint on paper. Note the patterns produced by the grain of the canvas, especially on the pencil simulation.*

ing paper creation such as glitter, leaves, nutmeg, cinammon, grass, flower petals, metal flakes, essential oils, food coloring and coffee grounds. The motion of these artifacts may provide strong visual cues since they are distinct in the visual field. Often artists will add materials to create such artifacts in order to personalize their materials. To simulate this, we allow the user to specify artifacts to use with each paper and the artifact density. We store a canonical set of artifacts as tiny images. At each pixel in these images is stored a height and color value. We generate a random set of artifact positions and draw them during the paper creation process into the height and color field. See Figure 3 for an example.

4. Dynamic Canvas Motion

4.1. Interactive Canvas Construction

There are two problems associated with the canvases presented in section 3.1. First, the model is mapped to a flat surface and second, it is not generated in real time. Because of these reasons, it isn't suitable for solving the dynamic canvas motion problem. Ideally, we would like a canvas model that isn't flat, but that maps to a flat surface uniformly. Also, we would like to be able to perform walkthroughs at interactive rates. While interactivity isn't vital in presenting a general solution to the dynamic canvas problem it is useful in producing a tool that is useable for interactive and real-time renderings. To achieve this, we associate individual fibers with each object in the scene (including a background model). Each object in our system is a piecewise linear triangle mesh. At runtime, a canvas is constructed from fibers placed on the surface of every visible object.

The major cost of constructing the canvas model is computing positions and direction vectors for each fiber and calculating the height field. We can compute an approximate version of this construction by pre-computing fiber values and then rendering each fiber as an alpha-blended line in OpenGL. By precomputing the average contribution of each fiber in the original canvas, we derive a reasonable alpha value to draw the fibers with. The color of each fiber is just the κ color value defined in Section 3.1.

In the original canvas model, fibers were placed randomly within the canvas. When the canvas is in motion, however, static placement of fibers in a background field may be problematic. For our interactive model, the system places fibers randomly on the surface of each triangular face of each model in the system. Rather than store an actual position in world coordinates, we represent each fiber position using barycentric coordinates. The barycentric coordinates of any fiber allow its position to be reconstructed using the coordinates of the face it lies on. Therefore, as each face is transformed in world space, its fibers will occupy a consistent position within the spatial domain of its 2D projection.

Our initial distributions of random barycentric coefficients were quite bad leading to obvious clumps of fibers near the center of triangles. It is clear that achieving a uniform distribution of random barycentric coefficients is very important. See [Tur90] for a method of uniformly generating barycentric coefficents. At runtime we compute fiber positions using the precomputed barycentric coordinates and the current position of the face. The other endpoint that defines the fiber is computed by representing the fiber is given by $newposition + (direction * FiberLength)$ where $(direction * FiberLength)$ is the (precomputed) direction vector multiplied by the user defined fiber length. Barycentric coordinates also allow us to handle model deformation, which is useful for animation. Even having obtained an even distribution of fibers in object space, we cannot guarantee that our distribution will be even in screen space due to perspective projection. As triangles approach grazing angles they may distort and overly weight the end closest to the viewpoint. In practice however, this effect was rarely seen and seemed to have little impact, possibly due to the quantity of triangles used and the relatively small amount of screen space each one occupied on average.

We scale the number of fibers that each triangle draws based on its area, measured in screen space. We obtain this measurement by projecting each vertex onto the screen plane and solving for the area of each visible triangle. The number of fibers to draw for each triangle to match the target canvas tone is $FibersToDraw = FibersPerPixel * TriangleArea$ where $FibersPerPixel = MaxFibers/(CanvasWidth * CanvasHeight)$. Here, $FibersPerPixel$ is the average number of fibers drawn per unit area of screen space (defined to be 1 pixel in our system) and $FibersToDraw$ is the number of fibers to draw for the triangle. This lets us match the tone of the canvas model we want to reproduce.

Our system generates a list of N sets of random barycentric coefficients, where N is usually between 5,000-10,000. We generate a random index into the list for each triangle which is used as the first fiber to draw. This ensures that each triangle draws a different set of fibers in order to minimize repeated patterns in the canvas. Otherwise, each triangle in the model draws similar sets of fibers, leading to visibly repeated patterns over each face which can be visually disorienting.

Though we have calculated the correct number of fibers to draw for each triangle, we will end up with tones that are too light where one face occludes another if both draw all of their fibers. Because of this problem, we only display fibers that sit on a visible portion of a triangle. An ID image (also known as an item buffer) is rendered that determines which triangular face is visible in each pixel of the screen. To control the tone of the canvas, we compare the starting position of each fiber to be drawn with the ID image. Fibers are only drawn in pixels that show their parent face. The effect of this is that only visible geometry will produce fibers to be drawn. Since fibers are drawn in 2D, each fiber takes up a constant amount of image space.

Handling fibers at the silhouettes of objects is problematic and we present two methods for doing so, each with advantages and disadvantages. In the first method, the fibers are allowed to overlap the edges of object boundaries. This has the effect of blending adjacent shapes and maintaining canvas continuity with the tradeoff that a discontinuity may appear at the boundary during animation due to the contrary motion of overlapping fibers. These discontinuities are lessened with smaller fiber lengths. The second method uses the stencil buffer to restrict drawing of fibers to the object that owns them. This method sacrifices canvas continuity at silhouette edges in order to get rid of the contrary motion discontinuities apprarent in the first method. Also, by drawing our ID image directly into the stencil buffer, we get a significant speedup since we don't have to read the ID image buffer during every render.

At runtime, we do the following: 1) Update all camera and model transformations, 2) Create an ID image, 3) Render our final image beginning with the dynamic canvas background, 4) Render the models. In our animations, we have rendered the models and silhouettes with an alpha value of .4 to simulate the semi-transparent nature of actual media (such as colored pencil or pastel). The light is positioned at a constant distance from the viewpoint. The accompanying animations show several paper and canvas models under transformations, using both stencil buffer and ID buffer methods for drawing silhouette fibers.

The result of these techniques is a canvas model whose constituent elements motion matches exactly the motion of the various surfaces in the scene. The significant features

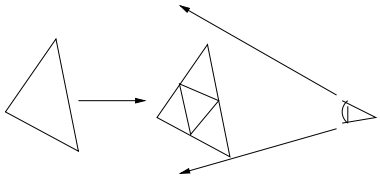


Figure 5: *Triangles that take up too much of the visual field are subdivided and new fibers are added using the barycentric coordinates based on the vertices of the child triangles.*

"stick" to the models under any type of canvas motion. Examples of this are shown in Figure 6. Because these results are designed to work in an animated environment, we refer the reader to the accompanying videos for a more accurate demonstration of the results of this paper. Note that the canvas seen when any of the animations are paused at any point is completely plausible.

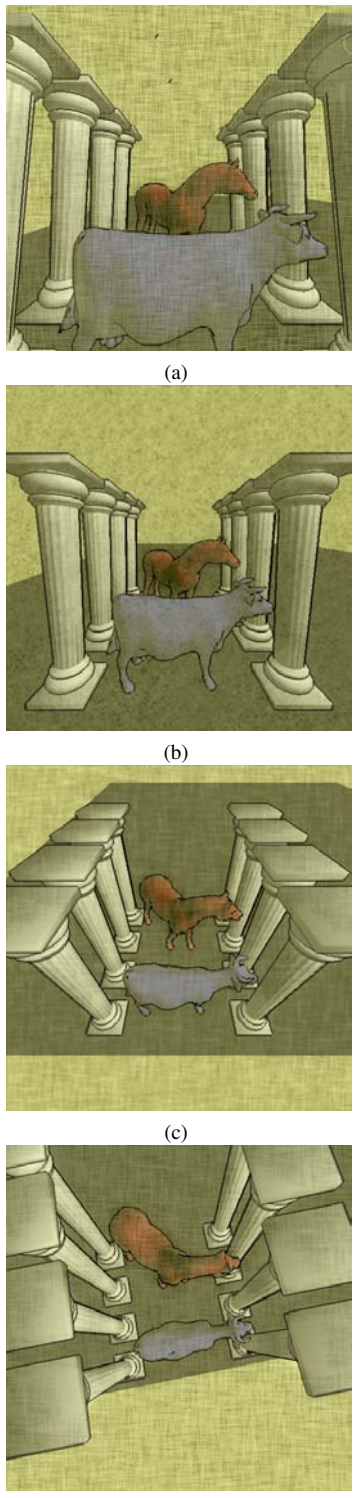
4.2. Infinite Zoom

To give the user freedom to perform arbitrary camera transformations, the ability to handle an infinite zoom must be provided. Because all of the fiber values are precomputed, after the maximum number of fibers, N, have been displayed on any face, there are no new fibers to display. If that triangle takes up too much of the visual field there will start to appear large gaps in the canvas where there are not enough fibers. Note that while [CTP*03] perform zooms to approximate translation, our model approximates translation automatically since it exists in 3D. Infinite zooms enable accurate level of detail at any scale.

To handle the infinite zoom, any triangle that requires more than N fibers is subdivided into four sub-triangles, as seen in Figure 5. Note that the positions of the triangle vertices are not changed as in many subdivision schemes. New vertices are added at the midpoint of existing edges. Then, new fibers are added based on the barycentric coordinates of the sub-triangles. This allows the continuous generation of new fibers in any area of the screen where more detail is needed. The fibers generated with this technique are consistent and reproducible. Furthermore, this method is fast since generally only a very small number of triangles needs to be subdivided.

4.3. A Background Model

The canvas model presented in Section 3.1 mapped the texture to a quadrilateral that filled the visual field completely. [CTP*03] showed that rotations of a background field are matched by warping canvas motion to a cylinder or sphere; the canvas is esentially mapped onto the sphere of directions around the viewpoint. As the viewpoint rotates, the spherical mapping matches the motion fields one would expect to



(a) (b) (c) (d)

Figure 6: *Screen shots from several of our dynamic canvas animations. Each image has different user-defined optionds applied to the paper/canvas model. While the animations need to be viewed for the full effect, notice that the canvas model is completely plausible for each snapshot.*

see of a background field. For this reason, we load a spherical triangular mesh and map the canvas directly onto it. In our model, the sphere always sits at a constant distance from the viewpoint (i.e. the center of the sphere is always translated to the viewpoint) emulating a background at an infinite distance from the viewer. The canvas sphere is rotated with the viewpoint, giving the impression of correct background motion fields under rotation. An error occurs when translating left/right/up/down since the background stays fixed. This may be acceptable since we're approximating a background at infinity which won't move much relative to us in any case. This can be fixed by using a very large sphere that does move under translations, with the tradeoff that the camera may move outside the background model due to some transformations.

5. Results

In this paper, we have shown a variety of techniques for constructing and animating realistic canvas models.

In Section 3 we presented a model for realistic canvas/paper synthesis. Our canvas model gives visually realistic results for a variety of paper and canvas types such as linen, watercolor paper, charcoal paper, drawing paper, canvas, various forms of cotton duck and has the potential to accurately simulate many other types of canvas/paper as well. Though we did not model any surfaces with large repeatable structures (such as the lines of a sheet of writing paper), there is no reason that these could not be included in this model. Expanding this static model to include larger structures and more types of complex fiber patterns would probably be the most relevant area for future work here. Because our dynamic canvas model relies on many local elements, very large structures (such as the graph lines on graph paper) may be difficult to achieve using this model.

The 2D generated canvases take about 1-3 seconds to generate on a 2 Ghz mobile Pentium 4 using unoptimized code. We typically generated between 200,000-500,000 fibers with fiber lengths set between 15-45 pixels. Our model is physically motivated rather than a full physical simulation of paper and canvas creation. Therefore, we feel there is future work to be done in creating complete phyical simulations of canvas and paper creation proceses.

The dynamic canvas model presented in Section 4 and using the ID buffer runs at about 3-5 frames a second on a mobile Pentium 4 2 Ghz using a ATI Radeon IGP 345M with 64 megabytes of shared graphics memory and unoptimized code. The stencil buffer version runs roughly 2-3 times as fast since there are no screen buffer reads involved in the rendering. We believe there are significant speedup gains to be made by combining our techniques with graphics hardware acceleration, though we have made no tests to support this claim.

6. Discussion

Because the results in NPR are so often subjective, we evaluate the success of our techniques in relation to the criteria laid out in Section 1. First, any snapshot position of the canvas appears to be a plausible canvas simulation. The distribution of fibers and the tone across the visual field are consistent in both foreground models and background. While the dynamic canvases did tend to be visually convincing, their quality was typically not quite as good as the static 2D version. Second, motion fields of the canvas and the objects in the scene are matched exactly. The features of the canvas appear to "stick" to the surface of the models. This provides important visual cues to the viewer and does not suffer from the shower door effect previously described. Third, we have shown how to view such canvases at any resolution by describing a fast robust technique for an infinite zoom. Fourth, every canvas is reproducible. Because the constituent elements of our canvas model move with scene objects, then a scene viewed from the same position twice will be reproduced no matter what camera motions have previously occured. This error happened in previous research when the canvas was stored separately from the scene - because movement of the background did not always match movement of the foreground or camera, background canvases would be in different positions for identical scenes.

The final metric, the distortion of canvas elements, is the hardest to judge. Distortion occurs when the motion of the background canvas does not match the motion of the foreground objects. Therefore we can measure the distortion as the difference between motion fields of the canvas and the scene between any two frames. Our motion fields match match nearly identically, which is a significant improvement over previous research, but there are problems where fibers overlap silhouette boundaries. Because of the the contrary motion of fibers lying on either side of the silhouette the motion fields of the opposing fibers may appear to cancel each other out thus leaving a *halo* of distortion around objects. This effect increases with longer fiber lengths. Some users reported that this effect wasn't noticeable whereas some stated that they couldn't concentrate on anything else during the animations. This effect is obvious when a single model is on the screen but seems to be diminished greatly when viewed in the context of several objects. Using the stencil buffer solution at sihouettes solved this problem. Since fibers no longer overlap at silhouettes, there was no disturbance in the visual field at object boundaries. The tradeoff with this technique is that the continuity of the canvas at the silhouettes is broken. This may be acceptable especially if silhouettes are drawn as solid lines (as is common in pen and ink and pencil rendering styles).

Having created a novel dynamic canvas it is relevant to ask: In what situations should it be used? Is it superior to previous techniques? While the animations show that the canvas features move with the models, it is precisely this emphasis

on the 3D quality of the models that may not be desireable in some situations. While these motion cues may be important for complex scenes, many NPR techniques that incorporate background canvases are essentially 2 1/2D or make use of simplistic 'toon models. In such cases, emphasizing the 3D nature of the system may be undesireable and distracting. In those instances, the work of [CTP*03] may be more appropriate since the canvas itself never changes. Our dynamic canvas seems to be more appropriate for complex NPR walkthroughs that are fully 3D in nature. The depth cues that it provides are important in such environments and do not break the immersive sensation as much as the shower door effect of previous methods. This implies that there is a tradeoff between the shower door effect and the 3D emphasis of our method, either of which may be appropriate depending on the application.

Overall, the results of this paper seem to satisfy our initial goals and criteria, allowing us to generate realistic canvases simulating the way canvas/paper is actually constructed and to generate such canvases dynamically in order to match the motion fields of scenes exactly. We believe this will be a useful tool for media simulations and in developing NPR animations and walkthroughs that do not break the immersive sensation due to opposing motion fields of models and background elements.

References

[CAS*97] CURTIS C. J., ANDERSON S. E., SEIMS J. E., FLEISCHER K. W., SALESIN D. H.: Computer-generated watercolor. *Computer Graphics 31*, Annual Conference Series (Aug. 1997), 421–430.

[CRE01] COHEN E., RIESENFELD R., ELBER G.: *Geometric Modeling With Splines*. A K Peters, LTD., 2001.

[CTP*03] CUNZI M., THOLLOT J., PARIS S., DEBUNNE G., GASCUEL J.-D., DURAND F.: Dynamic canvas for immersive non-photorealistic walkthroughs. In *Proc. Graphics Interface* (june 2003), A K Peters, LTD.

[DAJ*97] D.N.WOOD, A.FINKELSTEIN, J.F.HUGHES, C.E.THAYER, D.H.SALESIN: Multiperspective panoramas for cel animation.

[Dan99] DANIELS E.: Deep canvas in disney's tarzan. In *ACM SIGGRAPH 99 Conference abstracts and applications* (1999), ACM Press, p. 200.

[GG01] GOOCH B., GOOCH A.: *Non-Photorealistic Rendering*. AK Peters, Ltd., 2001.

[GS94] G.WINKENBACH, SALESIN D.: Computer-generated pen-and-ink illustration. In *SIGGRAPH* (1994).

[HP00] HERTZMANN A., PERLIN K.: Hertzmann, a., and perlin, k. painterly rendering for video and interaction. proceedings of npar 2000, 7–12. In *Painterly rendering for video and interaction* (2000).

[KGC00] KAPLAN M., GOOCH B., COHEN E.: Interactive artistic rendering. In *Proceedings of NPAR 2000* (2000), pp. 67–74.

[Mei96] MEIER B. J.: Painterly rendering for animation. In *SIGGRAPH* (1996), vol. 30, pp. 477–484.

[MSAB94] M.SALISBURY, SALESIN D., ANDERSON S., BARZEL R.: Interactive pen-and-ink illustration. In *SIGGRAPH* (1994).

[Per85] PERLIN K.: An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (1985), ACM Press, pp. 287–296.

[PHWF01] PRAUN E., HOPPE H., WEBB M., FINKELSTEIN A.: Real-time hatching. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), Fiume E., (Ed.), pp. 579–584.

[SB99] SOUSA M. C., BUCHANAN J. W.: Computer-generated graphite pencil rendering of 3D polygonal models. In *Computer Graphics Forum (Eurographics '99)* (1999), Brunet P., Scopigno R., (Eds.), vol. 18(3), The Eurographics Association and Blackwell Publishers, pp. 195–208.

[Toa83] TOALE B.: *The Art of Papermaking*. Sterling Publishing, 1983.

[Tur90] TURK G.: Generating random points in triangles. In *Graphics Gems I*, Glassner A., (Ed.). Academic Press, 1990.