Non-Photorealistic Virtual Environments[*]

Allison W. Klein Wilmot Li Michael M. Kazhdan Wagner T. Corrêa
Adam Finkelstein Thomas A. Funkhouser

Princeton University

Abstract

We describe a system for non-photorealistic rendering (NPR) of virtual environments. In real time, it synthesizes imagery of architectural interiors using stroke-based textures. We address the four main challenges of such a system – interactivity, visual detail, controlled stroke size, and frame-to-frame coherence – through image based rendering (IBR) methods. In a preprocessing stage, we capture photos of a real or synthetic environment, map the photos to a coarse model of the environment, and run a series of NPR filters to generate textures. At runtime, the system re-renders the NPR textures over the geometry of the coarse model, and it adds dark lines that emphasize creases and silhouettes. We provide a method for constructing non-photorealistic textures from photographs that largely avoids seams in the resulting imagery. We also offer a new construction, *art-maps*, to control stroke size across the images. Finally, we show a working system that provides an immersive experience rendered in a variety of NPR styles.

Keywords: Non-photorealistic rendering, image-based rendering, texture mapping, interactive virtual environments.

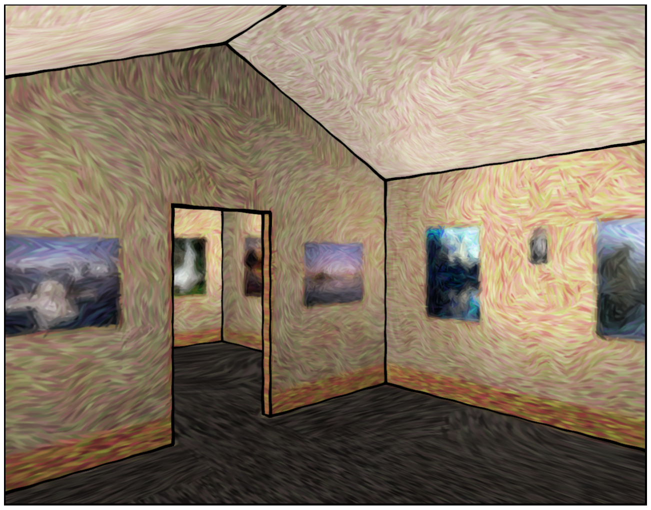


Figure 1: A non-photorealistic virtual environment.

1 Introduction

Virtual environments allow us to explore an ancient historical site, visit a new home with a real estate agent, or fly through the twisting corridors of a space station in pursuit of alien prey. They simulate the visual experience of immersion in a 3D environment by rendering images of a computer model as seen from an observer viewpoint moving under interactive control by the user. If the rendered images are visually compelling, and they are refreshed quickly enough, the user feels a sense of presence in a virtual world, enabling applications in education, computer-aided design, electronic commerce, and entertainment.

While research in virtual environments has traditionally striven for photorealism, for many applications there are advantages to non-photorealistic rendering (NPR). Artistic expression can often convey a specific mood (e.g. cheerful or dreary) difficult to imbue in a synthetic, photorealistic scene. Furthermore, through abstraction and careful elision of detail, NPR imagery can focus the viewer's attention on important information while downplaying extraneous or unimportant features. An NPR scene can also suggest additional semantic information, such as a quality of "unfinishedness" that may be desirable when, for example, an architect shows a client a partially-completed design. Finally, an NPR look is often more engaging than the prototypical stark, pristine computer graphics rendering.

The goal of our work is to develop a system for real-time NPR virtual environments (Figure 1). The challenges for such a system are four-fold: interactivity, visual detail, controlled stroke size, and frame-to-frame coherence. First, virtual environments demand interactive frame rates, whereas NPR methods typically require seconds or minutes to generate a single frame. Second, visual details and complex lighting effects (e.g. indirect illumination and shadows) provide helpful cues for comprehension of virtual environments, and yet construction of detailed geometric models and simulation of global illumination present challenges for a large virtual environment. Third, NPR strokes must be rendered within an appropriate range of sizes; strokes that are too small are invisible, while strokes that are too large appear unnatural. Finally, frame-to-frame coherence among strokes is crucial for an interactive NPR system to avoid a noisy, flickery effect in the imagery.

We address these challenges with image-based rendering (IBR). In general, IBR yields visually complex scenery and efficient rendering rates by employing photographs or pre-rendered images of the scene to provide visual detail. Not surprisingly, by using a hybrid NPR/IBR approach we are able to reap the benefits of both technologies: an aesthetic rendering of the scene, and visual complexity from a simple model. More subtly, each technology addresses the major drawbacks of the other. IBR allows us to render artistic imagery with complex lighting effects and geometric detail at interactive frame rates while maintaining frame-to-frame coherence. On the flipside, non-photorealistic rendering appeases many of the artifacts due to under-sampling in IBR, both by visually masking them and by reducing the viewer's expectation of realism.

[*] http://www.cs.princeton.edu/gfx/proj/NPRVE

At a high level, our system proceeds in three steps as shown in Figure 2. First, during off-line preprocessing, we construct an IBR model of a scene from a set of photographs or rendered images. Second, during another preprocessing step, we filter samples of the IBR model to give them a non-photorealistic look. The result is a non-photorealistic image-based representation (NPIBR) for use in interactive walkthroughs. Finally, during subsequent on-line sessions, the NPIBR model is resampled for novel viewpoints to reconstruct NPR images for display.

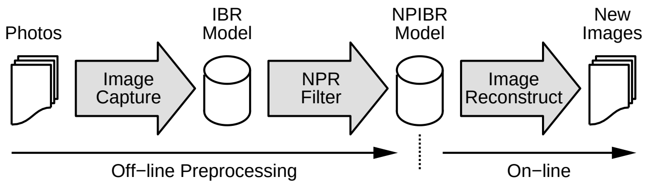


Figure 2: Overview of our approach.

This approach addresses many of the challenges in rendering NPR images of virtual environments in real-time. First, by executing the most expensive computations during off-line preprocessing, our system achieves interactive frame rates at run-time. Second, by capturing complex lighting effects and geometric detail in photographic images, our system produces images with visual richness not attainable by previous NPR rendering systems. Third, with appropriate representation, prefiltering, and resampling methods, IBR allows us to control NPR stroke size in the projected imagery. Fourth, by utilizing the same NPR imagery for many similar camera viewpoints rather than creating new sets of strokes for each view, our system acquires frame-to-frame coherence. Moreover, by abstracting NPR processing into a filtering operation on an image-based representation, our architecture supports a number of NPR styles within a common framework. This feature gives us aesthetic flexibility, as the same IBR model can be used to produce interactive walkthroughs in different NPR styles.

In this paper, we investigate issues in implementing this hybrid NPR/IBR approach for interactive NPR walkthroughs. The specific technical contributions of our work are: (1) a method for constructing non-photorealistic textures from photographs that largely avoids seams in images rendered from arbitrary viewpoints, and (2) a multiresolution representation for non-photorealistic textures (called *art-maps*) that works with conventional mip-mapping hardware to render images with controlled stroke size. These methods are incorporated into a working prototype system that supports interactive walkthroughs of visually complex virtual environments rendered in many stroke-based NPR styles.

The remainder of this paper is organized as follows. In Section 2 we review background information and related work. Sections 3-5 address the main issues in constructing, filtering, and resampling a hybrid NPR/IBR representation. Section 6 presents results of experiments with our working prototype system, while Section 7 contains a brief conclusion and discussion of areas for future work.

2 Related Work

The traditional strategy for immersive virtual environments is to render detailed sets of 3D polygons with appropriate lighting effects as the camera moves through the model [21]. With this approach, the primary challenge is constructing a digital representation for a complex, visually rich, real-world environment. Despite recent advances in interactive modeling tools, laser-based range-finders, computer vision techniques, and global illumination algorithms, it remains extremely difficult to construct compelling models with detailed 3D geometry, accurate material reflectance properties, and realistic global illumination effects. Even with tools to create an attractive, credible geometric model, it must still be rendered at interactive frame rates, limiting the number of polygons and shading algorithms that can be used. With such constraints, the resulting imagery usually looks very plastic and polygonal, despite setting user expectations for photorealism.

In contrast, image-based modeling and rendering methods represent a virtual environment by its radiance distribution without relying upon a model of geometry, lighting, and reflectance properties [5]. An IBR system usually takes images (photographs) of a static scene as input and constructs a sample-based representation of the plenoptic function, which can be resampled to render photorealistic images for novel viewpoints. The important advantages of this approach are that photorealistic images can be generated without constructing a detailed 3D model or simulating global illumination, and the rendering time for novel images is independent of a scene's geometric complexity. The primary difficulty is storing and resampling a high-resolution representation of the plenoptic function for a complex virtual environment [23]. If the radiance distribution is under-sampled, images generated during a walkthrough contain noticeable aliasing or blurring artifacts, which are disturbing when the user expects photorealism.

In recent years, a few researchers have turned their attention away from photorealism and towards developing non-photorealistic rendering techniques in a variety of styles and simulated media, such as impressionist painting [13, 15, 20, 24], pen and ink [28, 33], technical illustration [11, 27], ornamentation [34], engraving [25, 26], watercolor [4], and the style of Dr. Seuss [18]. Much of this work has focused on creating still images either from photographs, from computer-rendered reference images, or directly from 3D models, with varying degrees of user-direction. One of our goals is to make our system work in conjunction with any of these technologies (particularly those that are more automated) to yield virtual environments in many different styles.

Several stroke-based NPR systems have explored time-changing imagery, confronting the challenge of frame-to-frame coherence with varying success. Winkenbach *et al.* [32] and later Curtis *et al.* [4] observed that applying NPR techniques designed for still images to time-changing sequences yields flickery, jittery, noisy animations because strokes appear and disappear too quickly. Meier [24] adapted Haeberli's "paint by numbers" scheme [13] in such a way that paint strokes track features in a 3D model to provide frame-to-frame coherence in painterly animation. Litwinowicz [20] achieved a similar effect on video sequences using optical flow methods to affix paint strokes to objects in the scene. Markosian [22] found that silhouettes on rotating 3D objects change slowly enough to give frame-to-frame coherence for strokes drawn on the silhouette edges. We exploit this property when drawing lines on creases and silhouettes at run-time. Kowalski *et al.* [18] extends these methods by attaching non-photorealistic "graftals" to the 3D geometry of a scene, while seeking to enforce coherence among the graftals between frames. The bulk of the coherence in our system comes from reprojection of non-photorealistic imagery, so the strokes drawn for neighboring frames are generally slowly-changing.

Several other researchers, for example Horry *et al.* [17], Wood *et al.* [35], and Buck *et al.* [1], have built hybrid NPR/IBR systems where hand-drawn art is re-rendered for different views. In this spirit our system could also incorporate hand-drawn art, although the drawing task might be arduous as a single scene involves many reference images.

In this paper, we present a system for real-time, NPR virtual environments. Rather than attempting to answer the question "how would van Gogh or Chagall paint a movie?" we propose solutions to some technical issues facing an artist wishing to use NPR styles in a virtual environment system. Two visual metaphors represent the extremes in a spectrum of aesthetics one could choose for an "artistic" immersive experience. On one extreme, we could imagine that an artist painted over the walls of the model. In this case, the visual effect is that as the user navigates the environment the detailed stroke work is more or less apparent depending on her distance from the various surfaces she can see. In the other extreme, we could imagine that as the user navigates the environment in real-time, a photograph of what is seen is captured, and an artist instantaneously paints a picture based on the photograph. In this case, the visual effect suffers from either flickering strokes (lack of frame-to-frame coherence) or the "shower door effect" (the illusion that the paintings are somehow embedded in a sheet of glass in front of the viewer). Our goal is to find a compromise between these two visual metaphors: we would like the stroke coherence to be on the surfaces of the scene rather than in the image plane, but we would like the stroke size to be roughly what would have been selected for the image plane rather than what would have been chosen for the walls. The difficult challenge is to achieve this goal while rendering images at interactive rates.

We investigate a hybrid NPR/IBR approach. Broadly speaking, the two main issues we address are: 1) constructing an IBR representation suitable for NPR imagery, and 2) developing a IBR prefiltering method to enable rendering of novel NPR images with controllable stroke-size and frame-to-frame coherence in a real-time walkthrough system. These issues are the topics of the following two sections.

3 Image-Based Representation

The first issue in implementing a system based on our hybrid NPR/IBR approach is to choose an image-based representation suitable for storing and resampling non-photorealistic imagery. Of course, numerous IBR representations have been described in the literature (see [5] for a survey); and, in principle, any of them could store NPR image samples of a virtual environment. However, not all IBR representations are equally well-suited for NPR walkthroughs. Specifically, an IBR method for interactive walkthroughs should have the following properties:

A1) **Arbitrary viewpoints:** The image reconstruction method should be able to generate images for arbitrary novel viewpoints within the interior of the virtual environment. This property implies a 5D representation of the plenoptic function capable of resolving inter-object occlusions. It also implies a prefiltered multiresolution representation from which novel views can be rendered efficiently from any distance without aliasing.

A2) **Practical storage:** The image-based representation should be small enough to fit within the capacity of common long-term storage devices (e.g., CD-ROMs), and the working set required for rendering any novel view should be small enough to fit within the memory of desktop computers. This property suggests methods for compressing image samples and managing multi-level storage hierarchies in real-time.

A3) **Efficient rendering:** The rendering algorithm should be very fast so that high-quality images can be generated at interactive frame rates. This property suggests a hardware implementation for resampling.

Additionally, the following properties are important for IBR representations used to store *non-photorealistic* imagery:

B1) **Homeomorphic reprojection:** The mapping of pixel samples onto any image plane should be homeomorphic so that strokes and textures in NPR imagery remain intact during image reconstruction for novel views. This property ensures that our method can work with a wide range of NPR filters.

B2) **Predictable reprojection:** The reprojected positions of pixel samples should be predictable so that the sizes and shapes of strokes in reconstructed NPR images can be controlled. This property allows the system to match the sizes and shapes of strokes in NPR images to the ones intended by the scene designer.

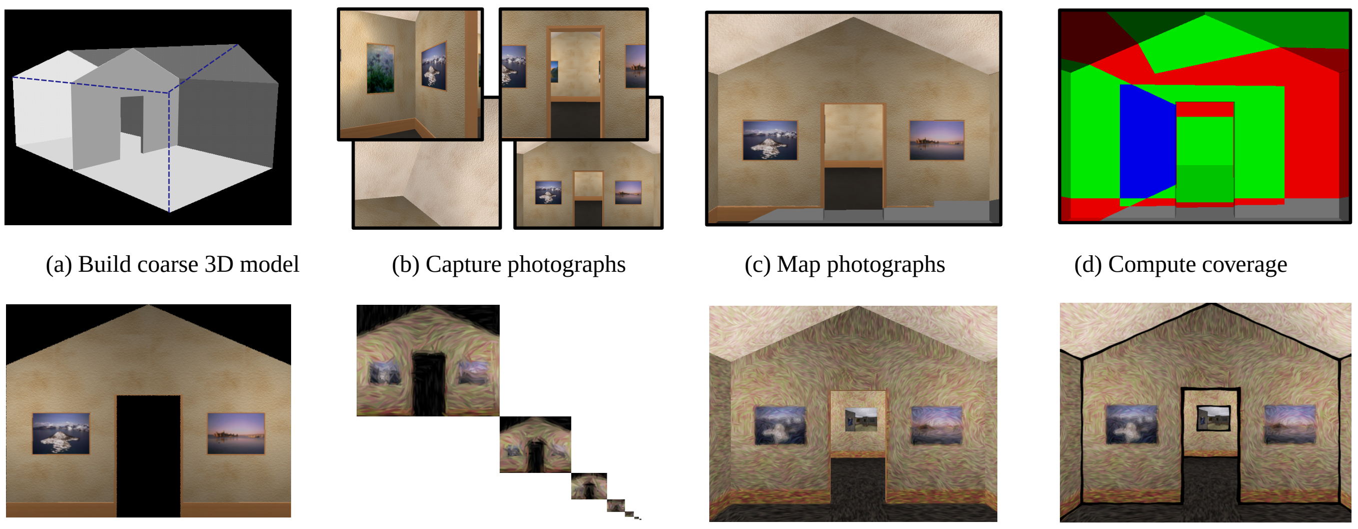
B3) **Filter Flexibility:** Pixel samples should be stored in a form that makes NPR filters simple and easy to implement so that support for multiple NPR styles is practical. This property provides scene designers with the aesthetic flexibility of experimenting with a variety of NPR styles for a single scene.

We have considered several IBR representations. QuickTime VR [2] is perhaps the most common commercial form of IBR, and its cylindrical panoramic images could easily be used to create NPR imagery with our approach. For instance, each panoramic image could be run through an off-the-shelf NPR image processing filter, and the results could be input to a QuickTime VR run-time viewer to produce an immersive NPR experience. While this method may be appropriate for some applications, it cannot be used for smooth, interactive walkthroughs, since QuickTime VR supports only a discrete set of viewpoints, and it would require a lot of storage to represent the interior of a complex environment, thereby violating properties 'A1' and 'A2' above.

Other IBR methods allow greater freedom of motion. However, in doing so, they usually rely upon more complicated resampling methods, which makes reconstruction of NPR strokes difficult for arbitrary viewpoints. As a simple example, consider adding cross-hatch strokes to an image with color and depth values for each pixel. As novel images are reconstructed from this representation, individual pixels with different depths get reprojected differently according to their flow fields; and, consequently, the cross-hatch stroke pattern present in the original depth image disintegrates for most views. This problem is due to a violation of property 'B1,' which is typical of most view-dependent IBR representations, including cylindrical panorama with depth [23], layered depth images [29], light fields [19], Lumigraphs [12], interpolated views [3], etc.

Our approach, based on textures, relies upon a hybrid geometry- and image-based representation. Radiance samples acquired from photographs are used to create textures describing the visual complexity of the scene, while a coarse 3D polygonal model is used to reason about the coverage, resolution, discontinuities, coherence, and projections of radiance samples for any given view. This approach satisfies all of the properties listed above. In particular, surface textures are a very compact form for the 5D plenoptic function, as inter-object occlusions are implicit in the hidden surface relationships between polygons of the coarse 3D model ('A1'). Also, storage and rendering can take advantage of the plethora of previous work in texture mapping [14], including multi-scale prefiltering methods ('A1'), texture compression and paging algorithms ('A2'), and texture rendering hardware implementations ('A3'), which are available in most commodity PC graphics accelerators today.

Textures are especially well-suited for NPR imagery, as the mapping from the texture sample space to the view plane is simply a 2D projective warp, which is both homeomorphic ('B1') and predictable ('B2'). As a consequence, our system can control the sizes and shapes of rendered strokes in reconstructed images by prefiltering NPR textures during a preprocessing step to compensate for the predictable distortions introduced by the projective mapping (the details of this method appear in the following section). Finally, we note that textures provide a simple and convenient representation for NPR filtering, as any combination of numerous commonly available image processing tools can be used to add NPR effects to texture imagery ('B3'). For instance, most of the NPR styles shown in this paper were created with filters in Adobe Photoshop.



(a) Build coarse 3D model (b) Capture photographs (c) Map photographs (d) Compute coverage

(e) Group texture (f) Generate art-maps (g) Run time walkthrough (h) Draw lines

Figure 3: Our process. Steps (a) through (f) happen as pre-processing, enabling interactive frame rates at run-time in steps (g) and (h).

Our specific method for constructing textures from images proceeds as shown in Figure 3a-d. First, we construct a coarsely-detailed polygonal model using an interactive modeling tool (Figure 3a). To ensure proper visibility calculations in later stages, the model should have the property that occlusion relationships between polygons in the model match the occlusion relationships between the corresponding objects in the environment. Second, we capture images of the environment with a real or synthetic camera and calibrate them using Tsai's method [30] (Figure 3b). Third, we map the images onto the surfaces of the polygonal model using a beam tracing method [9] (Figure 3c). The net result is a coverage map in which each polygon is partitioned into a set of convex faces corresponding to regions covered by different combinations of captured images (Figure 3d). Fourth, we select a representative image for each face to form a view-independent texture map, primarily favoring normal views over oblique views, and secondarily favoring images taken from cameras closer to the surface. Finally, we fill faces not covered by any image with a texture hole-filling algorithm similar to the one described by Efros and Leung [8]. Note that view-dependent texture maps could be supported with our method by blending images from cameras at multiple discrete viewpoints (as in [6, 7]). However, we observe that NPR filtering removes most view-dependent visual cues, and blending reduces texture clarity, and thus we choose view-independence over blending in our current system.

4 Non-Photorealistic Filtering

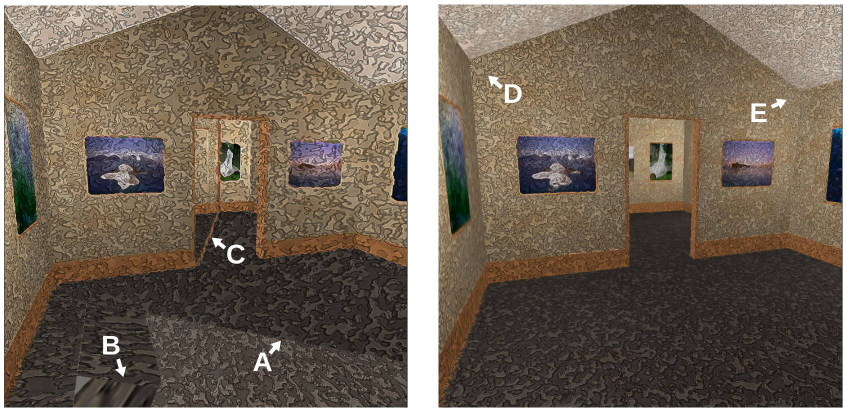
The second step in our process is to apply NPR filters to texture imagery. Sections 4.1 and 4.2 address the two major concerns relating to NPR filtering: avoiding visible seams and controlling the stroke size in the rendered images.

4.1 Seams

Our goal is to enable processing of IBR textures with many different NPR filters. Some NPR filters might add artistic strokes (e.g., “pen and ink”), others might blur or warp the imagery (e.g., “ink blot”), and still others might change the average luminance (e.g., “impressionist”) based on the pixels in the input texture. In all these cases, seams may appear in novel images anywhere two textures processed by an NPR filter independently are reprojected onto adjacent areas of the novel image plane. As a consequence, we must be careful about how to apply NPR filters so as to minimize noticeable resampling artifacts in rendered images.

The problem is best illustrated with an example. The simplest way to process textures would be to apply an NPR filter to each of the captured photographic images, and then map the resulting NPR images onto the surfaces of the 3D model as projective textures (as in [6, 7]). Unfortunately, this photo-based approach causes noticeable artifacts in reconstructed NPR images. For instance, Figure 4a shows a sample image reconstructed from photographic textures processed with a “ink blot” filter in Photoshop. Since each photographic texture is filtered independently and undergoes a different projective warp onto the image plane, there are noticeable seams along boundaries of faces where the average luminance varies (‘A’) and where the sizes and shapes of NPR strokes change abruptly (‘B’). Also, since this particular NPR filter resamples the photographic images with a large convolution kernel, colors from occluding surfaces bleed across silhouette edges and map onto occluded surfaces, leaving streaks along occlusion boundaries in the reconstructed image (‘C’).

We can avoid many of these artifacts by executing the NPR filter on textures constructed for each surface, rather than for each photographic image. This approach ensures that most neighboring pixels in reprojected images are filtered at the same scale, and it avoids spreading colors from one surface to another across silhouette edges. Ideally, we would avoid all seams by creating a single texture image with a homeomorphic map to the image plane for every potential viewpoint. Unfortunately, this ideal approach is not generally possible, as it would require unfolding the surfaces of 3D model onto a 2D plane without overlaps. Instead, our approach is to construct a single texture image for each connected set of coplanar faces (Figure 3e), and then we execute the NPR filter on the whole texture as one image (Figure 4b). This method moves all potential seams due to NPR filtering to the polyhedral edges of the 3D model, a place where seams are less objectionable and can be masked by lines drawn over the textured imagery.



a) NPR photo textures b) NPR surface textures

Figure 4: Applying NPR filters to surface textures avoids seams and warped strokes in reconstructed images.

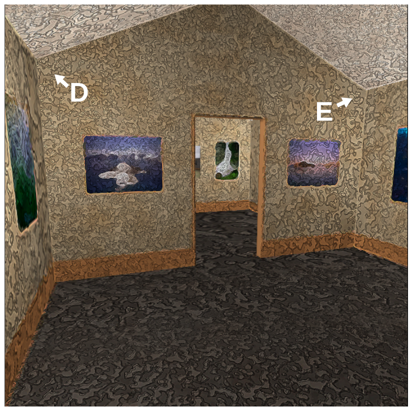


Figure 5: Scene rendered with art-maps. The stroke size remains roughly constant across the image.

4.2 Art Maps

This section addresses the problem of placing strokes into the textures in such a way that we have control over stroke size in the final image. Our challenge is a fundamental tension between frame-to-frame coherence and stroke size appropriate for the image plane. As the user moves toward a surface, the strokes on that surface *must* change in order to maintain an appropriate size in the image plane. Unfortunately, this means that we must either slowly blend from one set of strokes to another set, or suffer from a "pop" when they all change at once. Preferring the former effect, our compromise is to choose slowly-changing strokes, with some amount of blurring as they change, and to allow stroke size to vary somewhat with a *range* of sizes nearly appropriate for the viewing plane.

Our solution relies on the observation that the stroke size problem is analogous to choice of filter for projected imagery in photorealistic environments using conventional texture mapping. As the user navigates a photorealistic environment, the goal of texture mapping hardware is to select for every pixel p a filter f for the texture such that the size of f varies with the size of the texture space pre-image of p. Likewise, our goal is to place each stroke s in the texture such that as the user navigates the environment, the relative sizes of s and f in texture space stay constant. Thus, our strategy for management of stroke size is to leverage the great deal of work on pre-filtering imagery for texture mapping, most notably mip-maps [31]).

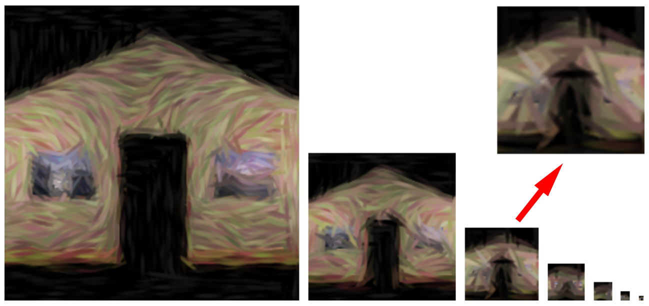


Figure 6: Art-maps work with conventional mip-mapping hardware to maintain constant stroke size at interactive frame rates.

We use a construction that we call "art-maps." The key idea is to apply strokes to each level of the mip-map, knowing that it is suitable for projection to the screen *at a particular size*. Figure 6 shows an example. To create this mip-map hierarchy, we simply filter the photorealistic images as in normal mip-mapping, but then apply an NPR filter to each level independently.

The strokes at each level of the mip-map hierarchy vary in size in powers of two relative to the whole image, just as pre-filtered mip-map levels vary the filter kernel size. Thus, when conventional texture mapping hardware selects a level of the mip-map hierarchy from which to sample a pixel, it will automatically choose a pixel from a set of strokes of the appropriate size. Furthermore, as it blends between levels of the mip-map hierarchy, it will likewise blend between strokes of appropriate size. So the effect is that strokes remain affixed to the surfaces in the scene, but as the user navigates through the environment, the strokes have roughly constant size in the image plane, as shown for example in Figure 5. Note that at locations marked 'D' and 'E' the stroke size is roughly the same. (In contrast, without art-maps, the strokes in these locations varies with the distance between the surface and the camera, as can be seen in Figure 4.) As the user moves toward a wall, the strokes shown for that wall will slowly blend from the strokes in one mip-map level to the next to maintain roughly constant image-space size. As the viewer moves, there is frame-to-frame coherence in the mip-map level chosen for the wall, and therefore there is visual coherence in the strokes. We suffer some amount of blending of strokes, because the mip-map level is generally non-integer; but we prefer this to either popping or lack of control over stroke size. The benefits of art-maps are that they are very simple to implement, and that they permit interactivity by relegating expensive NPR filtering to a preprocess and by exploiting texture mapping hardware for sampling at runtime.

A known problem for conventional mip-maps is that for very oblique polygons the mip-map is forced to choose between aliasing and blurring for one or both of the principle directions [14]. This problem is due to a round filter kernel in image space projected to a very oblong shape in texture space, which forces the use of a kernel that is either correctly sized in its long direction (giving aliasing in the short direction) or correctly sized in its short direction (giving blurring in the long direction). This filter problem manifests itself as stretched strokes when art-maps are applied (Figure 7a). A number of solutions to this problem have been proposed [14] – art-maps will work with any of them that stores multiple prefiltered

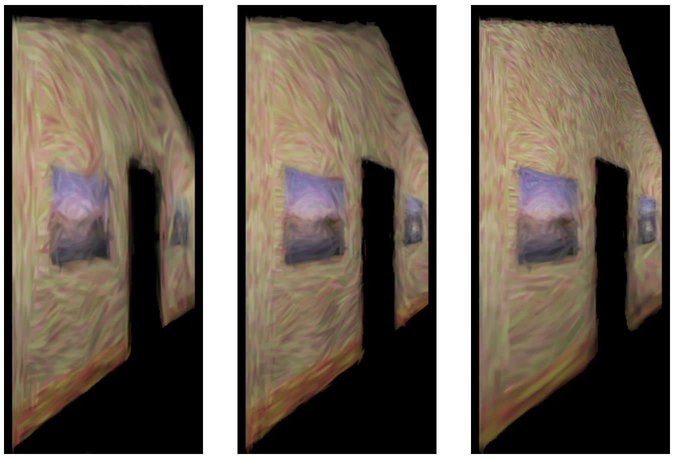

(a) art-maps only (b) with rip-maps (c) varying strokes

Figure 7: Art maps using generalizations of mip-maps.

versions of a texture (e.g., for different perspective warps). We have experimented with a generalization of mip-maps, called "rip-maps" [16]. As shown in Figure 8, rip-maps contain a cascading series of pre-filtered, off-angle images of the texture. An obliquely-projected texture may select one of the off-axis images from the rip-map; in the case of rip-maps with art-maps, the stroke shape will be corrected, as shown in Figure 7b. Our prototype renders this scene by recursively dividing textured polygons, selecting among rip-map textures in the subdivided regions. This method allows interactive control over stroke sizes in different areas of the image plane, as illustrated in Figure 7c; in this example, we use small strokes in the upper part of the image, and smoothly vary stroke size down to large strokes at the bottom of the image. Unfortunately, our current software implementation of rip-mapping is too slow for real-time rendering of complex scenes, and thus we use art-maps with conventional mip-mapping for our interactive walkthrough system. We note that it might still be possible to control the sizes of rendered strokes on a per-surface basis using various texture mapping parameters (e.g., LOD bias) that guide the selection of mip-map levels.

5 Interactive Walkthrough System

During the run-time phase, we simulate the experience of moving through a non-photorealistic environment by drawing surfaces of the coarse 3D model rendered with their art-map textures as the user moves a simulated viewpoint interactively.

Our run-time system loads all art-map levels for all surfaces into texture memory at startup. Then, for every novel viewpoint, it draws surfaces of the 3D model with standard texture mip-mapping hardware using the pre-loaded art-maps (as described in Section 4). The rendering process is fast, and it produces images with relatively high frame-to-frame coherence and nearly constant size NPR strokes, as blending between art-map levels is performed in texture mapping hardware on a per-pixel basis according to estimated projected areas.

To facilitate management of texture memory, we break up large textures into *tiles* before loading them into texture memory, and we execute view frustum culling and occlusion culling algorithms to compute a potentially visible set of surface tiles to render for every novel viewpoint [10]. These methods help keep the working set of texture data relatively small and coherent from frame-to-frame, and thus we can rely upon standard OpenGL methods to manage texture swapping when the total texture size exceeds texture memory.

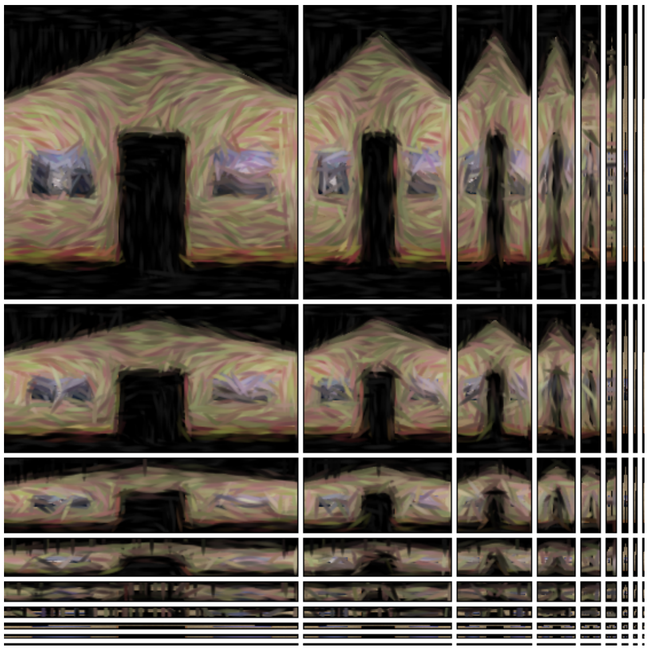


Figure 8: Art-maps can be applied to other, more generalized mip-mapping techniques such as RIP-maps.

Our hybrid geometry- and image-based approach allows us not only to render NPR textured surfaces, but also to augment the resulting images with additional visual information. For example, we sometimes apply photorealistic textures to an object in order to differentiate that object from others in the scene. We also use run-time geometric rendering to highlight interesting features of the environment. For instance, we draw wavy lines over silhouette edges and creases at the intersections of non-coplanar polygons, which helps mask objectionable artifacts due to seams and unnaturally hard edges at polygon boundaries. In our implementation, the lines are drawn as a 2D triangle strip following a sinusoidal backbone along the 2D projection of each visible edge in the 3D model. Since the frequency of the sine function is based on screen space distances, all of the lines drawn have a consistent "waviness," regardless of their orientation relative to the viewer. The lines help to clarify the geometry of the environment, especially when the NPR filter used is very noisy or produces low contrast textures. See Figure 3h for an example.

6 Experimental Results

We have implemented the methods described in the preceding sections in C++ on Silicon Graphics/Irix and PC Windows/NT computers and incorporated them into an interactive system for walkthroughs of non-photorealistic virtual environments.

To test the viability of our methods, we have performed experiments with several virtual environments rendered with different NPR styles. Tables 1 and 2 show statistics logged during our process for three of these environments, two of which are synthetic ("Museum" and "Gallery") and one of which is a real building captured with photographs ("Building"). All times were measured on a Silicon Graphics Onyx2 with a 195MHz R10000 CPU and InfiniteReality graphics.

Examining the timing results in Table 2, we see that the preprocessing steps of our method can require several hours in all. Yet, we reap great benefit from this off-line computation. The re-

Model name	Number of polygons	Surface area (inches^2)	Number of photos	Number of faces	Number of textures	Total MBs of textures	Total MBs of art-maps
Gallery	192	2,574,400	46	414	73	82	109
Museum	76	421,520	93	282	42	104	138
Building	201	931,681	18	815	114	118	157

Table 1: Quantitative descriptions of test environments and preprocessing results.

	Preprocessing								Run-time		
Model name	Capture photos	Calibrate photos	Map photos	Create textures	Hole filling	Create art-maps	Run NPR filter	Total preprocessing	Draw images	Draw lines	Total per frame
Gallery	1m 40s	—	0.4s	3m 30s	2h 02m	10m	30m	2h 47m	0.017s	0.025s	0.042s
Museum	1m 52s	—	0.8s	2m 53s	3h 34m	8m	40m	4h 26m	0.017s	0.014s	0.031s
Building	2h	2h	5.8s	4m 22s	3h 40m	14m	50m	8h 48m	0.056s	0.037s	0.093s

Table 2: Timing results for each stage of our process.

sult is visually compelling imagery rendered at interactive frame rates with high frame-to-frame coherence during run-time. Average frame refresh times measured during interactive walkthroughs of each model are shown in the right-most column of Table 2. The corresponding frame rates range from 11 to 32 frames per second, which are adequate to provide a convincing illusion of presence as the user moves interactively through a non-photorealistic environment.

Another result is the demonstration of our system's flexibility in supporting interactive walkthroughs in many NPR styles. Figures 9a-c show screen shots of the walkthrough program with the "Museum" environment after processing with different NPR filters. Creating each new set of NPR textures took around 40 minutes of preprocessing time, as only the last step of the preprocess ("run NPR filter") had to be re-done for each one. Then, the run-time program could immediately provide interactive walkthroughs in the new style. Figures 9d-f show images of the "Building" environment rendered in a watercolor style from different viewpoints. Each image took less than 1/10th of a second to generate. Notice how the size of the strokes in all the images remains relatively constant, even for surfaces at different distances from the viewer.

The primary limitation on the complexity of virtual environments and the resolution of imagery rendered with our system is the capacity of graphics hardware texture memory. In order to maintain interactive frame rates, all texture data for every rendered image must fit into the texture cache on the graphics accelerator (64MB in our tests). As a result, the number of surfaces in the virtual environment and the resolution of captured textures must be chosen judiciously. So far, we have generally constructed group textures with each texel corresponding to a 2 by 2 inch region of a surface, and we decompose group textures into 512 by 512 pixel tiles that can be loaded and removed in the texture cache independently. With these resolutions, our test environments require between 109MB and 157MB of texture data with art-maps (see the right-most column of Table 1), of which far less than 64MB is required to render an image for any single novel viewpoint (due to view frustum culling and occlusion culling). In our experiments, we find that the standard OpenGL implementation of texture memory management is able to swap these textures fast enough for interactive walkthroughs, at least on a Silicon Graphics Onyx2 with InfiniteReality graphics. While the frame rate is not perfectly constant (there are occasionally "hiccups" due to texture cache faults), the frame rate is usually between 10 and 30 frames per second – yielding an interactive experience for the user. More sophisticated texture management and compression methods could be used to address this issue in future work.

7 Conclusion

This paper describes a system for real-time walkthroughs of non-photorealistic virtual environments. It tackles the four main challenges of such a system – interactivity, visual detail, controlled stroke size, and frame-to-frame coherence – through image-based rendering of non-photorealistic imagery. The key idea is that an image-based representation can be constructed off-line through a sequence of image capture and filtering steps that enable efficient reconstruction of visually detailed images from arbitrary viewpoints in any non-photorealistic style. The technical contributions of this work include a method for constructing NPR textures that avoids seams in novel images and a multiscale texture representation (*art-maps*) that provides control over the size of strokes during interactive rendering. This work suggests a number of areas for future investigation:

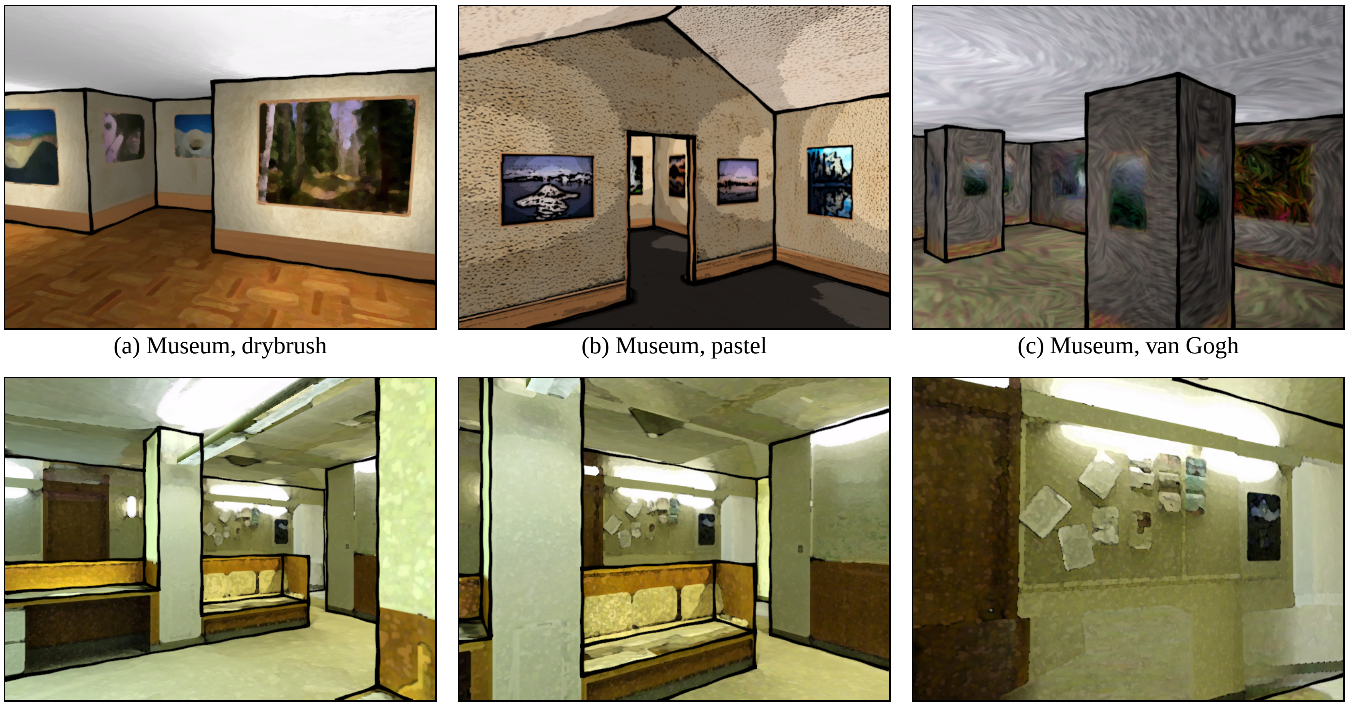
Augmenting the scene with geometry-based elements. Real-time NPR rendering of simple geometric objects in the scene – perhaps architectural accents such as a plant or a chair rendered in the NPR styles of Gooch *et al.* [11] or Kowalski *et al.* [18] – would enhance the sense of immersion while not greatly slowing our system.

View-dependent rendering. We have observed that many view-dependent geometric and lighting effects are visually masked by non-photorealistic rendering (see Section 3). Nonetheless, view-dependent texture mapping (e.g. [6, 7]) offers an opportunity to capture these effects for even better fidelity to the environment.

Better stroke coherence. As mentioned in Section 4.2, runtime blending between neighboring levels of the mip-map hierarchy causes visual blending between strokes in the art-maps. It may be possible to achieve better coherence between neighboring levels of the mip-maps, most likely by designing customized NPR filters that deliberately assign strokes in multiple levels of the art-maps at once. The desired visual effect might be that strokes grow and eventually split apart, rather than fading in, as the user approaches a surface.

Acknowledgements

The authors would like to thank several people for their assistance with this project: Lee Markosian taught us how to draw wavy lines quickly using triangle strips; Reg Wilson provided the implementation of Tsai's camera calibration algorithm; and John Hughes provided helpful discussion. This work was supported in part by Alfred P. Sloan Foundation Fellowships awarded to Adam Finkelstein and Thomas Funkhouser, an NSF CAREER grant for Adam Finkelstein, and generous gifts from Microsoft Corporation.


(a) Museum, drybrush (b) Museum, pastel (c) Museum, van Gogh

(d) Building, watercolor (e) Building, watercolor (f) Building, watercolor

Figure 9: Images of artistic virtual environments rendered during an interactive walkthrough.

References

[1] Buck, I., Finkelstein, A., Jacobs, C., Klein, A., Salesin, D. H., Seims, J., Szeliski, R., and Toyama, K. Performance-driven hand-drawn animation. *Proceedings of NPAR 2000* (June 2000).

[2] Chen, S. E. Quicktime VR - An image-based approach to virtual environment navigation. *Computer Graphics (SIGGRAPH 95)*, 29–38.

[3] Chen, S. E., and Williams, L. View interpolation for image synthesis. *Computer Graphics (SIGGRAPH 93)*, 279–288.

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

[5] Debevec, P. Image-based modeling, rendering, and lighting. *Course #35, SIGGRAPH 2000 Course Notes* (July 2000).

[6] Debevec, P. E., Taylor, C. J., and Malik, J. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics (SIGGRAPH 96)*, 11–20.

[7] Debevec, P. E., Yu, Y., and Borshukov, G. D. Efficient view-dependent image-based rendering with projective texture-mapping. *Eurographics Rendering Workshop* (June 1998), 105–116.

[8] Efros, A. A., and Leung, T. K. Texture synthesis by non-parametric sampling. *IEEE International Conference on Computer Vision* (1999).

[9] Funkhouser, T. A. A visibility algorithm for hybrid geometry- and image-based modeling and rendering. *Computers and Graphics, Special Issue on Visibility* (1999).

[10] Funkhouser, T. A., Teller, S. J., Sequin, C. H., and Khorramabadi, D. The UC Berkeley system for interactive visualization of large architectural models. *Presence 5*, 1 (January 1996).

[11] Gooch, A., Gooch, B., Shirley, P., and Cohen, E. A non-photorealistic lighting model for automatic technical illustration. *Computer Graphics (SIGGRAPH 98)*, 447–452.

[12] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. The lumigraph. *Computer Graphics (SIGGRAPH 96)*, 43–54.

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

[14] Heckbert, P. Survey of texture mapping. *IEEE Computer Graphics and Applications* (Nov. 1986).

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

[16] Hewlett Packard. HP PEX Texture Mapping, www.hp.com/mhm/WhitePapers/PEXtureMapping/PEXtureMapping.html.

[17] Horry, Y., ichi Anjyo, K., and Arai, K. Tour into the picture: Using a spidery mesh interface to make animation from a single image. *Computer Graphics (SIGGRAPH 97)*, 225–232.

[18] Kowalski, M. A., Markosian, L., Northrup, J. D., Bourdev, L., Barzel, R., Holden, L. S., and Hughes, J. Art-based rendering of fur, grass, and trees. *Computer Graphics (SIGGRAPH 99)*, 433–438.

[19] Levoy, M., and Hanrahan, P. Light field rendering. *Computer Graphics (SIGGRAPH 96)*, 31–42.

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

[21] Manocha, D. Interactive walkthroughs of large geometric databases. *Course #18, SIGGRAPH 2000 Course Notes* (July 2000).

[22] Markosian, L., Kowalski, M. A., Trychin, S. J., Bourdev, L. D., Goldstein, D., and Hughes, J. F. Real-time nonphotorealistic rendering. *Computer Graphics (SIGGRAPH 97)*, 415–420.

[23] McMillan, L., and Bishop, G. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH 95)*, 39–46.

[24] Meier, B. J. Painterly rendering for animation. *Computer Graphics (SIGGRAPH 96)*, 477–484.

[25] Mizuno, S., Okada, M., and ichiro Toriwaki, J. Virtual sculpting and virtual woodcut printing. *The Visual Computer 14*, 2 (1998), 39–51.

[26] Ostromoukhov, V. Digital facial engraving. *Computer Graphics (SIGGRAPH 99)*, 417–424.

[27] Saito, T., and Takahashi, T. NC machining with G-buffer method. *Computer Graphics (SIGGRAPH 91)*, 207–216.

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

[29] Shade, J. W., Gortler, S. J., wei He, L., and Szeliski, R. Layered depth images. *Computer Graphics (SIGGRAPH 98)*, 231–242.

[30] Tsai, R. Y. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation 3*, 4 (Aug. 1987), 323–344.

[31] Williams, L. Pyramidal parametrics. *Computer Graphics (SIGGRAPH 83)*, 1–11.

[32] Winkenbach, G., and Salesin, D. H. Computer-generated pen-and-ink illustration. *Computer Graphics (SIGGRAPH 94)*, 91–100.

[33] Winkenbach, G., and Salesin, D. H. Rendering parametric surfaces in pen and ink. *Computer Graphics (SIGGRAPH 96)*, 469–476.

[34] Wong, M. T., Zongker, D. E., and Salesin, D. H. Computer-generated floral ornament. *Computer Graphics (SIGGRAPH 98)*, 423–434.

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