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

Abstraction and Depiction of Sparsely Scanned Outdoor Environments

Hui Xu†, Nathan Gossett† and Baoquan Chen†

University of Minnesota at Twin Cities

Abstract

This paper describes various techniques and applications of rendering three-dimensionally digitized outdoor environments in non-photorealistic rendering styles. The difficulty in rendering outdoor environments is accommodating their inaccuracy, incompleteness, and large size to deliver a smooth animation without suggesting the underlying data deficiency. Standard rendering approaches often expose and inadvertently emphasize missing and noisy data, producing unpleasant images. Our use of non-photorealistic rendering allows us to de-emphasize these problems and produce aesthetically pleasing images. The key approach discussed in this paper employs artistic drawing techniques to illustrate features of varying importance and accuracy. We use point-based representations of the scanned environments and operate directly on the point-based models for abstraction and rendering. We develop a unified framework for producing sketchy, profile, painterly, cartoon, and intermingled styles. We describe a level-of-detail data structure, the continuous resolution queue, to promise coherent and consistent animation. We also leverage modern graphics hardware to achieve interactive rendering of large scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.0 [COMPUTER GRAPHICS]: General I.3.3 [COMPUTER GRAPHICS]: Picture/Image Generation[Digitizing and scanning]

1. Introduction

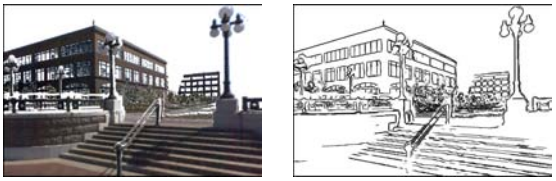
Capturing and animating real-world objects and scenes has attracted increasing research interest. To offer unconstrained navigation of these scenes, 3D representations are first needed. Advances in laser scanning technology are making this 3D acquisition feasible for increasingly large objects. The aim of the methods described in this paper is to scan outdoor environments and deliver interactive stylized navigation through them.

Outdoor environment scans have the following properties: **incompleteness:** a complete scan of every object in the environment is impossible to obtain due to the usual obstructions caused by intervening objects, and the constrained accessibility of the scanner; **complexity:** natural objects, such as trees and plants are complex in terms of their geometric shapes; **inaccuracy:** distant objects are less accurate due to scanning hardware limitations, and plants and trees can be moved by wind during the scanning process; **non-uniform sampling rate**; and **large data size**. Figure 1(a) shows an attempted photorealistic point-based rendering of such a scan, where holes and noise are apparent.

These properties raise unprecedented challenges for existing point-based rendering methods. Current scanning work deals primarily with short to medium range indoor scanning, and has focused on generating complete polygon meshes from point clouds. Fitting a complete polygon mesh to outdoor data is a daunting task. Smoothing noise and patching holes can be tedious and even prohibitive processes. Moreover, a fundamental problem of this approach is that such polygon meshes remove all the uncertainties and holes that existed in the original data. Thus, rendering a mesh provides a false impression of a model's accuracy and clarity.

In this paper, we opt to generate animations of outdoor scenes using artistic illustration, or non-photorealistic rendering (NPR). Artistic illustration, unlike traditional photorealistic rendering, can aesthetically depict objects of different importance by using different accuracy and drawing

† {hxu,gossett,baoquan}@cs.umn.edu



(a) (b)

Figure 1: *(a) Data deficiency is apparent in an attempted photorealistic rendering of outdoor scanned data. (b) It is less noticeable in a sketch rendering of the same data.*

styles [SPSF94]. Once a scene is depicted in these artistic styles, a viewer's expectations of scene accuracy are automatically reduced. Therefore, missing details, whether large or small, become less noticeable as demonstrated in the comparison of the two images in Figure 1. Moreover, even for a situation with accurate geometry, artistic illustration can still be more desirable. This is evident for many applications such as architectural design, where a certain level of abstraction better conveys the essence of the scene.

Our goal is to develop algorithms to generate NPR styles *directly* from point-based representations without first obtaining a complete mesh. This makes existing NPR methods (usually based on a well defined 3D geometry) not applicable. In this paper, we summarize our methods to process scanned data, build point-based models, extract features, and generate various artistic rendering styles [XC04, XGC04, XNYC04]. We describe a unified framework to render four basic styles: sketch, profile line, painterly, and cartoon. Many additional styles can be created by intermingling the basic styles in a single scene. We have mapped our algorithms onto commodity graphics hardware, leveraging the latest vertex and pixel programming features in order to provide an interactive navigation experience.

The rest of the paper is organized as follows. After a brief discussion of previous work (Section 2) and our data acquisition and processing pipeline (Section 3), we introduce the framework for non-photorealistic rendering (Section 4). We discuss four illustration styles - sketchy, profile lines, painterly, and cartoon in more detail in Section 5. We then present our results (Section 6) and demonstrate some applications of PointWorks (Section 7). Finally, we conclude with discussions and our plans for future work (Section 8).

2. Previous Work

Much of the current work in 3D scanning has been devoted to short or medium range scanning, particularly of statues and other historical artifacts [LPC*00, BRM*02]. These projects use relatively controlled scanning processes in that they deal with stationary objects small enough to scan from all angles, often under controlled lighting conditions. Scanning projects usually produce complete polygon meshes of the scanned objects. Acquisition of outdoor environments has been largely limited to image-based techniques [MB95]. These techniques are limited by a lack of 3D geometry knowledge, and therefore have constrained navigation.

Non-Photorealistic rendering has become an important branch of computer graphics in recent years. Researchers have extracted principles employed by artists and used them to guide computer-based art simulation. Many artwork styles have been explored, such as pen and ink [WS94, SWHS97], painting [Mei96, Her98], informal sketching [RC99], cartoon [LMHB00], and abstract stylization [DS02]. Cornish et al. [CRL01] have outlined several commonalities in NPR techniques for generating many highly varied artistic effects. Except for image based methods, most 3D techniques assume a complete polygon model as a starting point.

Lately point-based modeling and rendering have gained significant momentum, but most of the research in this area has been focused on improving rendering efficiency and/or visual quality [PZvBG00, RL00, RPZ02, DVS03]. Compared to polygon-based NPR techniques, relatively little attention has been devoted to point-based models. Gumhold et al. [GWM01] and Pauly et al. [PKG03] have developed methods for extracting and depicting geometric features from point models that can be used for NPR, but their methods favor a clean and complete point set. The main challenge in performing NPR on scanned point clouds is the inherent complexity, incompleteness, inaccuracy and inconsistent sampling rate of such data.

3. Data Acquisition and Processing

In this section, we provide a brief overview of our data acquisition and processing pipeline for the sake of completeness as some of the operations are conventional. The pipeline used in our system mainly includes acquisition, registration, segmentation, geometry estimation, and model construction.

- *Acquisition* - Our scanning device is Riegl Inc's LMS-Z360i 3D imaging sensor. It can measure a distance up to 200m with 12mm precision. A full panoramic scan ($360^o \times 90^o$) at 3000×792 resolution takes around four minutes and produces data in the form of range, intensity, and color images simultaneously.
- *Registration* - Multiple scans are needed to reduce occlusion artifacts for outdoor environments. We first align scans pairwise using the ICP algorithm [BM92], and then transform the scans into a global coordinate system.
- *Segmentation* - The scanned data is stored in the form of range images, where each pixel represents a point in 3D. The purpose of segmentation is to divide the points into meaningful objects such as buildings, trees, etc. This is essential for generating intermingled styles (see Figure 12). As byproducts, efficient view frustum culling and continuous resolution queues (see Section 4.4) can be effectively implemented. In our system, segmentation is done for every scan by a range image segmentation method (e.g. [HTZ04]) with user assistance.

- *Geometry estimation* - Geometric information regarding each point, such as size and normal direction, is necessary for operations such as visibility detection, back-face culling, stroke placement, etc. The size of a point can be estimated based on its range value and its pixel azimuth/altitude angle. The normal of a point is estimated by calculating a least-squares fitting plane to all points adjacent to it. As will be discussed in Section 4.1, the normal estimation error is also utilized to detect features for artistic rendering.
- *Model construction* - Based on the aforementioned procedures, a separate point-based model is constructed for each individual object by merging multiple scans and removing redundant points.

Starting from the next section, we present strategies to extract features directly from the point models and introduce the methods used to render them in artistic styles.

Figure 2: *Feature extraction of a panoramic scan. From top to bottom: gray-scale-encoded feature degrees, feature points (red: directional points; blue: non-directional points), and dithering points.*

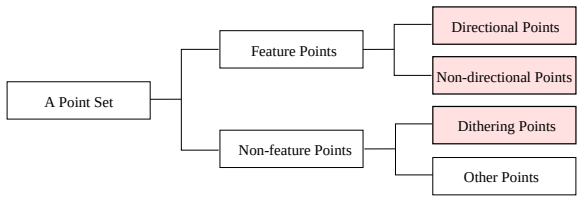


Figure 3: *Classification of points (points in the shaded rectangles are used in NPR).*

4. Feature Detection and General Rendering Pipeline

Given the inaccurate and incomplete data of our scans, we first perform classification of individual points according to their likelihood of belonging to an area with fine details, such as silhouette edges, sharp corners, and creases. Although there are many metrics that could be used to "define" this likelihood, we have found that many of the features we are interested in occur at places where the error variation of the estimated normal is high. Since we need to compute a normal for each point to improve rendering quality, we can identify feature points as a byproduct of a required operation.

4.1. Feature Degree

We use the term *feature degree* to denote the likelihood that a point belongs to a feature position. First, we utilize the normal error variation to define the feature degree for a point. Note that we refer to the *variation* in normal error rather than the actual error value. To understand this, consider the case of a sphere. Since a sphere is a uniform surface, it has no view-independent features. For any given point on the surface of the sphere, a normal estimation error would be high. However, since the error is uniform for the entire sphere, we are able to claim that there are no sharp corners on the sphere. To measure the normal error variation, we first encode the errors of individual points into a gray scale image using the same 2D coordinates in their original range image. We then apply an edge detection filter to the error encoded image. The filtering result can be directly used to define the feature degree.

The feature degree defined above is insufficient to describe non-geometrical features related to the color or intensity of the points. For example, this process does not detect a boundary between two areas with different colors on a flat surface. To detect color based features, we can augment the feature degree by also using an edge detection filter on the color image associated with the scan to obtain a color gradient value for each point. By assigning weights (e.g., 0.5 and 0.5) to the plane fitting error variation and the gradient value, we can assign a final feature degree to each point, with large feature degrees denoting points with high variation in normal estimation error and large color shifts. The top image in Figure 2 shows an example of the encoded feature degrees, where darker color indicates higher feature degree.

4.2. Point Classification

We further categorize points to determine the manner in which each point is drawn at rendering time. Figure 3 shows the overall point classification used in PointWorks. The points are classified as follows.

Once a feature degree is decided for each point, a threshold is then specified to classify the points that are of high feature degree to be *feature points* and the rest to be *non-feature points*. For each feature point, we further attempt to define its local direction, which is used to orient a stroke during rendering. This local direction is determined by calculating a 3D least-squares fitting line for the point's neighboring

feature points. In addition to using this direction for stroke placement, the estimation error of this line fitting is further used as a confidence rating. Points with a confidence rating below a certain threshold are classified as *non-directional feature points* or simply *non-directional points*, while those with high confidence ratings are classified as *directional feature points* or simply *directional points*. For example, points situated along the edge of a building would probably be fitted with a line and have a high confidence rating. These points would then be identified as directional points. Points representing the leaves of a tree would register as feature points since they would not form a uniform surface. However, leaf points would not be as likely to be positioned in a straight 3D line. The lines fitted to these points would have low confidence ratings, and these points would be classified as non-directional points. The middle image in Figure 2 shows an example of the feature points extracted from a panoramic scan of the University of Minnesota campus, in which directional points are drawn in red and non-directional points are drawn in blue.

To illustrate an object's shading tone, a subset of non-feature points that create a dithered estimation of the surface color may also be depicted. These points, termed *dithering points*, are selected through a dithering or halftoning operation [Ost01]. The bottom image in Figure 2 shows an example of the extracted dithering points.

4.3. General Rendering Pipeline

Once the point classification is done, points of different classification are depicted using strokes of various styles. The stroke style, choice of rendering points, and choice of rendering order determine the style of an output image as will be discussed in Section 5.

Rendering a subset of points as mentioned above does not guarantee correct visibility. For example, background objects may leak through the foreground objects. To address this issue, in our general rendering pipeline, each image is rendered in two passes. The first pass is used to generate a depth buffer called a visibility mask [PZvBG00], and the second pass renders the selected set of points. In the second pass, the depth buffer is initialized using the visibility mask and its updating is disabled. Moreover, since points are typically rendered using textured strokes, alpha blending is enabled in this pass to lessen artifacts.

4.4. Continuous Resolution Queue

We have also developed a data structure that allows us to easily control the density of points on the screen and produce frame-to-frame coherency during animation. If we were to render every point in every frame, the density of points on the screen would be dependant on the distance of each object from the viewpoint. In addition, if we were to only render some subset of points for each frame, choosing a substantially different subset of points in successive frames would cause flickering. We solve this problem by using a data structure called a *continuous resolution queue* to store points. After the points have been grouped by object or classification, the points in each group are randomly reordered into a linear queue, with separate queues for each object or surface. The purpose of this random reordering is to achieve an even distribution of points with regard to position. It is important to note that this reordering is done only once as a preprocessing step. After the queue is created, the ordering of the points within the queue remains the same every time the queue is accessed.

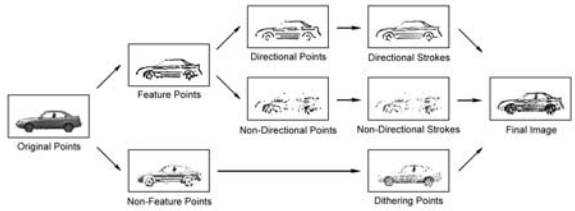


Figure 4: *The Sketchy Rendering Pipeline.*

This technique allows us to display a continuous level of detail for each object. In order to maintain a uniform density of points on the screen, the projected screen space of each object in our model is determined at rendering time. Depending on the amount of screen space occupied, only a certain number of the points for each object are rendered in the current frame. By always starting from the beginning of the queue, adjacent levels of detail use identical sets of points other than the one point that is added or removed between levels. This produces frame-to-frame coherency during animation.

5. Artistic Rendering

We strive to generate mainly four artistic styles: sketch, profile line, painterly and cartoon. We generate these styles by placing discrete elements such as paint strokes or stipples on a selected set of points during the second rendering pass. Different styles can be created by varying the stroke placement scheme (location and order) as well as the depiction of the strokes themselves.

5.1. Sketchy Rendering Styles

One of the styles we can produce with our framework is abstract sketches. We illustrate different classifications of points by employing different stroke styles as demonstrated in Figure 4.

Directional points are rendered with line segments or stylized line strokes in general. By orienting lines along the computed directions (see Section 4.2) of directional points,

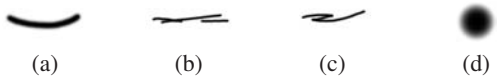


Figure 5: *(a) (b) (c) directional stroke textures; (d) point sprite texture.*

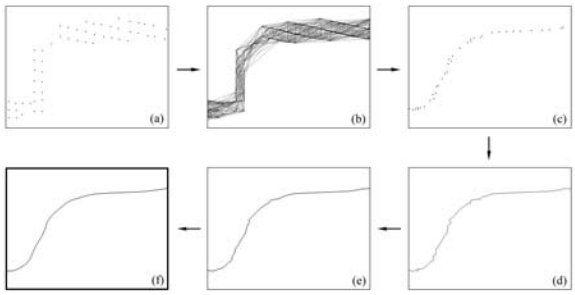


Figure 6: *The Profile Line Construction Process.*

we can produce a sketchy approximation of the edge contour. By using various textures to produce stylized lines on each point, we can control the appearance of the directional strokes. Figure 5(a)-(c) shows some example stroke textures used in our system.

Non-directional points may be rendered with isotropic textures such as splats as shown in Figure 5(d) to convey non-directional details. Alternately, very small strokes placed in a uniform direction or hatches may be used for non-directional points. In addition, dithering points may be rendered to create a dithered estimation of the surface color for geometrically uniform areas. Since dithering points do not represent directional features, they can be rendered the same way as non-directional points.

Given the benefits of using our continuous resolution queues, we can guarantee that the strokes placed in each frame are uniformly distributed at the desired density. We can also be assured that successive frames will use strokes placed on the same points to ensure coherency during animation.

5.2. Profile Lines

Images generated using the sketchy rendering pipeline contain many disconnected strokes. Using longer connected strokes to depict only object profiles, however, can achieve a higher degree of abstraction.

Object profiles are obtained by processing *directional* points. Figure 6 shows the process from feature points to profile lines. Each directional feature point (Figure 6a) is connected to all directional points in its neighborhood, forming connected graphs (Figure 6b). The 3D location of each directional feature point is then shifted in the direction of any point it is connected to as a preliminary smoothing step (Figure 6c). Any points located within a certain distance of each

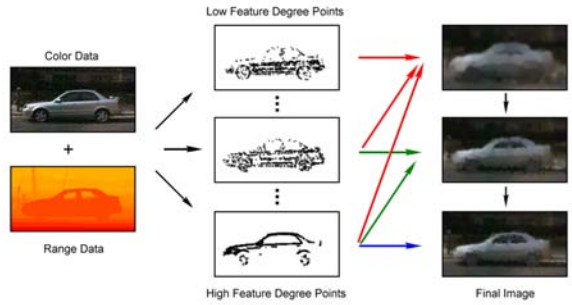


Figure 7: *The Painterly Rendering Pipeline.*

other are combined into a single point. A Minimum Spanning Tree (MST) is constructed for each graph (Figure 6d), with 3D distance and edge direction determining the weight for each potential edge. The 3D shift and point combination in the previous step significantly aids in constructing MSTs that accurately describe smooth edges. These MSTs are then used as an estimate of each object's abstract form.

B-splines are used to create a smoother appearance (Figure 6e). In addition to the normal smoothing associated with B-splines, another smoothing pass is made along each spline path, shifting point positions to eliminate high-frequency deviations (Figure 6f). The profile lines are constructed as a pre-processing step. These profiles exist as 3D structures rather than being constructed as 2D lines in image space.

5.3. Painterly Rendering Styles

The basic strategy we use to render images in a painterly style is to first render coarse details as large brush strokes, and then finer details as smaller brush strokes. We make use of the feature degree to identify points representing finer and coarser details (Figure 7).

A straightforward approach to implementing the iterative process would be to use a single classification of points for each stroke size, with low feature degree points producing large brush strokes and high feature degree points producing small brush strokes. We discovered, however, that we were able to obtain better coverage by using points with a wide range of feature degrees to produce large brush strokes. We then gradually shift to using only the highest feature degree points for the smaller strokes. For example, the crown of a tree consists mostly of high feature degree points. If we were to use only low feature degree points for the foundation layer of large brush strokes, the crown of the tree would consist only of small strokes. This would result in holes in the final image. Obtaining proper coverage for the whole iteration process means that the high feature degree points get reused many times.

To take advantage of this reuse, we sort all points (including non-feature points) from low to high feature degree and

Figure 8: *Enhancements of Painterly Rendering using stencil masking and directional strokes.*

divide the list evenly into bins. Each bin is then randomly ordered and placed in a continuous resolution queue. We reuse these queues for the entire process rather than maintaining a separate point list for each iteration.

The number of iterations is the same as the number of bins, but recall that each iteration does not use only one bin of points. Instead, the first iteration uses all of the bins, and each subsequent iteration uses one less bin until the last iteration, which uses only the bin with the highest feature degree points. The number of points rendered from each bin is based on the calculated screen coverage of each object so that the density of points on the screen remains consistent. Figure 7 illustrates the iteration process.

Note that since our bins are continuous resolution queues, the same high feature degree points that are used in early iterations are also used in later iterations. Each successive iteration uses more points from the bins that are left, so the remaining queues deliver the same points as the previous iteration, plus some additional points.

Enhancements of Painterly Rendering:
First, in order to maintain reasonable color transitions between strokes of different sizes, for each brush stroke we locate all points in its coverage and blend their colors. The blended colors are stored compactly and retrieved according to the current iteration at run-time. Second, we use a simple sky box to produce an appropriate background image for the environment. Third, we utilize a stencil mask to prevent out-of-bounds drawing above the skyline caused by large brush strokes in the first few iterations. The stencil test is disabled for the smaller brush strokes to permit some minor out-of-bounds drawing. This prevents the skyline from being too clean of an edge. The results of this operation can be seen by comparing the left and middle images in Figure 8. Finally, we place directional strokes on top of the finished painting to enhance the detail of a painterly style image as shown in the right image of Figure 8.

5.4. Cartoon Rendering Styles

Another style we can produce with our framework is cartoon rendering. Cartoon style images are created by drawing silhouette lines of an object in thick line styles and filling the object with a limited number of colors. The colors can be calculated and assigned to the object as a preprocessing step. The silhouette lines, however, need to be determined on-the-fly since they are view-dependent. Unlike polygon-based models, detecting and rendering silhouettes for point-based models, especially the outdoor scanned data, is a challenging task due to the lack of connectivity information.

We have developed a method for rendering silhouette lines of point-based models in real-time. Recall that we use a two-pass rendering pipeline in our framework. Points are rendered as opaque disks to generate a visibility mask in the first pass. The frame buffer is then cleared and a selected subset of points are rendered using strokes of various styles in the second pass. The basic strategy of our new method is to slightly *enlarge* each opaque disk in the first pass. In the second pass, unlike rendering other styles, the frame buffer is *not* cleared and *all* points are rendered. Each point is rendered as a splat with its regular size (hence smaller than its corresponding opaque disk size) on top of the frame buffer. Since the projection size is calculated to ensure hole-free surfaces, the opaque disks will be overdrawn by splats from the second pass. However, when the visibility mask is used, the extra edges of the enlarged opaque disks from the first pass will *not* be overdrawn when they are on depth discontinuities, usually at the silhouette boundaries. Therefore, the silhouettes are automatically revealed.

The colors of the splats in the second pass determine the appearance of the interior visible surfaces. To generate a cartoon style, we render the interior surfaces as large regions of constant colors. Since the scanned points are stored in a 2D form, the points can be partitioned into regions with homogeneous colors by applying an image segmentation method. Such segmentation is done in a preprocessing stage. At run-time, we use the average color in a segmented region to render all points in that region during the second pass. In this way, large regions of constant color are naturally formed in the final image.

6. Results

We have implemented our system using DirectX 9.0 on an nVidia GeForce FX 6800 graphics card with 256MB video memory. Our test PC has an Intel XEON 2.4GHz processor, 2GB of main memory, and runs Windows XP. All example images presented in this section are initially rendered with 1600×1200 resolution unless otherwise stated.

Figure 9 shows some examples of a data set rendered in different sketchy styles and profile lines. The sketchy styles are generated by depicting points of different classifications using different rendering primitives as discussed in Section 5.1. In Figure 9(a), only directional points are rendered using the textured stroke specified in Figure 5(a). Figure 9(b) adds non-directional points rendered as short line segments in a constant screen direction (45 degree). Dithering points are added and also rendered as short line segments to create a hatching effect in Figure 9(c). Clearly, different primitives cast different impressions of the same environment. Textured strokes with strong directionality likely convey features of higher confidence while short line segments drawn

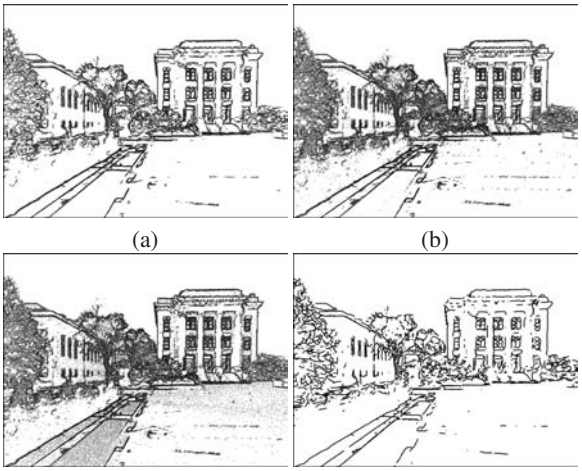


Figure 9: *Demonstration of sketchy and profile styles.*

Figure 10: *Demonstration of a painterly style.*

in a uniform direction likely convey either uncertain features or simply the tones of surfaces. To achieve a higher level of abstraction, the objects are rendered in profile lines as shown in Figure 9(d).

Figure 10 and Figure 11 demonstrate the painterly style and the cartoon style that our system is capable of generating. Figure 10 shows an image rendered in a painterly style enhanced by colored directional edge strokes. The sky background is simply generated through a sky box, and is not stylized. Figure 11 shows a campus scene rendered in a cartoon style. The silhouette rendering method described in Section 5.4 is used to generate the black thick outlines.

Since we use the same two-pass rendering pipeline to generate the above styles, it is possible for us to intermingle different styles within the same image, as shown in Figure 12. By choosing different styles for various objects, we can cause some to stand out, while others recede into the background. Our unified framework allows us to adjust the styles of individual objects on-the-fly. In the image, user-specified non-essential objects such as trees, bushes, and the ground

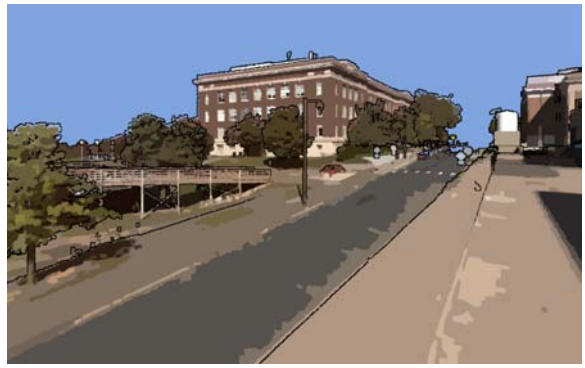


Figure 11: *Demonstration of a cartoon style.*

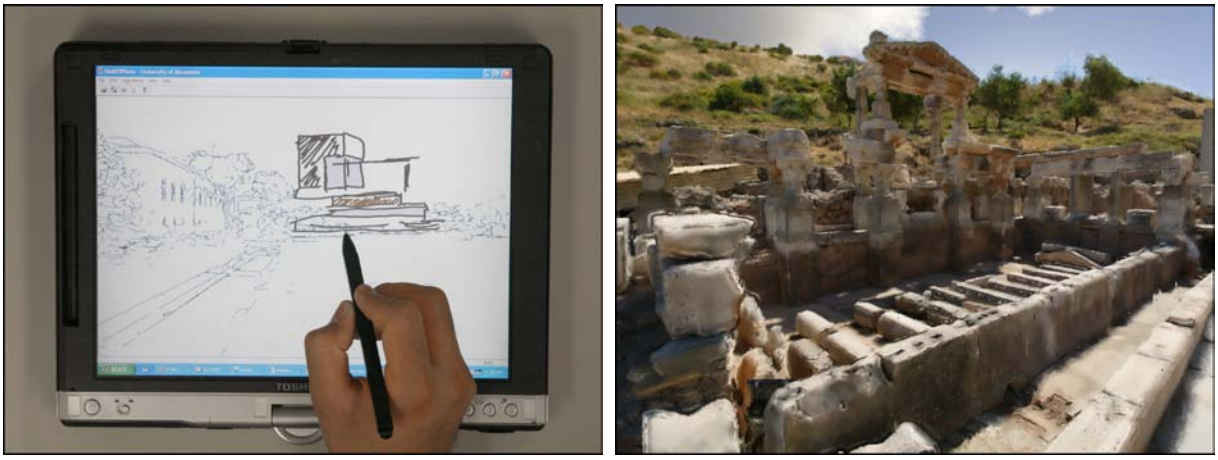
Figure 12: *Demonstration of an image rendered using intermingled styles.*

are depicted using long profile lines to achieve a higher level of abstraction. The two buildings in the environment are drawn with greater detail. Profile lines and dithering points are used for the left building and a painterly style for the right building. In the painterly rendering, thick black directional strokes are overlaid on top of the standard painterly rendering to highlight the edge features.

Finally, we demonstrate the rendering efficiency of our system using two data sets. The performances are evaluated while navigating through the scenes at 800×600 resolution. The average rendering speed of the sketchy style shown in Figure 9(c) is around 95 FPS, where the data has about 1.4 million points. As another example, the data shown in Figure 10 has about 1.3 million points. With the painterly rendering of this data, the average rendering speed achieved by our system is around 25 FPS.

7. Applications

We envision several applications for our Pointworks system. Our initial efforts have been largely focused on aiding Architectural design. Naturally, we have begun to explore the use of our system in creating stylized presentations of captured environments for artistic purposes. In addition, we see our system being useful for historical preservation of large artifacts.



(a) (b)

Figure 13: *(a) SmartPaper is used to sketch a design within the scanned and stylized environment. (b) A painterly rendering of scanned data from the excavation site of Ephesos in Turkey.*

7.1. Architectural Design

We have designed our system in response to the need for a unified Architectural design system. Presently, architects make use of several separate systems during the design of a new structure, including hand sketching, precise CAD modeling and artistic depictions of proposed designs amid existing environments. Our system provides an opportunity to integrate many formerly separate stages of design into a single system.

To aid with preliminary conceptual design, we provide a means to capture an existing environment and quickly remove any existing structures that will be destroyed to make way for new projects. Using this captured environment as a starting point, we hope that Pointworks, along with our 3D sketching tool SmartPaper [SC04] will allow designers to sketch new ideas directly into a captured environment (see Figure 13(a)). A user can use our stylized rendering in a variety of ways to facilitate design. The captured environment can be rendered in a style that matches the sketching style of the designer, creating a unified image involving both the existing environment and the new design. Alternately, the captured environment can be rendered in an abstract style to provide context to the new design without distracting a viewer from the focus of the image.

In contrast to the free-form conceptual design stage, CAD models usually consist of very precise polygonal models. Our system can import CAD models and render them within captured environments. However, rendering a polygonal model within a point cloud environment can produce disjointed images. This problem can be solved by rendering both the point models and polygonal models using NPR styles. Since NPR techniques have been widely developed for polygonal models already, our NPR techniques for point models provide a way to render both types of data in a consistent style.

7.2. Stylized Presentation

We have discovered that our stylized rendering system is able to produce aesthetically pleasing renditions of the environments we have captured. Our system could be used to automatically generate painterly renderings of important locations for use as decorations or souvenirs. A large (19" x 76") painterly rendering of a historic stone arch bridge located in Minneapolis is currently being displayed in Walter Library on the University of Minnesota campus.

7.3. Historical Preservation

We have used our scanner to capture historical sites, including the Mill City Ruins in Minneapolis and Scanners are also used to capture historical sites such as the Ephesos excavation site in Turkey. Although the lack of full accessibility and shear size of these sites made an exhaustive capture unobtainable, our multi-scan registration and NPR rendering styles can still provide pleasing renderings of these historical artifacts (see Figure 13(b)). Events such as the unexpected collapse of the “Old Man of the Mountain” rock formation in May of 2003 provide incentive to capture digital representations of historical artifacts before aging and accidents degrade their conditions.

8. Conclusions and Future Work

We have presented a framework for rendering sparsely scanned outdoor environments directly from point-based representations. By using artistic styles, we are able to reduce the effects of missing or incomplete data. Our framework produces coherent animations in a variety of styles. We

are able to offer interactive navigation of scenes using any of these styles on commodity graphics hardware.

Our system's ability to vary artistic styles allows users to have more control over scene depiction and interpretation. We believe this capability can be empowering and can find applications in various domains. We are encouraged by our early dialogues with architects regarding the use of our technique to visualize new designs in existing environments. Our ability to assign different styles to different objects has been especially well received. In related work, we have also been working on performing this stylization in a virtual reality environment where a large area tracker and head mounted display are used. Our system's performance can cope with the interactivity requirement of the VR system.

Finally, as a more long-term but rewarding task, we plan to further investigate the implications of employing artistic illustration for scene visualization. The evidence obtained for the effectiveness of our system in this regard has been encouraging. At the moment, however, we feel the methods applied in our system are based more on intuition than principle. It is our earnest intent to discover principles that can be used to guide both our system design and evaluation.

References

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 3 (February 1992), 239–256.

[BRM*02] BERNARDINI F., RUSHMEIER H., MARTIN I. M., MITTLEMAN J., TAUBIN G.: Building a digital model of Michelangelo's Florentine Pieta. *IEEE Computer Graphics and Applications 22*, 1 (2002), 59–67.

[CRL01] CORNISH D., ROWAN A., LUEBKE D.: View-dependent particles for interactive non-photorealistic rendering. In *Proceedings of Graphics Interface 2001* (2001), pp. 151–158.

[DS02] DECARLO D., SANTELLA A.: Stylization and abstraction of photographs. In *Proceedings of ACM SIGGRAPH 02* (2002), pp. 769–776.

[DVS03] DACHSBACHER C., VOGELGSANG C., STAMMINGER M.: Sequential point trees. *ACM Trans. Graph. 22*, 3 (2003), 657–662.

[GWM01] GUMHOLD S., WANG X., MCLEOD R.: Feature extraction from point clouds. In *Proceedings of 10th International Meshing Roundtable* (October 2001), pp. 293–305.

[Her98] HERTZMANN A.: Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of ACM SIGGRAPH 98* (1998), pp. 453–460.

[HTZ04] HAN F., TU Z., ZHU S.-C.: Range image segmentation by an effective jump-diffusion method. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9 (September 2004), 1138–1153.

[LMHB00] LAKE A., MARSHALL C., HARRIS M., BLACKSTEIN M.: Stylized rendering techniques for scalable real-time 3D animation. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering* (2000), pp. 13–20.

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINZTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of ACM SIGGRAPH 2000* (2000), pp. 131–144.

[MB95] MCMILLAN L., BISHOP G.: Plenoptic modeling: an image-based rendering system. In *Proceedings of ACM SIGGRAPH 95* (1995), pp. 39–46.

[Mei96] MEIER B. J.: Painterly rendering for animation. In *Proceedings of ACM SIGGRAPH 96* (1996), pp. 477–484.

[Ost01] OSTROMOUKHOV V.: A simple and efficient error-diffusion algorithm. In *Proceedings of SIGGRAPH 01* (2001), pp. 567–572.

[PKG03] PAULY M., KEISER R., GROSS M.: Multi-scale feature extraction on point-sampled models. In *Proceedings of Eurographics 03* (2003).

[PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: Surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH 00* (2000), Akeley K., (Ed.), pp. 335–342.

[RC99] RASKAR R., COHEN M.: Image precision silhouette edges. In *Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), pp. 135–140.

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: a multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 00* (2000), pp. 343–352.

[RPZ02] REN L., PFISTER H., ZWICKER M.: Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Proceedings of Eurographics 02* (2002).

[SC04] SHESH A., CHEN B.: SMARTPAPER – an interactive and user-friendly sketching system. In *Proceedings of Eurographics 2004* (August 2004), pp. 301–310.

[SPSF94] STROTHOTTE T., PREIM B., SCHUMANN A. R. J., FORSEY D. R.: How to render frames and influence people. In *Computer Graphics Forum* (1994), vol. 13(3), pp. 455–466.

[SWHS97] SALISBURY M. P., WONG M. T., HUGHES J. F., SALESIN D. H.: Orientable textures for image-based pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH 97* (1997), pp. 401–406.

[WS94] WINKENBACH G., SALESIN D. H.: Computer-generated pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH 94* (1994), pp. 91–100.

[XC04] XU H., CHEN B.: Stylized rendering of 3D scanned real world environments. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering* (2004), pp. 25–34.

[XGC04] XU H., GOSSETT N., CHEN B.: PointWorks: Abstraction and rendering of sparsely scanned outdoor environments. In *Proceedings of the 2004 Eurographics Symposium on Rendering* (June 2004), pp. 45–52.

[XNYC04] XU H., NGUYEN M. X., YUAN X., CHEN B.: Interactive silhouette rendering for point-based models. In *Proceedings of the 2004 Eurographics Symposium on Point-Based Graphics* (June 2004), pp. 13–18.