PointWorks: Abstraction and Rendering of Sparsely Scanned Outdoor Environments

Hui Xu, Nathan Gossett and Baoquan Chen

University of Minnesota Digital Technology Center at Twin Cities †

Abstract

This paper describes a system, dubbed PointWorks, for 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. The key method discussed in this paper is employing artistic drawing techniques to illustrate features of varying importance and accuracy.

We employ a point-based representation of the scanned environment and operate directly *on point-based models for abstraction and rendering. We develop a framework for producing mainly two artistic styles: painterly and profile lines. The framework first analyzes the input point models and performs a* fuzzy *classification. These points are then taken as stroke candidates during the rendering. At run time, a subset of points are selected view-dependently and strokes of various geometry and styles are placed at these points' screen positions. By varying the run-time stroke selection scheme and individual stroke rendering, various looks can be achieved. Strategies have also been employed to leverage modern graphics hardware for achieving interactive rendering of large scenes.*

Categories and Subject Descriptors (according to ACM CCS): 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 the scenes, 3D representations are first needed. Advances in laser scanning technology are making this 3D acquisition feasible for objects of ever larger scales. The aim of the methods described in this paper is to scan outdoor environments and deliver interactive walk-throughs of them. Outdoor environment scans demonstrate the following properties: (1) 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; (2) complexity – natural objects, such as trees and plants are complex in terms of their geometric shapes; (3) inaccuracy – distant objects are less accurate due to scanning hardware limitations, and plants and trees can be moved by wind during the scanning process; and (4) large data size. A conventional rendering of such a scan is shown in Figure 1, in which holes and noise are apparent.

These properties raise unprecedented challenges for existing methods, as most of them have been focused on generating a complete polygon mesh from points. With the aforementioned data properties of outdoor scans, fitting a complete polygon mesh is a daunting task. Certain heuristics or even manual controls have to be specified to smooth out noise and patch up holes, which can be a tedious and even prohibitive process. Moreover, a fundamental problem of this approach is that such polygon meshes remove all the uncertainties and holes existing in the original data. Thus, renderings of these models provide a false impression of the models' accuracy.

In this paper, we opt to generate animations of outdoor scenes using artistic illustration, or non-photorealistic rendering (NPR). Artistic illustration, unlike traditional photo-

† Email:{hxu, gossett, baoquan}@cs.umn.edu

realistic rendering, can aesthetically depict objects of different importance by using different accuracy and drawing styles [SPR*94]. 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 holes, become less noticeable. 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 in this paper is to develop algorithms to generate several non-photorealistic rendering (NPR) styles *directly* from point-based representations. During the NPR rendering, the level of abstraction is determined not only by the geometric and color features but also the scanning accuracy. In our recent work, we have experimented with generating sketchy NPR styles (mainly in a pen-and-ink style using short strokes) based on scanned point clouds [XC04]. Here we develop algorithms for generating painterly styles and more abstract styles like profile lines. Most importantly, we devise a unified framework for creating intermingled styles. Once this generalized framework has been developed, many additional styles can be created by mixing various styles in the same scene. For all these styles, smooth animation and a consistent NPR quality (e.g., stroke coherence and stroke density consistency) are guaranteed during navigation. Lastly, we have mapped our algorithms onto readily available commodity graphics hardware to leverage the latest vertex and pixel programming features to provide an interactive navigation experience.

We have developed a prototype system, dubbed PointWorks, that conducts all the necessary operations involved in preprocessing scanned data before they are sent for NPR rendering. The rest of the paper is organized as follows. After a brief discussion of the state-of-the-art of NPR (Section 2), we introduce our previous work (Section 3) as a precursor to the work of this paper. We then present in more detail two illustration styles, painterly and profile lines in Section 4. Finally, we discuss implementation details and present results (Section 5). We conclude with discussions and our plans for future work (Section 6).

2. Prior Art

Our system builds on top of existing developments in non-photorealistic rendering. Previous researchers have extracted principles employed by artists for guiding computer-based art simulation. Winkenbach and Salesin [WS94] and Salisbury et al. [SABS94] have summarized principles for generating pen-and-ink art styles. Meier [Mei96] has presented a framework for painterly rendering of 3D objects. Litwinowicz [Lit97] and Hertzman [Her98] further presented a framework for image-based painterly rendering. Although different groups of methods can generate distinctive artistic styles, these approaches can be generally classified as stroke-based

Figure 1: ***An attempted standard point-based rendering of scanned data.*** *Holes and noise due to lack of data are apparent. This scene is a detail from the scene depicted in the top image of Figure 5, which is rendered in a painterly style.*

non-photorealistic rendering [Her03] where images are constructed by discretely placing and depicting strokes on a virtual canvas.

The looks of individual strokes strongly reflect the interaction between the media, the drawing tools, and the artist creating a work. Methods have been developed to perform physics-based simulations for effects such as watercolor [CAS*97], calligraphy [Guo96], and pencil drawing [SB99]. Gooch et al. [GGSC98] also developed a lighting model used in artistic illustration.

For animated NPR, an important issue is consistency. Meier [Mei96] proposed a method that associates strokes with particles defined on the surface of objects. Since the strokes are associated with actual locations in space, they move smoothly across the screen in a consistent manner as the viewpoint shifts. Another issue for animation is maintaining a constant screen-space density of strokes. Too many or too few strokes can create a cluttered effect or fail to convey the underlying shape, respectively. In styles such as pen-and-ink drawing, stroke density also controls tone [SWHS97, WS94]. Simply associating strokes with screen-space increases particle density as objects recede into the distance. Thus, an adaptive way of changing the density of the particles according to object distance is desirable [CRL01].

Existing NPR methods assume a complete polygon model as a starting point. There has not been much work on conducting NPR rendering directly from point-based representations, and the existing work in point-based rendering strives for photorealism with high-quality and/or efficiency [PZvG00, RL00]. The main challenge in performing NPR rendering from scanned point clouds is the inherent inac-

curacy, incompleteness, and inconsistent sampling rate of the data (Figure 1). The authors of this paper have recently published a paper on generating some sketchy NPR styles (mainly pen-and-ink and stippling) based on scanned point clouds [XC04]. Independently, Pauly et al. [PKG03] have also proposed methods for extracting outline features from point models that can be used for NPR rendering. While Pauly's method favors a dense and complete point set, our work explicitly addresses the extreme non-uniformity and inaccuracy existing in real-world outdoor scans.

3. Our Previous Work

To set the stage for our discussion of new NPR style generation, we summarize our prior work of generating sketchy NPR styles [XC04]. First, outdoor environments are acquired through laser scanning using Riegl Inc's LMS-Z360 3D imaging sensor. The first line of processing is to obtain a separate point model for each individual object, such as a building, building wall or tree, by merging multiple scans from different scanning positions.

After point-based models are obtained, each point is further classified through a *fuzzy classification* as either a *directional feature point* (on an object's geometry or appearance boundaries with consistent local orientation), *non-directional feature point* (feature points without consistent local orientation), or *non-feature point* (the rest). Once this classification is achieved, points of different classification are depicted using strokes of various styles. While the directional feature points are usually drawn using line segments or textured strokes with their orientation guided by the point's direction, the non-directional points are drawn using strokes of uniform direction (pre-determined) or isotropic strokes such as circular point sprites. To illustrate an object's shading tone, a subset of the non-feature points are also depicted, using strokes similar to those of non-directional feature points. These points are selected through a conventional dithering operation. The left building in the bottom image of Figure 5 is depicted in this style.

Rendering a subset of points will not guarantee correct visibility; background objects may leak through the foreground objects. To address this issue, each image is rendered in two passes. The first pass aims to generates a visibility mask, in which points are rendered as opaque discs [PZvG00]. The second pass projects selected points, determines their visibility according to the visibility mask, and then places strokes at the visible points. The stroke style, choice of rendering points, and choice of rendering order determines the style of the image.

To address the animation consistency issue, a data structure called a *continuous resolution queue* has been used to easily control the density of points on the screen and to ensure coherence between frames. In this data structure, points of each object are randomly reordered into a linear queue. During the rendering, the projected screen area of each object determines the number of points used for the second pass rendering (stroke placement). The point set is always retrieved from the beginning of the queue. Others have used similar data structures in point-based rendering for different purpose [DVS03].

4. New NPR Rendering Styles

We adopt the same two-pass rendering pipeline as in the sketchy NPR rendering for generating new styles, mainly painterly, long profile lines, and the intermingling of multiple styles. We also leverage the point classification and the continuous resolution queue data structure. In this section, we discuss new operations in the second rendering pass for generating our new styles. Due to our use of a shared pipeline and data structures, different styles can be intermingled together in one single scene.

4.1. Painterly Rendering

One of the styles we can produce with our framework is painterly rendering. The basic strategy we use to render images in such a style is to first render coarse details as large brush strokes, and then finer details as smaller brush strokes. We make use of a fuzzy classification based on the feature degree (Section 3) to identify points representing finer and coarser details (Figure 2).

The straightforward approach to implementing the iterative process would be to use a single classification of points for each stroke size. We discovered, however, that we were able to obtain better coverage by using points with a wide variety 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 will consist 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. 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 classify all points by feature degree, but do not use any immediate threshold to differentiate between feature and non-feature points. We sort the points from low to high feature degree and 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

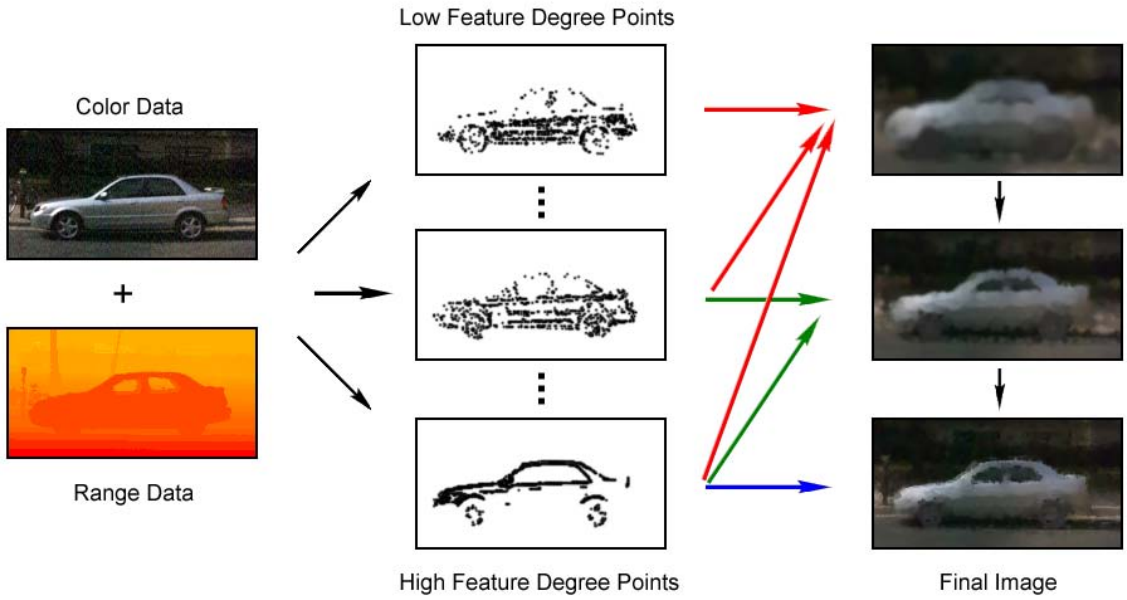

Figure 2: ***The Painterly Rendering Pipeline.*** *Scanned points are divided into bins based on feature degree. The bins are then used to produce iterations with decreasing brush size.*

points. The number of points rendered from each bin will be based on the calculated screen coverage of each object so that the density of points on the screen will remain consistent. Figure 2 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 will use more points from the bins that are left, so the queues will deliver the same points as the previous iteration, plus some additional points.

4.2. Enhancement of Painterly Rendering

Some points are used in multiple iterations, and thus will be drawn with strokes of different sizes, and will therefore illustrate different geometrical coverages. In order to maintain reasonable color transitions between strokes, we cannot simply use the color of a single point to determine the color of a stroke. Instead, for each brush stroke we locate all points in its coverage and interpolate their colors.

Our scanned data contains no points for parts of the environment that were outside the scanning range (notably the sky), so we use a simple sky box to produce an appropriate background image (a sky and generic ground color) behind the objects in the scene.

Since the first few iterations use large brush strokes, a problem arises where brush stokes extend beyond the borders of the objects they are meant to represent. In places where other, smaller brush strokes will be placed in later iterations, this is not a problem. The later brush strokes will simply cover up any out-of-bounds drawing from earlier iterations. Our randomized drawing order produces a natural painterly look. There are no points for the background, however, so any brush strokes mistakenly placed above the skyline, for instance, will not be covered up. To solve this problem, we generate a stencil mask during the first (visibility) pass of rendering. We then enable the stencil test for the first few iterations to prevent large brush strokes from covering up the background. The stencil test is then 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 3.

In many cases, placing extra emphasis on directional feature edges can aid in creating detailed structure for an image. By taking only the directional feature points from the sketchy rendering style, we can enhance the detail of a painterly style image by placing directional strokes on top of the finished painting. The color of these directional strokes can either be taken from the point colors (seen in the right image of Figure 3), or any user specified color (black directional strokes can be seen in the bottom image of Figure 5). Alternately, profile lines may be used for slightly smoother lines.

Figure 3: ***Painterly Refinement.*** *A painterly rendering without* (left) *and with* (middle) *the stencil mask. Note the improvement along the skyline. An additional iteration using very thin directional strokes is also shown* (right)*. Note the presence of edge features such as individual stones in the bridge.*

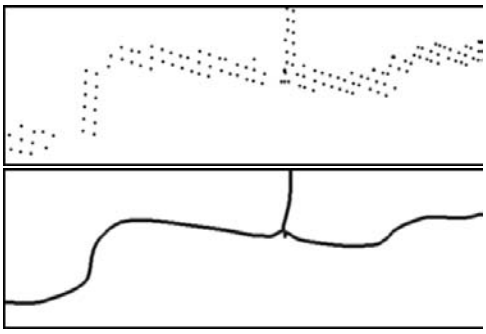

Figure 4: ***Profile Line Estimation.*** (Top) *An extreme close up of the directional points from a scan. The vertical line is the bottom of a flagpole that sits on top of a building. The points shown are all of the directional points recorded by the scanner and classified by our method.* (Bottom) *Profile lines fitted to the available points.*

4.3. Profile Lines

Images generated in the standard sketchy rendering pipeline contain many disconnected strokes. Using longer connected strokes, however, can achieve a higher degree of abstraction.

Object profiles are obtained by processing directional feature points. Each directional point D_0 is connected to all directional points in D_0's neighborhood, forming connected graphs. The 3D location of each directional point is then shifted in the direction of any point it is connected to as a preliminary smoothing step. Similar to the use of Tokens in [GCS02], any points located within a certain distance of each other are combined into a single point. Each graph is then expanded into a fully connected graph. A Minimum Spanning Tree (MST) is constructed for each graph, 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.

As a final step, B-splines are used to create a smoother appearance. 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. Branching points are not allowed to shift, thereby preserving proper connectivity. The numerous smoothing steps we apply produce satisfactory results, as seen in Figure 4. As an alternative, [PKG03] suggests the use of snakes as opposed to splines for creating smooth profile lines. The bottom image of Figure 5 demonstrates the clean appearance of profile lines for the trees, bushes and ground. 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.

4.4. Intermingled Styles

It is also possible for us to intermingle different styles within the same image. By choosing different styles for various objects, we can cause some to stand out, while others recede into the background. The bottom image of Figure 5 demonstrates this ability to visually distinguish objects within a complex scene. Our unified framework allows us to adjust styles on the fly.

5. Implementations and Results

We have implemented our NPR system using DirectX 9.0 on the nVidia GeForce FX 5800 graphics card with 128MB video memory. Our test PC has a 2.4GHz Pentium 4 processor, 1GB of main memory, and runs Windows XP.

In order to achieve interactive rendering of large point clouds, we seek a tight coupling of the CPU and GPU and leverage custom vertex shaders. The key run-time implementation of our NPR pipeline is the realization of the *continuous resolution queue* data structure. The randomized

feature point set is stored in vertex buffers residing in the video memory. The CPU is used to determine the number of strokes to be placed, based on a predefined density value and the object screen coverage area. Buildings are usually segmented so that each wall is a separate object in order to achieve accurate coverage estimates. Then, the GPU is used to retrieve the requested number of points from the vertex buffer starting from the beginning of the queue.

An additional implementation detail addresses the retrieval of point color in painterly rendering. Recall that the same point may be used in multiple iterations, and each point has multiple colors stored for brush strokes of different sizes. For storage efficiency, multiple colors are compacted together before being sent to the vertex buffer. A vertex program retrieves the right color for each point according to the current iteration.

Figure 5 demonstrates several representative artistic styles that our system is capable of generating. The top image represents a painterly style enhanced by colored directional edge strokes. The sky background is simply generated through a sky box, and is not stylized. Compared with the image of the same scene generated using photorealism (Figure 1), this artistically rendered image has gracefully masked out data deficiencies such as holes and noise that existed in the original point data.

The bottom image of Figure 5 illustrates our system's capability in adjusting different styles for different objects within a single scene. Here, user-specified non-essential objects such as trees, bushes, and ground are depicted using long profile lines. The two buildings in the environment are drawn with greater detail. Pen-and-ink and painterly styles are used for the left and right building, respectively. In the pen-and-ink rendering, directional, non-directional, and non-feature points are used for depiction. In the painterly rendering, thick black directional strokes are overlaid on top of the standard painterly rendering to highlight the edge features.

Also, as shown in the companion video, our system produces smooth animation thanks mainly to our continuous resolution queue data structure.

Lastly, we show the rendering efficiency of our system in Table 1. In Table 1(a), the performance is evaluated while navigating through the top scene shown in Figure 5. Three representative frame rates are reported. Table 1(b) reports the typical performance for the bottom scene in Figure 5 and then breaks down to the rendering of objects with different styles. This table shows that our system offers interactivity at comfortable rates when exploring large outdoor environments. This interactivity is achieved by efficient utilization of commodity graphics hardware, and our avoidance of expensive operations such as the point sorting steps used in other approaches [Mei96].

Table 1: ***Rendering performance***

(Image resolution 800×600*, 6 iterations used for painterly rendering).*

$\mathbf{N}$*: the number of points in the scene;*
$\mathbf{N}_f$*: the number of points actually rendered in the first pass (after applying view frustum culling and continuous resolution queue);*
$\mathbf{N}_p$*: the total number of points rendered for all painterly iterations in the second pass;*
$\mathbf{N}_l$*: the number of splines rendered for the profile;*
$\mathbf{N}_s$*: the number of points rendered for the sketchy object.*

	First Pass	Second Pass	
$\mathbf{N}$	$\mathbf{N}_f$	$\mathbf{N}_p$	FPS
1,294K	669K	1,103K	10
	613K	1,012K	11
	584K	975K	12

(a) The top scene in Figure 5.

	First Pass	Second Pass			
$\mathbf{N}$	$\mathbf{N}_f$	$\mathbf{N}_l$	$\mathbf{N}_s$	$\mathbf{N}_p$	FPS
1,391K	707K	104K	20K	61K	31

(b) The bottom scene in Figure 5.

6. Conclusions and Future Work

We have presented a framework for rendering sparsely scanned outdoor environments. By using NPR 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.

For outdoor scanning, obtaining an accurate and complete geometry is infeasible. As evident from our rendering, stylized rendering can gracefully mask out data deficiency. 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.

We have identified and begun several tasks to further en-

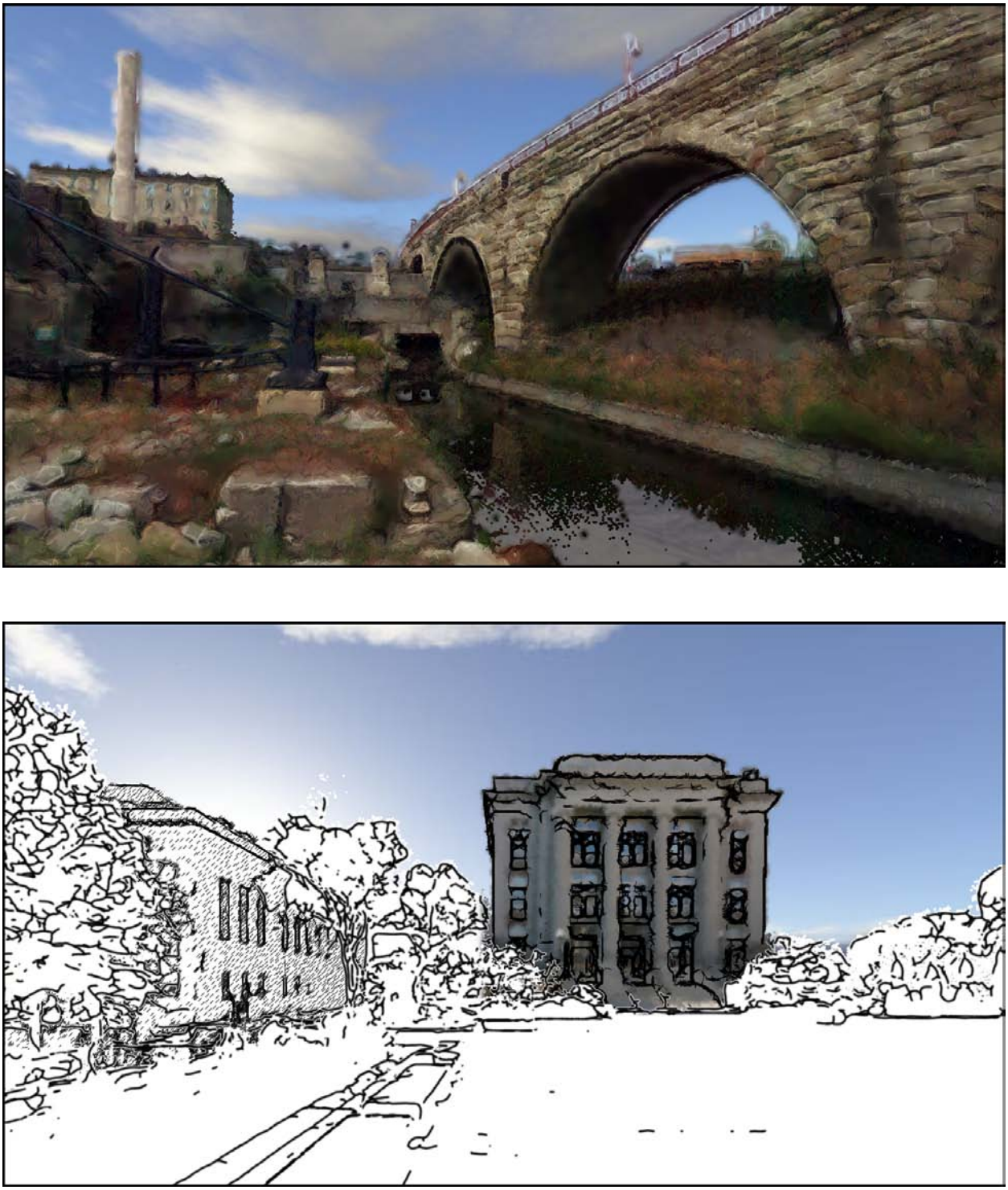

Figure 5: ***Demonstration of different NPR styles rendered by PointWorks.*** Image resolution: 1600×900. (Top) *A painterly rendering. Compared with Figure 1, this artistically rendered image is free of holes and gracefully masks out data deficiencies such as noise and insufficient sampling.* (Bottom) *An image rendered using different styles for different objects: long profile lines for trees and ground, pen-and-ink for the building at left, and painterly for the center building. In the painterly rendering, thick black directional strokes are overlaid to highlight the edge features.*

hance our system. We aim to explore additional stylizations. The established unified framework provides opportunities to produce more combined styles. Along the same line of pursuit, we strive to conduct closer simulation of various artistic media. We also plan to investigate the recovery of objects' ambient color so that relighting can be conducted by adopting the method in another of our previous works [NXYC03]. Finally, as a more long-term but rewarding task, we plan to further investigate the implications of employing artistic illustration for data 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

[BR02] BERNARDINI F., RUSHMEIER H.: The 3D model acquisition pipeline. *Computer Graphics Forum 21*, 2 (2002), 149–172.

[CAS*97] CURTIS C. J., ANDERSON S. E., SEIMS J. E., FLEISCHER K. W., SALESIN D. H.: Computer-generated watercolor. In *Proceedings of SIGGRAPH 97* (1997), pp. 421–430.

[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 SIGGRAPH 2002* (2002), pp. 769–776.

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

[GCS02] GOOCH B., COOMBE G., SHIRLEY P.: Artistic vision: painterly rendering using computer vision techniques. In *Proceedings of the 2nd Annual Symposium on Non-Photorealistic Animation and Rendering* (2002), pp. 83–90.

[GGSC98] GOOCH A., GOOCH B., SHIRLEY P., COHEN E.: A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of SIGGRAPH 98* (1998), pp. 447–452.

[Guo96] GUO Q.: Generating realistic calligraphy words. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E78A*, 11 (1996), 1556–1558.

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

[Her03] HERTZMANN A.: A survey of stroke-based rendering. *Computer Graphics and Applications, IEEE 23*, 4 (2003), 70–81.

[KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. *ACM Trans. Graph. 22*, 3 (2003), 856–861.

[Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *Proceedings of ACM SIGGRAPH 97* (1997), pp. 407–414.

[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.

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

[MKG*97] MARKOSIAN L., KOWALSKI M. A., GOLDSTEIN D., TRYCHIN S. J., HUGHES J. F., BOURDEV L. D.: Real-time nonphotorealistic rendering. In *Proceedings of ACM SIGGRAPH 97* (1997), pp. 415–420.

[NXYC03] NGUYEN M. X., XU H., YUAN X., CHEN B.: Inspire: An interactive image assisted non-photorealistic rendering system. In *Pacific Graphics* (2003), pp. 472–476.

[PKG03] PAULY M., KEISER R., GROSS M.: Multi-scale feature extraction on point-sampled surfaces. *Eurographics 2003 22*, 3 (2003).

[PZvG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH 2000* (2000), pp. 335–342.

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

[SABS94] SALISBURY M. P., ANDERSON S. E., BARZEL R., SALESIN D. H.: Interactive pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH 94* (1994), pp. 101–108.

[SB99] SOUSA M. C., BUCHANAN J. W.: Computer-generated graphite pencil rendering of 3D polygonal models. *Computer Graphics Forum 18*, 3 (1999), 195–208.

[SPR*94] STROTHOTTE T., PREIM B., RAAB A., SCHUMANN J., FORSEY D. R.: How to render frames and influence people. In *Computer Graphics Forum (13) 3* (1994), pp. 455–466.

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-D shapes. In *Proceedings of ACM SIGGRAPH 90* (1990), pp. 197–206.

[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.

[WS96] WINKENBACH G., SALESIN D. H.: Rendering parametric surfaces in pen and ink. In *Proceedings of ACM SIGGRAPH 96* (1996), pp. 469–476.

[XC04] XU H., CHEN B.: Stylized rendering of 3D scanned real world environments. In *Proceedings of the 3rd Annual Symposium on Non-Photorealistic Animation and Rendering* (2004). to appear.

[ZPKG02] ZWICKER M., PAULY M., KNOLL O., GROSS M.: Pointshop 3D: an interactive system for point-based surface editing. In *Proceedings of ACM SIGGRAPH 2002* (2002), pp. 322–329.