Abstract

We present a method to aid in the generation of pen-and-ink style renderings of complex geometry. Most illustration algorithms focus on rendering a small number of surfaces with as much detail and expression as possible. These methods break down when the scene is composed of many small, overlapping details that are not individually resolvable. We propose a hybrid 2D/3D pipeline that incorporates image processing with the full scene geometry to extract regions which may require special handling. For silhouette renderings, grayscale value is matched to a sample rendering while preserving important edges and texture. For hatched areas, the approximate surface properties of each logical region is represented. The result is an image that is more clear for informative purposes, and more artistic for aesthetic purposes.

Keywords: Image processing, non-photorealistic rendering, pen-and-ink, segmentation, silhouette

1 Introduction

As computer graphics has matured, more interest is being given to non-photorealistic techniques that attempt to emulate artistic illustration. Non-photorealistic rendering allows the generation of images that are pleasing from an artistic standpoint while also providing several concrete benefits over traditional methods: because the representation is not constrained to be realistic, a variety of enhancements can be made including making important edges more visible and using brush stroke directions to imply surface curvature.

Much research has gone into developing precise and artistically pleasing rendering methods in a variety of mediums, from pen-and-ink to watercolor. Typical applications of these algorithms are simple, well-behaved geometry of single objects such as sculptures. Many real-world scenes, however, include large amounts of complex overlapping geometry, and, as computer graphics progresses, we expect to see more of these increasingly detailed scenes. Complexity is challenging for any rendering technique due to the large amounts of resources required for modeling and processing. However, it is especially problematic for non-photorealistic techniques because artists treat complex regions very differently from simple, well-behaved surfaces. This treatment is due to a variety of factors, including limitations of the medium, the time and effort required, the desire for simple, clear illustrations, and artistic aesthetic.

The pen-and-ink style of rendering is particularly prone to over-detailling. While it has the advantage that very small details and surface properties can be very accurately and simply represented, it is difficult to represent large amounts of detail in small areas. The binary nature of the medium means that if too many lines are drawn close together, the result will be an incomprehensible black mass. Even if lines are drawn far enough apart to be well-resolved, many small lines in a limited area can be distracting and convey a limited amount of information.

Artists deal with the problem of complexity by abstracting away details; it is neither practical, nor artistically desirable, to draw every small detail in the scene. Using domain knowledge, artistic aesthetic, and knowledge of how the viewer will perceive the scene, the artist picks important details to represent exactly, and represents the rest through abstraction and suggestion. The result is an image where objects and features of interest are very clearly defined, and additional details are represented at a minimum level.

Our work attempts to deal with highly complex geometrical models in a way that gives clear, meaningful, and artistically believable renderings. The integration of a two-dimensional image processing step into the three-dimensional rendering pipeline provides more information to the renderer. We use this information to make choices about handling complex geometry, but it can also be used to provide additional stylistic control.

The two primary pen-and-ink styles currently supported are silhouette edge rendering, in which only edges are drawn, and hatching, in which small groups of parallel lines are drawn. These two methods can be combined in different amounts to achieve a range of rendering styles imparting different information and style. Non-complex areas are not directly addressed by this research. While these methods can be used for all areas of an image, it is likely better results can be achieved in simple areas using traditional pen-and-ink surface illustration algorithms.

2 Related work

There has been increasing interest in non-photorealistic rendering. Recent years have seen the development of a wide range of algorithms designed to emulate specific artistic styles and to apply artistic principles generally to achieve more meaningful images. The approaches to non-photorealistic rendering, with an emphasis on the pen-and-ink techniques used in this project, can be divided into two primary areas. First, there are geometric techniques which work from meshes or other three-dimensional primitives. Second, image-based techniques exist that take as input source renderings or photographs. Image-based techniques allow a higher-level of abstraction because they are not constrained by geometry, but because of the lack of geometry, such methods can miss important details that may not be clearly present in the image.

© ACM 2004. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in the Proceedings of NPAR 2004.
2.1 Geometry-based illustration

One of the earliest works on pen-and-ink rendering was [Winkenbach and Salesin 1994], who use pre-generated hierarchical textures on simple planar geometry. The technique was extended in [Winkenbach and Salesin 1996] to automatically generate textures based on the natural parameterization of a surface. Rendering arbitrary surface meshes has been done using principle curvature direction to guide stroke generation [Girshick et al. 2000] and two-dimensional direction fields based on surface curvature along silhouette edges [Hertzmann and Zorin 2000].

While most of the above techniques can not be run interactively, there has also been real-time pen-and-ink rendering work. [Praun et al. 2001] pre-compute a set of textures called tonal art maps for the model using a natural parameterization. For arbitrary surfaces, lapped textures [Praun et al. 2000] are used for generating a tesselating an object given an example texture. The technique uses hardware multi-texturing to blend between differing resolutions of the maps to create temporally coherent, interactive shading. A different method for rendering that achieves interactive frame rates is [Secord et al. 2002], which places primitives such as hatches and stipple according to a probability function to match tone and preserve details.

Silhouettes are important to many artistic rendering styles. [Isenberg et al. 2003] give an introduction to silhouette extraction and a comparison of several different approaches. Because brute-force silhouette edge detection is straightforward, most work has focused on interactivity or quality. [Raskar and Cohen 1999] present a fast and simple method for rendering silhouette edges in hardware. However, this method is not suited to techniques that stylize the silhouette edges using pen or brush strokes because the actual edges are never calculated. Fast software-based methods include the edge buffer [Buchanan and Sousa 2000] and a four-dimensional octree [Claes et al. 2001]. Higher-quality methods are also used, including [Sousa and Prusinkiewicz 2003], who extract important feature and silhouette lines to create minimal line drawings of objects.

This work is also related to [Deussen and Strothotte 2000], who generate pen-and-ink illustrations of trees. By rendering leaves using simple leaf-shaped primitives or circles and drawing outlines when the depth difference exceeds a given threshold, this method abstracts away unimportant detail and produces believably artistic results.

2.2 Image-based illustration

Interactive techniques allow greater artistic expression by allowing humans to make artistic choices and enhancements based on domain knowledge while freeing them from the more tedious tasks such as drawing individual strokes. [Salisbury et al. 1994] describe a system that presents an image and allows the user to "paint" texture onto areas of the image which are automatically adjusted to match the reference grayscale. In the higher-level system described by [Salisbury et al. 1997], the user edits a vector field over the image which the computer uses to align pen strokes. A somewhat different approach is taken by [Rössl and Kobbelt 2000], in which the user edits a segmentation which the system uses to guide stroke placement.

3 A General Approach for Non-Photorealistic Rendering

Artistic rendering is a balance between the two opposing concepts of detail and abstraction. Important parts of an image should drawn in as much detail as is understandable to the viewer. Less critical areas should be drawn more abstrackly to focus the viewer’s attention to the important details while providing the necessary context to interpret the entire image. Neither fully two-dimensional image-based techniques nor three-dimensional geometry-based techniques can give the full range of stylistic choices. Important edges and material properties can be lost in image processing, and purely geometric algorithms do not take into account how the viewer will perceive the scene.

Our pipeline for dealing with this challenge integrates two-dimensional image processing into a three-dimensional rendering pipeline as shown in Figure 1. During the final rendering, all ge-
ometry and segmentation information are available, combining the advantages of both image and geometric techniques.

This process uses a series of intermediate renderings on which segmentation or other image processing steps are performed. We render several variables, including grayscale, edges, surface normal, depth, and object ID (for reverse-mapping from the rendering to the geometry). These variables can, in general, be any material property, including object tags or complex texture characterizations. The idea is similar to that of [Hamel and Strothotte 1999], who also use a series of intermediate renderings to capture image information. However, Hamel and Strothotte discard the geometry once the intermediate images are created, and their goal is to characterize the surfaces to copy rendering style. In contrast, we preserve the geometry so as to have as much information as possible in the final stage, and have the goal of synthesizing style. Intermediate renderings are also used by [Deussen and Strothotte 2000] to determine which leaf edges should be rendered. This work is a generalization of Deussen’s technique, which relies on domain knowledge to achieve very high quality tree renderings. We use additional intermediate rendering modalities and more advanced two-dimensional image processing to handle areas that may need special treatment.

The intermediate image processing step can give important clues as to how a viewer will perceive the scene that can then be accounted for in the final rendering. For example, two very separate objects with different material properties may, due to specific lighting and viewing conditions, appear ambiguous in the final image. Figure 2 shows such a case, where the right faces of the cubes look the same, as do the top face of the front cube and the front face of the back cube.

The segmentation indicates this potential conflict because the very separate surfaces will likely fall into the same extracted grayscale segments. Depending on the desired style, several steps may be taken. Figure 2 shows two examples. In part (b), these similar areas of the surfaces are merged and filled with a single stroke, corresponding to a more abstract drawing style. The opposite approach would be to resolve the ambiguity by enhancing the contrast at the ambiguous boundaries as in part (c). Many other stylistic choices are also possible, even in this simple example.

The remainder of this paper discusses applications of the hybrid 2D/3D pipeline to silhouette rendering and hatching in order to deal with very small, overlapping geometry.

4 Complex silhouette rendering

Much of the information in pen-and ink illustrations is contributed by the silhouette outlines. However, the portions of an image with very high geometric complexity have large numbers of silhouette lines that can interfere with each other and make the result meaningless. There are two, sometimes conflicting goals, that we would like to fulfill. First, as much texture and important silhouette lines should be preserved as possible. Second, the approximate grayscale built up by the lines should accurately reflect a reference. The use of a reference ensures that grayscale value is consistent across the scene, both within complex regions, and in relation to non-complex regions potentially rendered with other algorithms. Grayscale value can give important visual clues about shape which are lost if its use is inconsistent. Figure 3 shows the reference grayscale rendering and all extracted silhouette edges for Tree 1.

4.1 Generating complexity maps

The first step in our method is to identify complex regions that may require special treatment. For this purpose, we use a complexity map: a two-dimensional array with values corresponding to the approximate complexity of the scene in the nearby area. Texture maps, hierarchical stroke textures, and procedural shaders are all sources of complexity in a scene. This work focuses on geometric complexity, which we measure by rendering all visible silhouette edges and blurring the result with a Gaussian kernel of a user-specified radius. Complex regions are therefore defined to be those areas where the complexity map value is above a given threshold, meaning that there are a large number of silhouette edges in the vicinity. Figure 4 shows a complexity map for Tree 1.

The size of the Gaussian kernel should reflect how close lines can be without becoming ambiguous or confusingly complex. It depends on the output resolution, the output image size, and artistic aesthetic. Small kernels should be used when a very detailed output image is required, as strokes can be very close without being merged by the blur operation. Larger kernels are appropriate for images designed to be viewed at a greater distance, reproduced at low resolution, or when less detailed, more impressionistic output is desired.

Because a blur operation is used to generate these regions, they will necessarily be somewhat indistinct. In some cases, however, it is desirable to have precise boundaries for complex areas. For example, a scene might include a tree overlapping a house, and it
is desirable to have a sharp boundary between the tree, which is complex, and the house, which consists of non-complex geometry. Therefore, the areas that are known not to be complex should be removed from the complexity map.

To identify non-complex areas we use a simple size metric. All polygons with an area above a given threshold are considered to be non-complex, since they are not likely to contribute to the perceived complexity of a region, and visible portions of these polygons are subtracted from the complexity map. Unfortunately, there are some common cases that are not handled well by this metric, and it is not appropriate for curved surfaces. Ideally, the input would be tagged to help identify such regions. For untagged input, it may be advantageous to group geometry based on location or similarity, and compute the complexity map for each of these regions separately.

4.2 Separating regions

Complex areas are separated into regions requiring different treatment during silhouette rendering. These areas are identified using a combination of the target grayscale rendering and a rendering of the silhouette edges that has been blurred. The blurred edge rendering represents the approximate grayscale value that the viewer will perceive if all edges are drawn, and uses the same user-specified kernel as the complexity map. This blurred silhouette edge rendering is identical to the complexity map in our implementation, but this is not generally the case. More general definitions of complexity are also possible that would contribute to the complexity map, and it is possible that additional candidate strokes should be considered in the silhouette rendering.

4.3 Rendering lightly-colored complex regions

Representing very complex regions using silhouette edge lines often produces very dark masses of lines, but lighting and surface properties may dictate that the area should actually be a light color as in Figure 5(a). For these areas, silhouette lines should be removed so that the correct value is represented, yet the characteristics of the geometry is preserved as much as possible. We use a depth-based method to identify important silhouette lines, and add the goal that the selected lines should be arranged such that the desired grayscale is approximated.

The importance of a silhouette line segment is measured by the difference in depth on each side of the segment. Those line segments separating objects very far apart in Z are most likely to represent the edges of large logical regions such as clumps of leaves or the outermost edges of a group of objects. Although not currently supported, manually-generated tags or spatial partitioning could also be used to identify important boundaries.

The depth difference for each candidate silhouette line segment is measured at its midpoint. This method is fast and simple, and is appropriate for the finely-tessellated polygonal models we use. We found that measuring this difference using the Z-buffer does not work well for small regions with large numbers of edges since visible surfaces may not all be well-resolved in the Z-buffer. If the model has large numbers of very long line segments in the complex regions, depth differences may vary significantly over the length of the line. To support these types of models, one could evaluate the depth difference at intervals along each line and break them into smaller lines of relatively constant depth difference.

All candidate lines in a given light region are checked in order of
importance to determine which should be included in the final rendering. If the addition of a line decreases the root mean square error between the blurred rendering of all previously selected lines and the target grayscale rendering, it is added to the selection. Since lines are checked in order, the most important lines that can contribute the necessary grayscale are always used.

In many cases, it is also desirable to ensure that the most important boundaries are drawn even if they make an area darker than is indicated in the reference rendering. Therefore, important lines should be drawn even if they do not make a good grayscale contribution. Moreover, the importance threshold should be a tunable parameter so that the user can select between renderings with high grayscale fidelity and low detail, or renderings with high detail differentiation and low grayscale fidelity.

At first, we experimented with a scaling factor that rewards the grayscale contribution of lines proportionally to their importance. Thus, a somewhat important line could have a small negative impact on grayscale fidelity and still be accepted, and a very important line could have a larger negative impact on grayscale fidelity and still be accepted. However, we found that this measure did not produce the desired results because important lines tend to appear in clumps around the outlines of groups of objects. Once several such lines are drawn close together, the error becomes so high that no additional lines can be drawn in the area regardless of importance. When the boundaries of the groups of objects are broken up, much of the advantage of having the boundary regions are lost.

The final implementation uses a hard importance threshold. All lines above the given threshold are always drawn, and then lines below the threshold are drawn to achieve the desired grayscale contribution. Therefore, our silhouette edge rendering method in complex regions reduces to a method very similar to the leaf-drawing algorithm of [Deussen and Strothotte 2000] if an all-white target image is used, meaning that all grayscale contribution is ignored. An example of the important edge detection is shown in Figure 6.

4.4 Darkly-colored complex regions

There are some complex regions that, despite having a large number of lines, are still not dark enough to represent the proper grayscale in the given reference rendering, as shown in Figure 5(b). Although these areas are usually small due to the high edge density of complex areas, the extra contrast can noticeably improve the output image. A simple way to darken these regions is to cross-hatch over the top of the silhouette edges. This technique is used by many artists, especially when a more abstract representation is desired. For other styles, however, it is desirable to avoid the regularity of cross-hatching and generate additional strokes that match the texture of the geometry.

Our method begins with the observation that areas of high edge complexity also tend to have high local depth complexity. In the case of a tree, there are many nearby leaves or portions thereof that are not visible because they are obscured by closer leaves. For silhouette edge renderings, additional lines that would otherwise be obscured can be drawn. Since we know the region is complex, individual geometry is not likely to be clear and the addition of “incorrect” lines is not likely to be distracting. Furthermore, these lines are likely to match the texture of the nearby visible lines.

The use of these additional edge lines mimics the way an artist may focus on building up value in darkly-colored complex regions rather than representing precise geometry. First, all visible silhouette edges are drawn. Then, the set of additional candidate edges are identified. These edges are those in the region that are not visible, but which are within some threshold difference from the topmost object. These lines are sorted and added according to importance and grayscale contribution in the same manner as discussed above for silhouette edges in very light areas. If additional grayscale value is still required, cross-hatches can be drawn over the top.

5 Complex region hatching

As with silhouette edge rendering, hatching is very well-suited to simple surfaces, but does not work well when there are many overlapping surfaces in small areas. In these cases, image-based techniques are often used to abstract away the complex geometry to give a simple, understandable, and artistically believable image.
Because of a high degree of complexity and the difficulty of controlling long pen strokes, artists often draw small stroke groups with similar properties with the aim of approximating local geometric behavior. An artist also keeps in mind the exact geometry being represented and is careful to respect the boundaries of important features when generating strokes, an advantage over two-dimensional image-processing techniques which can abstract away many important but subtle details.

To approximate this balance of abstraction and detail preservation used by artists, a 2\textsuperscript{nd}-dimensional image processing technique [Hamel and Strothotte 1999] is used. This technique combines multiple renderings of the scene along with image segmentation to extract regions that can be hatched with consistent pen strokes. The segmentation process extracts the high-level information about how the user will perceive each region. The use of multiple modalities helps fulfill the often opposing desire to respect important boundaries even if they are not visible. Rendering modalities currently in use are a grayscale rendering, a depth rendering and a rendering of the angle of the surface normal projected onto the screen. Any additional rendering modalities representing surface texture or other properties are also straightforward to incorporate.

5.1 Segmentation

Segmentation is used to divide the image into areas that can be drawn with a consistent pen stroke. We use two segmentation strategies depending on the modality of the input rendering. For the grayscale and the normal angle images, a simple region growing segmentation method is used. For the depth rendering, we use a region growing technique based on gradient. This gradient method allows the grouping of the large, smooth areas of the gradient rendering that often occur with surface going perpendicular to the view plane. Figure 7 shows the resulting segmentation for Tree 1.

More complex and accurate segmentation algorithms are not likely to improve the output image significantly. First, because multiple renderings are segmented and the results are combined, the importance of the success of any given segmentation is decreased. Second, the goal is only to produce small regions of similar attributes rather than emulate a person’s understanding of the scene. Unusual choices of boundaries are often covered up by, and are even an expected part of, the artistic technique.

5.2 Stroke generation

Once all stroke regions have been generated by intersecting the segmentations, each is separately filled with pen strokes. As with some artistic styles, no attempt is made to follow the surface exactly, but rather use small groups of parallel lines to represent approximate surface properties. This approach is better suited to very complex regions where many surfaces may be present and the exact representation is very complex. For rendering more simple regions, a traditional surface rendering approach would probably give a better impression of surface properties.

When generating strokes for each segmented region, the angle of the pen strokes in each region is determined based on the average angle of all pixels in the normal angle buffer. Cross hatches are then drawn in the region to build up the correct grayscale value first parallel to the average angle, then perpendicular, then in each diagonal direction. An example is shown in Figure 8.

6 Results

We divide processing into a three stage pipeline as in Figure 1. The first stage is the three-dimensional processing stage which works from the source geometry. After the user interactively picks view parameters, the system generates the set of reference renderings required for the segmentation stage and calculates all silhouette edges required for edge processing. The resulting grayscale and silhouette edge renderings are shown in Figure 9(a) and (b), respectively.

The second stage is the two-dimensional processing stage, which works only from the set of reference renderings produced in the previous stage. Here, the reference renderings are segmented and blurred, and the set of regions that will be used for groups of hatching strokes is generated. The regions that define groups of cross hatches are derived from the intersection of the segmentations of each of the modalities. This means that lines will cross geometry that has similar depth, color, and normal. However, as discussed in section 3, there exists a wide range of choices and exploiting them fully is the subject of future research. The combined segmentation is shown in Figure 9(c). Also in the second stage, the target grayscale approximation is generated by blurring the grayscale buffer (d), the complexity map is generated from a rendering of all edges (e), and those regions requiring lightening (f) or darkening (g) are generated.

The last stage is where the actual strokes are generated. One stylistic option is a pure silhouette rendering such as Figure 9(h). For very complex regions, these types of renderings can capture the approximate texture and grayscale value of a sample rendering by drawing only silhouette edges. Another option is a cross-hatched
Figure 9: The pipeline for handling complex regions, starting with the grayscale target and finishing in the two distinct rendering styles the method supports, silhouette edge renderings and hatching.

Figure 10: Left: Drawing only very important silhouette lines with no grayscale matching gives a very simple, open result. Right: Adding cross-hatching without segmentation. The result is approximately the correct grayscale, but the interior is abstract.
rendering such as 9(i) which uses the segmentation to provide a hatched version of the scene. By combining these two methods in different amounts, a range of styles can be generated, several of which are shown in Figure 10 and 11. Also, observe that Figure 12 is a blending of the information in Figure 3(b) and Figure 8.

7 Conclusion

Computer-generated images are becoming increasingly complex as they capture more of the tremendous detail present in everyday scenes. Both traditional and non-photorealistic rendering algorithms must adapt to handle this complexity. Our goal is to improve the quality of highly complex regions when drawn in silhouette and hatched pen-and-ink rendering styles, in terms of both believability and clarity. Unnecessary or incomprehensible details should be removed while preserving important details and as much texture as possible.

Our approach is to combine the two-dimensional image processing techniques and three-dimensional geometry-processing techniques in a hybrid pipeline to leverage the strengths of each. An intermediate rendering stage, combined with image processing techniques and three-dimensional geometry-processing techniques such as segmentation provides additional information to the information is combined with the original geometry to generate the final stylized rendering.

For silhouette edge renderings, complex areas are identified and important edges in those areas are drawn to match the grayscale value of a target rendering. For hatching, small blocks of the image with similar properties are extracted, allowing abstraction of low-importance details but preservation of the more important boundaries. These techniques, each with their own large range of stylistic control, can be combined to form a continuum of styles.

The generalized hybrid 2D/3D pipeline provides a large amount of information about the scene. This work focuses narrowly on generating pen-and-ink illustrations of regions with high geometric complexity, and much additional work remains. First, the line generation algorithms we use to illustrate the technique are simple and produce relatively primitive line drawings compared to works that have focused on emulating specific styles. Implementing state-of-the-art surface illustration algorithms for large geometry and image-based hatching algorithms for detailed areas should produce much more believable results.

Once more advanced illustration algorithms are in place, the challenge will be specifying the desired style for a specific rendering. The pipeline provides many additional variables in addition to purely geometrical information that characterize regions in screen space. Using these additional degrees of input freedom to define style will require the development of more advanced example-based or interactive approaches.

8 Acknowledgments

This work has been sponsored in part by the U.S. National Science Foundation under contracts ACI 9983641 (PECASE award), ACI 0325934 (ITR), and ACI 0222991. The tree models are from the xfrog package. The houseplant model used in Figure 1 is from 3dplants.com and was modeled by Drew Costigan.

References


Figure 12: Tree 1 drawn with a simplified silhouette rendering and segmentation-based hatch regions.


