Eurographics Symposium on Rendering 2003
Per Christensen and Daniel Cohen-Or (Editors)

A Stained Glass Image Filter

David Mould

Abstract

Medieval stained glass windows are a stylized artform that has not previously been thoroughly treated in the computer graphics literature. In this paper, we present an automated method for transforming an arbitrary image into a stained-glass version of that image. The key issues in designing a stained glass window are the tile boundaries and tile colors. We use erosion and dilation operators to manipulate and smooth an initial region segmentation tiling; we choose tile colors from the palette of heraldic tinctures; and finally, we render a displacement-mapped plane to obtain our final image.

1. Introduction

Practitioners of computer graphics have long been interested in alternatives to photorealism. Nonphotorealistic styles such as oil painting [12], pen-and-ink illustration [19], copperplate engraving [17], and many others have been automated. In recent years, there has been considerable attention put to mosaics [9, 10, 7]; yet comparatively little attention has gone to the historical successor to the mosaic, the stained glass window.

The earliest known stained glass windows date from the seventh century [16] and while the craft had its heyday in the fourteenth century [1, 16] stained glass windows continue to be built today. Although glassmaking technology has changed a great deal since the middle ages, the procedures for designing windows have changed little; Osborne [16] remarks that the instructions of Theophilus, written in the eleventh century, would seem reasonable to a modern artist. One notable difference lies in the characteristics of the glass: although modern methods have greatly extended the range of dyes for staining glass, the palette of colors available to the medieval worker was quite limited [1, 16, 8]. Further, even when the desired color might in principle be available, variations in raw materials and manufacturing processes could alter the final color.

The process for building a stained glass window involves first designing a composition, or *cartoon*, indicating the arrangement of tiles. The cartooned shapes are cut out of colored glass, assembled, and fixed in place with lead solder, or *leading*. Craftsmen preferred to minimize the heavy opaque lines of the leading, and this meant avoiding using many small pieces. However, medieval glassmaking technology did not permit the manufacture of large flat pieces, and so the practical size of tiles was limited.

Economic factors also affected window design. Medieval glass was extremely costly, and if in assembling a window a tile should break, the glass could not be replaced. Rather, the broken pieces were cut again and leaded into place, potentially disrupting the initial design. Designers would avoid calling for shapes likely to break.

In the following, we propose an image filter for stained glass windows. We give algorithms for constructing a cartoon from an initial segmentation, and we suggest a palette akin to the limited palette available in medieval times. Lastly, we show examples of the application of our method to some test images.

2. Previous Work

Many extant techniques model other nonphotorealistic modes, but stained glass windows have been little treated. Nonphotorealistic techniques frequently utilize specialized methods not applicable to other artistic styles; in the following, we confine our discussion to describing the past work with most direct bearing on ours.

A technique based on Voronoi regions [15] has been used in PhotoShop [14]. The simplicity of this method makes it attractive, but because the Voronoi regions are placed with no regard for image content, the tiling has little to do with the underlying image. Also, the Voronoi tiles themselves are not fully appropriate, since stained glass windows invariably contain some nonconvex tiles. In the PhotoShop filter, the original image begins to reemerge as the tile size becomes

small, but in this case the filtered image resembles a mosaic much more than it does a stained glass window.

The work on mosaics [9, 10, 7] resembles ours most closely. However, stained glass windows differ from mosaics in crucial ways. Mosaic tile shapes are predefined, and mosaic tiles are very small relative to the overall picture. Neither of these two conditions holds for stained glass windows. Stained glass windows have two concerns: to align tile edges with image edges, and to form tiles which may be straightforwardly cut from glass. The first of these concerns is shared by the mosaicists, but the second is not.

Elber and Wolberg [7] place mosaic tiles along feature curves and their offset curves; these authors' concerns about the self-intersection of offset curves we experience as a desire to subdivide narrow tiles. While they trace offset curves with tiles, we place new leading lines across narrow bottlenecks, employing similar mathematical methods but achieving distinctly different results.

The image simplification technique presented by DeCarlo and Santella [6] is also related to our methods. These authors share our concerns of seeking to emphasize region boundaries with lines and to segment the original image into large homogeneous regions. However, their goal is to produce a new modality and they do not attempt to conform to the requirements imposed by stained glass tiling: in particular, that regions be approximately convex, and that heavy lines denote the borders of each region. These requirements heavily inform our algorithms.

We rely on existing operators of mathematical morphology, which we adapt slightly to this context. Our work is also informed by scholarship on stained glass windows, especially that of Armitage [1] and Osborne [16].

3. Algorithms

In constructing the stained glass window, we process the initial image through several stages. First, we obtain an initial segmentation of the image. Next, we massage this segmentation to obtain an appropriate tiling: one having smooth boundaries and approximately convex pieces, and lacking excessively large or excessively small pieces. We next choose a color for each tile. Finally, we apply a displacement map to a plane, representing the leading and irregularities in the glass, and render the result.

The initial segmentation is given by the image processing system EDISON [4, 5, 11]. Other automated techniques could be used instead; another alternative would involve a human assisting the segmentation with a tool such as intelligent scissors [13]. In this section, we describe the algorithms used to transform this initial segmentation into the final stained glass image.

3.1. Mathematical Morphology

We employ the erosion and dilation operators from mathematical morphology [18, 2] to perform region smoothing. Both alter an initial image by use of a *structuring element*, itself a two-dimensional image. For a binary image I, and a structuring element S with radius r, the image M modified by erosion has pixels given by

$$M_{i,j} = 0 \text{ if } \exists u,v \in \{-r,r\} \text{ st } (S_{u,v} = 1 \,\&I_{i+u,j+v} = 0) \quad (1)$$
$$= 1 \text{ otherwise.} \quad (2)$$

Dilation is defined similarly:

$$M_{i,j} = 1 \text{ if } \exists u,v \in \{-r,r\} \text{ st } (S_{u,v} = 1 \& I_{i+u,j+v} = 1) \quad (3)$$
$$= 0 \text{ otherwise.} \quad (4)$$

Informally, the erosion operator involves running an eraser around the rim of the initial region, and the dilation operator involves running a brush around the rim. The shape of the eraser or brush is given by the structuring element.

We amend the erosion operator to permit erosion of multiple regions simultaneously. We use the special code E to mark eroded areas:

$$M_{i,j} = E \text{ if } \exists u,v \in \{-r,r\} \text{ st} \quad (5)$$
$$(S_{u,v} = 1 \& I_{i+u,j+v} \neq I_{i,j}) \quad (6)$$
$$= I_{i,j} \text{ otherwise.} \quad (7)$$

Dilation operates similarly, but simultaneous dilation of multiple regions demands a distance metric, to be encoded in the structuring element: the pixels $M_{i,j}$ take the value of the nearest non-eroded region. The notion of simultaneous dilation is critical to our later algorithms.

Owing to the use of a distance metric, simultaneous dilation has close links to Voronoi regions. Formally, the regions generated by our simultaneous dilation are equivalent to regions generated by computing the Voronoi regions of all non-eroded pixels, and taking the unions of all of those Voronoi regions which originated from the same connected component of the image.

3.2. Cartoon

The *cartoon* is the design drawing for the window composition. We abuse terminology slightly and use "cartoon" to refer to the planned arrangement of tiles.

Our algorithms process the initial segmentation to produce a related tiling whose tiles have the desired characteristics, using our previously defined erosion and dilation operators.

We have a number of criteria for our tiles: they should have piecewise smooth edges; they should be approximately convex; and they should be neither excessively large nor excessively small. In the following, we show how each of these criteria is met.

Figure 1: *Region smoothing. Original region boundaries, left; the eroded region boundaries, middle; and the completely opened regions, right.*

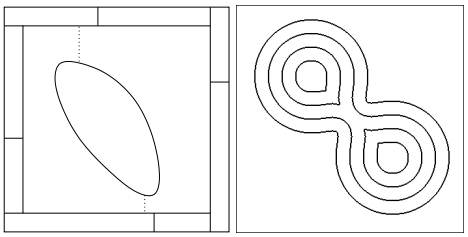

Figure 2: *Undesirable region shapes. Left, an island and a lake, and a possible lake subdivision; right, a bottleneck and eroded versions thereof.*

3.2.1. Region Smoothing

We meet the first criterion by simultaneously opening all regions (erosion followed by dilation). The erosion step marks all pixels which are too near any boundary pixel; this creates wide boundaries with smooth edges. The dilation step reclaims the eroded territory, classifying each previously eroded pixel with the nearest region. In our reported results, our structuring element encodes a Euclidean distance metric; alternatives such as Manhattan distance are also possible. We prefer the Euclidean distance because it produces smooth edges, but if piecewise linear edges are instead preferred, the Manhattan distance should be used. Fig. 1 shows the progression of the image regions: first the initial regions, then the eroded regions, and finally the regions after the full opening operation has occurred.

3.2.2. Islands

A rare type of undesirable region is the "island", a region which does not touch the image boundary and which shares a border with exactly one other region. The surrounding region we call the "lake". We do not want to eliminate the island entirely, because it can contain important semantic information – it may represent an eye, or a window – but medieval glassmakers never attempted to construct lake-type tiles. See Fig. 2 for an example of a lake-island pair.

We eliminate islands and lakes by subdividing the lake. We choose a random direction; the extremal points of the island (maximum and minimum) along this direction form the seeds of two new regions. We simultaneously dilate both seeds into the lake, replacing it with two new regions. This process removes the undesirable case while preserving previously existing boundaries.

3.2.3. Bottlenecks

Having performed the initial region opening and checked for islands, we are left with tiles having smooth boundaries but potentially undesirable shapes. We seek to eliminate regions consisting of two subregions linked by a narrower bottleneck; an example of such a region is sketched in Fig. 2. Not only are such regions aesthetically suspect, they are technically awkward: the bottleneck is a weak point and the tile is apt to break. either as it is being cut out or as it is being put into place. Regions free of such bottlenecks we refer to as "approximately convex"; we demand that all our tiles be approximately convex.

Our method for detecting and subdividing such shapes operates as follows. We test a region by progressively eroding its boundaries and checking whether the eroded region has multiple connected components [18]. If at some stage of erosion multiple components appear, we label the components separately and simultaneously dilate them back into the eroded territory. This process automatically finds a smooth new boundary situated within the bottleneck.

3.2.4. Region Sizes

Lastly, we want to control the minimum and maximum sizes for tiles. The window designer will try to avoid including very small tiles in the design, because such tiles will be dominated by the surrounding leading. Note, however, that small tiles may appear in the final window, owing to accidental breakage of larger tiles as the window is being assembled. Very large tiles are to be avoided as well, for the simple reason that medieval manufacturing processes could not produce large flat pieces of glass. We eliminate small regions very straightforwardly – marking them as eroded and dilating their neighbours into the eroded area. The subdivision of large regions requires more elaborate methods.

If a region's size exceeds the maximum allowed, we proceed as follows. First, we perform progressive erosion and connected component counting to determine whether a good subdivision location exists; if so, we use this subdivision. If not, we choose a random direction and mark the region's extremal points in this direction as the seeds of two new regions; we dilate these points into the large region, and the classification thus obtained forms our subdivision. If any newly formed subregions are too large, we recursively subdivide them. Finally, we perform the entire battery of region-tidying processes on the set of regions arising from the original large regions; although doing so is largely redundant, it can produce some improvement in the final set of tiles, ensuring that no malformed tiles were produced in the subdivision process.

We define region sizes by pixel count. In this paper, we have used a minimum region size of 500 pixels, and a maximum region size of 10,000. Note that in practice few regions near the minimum size appear, since small regions are susceptible to destruction by the boundary smoothing process.

3.2.5. Supervised Backgrounds

Our method is intended to work on any image, with no semantic information provided, and consequently we have no means of distinguishing which elements of the image are important and which are not. Real medieval stained glass windows, however, often distinguish between foreground and background elements and tile them differently.

We therefore also provide a "supervised" mode, where the user may mark regions as foreground or background; the background regions are then tiled with geometric shapes, or fragments thereof, enhancing contrast with the foreground. Of course, the system may run instead in unsupervised mode, where no differentiation among regions is made.

We have implemented two simple styles of background tiling: a regular diamond pattern and an irregular checkered pattern. The latter we refer to as *ladders*. Later, we show cartoons with supervised backgrounds tiled both ways.

3.2.6. Cartoon Summary

In sum, we perform the following sequence of operations to obtain our cartoon. We begin with an initial segmentation provided by EDISON [4, 5, 11]. We smooth these segments with an opening operation. Next, we subdivide all lakes and bottlenecks. We next modify all regions with inappropriate sizes, eliminating those which are too small and subdividing those which are too large. Finally, we redo our opening and bottleneck removal operations to ensure that all newly constructed regions are free of defects. With the cartoon complete, we are ready to proceed to coloring and rendering.

3.3. Colored Glass

Our chief remaining task is to color the tiles. As previously mentioned, the colors of glass available in medieval times were restricted, and as our goal is to mimic medieval stained glass, we will employ a limited palette ourselves. The bold colors of medieval stained glass favor stylization over representation. In turn, we seek a stylized palette evocative of the medieval palette.

Armitage [1] makes a connection between stained glass and heraldry, two art forms with similar aims of clarity and stylization. Also, Osborne's list of medieval colors [16] corresponds closely to the set of heraldic tinctures. We therefore propose adopting the heraldic palette for our stained glass windows. It is broad enough to cover the historically available colors, while being an authentic medieval colorset.

The heraldic tinctures are a set of seven colors [1, 3], or more properly five colors and two metals. (We disregard the furs, which are patterns placed over the colors.) The metals are *or*, gold, or yellow; and *argent*, silver, or white. The colors are *gules*, red; *azure*, blue; *vert*, green; *purpure*, purple; and *sable*, black. Table 3.3 lists the tinctures, the corresponding colors, and indicates the historical coloring agent. We have chosen RGB triples representing each heraldic color. Then, for a given tile, we determine its average color in the original image and the distance of this color from each heraldic color; the tile is colored with the nearest heraldic color.

Tincture	color	coloring agent
Or	Yellow (gold)	ferric oxide, uranium
Argent	White (silver)	none (clear glass), tin oxide (opaque)
Gules	Red	copper, silenium, gold
Azure	Blue	cobalt
Vert	Green	copper, chromium
Purpure	Purple	nickel (in potash lead glass)
Sable	Black	N/A

Table 1: *The heraldic tinctures.*

We make two modifications to this basic algorithm. First, we shift the tile color slightly in a random direction, to represent variation in the glassmaking. Second, since black is not a desirable color in stained glass, we map sable to an RGB triple close to white.

The above color-choosing algorithm is used for foreground tiles. If the user has indicated a background, we use the heraldic color least prevalent in the foreground for the background tiles.

3.4. Rendering

Given the final cartoon, we compute a displacement map which consists of large displacements near tile boundaries, representing the leading, and a small-valued irregular displacement map over the rest, representing imperfections in the glass surface. The leading is given a metallic color and the glass tiles their previously chosen heraldic color; the whole is then rendered with a single close light, and the result is our final stained glass image.

4. Results

Here we show a collection of images generated by our methods. We also show the stages the images go through in the course of processing.

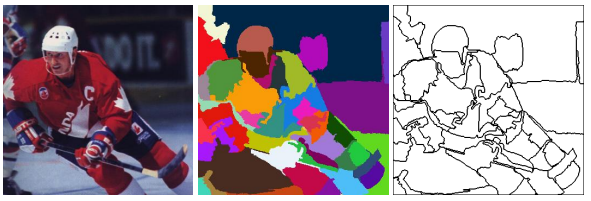

Figure 3: *The original Gretzky image, left; the segmentation, centre; and the region boundaries, right.*

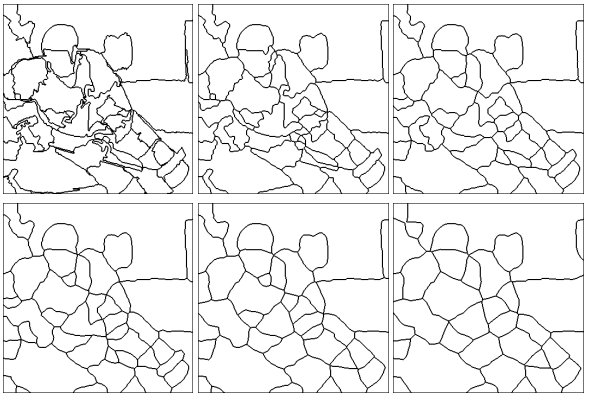

Figure 4: *The original image, then openings with erosion masks of radius 4, 6, 8, 10, 12.*

Fig. 3 contains the original Gretzky image and the initial segmentation. The segmentation is eroded by a circular structuring element. Results arising from different sizes of structuring element are shown in Fig. 4; the larger elements perform more smoothing, but remove more detail, and some balance between these must be sought. We suggest a mask size of 17×17, although the user may choose a different size, or even modulate the mask across the entire image.

Once the initial regions have been smoothed, we process the tiles to remove undesirable shapes and sizes of tiles. The final tilings, under both supervised and unsupervised conditions, are shown in Fig. 5. Next, having completely processed the tiles, we create a planar surface – the window – and apply a displacement map representing imperfections in the glass. Leading is applied between tile boundaries, and the resulting object is rendered. Final results appear in Fig. 6.

Some image pairs are shown in Fig. 7, both supervised and unsupervised. The aquarium image shows results from separately marking two backgrounds.

The success of our method depends crucially on the initial segmentation; images which are difficult to segment, such as the mandrill, produce incoherent stained glass images. Also, shading information is lost, so that the shapes of objects such as the peppers become difficult to discern. It is when the

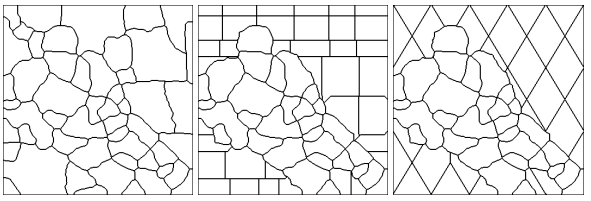

Figure 5: *Final tilings with different background conditions: left, unsupervised; middle, supervised with ladders; right, supervised with diamonds.*

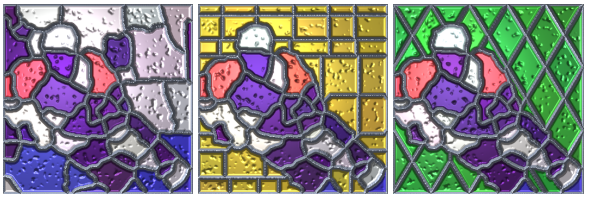

Figure 6: *Stained glass windows from the Gretzky image.*

figures in the image can be adequately characterized by their silhouettes that our technique is most successful.

5. Conclusions

We have presented an automated method for transforming a given image into a stained glass version of the image. We have set out certain rules to which our tiles must conform, and given algorithms which produce a tiling satisfying the conditions; and we have proposed a set of colors, the heraldic tinctures, which both evoke the character of medieval stained glass and which approximate the palette of colors available to the medieval glassworker.

The windows produced by our method are unpainted windows – that is, the colored glass separated by leading – and we rely on the shapes of the tiles to convey the sense of the image. Thus, not every original image is suitable for reproduction in stained glass; while we can transform any input image, images with a lot of high-frequency detail may be difficult to recognize. To a great extent this can be ameliorated by supervision, in the case that a distinct foreground figure should be emphasized. Nonetheless, simply by the nature of the medium, images can not always be faithfully reproduced.

Reproduction is not our aim, however. Stained glass inherently demands stylization of its subjects, and not all images present clear figures to be stylized. When such figures are available, the results are striking.

5.1. Future Work

We have presented an algorithm for the unadorned window, but real stained glass windows were often painted. Further,

Figure 7: *Stained glass image pairs.*

complexity was sometimes added to the tiles through etching and nonuniform staining of tiles, a practice which first became popular in the 14th century [1]. Some attention could be put to these aspects of the window.

A more complex modeling of the glass could be undertaken. In practice, the window designer chose the desired piece of glass from his collection, often preferring particular pieces for their texture. We have given only a cursory treatment to the nonuniform properties of the glass, and much more work could be done in this area.

Finally, we have cast our problem in terms of an image filter, mapping a plane image to a window. An opportunity for future work lies in building a 3D renderer with stained-glass-style output; with a geometric approach, we can in principle produce coherent animations rather than single images.

References

1. ARMITAGE, E. L. *Stained Glass: History, Technology, and Practice*. Charles T. Branford Company, Newton, 1959.
2. ARTHUR R. WEEKS, J. *Fundamentals of Electronic Image Processing*. SPIE Optical Engineering Press, Bellingham, 1996.
3. CHILD, H. *Heraldic Design*. G. Bell and Sons, London, 1965.
4. CHRISTOUDIAS, C., GEORGESCU, B., AND MEER, P. Synergism in low-level vision. *Internat'l Conference on Pattern Recognition 4*, 16 (August 2002), 150–155.
5. COMANICU, D., AND MEER, P. Mean shift: a robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Machine Intell. 24*, 4 (May 2002), 603–619.
6. DECARLO, D., AND SANTELLA, A. Stylization and abstraction of photographs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 769–776.
7. ELBER, G., AND WOLBERG, G. Rendering traditional mosaics. *The Visual Computer 19*, 1 (2003), 67–78.
8. GRODECKI, L., AND BRISAC, C. *Gothic Stained Glass*. Thames and Hudson, London, 1985.
9. HAUSNER, A. Simulating decorative mosaics. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 573–580.
10. KIM, J., AND PELLACINI, F. Jigsaw image mosaics. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 657–664.
11. MEER, P., AND GEORGESCU, B. Edge detection with embedded confidence. *IEEE Trans. Pattern Anal. Machine Intell. 23*, 12 (December 2001), 1351–1365.
12. MEIER, B. J. Painterly rendering for animation. In *SIGGRAPH 96 Conference Proceedings* (1996), ACM Press, pp. 477–484.
13. MORTENSEN, E., AND BARRETT, W. Intelligent scissors for image composition. *Proceedings of SIGGRAPH 95* (August 1995), 191–198. ISBN 0-201-84776-0. Held in Los Angeles, California.
14. O'QUINN, D. *Photoshop 6 Shop Manual*. New Rides Publishing, Indianapolis, 2001.
15. O'ROURKE, J. *Computational Geometry in C*. Cambridge University Press, Cambridge, 1994.
16. OSBORNE, J. *Stained Glass in England*. Alan Sutton Publishing, Phoenix Mill, 1997.
17. OSTROMOUKHOV, V. Digital facial engraving. *Proceedings of SIGGRAPH 99* (August 1999), 417–424. ISBN 0-20148-560-5. Held in Los Angeles, California.
18. SHAPIRO, L., AND STOCKMAN, G. *Computer Vision*. Prentice-Hall, Upper Saddle River, 2001.
19. WINKENBACH, G., AND SALESIN, D. Computer-generated pen-and-ink illustration. In *Computer Graphics (SIGGRAPH '94 Proceedings)* (Aug. 1994), vol. 28, pp. 163–170.

Figure 8: *Stained glass image pairs. In the landscape image only, we mapped sable to green.*

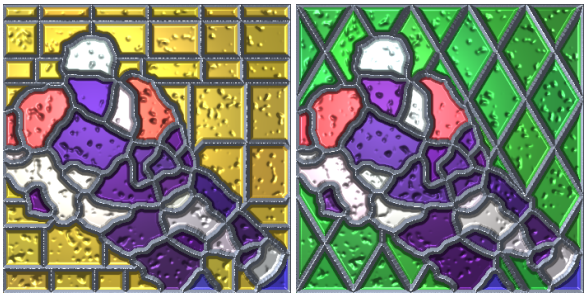

Figure 9: *The Gretzky image with supervised backgrounds.*