Automatic Stained Glass Rendering

Vidya Setlur and Stephen Wilkinson

Nokia Research Center

Abstract. Based on artistic techniques for the creation of stained glass, we introduce a method to automatically create images in stained glass stylization of images. Our algorithm first applies segmentation, and performs region simplification to merge and simplify the segments. The system then queries a database of glass swatch images and computes an optimal matching subset based on color and texture metrics. These swatches are then mapped onto the original image and 3D rendering effects including normal mapping, translucency, lead came and refraction are applied to generate the stained glass output.

1 Introduction

Stained glass is an extraordinary and vibrant medium for creating significant art and expression using a balance of light source, color and visual perception. This ancient art essentially includes tracing a pattern onto the glass and the glass is scored and cut. The artist then grinds the pieces to fit, foils each piece and reassembles, soldering the units together with lead came (Lead came is made from lead with small amounts of other metals such as copper, tin, antimony and bismuth) [1].

Gothic or medieval form of stained glass art involves the assembly of a complex mosaic of bits of colored glass joined with lead into an intricate pattern. Medieval craftmen were more interested in illustrating and idea rather than creating more natural looking art. Hence, when medieval stained glass is viewed, it appears not as a picture, but often as a network of black lines and colored light [2].

Modern or Tiffany stained glass on the other hand, involves a controlled level of artistic abstraction. Large homogenous regions comprise each glass segment while minute details are ignored in the final creation of the art piece. The effect of such abstraction allows one to appreciate the gross form and shape of the object without being distracted by superfluous features [3]. Our algorithm is largely motivated by modern stained glass as a tool for artistic abstraction.

Abstraction is often used by artists in visual art for representing objects to be more identifiable, that is, to some degree, stress the essential rather than the particular [4]. Artists rarely try to faithfully reproduce images, but even when this is the goal, it is impossible for them to achieve it, because of the intrinsic reproductive limits of the medium. However, visual abstraction is not always imposed by the medium. It may be a deliberate choice of the artist. Stained glass artists typically use one glass sample for each large homogenous region to minimize labor involved in glass cutting and applying lead came, but also to make the art piece more distinctive and identifiable.

The contribution of this paper is the automatic creation of modern stained glass stylization effects for images (Figure 1). This includes optimizing the segmentation to minimize the number of fragmented segments, mapping glass image swatches onto each

H.-P. Seidel, T. Nishita, and Q. Peng (Eds.): CGI 2006, LNCS 4035, pp. 682–691, 2006.

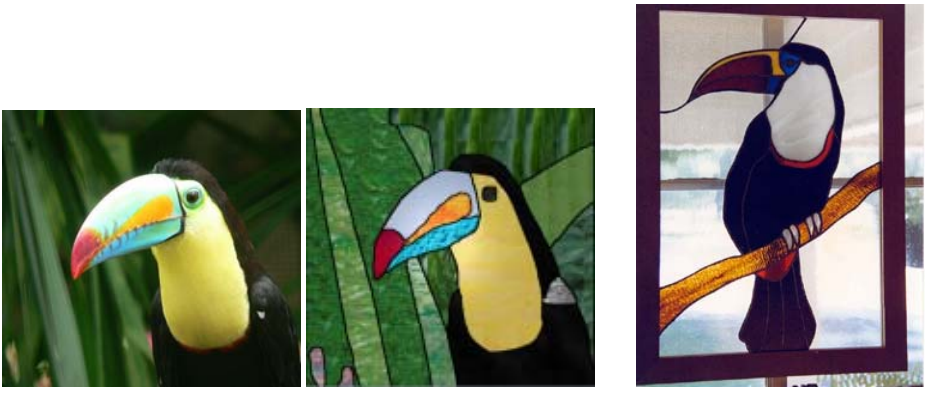


Fig. 1. Automatic stained glass rendering of an image. From left to right: The source image. The stained glass result generated by our system. An actual stained glass artpiece.

segment based on color and texture properties, applying $3D$ surface properties such as normal maps and lead came and using hardware accelerated graphics techniques for real time rendering of the stained glass results.

2 Related Work

Like many non-photorealistic techniques, our work also draws inspiration from the concept of stylized abstraction of images, preserving the form and shape and minimizing high level detail [5, 6, 7, 8, 9, 10]. Our algorithm represents another flavor of stylized image abstraction by applying stained glass techniques as a form of expressive rendering.

Though there exist commercial software like Adobe Photoshop [11], Corel Paint [12], Glass Eye 2000 [13] that allow users to apply stained glass filters to images, the process often requires the user to manually split the image into segments and select the degree of translucency/color and texture for each region and apply leading. This is typically manually intensive for large sets of images.

On a similar tangent, there has been some work on creating mosaic patterns in images. While mosaic and stained glass do share some similarities based on segmentation of the picture, the biggest difference between stained glass and mosaics has to do with how they use light, and how they are assembled. Stained glass is designed so that light passes through it, while mosaics are generally used to adorn a surface such as a floor or wall. Also, mosaics are created by placing small segments or tiles rather than larger homogenous regions in the case of stained glass [14]. Kim and Pellacini introduce a new kind of mosaic, called Jigsaw Image Mosaic (JIM), where image tiles of arbitrary shapes are used to compose the final picture as compactly as possible [15]. Hausner presents a method for simulating decorative tile mosaics by using centroidal Voronoi diagrams to arrange points in regular hexagonal grids [16].

Our paper is motivated by the modern looking Tiffany glass style, while previous non-photorealistic work on stained glass [17], tends to emphasize a more medieval-style . The novelty in our approach is a glass swatch selection process, based on color

texture matching rather than cartoonization. Glass image swatches enhance the amount of interesting variations in color and texture features in the result.

3 The Stained Glass Rendering Process

Figure 2 summarizes the algorithm. We first segment the source image into regions. In Section 3.1 we discuss the techniques involved in segmenting the image, and combining adjacent regions based on their spatial distribution of color/intensity. We then retrieve glass image swatches from a database for each segment in the image based on color and texture features. These swatches are then enlarged using texture synthesis and replace the original image segments. These steps are described in Section 3.2. The next step involves creating normal maps to highlight the textural variations in the image segments and leading came between segments (Section 3.3). Finally, this image is rendered in real time with effects such as lighting, refraction, translucency as described in Section 3.4.

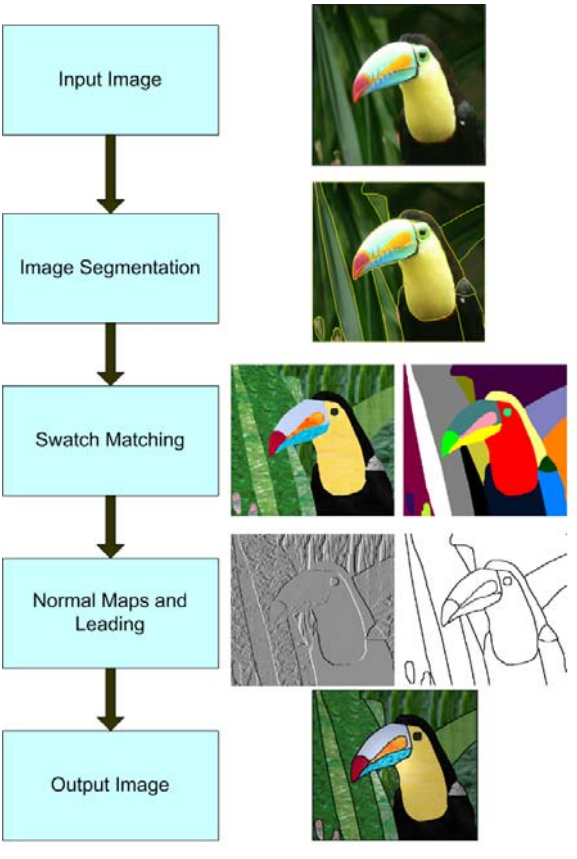


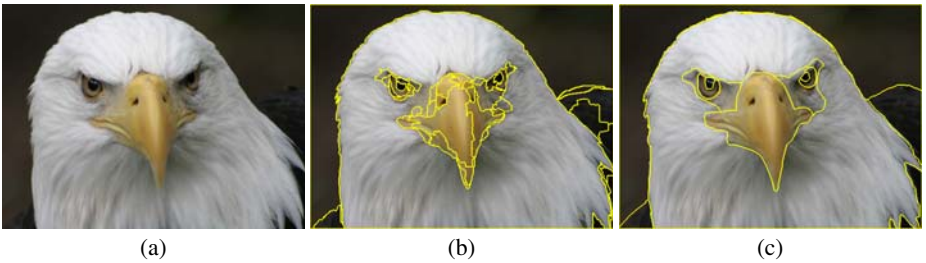
Fig. 2. Flowchart of the stained glass rendering process. The algorithm minimizes fragmented and small segments, matches glass swatches to each segment to create different stained glass patterns.

3.1 Image Segmentation

In order to create glass fragments in the stained glass rendering of the image, we must first segment the image. We apply mean-shift image segmentation [18] to decompose the given image into homogeneous regions. The advantages of this approach include flexible modeling of the image and noise processes, consequent robustness in segmentation and simplicity of the algorithm.

The segmentation routine takes as input, the parameters: spatial radius h_s, color radius h_r, and the minimum number of pixels M that constitute a region. As with other segmentation methods, choosing optimal parameter values is often difficult. Therefore we over-segment the image using lower values of h_r and M and merge adjacent regions based on color and intensity distributions in a perceptually uniform color space, CIE-Luv. In practice, values of $h_s = 7, h_r = 6$, and $M = 150$, tend to work well for over-segmentation for most images.

Our system then performs region simplification by merging similarly colored adjacent regions of the segmented image and then smoothening the outlines of the regions. We first apply a color distance measure called histogram intersection [20] to determine color similarity between regions. Often the segment boundaries are too noisy, resulting in a high degree of jagginess that is not typical of modern stained glass. If the segment outlines are considered to be Bezier curves, reducing the number of control points in the outlines creates a more optimal collection of smoother curves. The second part of the region simplification involves the use a vertex reduction technique, which is a simple and fast O(n) algorithm [21]. Control points with minimum separation are simplified iteratively until a given tolerance value is met. In practice, we have found that a tolerance value of 0.25 tends to work well with most images used in our system. Figure 3 illustrates an example of this technique.



(a) (b) (c)

Fig. 3. Image segmentation. a) The original image b) Applying mean-shift with parameters $h_s = 7$, $h_r = 6$, and $M = 150$. d) Performing region simplification on (b).

3.2 Swatch Selection

After applying segmentation, we query an image database of glass swatches to map a subset of swatches to each segment of the image. The collection of swatches comprise of 2489 color images from an online image stock database [22]. As the swatches are 64×64 in dimension, and the image segments onto which they are composited are often larger (target images are typically 640×480), we apply texture synthesis using

graphcuts to enlarge these swatches [23]. Most of the swatches tend to be homogenous, and hence the resulting larger texture synthesized swatches have minimal visual artifacts.

The method of query we use is a combination of the image segments' color and texture information. The color property indicates the significant colors that contribute to the appearance of an image segment, while the texture property refers to the spatial property of the segment. Swatch selections using only color features often do not match high frequency information in the segments. For example, texture features extracted from a scene of a field of grass can be distinguished from that of trees, whereby color alone may not provide sufficient determination. Combining both color and texture creates a more effective characterization of the image content, and is commonly used in image retrieval [24]. The goal of our work is to also utilize existing colored glass swatch images, attempting to generate results consistent with stained glass imagery. Figure 4b illustrates an example when our system applies only color based swatch selection. Although the dominant colors of the swatches chosen are consistent with the colors in the respective image segment, high frequency details may not be accurately represented. However, using a combination of color and texture tends to retain this high frequency information, thus serving a dual purpose of a more accurate abstraction, and interesting aesthetics.

The swatch retrieval process utilizes histogram techniques to compute color distances and employs Gabor filters as a channel energy model to compute image texture. Color matching is done again using histogram intersection. Although there are several algorithms to compute Gabor filters, our method is based on that of Manjunath et al [25].

Filtering an image $I(x,y)$ with Gabor filters g_{mn} results in its Gabor wavelet transform W_{mn} to be:

$$W_{mn}(x,y) = \int I(x_1,y_1)g^*{}_{mn}(x-x_1,y-y_1)dx_1dy_1$$

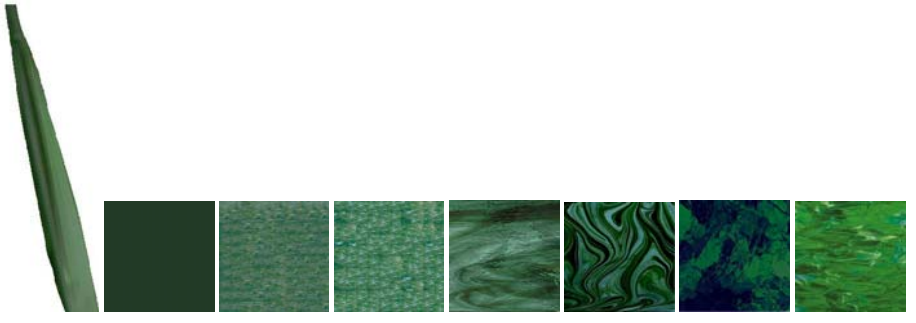
The mean and standard deviation of the magnitude $|W_{mn}|$ are used for the feature vector. Given an input query of color and textural features of a segment in the image, the swatches in the database are firstly ranked using color features. Then the top ranked images are re-ranked according to their texture features. We first select the top 5 best matched swatches based on histogram intersection and then further refine the retrieved set by applying Gabor feature vectors to select the top 3 best matched swatches. A random swatch is then picked from the 3 and mapped onto the input image segment. Figure 5 illustrates the process.

3.3 Computing Normal Maps and Leading

The results of the segmentation process are then prepared for real-time rendering. The goal was to demonstrate a realistic representation of a pane of stained glass using programmable 3D hardware on a typical PC. The examples provided in this paper use an Nvidia 6600*GT* and an ATI 9700. The shaders include approximations of the physical properties of a thin sheet of non-uniform glass including specular bump-mapped surface, reflection and refraction as well as appropriate appearance of the lead came.

(a) (b) (c) (d)

Fig. 4. Comparison of swatch selection based on only color and a combination of color and texture. a) Original image. b) Color based swatch selection. c) One example of color + texture based swatch selection. d) Another example of color + texture based swatch selection.



Histogram distances d_{hist} from left to right: 0.3224, 0.3775, 0.4224, 0.6893, 0.7013, 0.8633, 0.9422

Gabor feature vectors from left to right: 0.0138, 0.0156, 0.1329, 0.2003, 0.3852

Fig. 5. Using color and texture features to retrieve swatches from an image database given an input image segment. We first select the top 5 best matched swatches based on histogram intersection and then further refine the retrieved set by applying Gabor feature vectors to select the top 3 best matched swatches. A random swatch is then picked from the 3 and mapped onto the input image segment.

As a preprocessing step, normal maps are constructed for the sample glass swatches used in the output color image. Since no actual surface texture information is available for these samples, an approximation is used: each swatch is converted to a greyscale image and the image gradient intensity was mapped to height and run through a normal-map generation process. Then, a second pass over the output of the segmentation algorithms copies the texels from the normal maps rather than the glass color images to create an image representing the approximate texture of the glass surface for all texels in the segmented image.

Next, information about the lead came is generated in a separate preprocessing step. This information is generated from the image containing the segment IDs (where each segment is represented with a unique color). To locate where the lead came should be

placed, an edge detection filter is run on the ID image to locate the boundaries between each segment. These edges are dilated slightly to provide a thicker border appropriate to stained glass. The result of the dilation filter is then again interpreted as a height field by mapping the image gradient intensity to height and passed through a normal-map generation process. The grayscale edge image is copied to the alpha channel of the lead came texture to allow us to blend the segment interiors from the opaque lead border. The edge detection, dilation, and conversion to a normal map could all be performed within shaders on the GPU. However in this example, these masks are not dynamic and therefore results were simply precomputed and stored as textures.

3.4 Real Time Rendering and Shading

These examples provide an approximation of a realistic piece of stained glass suitable for real-time rendering. This effect uses two shaders and three passes. The first shader builds the glass effect from several intermediate terms which are illustrated in Figure 6. Then the preprocessed per-pixel normal-map generated earlier is used to calculate per-pixel diffuse lighting contributions. The contributions of both reflected light and refracted light relative to the viewer are also calculated with these surface normals. These reflection and refraction vectors are used to lookup into a dynamic cube-map of the environment to provide realistic appearance of the environment. Next, the reflection contribution is modulated by an approximation to the Fresnel term [26]. Finally for this pass, the specular contributions are calculated and combined separately using the per-pixel normals. The alpha component used for compositing to the frame buffer is set to the specular intensity to prevent the specular highlights being blended with the scene.

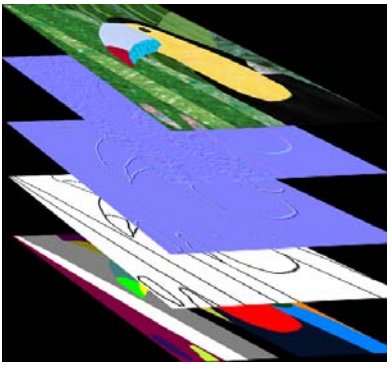


Fig. 6. Precalculated inputs (normals, color and alpha mask) for the shader lighting calculations

The final step is to provide the appearance of the lead came between the pieces of glass. The dilated edge mask is used in a shader program with an additional pass over the geometry. Any texels within the dilated edges are shaded using lights in the scene. However, only diffuse lighting with a constant dark blue-grey color is used to simulate the dull, fully diffuse appearance of real lead. The normal-map calculated from the dilated edge map is used to give the appearance that the leading is rounded. The

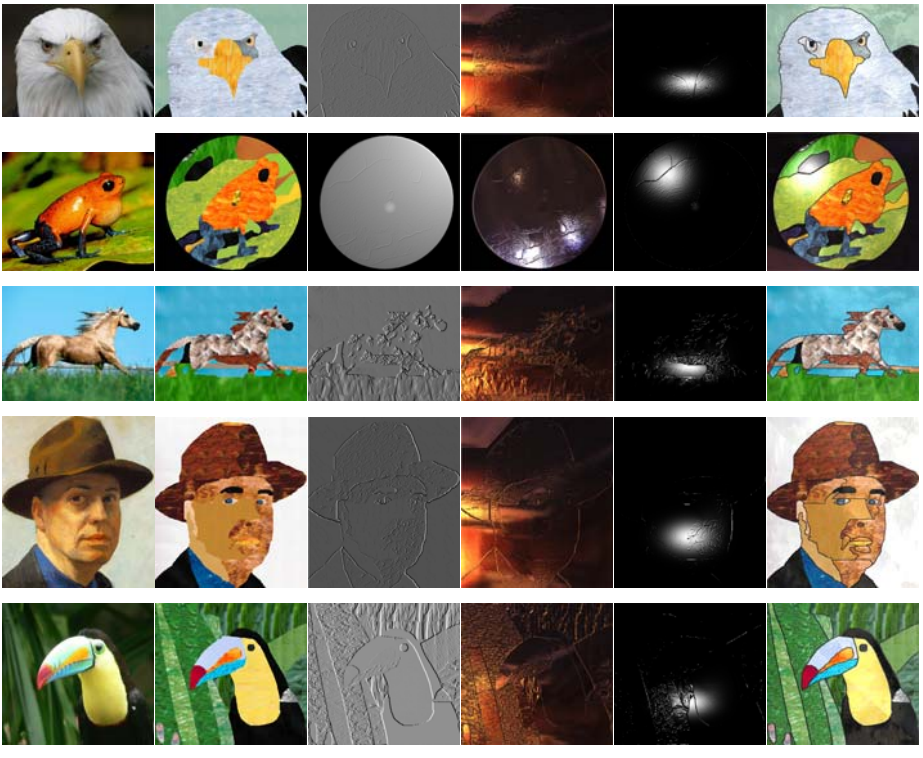


Fig. 7. An example set of input images, their intermediate results and the final rendered stained glass output

results of this pass are composited with the previous pass using the information in the alpha channel of the edge mask texture where texels within the lead came are the only fragments blended into the scene.

4 Conclusion

This paper documents an automated technique that filters an image to create results stylistically similar to modern stained glass artwork. Our contributions include the use of image segmentation, selective merging of adjacent segments and the matching of swatch images to the image segments based on color and texture features. We also provide several examples of this technique rendered in real time on readily available programmable PC 3*D* hardware. We have shown that this technique can be used to automatically create realistic representations of stained glass as well as capturing some of the style, vibrancy and expression of the art form.

References

[1] Robert Metcalf. *Making Stained Glass: a Handbook for the Amateur and the Professional*. New York: McGraw-Hill, 1972.

[2] Heribert Hutter. *Medieval Stained Glass*. New York: Crown Publishers, 1964.

[3] Mario Amaya. *Tiffany Glass*. New York: Walker, 1967.

[4] San Hunter and John Jocobis. *Modern Art: Painting/ Sculpture/ Architecture*. Prentice-Hall, Inc., 1976.

[5] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. In *SIGGRAPH 2002: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 769–776. ACM Press, 2002.

[6] Paul Haeberli. Paint by numbers: abstract image representations. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 207–214. ACM Press, 1990.

[7] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460. ACM Press, 1998.

[8] Bruce Gooch, Greg Coombe, and Peter Shirley. Artistic vision: painterly rendering using computer vision techniques. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*. ACM Press, 2002.

[9] John Lansdown and Simon Schofield. Expressive rendering: A review of nonphotorealistic techniques. *IEEE Comput. Graph. Appl.*, 15(3):29–37, 1995.

[10] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH 1994: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100. ACM Press, 1994.

[11] http://www.adobe.com. Adobe Photoshop.

[12] http://www.corel.com. CorelPaint.

[13] http://www.dfly.com/glasseye.html. Glass Eye 2000.

[14] Emma Biggs. *Encyclopedia of Mosaic Techniques*. Running Press Book Publishers, 1999.

[15] Junhwan Kim and Fabio Pellacini. Jigsaw image mosaics. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 657–664, New York, NY, USA, 2002. ACM Press.

[16] Alejo Hausner. Simulating decorative mosaics. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 573–580. ACM Press, 2001.

[17] David Mould. A stained glass image filter. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 20–25. Eurographics Association, 2003.

[18] Peter Meer and Bogdan Georgescu. Edge detection with embedded confidence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12), 2001.

[19] Yining Deng and b. s. Manjunath. Unsupervised segmentation of color-texture regions in images and video. volume 23, pages 800–810, Washington, DC, USA, 2001. IEEE Computer Society.

[20] M Swain and D Ballard. Color indexing. *International Journal on Computer Vision*, 7(1):11–32, 1991.

[21] David Douglas and Thomas Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. volume 10, pages 112–122, 1973.

[22] http://www.indexstock.com. IndexStock Photo Database.

[23] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. volume 22, pages 277–286, New York, NY, USA, 2003. ACM Press.

[24] BJohn R. Smith and Shih-Fu Chang. Automatic image retrieval using color and texture. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 18, pages 837–842. IEEE Press, 1996.

[25] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 18, pages 837–842. IEEE Press, 1996.

[26] Chris Brennan. *Accurate Environment Mapped Reflections and Refractions by Adjusting for Object Distance*. Wordware Publishing, Inc., 2002.