Rendering Traditional Mosaics

Gershon Elber
Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000, Israel
gershon@cs.technion.ac.il

George Wolberg
Department of Computer Science
City College of New York / CUNY
New York, NY 10031
wolberg@cs.ccny.cuny.edu

Abstract

This paper discusses the principles of traditional mosaics, and describes a technique for implementing a digital mosaicing system. The goal of this work is to transform digital images into traditional mosaic-like renderings. We achieve this effect by recovering freeform feature curves from the image and laying rows of tiles along these curves. Composition rules are applied to merge these tiles into an intricate jigsaw that conforms to classical mosaic styles. Mosaic rendering offers the user flexibility over every aspect of this craft, including tile arrangement, shapes, and colors. The result is a system that makes this wonderful craft more flexible and widely accessible than previously possible.

Keywords: mosaics, nonphotorealistic rendering, offset curves, Voronoi diagrams

1 INTRODUCTION

Nonphotorealistic rendering has enjoyed a surge of interest in recent years. Rendering algorithms have been introduced to mimic various classical artforms, including engraving [22, 29], pen-and-ink illustrations [27, 23], digital watercolors [3], line art drawing [7, 6, 15], expressive painting [19, 11], and Celtic art [10]. This paper addresses computer techniques to render one of the most ancient of classical artforms: mosaics.

Mosaics are designs and pictures formed from the juxtaposition of small tesserae (tiles) of stones, terracotta, or glass. The ancient art of mosaic is among the oldest, most durable, and most functional artforms. It was used in ancient Greece and Rome to adorn architectural surfaces. Intricate and fascinating mosaics were prominent in floor pavements, wall murals, and vault (ceiling) decorations.

The durability of the materials used has allowed mosaics to survive the ravages of time far more gracefully than paintings. As a result, there is much evidence that mosaics permeated many cultures and periods. They played a central role in Greco-Roman, early Christian, Byzantine, Islamic, medieval, and post-Renaissance art. Mosaics continue to be pervasive today in public buildings, plazas, subways, gardens, and restaurants.

Mosaics derive much of their splendor from scale. Upon close scrutiny, the skillful placement of tiles and the intricate tesselations that define the work are prominently visible. At a larger scale, the tiles fit together like jigsaw pieces into an abstract puzzle, forming a unique and striking blend of colors, designs, and images. The interplay between these different levels of abstraction, and our ability to resolve the "big picture" from the individual tiles, is what makes mosaics visually compelling.

It is important to distinguish the term "mosaic" from its uses in other fields. In image processing and computer vision, an image mosaic refers to a single large image stitched together from several smaller images. This is necessary for panoramas and terrain imagery, where a single image is not adequate to capture a sufficiently large field of view. In visual effects work for commercial advertising, photomosaics has recently been used to piece together many small images to form a single composite image [25]. This approach is more closely related to halftones where the true color of a region is approximated by an appropriately chosen textured pattern. Carrying this analogy further, classical mosaics can be considered to be generalizations of the uniform tessellation of digital images, whereby pixels (tiles) are not confined to a rectilinear grid, but rather may be oriented along arbitrary curves.

This paper presents a technique for simulating the classical mosaic artform. Rendering mosaics is essentially an exercise in establishing a tessellation that conforms to the principal features and strokes in a digital image. Once feature curves are extracted from the image, we compute offset curves to delimit rows of rectangular mosaic tiles. Since the offset curves may self-intersect, we trim them using Voronoi diagrams that are computed using a Z-buffer based approach. Finally, composition rules are applied to merge these tiles into an intricate jigsaw that conforms to classical mosaic styles.

An early attempt at mimicking traditional mosaics was described in [11]. In that work, a Z-buffer approach is used to compute the Voronoi diagram of a set of points in the plane, i.e., Dirichlet domain. This use of a Z-buffer to compute the Dirichlet domain is also described in the OpenGL user manual [28]. The computed regions of the Dirichlet domain comprise a simple tiling of the plane that crudely mimics a traditional mosaic. The method does not attempt to align mosaic tiles along prominent strokes, a property common to classical mosaics.

There are several commercial software packages that claim to offer mosaic-like renderings. The Adobe Photoshop plug-in attempts to achieve this effect by tesselating the image into tiles. Unfortunately, the tiles do not conform to the principal features and strokes in the image. More compelling visual effects are possible in Painter from Metacreations through the use of hand-drawn strokes along which a trail of mosaic tiles are rendered. The tile colors are sampled from an underlying image. Although the resulting mosaics can be quite impressive, this manual approach is very tedious and cumbersome as the user must draw all the strokes that constitute the rendering. Successive rows of tiles are due to painting successive strokes, i.e., there is no attempt at inferring adjacent strokes from a few key hand-drawn strokes.

Recent work has attempted to address the tile alignment problem. In [12], tiles are made to conform to user-specified image features. A centroidal Voronoi diagram (CVD) is used to create a point distribution in the plane. A CVD is a Voronoi diagram with the property that each site is located at the centroid of its region. An iterative algorithm is invoked by which a Voronoi diagram is computed from a set of site points, and the site of each region is then moved to the region's centroid. Although no proof is given, the regions of the resulting site points empirically converge to a nearly optimal tiling of the plane. The tiling is hexagonal when minimizing a Euclidean distance metric. Since square tiles are desired, a Manhattan distance metric is used. A gradient field derived from image feature curves is then used to orient the tiles properly.

In this paper, a different approach is taken. Examining the results of [12], we note that many of the resulting tiles are misaligned, an artifact that is clearly present in regular domains where a uniform tile placement is expected. In traditional mosaics, several rows of tiles are precisely placed along the feature curves, a behavior that the gradient field of [12] cannot preserve. This paper overcomes these difficulties as well as others by offering a more precise and geometrically oriented approach.

This paper is organized as follows. Section 2 reviews the history of the mosaic artform. Section 3 presents the algorithm, including the extraction of feature curves, trimmed offset curves, and tile placement. Examples are given in Section 4. Finally, Section 5 discusses conclusions and future work.

2 HISTORICAL BACKGROUND

The history of mosaics dates back several thousand years. Some of the earliest known mosaics adorn the exteriors of buildings constructed during the third millennium BC at Uruk in Mesopotamia. There, the Sumerians embedded long terracotta cones into walls to protect the underlying structures and to decorate their surfaces with colorful geometric patterns [4]. In ancient China, mosaics made of pebbles were important features in gardens and pavements. Pebble mosaic floors dating back to the eighth century BC have also been discovered in Gordium, a town near Ankara, Turkey. The pebbles were arranged in simple geometric patterns.

Mosaic art later flourished in ancient Greece and Rome, where its practitioners created some of the most beautiful artworks in history. This golden period coincided with the introduction and widespread adoption of a new mosaic material: natural stone cut into small tiles, called *tesserae*. Since tesserae were cut into triangular, square, and rectangular shapes, they could be fit together more closely than pebbles, and the resulting artwork process was refined. This advent began in the Hellenistic period, during the reign of Alexander the Great (336-323 BC) and was later embraced by the Romans. Some of the earliest

Roman works were created in Pompeii during the second or first century BC and were preserved in good detail by the eruption of Mount Vesuvius in 79 AD.

Mosaics were popular for rendering geometric patterns, vegetal motifs, and figure compositions used in pavements from the fourth century BC to the early Christian period in the third century [17]. By this time, mosaics moved from floors onto walls and depictions of scenes from the Old and New Testament became prevalent. Glass smalti, which had been used sparingly in mosaics floors as early as the third century BC, became increasingly popular and ultimately dominated the early Christian mosaics.

The Byzantine era, between the fifth and fifteenth centuries, saw the greatest growth of mosaic as an artform. Mosaics were no longer confined to discrete panels but now covered entire walls and ceilings. Immense figures became important design elements and matched the scale of the surrounding architecture [4]. Chief innovations during the Byzantine era included the use of glazed, silver, and gold tesserae. Their reflective qualities were enhanced by setting them at oblique angles to create an undulating surface that exploited reflection. Golden halos set in this manner appear to shimmer with light, in stark contrast with the dull stone tesserae used for surrounding figures [4]. The most well-known mosaics of this era are found in Ravenna, Italy, the last Imperial capital of Italy.

The use of mosaics spread to mosques between the seventh and tenth centuries. Beautiful geometric designs adorn the Great Mosques in Cordoba and Damascus, and the Dome of the Rock in Jerusalem.

The artform went through a period of decline after the Renaissance due to the rising influence of paintings. The eighteenth and nineteenth centuries saw a revival. Mosaics are again popular for adorning architectural surfaces and public spaces. Interested readers may consult [17, 2, 1, 26] for further background.

3 ALGORITHM

A key observation derived from traditional mosaics, is the fact that they emphasize feature curves of importance in the picture. Rows of tiles are arranged along these feature curves. Here, two typical arrangements can be found. The rows along feature curves continue throughout the picture, or alternatively a small number of rows follows the feature curves whereas a background tiling covers the rest of the picture.

In order to emulate the placement of mosaic tiles along feature curves, we seek a model that is able to compute arrangements of tiles along freeform curves. An algorithm that is capable of such a requirement must obey the following steps:

- Detecting and extracting feature curves from the picture. This stage is described in Section 3.1.

- Computing the offsets of these feature curves for all the rows of tiles that are expected to follow this

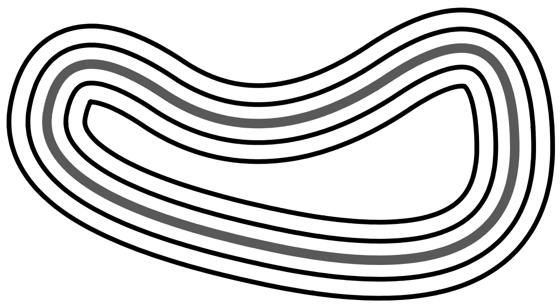

Figure 1: Parallel curves to a given curve (in gray) can be computed as offsets of the given curve.

feature curve. The offset computation procedure is presented in Section 3.2.

- Trimming the offset curves in self intersecting locations so the tiling is indeed proper. The problem, as well as the computation of properly trimmed offsets, is described in Section 3.3.
- The placement of the tiles of the mosaics. This process is presented in Section 3.4.

3.1 Extraction of Feature Curves

The automatic extraction of feature curves from an image is a computer vision problem [24] that is beyond the scope of this work. In this work, we shall require the user to specify feature curves in much the same manner that it is done for morphing [16] and digital facial engraving [22]. Herein, we emphasize this feature extraction stage in the context of placement of mosaic tiles. Feature curves in real mosaics are, almost exclusively, edges that delineate the foreground from the background. In that respect, they are relatively simple to extract using image processing techniques. In practice, however, we found that the semi-automatic approach is much more attractive and tools such as intelligent scissors [21] could be employed toward the definition of the feature curves. In this application, we assume that the feature curves were extracted and are least squares fitted into freeform B-spline curves. Hence, a set $\mathcal{C} = \{C_i(t)\}_{i=0}^{n-1}$, $t \in [0,\ 1]$ of n feature curves in the original picture is the result of this stage.

3.2 Offset of Feature Curves

The mosaic tiles are to be arranged along parallel curves to the feature curve $C_i(t) = (x_i(t), y_i(t))$. Denote by δ the width of a square tile. Then, we seek to compute m parallel curves $C_i^j(t)$, $0 \leq j \leq m-1$ to $C_i(t)$, δ distance apart, as in Fig. 1.

Since $C_i(t)$ is a planar curve, the unit tangent field, $T_i(t)$ and the unit normal fields, $N_i(t)$, of $C_i(t)$ are also planar. Let

$$T_i(t) = \frac{(x_i'(t), y_i'(t))}{\sqrt{(x_i'(t))^2 + (y_i'(t))^2}},$$

be the unit tangent field of $C_i(t)$. Differentiating with respect to the arc length parameter s yields $\langle T_i'(s), T_i(s)\rangle = \kappa(s)\ \langle N_i(s), T_i(s)\rangle = 0$, where $\kappa(s)$ is the scalar curvature field of $C_i(t)$ and $\langle T_i(t), T_i(t)\rangle = 1$. $T_i(t)$ is orthogonal to $N_i(t)$ in the plane, and hence we constructively have,

$$N_i(t) = \frac{(-y_i'(t), x_i'(t))}{\sqrt{(x_i'(t))^2 + (y_i'(t))^2}},$$

Moreover, from the Frenet equations [5] we know that $N_i'(s) = -\kappa T_i(s)$ for a planar curve. With this differential geometry analysis we are ready to show that the offset curves,

$$\begin{aligned} C_i^j(t) &= C_i(t) + j\ \delta\ N_i(t), \\ &\qquad 0 \le i \le n-1, \quad 0 \le j \le m-1, \\ &= C_i(t) + j\ \delta\ \frac{(-y_i'(t), x_i'(t))}{\sqrt{(x_i'(t))^2 + (y_i'(t))^2}}, \end{aligned} \tag{1}$$

do indeed satisfy the parallel conditions. In other words, $C_i(t)$ and $C_i^j(t)$ are parallel for all j and t in the domain. Two curves are said to be parallel if their tangent fields point in the same direction:

$$\begin{aligned} C_i^{j\prime}(t) &= C_i'(t) + j\ \delta\ N_i'(t), \\ &= \alpha T_i(t) + \beta T_i(t), \\ &= (\alpha + \beta) T_i(t), \\ &= \gamma C_i'(t), \end{aligned} \tag{2}$$

for some scalars $\alpha,\ \beta,\ \gamma\ \in \mathbb{R}$.

Since $N_i(t)$ is not rational, one cannot represent the offset as a B-spline or a NURB curve, even if $C_i(t)$ is a B-spline curve. Numerous approximation methods were derived for rational offsets of rational curves and a recent survey can be found in [9]. Here, we only assume the availability of a robust offset approximation scheme of B-spline curves that provides an error bound on the approximation. In this mosaic application, the error bound is selected to be in the order of a single pixel.

Unfortunately, even if we employ a robust offset approximation scheme that creates a rational form, the result can be invalid in many cases. If the curvature κ of the curve is larger than $\frac{1}{j\delta}$, the offset curve will *self intersect*.

Examining Eq. (2), $C_i^{j\prime}(t)$ might vanish if $\alpha = -\beta$. Intrinsically, this condition occurs when $j\ \delta\ \kappa = 1$. If the offset amount, $j\ \delta$, equals the radius of curvature, $\frac{1}{\kappa}$, $C_i^{j\prime}(t)$ *vanishes into a cusp.* Furthermore, when

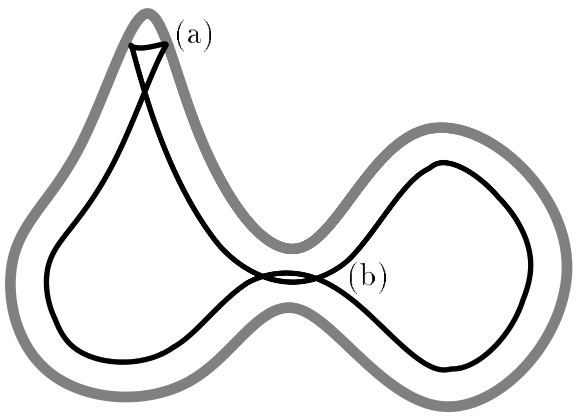

Figure 2: Local (a) and global (b) self intersections in an offset of a freeform curve.

$j\ \delta > \frac{1}{\kappa}$, the direction of the tangent vector of $C_i^{1\prime}(t)$ is reversed with respect to $C_i'(t)$! This type of self intersection, which we denote as local, is seen in Fig. 2(a).

Self intersections in the offset could also occur between two independent locations of the curve, a type of self intersection we denote as global. Fig. 2(b) shows one such example.

3.3 Trimming the Offset

The elimination of the self intersection in offset approximations is a very difficult problem. In essence, we seek the true offset of the curve $C_i(t)$. Recall that $\|C_i^j(t) - C_i(t)\| = j\ \delta$, $\forall t$ in the domain and up to the offset approximation accuracy. We seek all points $C_i^j(t_0)$ in curve $C_i^j(t)$ such that $\|C_i^j(t_0) - C_i(t)\| \geq j\ \delta$, $\forall t$ in the domain.

As demonstrated in Section 3.2, two reasons could require the trimming of the offset curve: a local and a global self intersection. Due to the distinctive intrinsic curvature property of local self intersections, they are fairly simple to detect [8]. In contrast, global self intersections are far more difficult to detect due to the lack of any intrinsic behavior that could be analyzed beyond the computation of the intersections themselves.

Requiring a highly robust trimming procedure for offset curves, the traditional numeric approach is difficult to employ. However, the application at hand is discrete which suggests that a discrete solution might be sufficient. In [14, 18], Voronoi diagrams are computed using a discrete image-based approach that approximates the Euclidean distance. In [11], the use of a Z-buffer approach to the computation of a discrete Voronoi diagram of a set of points in the plane is introduced, an approach that is also cited in the OpenGL user manual [28]. In [13] this approach is extended to employ graphics hardware. One can exploit the Z-buffer based computation of Voronoi diagrams to robustly trim the offset curves, benefiting

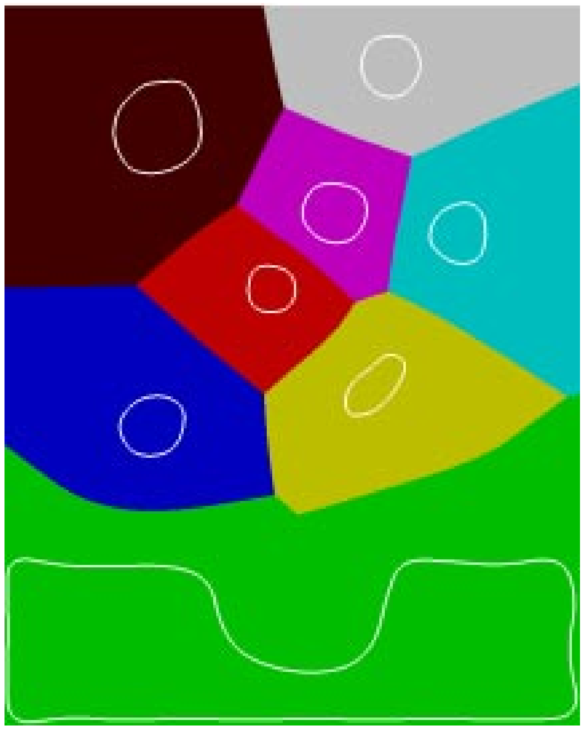

Figure 3: A Voronoi diagram could be approximated discretely, using a Z-buffer based approach. Feature curves of the sunflower drawing shown in Fig. 12.

from the inherent robustness of the Z-buffer paradigm.

Interestingly enough, the application of mosaics is discussed in [11] as well as in [13], employing the *Voronoi cells* as the tiles of the mosaics. A Voronoi cell of planar entity $C_i(t)$ contains all the pixels that are closer to $C_i(t)$ than to any other planar entity $C_j(t)$, $j \neq i$. In [11, 13], however, the Voronoi diagrams are computed on a collection of points that are distributed more densely along edges in the image. They do not attempt to compute offset curves. Herein, we use Voronoi diagrams only to assist in trimming the offset curves.

Let the orthographic viewing direction be the $+z$ direction. Then, for each point in $C_i(t_0) \in C_i(t)$, construct the cone $z = +\sqrt{(x - x_i(t_0))^2 + (y - y_i(t_0))^2}$ and render it into the Z-buffer in a color that is unique to entity $C_i(t)$. Doing so for all n curves in the plane, one ends up with color coded Voronoi cells such that the Voronoi cell of $C_i(t)$ is uniquely painted with the color allocated to entity $C_i(t)$. See Fig. 3 for an example.

In practice, $C_i(t)$ is approximated by linear segments and the sweep of a cone along $C_i(t)$ is approximated by small ravine-like shapes, such as the one presented in Fig. 4. Hence, the process of sweeping of a cone along the interior of the curve can be approximated by linear segments and two quadrilateral polygons,

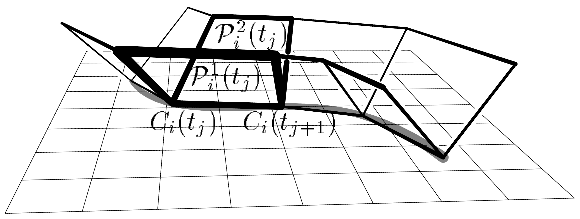

Figure 4: One interior segment of the piecewise linear approximation of $C_i(t)$ contributes two quadrilaterals, $\mathcal{P}_i^1(t_j)$ and $\mathcal{P}_i^2(t_j)$, to the cones swept along $C_i(t)$.

$\mathcal{P}_i^1(t_j)$ and $\mathcal{P}_i^2(t_j)$ emanating at 45 degrees from the opposite sides of each linear segment, $(C_i(t_j), C_i(t_{j+1}))$. Again, see Fig. 4.

We are now ready to adapt the discrete, Z-buffer based, Voronoi diagram computation scheme to our aid. Before we present this adaptation, we must also realize a crucial difference. If point P is in the Voronoi cell of curve $C_i(t)$ it is clearly closer to $C_i(t)$ than to any other curve. In other words, there exists $C_i(t_0)$ such that $\|C_i(t_0) - P\| \leq \|C_k(t) - P\|$, $\forall k, t$. However, here one needs to find this t_0 so as to align the tile at P to be parallel to the curve $C_i(t)$ at t_0. Finding t_0 in $C_i(t)$, given a point P in the Voronoi cell of $C_i(t)$ is a query that the basic Z-buffer approach to the computation of the Voronoi diagram cannot answer in the forms presented by [11, 14, 13].

Let point $C_i(t_0)$ be denoted as the *foot point* of P. We need to extend this Z-buffer approach to support the efficient detection of the foot points. Instead of giving a unique color to the entire Voronoi cell of $C_i(t)$, we are going to allocate a unique color to each pixel of $C_i(t)$ as it is rendered in the image space. Let $(C_i(t_j), C_i(t_{j+1}))$ be one linear segment approximating $C_i(t)$ (Fig. 4). We then color vertex $C_i(t_j)$ with a unique color index, $\mathcal{C}_i(t_j)$, that equals to,

$$\mathcal{C}_i(t_j) = \mathcal{C}_i(0) + \sum_{k=1}^{j} \|C_i(t_k) - C_i(t_{k-1})\|,$$

having $t_0 = 0$.

In other words, each vertex is assigned a color index that equals the chord length of the curve up to that vertex. Having a typical color space of 24 bits and a typical chord length on the order of a thousand pixels, one can allocate unique color indices to tens of thousands of curves before exhausting the entire color space. But now we have assigned every rendered pixel of every curve with a unique color. Assign the same colors of vertices $C_i(t_j)$ and $C_i(t_{j+1})$ to the two other vertices of quadrilaterals, $\mathcal{P}_i^1(t_j)$ and $\mathcal{P}_i^2(t_j)$. Then, rendering polygons $\mathcal{P}_i^1(t_j)$ and $\mathcal{P}_i^2(t_j)$ with the prescribed vertex colors would assign a *unique color for each pixel over all Voronoi cells.* Given point P in the Voronoi cell of $C_i(t)$, one needs to examine that color under P and search for that color in $C_i(t)$. Because the colors are assigned based on the chord length,

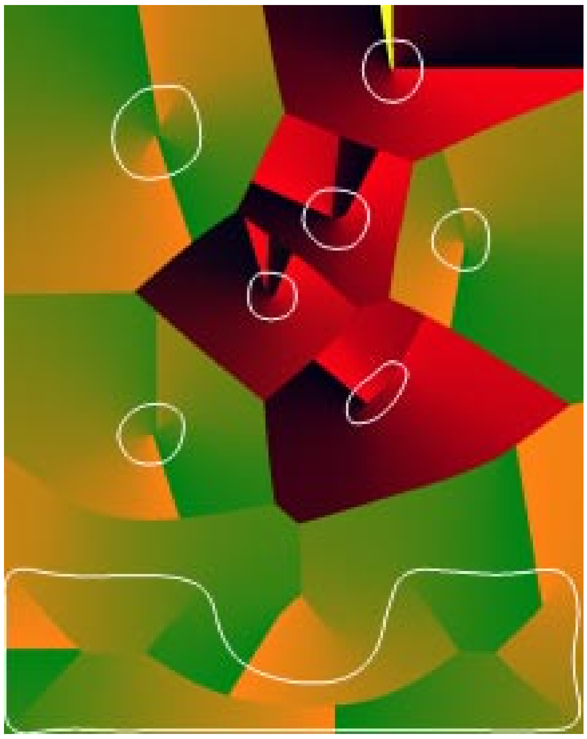

Figure 5: The extended Voronoi diagram that assigns each pixel along the path a unique color. Compare with Fig. 3.

this search becomes trivial. Moreover, as will be seen in the next section, tiles are placed consecutively, and one could also exploit this spatial coherence as we march along an offset curve and place one tile after another.

Fig. 5 shows the same voronoi diagram of Fig. 3, but this time with the assignment of each pixel along the path, with a unique color.

3.4 Placement of Mosaic Tiles

Having been able to compute parallel curves and properly trim them, the last stage is the placement of the tiles. One can march along the parallel curves and place the tiles so that they are tangent to the curve along one edge of the tile while packed as closely as possible against the adjacent tiles in this row. This intuitive description is followed in the algorithmic approach taken. We place one tile at a time along a row. The next tile is oriented to be tangent to the curve, while we “slide” the tile in until it comes into contact with the previous tile (see Fig. 6).

While this simple algorithm works reasonably well in low curvature areas, near high curvature regions of the curve, the gaps could be quite large (see Fig. 6). Even more disturbing are the artifacts that result

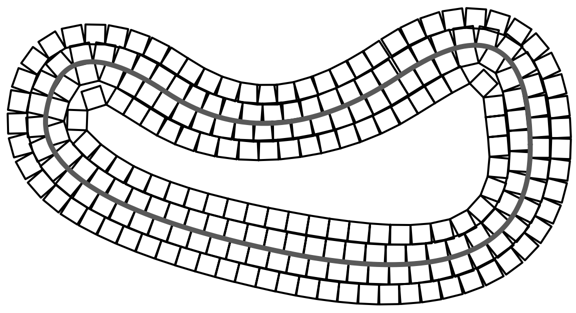

Figure 6: Tile placement for the curve in Fig. 1.

near the trimmed zones. Here, the two fronts of two different Voronoi cells of the curves meet and we must decide what tile from what front to employ. Clearly, one can decide to break the tiles into non square or even non rectangular shapes. Straight clipping all too clearly demarcates the meeting edges of the different Voronoi cells and these undesirable artifacts can be visible in the final result, as will be demonstrated in the Section 4.

Superior results can be obtained by randomly allowing the penetration of tiles from one front into its adjacent front, in an attempt to alleviate some of these artifact. Examples of this approach, as well as others, can be found in the next section.

4 EXAMPLES

This section demonstrates the algorithm on various input images. Figures 7 through 9 depict several pairs of input and output images. In Fig. 10, the feature curves used to place the tiles of Fig. 8 and Fig. 9 are shown. Notice that the tiles conform to image features. Furthermore, the composition of tiles across wavefronts is well-behaved. Fig. 11 depicts the use of only a few lines of tiles along the features curves and the use of either uniform squares or hexagonal tiles as background. Fig. 12 shows a similar example of only few lines of tiles over a background formed of diamond and hexagonal tiles.

Tiles need not necessarily be of uniform color. Fig. 14 is a preliminary attempt to employ the bank of images shown in Fig. 13, to tile a mosaic. In Fig. 14(a), a mosaic similar to that of Fig. 8 is shown except that the tiles are now actual images. Fig. 14(b) shows a closeup of the photomosaic of the back of the dinosaur head.

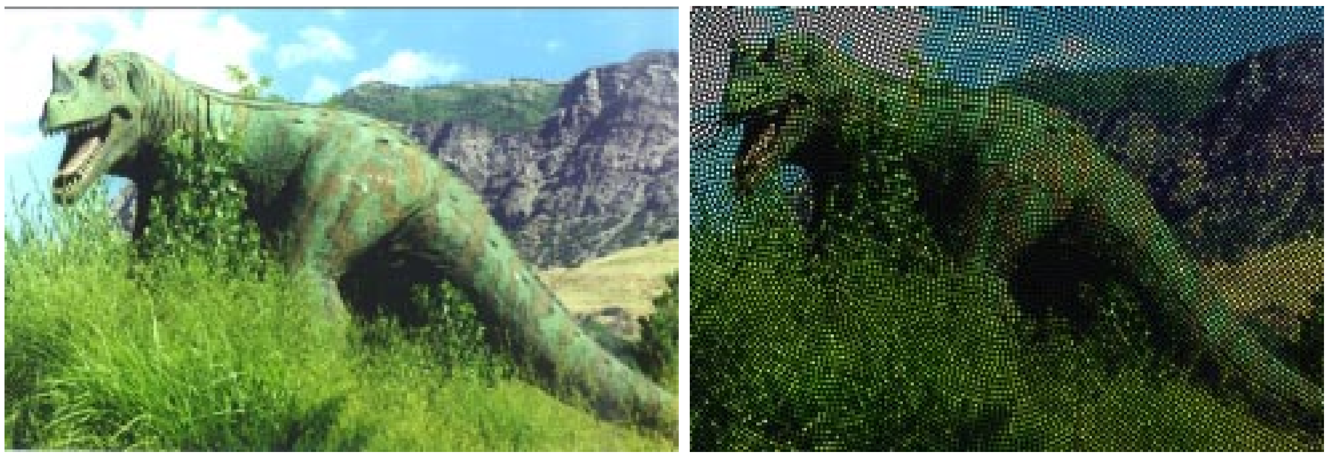

Figure 7: An image of a dinosaur (left) and a mosaic reconstruction (right). Image from the Utah dinosaur museum in Ogden.

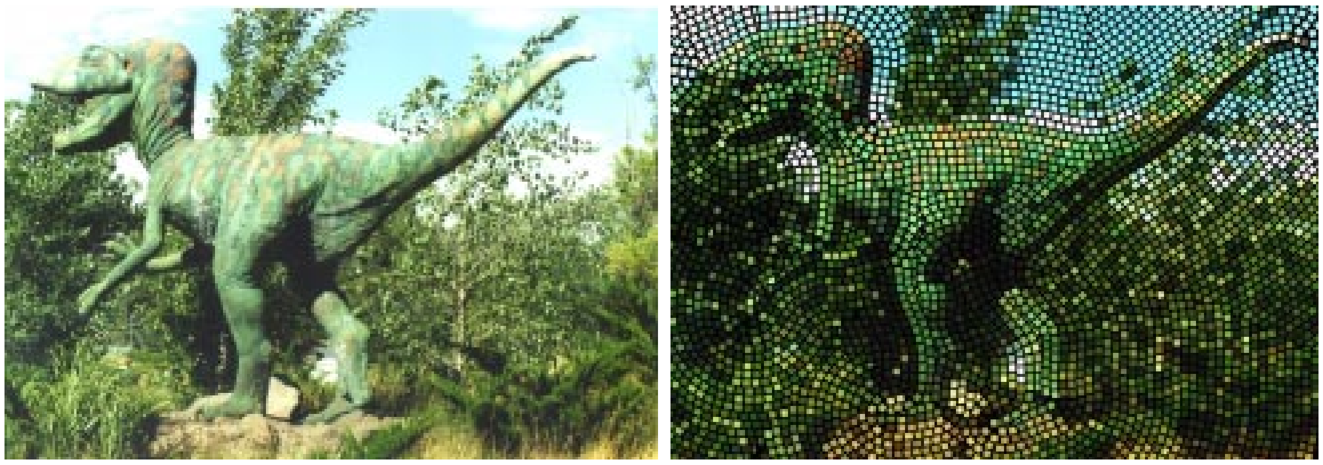

Figure 8: An image of a dinosaur (left) and a mosaic reconstruction (right). Image from the Utah dinosaur museum in Ogden.

5 CONCLUSIONS

We have presented a technique for rendering traditional mosaics. The proposed system exploits extracted and drawn feature curves to generate a tessellation in which to lay down tiles. The colors of the tiles are sampled from the underlying image. We reviewed the use of Voronoi diagrams to help trim the offset curves. A fast algorithm based on a hardware Z-buffer was used for this purpose [13].

Although the use of Voronoi diagrams has been suggested in the literature to produce mosaic images, it is important to note that we use Voronoi diagrams only as a means to trim offset curves. The straightforward application of Voronoi diagrams for tessellation, as is used in some commercial software, does not produce

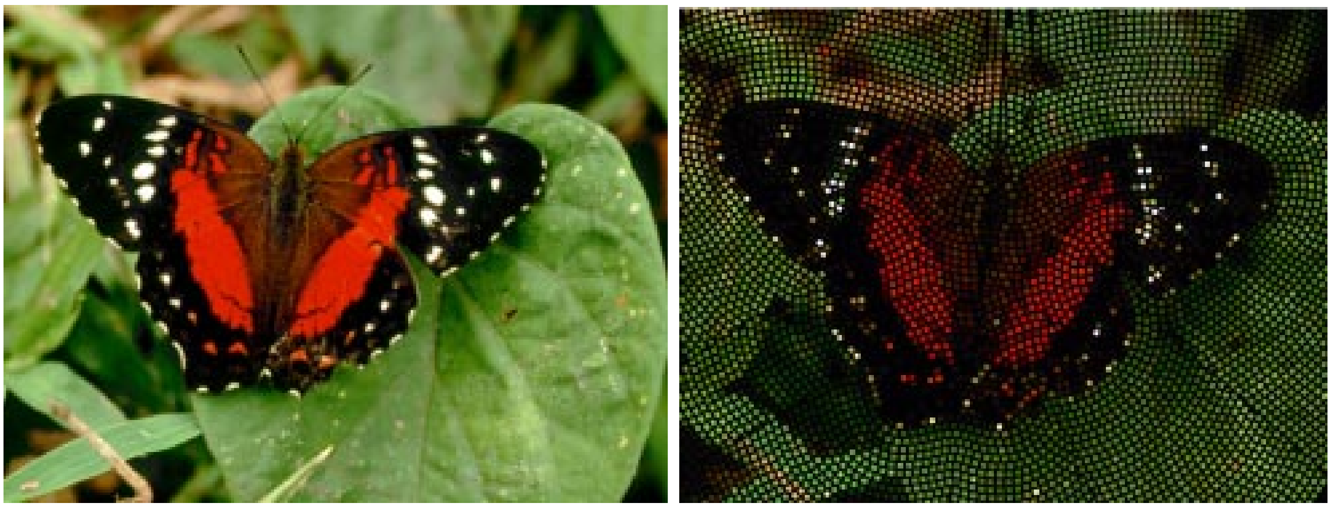

Figure 9: A butterfly picture (left) and mosaic reconstruction (right).

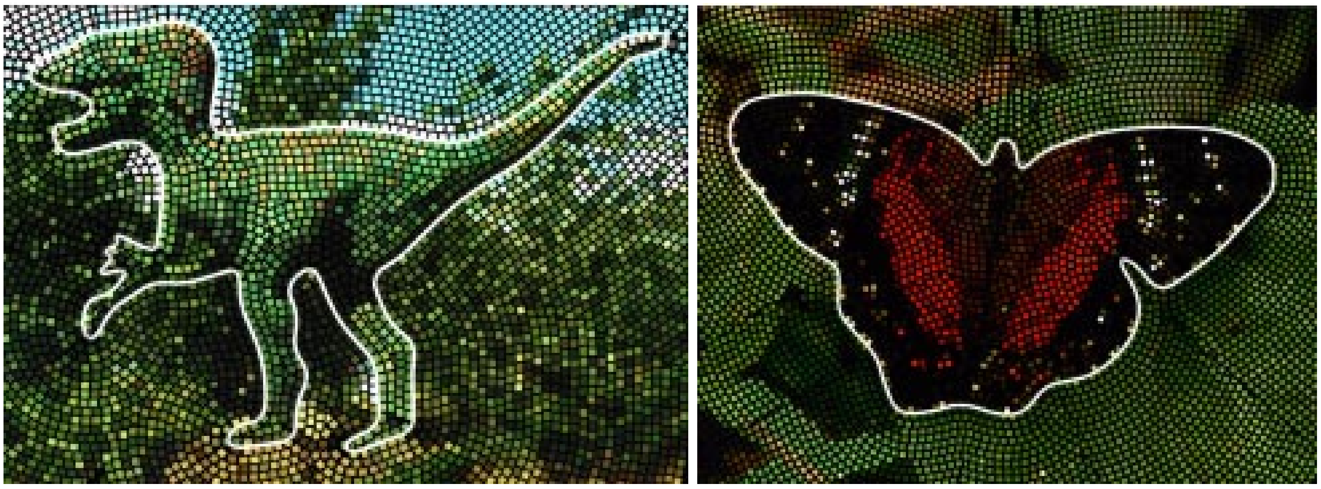

Figure 10: The feature curves used to place the tiles are shown for Fig. 8 (left) and Fig. 9 (right).

tile placement consistent with traditional mosaics.

While this paper lays out the basic approach to perform traditional mosaicing, a number of enhancements are possible. Future work remains in implementing the ideas described below.

The presented approach leaves visual artifacts in the forms of virtual lines along the skeleton of the feature curves, or the locus of singular points of the offset curves. Further, due to the precise nature of the proposed approach, the placement of the tiles seems too precise at times. One can expect that by employing a gradient field, proposed by [12], this synthetic artifact can be mitigated. Here, tiles in the neighborhood of the skeleton will be allowed to move under the influence of the gradient field, an amount

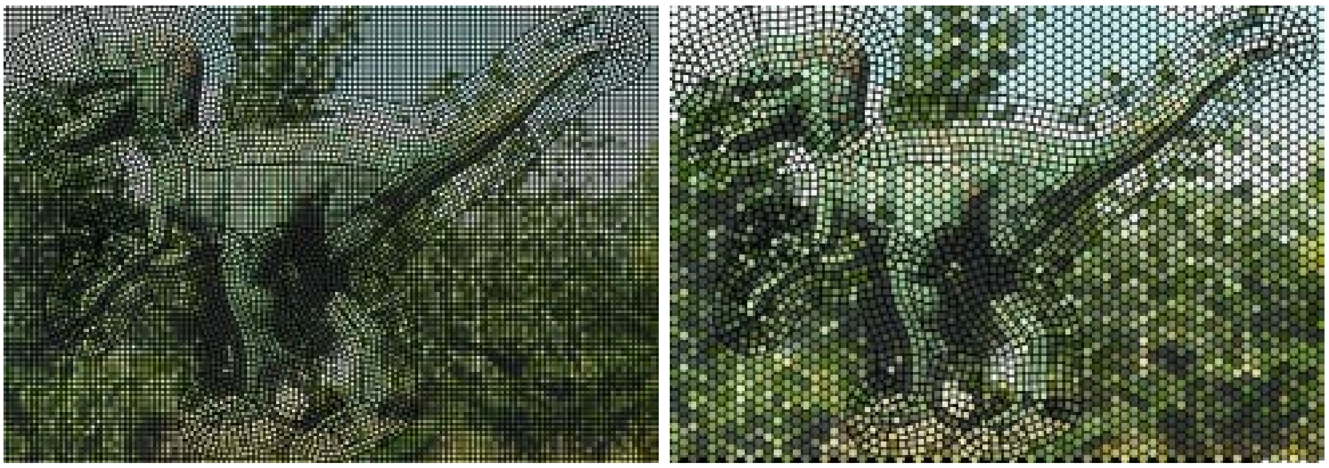

Figure 11: Placement of few rows of tiles along feature curves. Small tiles and square background tiles (left) and large tiles and hexagonal background tiles (right). Compare with Fig. 8.

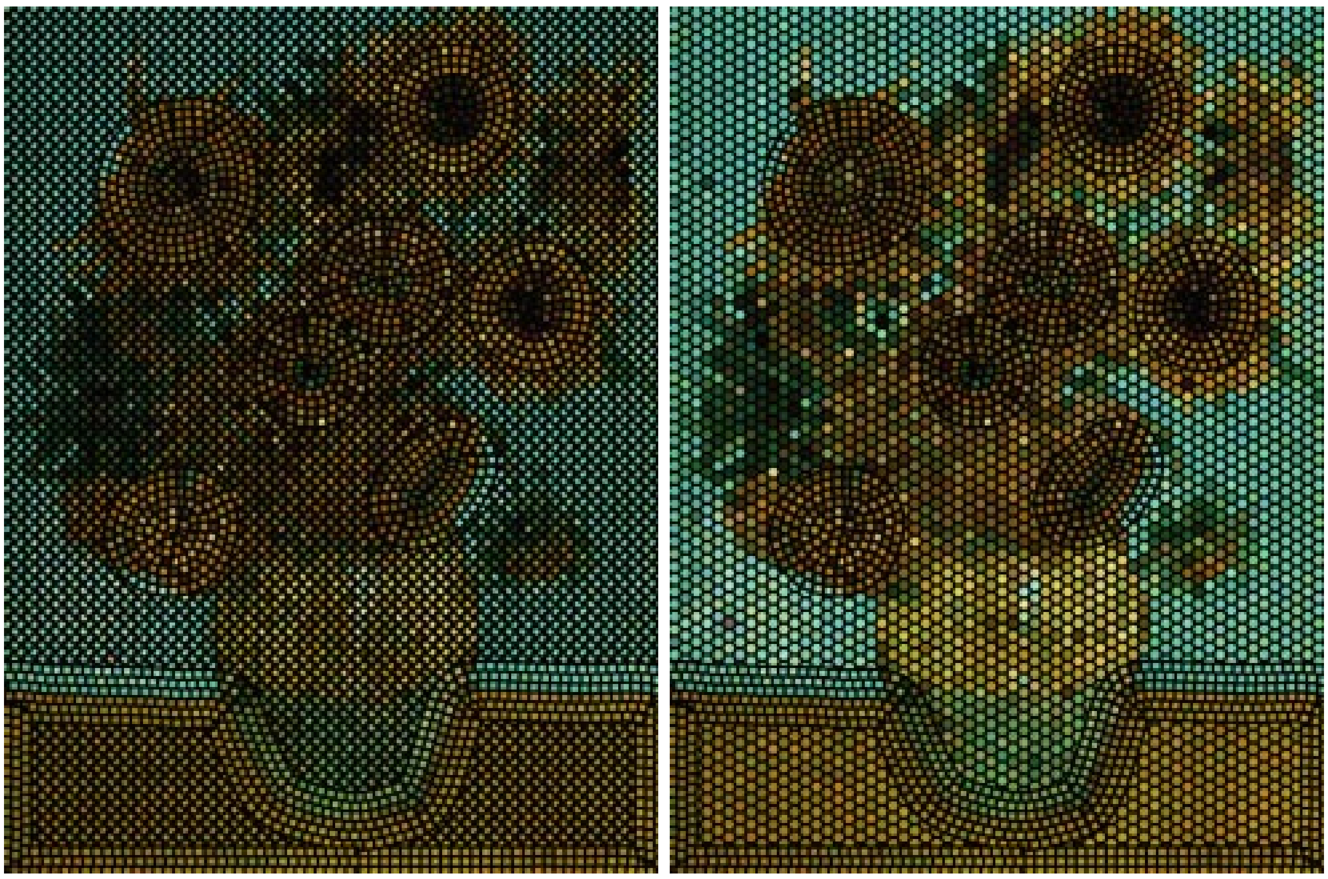

Figure 12: Placement of few rows of tiles along feature curves. Diamond background tiles (left) and hexagonal background tiles (right).

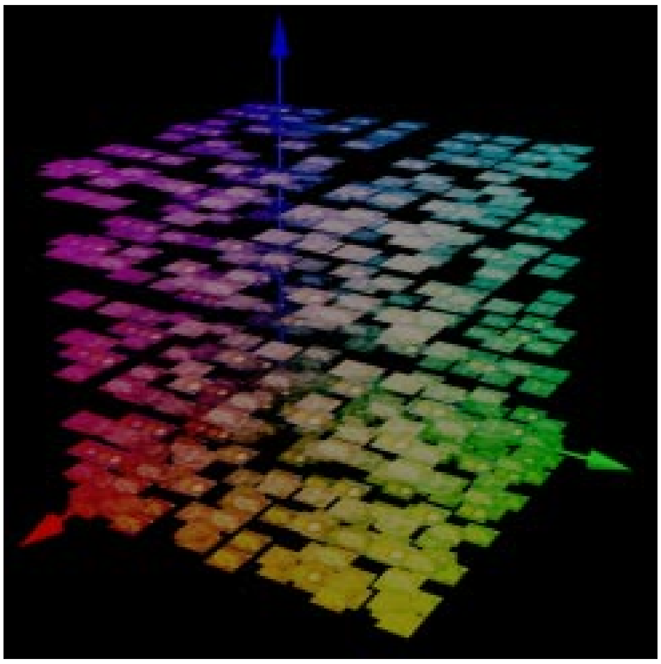

Figure 13: Tiles could employ images instead of a uniform color. Presented here is a bank of images in RGB space that is used in Fig. 14 to tile a mosaic.

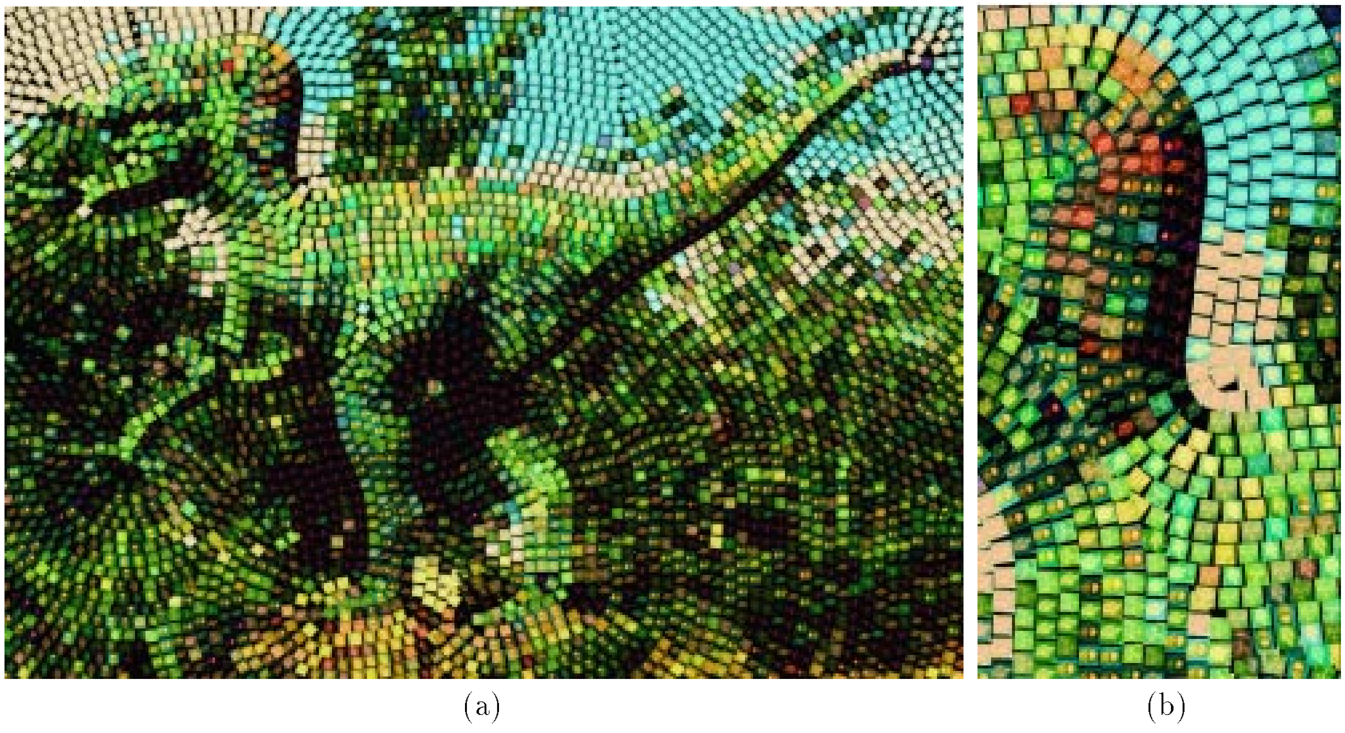

Figure 14: Generalized photomosaic of dinosaur. See also Fig. 8 and Fig. 11.

that is a function of the distance to the skeleton, preserving the overwhole price placement of tiles.

Visually interesting effects are possible if we use tiles textured with another image. This idea is actually a generalization of photomosaics [25]. In photomosaics, an abstract version of an image is generated by packing a dense set of smaller images into a nonuniform upright grid. In mosaics, we may abandon the rectilinear grid with intricately flowing rows of tiles, each consisting of a small image.

Although traditional mosaics place tiles along rows that grow from designated feature curves, it seems reasonable to consider other tile placement strategies. Indeed, the digital engraving system described in [22] shares similar goals for orienting lines in response to image features. This suggests the use of engraving lines, as produced by that system, for tile placement.

The mosaic image need not be constructed from flat two-dimensional tiles. Instead, we can render the image in 3-D with any desirable reflection model. Interesting effects are possible if we integrate the stone-generation algorithm of [20] to render the mosaic tiles. It is interesting to note that during the Byzantine era, mosaic tiles were often set at oblique angles to achieve a shimmering effect. This effect can be simulated easily once we impose specular reflection on the 3-D tiles.

The proposed system deals primarily with rectangular and triangular tiles. A future enhancement will permit the user to introduce other tile shapes, including hexagons, diamonds, and irregular shapes.

There are several benefits to a digital mosaic rendering system. Every aspect of the creative process can now be edited and rendered at near real-time rates. The user may readily alter the set of feature curves, edit the offset curves, designate the composition rules, and control the tile shapes and colors. The result is a system that makes this wonderful craft more flexible and widely accessible than previously possible.

References

[1] Bertelli, C. *Les Mosaiques.* Bordas Publishers, Paris, 1993.

[2] Chavarria, J. *The Art of Mosaics.* Watson-Guptill Publications, New York, 1999.

[3] Curtis, C. J., Anderson, S. E., Seims, J. E., Fleischer, K. W., and Salesin, D. H. Computer-generated watercolor. *Computer Graphics (Proc. SIGGRAPH '97)* (1997), 421–430.

[4] Dierks, L. *Making Mosaics.* Sterling Publishing Co., New York, 1997.

[5] do Carmo, M. P. *Differential Geometry of Curves and Surfaces*, 2nd ed. Academic Press, 1990.

[6] Elber, G. Line art rendering via a coverage of isoparametric curves. *IEEE Trans. Visualization and Computer Graphics 1*, 3 (September 1995), 231–239.

[7] Elber, G. Line art illustrations of parametric and implicit forms. *IEEE Trans. Visualization and Computer Graphics 4*, 1 (1998), 71–81.

[8] Elber, G., and Cohen, E. Error bounded variable distance offset operator for free form curves and surfaces. *International Journal of Computational Geometry & Applications 1*, 1 (March 1991), 67–78.

[9] Elber, G., Lee, I.-K., and Kim, M.-S. Comparing offset curve approximation methods. *IEEE Computer Graphics and Applications 17*, 3 (May-June 1997), 62–71.

[10] Glassner, A. Celtic knotwork. *IEEE Computer Graphics and Applications 19*, 5 (September-October 1999), 78–84.

[11] Haeberli, P. Paint by numbers: Abstract image representations. *Computer Graphics (Proc. SIGGRAPH '90) 24*, 4 (1990), 207–214.

[12] Hausner, A. Simulating decorative mosaics. *Computer Graphics (Proc. SIGGRAPH '01)* (2001), 573–580.

[13] Hoff, K. E., Culver, T., Keyser, J., Lin, M., and Manocha, D. Fast computation of generalized voronoi diagrams using graphics hardware. *Computer Graphics (Proc. SIGGRAPH '99)* (1999), 277–286.

[14] Hoffmann, C. Computer vision, descriptive geometry, and classical mechanics. In *Computer Graphics and Mathematics*, R. Falcidieno, I. Herman, and C. Pienovi, Eds. Springer Verlag, Berlin, 1992, pp. 229–243. Also available as Technical Report, CSD-TR-91-073, Computer Science Department, Purdue Univeristy, October, 1991.

[15] Lansdown, J., and Schofield, S. Expressive rendering: A review of nonphotorealistic techniques. *IEEE Computer Graphics and Applications 15*, 3 (May 1995), 29–37.

[16] Lee, S., Chwa, K.-Y., Shin, S. Y., and Wolberg, G. Image metamorphosis using snakes and free-form deformations. *Computer Graphics (Proc. SIGGRAPH '95)* (1995), 439–448.

[17] Ling, R. *Ancient Mosaics*. Princeton University Press, New Jersey, 1998.

[18] Mantanari, U. A method for obtaining skeletons using a quasi-euclidean distance. *Journal of Computing Machinery 16*, 4 (1968), 534–549.

[19] Meier, B. Painterly rendering for animation. *Computer Graphics (Proc. SIGGRAPH '96)* (1996), 477–484.

[20] Miyata, K. A method of generating stone wall patterns. *Computer Graphics (Proc. SIGGRAPH '90) 24*, 4 (1990), 387–394.

[21] Mortensen, E. N., and Barrett, W. A. Intelligent scissors for image composition. *Computer Graphics (Proc. SIGGRAPH '95)* (1995), 191–198.

[22] Ostromoukhov, V. Digital facial engraving. *Computer Graphics (Proc. SIGGRAPH '99)* (1999), 417–424.

[23] Salisbury, M. P., Wong, M. T., Hughes, J. F., and Salesin, D. H. Orientable textures for image-based pen-and-ink illustration. *Computer Graphics (Proc. SIGGRAPH '97)* (1997), 401–406.

[24] Shapiro, L. G., and Stockman, G. C. *Computer Vision.* Prentice-Hall, Upper Saddle River, N.J., 2001.

[25] Silvers, R. *Photomosaics.* Owl Books / Henry Holt and Co., Michael Hawley, New York, 1997.

[26] Vance, P., and Goodrick-Clarke, C. *The Mosaic Book.* Conran Octopus Limited, London, 1994.

[27] Winkenbach, G., and Salesin, D. H. Rendering parametric surfaces in pen and ink. *Computer Graphics (Proc. SIGGRAPH '96)* (1996), 469–476.

[28] Woo, M., Neider, J., Davis, T., and Shriener, D. *Open GL Programming Guide, Third Edition, The Official Guide to Learning Open GL Version 1.2.* Addison Wesley, Massachusetts, 1999.

[29] Y. Pnueli, A. B. Digidürer – a digital engraving system. *Visual Computer 10* (1994), 277–292.