Rendering Traditional Mosaics

Article *in* The Visual Computer · March 2003

DOI: 10.1007/s00371-002-0175-x · Source: DBLP

2 authors:

Gershon Elber
Technion - Israel Institute of Technology
283 PUBLICATIONS **5,595** CITATIONS

SEE PROFILE

George Wolberg
City College of New York
61 PUBLICATIONS **4,538** CITATIONS

SEE PROFILE

The Visual Computer

Rendering traditional mosaics

Gershon Elber[1],
George Wolberg[2]

[1] Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel
E-mail: gershon@cs.technion.ac.il
[2] Department of Computer Science, City College of New York/CUNY, New York, NY 10031
E-mail: wolberg@cs.ccny.cuny.edu

Published online: 28 January 2003

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 free-form 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.

Key words: 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 art forms, including engraving [22, 29], pen-and-ink illustrations [23, 27], digital watercolors [3], line-art drawing [6, 7, 15], expressive painting [11, 19], and Celtic art [10]. This paper addresses computer techniques to render one of the most ancient of classical art forms: 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 art forms. 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 [4].

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. Interested readers may consult [1, 2, 17, 26] for further background.

Mosaics derive much of their splendor from scale. Upon close scrutiny, the skillful placement of tiles and the intricate tessellations 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 general-

The Visual Computer (2003) 19:67–78
Digital Object Identifier (DOI) 10.1007/s00371-002-0175-x

izations 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 art form. 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. 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, namely, the 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. An Adobe Photoshop plug-in attempts to achieve this effect by tessellating 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; that is, 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 presents the algorithm, including the extraction of feature curves, trimmed offset curves, and tile placement. Examples are given in Sect. 3. Finally, Sect. 4 discusses conclusions and future work.

2 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 Sect. 2.1.
- Computing the offsets of these feature curves for all the rows of tiles that are expected to follow this

feature curve. The offset computation procedure is presented in Sect. 2.2.
- 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 Sect. 2.3.
- The placement of the tiles of the mosaics. This process is presented in Sect. 2.4.

2.1 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 free-form 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.

2.2 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 $C_i^j(t), 0 \le j \le m-1$ 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

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

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

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

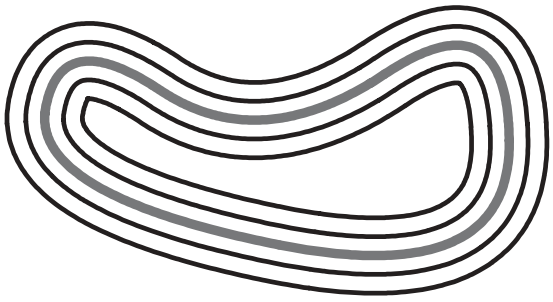


Fig. 1. Parallel curves to a given curve (in *gray*) can be computed as offsets of the given curve

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,

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

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:

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

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 have been derived for rational offsets of rational

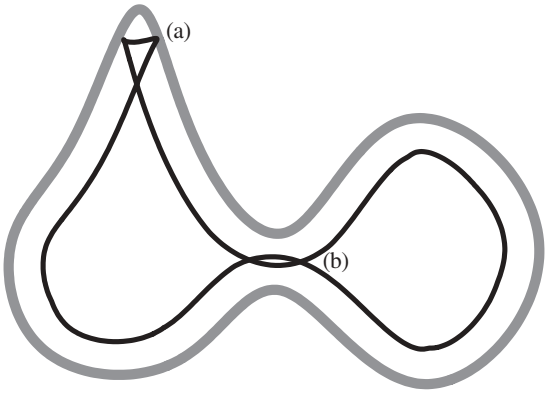


Fig. 2. Local (a) and global (b) self-intersections in an offset of a free-form curve

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

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

2.3 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.

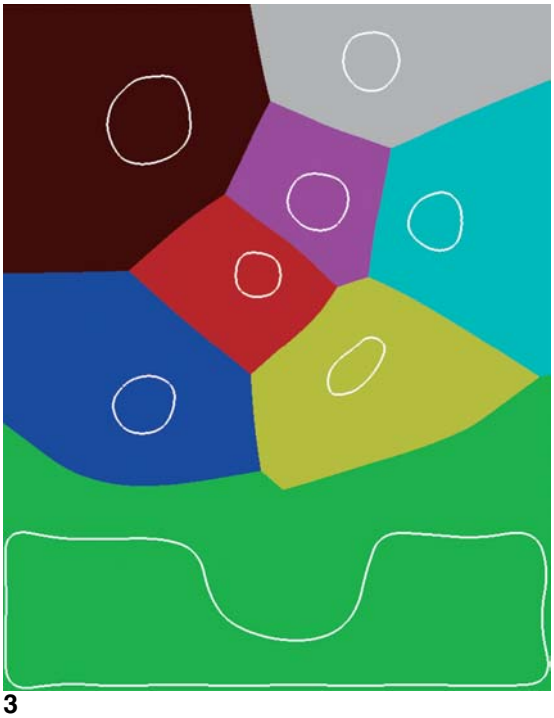
As demonstrated in Sect. 2.2, two reasons could require the trimming of the offset curve: a local and a global self-intersection. Due to the distinctive intrinsic curvature property of local self-intersections, they are fairly simple to detect [8]. In contrast, global self-intersections are far more 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 computation of Voronoi diagrams to robustly trim the offset curves, benefiting from the inherent robustness of the Z-buffer paradigm.

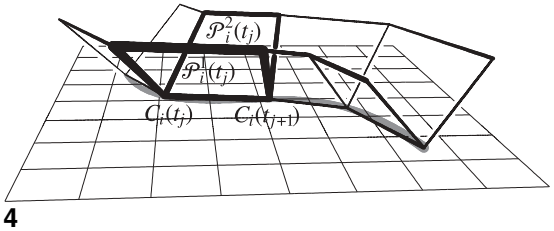
Interestingly enough, the application of mosaics is discussed in [11] as well as in [13], employing the Voronoi cells as the tiles of the mosaics. A Voronoi cell of planar entity $C_i(t)$ contains all the pixels that are closer to $C_i(t)$ than to any other planar entity $C_j(t)$, $j \neq i$. In [11, 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



3



4

Fig. 3. A Voronoi diagram could be approximated discretely, using a Z-buffer. Feature curves of the sunflower drawing shown in Fig. 11

Fig. 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 $C_i(t)$

the one presented in Fig. 4. Hence, the process of sweeping a cone along a curve can be approximated by linear segments and two quadrilateral polygons per segment. The two polygons, $\mathcal{P}_i^1(t_j)$ and $\mathcal{P}_i^2(t_j)$, form the ravine that emanates from the curve segment $(C_i(t_j), C_i(t_{j+1}))$. They each fan out of the curve at a 45-degree angle. Again, see Fig. 4.

We are now ready to adapt the discrete, Z-buffer, 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 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, 13, 14].

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, $\mathcal{C}_i(t_j)$, such that

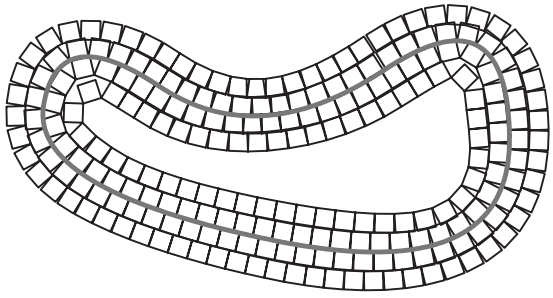
$$\mathcal{C}_i(t_j) = \mathcal{C}_i(0) + \sum_{k=1}^{j} \|C_i(t_k) - C_i(t_{k-1})\|,$$

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. 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, 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.

Figure 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.

5



6

Fig. 5. The extended Voronoi diagram that assigns each pixel along the path a unique color. Compare with Fig. 3
Fig. 6. Tile placement for the curve in Fig. 1

2.4 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 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 Sect. 3.

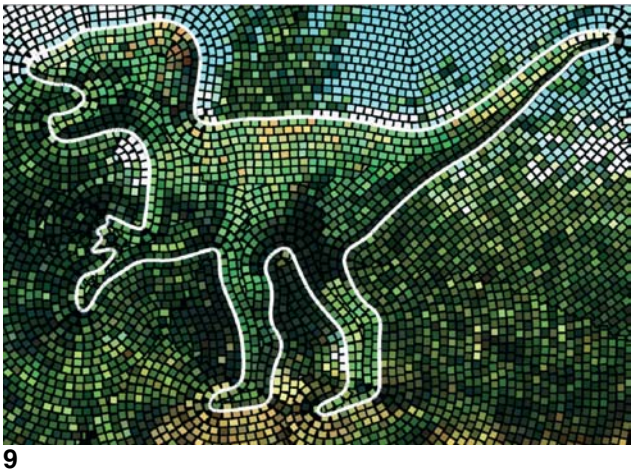
Superior results can be obtained by randomly allowing the penetration of tiles from one front into its adjacent front, in an attempt to alleviate some of these artifact. Examples of this approach, as well as others, can be found in the next section.

3 Examples

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

7

8

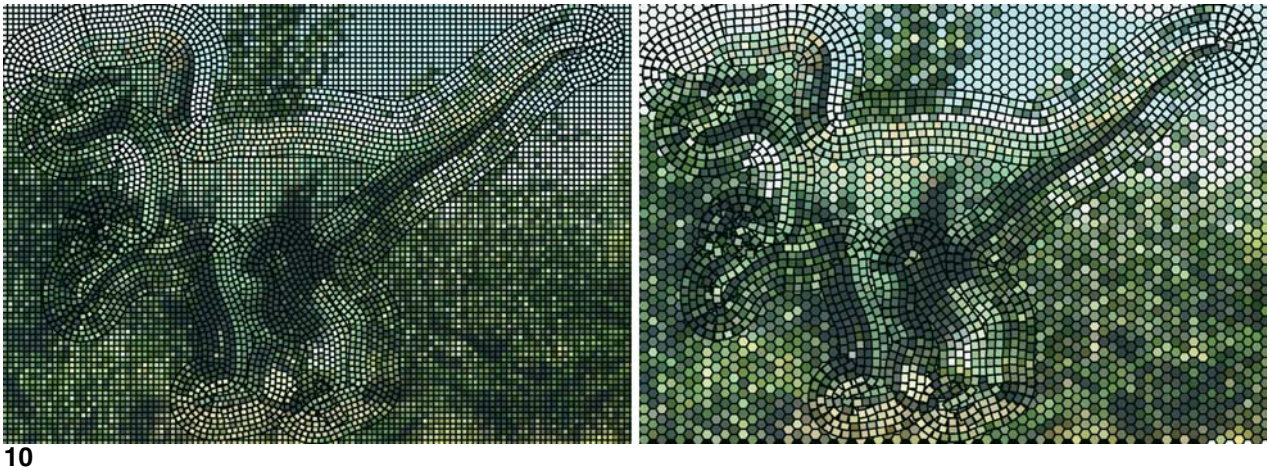


9

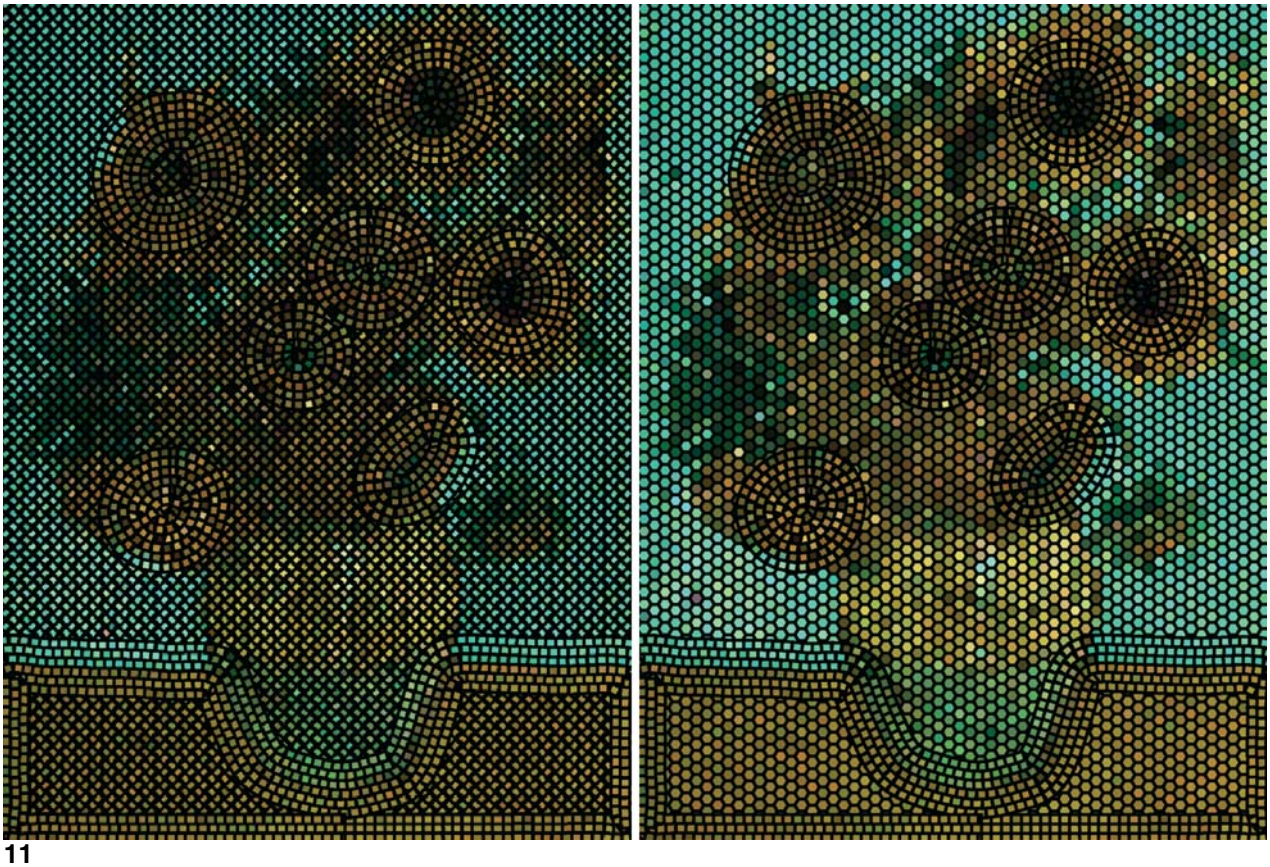
Fig. 7. An image of a dinosaur (*left*) and a mosaic reconstruction (*right*). Image from the Utah dinosaur museum in Ogden
Fig. 8. An image of a dinosaur (*left*) and a mosaic reconstruction (*right*). Image from the Utah dinosaur museum in Ogden
Fig. 9. The feature curves used to place the tiles for Fig. 8

and hexagonal tiles. Note that the feature curves for Fig. 11 were shown in Fig. 3. The seven flowers in Fig. 11 correspond to the seven oval shapes in Fig. 3.

The tile size may vary across the mosaic. Figure 11a depicts an example in which the mosaic tiles grow larger as the tiles move away from the feature curves. The same mosaic pattern is rendered with less grout



10

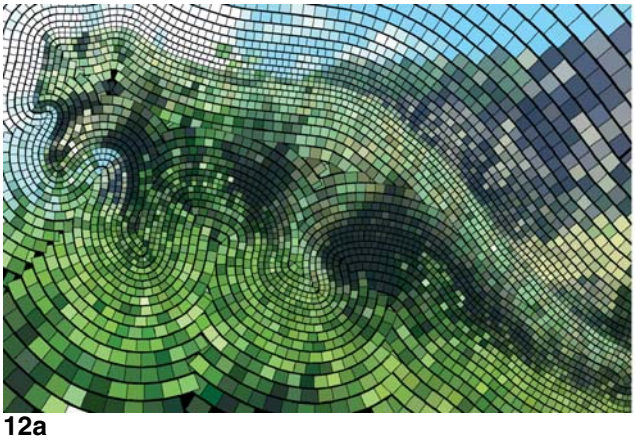


11

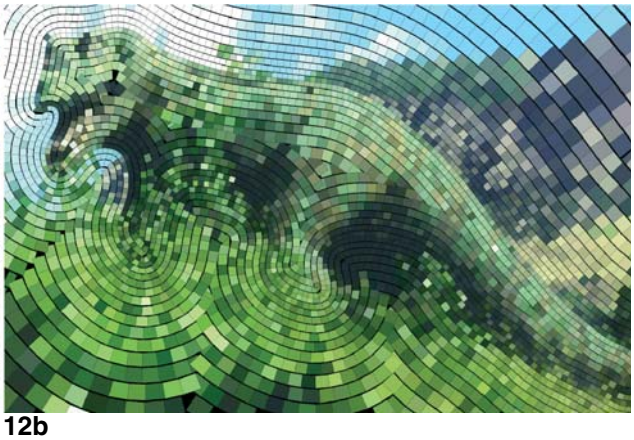
Fig. 10. Placement of few rows of tiles along feature curves. Small tiles and square background tiles (*left*) and large tiles and hexagonal background tiles (*right*). Compare with Fig. 8

Fig. 11. Placement of few rows of tiles along feature curves. Diamond background tiles (*left*) and hexagonal background tiles (*right*)

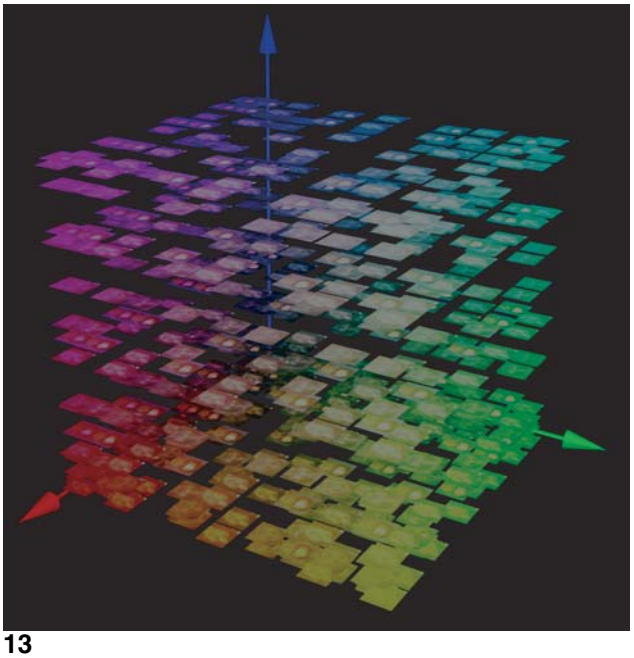
spacing in Fig. 11b. The effect of grout on overall color perception is an important issue that remains to be investigated. Currently, the user must apply artistic discretion when specifying the grout size and texture. Tiles need not necessarily be of uniform color. Figure 14 is a preliminary attempt to employ the bank of images, shown in Fig. 13, to tile a mosaic. In Fig. 14a, a mosaic similar to that of Fig. 8 is shown



12a



12b



13

Fig. 12. Spatially varying tile size: **a** moderate grout spacing; and **b** minimal grout spacing

Fig. 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

except that the tiles are now actual images. Figure 14b shows a close-up of the photomosaic of the back of the dinosaur head.

4 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 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.

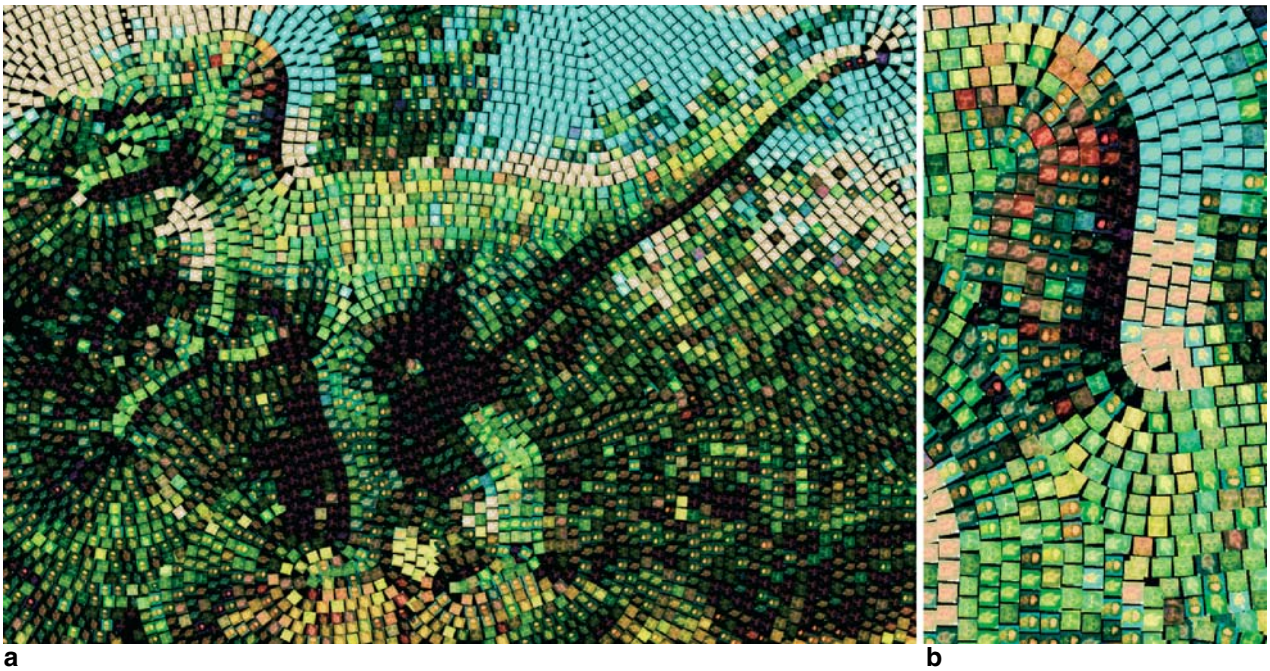


Fig. 14. Generalized photomosaic of dinosaur. Also see Figs. 8 and 10

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 that is a function of the distance to the skeleton, thereby preserving the overall precise 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.

It is also possible to scan a set of mosaic tiles to create a library of interesting tile textures. Then, the underlying color of the digital image in the tile neighborhood can be applied to tile textures randomly selected from the library to produce a visually interesting effect.

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 2-D 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 (1993) Les mosaiques. Bordas, Paris
2. Chavarria J (1999) The art of mosaics. Watson-Guptill, New York
3. Curtis CJ, Anderson SE, Seims JE, Fleischer KW, Salesin DH (1997) Computer-generated watercolor. In: Proceedings of SIGGRAPH '97. ACM SIGGRAPH, New York, pp 421–430
4. Dierks L (1997) Making mosaics Sterling, New York
5. do Carmo MP (1990) Differential geometry of curves and surfaces, 2nd edn. Academic Press
6. Elber G (1995) Line art rendering via a coverage of isoparametric curves. IEEE Trans Vis Comput Graph 1(3):231–239
7. Elber G (1998) Line art illustrations of parametric and implicit forms. IEEE Trans Vis Comput Graph 4(1):71–81
8. Elber G, Cohen E (1991) Error bounded variable distance offset operator for free form curves and surfaces. Int J Comput Geom Appl 1(1):67–78
9. Elber G, Lee I-K, Kim M-S (1997) Comparing offset curve approximation methods. IEEE Comput Graph Appl 17(3):62–71
10. Glassner A (1999) Celtic knotwork. IEEE Comput Graph Appl 19(5):78–84
11. Haeberli P (1990) Paint by numbers: abstract image representations. Comput Graph (Proceedings of SIGGRAPH '90) 24(4):207–214
12. Hausner A (2001) Simulating decorative mosaics. In: Proceedings of ACM SIGGRAPH 2001. ACM SIGGRAPH, New York, pp 573–580
13. Hoff KE, Culver T, Keyser J, Lin M, Manocha D (1999) Fast computation of generalized voronoi diagrams using graphics hardware. In: Proceedings of SIGGRAPH '99. ACM SIGGRAPH, New York, pp 277–286
14. Hoffmann C (1992) Computer vision, descriptive geometry, and classical mechanics. In: Falcidieno R, Herman I, Pienovi C (eds) Computer graphics and mathematics. Springer, Berlin Heidelberg New York, pp 229–243; Also available as Technical Report, CSD-TR-91-073. Computer Science Department, Purdue Univeristy, October, 1991
15. Lansdown J, Schofield S (1995) Expressive rendering: a review of nonphotorealistic techniques. IEEE Comput Graph Appl 15(3):29–37.
16. Lee S-Y, Chwa K-Y, Shin SY, Wolberg G (1995) Image metamorphosis using snakes and free-form deformations. In: Proceedings of SIGGRAPH '95. ACM SIGGRAPH, New York, pp 439–448
17. Ling R (1998) Ancient mosaics. Princeton University Press, Princeton, N.J.
18. Mantanari U (1968) A method for obtaining skeletons using a quasi-euclidean distance. J Comput Mach 16(4):534–549
19. Meier B (1996) Painterly rendering for animation. In: Proceedings of SIGGRAPH '96. ACM SIGGRAPH, New York, pp 477–484
20. Miyata K (1990) A method of generating stone wall patterns. Comput Graph (Proceedings of SIGGRAPH '90) 24(4):387–394
21. Mortensen EN, Barrett WA (1995) Intelligent scissors for image composition. In: Proceedings of SIGGRAPH '95. ACM SIGGRAPH, New York, pp 191–198
22. Ostromoukhov V (1999) Digital facial engraving. In: Proceedings of SIGGRAPH '99. ACM SIGGRAPH, New York, pp 417–424
23. Salisbury MP, Wong MT, Hughes JF, Salesin DH (1997) Orientable textures for image-based pen-and-ink illustration. In: Proceedings of SIGGRAPH '97. ACM SIGGRAPH, New York, pp 401–406
24. Shapiro LG, Stockman GC (2001) Computer vision. Prentice-Hall, Upper Saddle River, N.J.
25. Silvers R, Hawley M (eds) (1997) Photomosaics. Henry Holt, New York
26. Vance P, Goodrick-Clarke C (1994) The mosaic book. Conran Octopus, London
27. Winkenbach G, Salesin DH (1996) Rendering parametric surfaces in pen and ink. In: Proceedings of SIGGRAPH '96. ACM SIGGRAPH, New York, pp 469–476
28. Woo M, Neider J, Davis T, Shriener D (1999) OpenGL programming guide, 3rd edn. The official guide to learning OpenGL version 1.2. Addison Wesley, Mass.
29. Pnueli Y, Bruckstein AM (1994) Digidürer – a digital engraving system. Vis Comput 10:277–292

Photographs of the authors and their biographies are given on the next page.

GERSHON ELBER is an associate professor in the Computer Science Department, Technion, Israel. His research interests span computer-aided geometric designs and computer graphics. Professor Elber received a BS in computer engineering and an MS in computer science from the Technion in 1986 and 1987 respectively, and a PhD in computer science from the University of Utah, US, in 1992. He is a member of the ACM and IEEE. Professor Elber can be reached at the Technion, Israel Institute of Technology, Department of Computer Science, Haifa 32000, ISRAEL.

GEORGE WOLBERG is a full professor of computer science at the City College of New York. He received his BS and MS degrees in electrical engineering from Cooper Union in 1985, and his PhD degree in computer science from Columbia University in 1990. He has published over 30 research papers in image processing, computer graphics, and computer vision, and holds two US patents. He is the recipient of a 1991 NSF Presidential Young Investigator Award, the 1997 CCNY Outstanding Teaching Award, a 1998 NASA Faculty Award for Research, and the 2000 Mayor's Award for Excellence in Science and Technology (awarded by Mayor Giuliani). Professor Wolberg is the author of *Digital Image Warping* (IEEE Computer Society Press, Los Alamitos, Calif., 1990), the first comprehensive monograph on warping and morphing. He is a senior member of the IEEE.