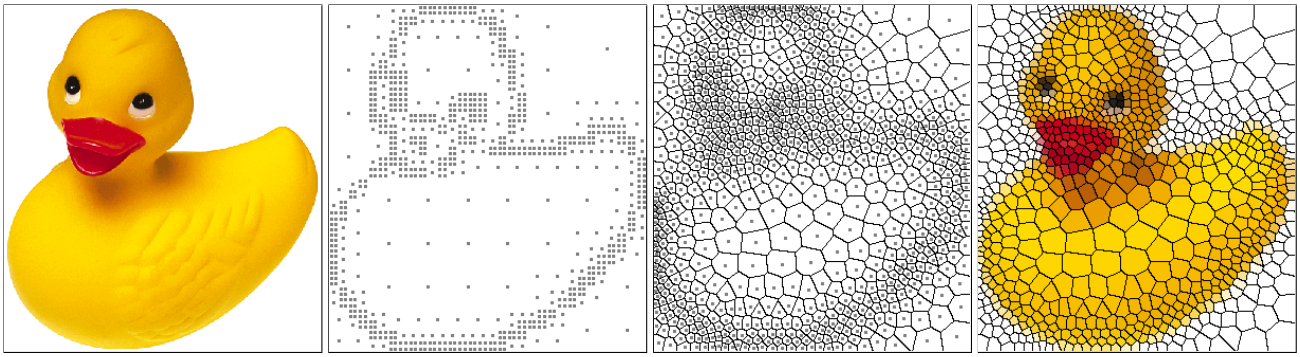
Simple Adaptive Mosaic Effects

Geisa Martins Faustino Luiz Henrique de Figueiredo
IMPA–Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina 110, 22460-320 Rio de Janeiro, RJ, Brazil



Abstract

We present an algorithm that creates a mosaic effect for an image in an adaptive and automatic fashion. The algorithm is automatic because it does not need user intervention, except for the choice of a couple of parameters. The algorithm is adaptive because it creates tiles whose sizes are adapted to the features of the image. This is achieved by using a centroidal Voronoi diagram with a density function that emphasizes image features.

1 Introduction

Mosaic effects add artistic touches to existing images in a way that resembles ancient mosaics or stained-glass windows. These effects are obtained by decomposing the original image into cells (also called *tiles*) and painting those cells with a color that approximates the color distribution inside the cell in the original image. The goal is to generate a simplified image that still gives the overall impression of the original image.

This paper describes an algorithm that creates mosaic effects automatically, with little user intervention, generating an image that is adapted to the features in the original image. Like some of the previous approaches, we also use centroidal Voronoi diagrams [4]. However, we start from an adaptive sampling of the image and use a density function that emphasizes image features. The overall effect is that the sizes of the mosaic tiles are adapted to the features of the image, being large in regions of less detail and small in regions of great detail. This is our main contribution.

The banner figure above shows the main steps in our approach. A given image is sampled adaptively by placing points near its features. Then a centroidal Voronoi diagram is computed from these sample points. Finally, the Voronoi cells are painted with an adequate color or texture.

2 Previous work

The earliest work on mosaic effects is probably the one by Haeberli [7]. That paper described a general technique for painting several effects interactively over an image, and one of the examples was exactly a mosaic effect using Voronoi diagrams. Since that was an interactive painting system, the artist was in direct control of the amount of detail revealed in the mosaic. Haeberli also mentioned briefly that an iterative relaxation technique could be used to create those effects automatically, but he gave little detail on that technique. Moreover, the images created in this fashion were said to take “several hours” to compute.

Hoff et al. [9] described a fast, hardware-based, technique for computing discrete Voronoi diagrams. One application they describe briefly is mosaic effects, which they obtain from Voronoi diagrams of random sites on the image. Because the Voronoi diagram is computed by the graphics hardware, it can be done in real time, even for a large number of sites. This allows the artist to tune the mosaic effect

interactively. Hoff et al. also mentioned that mosaics resembling real-life mosaics could be obtained by adapting the sites to the features of the image, but gave no details on how these sites could be chosen.

Hausner [8] extended the approach of Hoff et al. [9] to compute centroidal Voronoi diagrams that are aligned to the edges of the image. He also replaced Euclidean distance by the Manhattan distance. This allows mosaics to be made of square tiles, giving a nice artistic effect similar to real mosaics. Although Hausner briefly explained how to make the size of the tiles vary according to the features of the images, his method still requires that artists select the edges they want to emphasize in the mosaic. (This requirement may be a feature, from the point of view of the artist.)

Elber and Wolberg [5] extended the work of Hausner [8] by trying to mimic the placement of tiles done by artists: tiles are traditionally placed not only near selected image edges but also along curves parallel to them. Their approach is thus geometrical and relies on the computation and trimming of offset curves. Voronoi diagrams are used only to assist in the trimming phase. The result are mosaic effects that have many features of real mosaics, even though offset curves impose a regular patterns in the final image.

The work of Hausner [8] and Elber and Wolberg [5] still represent the best attempts at simulating real mosaics with computer graphics. Another approach to mosaic effects is to try to reproduce stained-glass windows. This effect also decomposes the original image into tiles, but they are painted in a different way because the goal is to evoke the visual effects of the sunlight onto stained-glass windows. Along that line lie the work of Dobashi et al. [3] and Mould [11], which we now discuss.

Dobashi et al. [3] proposed a method for creating mosaic effects that uses ordinary Voronoi diagrams and renders them in a stained-glass fashion. The method starts with a Voronoi diagram for sites chosen as slight perturbations of the centers of a regular hexagonal mesh. This initial Voronoi diagram is thus approximately centroidal. Then the sites are moved to try to capture the global features of the original image and the Voronoi diagram is recomputed. The result is a Voronoi diagram whose edges are aligned to the main edges of the original image. Two visual effects are used: Voronoi edges are painted in wide lines whose width depend on the color difference between adjacent regions; and Voronoi regions are painted using Gouraud shading based on perturbed color values at the vertices of each region.

Mould [11] described a complete algorithm that creates a stained-glass effect on a given image. The image is first segmented automatically and then regions appropriate for tiles are extracted from this segmentation using mathematical morphology. Care is taken to avoid undesirable shapes. Finally, the regions are colored trying to mimic the coloring used in medieval times. This implies the use of a limited palette. The rendering is completed by using large displacement maps near the edges to mimic leading and small displacement maps in tile interiors to mimic glass imperfections. This method works very well for images that have clear silhouettes, producing impressive stained-glass renditions. The method does not work so well for images that are difficult to segment, such as those whose visual clues reside mainly in subtle shading.

The work on mosaic effects described above should not be confused with the work on image mosaics or mosaicing, whose goal is to try to mimic the original image by glueing many small images selected from a database [6, 10].

3 Our approach

Our approach is closer to that of Hausner [8] and Dobashi et al. [3] in that we compute Voronoi diagrams starting from an initial set of sites which are then adjusted to the image features. Also, like Dobashi et al. [3], our final images display a stained-glass effect. However, our approach differs from previous approaches in that we compute centroidal Voronoi diagrams for a density function that directly reflects image features. This makes the method both automatic and adaptive.

Our method has three main steps, which we shall discuss in detail in the sequel:

1. Sample the image adaptively, finding a number of seed points.
2. Compute the centroidal Voronoi diagram of the seeds, using a density map computed from the original image.
3. Paint each Voronoi cell.

3.1 Sampling the image

The goal of this step is to find a set of *seeds* from which we shall compute a centroidal Voronoi diagram.

To find the seeds, we sample the image adaptively using a quadtree. The seeds are the centers of the leaf cells in this quadtree. A cell is a leaf in the quadtree when the color of all its pixels are close to the average color in the cell. More precisely, for each cell C, we test whether

$$\max_{p \in C} d(I(p), c)^2 \leq \varepsilon,$$

where $I(p)$ is the color of the pixel $p \in C$, c is the average color in C (obtained by averaging the red, blue, and green components separately), and ε is a user-selected tolerance. The distance d between two colors is just the Euclidean distance in RGB space. To avoid getting to single-pixel cells, we also stop the subdivision when cells get too small (for instance, when they have less than 64 pixels).

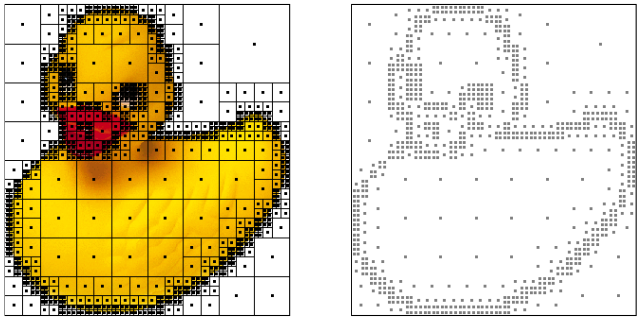


Figure 1. Quadtree decomposition (left) and sample points (right).

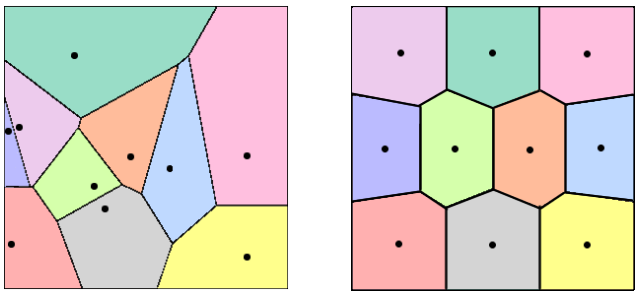


Figure 2. Voronoi diagram (left) and centroidal Voronoi diagram (right) [4].

This simple sampling method generates two sets of points: points that are clustered around the image edges, and points in the middle of regions of low detail. Both types of points are needed for a fair sampling of the image. Figure 1 shows the quadtree and the sample points for the duck image shown in the introduction.

Because the sampling is just meant to find starting sites for computing the centroidal Voronoi diagram, this simple sampling method is quite adequate. More sophisticated sampling methods might sample the image better, but would not necessarily have much influence in the final result. We have not pursued this line of research.

3.2 Finding the cells

Starting from the seeds found in the sampling step, we compute a centroidal Voronoi diagram.

The *Voronoi diagram* of a set of points, called *sites*, is a decomposition of the space into cells, one cell for each site. The cell corresponding to a site p is the set of all points in space whose closest site is p. For a survey of Voronoi diagrams, see [1, 12]. A *centroidal Voronoi diagram* is a Voronoi diagram in which each Voronoi site is the centroid of its Voronoi cell. For a survey of centroidal Voronoi diagrams, see [4]. Figure 2 shows a Voronoi diagram and a centroidal Voronoi diagram. Note the regular geometry of the cells in the centroidal Voronoi diagram.

Centroidal Voronoi diagrams are very rare. However, any Voronoi diagram can be transformed into a centroidal Voronoi diagram by the following simple iterative procedure: replace each Voronoi site by the centroid of its Voronoi cell, recompute the Voronoi diagram for the new sites, and repeat. This procedure is known as *Lloyd's algorithm*. Formally, this algorithm is only known to converge in dimension 1 and then only for constant density functions. In practice, it performs well in the general case, even if it is a bit slow to converge. (However, as we shall see, we do not need to reach convergence.)

As hinted above, a major ingredient in definition of a centroidal Voronoi diagram is an underlying *density function*. The centroids of the Voronoi cells are computed using a given density function. Formally, the centroid of a region V according to a density function μ is the point z given by

$$z = \frac{\int_V x\mu(x)\,dx}{\int_V \mu(x)\,dx}.$$

Note that the density function μ enters only in the computation of the centroid. It does not enter in the computation of the Voronoi diagram, which is still computed using the Euclidean metric.

The power of centroidal Voronoi diagrams in applications lies in choosing a suitable density function. Centroidal Voronoi diagrams adapt themselves to the mass distribution implied by the density function, having larger cells where the density is low and smaller cells where the density is high. (See Figure 3.) This is exactly the adaptive feature that we look for in our mosaic effects. It is also very useful in other applications, such as mesh generation. In this case, the fact that a centroidal Voronoi diagram is still an ordinary Voronoi diagram allows meshes to be based on the Delaunay triangulation, which has many nice properties. Centroidal Voronoi diagrams have recently attracted much attention for graphics applications.

The density function that we use for creating mosaic effects in images is the norm of the gradient of the luminance of the image. In other words, we convert the image to gray and compute the Euclidean norm of the gradient for the gray image. The gradient is computed using central differences. Computing image gradients is a traditional tool in image segmentation, and so it is not surprising that it captures image features well. Figure 4 shows the gradient image of the duck image. The darker the pixel, the higher the density.

Using the gradient of the image as density function forces the corresponding centroidal Voronoi diagram to adapt itself locally to the edges of the image. This helps to

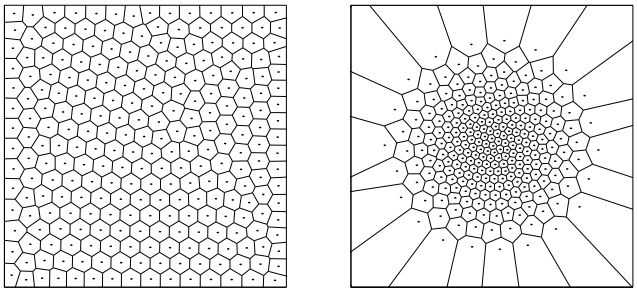


Figure 3. Centroidal Voronoi diagram for constant density (right) and variable density (left) [4].

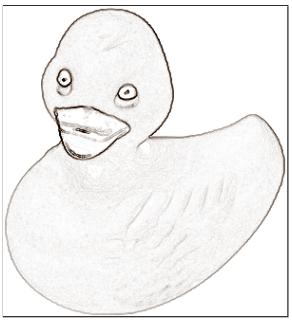


Figure 4. Image gradient as density function.

preserve the main visual features of the original image in the final mosaic. Other density functions can be used, depending on what the artist wishes to emphasize in the mosaic. Research along this line is left for future work.

Figure 5 (left) shows the Voronoi diagram of the initial seed points. Note how the quadtree sampling imposes a regular structure on the Voronoi diagram. Figure 5 (right) shows the corresponding centroidal Voronoi diagram, computed after 10 iterations of Lloyd's algorithm. Note how the cells have nicer geometry and are better distributed according to the features of the image.

The centroid z of a region V in the image is given by

$$z = \frac{\sum_{x \in V} x\mu(x)}{\sum_{x \in V} \mu(x)},$$

where the sums run over all pixels x in V. Of course, these sums are the discrete analogues of the integrals above.

Note that the centroid is probably not at a pixel location. In other words, although the pixels x in V have integer coordinates, the centroid of V must be allowed to have fractional coordinates. This is an important detail: truncating or rounding the position of the centroids to integers interferes with the convergence of Lloyd's algorithm and may generate ugly visual effects.

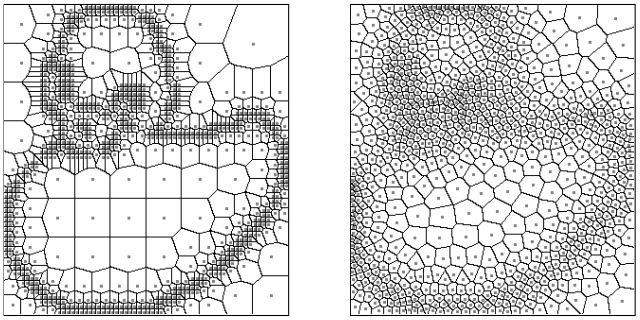


Figure 5. Initial Voronoi diagram (left) and centroidal Voronoi diagram (right).

Although we could have used the graphics hardware for computing Voronoi diagrams, as Hoff et al. [9] did, we have opted to compute them directly, by finding which site is closest to each pixel. Internally, the Voronoi diagram is represented as a matrix that labels each pixel with the index of the site closest to it. (Ties are broken arbitrarily.) This method is straightforward to implement and not too slow in current machines.

One consequence of representing the Voronoi diagram by a label matrix is that the Voronoi regions required for computing the centroids are not explicitly represented; they have to be extracted from the label matrix. Nevertheless, it is simple to compute the centroids without explicitly extracting the Voronoi regions: A single pass through the label matrix allows us to compute all the required sums simultaneously, by incrementally updating the sum corresponding to each entry in the label matrix.

3.3 Painting the cells

Once the image has been decomposed into tiles by a suitable centroidal Voronoi diagram, we have to paint the tiles and the edges. There are several alternatives for this.

The simplest alternative is to color a tile with the color of the corresponding site in the image. Because, as discussed before, the site is probably not exactly at a pixel location, we have to use an approximate color. The simplest approximation is the color of the pixel closest to the site.

A better alternative, which is equally simple to implement, is to color a tile with the average color of the image in the tile. This gives a better visual result. However, this technique naturally introduces a blur effect near image edges, specially in images that have a solid background (such as the duck). The effect is much less pronounced in images with complex backgrounds. The reason for this blur effect is that Voronoi cells tend to straddle image edges. If the effect is annoying, it can probably be removed a post-processing step, but we have not pursued this.

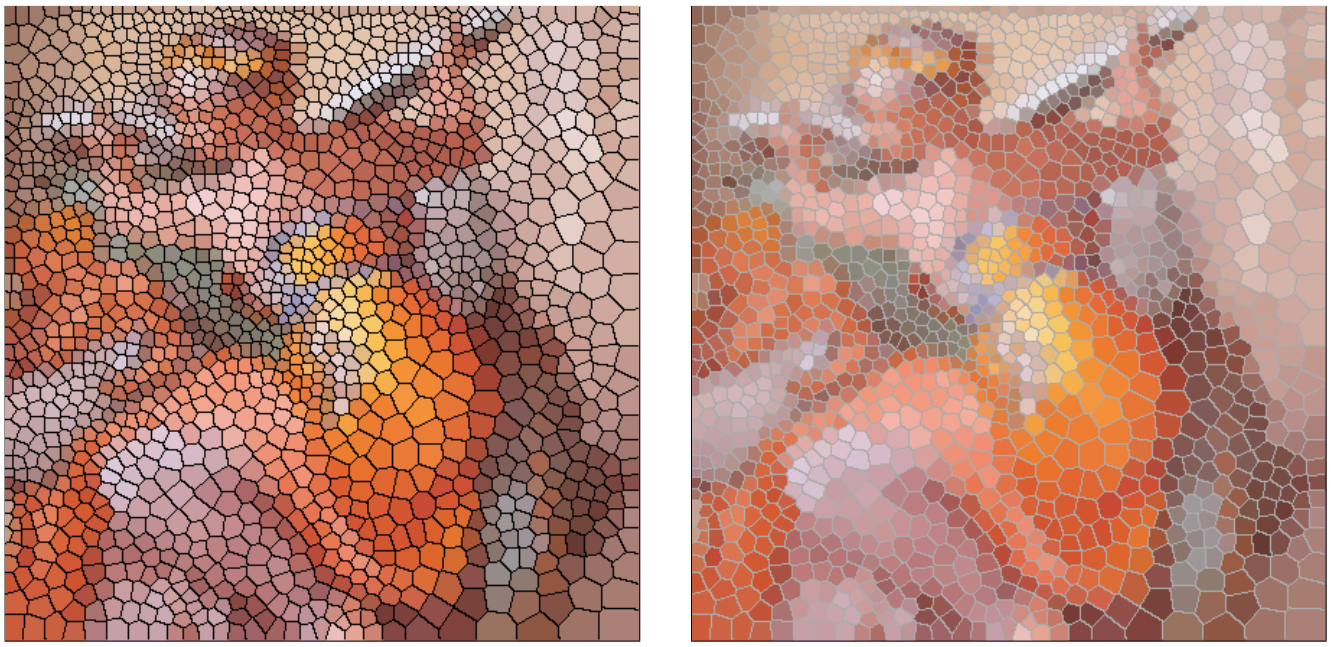


Figure 6. Mosaic effect obtained from painting Voronoi edges with a background color.

We find the Voronoi edges directly from the label matrix, using simple edge detection: pixels whose neighbors have different labels are on an edge. We only check the neighbors below and to the right of a pixel; this ensures that the Voronoi edges are one pixel wide.

Painting the edges with black gives a stained-glass effect. Painting the edges with a light color close to overall background color of the original image gives a mosaic effect. See Figure 6.

There are many other ways to add artistic touches in the rendering step. A thorough investigation of the artistic possibilities was not the main focus of our work; we describe here just a few simple techniques.

A simple way to add realism to stained-glass effects is to add glass texture by copying the attenuation factor from a photograph or high-quality rendering of an illuminated glass window. This is similar to the displacement maps used by Mould [11], but is more realistic (though perhaps not as interesting artistically). On the other hand, the result mosaic may appear to be painted onto the glass, instead of being composed of little glass panes. Another way is to add synthetic illumination based on a simulated 3D geometry inside each cell. The challenge here is to choose a nice geometry. Figure 7 shows these two kinds of artistic touches.

4 Results

We tested our algorithm on several images. The results are shown in Figures 8 and 9, which show the original image, a mosaic effect, a painterly effect obtained by not painting Voronoi edges, and a quilt effect obtained by simulating an illuminated round surface over each Voronoi cell. The table below gives the parameters used for creating those images: the tolerance ε and the minimum cell size used in the sampling. It also lists the number of seed points found. In all cases, we ran 10 steps of Lloyd's algorithm.

Image	ε	min cell	seeds
Duck	0.20	81	868
Butterfly	0.45	81	1012
Die	0.15	64	646
Garfield	0.30	81	1618
Flower	0.20	81	1066
Peppers	0.20	81	352
Michelangelo1	0.25	81	2029
Michelangelo2	0.25	81	877
Sibyl	0.12	81	1675
Street	0.25	81	2176

The test images (and others) and additional visual effects are available in full size and color at `http://www.impa.br/~geisamf/sibgrapi2005/`.

5 Conclusion

Our algorithm produces nice mosaic effects with little user intervention. The algorithm is simple to implement and does not require special hardware. Nevertheless, if the artist wants interactive intervention, the algorithm can be

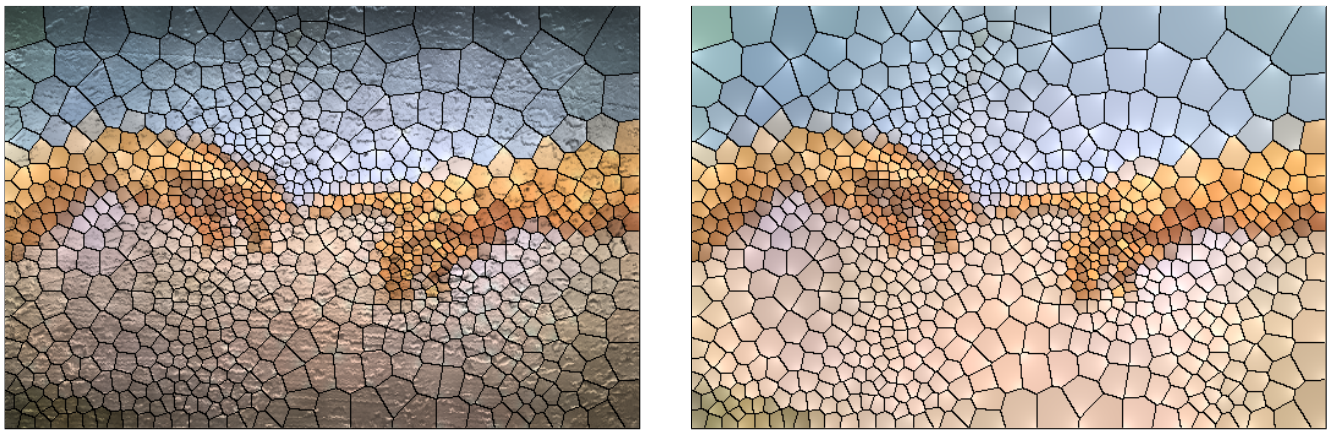


Figure 7. Two stained-glass effects: glass texture (left) and synthetic illumination (right).

modified to use the graphics hardware to compute Voronoi diagrams [9].

There are several questions left open by our approach: Would using a metric in a perceptual color space give a better initial sampling? Does a better initial sampling influence the final result? What other density functions can be used? How does one choose a density function if one wants a different emphasis in the final result? What kind of interactive control is suitable for artists?

We have found that the density function that we chose works only for the local adaption of the Voronoi diagram to the features of the image. This happens apparently because the gradient has a limited range and varies too smoothly. For this reason, we only ran a few steps of Lloyd's algorithm. Waiting for convergence would create an almost uniform Voronoi diagram and destroy the adaptive effect. By contrast, the exponentials used as density functions in [4] seem to provide global adaption.

Moreover, as mentioned in Section 3.3, the Voronoi cells obtained from this local adaption straddle image edges, which may introduce blur for some images. Previous approaches (e.g., Dobashi et al. [3]) have worked hard to try to align Voronoi edges with image edges. In our framework, the natural way for trying to do that would be to find a suitable density function that captures this alignment. We have not pursued this any further, but it remains a very interesting line of research. The work of Hausner [8] is a step in that direction, but he only used constant-density centroidal Voronoi diagrams.

Acknowledgments. G. M. Faustino is partially supported by a CAPES M.Sc. scholarship. L. H. de Figueiredo is partially supported by a CNPq research grant. The authors are members of Visgraf, the computer graphics laboratory at IMPA, which is sponsored by CNPq, FAPERJ, FINEP, and IBM Brasil. The authors thank Luiz Velho for his careful reading and suggestions, and Lucia Darsa and Bruno Costa for the die image from [2].

References

[1] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[2] L. Darsa and B. Costa. Multi-resolution representation and reconstruction of adaptively sampled images. In *Proceedings of SIBGRAPI '96*, pages 321–328, 1996.

[3] Y. Dobashi, T. Haga, H. Johan, and T. Nishita. A method for creating mosaic images using Voronoi diagrams. In *Proceedings of Eurographics 2002 Short Presentations*, pages 341–348, 2002.

[4] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[5] G. Elber and G. Wolberg. Rendering traditional mosaics. *The Visual Computer*, 19(1):67–78, 2003.

[6] A. Finkelstein and M. Range. Image mosaics. In *Proceedings of the 7th International Conference on Electronic Publishing*, Lecture Notes In Computer Science; Vol. 1375, pages 11–22. Springer-Verlag, 1998.

[7] P. Haeberli. Paint by numbers: abstract image representations. In *Proceedings of SIGGRAPH '90*, pages 207–214. ACM Press, 1990.

[8] A. Hausner. Simulating decorative mosaics. In *Proceedings of SIGGRAPH 2001*, pages 573–580. ACM Press, 2001.

[9] I. Kenneth E. Hoff, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of SIGGRAPH '99*, pages 277–286. ACM Press, 1999.

[10] J. Kim and F. Pellacini. Jigsaw image mosaics. In *Proceedings of SIGGRAPH 2002*, pages 657–664. ACM Press, 2002.

[11] D. Mould. A stained glass image filter. In *Proceedings of the 14th Eurographics Workshop on Rendering*, pages 20–25. Eurographics Association, 2003.

[12] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., 1992.

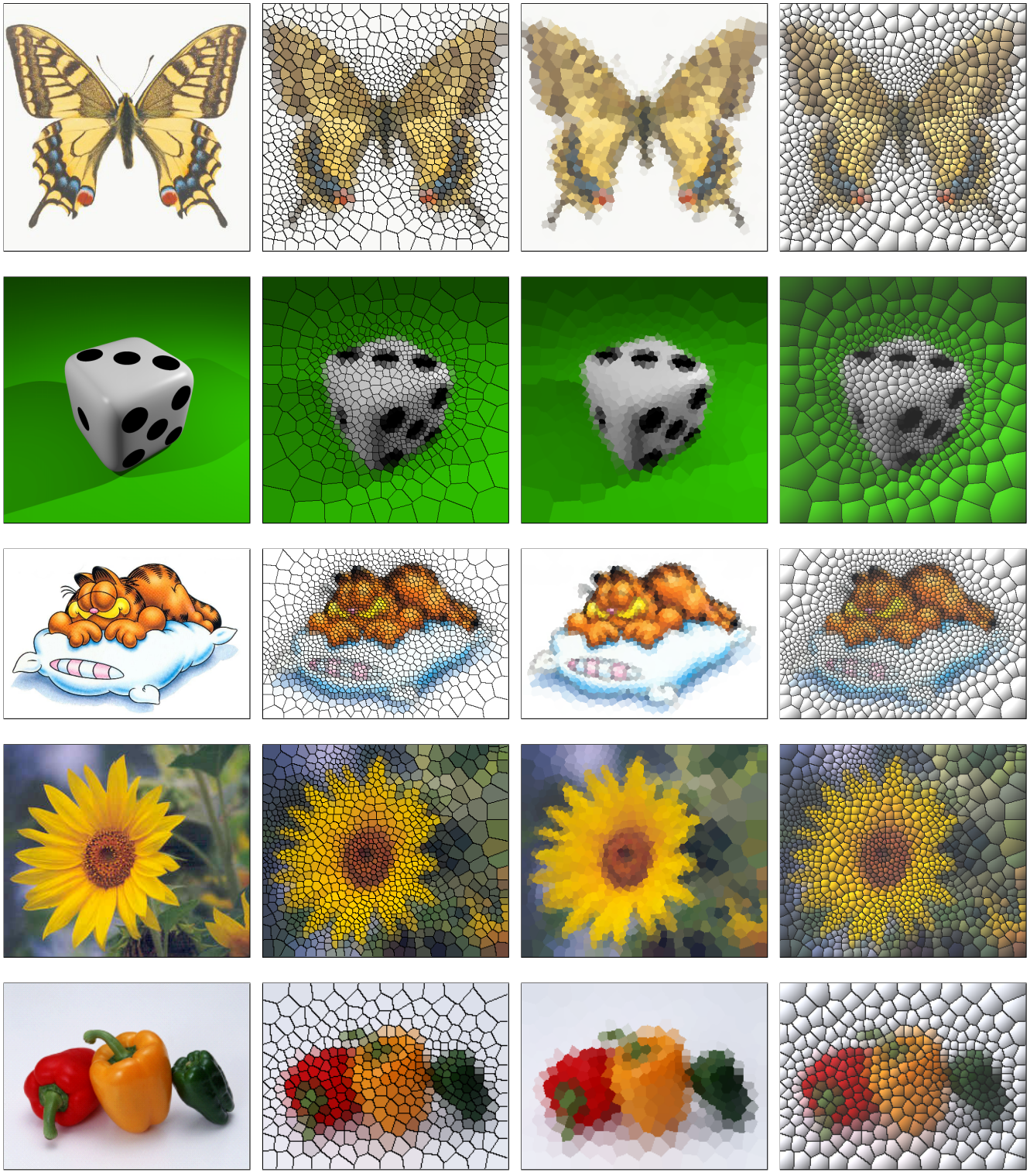


Figure 8. From left to right: original image, mosaic effect, painting effect, quilt effect. From top to bottom: Butterfly, Die, Garfield, Flower, Peppers.

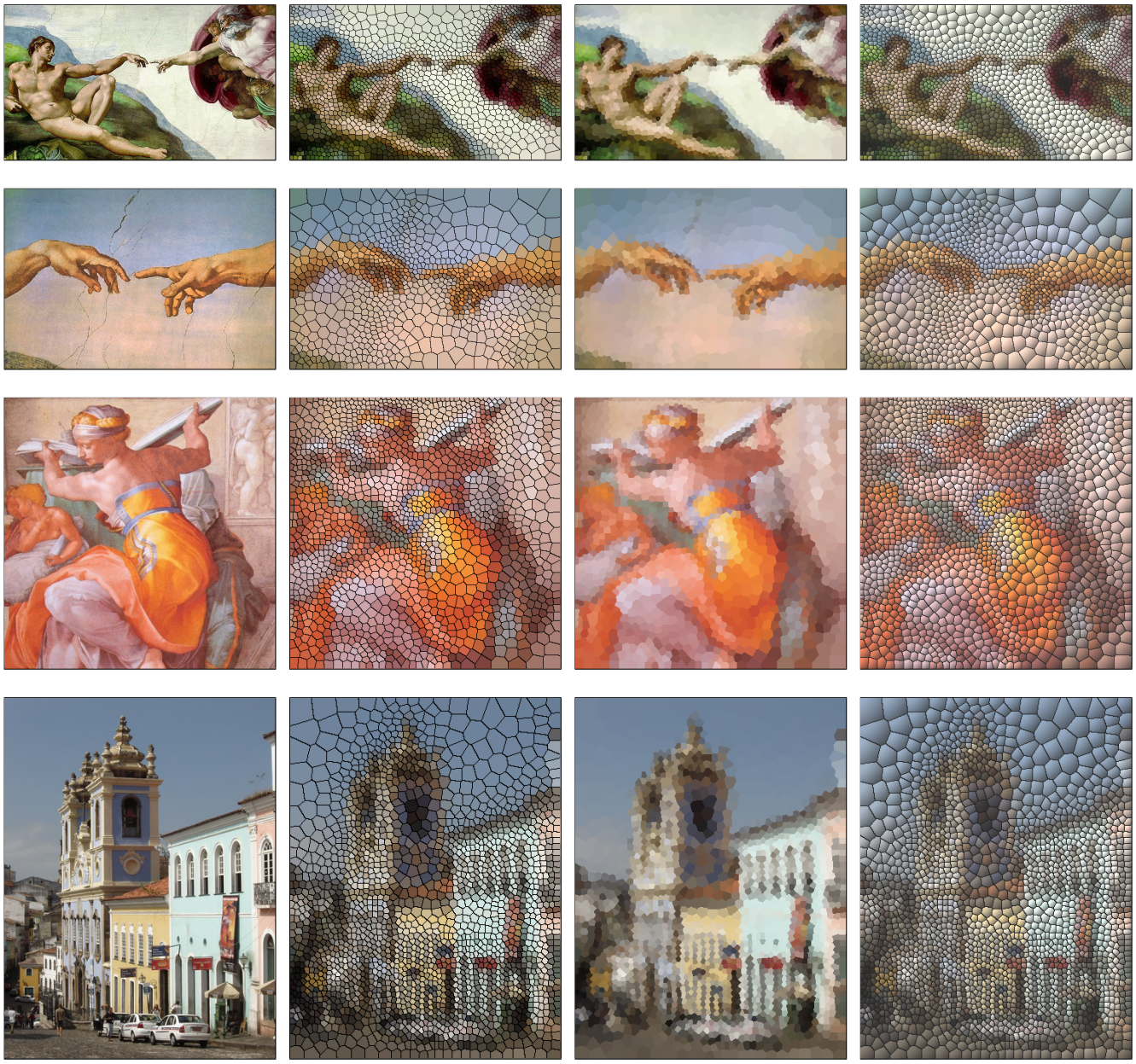


Figure 9. From left to right: original image, mosaic effect, painting effect, quilt effect. From top to bottom: Michelangelo1, Michelangelo2, Sibyl, Street.

Note added in press. We have just come across this paper, which proposes a new approach for simulating real mosaics: G. Di Blasi and G. Gallo, Artificial mosaics, to appear in *The Visual Computer* (doi:10.1007/s00371-005-0292-4).