Computational Aesthetics in Graphics, Visualization and Imaging (2005)
L. Neumann, M. Sbert, B. Gooch, W. Purgathofer (Editors)

Fast Techniques for Mosaic Rendering

G. Di Blasi, G. Gallo, M. Petralia

D.M.I., University of Catania

Abstract

Art often provides valuable hints for technological innovations especially in the field of Image Processing and Computer Graphics. In this paper we survey in an unified framework three methods to transform a raster input image into good quality mosaics: artificial mosaic, photomosaic and puzzle image mosaic. The common and different ideas among these methods are reported. The main goal of all the methods is to produce good results in an acceptable time and without user intervention. Examples reported in the paper show how the right mixture of mathematical tools may lead to impressive results.

Categories and Subject Descriptors (according to ACM CCS): J.5 [ARTS AND HUMANITIES]: Fine arts

1. Introduction

The creation of digital mosaics of artistic quality is one of the challenges of the Computer Graphics and is one of the most recent research directions in the field of Non-Photorealistic Rendering. Digital mosaics are illustrations composed by a collection of small images called "tile". The tiles "tessellate" a source picture in order to reproduce it in a "mosaic-like" style. Starting from the same source image it is possible to create different kind of digital mosaics depending on the choice of the tile dataset and the imposed constraints to positioning and deformations.

The first step to solve the problem of the creation of digital mosaics is to reformulate the problem itself into a mathematical framework. In particular it is possible to put the mosaic construction from a source raster image in terms of a mathematical optimization problem as follows:

> Given a rectangular region I_2 in the plane R_2, a tile dataset and a set of constraints, find N sites $P_i(x_i, y_i)$ in I_2 and place N tiles, one at each P_i, such that all tiles are disjoint, the area they cover is maximized and the constraints are verified as much as possible.

The definition above is general and is suitable for many applications even beyond Computer Graphics field. Within this framework the problem can be viewed as a particular case of the "cover problem" or as a "search and optimization problem". The mosaic construction as formulated above can also be regarded as a "low-energy configuration of particles problem".

In our case three different definitions can be given to solve specific problems:

> *Artificial Mosaic* - Given an image I_2 in the plane R_2 and a vector field $\Phi(x,y)$ defined on that region representing the edges of I_2, find N sites $P_i(x_i, y_i)$ in I_2 and place N rectangles, one at each P_i, oriented with sides parallel to $\Phi(x,y)$, such that all rectangles are disjoint, the area they cover is maximized and each tile is colored by a color which reproduces the image portion covered by the tile.
>
> *Photomosaic* - Given an image I_2 in the plane R_2, a dataset of small rectangular images and a regular rectangular grid of N cells, find N tile images in the dataset and place them in the grid such that each cell is covered by a tile that "reminds" the image portion covered by the tile.
>
> *Puzzle Image Mosaic* - Given an image I_2 in the plane R_2, a dataset of small irregular images and an irregular grid of N cells, find N tile images in the dataset and place them in the grid such that the tiles are disjoint and each cell is covered by a tile that "reminds" the image portion covered by the tile.

Different solutions have been proposed to solve the above problems in particular respectively in [Hau01], [SH97] and [KP02] the proposed solutions lead to good aesthetic

results. Unfortunately the required computation time is often prohibitive and does not allow to develop the above techniques as standard plug-ins in a typical user-end software. Further user intervention is needed to perform the task. For these reasons in this paper we review three recent techniques ([DG05a], [DP05a] and [DGP05a]) to transform a raster input image into good quality mosaic; these methods outperform the previous ones in terms of computational cost leading to good aesthetic results. Further no user intervention is needed.

Each method introduces a novel idea (or a new way to use an old idea) in the field of Computer Graphics; in [DG05a] the concept of "directional guidelines" is presented; this image feature characterizes the semantic of the picture that one wishes to render in mosaic. Directional guidelines are related with the salient edges of the image and it is, "per se", an interesting and challenging problem to automatically provide them. In [DP05a] the Antipole strategy [CFP*04] is used to speed up the photomosaic rendering showing how this data structure is suitable to solve NPR problems. Finally in [DGP05a] the previous ideas are merged to produce good quality puzzle image mosaic in an acceptable computation time.

The rest of this paper is organized as follows: in Section 2 we summarize a complete history of digital mosaic, Section 3 explains the algorithm to detect directional guideline and Section 4 shows how to use this result to obtain ancient mosaics. In Section 5 we present the Antipole strategy. Section 6 is devoted to present the method to create photomosaic, while Section 7 presents the Puzzle Image Mosaic technique. In Section 8 we show the experimental results. Finally in Section 9 we suggest directions for future work and research.

2. History of digital mosaic

Computer Graphics attempts to simulate mosaics inscribe themselves into the broader area of non-photorealistic rendering (NPR). In this section we limit our review only to the published works that explicitly name themselves as "mosaic". Although mosaics are a traditional art form attempts to simulate them in the digital realm are recent. Commercial image processing software (the examples in Figure 1a and Figure 1b have been produced with Adobe Photoshop ©) provide "mosaic filters" to obtain tessellated images.

More sophisticated approaches try to adopt smart strategies using computational geometry together with image processing. Haeberli [Hae90] used Voronoi diagrams, placing the sites at random and filling each region with a color sampled from the image. This approach tessellates the image, but tile shapes are too variable and do not attempt to follow edge features (see Figure 1c). This technique is also available in many user-end applications usually under the name of "crystallization" and it simulates the typical effect of some glass windows in the churches. In [DHJN02] Dobashi et al. reprised the Haeberli's idea obtaining good results (see Figure 1d).

"Photomosaic" [SH97] transform an input image into a rectangular grid of thumbnail images (see Figure 1e). In this approach the algorithm searches in a large database of images for one that approximates a block of pixels in the main image. The resulting effect is very impressive, but even in this case no edge features are respected. The idea was successively extended by Klein et al. [KGFC02] to videos obtaining a video mosaic. Recently Di Blasi and Petralia [DP05a] presented an approach to speed up the search process based on the Antipole strategy [CFP*04].

Hausner [Hau01] obtains very good results using centroidal Voronoi diagrams, edge features, L_1 (Manhattan) distance and graphic hardware acceleration to optimize the results (Figure 1f). A very advanced approach to the rendering of traditional mosaics is presented in [EW03]. This technique is based on offset curves that get trimmed-off the self intersecting segments with the guidance of Voronoi diagrams. The algorithm requires a mathematical description, as B-splines, of the edges and allows a very precise tile placement (Figure 1g). Other bonus of this approach is the use of variable size tiles. Although the results are very good the technique seems limited to the case of a single, user-selected and close edge curve. Another approach for the creation of ancient mosaics is presented in [DG05a]; this approach is based on directional guidelines, distance transform, mathematical tools and century proved ideas from mosaicists and leads to impressive results (Figure 1h).

Kim and Pellacini [KP02] introduce a mosaicing technique where image tiles of arbitrary shapes are used to compose the final picture. The idea is quite similar to the photomosaic, but the final effect is very different and interesting (Figure 1i). Another approach for the creation of the same kind of mosaics is presented in [DGP05a]; this approach is based again on the Antipole strategy and leads to impressive results in an acceptable computation time (Figure 1j).

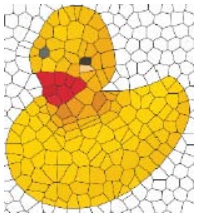
For sake of completeness we also cite "Escherization" [KS00], a technique that produces tilings of the plane using slightly distorted version of an image (Figure 1k). It

relies on symmetry groups and regular tilings. It is very different from the other kind of methods we reviewed above and it is aimed to the production of a sophisticated kind of aesthetic effects different than mosaics.

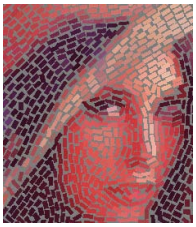

a. Adobe Photoshop ©


b. Adobe Photoshop ©

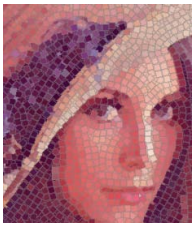

c. Haeberli


d. Dobashi et al.


e. photomosaic


f. Hausner


g. Elber and Wolberg


h. Di Blasi and Gallo


i. Jigsaw Image Mosaic


j. Puzzle Image Mosaic

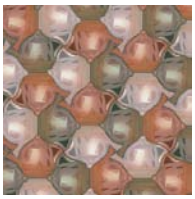
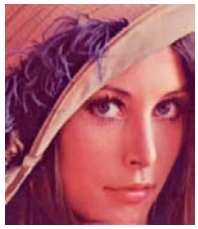

k. Escherization effect

Figure 1: *Mosaic effects*

3 Directional Guidelines Detection

In this section we will present a technique that can be used to automatically detect the directional guidelines of an image. To solve this kind of problem the edge detection algorithms available in literature (see for example [MG01]) are of little use, because here we are searching for "directional guidelines", which are perceptual features not always identifiable with the conventional edges especially in the case of photographic images (see Figure 2). Observe that what we call "directional guidelines" is strongly related with Marr's primal sketch idea (see [Mar82]).


a. The input image


b. Edge obtained by the algorithm proposed in [Mee01]


c. Directional guideline obtained by our algorithm

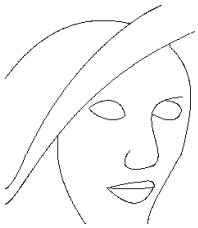

d. Directional guideline obtained manually

Figure 2: *Edge detection versus directional guideline detection*

The technique to compute guidelines is very simple but effective. It works on the luminance channel of an image and starts performing an histogram equalization. It then convolves the image with the origin-centered 2D Gaussian function (σ=16):

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{\sigma^2}}$$

This leads to a new image I_1 (a smoothed version of the original image). Let μ be the mean value of I_1 and let Σ be the variance value. It is hence possible to compute the I_2 image given by the function:

$$I_2(x,y)=\begin{cases}1 & if\ |I_1(x,y)-\mu|>T;\ where\ T=\frac{\Sigma}{4}\\ 0 & elsewhere\end{cases}$$

The threshold value T has been chosen because we have, by trial and error, obtained with such a choice the best results in the successive processing. Finally it convolves I_2 with a Laplace edge detector obtaining, after removing isolated points, the directional guidelines. Figure 3 visualizes the successive steps of the algorithm.


a. The input image


b. The equalized gray scaled image


c. The segment image


d. The directional guidelines

Figure 3: *Directional Guidelines Detection*

4. The Artificial Mosaic

We begin this Section with a more accurate discussion of what the mosaicists do when they create a ancient mosaic. Later we show how this may be translated into an algorithm.

4.1 How the Mosaicists Work

To create a mosaic the artisans first outline the shapes of the image they want to obtain, next they fill the shapes with a sequence of parallel (offset) curves and finally they place the tiles along such curves. These concepts, illustrated in any standard "mosaic producing" handbook (see for example [Kin03], [Nit04]), are clearly illustrated in Figure 4.

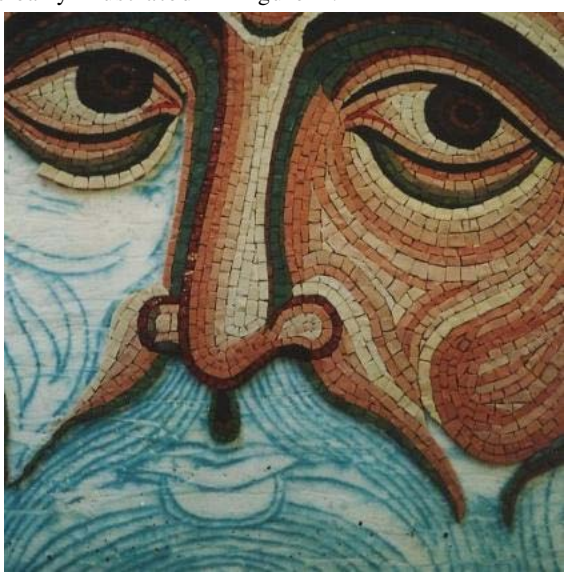


Figure 4: *How the mosaicists work (image from [Tum05])*

The first two steps of the creation of a mosaic are very simple and usually do not represents a problem for the mosaicists. The last one is the more complex one, because mosaicists have a limited set of tile shapes. Usually only rectangular shapes are available, so they must adapt (by cutting) the tiles to insert them in the figure they are realizing. This traditional approach to the problem together with the commonly adopted solutions are very clear observing again Figure 4.

4.2 How to Emulate the Mosaicists' Work

We now suppose to have an image and its directional guidelines as input (Figure 5a and Figure 5b). Using the directional guidelines we evaluate for each pixel of the image the distance transform [HS92], i.e. its minimum distance from any guideline pixel, obtaining a matrix (*dtM*) that is illustrated in Figure 5c (here nearest pixels are white, farthest pixels are black and guideline pixels are yellow). The use of the distance transform in the field of NPR was previously proposed by Gooch et al [GCS02], for a different purpose.

Starting from the distance transform matrix we can obtain another two matrices needed to perform the final mosaicing: the gradient matrix (*gM*) and the level line matrix (*llM*). These matrices are computed as follows:

$$gM(x,y)=\arctan\frac{dtM(x,y+1)-dtM(x,y-1)}{dtM(x+1,y)-dtM(x-1,y)}$$

$$llM(x,y)=\begin{cases}1 & if\ module(dtM(x,y),2\cdot tSize)=0\\ 2 & if\ module(dtM(x,y),2\cdot tSize)=tSize\\ 0 & elsewhere\end{cases}$$

where *tSize* is the user- selected size for the tiles. The *gM* and *llM* matrices are illustrated in Figure 5d and Figure 5e (in Figure 5e, black pixels have value 1, green pixels have value 2).
Observe that:

1. there is no need for the tiles to be square they can have any aspect ratio; however only one dimension, *tSize*, is required to compute the *llM* matrix;
2. the function used to compute the *llM* matrix can be easily adapted in order to prepare the image to the accommodation of variable size tiles as in [EW03].

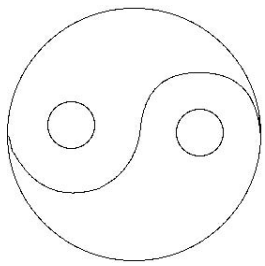
We are now ready to place the tiles, initially all of the same shape and size, using the pixels in *llM* with value 2. Observe that such pixels form chain- like sequences. Of course in the process of placing the tiles their shape has to be altered in order to resolve overlapping.

More precisely the algorithm proceeds as follows:

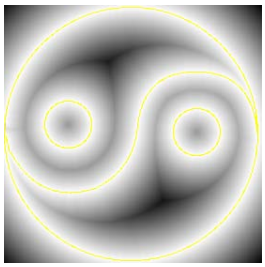
- while there are chains of pixels with value 2 not yet processed:
  a. select a chain;
  b. starting from an arbitrary pixel on it "follow" the chain;
  c. place new tiles at regular distances along the path (the orientation of the tiles is assigned using the gradient information from matrix *gM)*.

The distance along the chain that separates successive tiles is equal to *sSize* when tiles of dimension *tSizexsSize* have been adopted.

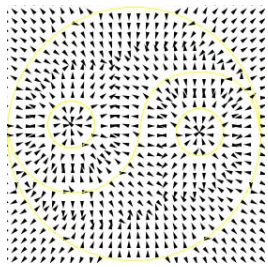
a. The input image



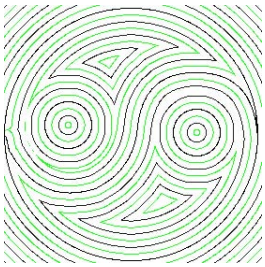
b. The directional guidelines



c. The distance transform matrix



d. The gradient matrix



e. The level line matrix

Figure 5: *The input of our algorithm and the matrices used by the algorithm*

If tiles of fixed size and shape are positioned only according to the method described insofar two main difficulties arise:

1. tiles may overlap;
2. a single tile may cover an area across the "black pixels lines" (i.e. the pixels with value 1 in *llM*).

Both of these effects are unpleasant. In particular the problem in 2 completely destroys the guideline patterns and would result in blurred images.

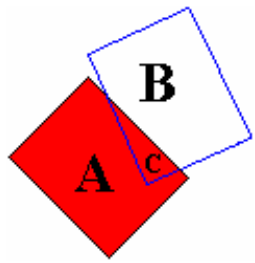
To address these difficulties we adopt a very simple strategy. The overlapping of tiles is easily detected by maintaining a boolean mask of covered pixels. If a tile that we are trying to place contains pixels already covered by previously placed tiles we change the original rectangular shape of the tile "cutting away" the overlapping pixels (see Figure 6a and Figure 6b); if a new tile crosses any "black pixel line" it is trimmed against this line (see Figure 6c and Figure 6d).

Note that until now our tiles have been placed leaving no grout space.

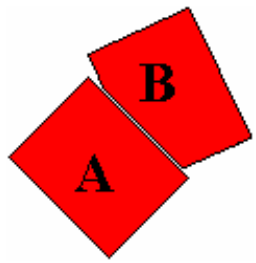
Once the tile positioning and cutting phase has been completely carried out a couple of post- processing steps have to be performed in order to achieve a pleasant aesthetic effect.

First, as it has been pointed earlier in the paper, "grout spaces" between tiles are important. To achieve the effect of cement showing through tiles a downscaling of each tile is done. This frees some pixels that will be assigned a unique color for concrete under the mosaic.

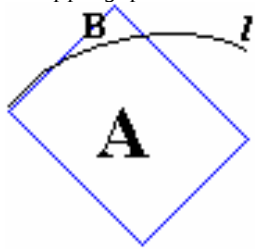
Second, for each tile we calculate a uniform color equal to the color of the pixel corresponding to its center in the source image. Other choices may lead to different artistic effects.



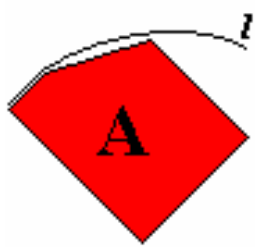
a. tile A has previously placed and tile B is about to be placed. Triangle C is the set of overlapping pixels



b. tile A is left unchanged, the shape of tile B is changed "cutting away" triangle C



c. tile A is about to be placed; it crosses the border l: region B is beyond the border



d. tile A has been trimmed against l: region B has been cut away

Figure 6: *How to cut the tiles*

5. The Antipole Clustering Strategy

The Antipole Tree Data Structure is suitable for searches over large record sets embedded into a metric space (X, d). Records are grouped into clusters of bounded radius by an efficient clustering algorithm: the Antipole Tree Clustering [CFP*04]. The clustering algorithm works in such a way that "far" elements lie in different clusters. The algorithm is able to find a pair (A, B) (called Antipole), such that A and B are far apart, in linear time. Then, elements of the set are partitioned according to their proximity to one of the two Antipole endpoints. This splitting procedure is repeated recursively on each subset and it will produce a binary tree whose leaves are the final clusters. The Antipole Tree Data Structure leads to an efficient nearest neighbor search. The search, starting from the root, proceeds by following the path in the tree, which guarantees to find the nearest cluster centroid pruning the impossible branches. A backtracking search explores the remaining branches of the tree to assure a correct answer. This results in a nearest neighbor search procedure which is faster than the linear Nearest Neighbor search.

6. Photomosaic

The algorithm can be ideally divided into two different steps: database acquisition and photomosaic creation. The following subsections explain in detail these steps.

6.1 Database Acquisition

This first step acquires the database of images and creates the Antipole data structure. The acquisition is very simple: it partitions each image of the database into 9 equal rectangles arranged in a 3x3 grid and computes the RGB mean values for each rectangle. This leads to a vector x composed by 27 components (three RGB components for each rectangle). x is the feature vector of the image in the data structure. When all the images in the database have their own feature vector the Antipole clustering can be performed as explained in the previous section. At the end of this step the Antipole tree is ready for photomosaic creation. Note that, since this process doesn't depend on the input image, it may be performed only once on the whole database.

6.2 Photomosaic Creation

The photomosaic creation is very simple and easy to explain in few steps. First it subdivides the input image into a regular grid, then each cell of the grid into another 3x3 sub-grid. Second it computes the RGB mean values for each sub-cell of the sub-grid. This leads to a vector x composed by 27 components (three RGB components for each sub-cell). x is the feature vector of the cell and can be used to perform the search in the Antipole tree. After performing the best matching it resizes the selected tile to fit and paint it over the cell. The concept of minimum distance between equal tiles has been implemented in order to improve the final result: if the algorithm chooses a tile, then it cannot be chosen again in its neighborhood (whenever this is possible).

7. Puzzle Image Mosaic

7.1 Shape Similarity and Distance

In this subsection we describe how to map a tile into the metric space X in order to create the Antipole data structure. The mapping is very simple: the characterizing features of a tile (for this kind of problem) are its shape and colour. The shape of a tile is composed by the pixels of the image having a non-transparent colour. In our approach the colour value is not considered in this step and it will be taken into account only in a second moment.

There are many techniques to map a shape into a metric space and to evaluate the distance (similarity) between shapes (see for example [LL00] and [SF00]). Here we use a simple but effective method. First we evaluate the shape's center of mass. Then we subdivide the shape into 90 segments, obtaining 90 vertices. Now we compute the Euclidean distance of each vertex from the center of mass and normalize the value in [0,1]. The normalization is done in order to make the distances "scale independent". This leads us to a vector x composed by 90 components. x is the feature vector of the image in the data structure. The shapes distance is computed evaluating the Euclidean distance between feature vectors. The computation takes into account all the possible shifting between the two arrays (that is all the possible mutual rotations of the two shapes). This operation is done in order to make the distance "rotation independent" and "starting point independent". Since a shape is subdivided in 90 segments a rotation error of at most 4 degree ($\pi/45$ radians) is committed: we consider this error acceptable for our purposes.

When all the images in the database have their own feature vector the Antipole clustering can be performed as explained in the previous Section. At the end of this step the Antipole tree is ready for PIM creation. Note that, since this process does not depend on the source image, it

may be performed only once on the whole database.

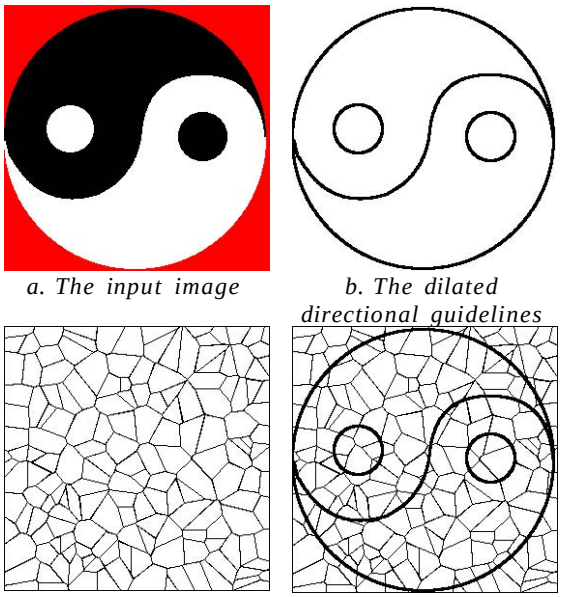
7.2 Merging All Together

In this Subsection we describe how to obtain the final effect. Figure 7 shows the main steps of our algorithm.

We start with an input image (Figure 7a), in the first step the algorithm performs the directional guideline detection (as described in Section 3) and the morphological operation "dilate" obtaining the image G shown in Figure 7b (the dilatation is performed only for better aesthetic results and it does not affect the subsequent steps).

The second step computes a Voronoi diagram V of the same size of the input image; the set of points is randomly chosen and its cardinality is inversely proportional to the median size of the Dirichlet Regions (see Figure 7c).

The third step merges the images G and V obtaining the image R shown in Figure 7d.



a. The input image *b. The dilated directional guidelines*

c. The Voronoi Diagram *d. The final subdivision*

Figure 7: *The main steps of the PIM algorithm*

Now the most important step of the algorithm takes place. For each region R_i of R, we perform the algorithm described in Subsection 7.1 in order to obtain the feature vector x of R_i. x can hence be used to perform the search in the Antipole tree. After performing the best matching we:

1. perform a simple colour shifting in order to align the median colour of the selected tile with the median colour of R_i;
2. rotate and resize the tile to fit and paint it over the region.

8. Experimental Results and Examples

To illustrate the effectiveness of the proposed techniques we report some examples and quantitative results. The algorithms has been implemented in Java2 Standard Edition 1.4.2 and all experiments have been carried out on a PC Athlon XP-M 1800+, 192MB RAM, with Windows XP Home Edition. To allow the reader to directly test the quality of the algorithms three applets are free available respectively at the URLs [DG05b], [DP05b] and [DGP05b] at the same URLs are also available for download the JGimp plug-ins and the Java applications. Some examples of the proposed algorithms are reported in Figures 8 and 9.

Timing results (Table 1, Table 2 and Table 3) show that the algorithms are fast enough to be used as a plug-in in a typical user-end software. Note that the total mean time in Table 2 and Table 3 takes into account the Database Acquisition Mean Time (3.475 sec. for photomosaic and 176.384 sec. for PIM): this operation may be executed only once on the whole database.

9. Conclusions & Future work

In this paper we reviewed three new methods to speed-up the creation of digital mosaics. Experimental results show the soundness of the algorithms.

There are several ways to improve the aesthetic of the results and several ideas started from these works:

1. automatic optimized choices of tile scale relative to each input image is an open problem worth of further investigations;
2. generalization of the "mosaicists' heuristic" to other kind of primitive based non photorealistic image processing seems possible and quite promising;
3. the extension of mosaic technique to other kind of mosaics as proposed in [EW03];
4. the use of Antipole tree or other data structures in other fields of non-photorealistic rendering to speed-up the rendering process;
5. a different method to better find the directional guidelines is an important research investigation issue;
6. extension of the proposed methods for mosaic rendering of 3D surface is probably the most exciting direction of research.

References

[CFP*04] Cantone D., Ferro A., Pulvirenti A., Reforgiato Recupero D., Shasha D., *Antipole Tree indexing to support range search and K-nearest neighbor search in metric spaces*. Accepted to IEEE Transactions on Knowledge and Data Engineering, 2004

[DG05a] Di Blasi G., Gallo G., *Artificial Mosaic*. The Visual Computer 21, pp. 373-383

[DG05b] Di Blasi G, Gallo G., the Artificial Mosaic Creator applet www.dmi.unict.it/~gdiblasi/mosaic/mosaic.html, JGimp plug-in and Java application www.dmi.unict.it/~gdiblasi/mosaic/mosaic.jar, 2005

[DP05a] Di Blasi G., Petralia M., *Fast Photomosaic*. In poster proceedings of ACM/WSCG2005

[DP05b] Di Blasi G., Petralia M., The Photomosaic Creator applet. www.dmi.unict.it/~gdiblasi/photomosaic/photomosaic.html JGimp plug-in and Java application www.dmi.unict.it/~gdiblasi/photomosaic/photomosaic.jar, 2005

[DGP05a] Di Blasi G., Gallo G., Petralia M., *Puzzle Image Mosaic*. In proceedings of IASTED VIIP2005

[DGP05b] Di Blasi G., Gallo G., Petralia M., the Puzzle Image Mosaic Creator applet www.dmi.unict.it/~gdiblasi/PIM/PIM.html, JGimp plug-in and Java application www.dmi.unict.it/~gdiblasi/PIM/PIM.jar, 2005

[DHJN02] Dobashi J., Haga T., Johan H., Nishita T., *A Method for Creating Mosaic Images Using Voronoi Diagrams*. In proceedings of Eurographics2002

[EW03] Elber E., Wolberg G., *Rendering Traditional Mosaics*. The Visual Computer 19, pp. 67-78

[GCS02] Gooch B., Coombe G., Shirley P., *Artistic Vision: Painterly Rendering using Computer Vision Techniques*. In proceedings of NPAR2002, pp. 83-90

[Hae90] Haeberli P., *Paint by Numbers*. In proceedings of SIGGRAPH1990, pp. 207-214

[HS92] Haralick R., Shapiro L., *Computer and Robot Vision* - Vol. 1. Addison-Wesley Publishing Company, 1992

[Hau01] Hausner A., *Simulating Decorative Mosaics*. In proceedings of SIGGRAPH2001, pp. 573-580

[Kin03] King S., *Mosaic Techniques & Traditions: Projects & Designs from Around the World*. Sterling, 2003

[KGFC02] Klein A.W., Grant T., Finkelstein A., Cohen M.F., *Video Mosaics*. In proceedings of NPAR2002, pp. 21-28

[KS00] Kaplan C., Salesin D., *Escherization*. In proceedings of SIGGRAPH2000, pp. 499-510

[KP02] Kim J., Pellacini F., *Jigsaw Image Mosaics*. In proceedings of SIGGRAPH2002, pp. 657-664

[LL00] Latecki L.J., Lakaèmper R., *Shape Similarity Measure Based on Correspondence of Visual Parts*. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1185-1190, October 2000

[Mar82] Marr D., *Vision*. W.H. Freeman and Company, New York, 1982

[MG01] Meer P., Georgescu B., *Edge Detection with Embedded Confidence*. Transaction on Pattern Analysis and Machine Intelligence 23 (12), pp. 1351-1365, 2001

[Nit04] Nittolo F., Il mosaico. http://www.ravennarte.it/rarte-ing/mosaico.htm, 2004

[SF00] Sako Y., Fujimura K., *Shape Similarity by Homotopic Deformation*. The Visual Computer, pages 47-61, February 2000

[SH97] Silvers R., Hawley M., *Photomosaics*. Henry Holt, New York, 1997

[Tum05] Tumminello S., Descrizione e Tecnica utilizzata nei mosaici del Duomo di Monreale. http://www.parrocchie.it/monreale/sscrocifisso/italia/mosaici.htm, 2005

Size	*Guideline Mean Detection Time (sec.)*	*Mosaic Mean Time (sec.)*	*Total Mean Time (sec.)*
600x600	1.402	10.485	11.887
800x600	1.402	12.689	14.091
593x886	1.602	15.182	16.784
1024x768	2.243	22.142	24.385

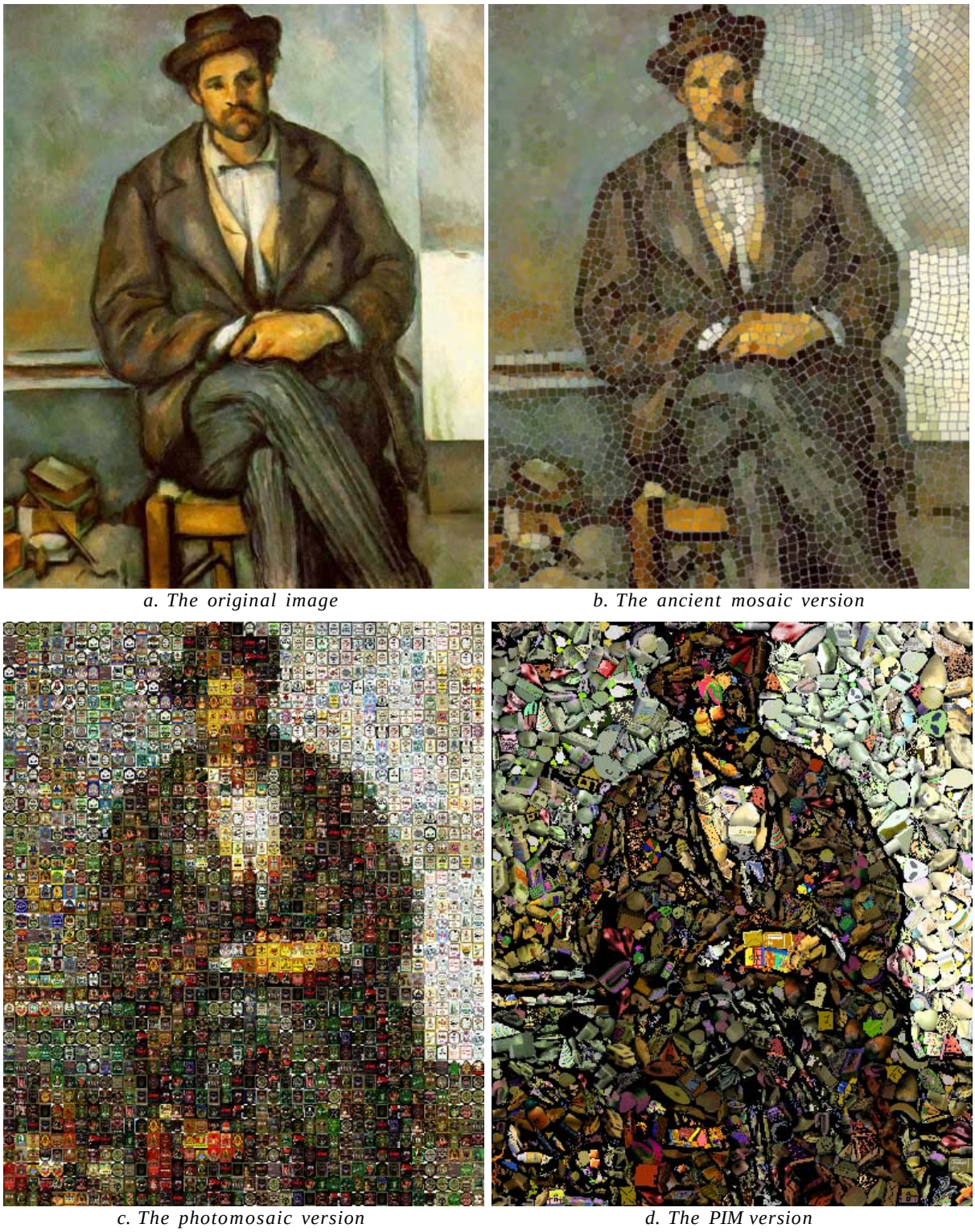
Table 1: *Timing results of artificial mosaic*

Size	*Total Mean Time (sec.)*	*Size*	*Total Mean Time (sec.)*
275x276	6.701	640x480	16.044
320x240	5.980	600x600	16.053
400x327	7.511	800x600	19.058
400x486	10.265	593x886	24.786
407x550	11.176	970x676	25.614
512x512	12.459	1024x768	32.487

Table 2: *Timing results of photomosaic (1417 tiles, tile size of 10x10 pixels, minimum distance of 5 tiles)*

Size	*Guideline Detection Mean Time (sec.)*	*PIM Mean Time (sec.)*	*Total Mean Time (sec.)*
275x276	741	14431	191556
400x486	1738	51093	229215
600x600	2069	88678	267131
896x601	4597	106714	287695
899x615	4427	111911	292722

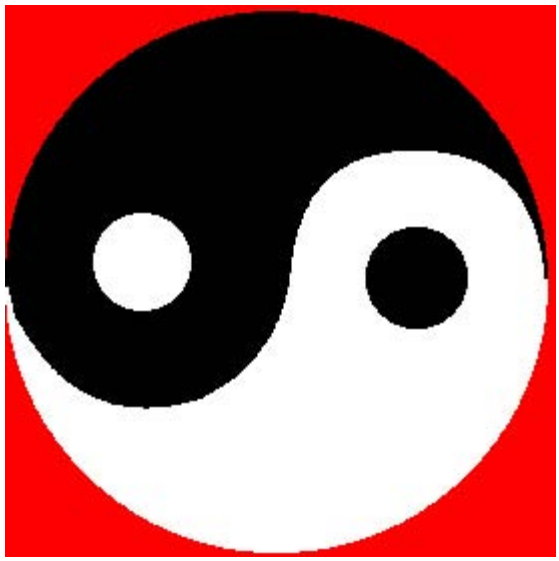
Table 3: *Timing results of PIM (1025 tiles. median size of Dirichlet Regions of 15 pixels)*



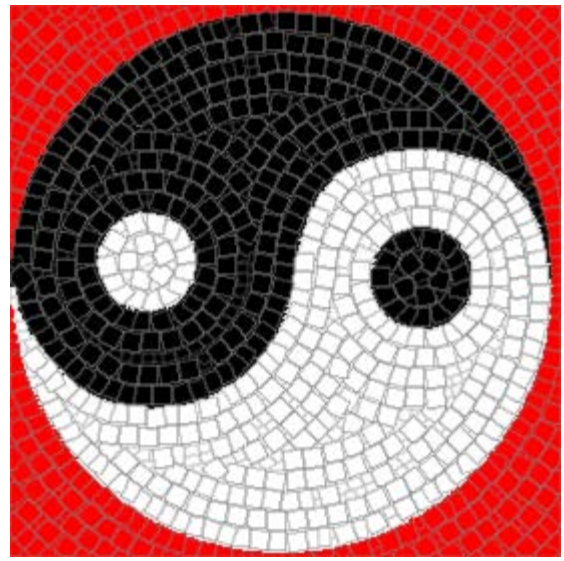
a. The original image *b. The ancient mosaic version*

c. The photomosaic version *d. The PIM version*

Figure 8: An *example of the proposed technique applied on a Cezanne's painting*



a. The original image



b. The ancient mosaic version

c. The photomosaic version

d. The PIM version

Figure 9: *Another example of the proposed technique applied on the Yin/Yang image*