Eurographics Italian Chapter Conference (2006)
G. Gallo and S. Battiato and F. Stanco (Editors)

Smart Ideas for Photomosaic Rendering

Gianpiero Di Blasi, Giovanni Gallo, Maria Pia Petralia
Dipartimento di Matematica ed Informatica
Università di Catania
via A. Doria, 6 – 95125 Catania (Italy)
email:{gdiblasi, gallo, petralia}dmi.unict.it

Abstract

Photomosaic is a technique which transforms an input image into a rectangular grid of thumbnail images preserving the overall appearance. The typical photomosaic algorithm searches from a large database of images one picture that approximates a block of pixels in the main image. Since the quality of the output depends on the size of the database, it turns out that the bottleneck in each photomosaic algorithm is the searching process. In this paper we present a technique to speed-up this critical phase using the Antipole Tree Data Structure. This improvement allows the use of larger databases without requiring much longer processing time, leading to very interesting results. Further we present some ideas to render an image in a "photomosaic style". These techniques are based on QuadTree and fractal images and allow the creation of very interesting "photomosaic effects".

Categories and Subject Descriptors (according to ACM CCS): J.5 [Arts and humanities]: Architecture, Fine Arts

1. Introduction

Photomosaic is a technique which transforms an input image into a rectangular grid of thumbnail images. Most people know them from seeing posters and magazine covers done by Robert Silvers [SH97], although the origin of photomosaic idea is older.

The bottleneck in each photomosaic algorithm is the search in a large database of images to find a best match. This search is usually sequential and the time required to perform this task is, hence, high. Some photomosaic algorithm relies on the off-line construction of a suitable data structure to speed-up the matching process. However the overall time required to create this structure could be excessive.

In this paper we present a new method to speed-up the search process. This technique is based on the Antipole Tree Data Structure [CFP*05] and allows us to obtain impressive effects in an efficient manner. The Antipole Tree Data Structure is suitable for searches over large record sets embedded into a metric space (X, d). In our approach, each picture in a database becomes a point in X. Distance d between two pictures is computed in a straightforward way using the RGB values. Images are grouped into clusters of bounded radius by an efficient clustering algorithm: the Antipole Tree Clustering [CFP*05]. The clustering algorithm works in such a way that "far" elements lie in different clusters. The algorithm is able to find, in linear time, a pair (A, B) (called Antipole), such that A and B are far apart. 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.

Although the very interesting and beautiful effects obtained by photomosaic, few enhancements have been proposed in order to improve the photomosaic rendering technique. For these reasons in this paper we also present some ideas to create other "photomosaic style" images.

These ideas can be thought as a starting point for the creation of a "photomosaic framework". In particular we present two techniques called: QT-Photomosaic and FQT-Photomosaic. In QT-Photomosaic we use the QuadTree splitting technique [FB74] to subdivide the input image into a non-regular rectangular grid. The splitting is based on the mean RGB values of the image and its variances. In FQT-Photomosaic we extend the QuadTree idea in order to produce a fractal photomosaic image [BH93]. To obtain this results we restrict the database of thumbnail images to the image itself and its subQuadTree-images. A preliminary and partial version of the results reported here has been presented in [DP05].

The rest of this paper is organized as follows: in Section 2 we present an history of photomosaic, Section 3 explains the Antipole strategy. In Section 4 we explain our algorithm and in Section 5 we show some experimental results. Finally in Section 7 we suggest directions for future works and researches.

2. History of Photomosaics

Even before the computer's era, the process of making

pictures from other pictures was well known. In 1973 Leon Harmon [Har73] presented a work including several "block portraits" (see Figure 1a). Harmon used these "pixelated" portraits to study human perception and the automatic pattern recognition issues. For example he used them as a demonstration of the minimal condition to recognize a face. The image in Figure 1a is just a very low resolution rendering (252 pixels) of a gray image of Lincoln. Each pixel is seen as a "tile". This image of Lincoln needs to be viewed at a distance to see the face. The fact that Lincoln's face is so well known makes its recognition easier.

In 1976 Salvador Dali [ND01] completed the painting in Figure 1b titled "Gala contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln (Homage to Rothko)". Note that Lincoln's face is made up by pictures with full tonal ranges; perhaps the earliest example of this technique.

The nude taking up several tiles is Dali's wife Gala and one of the tiles is Harmon's gray scale image of Lincoln; so not only Dali did an appropriate Harmon's Lincoln portrait into the overall composition, but he also reincorporated a smaller gray scale version into a single tile. If not the first, this is certainly the most well known image made from other images.

In the 1970's the American artist Chuck Close [Clo70] began producing precisely gridded paintings (see Figure 1c). Some of these had a quality that reminds Impressionists, while others seem to be computer-generated or computer influenced.

The earliest example of making a tile more than a single pixel came from an artist toying with Adobe Photoshop© [Ado06] and the earliest example of a photographic computer mosaic is the image created by Dave McKean for Vertigo/DC comics in 1994 (see Figure 1d).

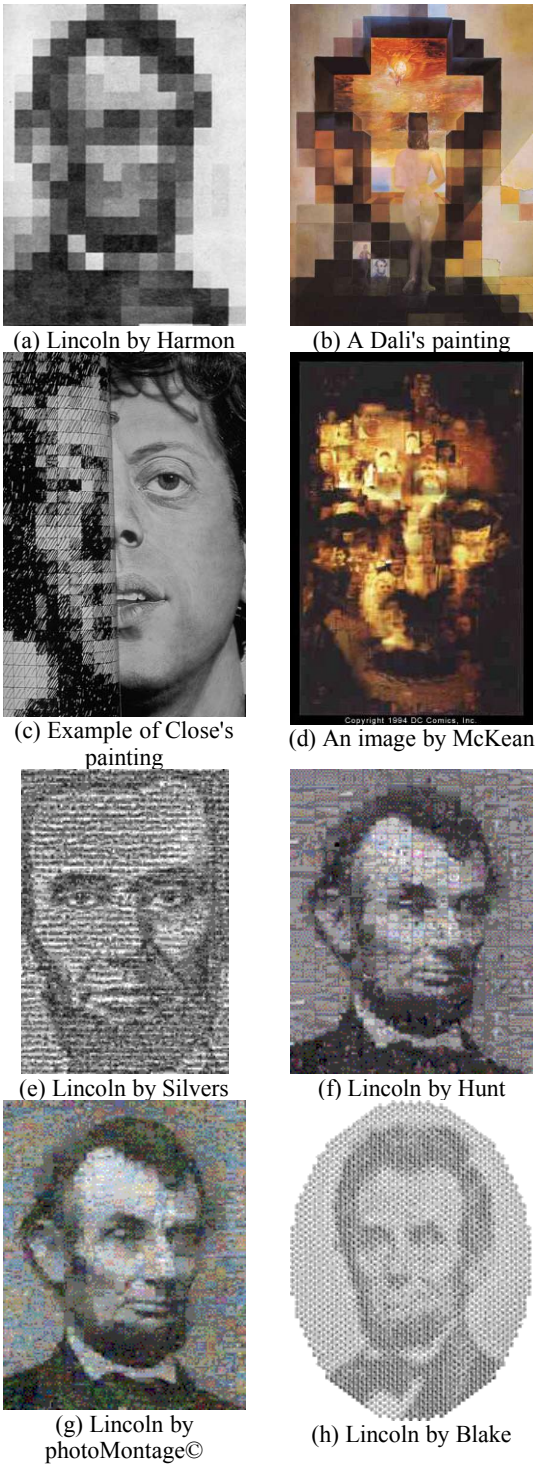
Computer replicated images made from tiled digital photographic pictures is recent because it requires large computational resources. Robert Silvers [SH97] began working on the first photomosaic while he was a graduate student at the MIT Media Lab. Each tile in his images represents much more than a single value. Smaller pictures match the overall image in tone, texture, shape and color. Silvers was commissioned by the US Library of Congress to create the portrait of Lincoln in Figure 1e using archived photos of the American Civil War.

Today, Runaway Technology [Run06] (Silvers' company) produces photomosaic images as logos and illustrations for individuals, corporations and publications.

William Hunt [Hun98], a computer programmer, created the image in Figure 1f. He used three different size tiles to change the look of the grid.

ArcSoft photoMontage© [Arc06] is one of several commercial programs available nowadays. The image in Figure 1g was made using this software.

Scott Blake's image of Abraham Lincoln in Figure 1h is rendered by using 42 portraits of all US Presidents [Bla98]. He arranged the presidents according to their gray scale density. He offset the tiles on a beehive grid pattern to produce a hypnotic effect and used the oval portraits to fill the space in a more efficient manner.



(a) Lincoln by Harmon (b) A Dali's painting

(c) Example of Close's painting (d) An image by McKean

(e) Lincoln by Silvers (f) Lincoln by Hunt

(g) Lincoln by photoMontage© (h) Lincoln by Blake

Figure 1: *Photomosaics.*

The original Silver's idea was successively extended by Klein et al. KGFC02 to videos obtaining a video mosaic and recently [DP05] we presented an approach to speed up

the search process based on the Antipole strategy [CFP*05].

3. The Antipole Strategy

The Antipole Clustering Strategy [CFP*05] of bounded radius is a top-down procedure that starts with a given finite set of points X in a metric space. The first check is if there exists a pair of points in X such that their distance is longer than the radius. If this is the case, the (A, B) pair is called Antipole and the set is partitioned by assigning each point of the splitting subset to the closest endpoint of the Antipole (A, B). Otherwise the splitting is not performed and the given subset is itself a cluster. Figure 2 shows the Antipole Tree construction.

Once the data structure has been built a suitable nearest neighbor algorithm can be designed. 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.

Notice that the Antipole Tree Data Structure is organized in such a way that linear scanning during the search may be avoided and in [CFP*05] a comparison between Antipole and Mtree empirically shows that Antipole computes less distances than Mtree. This results in a nearest neighbor search procedure which is faster than the linear nearest neighbor search. In our case, after empirical tests, we chosen Antipole Tree Data Structure because it performs better than other nearest neighbor techniques for the photomosaic problem.

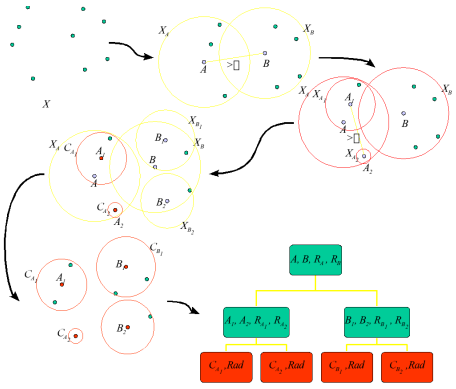


Figure 2: *The Antipole Tree construction*

4. The Proposed Algorithm

The proposed algorithm can be ideally divided into two different steps: database acquisition and photomosaic creation (with QT and FQT extensions).

4.1. Database acquisition

In this first step we acquire the database of images and create the Antipole Tree Data Structure. The acquisition is very simple: we partition each image of the database into 9 equal rectangles arranged in a 3x3 grid and compute the RGB mean values for each rectangle. This leads us to a vector x composed by 27 components (three RGB components for each rectangle). Figure 3 shows this process.

Figure 3: *Vector x creation*

x is the feature vector of the image in the data structure. When all images in the database have their own feature vector the 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 does not depend on the input image, it may be performed only once on the whole database.

4.2. Photomosaic creation

The photomosaic creation is very simple and easy to explain in few steps. First we subdivide the input image into a regular grid, then each cell of the grid into another 3x3 sub-grid. Second we compute the RGB mean values for each sub-cell of the sub-grid. This leads us 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 we resize the selected tile to fit and paint it over the cell. We implemented the concept of minimum distance between equal tiles to improve the final result: if we choose a tile, then it cannot be chosen again in its neighborhood (whenever this is possible).

4.3. QT-Photomosaic and FQT-Photomosaic

A Quadtree [FB74] is a tree having four branches at each node. Quadtrees are used in the construction of some multidimensional databases (e.g., cartography, computer graphics, and image processing). The basic principle of a Quadtree is to cover a planar region of interest by a square, then recursively partition squares into smaller squares until each square contains a suitably uniform subset of the input (based on some features); for instance this can be used to compress bitmap images by subdividing until each square has the same color value. This and related partitions have many applications in computational geometry and geometric applications, including data clustering, shape representation, n-body simulation in astronomy and molecular modeling, and mesh generation.

In our problem the Quadtree is used to subdivide an image in squares such that each square has a RGB variance values lower than an user selected parameter. This leads to a new subdivision of the image, different than the classic

rectangular regular grid.

Once we have obtained the Quadtree, it is possible to apply the same algorithm described in the previous Subsection in order to find the best matching tile in the Antipole Tree. After performing the best matching, like in the previous case, we resize the selected tile to fit and paint it over the cell.

Using this subdivision technique, it is not possible to implement the concept of minimum distance between equal tiles: the concept of cell's neighborhood it is not easily applicable in a Quadtree.

In FQT-Photomosaic, we extend the QuadTree idea in order to produce a fractal photomosaic image [BH93]. To obtain this results we restrict, for each square, the database of thumbnail images to the image itself and its sequence of subQuadTree-images containing the square under observation. To perform this task we need to recursively compute the Antipole Tree for each square. This operation can be optimized using several techniques coming from dynamic programming.

5. Experimental Results

To illustrate the effectiveness of the proposed technique we report some examples and quantitative results. The algorithm 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 test directly the quality of our algorithm an applet is available at the URL [DGP05], at the same URL is also available for download a JGimp plug-in and a Java application.

The examples in Figures 4and 5 show the effectiveness of the proposed technique and that our method performs well on different kind of images (graphics, photos, paintings, etc.).

Timing results (Table 1) show that our algorithm is fast enough to be used as a plug-in in a typical user-end software. Note that the total mean time in Table 1 takes into account the Database Acquisition Mean Time (3.475 sec.): this operation may be executed only once on the whole database.

It is straightforward to verify that the time asymptotic complexity is linear in the number of cells of the image.

6. Conclusions & Future Works

In this paper we presented some ideas to speed up the searching process and to improve the quality of photomosaic image rendering.

There are several ways to improve the aesthetic of our results and several ideas started from this work:

1. the extension of photomosaic technique to other kind of mosaics;
2. the use of Antipole tree or other data structures in other fields of non-photorealistic rendering to speed-up the rendering process;
3. extension of our method for photomosaic rendering of 3D surface is probably the most exciting direction of research.

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 1: *Timing results (using a database of 1417 tiles, a tile size of 10x10 pixels and a minimum distance between equal images of 5 tiles).*

References

[Ado06] Adobe Photoshop, http://www.adobe.com, 2006.

[Arc06] ArcSoft PhotoMontage, http://www.arcsoft.com, 2006.

[BH93] BARNSLEY M.F., HURD L.P.: *Fractal Image Compression*. AK Peters Ltd., 1993.

[Bla98] BLAKE S., http://www.barcodeart.com/, 1998.

[CFP*05] CANTONE C., FERRO A., PULVIRENTI A., REFORGIATO RECUPERO D., SHASHA D.: Antipole Tree indexing to support range search and K-nearest neighbor search in metric spaces. *IEEE/TKDE* 17, 4 (2005), pp. 535-550.

[Clo70] CLOSE C., http://www.chuckclose.coe.uh.edu/, 1970.

[DGP05] DI BLASI G., GALLO G., PETRALIA M., The Photomosaic Creator applet www.dmi.unict.it/~gdiblasi/photomosaic/photomosaic.html and JGimp plug-in and Java application www.dmi.unict.it/~gdiblasi/photomosaic/photomosaic.jar (2005).

[DP05] DI BLASI G., PETRALIA M.: Fast Photomosaic. In *Poster Proc of ACM/WSCG2005 (*2005).

[FB74] FINKEL R.A., BENTLEY J.L.: *Quad trees: A data structure for retrieval on composite keys*. Acta Informatica 4 (1974), pp. 1-9.

[Har73] HARMON L.D.: The Recognition of Faces. *Scientific American* 229, 5 (1973).

[Hun98] HUNT W.L. http://home.earthlink.net/~wlhunt/, 1998.

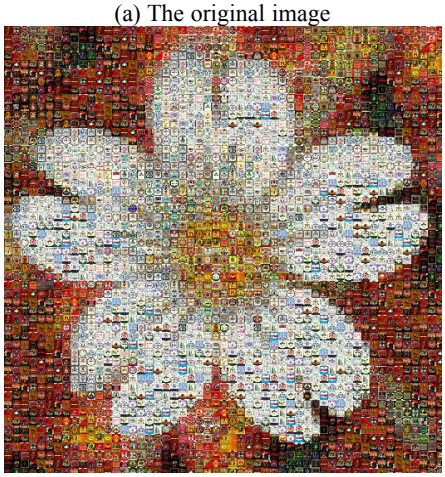
[KGFC02] KLEIN A.W., GRANT T., FINKELSTEIN A., COHEN M.F.: Video Mosaics. In *Proc. NPAR2002* (2002).

[ND01] NERET G., DESCHARNES R.: *Dali: The Paintings*. Taschen, 2001.

[Run06] Runaway Technology, http://www.photomosaic.com/, 2006.

[SH97] SILVERS R., HAWLEY M.: *Photomosaics*. Henry Holt, 1997.

(a) The original image



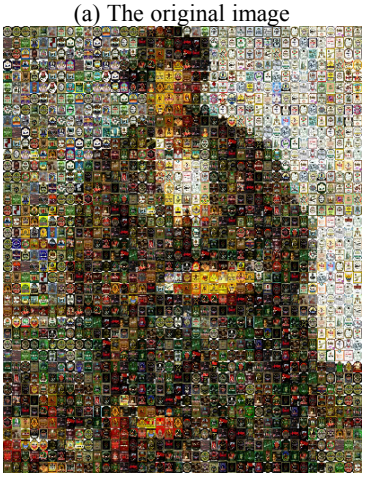
(b) photomosaic

(c) QT-photomosaic

(d) FQT-photomosaic

Figure 4: *Some examples of photomosaic*

(a) The original image



(b) photomosaic

(c) QT-photomosaic

(d) FQT-photomosaic

Figure 5: *Other examples of photomosaic*