GENERATING PHOTOMOSAICS: AN EMPIRICAL STUDY

Nicholas Tran*

Keywords: photomosaics, dynamic programming, string matching.

Abstract

Photomosaics are images obtained from assembling a large number of small photographs called tiles. They were invented in 1995 by Rob Silvers while working on his master's thesis under Michael Hawley [7]. Besides their obvious values in art and entertainment, photomosaics could potentially be useful in digital copyrighting, compression, and complexity theory. This paper proposes to measure i) the effectiveness of algorithms for generating photomosaics in terms of *similarity* to the original image, *granularity* of the individual tiles, and *variety* of the selected tiles; and ii) the costs of these algorithms in terms of running time and tile library size. The effectiveness and costs of two photomosaic algorithms based on string matching techniques are studied. Results suggest various direct and inverse linear relationships between similarity, granularity, variety, and the library size.

1 Introduction

A *photomosaic* of an image is obtained by assembling a large number of smaller unrelated photographs called *tiles*, so that each tile approximates a small block of the image. A good photomosaic is striking, because it cleverly puts together otherwise ordinary and unrelated features of the individual tiles into a coherent larger framework. Generating a photomosaic requires more effort than assembling a large number of plain black and white tiles to obtain a typical newspaper photograph. Photomosaics were invented by Rob Silvers, then a graduate student at the MIT Media Lab working on his master's thesis. A nice collection of photomosaics has appeared in [7], as well as on the covers of recent issues of many magazines, ad campaigns, and movie posters [6]. See Figures 7 and 8 for an example of photomosaics.

*Department of Computer Science, Wichita State University, Wichita, KS 67260-0083. Email: tran@cs.twsu.edu.

SAC '99, San Antonio, Texas

The introduction of [7] outlines a brief history of ideas leading to photomosaics. Making pictures using a restricted set of building blocks is an idea dated back to the works of the painter Giuseppe Arcimboldo in the 16th century (he particularly liked vegetables, games, and fish as building blocks). More recent literary works by Lewis Carroll [1] and John Hollander [2] contained shapes of animals assembled from English words, presumably by hand. The idea of using a computer to help in the decomposition of a picture into building blocks was introduced by Ken Knowlton in his software DominoPix [3], which allows constructing portraits out of sets of dominoes (see also [4]). Silvers' idea of using photographs as building blocks took advantage of recent advances in semiconductor and storage technologies resulting in powerful graphics workstations and vast databases of pictures.

Besides their obvious applications in art and entertainment, photomosaics could potentially be useful in protecting digital copyrighted materials. For example, preview samples of a proprietary picture can be prepared by making a photomosaic of it using a set of building blocks bearing the owner's identification. This would give enough information about the picture and the owner and yet not enough for the picture to be used in other purposes. When viewed as a method of organizing information, photomosaics could have applications in compression and database technology. Photomosaics are also good examples of the diagonalization technique in complexity theory (a photomosaic has something in common with each tile) and may have further applications in this field.

The problem of generating good photomosaics has not been studied in a formal setting. One reason is because the area is so new; not many implementations of photomosaics exist, and in fact, not much is known even about Silvers' proprietary implementation. Another reason is that the quality of photomosaics depends in part on properties of the human visual system and thus must be studied empirically. For example, how closely must each tile approximate the corresponding block of the original image ? How large can each tile be before details in the original image are lost ? How many tiles must the library have ? How many times can a tile be used before the photomosaic becomes monotonous ?

This paper proposes a quantitative method for studying algorithms for generating photomosaics by focusing on their effectiveness and costs, where

- effectiveness is measured in three aspects: *similarity* between the photomosaic and the original image, *granularity* of details in each individual tile, and *variety*

of the selected tiles. Intuitively, a good photomosaic should resemble the original image and yet consists of tiles that are large enough and different enough to make the decomposition nonmonotonous;

- costs are measured in two aspects: *running time* of the generation, and *library size*, the number of tiles available for selection.

This framework is then used to study two photomosaic algorithms that are based on string matching techniques. Both algorithms seek to minimize a distance measure based on pixel values between a block of the original image and the selected tile. The first method treats the comparison as an exact matching problem. It computes for each tile the sum of all absolute differences in pixel values and selects the tile with the minimum sum. In contrast, the second method treats the comparison as an inexact matching problem. It treats each row of pixels in a tile as a string and performs a string alignment algorithm [8] to find the optimal alignment score between this row and the corresponding row in the original image. The selected tile has the maximum sum of the optimal scores for all rows.

The two algorithms were used to generate photomosaics of a picture of Wushock, the Wichita State University mascot, using different tile dimensions and tile library sizes, and the running times, variety, and similarity were recorded. Variety was measured as the number of tiles actually used to create the photomosaic. Similarity was measured as the minimum distance in feet at which no significant difference could be perceived by the author between the original and the photomosaic. The LUG library was used to facilitate handling of graphics files in various formats [5]. Original images and tiles were in JPEG or GIF formats. Output photomosaics were generated in PBM format.

The results obtained indicate that the minimum distance increases proportionally to the square root of the tile area, and that the minimum distance roughly decreases proportionally to the number of tiles used. The performance of the two algorithms was compared, and it was found that the first method has a faster running time, which is proportional to product of the area of the original image and the library size. In contrast, the running time of the second method is proportional to the product of the area of the original image, the library size, and the width of a tile. Moreover, photomosaics produced by the first method were more similar to the original image, but those produced by the second had more variety.

The rest of this paper is organized as follows. Section 2 describes the details of the two algorithms, Section 3 describes the experiment setup and the results obtained, and Section 4 discusses the implications of this empirical study.

2 Algorithms

We describe in this section two algorithms for generating photomosaics based on string matching techniques. Neither algorithm attempts shape or edge analysis. Rather, each seeks to minimize a distance measure based purely on differences of pixel values. Recall that each pixel value is a triple of eight-bit binary numbers representing the intensity of its red, green, and blue components. The original image is divided into rectangular blocks of the same dimensions as the tiles; the leftover strips on the right and bottom edges are ignored. Both algorithms scan the blocks of the original image from left to right and top to bottom, replacing each block by a tile before considering the next.

In the following we will denote the width and height of the original image as W and H, the width and height of a tile as w and h, and the number of tiles as T.

2.1 Algorithm I: L_1 distance

Perhaps the simplest method possible, this algorithm in its basic form replaces a rectangular block of the original image with the tile whose L_1 distance to the block is minimum. More precisely, denote a rectangular block of the original image by the matrix $A_{w\times h}$, a tile k by the matrix $T^k_{w\times h}$, and the red, green, and blue components of pixel $A(i,j)$ by $A_r(i,j)$, $A_g(i,j)$, and $A_b(i,j)$ respectively. Then the algorithm seeks to minimize the following distance over all tiles in the library: $d = \min_k \sum_{i=1}^{w}\sum_{j=1}^{h} |A_r(i,j) - T^k_r(i,j)| + |A_g(i,j) - T^k_g(i,j)| + |A_b(i,j) - T^k_b(i,j)|$.

To increase the variety measure of the generated photomosaics, two modifications are made to the basic algorithm. First, a limit is imposed on the number of times a tile can be selected. Second, a minimum distance is required between any two copies of a tile. To implement these changes, the count and last selected position is maintained for each tile.

Given an image of size WH and a library of T tiles of size wh, there are $O(\frac{WH}{wh})$ blocks, and selecting the tile to replace each block takes $O(Twh)$ time, so the total running time of this algorithm is $O(\frac{WH}{(wh)}Twh) = O(WHT)$.

2.2 Algorithm II: Alignment

This method treats the rows of a block of the original image as strings and computes the optimal alignment score between them and the corresponding rows of each tile. An alignment of two strings s and t (of possibly different lengths) is obtained inserting spaces in the strings so that their lengths become the same. Note that there are many possible alignments. For example, two alignments of the strings "mastery" and "mars" are

```
masters        masters
ma   rs        m   ars
```

The score of an alignment is computed by considering the pairs of characters given by the alignment: a match scores m, a mismatch scores d, and a gap scores g, where m, d, and g are some chosen values. For the above example, if $m = 1$, $d = -1$, and $g = -2$, then the alignment scores are -2 and -4. Alignment scores are related to edit distances and are used to measure similarity between two almost identical objects. Alignment is well suited for inexact matching and is used extensively in computational biology to detect relationships between DNA strands [8].

Back to the algorithm, where the strings are actually two rows of pixel values, the values of m, d, and g are chosen as 0, $-(|s_r(i) - t_r(j)| + |s_g(i) - t_g(j)| + |s_b(i) - t_b(j)|)$, and -30, respectively. Note that $-3 * 2^8 \le d \le 0$, and so the more negative values of g enforce more exact matching.

The optimal alignment score between two rows of pixel values is the maximum score among all alignments. This value can be computed using dynamic programming. Formally, given two rows s and t, define $D(i,j)$ to be the optimal alignment score between the two subrows $s[1..i]$ and $t[1..j]$. $D(w,w)$ is the value we are looking for (recall that w is the width of the tiles). The following recurrence relation gives

us a method to compute the solution:

$$D(i,j) = \max \begin{cases} D(i-1,j-1) - |s_r(i) - t_r(j)| - \\ \quad |s_g(i) - t_g(j)| - |s_b(i) - t_b(j)|, \\ D(i-1,j) - 30, \\ D(i,j-1) - 30. \end{cases}$$

The boundary conditions are given by $D(1,i) = -i * 30$ and $D(j,1) = -j * 30$. The elements of the matrix D can be computed by initializing the first row and column with the boundary conditions and then evaluating the elements from left to right and top to bottom. This is possible since the value of $D(i,j)$ depends only on $D(i-1,j-1)$, $D(i-1,j)$, and $D(i,j-1)$. The running time of evaluating $D(w,w)$ is $O(w^2)$.

For each tile, the optimal alignment score is found for each row, and their sum computed. The tile having the largest sum is selected to replace the block of the original image under consideration. Again, to increase the variety measure, a limit is imposed on the number of times a tile can be used, as well as a minimal distance between any two copies of a tile.

Given an original image of size WH and a library of T tiles of size wh, there are $O(\frac{WH}{wh})$ blocks, and selecting the tile to replace each block takes $O(Tw^2h)$ time, so the total running time of this algorithm is $O(\frac{WH}{wh}Tw^2h) = O(WHTw)$.

3 Setup and Results

The two algorithms were implemented in C in about 500 lines of code. Support for handling different graphics file formats and for common graphics operations was provided by the LUG and JPEG libraries. The program received the names of the original image and tiles on its command line. The tiles were preprocessed and grouped into libraries of predetermined dimensions from raw photographs (in JPEG or GIF format) collected from webpages on the World Wide Web. Before the raw photographs were compressed, cropping was performed if necessary at the bottom edge and equally at the left and right edges to preserve the original aspect ratio. The program displayed the original image and successively replaced each block with the selected tile from left to right and top to bottom. At the end, the program output the photomosaic in PBM format and statistics such as number of blocks, and number of tiles used. The CPU time was measured with the built-in `tcsh` shell command `time`.

The program was run on a Pentium 2 233MHz workstation running Debian/GNU Linux 1.2 using a fixed original GIF image of Wushock, the Wichita University mascot (970 pixels wide by 864 pixels high). Algorithms I and II were run on tile libraries of dimensions 16x12, 20x15, 24x18, 28x21, 32x24, 40x30, 48x36, and 60x45. Each library contained 2170 tiles obtained from the same set of photographs. Algorithm I was also run on tile libraries of dimensions 32x24. The sizes of these libraries were 128, 357, 739, 1162, and 1642. The output photomosaics were then compressed using `xv` to 50% of its original size (to fit on a 8.5x11 sheet) and printed with a Postscript printer. Each such printed copy was placed side by side with a copy of the original, and the minimum distance at which no significant difference between the two could be perceived by the author was recorded. The measurements are summarized in Figures 1, 2, and 3. A selection of these photomosaics, compressed to 12% of their original size, appears in Figures 4, 5, and 6.

Tile Size	CPU (sec.)	Blocks	Tiles Used	Dist. (ft.)
16x12	967	4320	1017	12
20x15	1001	2736	704	17
24x18	1040	1920	555	25
28x21	1000	1394	452	25
32x24	1017	1080	380	27
40x30	990	672	263	31
48x36	1019	480	217	39
60x45	997	304	148	48

Figure 1: Algorithm I on various tile dimensions (library size = 2170)

Tile Size	CPU (sec.)	Blocks	Tiles Used	Dist. (ft.)
16x12	7758	4320	1948	28
20x15	8932	2736	1393	38
24x18	10597	1920	1044	38
28x21	14178	1394	806	51
32x24	13854	1080	657	52
40x30	16333	672	433	63
48x36	23485	480	332	71
60x45	28622	304	228	82

Figure 2: Algorithm II on various tile dimensions (library size = 2170)

Library Size	CPU Time (sec.)	Tiles Used	Dist. (ft.)
128	47	128	52
357	148	255	38
739	333	293	37
1162	536	332	36
1642	765	362	31

Figure 3: Algorithm I on various library sizes (tile dimensions = 32x24, 1080 blocks)

4 Discussion

The experimental data in Figures 1, 2, and 3 show that the running times of both algorithms agree well with their derived bounds. In both cases, the results suggest a strong linear relationship between similarity and granularity: the minimum distance increases proportionally with the tile area. The evidence for a linear relationship between similarity and library size is less strong, but it does show that similarity improves with increased library size. (Fig. 3 does not give data for Alg. II due to its high running time, but the same relationship is likely to hold.) However the relationship is more direct between similarity and the actual number of tiles used. This suggests that the lack of a strong relationship between similarity and library size may be due to the quality of the library, i.e. how diverse the tiles are. Determining what constitute a diverse tile library is an important direction for future research. These relationships are potentially helpful in determining the tile dimensions and library size that must be used to achieve a desired effect (eg. preparing a billboard to be visible from a certain distance range.)

The prohibitive running time of Alg. II makes it much less practical than Alg. I, although a few heuristics to reduce its running time is known. A comparison of the photomosaics produced by Algorithms I and II show that those produced by the former are much more faithful in color and shape to the original, and the ones produced by the latter are more "busy" with details. This observation agrees with the experimental data, which show that the minimum distances for Alg. II were twice as long as those for Alg. I, but the numbers of tiles used were also doubled. (The data also suggests that Silvers' method is probably very similar to Alg. I.) Thus the choice of which algorithm to use depends on which of the two conflicting goals (similarity or variety) is considered more important in good photomosaics. In fact, Alg. I could be considered as a special case of Alg. II with a very negative value for g, the gap penalty. (We used -30, a small value, for this study.) It would be interesting to determine the range of values for g that would yield both satisfactory similarity and variety.

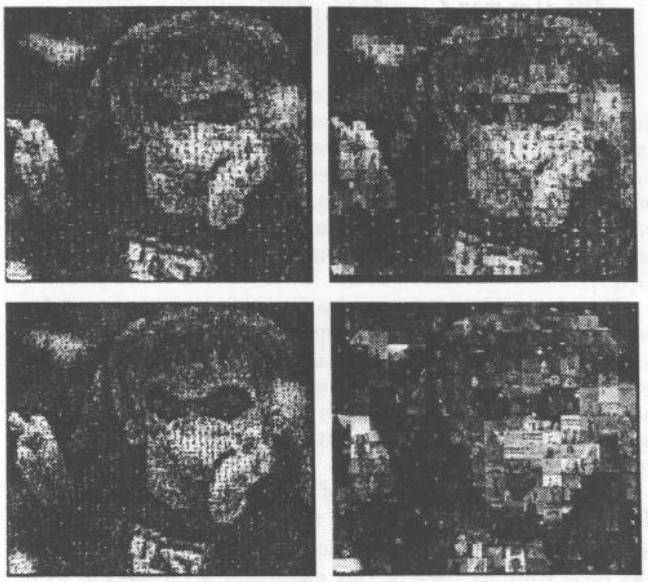

Figure 4: Photomosaics from Figure 1: (clockwise) 20x15, 28x21, 40x30, 60x45

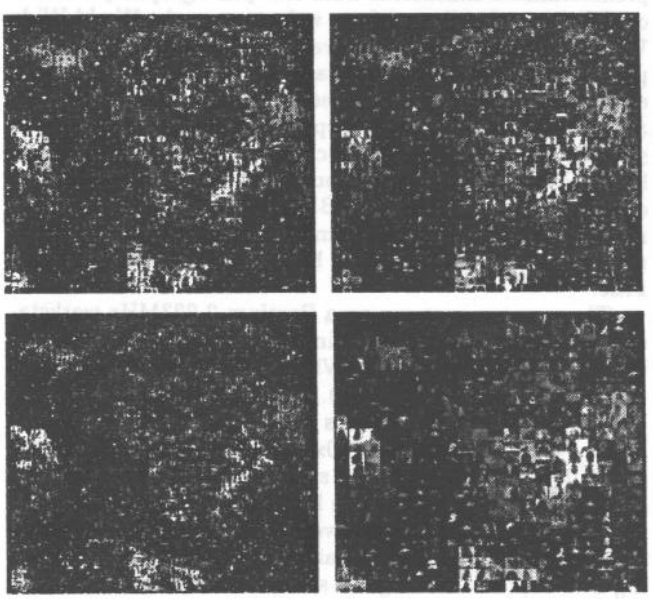

Figure 5: Photomosaics from Figure 2: (clockwise) 16x12, 24x18, 32x24, 48x36

References

[1] CARROLL, L. *Alice's Adventures in Wonderland*, Dover Thrift ed. Dover Publications, Inc., Mineola, NY, 1993.

[2] HOLLANDER, J. Swan and Shadow. In *The Norton Anthology of Poetry*, A. Allison, H. Barrows, C. Blake, A. Carr, A. Eastman, and H. English, Eds., 3rd (shorter) ed. W. W. Norton & Co., NY, 1983, p. 796.

[3] KNOWLTON, K. US Patent No. 4,398,890, Representation of Designs.

[4] KNUTH, D. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1989.

[5] RIVERO, R., AND OTERO, A. LUG: Graphics utilities library. Tech. rep., Mathematics Department, University of Oviedo, Spain, 1993.

[6] SILVERS, R. www.photomosaic.com.

[7] SILVERS, R., AND HAWLEY, M. *Photomosaics*. Henry Holt and Company, Inc., 1997.

[8] SMITH, T. F., AND WATERMAN, M. S. Identification of common molecular subsequences. *Journal of Molecular Biology 147* (1981), 195–197.

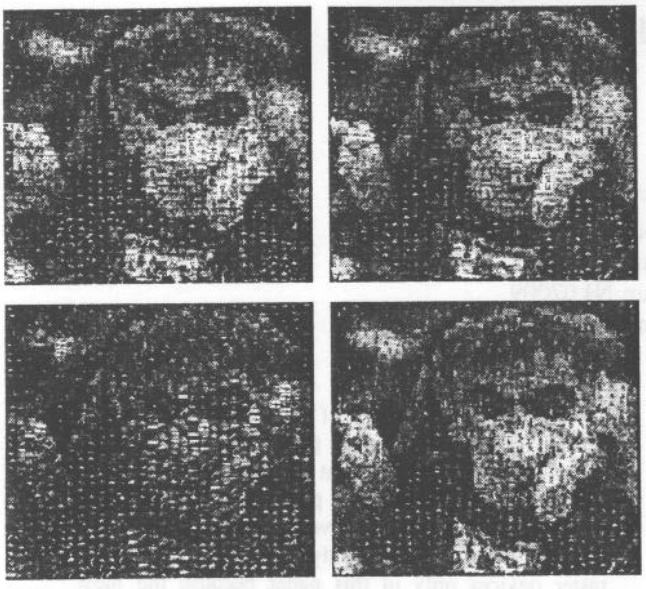

Figure 6: Photomosaics from Figure 3: (clockwise) 128, 357, 739, 1162

Figure 7: The original image

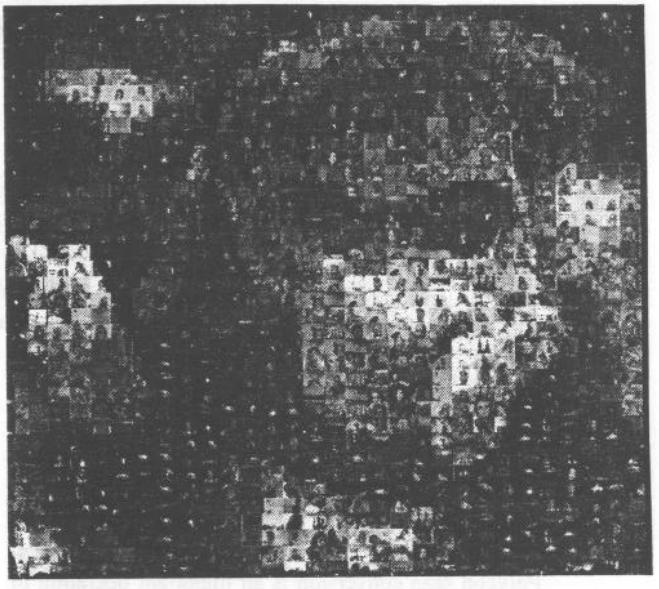

Figure 8: Algorithm I: 32x24

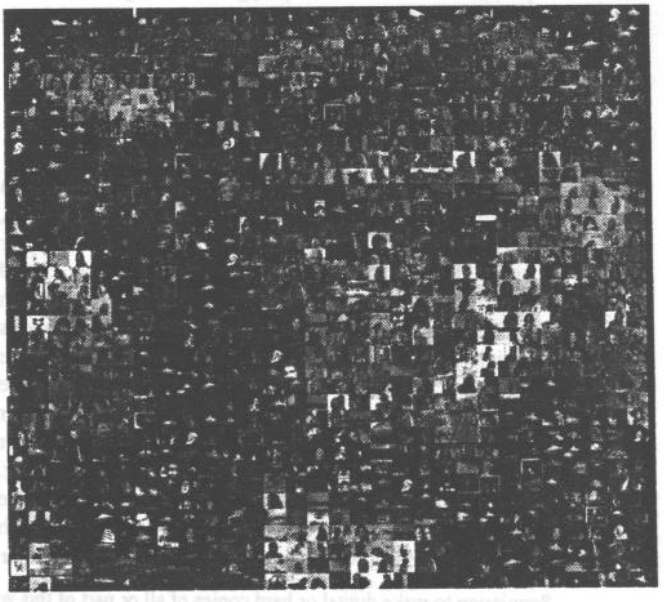

Figure 9: Algorithm II: 32x24