A Randomized Iterative Improvement Algorithm for Photomosaic Generation

Harikrishna Narasimhan
Department of Computer Science and Engineering,
College of Engineering, Guindy,
Anna University, Chennai, India
nhari88@gmail.com

Sanjeev Satheesh
Department of Computer Science and Engineering,
College of Engineering, Guindy,
Anna University, Chennai, India
chandran.sanjeev@gmail.com

***Abstract*—A photomosaic is an image assembled from smaller images called tiles. When a photomosaic is viewed from a distance, it resembles a desired target image. The process of photomosaic generation can be viewed as an optimization problem, where a set of tiles needs to be arranged to resemble a target image. We impose a constraint on the number of times a tile image can be repeated in a photomosaic. A randomized iterative improvement algorithm is used to generate photomosaics and the intermediate results are used to produce interesting animations. We show that the proposed technique is more efficient than genetic algorithm.**

Keywords–Photomosaic; randomized iterative improvement (RII) algorithm; neighborhood search; genetic algorithm (GA); animation; metaheuristic

I. Introduction

Photomosaics are images assembled from smaller images known as tiles. When viewed from a distance, the individual tiles in a photomosaic lose their details and a desired target image is revealed. Photomosaics are digitized refinements of traditional mosaics, which are composed of many pieces of glass, stone or ceramic tiles of similar shapes and sizes to make up larger images.

The concept of photomosaics originated in a computer graphics system called DominiPix [1], in which pictures are constructed from sets of dominoes. Silvers later conceived of the idea of dividing a target image into smaller regions, eventually founding a company that now produces photomosaics on a commercial basis [2], [3]. Further works by Kim and Pellacini [4] and Li and Yuan [5] have contributed much to the problem by exploring a variety of techniques for generating photomosaics. A thorough survey of the digital mosaic generation techniques is provided by Battiato [6].

According to Tran [7], the effectiveness of a photomosaic generating algorithm is determined by three factors: similarity between the photomosaic and original image, granularity of individual tiles and the variety of selected tiles. To increase the variety of tiles, the number of repetition of tiles in a photomosaic must be reduced.

The earliest study of generating animations by rearranging tiles from frame to frame was done by Smith, Liu, and Klien. In [8], they explore the generation of temporally coherent mosaics, wherein the individual frames are chosen to present a coherent sequence in time. The application of evolutionary search techniques to the generation of photomosaic animations can be seen in [9], [10] and [11]. Ciesielski, D'Souza, Berry, and Trist evolved grayscale self referential animation of the target image using genetic algorithm. In [10] and [11], a comparison between the performance of genetic algorithms and genetic programming for photomosaic generation can be seen. In the case of genetic programming, arbitrary placement and rotation of tiles were allowed.

A bottleneck in photomosaic rendering is the search process. In [12], the Antipole Tree data structure is proposed to speed up the search. However, the technique does not consider repetition of tiles. Reuse of tiles is also not a constraint in the evolutionary techniques mentioned earlier. The aim of our work is to generate high quality photomosaics with limited tile repetitions at a reasonable speed.

Animations using mosaic tiles produce interesting effects, where the final picture is revealed through a process of gradual disclosure. Unlike most search and optimization problems, where the intermediate results are unnecessary after completion of the search, photomosaic generation provides the developer with the unique opportunity to make use of all or most of the intermediate steps. This is aptly exploited in our work.

The rest of the paper is organized as follows: We define the problem under consideration in Section II. A randomized iterative improvement algorithm for photomosaic generation is described in Section III. The experiments and results have been presented in Section IV and we present an analysis in section V. We conclude the paper in Section VI.

II. Photomosaic Generation

The problem of photomosaic generation can be defined as follows: Given a target image and a set of tiles, the problem is to arrange the tiles in a two-dimensional grid such that the resultant arrangement resembles the target image as close as possible.

If the dimension of the target image is $M \times N$ and that of the tiles is $X \times Y$, the grid would consists of M / X rows and N / Y columns. Note that each cell in the grid can be occupied by only one tile.

As mentioned earlier, repetition of tile images in a photomosaic needs to be avoided. We therefore impose a constraint on the reuse of tile images in a photomosaic. Let n_m be the maximum number of times a tile image can be used in a photomosaic. Then, photomosaic generation becomes an optimization problem, where a set of tiles has to be arranged to resemble a target image such that no tile is used more than n_m times.

It is obvious that an exhaustive search would be inefficient for this problem. Instead, metaheuristic search algorithms used for optimization problems like scheduling could be applied to solve this problem. In this paper, we investigate the effectiveness of one such algorithm.

III. Randomized Iterative Improvement (RII) Algorithm

A randomized iterative improvement algorithm was proposed by Abdullah, Burke, and McCollum [13] to solve the university course timetabling problem. A total of 11 composite neighborhood structures were used in the algorithm. The use of neighborhood structure added flexibility to the search process and yielded good results. Here, we adopt the randomized iterative improvement algorithm for photomosaic generation.

A. Neighborhood Structures

Through experiments it was found that two neighborhood structures are sufficient to obtain high quality photomosaics at a reasonable speed. The neighborhood structures have been listed below:

N1: Select a single tile position at random and replace the tile with another randomly chosen tile.

N2: Select two tiles positions at random and swap the corresponding tiles.

B. Fitness Evaluation

As in [9], the fitness of a photomosaic is evaluated as the sum of the pixel differences between the target image and photomosaic. The fitness function is given below.

$$\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{3}\left|img_{target}(i,j,k) - img_{mosaic}(i,j,k)\right|$$

Here, img_{target} is the target image and img_{mosaic} is the generated photomosaic image. Also, k = 1, 2, and 3 correspond to the red, green, and blue component of a pixel respectively and for an image *img*, $img(i,j,k)$ is the k^{th} color component of the pixel (i, j).

C. Algorithm Description

The algorithm generates photomosaics with limited reuse of tiles. It starts with an initial feasible mosaic and applies the neighborhood structures during each iteration. If the best neighbor in the iteration is better than the best mosaic encountered till then, it is accepted. Otherwise the exponential Monte Carlo acceptance criterion [14] is applied. The algorithm has been outlined in Fig. 1.

In the algorithm, *mosaic* is a two-dimensional array containing the tile numbers assigned to the photomosaic. The routine initialize(n_m) produces a photomosaic with randomly assigned tiles such that each tile is used at most n_m times. The routine count(*mosaic*, *k*) gives the number of times the tile numbered *k* is used in the photomosaic *mosaic* and the routine fitness(*mosaic*) is used to evaluate the fitness of a given photomosaic.

```
mosaic ← initialize($n_m$)
best ← mosaic
$f_{best}$ ← fitness(mosaic)
while termination criteria not met
    $neigh_1$ ← mosaic
    Select a random tile position $(i, j)$
    Select a tile $k$ such that count(mosaic, $k$) < $n_m$
    $neigh_1(i, j)$ ← $k$
    $neigh_2$ ← mosaic
    $f_1$ ← fitness($neigh_1$)
    Select two random tile positions $(i_1, j_1)$ and $(i_2, j_2)$
    Swap tile $neigh_2(i_1, j_1)$ with $neigh_2(i_2, j_2)$
    $f_2$ ← fitness($neigh_2$)
    if $f_1 < f_2$ then
        neigh ← $neigh_1$
        $f$ ← $f_1$
    else
        neigh ← $neigh_2$
        $f$ ← $f_2$
    end if
    if $f < f_{best}$ then
        mosaic ← neigh
        best ← neigh
        $f_{best}$ ← $f$
    else
        $\delta$ ← $(f - f_{best})/10$
        Generate random number $r \in [0, 1]$
        if $r < e^{-\delta}$ then mosaic ← neigh
    end if
end while
```

Figure 1. Randomized Iterative Algorithm for Photomosaic Generation.

IV. Experiments And Results

We attempted to generate photomosaics for 3 image sizes: 400 × 600, 600 × 800 and 800 × 1000 with tiles of dimension 10 × 10 using both GA and RII algorithm. Tiles were selected from a database of 8227 images obtained from [15] and [16]. The parameter n_m was set to 10. Plots of fitness function against time for the two algorithms are given in Fig. 4.

A. Genetic Algorithm

The basic configuration of the genetic algorithm is similar to [9]. The photomosaic is represented by a two-dimensional matrix containing the tile numbers assigned to the grid. The matrix in row order forms the chromosome. Note that in a chromosome, a tile image can be used at most n_m times. The fitness of the chromosome is evaluated in the same way as in the RII algorithm. The initial population of feasible chromosomes is generated randomly. Table I lists the parameters values. All parameters were finalized after much experimentation. Small population sizes tended provide quicker improvements but were susceptible to local optima.

During reproduction, parent chromosomes are selected using roulette wheel selection and a uniform crossover is

TABLE I. PARAMETERS OF GENETIC ALGORITHM

Parameter	Value
Population Size	20
Crossover Rate	0.7
Mutation Rate	0.005
Elitism Rate	0.1
Max. Generations	10,000
Crossover	Uniform Crossover
Selection	Roulette Wheel Selection

applied. The resultant chromosomes may contain more than the permitted number of repetitions. These repetitions are eliminated by replacing one or more tiles with a randomly chosen tile.

Mutation is carried out by replacing a tile in a chromosome with a random tile such that no tile is used more than n_m times. Each tile in a chromosome has a probability 0.005 of being replaced.

B. Randomized Iterative Improvement Algorithm

A photomosaic generated using the RII algorithm can be seen in Fig. 2 along with the original image. The dimension of each tile used was 10 × 10 and that of the target image was 460 × 640. A total of 2944 tiles were selected from a database of 8227 images. The algorithm was run for 1000000 iterations with each tile used for a maximum of 10 times.

The similarity of a photomosaic with the target image depends on the parameter n_m. In Fig. 5, two photomosaics generated from the same target image are shown, where Fig. 5(a) corresponds to a photomosaic with no repetition (n_m = 1), while Fig. 5(b) corresponds to a photomosaic generated with n_m = 10.

The proposed algorithm was implemented using Matlab 7.1 and was run for images of different sizes. The algorithm was terminated when there was no improvement in the fitness for 200 iterations. In each case, the dimension of the tiles was 10 × 10 and the parameter n_m was set to 10. The results show that the algorithm can generate high quality photomosaics at a reasonable speed.

Photomosaic animations can be produced by making use of the mosaic obtained after each iteration of the algorithm. Whenever there is a significant change in fitness, the corresponding photomosaic is taken as frame in the animation. Few frames of a photomosaic animation produced using our algorithm are shown in Fig. 6.

The code for the algorithm along with sample outputs is available at http://sourceforge.net/projects/pmoz/ and http://code.google.com/p/pamoz/.

V. ANALYSIS

Fig. 3 gives an indication of how the algorithm scales with image sizes. At normal sizes of 400 × 600, 600 × 800 and 800 × 1000, the algorithm scales linearly. We also

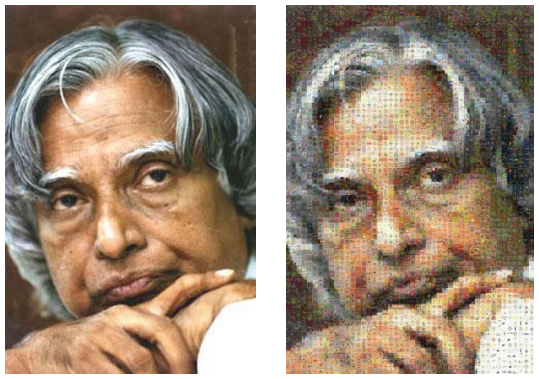
Figure 2. Photomosaic generated by the RII algorithm. The target image is on the left while the corresponding photomosaic is on the right.

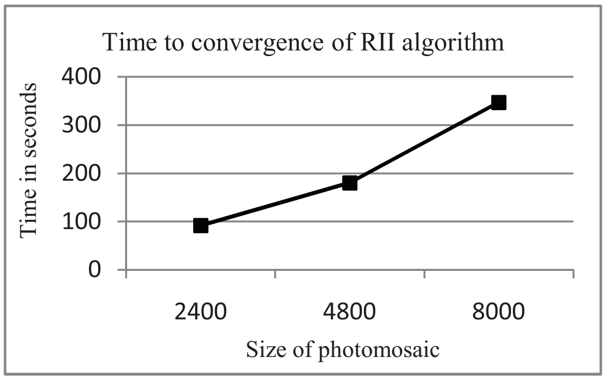

Figure 3. The curve shows the scaling rate of the RII algorithm, plotting the time in seconds against size of the photomosaic.

found that the scaling is a complex function of the quality of the images in the tile pool, the size of the tiles in the pool, and the color distribution in the picture.

Fig. 4 is a plot that shows the rate of convergence of the two algorithms over different image sizes. Every 5 seconds, the fitness of the current mosaic (in the case of RII) and the fittest member of the population (in the case of GA) are noted down. The number of evaluations or the number of iterations required could not be used for comparison. The evaluation function is much simpler in the case of RII. Majority of the tiles remain unaltered and the pixel-wise difference of only the altered tiles need to be measured.

The curves in Fig. 4 are a plot of the averaged fitness across 30 runs at the same time. In the case of the GA, the average of only the fittest individuals is plotted. The curve also shows a 95% confidence interval for both the RII algorithm and the GA. It is seen that the deviation is very small, and is much better than the GA.

It is evident from the provided plot that the RII algorithm is sufficient for the problem of photomosaic generation. We hypothesize that in cases where there is no good genotype-phenotype mapping, smaller populations or even population less metaheuristics should suffice. The chromosome representations used here and in [11] in particular do not seem to lend themselves to the mutation and the

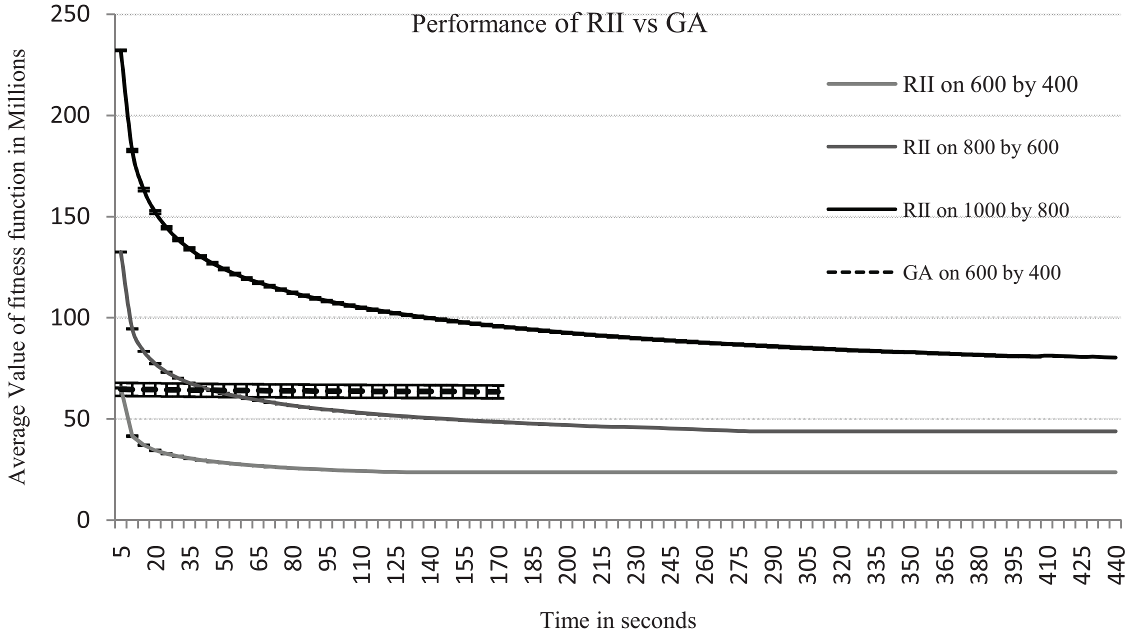

Figure 4. Plot of fitness function against time for GA and RII algorithm.

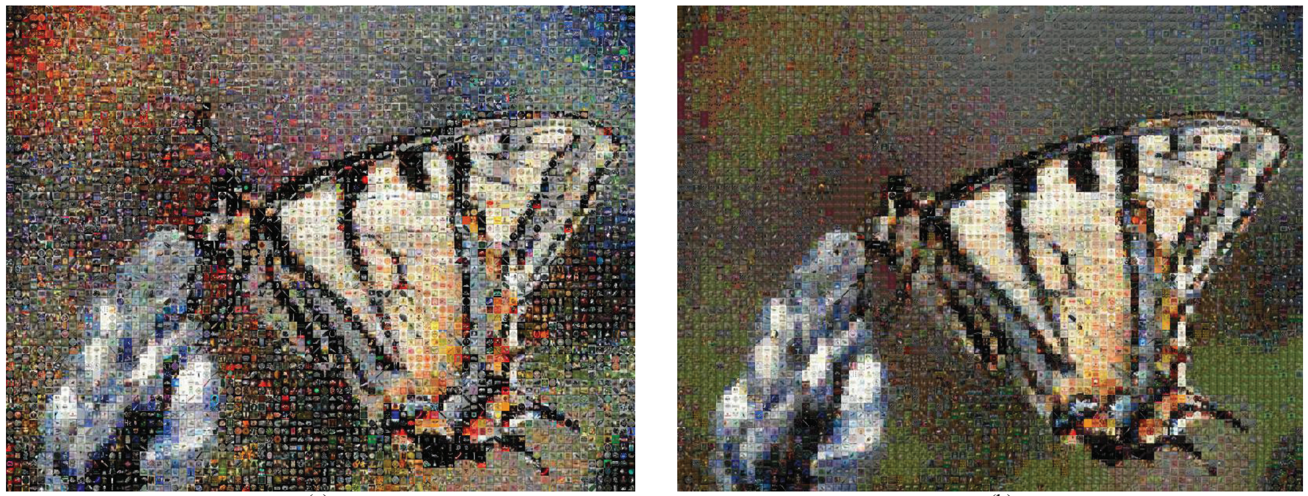

Figure 5. Photomosaics generated from the same image: (a) $n_m = 1$ (b) $n_m = 10$.

recombination operators proposed to produce better quality photomosaics.

The genetic algorithm based approach converges very slowly and has a very poor fitness even after the parameters were optimized. In an equivalent time, the RII had advanced to mosaics, where the basic features of the target image are easily recognized.

VI. Conclusions

We have implemented a metaheuristic algorithm for photomosaic generation with limited repetition of tiles. A randomized iterative improvement algorithm with suitable neighborhood structures has been used to creating high quality photomosaics. The performance of the proposed algorithm has been compared with that of genetic algorithm implemented in a previous work and it has been shown that RII is more effective with its approach. Using the intermediate results of the algorithm, interesting animations have also been produced.

Future direction of work would be to come with a better fitness function for the problem. In this paper, the fitness of a photomosaic is calculated using a pixel-by-pixel difference between two images. Another interesting direction of work would be to look at better chromosome representation. In this case, only a number indicating a unique tile is used. A better

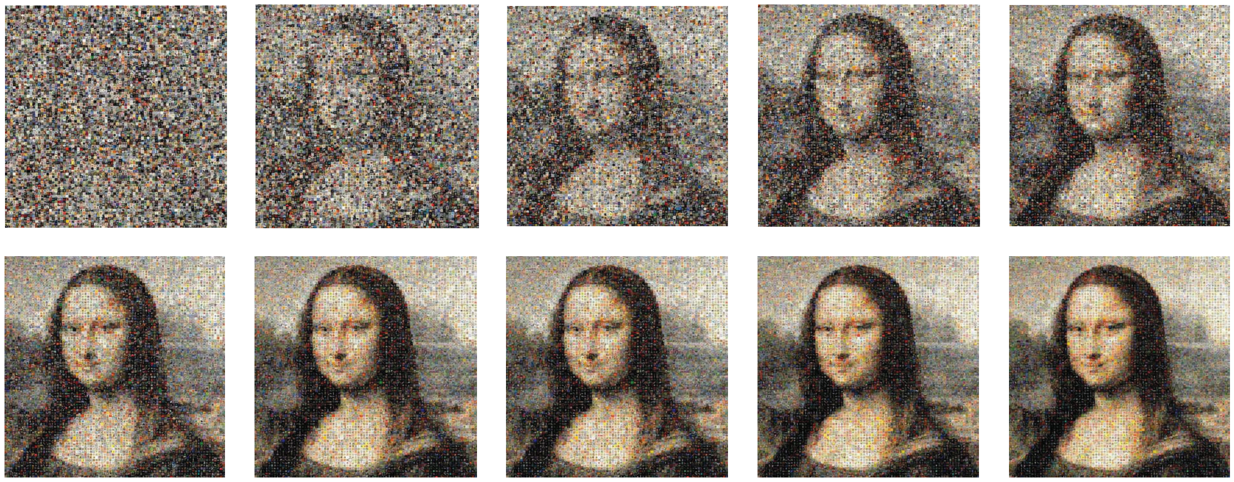

Figure 6. Sample Frames from a Photomosaic Animation

representation should ideally be more indicative of the feature of the tile, therefore making the fitness evaluation quicker. At the same time, a delicate balance needs to be struck so that building, mutating and recombining the chromosomes do not become computationally very expensive. We also intend to test the performance of the algorithm when transformations like rotation and flip are allowed on individual tiles. A way of comparing photomosaics generated by algorithms described in Section I would also be useful. The performance of other metaheuristic search algorithms in photomosaic generation would also be investigated.

ACKNOWLEDGMENT

The authors wish to thank Dr. S. Sridhar for providing valuable suggestions on photomosaic generation.

REFERENCES

[1] K. Knowlton, “DominoPix,” US Patent No. 4398890, Representation of Designs, 1983, http://www.metron.com/DominoPix, Visited 25-Jul-2009.

[2] R. Silvers and M. Hawley, “Photomosaics,” Henry Holt, New York, 1997.

[3] R. S. Silvers, “Digital composition of a mosaic image,” US Patent No. 6137498, Oct. 2000.

[4] K. Kim and F. Pellacini, “Jigsaw image mosaics,” Proc. ACM-SIGGRAPH International Conference on Research and Development in Graphics and Interactive Techniques, San Antonio, Texas, Jul. 2002, pp. 657–663.

[5] X. Li and Y. Yuan, “Artistic mosaic generation,” Proc. 3rd International Conference on Image and Graphics, Hong Kong, Dec. 2004, pp. 528–531.

[6] S. Battiato, G. Di Blasi, G. M. Farinella, and G.A. Gallo, “Survey of digital mosaic techniques,” Proc. Eurographics Italian Chapter Conference, 2006.

[7] N. Tran, “Generating photomosaics: an empirical study,” Proc. 1999 ACM Symposium on Applied Computing San Antonio (SAC '99), Texas, United States, Feb. 28 – Mar. 02, 1999. SAC '99, ACM, New York, NY, pp. 105-109.

[8] K. Smith, Y. Liu, and A. Klein, “Animosaics,” Proc. ACM-SIGGRAPH International Conference on Research and Development in Graphics and Interactive Techniques, Los Angeles, CA, Jul. 2005, pp. 201–208.

[9] V. Ciesielski, D. D’Souza, M. Berry, and K. Trist, “Generation of self-referential animated photomosaics,” Proc. ACM Multimedia (MM ’07), Augsburg, Bavaria, Germany, Sep. 2007, pp. 489–492.

[10] G. Wijesinghe, S. Mat Shah, and V. Ciesielski, “Grid vs. arbitrary placement of tiles for generating animated photomosaics,” Proc. 2008 IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, China, Jun. 2008, pp. 2734-2740.

[11] S. Mat Shah, V. Ciesielski, D. D’Souza, and M. Berry, “Comparison between genetic algorithm and genetic programming performance for photomosaic generation,” Proc. Seventh International Conference on Simulated Evolution And Learning (SEAL ’08), LNCS 5361, Dec. 2008, pp. 259–268.

[12] G. Di Blasi and M. Petralia, “Fast photomosaic,” Poster proc. The 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2005, Plzen, Czech Republic, Jan. 31 – Feb. 4, 2005, p. 15-16 .

[13] S. Abdullah, E. K. Burke, and B. McCollum, “Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem,” in Metaheuristics - Progress in Complex Systems Optimization, Springer Operations Research / Computer Science Interfaces Book, K.F. Doerner, M. Gendreau, P. Greistorfer, W.J. Gutjahr, R.F. Hartl, and M. Reimann, Eds., 2007, pp. 153-169.

[14] M. Ayob and G. Kendall, “A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine,” Proc. Fourth International Conference on Intelligent Technologies (InTech’03), Thailand, Dec. 2003, pp. 132-141.

[15] Mosaic Creator, http://www.aolej.com/mosaic/index.html, Visited 24-Jul-2009.

[16] T. Gray, “Making photo mosaics,” Wolfram Blog. http://blog.wolfram.com/2008/05/02/making-photo-mosaics/, Visited 24-Jul-2009.