A Hillclimbing Approach to Image Mosaics

Chris Allen

Faculty Sponsor: Kenny Hunt, Department of Computer Science

ABSTRACT

This paper presents a hillclimbing approach to image mosaic creation. Our approach significantly differs from previous approaches by being non-deterministic and allowing for tile overlapping, which produces a collage effect. Results show that, if refined, the hillclimbing approach to mosaic creation may have the potential to provide a viable alternative to other techniques.

INTRODUCTION

Mosaics are images that are created by arranging many smaller components, called tiles, on a canvas. This art form has been around since ancient times and has taken a variety of forms over the years. Recently, many have researched ways to automate the creation of mosaics through the use of computers. The Simulated Decorative Mosaic [Hausner, 2001] algorithm generates mosaics that are composed of many small, uniformly colored tiles. Photomosaics [Silvers and Hawley, 1997] are created by placing rectangular tile images in a contiguous fashion on a canvas such that they resemble the original picture at far distances. Most important to our project, the Jigsaw Image Mosaic [Kim and Pellacini, 2002] generates mosaics from many smaller tile images, which are placed at arbitrary points, rotated by an arbitrary angle, and deformed in non-uniform but visually appealing ways.

In this paper, we introduce a new approach to image mosaics that uses an optimization heuristic in the form of a hillclimbing algorithm. Like the Jigsaw Image Mosaic, our mosaics are created from arbitrarily shaped tile images that are transformed to fill objects within a target scene. This approach deviates from previous work in two ways: first, it relies on a random initialization step, thereby rendering the creation of image mosaics a non-deterministic process and second, we allow tiles to overlap, which creates a collage effect in the output image.

FORMAL PROBLEM STATEMENT

The problem we set out to solve can be stated thus: Given a target image and a set of arbitrarily shaped image tiles♥, find an optimal arrangement of *N* tiles such that they resemble the target picture as closely as possible. Tiles in this instance can be placed anywhere on the canvas, and can be rotated or scaled by arbitrary values. Formally, we define a four dimensional vector *A* as an arrangement for tile *i* in the set of *N*:

$$A_i = (XPosition_i, YPosition_i, Scale_i, Rotation_i), i \in 1...N$$

An output image can then be constructed from the arrangement vector, *A*, by sequentially painting each tile on an initially empty canvas using the specified translation, rotation and scaling parameters. (Note that order of tile placement is significant in this definition since tiles that occur early in the arrangement may be obscured by tiles that occur later in the arrangement.) For the purpose of this paper, we define a function *I* that renders an arrangement vector, creating a mosaic:

$$I(A) = Mosaic$$

In order to evaluate the quality of a given tile arrangement *A*, a function, *Diff(Target, I(A))*, is defined that gives the difference between the target image and *I(A)*. The problem is then to search for a tile arrangement *A* such that

♥ It should be noted that within the context of this paper *tile* will refer to an arbitrarily shaped image.

Diff(Target, I(A)) is minimized. To construct a mosaic using *N* tiles from a library of *M* tiles, we must choose *N* tiles and position, scale, and rotate each of them to minimize the *Diff* function. The magnitude of the search space is seen to be O((*M*-*N*)!) and is therefore not amenable to any exhaustive search technique.

THE HILLCLIMBING ALGORITHM

We approach mosaic creation as a function minimization problem. A hillclimbing heuristic is used to achieve the goal of minimizing the *Diff* function discussed above. A heuristic search is particularly suitable for our problem, since the enormity of the search space renders any brute-force search computationally prohibitive.

Hillclimbing algorithms, in general, work by starting at a random point in an *N*-dimensional search space and moving in the direction of the maximum gradient until it converges to an optimal point.[♠] One such hillclimbing technique finds optimal solutions by considering perturbations of each of the current solution's *N* parameters and accepting the perturbations that result in the greatest minimization of error. For this paper, we will assume that at each iteration of the search only two perturbations (+/-) of each parameter are considered, thus giving a neighborhood size of 3^*N* at each stage of the search. In order to reach an optimal solution, hillclimbing algorithms must have some way to evaluate a given solution in order determine which parameter alterations to keep; this function is called an evaluation function. The general hillclimbing algorithm works as follows:

1. Generate a random solution.
2. For each parameter *p* in the solution:
    a. Alter *p* by +*dp* and –*dp*.
    b. Evaluate each alteration. If either alteration improves the solution, keep it; otherwise, revert to previous parameter settings.
3. If an acceptable solution has been reached, terminate execution; otherwise, go to step 2.

In order to apply this approach to our problem domain we first segment the target image into distinct objects, or regions, and find a solution for each region. In our case, a region is characterized by a grouping of connected, similarly colored pixels. Decomposing the target image reduces the complexity of the search by allowing us to find local solutions for regions that are spatially coherent and similar in color. This step is currently done manually but future extensions would allow this to be an automated process.

After the target image has been segmented, the main objective is then to pack each region with tiles such that the arrangement *structurally* resembles the target region, since the region is essentially monochromatic. The global solution for a target image is thus roughly[♣] given by the union over the solutions for each region:

$$globalSolution \approx \bigcup_{x} region_x$$

Working within the hillclimbing framework, a solution tile arrangement for a given region is initialized with random placement, rotation, and scaling for each tile. Each parameter (position, rotation, scale) of every tile is then perturbed. If a given perturbation decreases error, it is accepted; if not, the tile parameters remain the same. This results in 3^4 possible combinations for a single tile perturbation, since the value of each of the four dimensions can increase, decrease or remain unaltered.

The evaluation function (called the *Diff* function above) for our problem compares the structure of a region in the target image with the structure of a rendering of a candidate tile arrangement *A*. In this process, a candidate solution *A* is rendered to generate an image *I(A)*. The evaluation function then compares the target region with *I(A)*. A candidate solution that closely resembles the target region yields a low error, whereas candidate solutions that are morphologically dissimilar to the target region generate a high error.

[♠] Unlike an exhaustive search, the point of convergence may, however, be a locally optimal solution rather than a globally optimal solution.

[♣] It is not quite the union, since regions may overlap.

Formally, the structural difference between the rendered solution, *I(A)*, and the target region is given by the percent of pixels in *I(A)* that are part of the rendered region but do not correspond to a pixel that belongs to the target region (note that the *areaOf* function returns the area of the segmented region in the argument image):

$$Diff(Target, I(A)) = \frac{(\sum_{x,y} PixelDiff(Target, I(A), x, y))}{areaOf(Target)}$$

Accordingly, we need a method to measure the difference (or error) between two pixels. For a given (x, y) position, an error is produced if the pixel in that location for either image is part of the region, whereas the pixel in that location for the other image is not part of the region. This can be stated formally in terms of an exclusive-or operation on both pixels (where a given pixel is *true* if it is part of the region and *false* if it is not):

$$PixelDiff(Target, I(A), x, y) = Target_{x,y} \oplus I(A)_{x,y}$$

While the above evaluation function ignores color values when comparing images, color information is taken into account in the initialization step. Whereas initial tile placement, rotation and scaling are random, the tiles themselves are selected according to their resemblance to the portion of the target object on which they will be placed. For a given placement, rotation, and scaling, a tile is chosen that minimizes the root-mean-squared error (RMS error) for that subregion. The RMS error between two same sized images is given by the following equation (note that the *numBands* function returns the number of color bands in the argument image):

$$RMS(Img1, Img2) = \frac{\sum_{x,y,band} \sqrt{(Img1(x, y, band) - Img2(x, y, band))^2}}{areaOf(Img1) * numBands(Img1)}$$

IMPLEMENTATION

All code for this project was implemented in Java. Major classes include *Mosaic*, which contains the main mosaic algorithm, and *Collage*, which is a representation of a mosaic solution. In order to represent a distinct region on an image canvas, our implementation requires image formats that support an alpha channel. Region identification was accomplished by segmenting each tile image and distinguishing region pixels from background pixels by asserting the alpha channel of background pixels. In our case, all images were GIF formatted.

RESULTS

Figure 1 shows a diagram of the entire process of mosaic creation. The original image is first manually decomposed into twelve regions. A solution is then found for each region, and the regions are combined to form a global solution to the target image. Results for our project vary based on the complexity of the target scene. However, for most scenes with clearly demarcated regions it performed reasonably well.

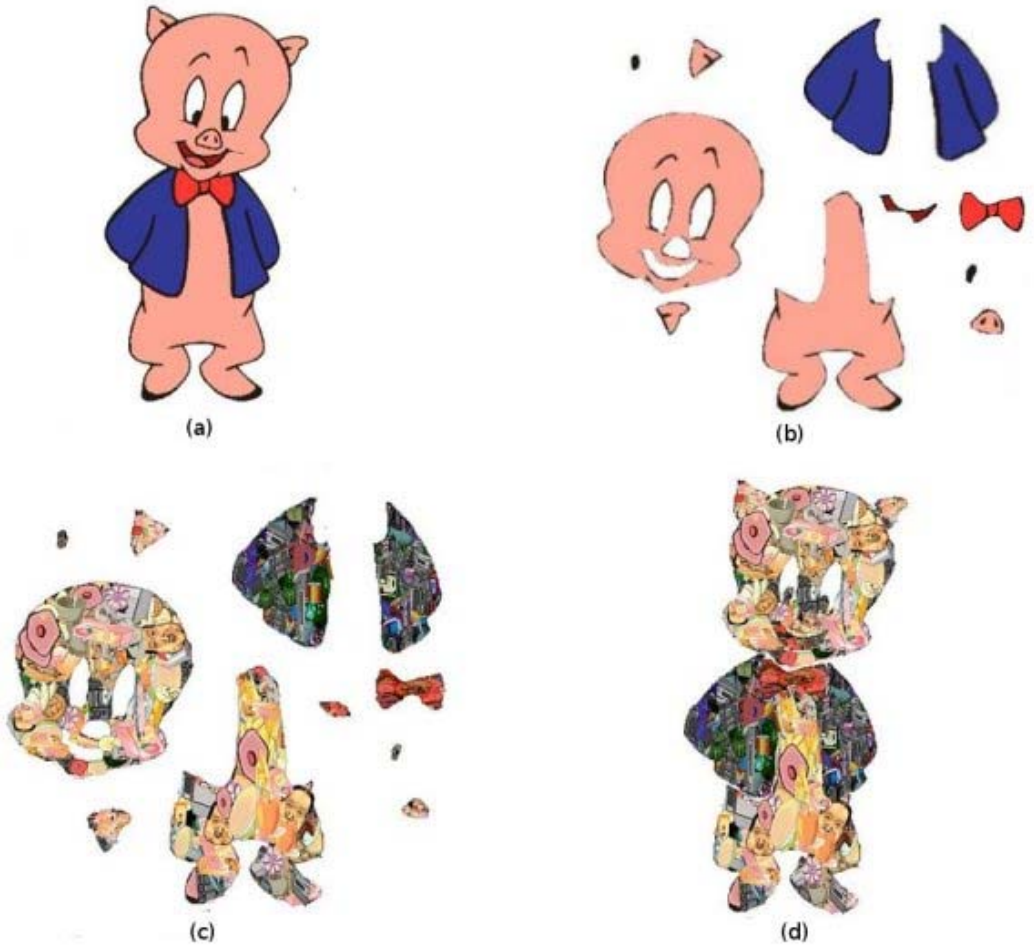

Figure 1: The mosaic creation process. The target image (a) is first segmented into multiple regions (b). A mosaic solution is then found for every region (c). Finally, the mosaic solutions for each region are layered to produce the final result (d).

Since it is impractical to display error minimization statistics for every region in a scene, in Figure 2 we give some idea of the performance of our algorithm by presenting error minimization statistics for a rather large region taken from Figure 1.

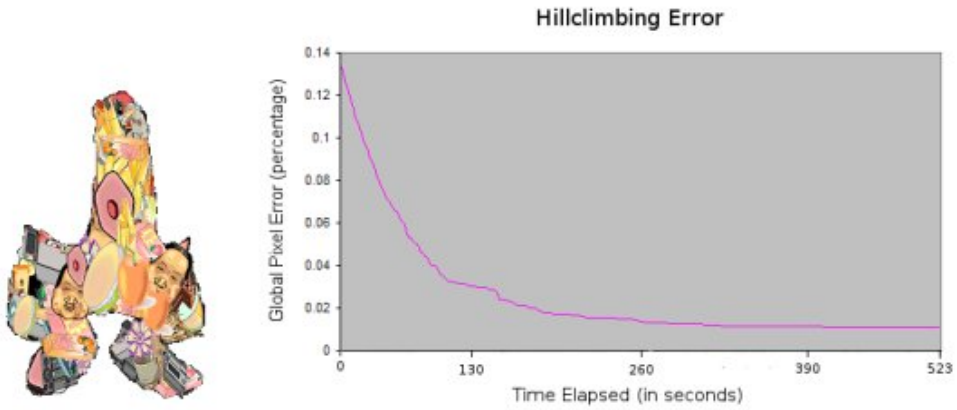

Figure 2: A mosaic solution for an individual region (left) and the hillclimbing error minimization statistics for that region. As shown in the graph, global error for that region is reduced from 14% to 1% in 523 seconds (the test was run on a Pentium M 1.6 Mhz machine).

Figure 3 (appendix) shows an additional mosaic solution for a text target image.

LIMITATIONS

Hillclimbing algorithms are not guaranteed to converge to globally optimal solutions. However, this has not proven to be an obstacle in our case, as nearly all tests have yielded visually acceptable solutions. This is likely due to the existence of many acceptable local minima within the search space of our problem.

The most glaring limitation of our mosaic algorithm is the inability to represent fine detail within the target scene. Since the tile arrangement at the time of tile selection is often drastically different from the final arrangement, our algorithm assumes some degree of homogeneity with respect to color in each target region. This way, a tile chosen in a certain subregion will not look abnormal when placed at another point in the target region. However, this assumption is reasonable if the scene regions are accurately segmented.

In addition, our implementation relied on a significant amount of manual effort. Decomposition of the target scene and the layering of the mosaic scene were both done manually.

FUTURE WORK

Tile selection was easily the most intractable problem in our approach to image mosaics. Although our RMS-based selection generally picks suitable tiles, difficulty arises when tiles are initially placed outside of the target region. We attempted to solve this problem by restricting the initial center placement of tiles to pixels that are known to be part of the target region; however, this does not always ensure better results, which is evidenced by the number of anomalous tiles in our mosaics. Tile selection might be improved by using the centroid of the tile object, rather than the center of the canvas, to determine initial placement. Another solution might be to delay tile selection until all tiles are completely on the target object.

REFERENCES

[1] Kim and Pellacini. Jigsaw Image Mosaic. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press/ACM SIGGRAPH, 657-662, New York, 2001.

[2] Silvers and Hawley. *Photomosaics*, New York: Henry Holdt, 1997.

[3] Hausner. Simulating Decorative Mosaics. In Proceedings of ACM SIGGRAPH 2001, ACM Press/ACM SIGGRAPH, E. Fium, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 499-410, New York, 2001.

APPENDIX

Figure 3: An example of a mosaic (below) created using a text target image (above).