Video Mosaics

Allison W. Klein[1] **Tyler Grant**[1] **Adam Finkelstein**[1] **Michael F. Cohen**[2]
[1]**Princeton University** [2]**Microsoft Research**

Abstract

We present a method for creating a *video mosaic*, a two-dimensional arrangement of small *source* videos (tiles) that suggests a larger, unified *target* video. We develop a distance measure to assess the match between source and target based on average color and also three-dimensional wavelet decomposition signatures in the YIQ color space. We also introduce a dynamic programming algorithm that automatically chooses the smaller tiling sub-sequences from a large collection of candidate source video sequences to best match the target video. After the selection process, the color in the tiling videos is automatically adjusted to better suggest the target video. Finally, our method also supports the use of different tiling shapes to create an additional level of visual interest.

Keywords: Animation, Temporal Aliasing, Video, Non-photorealistic Rendering

1 Introduction

Many artists have exploited people's fascination with layered imagery – images composed of individually recognizable elements. At the end of the 16^{th} century, Giuseppe Arcimboldo created human portraits from collections of semantically similar objects such as fruits, birds, or fish [Mataev and Mataev 2001]. In the early 20^{th} century, Arthur Mole took photographs in which he used thousands of individuals as human pixels to form images of famous people or objects [Zucman 2001]. The artist Chuck Close paints huge, recognizable human portraits composed many small abstract tiles [Wye 1998].

A visually layered medium has recently emerged in the digital domain: *image mosaics* forms a large image by arranging (and sometimes adjusting the colors in) a collection of small images (e.g. Finkelstein and Range [1998], Silvers and Hawley [1997]).

Our goal is to extend image mosaics into a third dimension (time) by creating *video mosaics* out of a collection of small, tiling videos. (See Figure 1 for an example frame from a video mosaic, as well as the accompanying video clips for the dynamic version.) Because video mosaics are visually intriguing and can contain multiple layers of meaningful images, they are applicable for a variety of artistic and commercial uses.

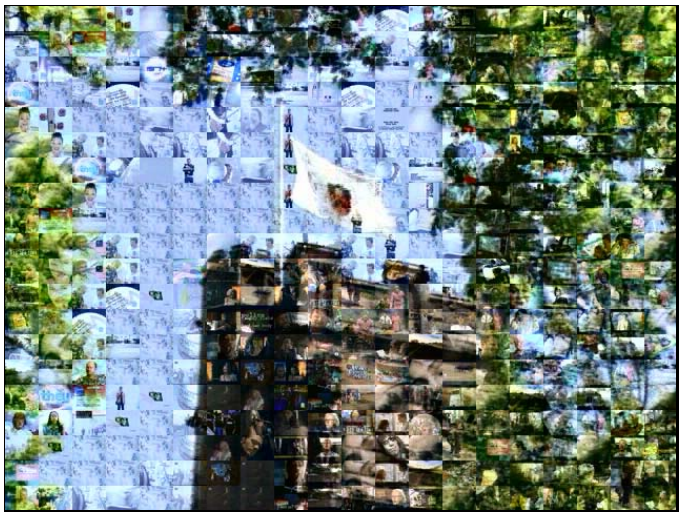

Figure 1: A frame from a video mosaic

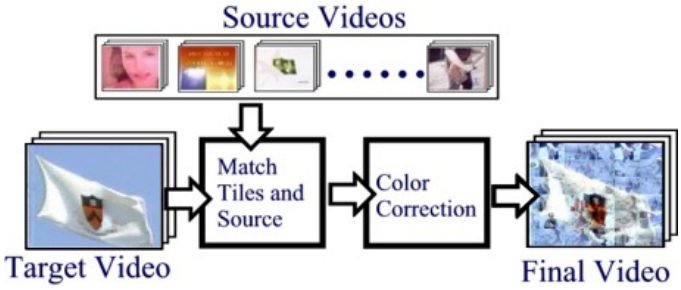

Figure 2: Schematic of video mosaic creation

The challenges we address are:

- Finding a suitable spatiotemporal arrangement of tiling videos that best match a given target video
- Maintaining a degree of temporal coherence
- Working within the limited spatial resolution available in video
- Performing a suitable color correction on selected tiles to mimic the target

Figure 2 shows a high-level schematic of our process. We take as inputs a *target* video and a collection of other *source* videos. We then match the source videos to the target in predefined regions, or *tiles*, in the 2D image plane and along the time axis. The output from this step is then color corrected to achieve a better visual balance between the large (target) and small (tiling) images.

The crux in creating video mosaics lies in the matching step: specifically, how to resolve the inherent tension between achieving good image matches and maintaining frame-to-frame coherence. Randomly assigning subsequences from the tiling videos to the

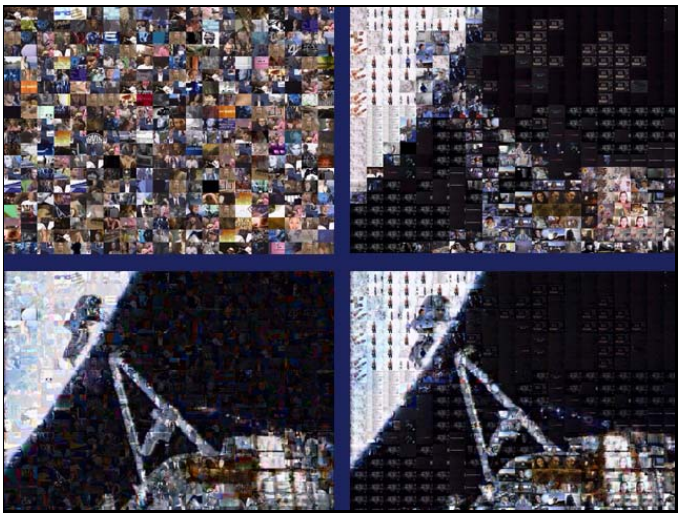

Figure 3: Random (left) vs. selected tiles (right), and before color correction (top) vs. color corrected (bottom). Note that the random tiles are almost completely washed out in the color corrected version since the original selection of tiles has no resemblance to the target.

target video may not provide visually satisfying results because there is little color or edge matching, and so the color correction step can totally overwhelm the original source videos. (See Figure 3 and associated videos.)

Therefore, we have chosen to build an algorithm capable of efficiently selecting subsequences that best match the target. An obvious solution might be to simply perform frame-by-frame matches. First, pick a measure of image distance. Next, for each tile area in a specific frame of the target video, calculate its distance with every frame in the tiling collection, and then pick the closest frame. Such an approach will cause any individual frame of the resulting video mosaic to match the target as well as possible. However, if the target video area is almost static, the resulting video may contain the same source frame over and over. Even worse, if the target video area is dynamic, different tiling frames will be chosen for each target frame, and, at 30 frames per second, the resulting video mosaic will be noisy and unpleasant.

A better solution is to pick out the best *subsequences* from the collection of tiling videos to provide both visual matching and temporal stability. Searching through the entire space of possible subsequences and possible alignments with the target tile sequence is a computationally challenging global optimization problem, and one of the technical contributions of this paper is an algorithm to efficiently solve this optimization problem. We also present an efficient method for measuring the differences between two image sequences. Based on wavelet coefficients, the measure is sensitive to color, positional, and temporal information.

General contributions of this paper are:

- A new medium for layered video imagery.
- Simple and flexible difference measures between video sequences.
- A framework for efficiently finding optimally matching video sub-sequences while maintaining coherence.

The paper organization is as follows: Section 2 discusses related work. Section 3 presents our solution to the optimization problem, while Section 4 presents some video mosaics created using our method. Finally, Section 5 concludes with a discussion and some possible future work.

2 Related Work

Our work is most directly related to prior work in image mosaics, non-photorealistic video, and algorithms for gene sequence comparisons in computational biology.

The work of Finkelstein and Range [1998] extends traditional halftoning methods and the work of Ostromoukhov and Hersch [1995] with a fully automatic process for color correcting an image mosaic to more closely resemble its target. Because our system uses this color correction technique, we discuss it in more detail in Section 3.3. Selecting the best source image for each tile of the destination image is a key challenge in creating image mosaics. Finkelstein and Range use the wavelets based algorithm of Jacobs *et al.* [1995] as one solution. We have a similar challenge per video frame, and we use a variation of this technique, discussed in Section 3.

As noted earlier in [Finkelstein and Range 1998], image and video mosaics may be thought of as a form of non-photorealistic rendering using images as opposed to brush strokes. Litwinowicz [1997] and later Hertzmann and Perlin [2000] described algorithms for painterly rendering of video, using optical flow to make brush strokes follow pixels, thereby attempting to provide frame-to-frame stroke coherence while minimizing the "shower door" effect. In contrast, video mosaics exploit the shower door effect: the tiled nature of the result is part of the appeal of this form of layered imagery. However, as with painterly rendering of video, we must provide a form of frame-to-frame coherence; we favor longer matching video clips over shorter clips. It might be interesting to try tilings that somehow reflect and move with important features in the target video, but we have not explored this here.

Finding good sequence matches over the entire set of possibilities is the core computational problem for us. For their video textures work, Schödl *et al.* [2000] find pairs of frames within a sequence that are good matches, using an L_2 distance metric. These pairs are then used as jumping points to generate a continuous, varying stream of images from that single original sequence. In contrast, we seek matches between portions of a single target video and lots of possible source videos. Computational biology algorithms for gene sequence comparison inspired our dynamic programming solution. The Needleman-Wunsch [1970] algorithm calculates optimal global alignments between two gene sequences, treating them as strings in which each nucleotide is a character. A score is assigned for each character-to-character comparison – positive scores for exact matches and some substitutions, negative scores for other substitutions, deletions, and the insertion of spaces. The Smith-Waterman algorithm [1981] is similar, but sequence comparisons are local, not global [par 2001]. Among the differences between these algorithms and ours is that the Needleman-Wunsch and Smith-Waterman algorithms are only trying to match pairs of sequences, while we are trying to match many sequences to a single query sequence. In addition, we cannot allow a target frame to be deleted or matched to a space (an empty frame), both acceptable operations in the other two algorithms.

Finally, our work should not be confused with video mosaics from the computer vision community (e.g.[Szeliski 1996]). These papers are concerned with stitching together many videos of a single scene to produce a high-resolution video with a wide field of view. This stitching process is designed to minimize or smooth out individual differences in the components videos. In contrast, we benefit from the individual differences in our tiling videos by stitching them together to recreate an entirely different video.

3 Creating a Video Mosaic

Referring back to Figure 2, creating a video mosaic involves the following steps:

1. Select a target video, T.
2. Select a corpus of video sources, S.
3. Select a tiling structure. This may be a simple as an $n \times n$ grid of rectangles or a more complex tiling pattern.
4. **For each** tile of the target video:
   (a) **For each** frame of the target video:
      i. Select the best frame S_j from the source videos matching the current tile in the current frame of T.
      ii. Color correct S_j to mimic the current tile in the current frame of T.
      iii. Paste S_j into the tile for output.

Obviously, the selection of the *best* frame from the source videos for each tile of the output video is a key step in the process. The two relevant questions are: how should we define *best*, and how should we efficiently find the best frame from amongst all possible source frames? We will address each of these questions in turn. We also will discuss the color correction step in some detail.

3.1 Defining the Best Source Frame

There are a number of considerations in defining the best match between a tile in the target video clip and all possible sub-sequence arrangements of source videos, including:

1. Average color
2. Distribution of color
3. Temporal (motion) aspects
4. Temporal coherence in the tiles

The first three criteria refer to how well does a possible frame from the source videos match the tile area of the target. The fourth criterion indicates that we may want to choose the sequential source frames for a particular tile over time to avoid having the tile exhibit lots of jumping. In other words, each tile should display a *video*, not simply a random series of frames. Balancing these considerations while delivering an efficient solution to this complex optimization problem is at the heart of this paper.

Color Matching

The first two criteria require choosing a measure of the color difference between any possible source frame and a given tile of a target video frame. A simple choice would be an L_2 norm (i.e., sum of squared pixel differences). However, the cost of pixel-wise comparisons between each tile and every source frame is very expensive. (Our initial experiments indicate that the cost of computing video comparable in size to those found in Section 4 would take on the order of *months* using L_2.)

Another option is to first down-sample the images and then use the L_2 measure on these lower resolution images. However, this has the drawback that small but important details may be lost. Instead we use a slightly modified implementation of the fast image querying system created by Jacobs *et al.* [1995]. Given a rough hand drawing, Jacobs *et al.*were able to quickly compare a large corpus of possible matches to find the image the user was seeking. Our problem is similar: we are given a target tile and want to rank possible matches from amongst the corpus of source frames.

The algorithm can be described as follows: first, we perform a standard two-dimensional Haar [Stollnitz et al. 1996] wavelet decomposition on each color channel of every image in our corpus of source videos, S. Then, for each color channel, we store just the overall average color plus the indices and signs (+ or -) of the 30 largest-magnitude wavelet coefficients in that channel. We use 30 coefficients because this value provides good results without using creating too high a burden in terms of memory usage; this value is also relatively close to the value that Jacobs *et al.*found to be optimal for scanned queries (40 coefficients). The indices for all of the database images are then organized into a single data structure optimized for searching. This data structure is a set of six arrays, called the *search arrays*, with one array for every combination of sign (+ or -) and color channel (such as Y, I, and Q).

As an example, let D^c_+ denote the "positive" search array for the color channel c. Each element $D^c_+[i,j]$ of this array contains a list of all images having a large positive wavelet coefficient $[i,j]$ in color channel c. Similarly, each element $D^c_-[i,j]$ of the "negative" search array points to a list of images with large negative coefficients in c. The search arrays are created as a preprocess for a given corpus of source videos and then stored on disk.

Constructing these search arrays will enable us to efficiently measure the "distance" between a given tile in our target video and each source video frame by allowing us to quickly compare how many significant wavelet coefficients the query has in common with potential targets. We perform the same wavelet decomposition described above on the target tile, again keeping just the overall average color plus the indices and signs of the largest 30 coefficients in each color channel. We construct a distance measure by first computing the differences between the target tile's average intensity in each channel c and those of the source video images. Next, for each of the 30 non-zero, truncated wavelet coefficients in the target tile, we decrement the distance score for all source videos that share a matching coefficient. By only counting matches (rather than mismatches) we can perform the count using a list traversal that only touches the relevant images.

We compute the distance measure in each of three color channels, YIQ (luminance plus two chromaticity channels). Video signals are typically encoded in YIQ to allow efficient compression that takes into account the differing perceptual sensitivity of the human visual system to each channel. We use this fact as well to weight the relative importance of each channel with ratios 8:3:1, the ratios of bandwidths assigned to these channels in NTSC broadcast television. There are significantly different ranges and variations of the value in each of the three YIQ channels, and also between the average colors and wavelet counts. We thus determine 6 individual normalization factors for the average differences and wavelet match counts in each of the three channels. These are based on the observed standard deviations in a sampling of video source; σ_{aY}, σ_{aI}, σ_{aQ} are the observed standard deviations of the average colors, while σ_{wY}, σ_{wI}, σ_{wQ} are the observed standard deviations in the number of wavelet matches for each color channel. The final weighting of the contributions also includes a user set value, W_a, to indicate the relative importance they want to give to average color vs. the structural aspects derived from the wavelet coefficients.

Putting this all together we get:

$$Dist_0 = \sum_{Y,I,Q} W_c \left(\frac{W_a}{\sigma_a} |C_t - C_s| - \frac{1}{\sigma_w} MatchCount \right) \quad (1)$$

where: W_c are the relative weights (8, 3, and 1) of the color channels, W_a is the factor to adjust the relative importance of the overall color average as compared to the wavelet coefficients (this factor can also be modified per color channel), C_t and C_s are the average colors of the target and source, and $MatchCount$ is a count of the number of matches (in both index and sign) of the 30 largest wavelet coefficients. Each term is taken to have components in each of the YIQ channels.

Temporal Matching

The wavelet formulation lends itself naturally to also accounting for our third criterion, matching the temporal characteristics of the target and source. Rather than wavelet transforming each frame independently, we run a 3D wavelet transform on a small window in time on both the source and target videos. For efficiency, we chose to look at a 16-frame window. We also look only at the top 5 spatial levels of coefficients, thus in essence we are matching a 16×16 spatial resolution $\times 16$ frames in time block of video. As in Equation 1 above, the final distance measure is based on the average color and number of matches in the largest 30 coefficients in each of the YIQ color channels across this spatiotemporal block of video.

Maintaining Coherence

Finally, we address the fourth criterion, maintaining coherence from frame to frame of the result. This requires a different type of measure since, unlike the previous measures, it can not be formulated as a separate decision for each frame of the result.

To minimize a noisy solution, we reward the choice of a source frame if it is the frame that immediately follows the one chosen for the previous output frame. Or conversely, we penalize switching the choice of source video from frame to frame. Thus, there is one addition term for the final distance measure:

$$Dist = Dist_0 + SwitchCost(k, j) \quad (2)$$

where the $SwitchCost$ is 0 if frame k immediately follows frame j in the source. Otherwise, $SwitchCost$ is a user modifiable constant.

3.2 Finding the Best Matching Sequence

With the introduction of the switching cost, finding the best source video frame to tile a target frame changes from a local search (over the current frame) to a global one involving the previous frame, and by extension, the entire target video.

The global optimization problem can be stated as minimizing the sum of final distances over all tiles, over all target frames, and over all possible source frames for each tile. Since there is no inter-tile term in the distance measure we can address the optimization problem for a single tile and then apply this independently for each tile. More formally, for each tile, we seek some sequence of frames $U = \{U_0, U_1, \ldots\}$, where the length of U, $|U|$, equals $|T|$, the length of the target video, and each frame $U_i \in S$, the collection of all source frames. We seek the minimum

$$U^* = \min_U \sum_i Dist(U_i, T_i) \quad (3)$$

We solve the global optimization problem for each tile with a dynamic programming approach. The approach involves building a table (see Figure 4) that has as many rows as frames in T, $|T|$, and as many columns as the total number of frames in all the source videos, $|S|$. Each entry $Value(i, j)$ of the table will contain the minimum total cost of *all possible sequences* from U_0 to U_i that end with the j^{th} source frame. Thus:

$$Value(i, j) = Dist_0(T_i, S_j) + MinPath(i, j) \quad (4)$$

where

$$MinPath(i, j) = \min_{k=1\cdots|S|}(Value(k, j-1) + SwitchCost(k, j))$$

We also save the source index of the minimum value from the previous row, k_{min}. This reduces searching for a minimum path

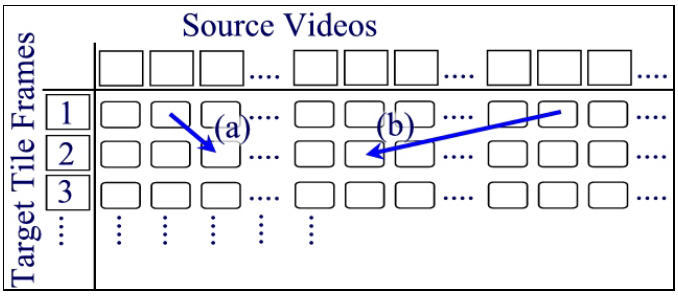

Figure 4: Dynamic programming, for each tile, to select best source frames for each target frame. Each cell of the table contains the value of minimum cost path to that cell, plus a pointer to the cell in the row above that generated the path. Path (a) does not incur the SwitchCost since the source frames are in sequence, unlike path (b).

over all sources frames (the term $\min_{k=1\cdots|S|}$ above) to a choice between two options. The minimum path must continue from either k_{min} or from $k-1$, the previous frame in the source video, in which case we pay no switching cost. The computational cost of filling in the table for each tile is thus $O(|T| \times |S|) \times O(Dist_0 calculation)$.

In addition to storing the $Value$ at each location in the table, we also store k, the source index of the previous frame that led to this minimum entry. When we have completed filling in the table, the optimal sequence, U^*, is found simply by choosing the minimum entry in the last row of the table and then walking back up through the table following the k index values.

3.3 Color Correction

Given U^* for each tile we can now display a video mosaic created by simply playing these sequences for each tile. Although there are no pixels derived directly from the target video, this new video will mimic the target to some extent. As a final step, we now perform a color correction that will change each pixel from U^* to more closely match the target while still maintaining the integrity of the source videos.

Our goal is to adjust the pixels in the source image so that their local average color matches that of the target. We use the method of [Finkelstein and Range 1998], a simple scheme motivated by digital half-toning. The objective is to match the color of the tiling frame to the color of the region in the target frame covered by the tile. If the target frame has a constant color x across this region, then the tiling frame should be adjusted so that its average color a is equal to x. If the brightness of the target frame ranges from dark on the left side to light on the right side, then the brightness of the tiling frame should be adjusted to match this gradient as well. In addition, features in the tiling frame should be preserved as much as possible.

The solution is a set of *correction rules* specified by [Finkelstein and Range 1998]. These rules map a color in the frame to a color in the final mosaic so that the region of the mosaic covered by the tiling frame will have the desired average color a. We perform a linear shift and scale of the histogram of colors in the source image so that the average color of this adjusted source image is the color of the target pixel. Specifically, if we can use only a shift without sending any of the colors in the tile out of range, then we apply this shift. Otherwise, we shift the color values as much as possible, then scale the resulting colors until the desired average is attained. We apply this operation *for every pixel* in every tile in the mosaic. For each of these pixels, the color of the corresponding pixel in the *original* target frame is supplied as the desired average color a. Note that this operation is not simply a blend of the source and target pixels, and in fact achieves a much better local match to the target color while maintaining the integrity of the source

Figure 5: A simple Video Mosaic in which the same video is used as both source and target.

tile. By having each pixel's value be shifted or scaled according to the value of the corresponding pixel in the original target frame, areas of complex color variation in the original frame are preserved. Similarly, areas of constant color are also preserved. For further details, see [Finkelstein and Range 1998].

4 Results

We present a number of results from the Video Mosaic system. All figures have an accompanying video which readers are encouraged to view.

Figure 5 shows a simple but appealing video mosaic - tiling a video with itself. Part of the appeal comes from the fact that the motion in the smaller videos mirrors that in the larger videos. Because this video requires no real computation to generate, it only takes a few minutes.

The remaining figures in the paper were generated with the method described in Section 3. All results were generated on a PentiumIII 933MHz machine with 512MB of RAM. We used three collections of source videos. The first, hereafter referred to as the "TV" footage, was 1 hour long. It was captured from MTV, VH1, and Comedy Central. The preprocess to extract and store the indices of the 30 largest wavelet coefficients took 33 minutes. The second corpus, hereafter referred to as the "documentary," consisted of 45 minutes of footage from a nature documentary about the Florida Everglades and took 23 minutes to preprocess. The final collection of source videos, which required 17 minutes for the wavelet coefficient preprocessing, was 30 minutes of video footage taken on and around the Princeton University campus. All video mosaics were generated using the TV footage as source videos

Video Mosaic	Length (secs)	Computation Time (h:min)	Switch Cost	W_a
Figure 1	10	3:47	$AvgDist\times$ 1.0	1.0
Figure 6	8	3:15	$AvgDist\times$ 1.0	1.0
Figure 9	10	3:47	$AvgDist\times$ 1.0	20.0
Figure 10	10	3:47	$AvgDist\times$ 1.0	1.0
Figure 11	10	2:58	$AvgDist\times$ 3.0	1.0
Figure 13	11	4:47	$AvgDist\times$ 2.0	1.0
Figure 14	10	5:00	$AvgDist\times$ 3.0	1.0

Table 1: Video mosaic generation statistics.

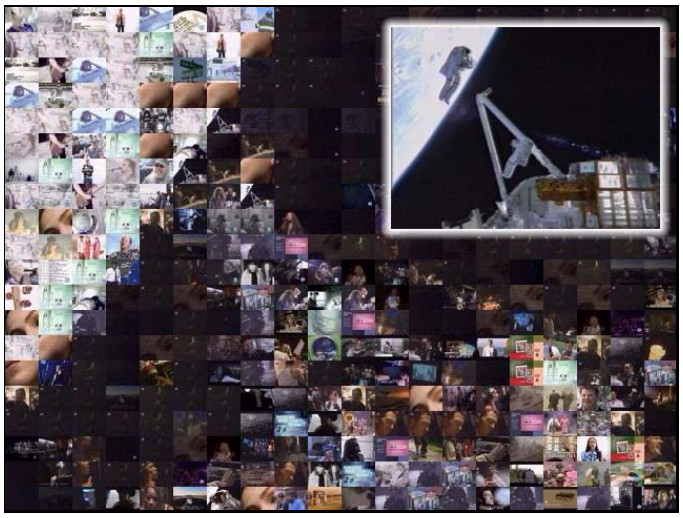

Figure 6: Video Mosaic after tile selection but before color correction. (Original frame is in upper right corner). Note how the automatically selected source frames help delineate shapes in the original.

Figure 7: Same frame as Figure 6 but after color correction.

except for Figures 11 and 12 which used the documentary footage.

Some timing statistics for the video mosaics, plus values for $SwitchCost$ and the weight given to average color matching versus edge matching (W_a from Section 3), can be found in Table 1. The $AvgDist$ referred to as part of the $SwitchCost$ is the average $Dist_0(T_i, S_j)$ value calculated across the dynamic programming table. For all of these video mosaics, the color correction took only an additional 2 minutes.

Figure 6 and Figure 7 show the sequence from the original video (the upper right corner of Figure 6), to a video mosaic created based on edge matching but without color correction, to the final, color-corrected video mosaic. The mosaic is 8 seconds long, contains 400 tiles in a 20 by 20 grid, and uses the TV footage as source. Figure 6 shows how our method automatically chooses source frames that delineate shapes in the target video frame.

Figures 8, 9, and 10 show the difference between weighting the distance metric towards color versus edge matching. Note how the mosaic created with an emphasis on color matching (the left-hand image in Figure 8) looks better before color-correction, but shows very noticeable grid artifacts in the color-corrected version, Figure 9. In contrast, the video mosaic created using

edge matching, the right-hand image in Figure 8, does not yield as pleasing results without color correction, but with color correction offers a smoother final image.

Figures 11 and 12 show additional results, both generated using the nature documentary as both source and target video. We particularly like the idea of using thematically related footage for both sources and targets. We take this idea a step further in Figure 12 by using a leaf-like pattern as the tiling pattern. Figures 13 through 16 show some additional results.

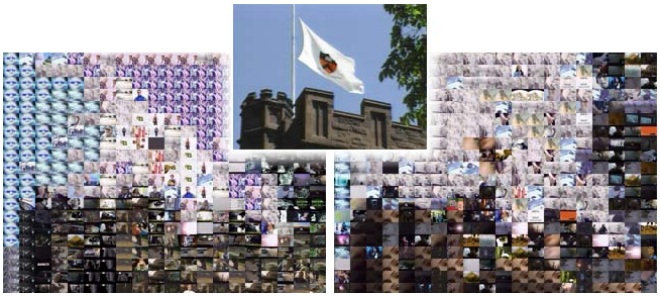

Figure 8: Two results before color correction. Users may choose to emphasize the average color match (left) or wavelet matches (right), yielding different results. Original is at top.

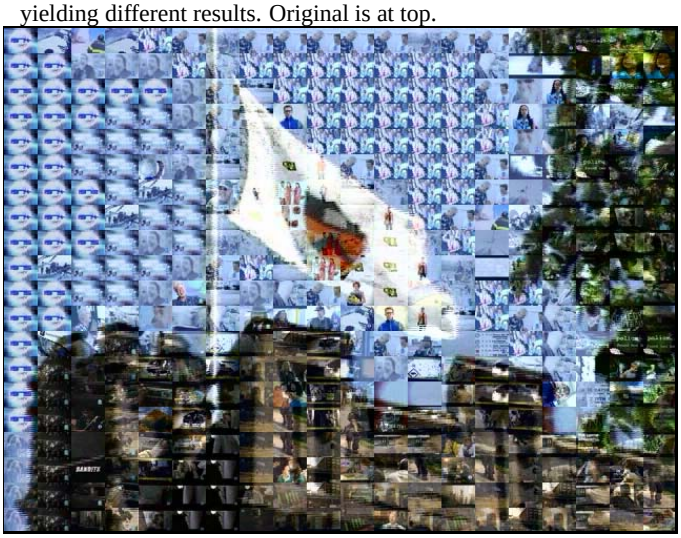

Figure 9: After color correcting the video mosaic emphasizing average color matches.

Figure 10: After color correcting the video mosaic emphasizing wavelet matches. Note that color correction adjusts colors while maintaining the edge/smoothness.

5 Conclusion

We have presented a system for creating video mosaics – an arrangement of small videos that suggests a larger, unified video sequence. We solve the main challenge of providing both visual matches and temporal stability in a computationally efficient way, while still being flexible enough to accommodate a variety of matching criteria.

There are many more possibilities for altering the final look of the video mosaic:

- **Different tile shapes and animated tiles:** There is no reason the tiles need to be simple rectangles. In fact, if one segments the target video into semantically meaningful regions, the tiles shapes could be modified to follow these region boundaries.

- **Allowing temporal distortions:** Tiling sequences to be temporally distorted (slowed down or sped up) to achieve better matches. Finding the best temporal distortions presents an interesting problem.

- **Different matching criteria:** Doubtless, there are other matching criteria worth investigating. Given a large enough corpus of source video, one might also score the match based on semantic information.

- **Automated video collection:** Not surprisingly, a large, diverse collection of source video footage produces better results since there are more possibilities for finding a good match. However, we found getting good input videos to be difficult. For example, collecting footage from television is often unsatisfying because cuts often happen too frequently (for example, every 2 or 3 seconds on a station such as MTV) or the subject matter itself does not change frequently enough (you get an hours worth of elephants on a PBS nature special). An automated process might be able to look through large volumes of video footage and segment out contiguous shots of desired length.

- **Incorporating video textures:** Another way of collecting good tiling footage may be to synthesize longer sequences from shorter ones using the video textures framework [Schödl et al. 2000].

- **Infinite mosaics:** Designing infinite, circular video mosaics would provide fun results. The end goal would be a video mosaic that allows you to zoom into one of the tile images to get another video mosaic and then repeat this process until you are finally zooming back into the original mosaic.

- **Parallelism:** Our approach should be easily parallelized, since each tile is independent of the others in both matching and color correction.

Acknowledgements

This work was supported in part by a Microsoft Research Fellowship.

Figure 11: A video mosaic of some young girls walking. Note how a fire sequence from the nature documentary has been used to tile the sandy path.

Figure 12: Video mosaic using non-rectangular tiles. Tile mask in lower left.

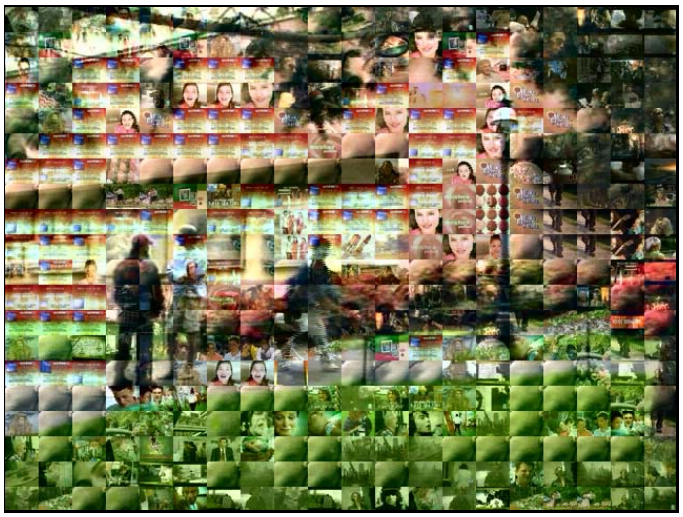

Figure 13: A video mosaic of students on a college campus.

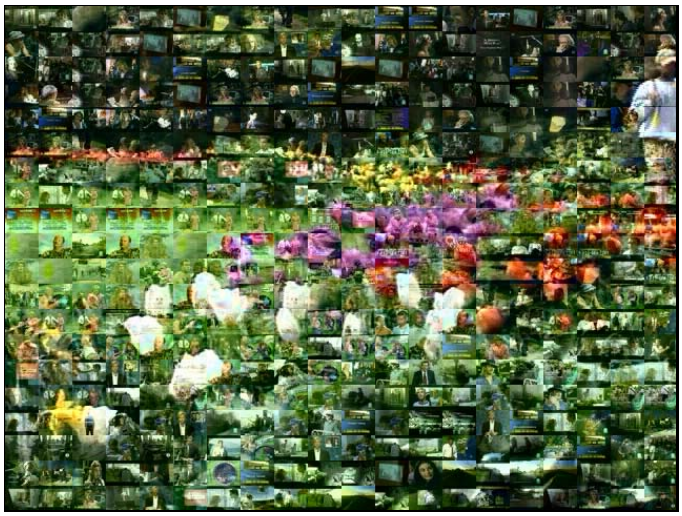

Figure 14: Video mosaic of a student walking through a garden.

References

FINKELSTEIN, A., AND RANGE, M. 1998. Image mosaics. *Proceedings of RIDT 1998*, 11–22.

HERTZMANN, A., AND PERLIN, K. 2000. Painterly rendering for video and interaction. *Computer Graphics (Proceedings of NPAR 2000)*, 7–12.

JACOBS, C. E., FINKELSTEIN, A., AND SALESIN, D. H. 1995. Fast multiresolution image querying. *Computer Graphics 29*, 277–286.

LITWINOWICZ, P. 1997. Processing images and vdeo for an impressionist effect. *Computer Graphics (Proceedings of SIGGRAPH 97) 31*, 407–414.

MATAEV, O., AND MATAEV, H., 2001. Olga's gallery. `http://www.abcgallery.com/A/arcimboldo/arcimboldo.html`.

NEEDLEMAN, S. B., AND WUNSCH, C. D. 1970. A general method applicable to the search of similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology 48*, 443–453.

OSTROMOUKHOV, V., AND HERSCH, R. D. 1995. Artistic screening. *Computer Graphics (Proceedings of SIGGRAPH 95) 29*, 219–228.

2001. Paracel algorithm primer: similarity searching algorithms. `http://www.paracel.com/faq/faq_algorithm_primer.html`.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. *Computer Graphics (Proceedings of SIGGRAPH 00) 34*, 489–498.

SILVERS, R., AND HAWLEY, M. 1997. *Photomosaics*. Henry Holt & Company, Inc, New York, New York.

STOLLNITZ, E., DEROSE, T., AND SALESIN, D. H. 1996. *Wavelets for Computer Graphics*. Morgan Kaufmann, San Francisco, California.

SZELISKI, R. 1996. Video mosaics for virtual environments. *IEEE CG&A* (March), 22–30.

WATERMAN, M. S., AND SMITH, T. F. 1981. Identification of common molecular subsequences. *J. Mol. Biol. 147*, 195–197.

WYE, D., 1998. Museum of modern art website. `http://www.moma.org/exhibitions/close/`.

ZUCMAN, G., 2001. Artboy. `http://www.ioc.net/~artboy/P/mole.html`.

Figure 15: A sequence of three images taken from the walking girls mosaic.

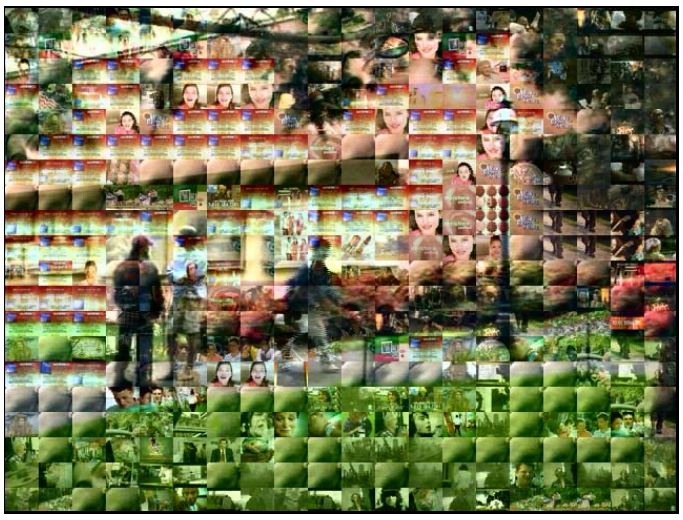

Figure 16: A sequence of three images taken from the college campus mosaic.