Image Mosaicing with Motion Segmentation from Video

Augusto Román and Taly Gilat
EE392J – Digital Video Processing
Winter 2002

Introduction:

Many digital cameras these days include the capability to record video sequences. A video taken standing in one place and looking around could be used to create a single panoramic image. Using motion estimation to align each frame, the overall image can be accumulated across the range of view covered by the video. Assuming a reasonable frame rate and a reasonable angular speed (little or no blurring), the motion from one frame to the next should be predictably small.

Some current methods of taking panoramic pictures are:
(1) Using a very-wide angle lens (up to 360 degrees!)
(2) Taking a number of pictures and manually aligning & stitching them together

Method (1) can be rather expensive and not accessible to the average consumer. Method (2) can be difficult, time- consuming, and may have problems with seams between images due to lighting, angle, and motion.

Our method should allow a number of potential advantages:

- Cheap - Any device capable of generating a movie (DV camcorders and many digital cameras) could be used.
- Seamless - Combining a number of frames for each output pixel should reduce seam artifacts
- Infinite FOV - Can also be used simply to capture large images but scanning across a desired scene.

Depending on time constraints, the following possible extensions offer further advantages:

- Occlusion - Using motion segmentation, moving objects can be eliminated.
- Motion playback - Using motion segmentation, the resulting panorama could include moving objects.
- Superresolution - Multiple overlapping images from a low-resolution video could be combined to create a higher resolution panorama.
- Substituting Layers - By identifying similar layers across frames, layers can be extracted and replaced in a video, allowing, for example, the substitution of the background.

The approach we are implementing for transforming the video sequence into a panoramic image is based on the work of Wang and Adelson. The algorithm proposed by Wang and Adelson (referred to as the WA algorithm) segments objects in a video sequence based on motion, so that overlapping layers represent the video. Each layer contains objects undergoing similar motion and a motion model for describing the objects through the

video sequence. The algorithm accumulates the layers by combining the contributions from all video frames. These accumulated objects are exactly what is needed for panoramic image mosaicing. The algorithm also presents a method for determining the depth relationships of the segmented objects, so that occlusions can be dealt with correctly.

Although Wang and Adelson present several uses for the algorithm, their primary motivation for the layered representation is improved video coding. In this report we will focus on the application of the WA algorithm to image mosaicing. Since [1] and [2] provide a thorough description of the algorithm, we will give a brief overview and elaborate on our proposed enhancements, modifications, and results.

The WA Algorithm

The goal of the algorithm is to segment the images into regions with similar motion, and then to describe this motion. The motion models are necessary for accurate segmentation, while the segmented regions are required for determining the motion model. The algorithm uses an iterative approach to solve this chicken and egg problem.

A basic flow chart of the implementation is shown in Figure. 1.

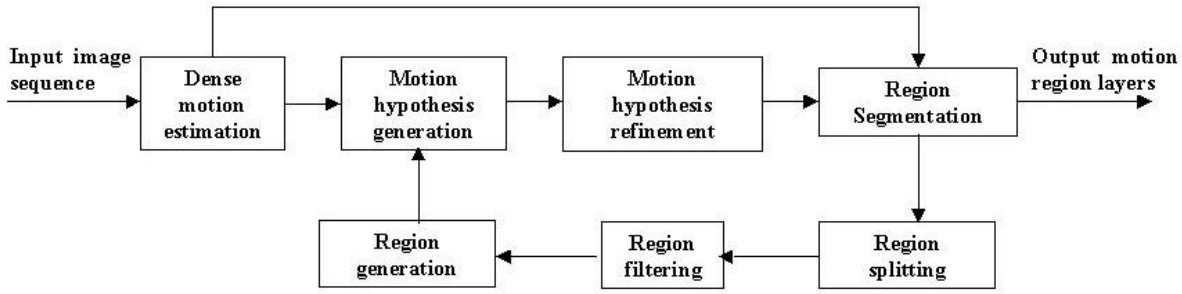

Figure 1. Flow chart of algorithm implementation. Adapted from [1].

Dense Motion Estimation

Because all the following layers depend on and reference the original dense motion field, it is essentially the "gold standard" and as such must be as accurate as possible for the rest of the algorithm to work. The dense motion estimation was initially implemented using a full-search block-based routine. Estimating the motion on a per-pixel level causes this method to be unacceptably slow – on the order of two hours per image pair. In addition, while this approach minimizes the MSE difference between the two frames, it does not necessarily correspond well to the true motion between the frames.

Instead, we used phase correlation to determine the motion between pairs of frames. This method works by taking a relatively large block of the image and computing the cross-correlation using the Fourier transform. This identifies several possible motions contained within this block. Each pixel in the center of the block is then tested with each of those possible motions and the motion with the minimum MSE is selected for that

pixel. Because the number of possible motions is small, generally not more than 8 or so, this method offers an enormous efficiency boost over the full-search approach.

Only the pixels in a central region of the block are assigned motion vectors to avoid noise due to aliasing. Thus, subsequent blocks must overlap the previous block such that the central regions fill the image. That is, if a 16x16 region in the center of a 128x128 block is assigned motion vectors from the larger blocks' phase correlation, then the next image block computed would have to be shift over by only 16 pixels so that the center regions line up.

Pseudocode for the algorithm is as follows:

`f1 = normalize( frame(i) )`
`f2 = normalize( frame(i+1) )`
`for each block B`
	`Select input frames`

`b1 = f1(B) * window(B)` `b2 = f2(B) * window(B)`	`Iterate over each block choosing corresponding blocks from each image`
`FB1 = fft2( b1 )` `FB2 = fft2( b2 )`	`Compute spectrum of each block`
`CPS = ( FB1 * conj(FB2) ) /` `abs(FB1*conj(FB2))`	`Compute cross-power spectrum`
`PCF = ifft2( CPS )`	`Compute phase correlation function`
`possible_motions = peaks( PCF )`	`Select possible motions`

`for each pixel in f2`
`for each possible_motion p`
`compute block match err (f1+p,f2)`
`select min err(p)`
	`Test each pixel against possible motions`

For a more detailed description of the theory behind the basic phase correlation algorithm, see section 6.4.5 (pp 162 – 163) of [3]. For our algorithm, we used a block size of 64 and a block shift of 16. Using the phase correlation algorithm, the processing time per frame was reduced from over two hours per frame pair (with the full search) to under 30 seconds.

In attempting to improve the accuracy of the resulting motion fields, we tried implementing a number of modifications to the basic algorithm. First, we normalized the input images to a specific mean and standard deviation to compensate for brightness variation across the video. Also, we wanted the algorithm to increase the validity of the estimated motion in highly textured areas of the image. In order to accomplish this, we derived an "amount of texture" image to each corresponding input frame by applying a high-pass filter to each input frame. This image was used to weight the pixels of input frame and thus weight their influence on the correlation. Eventually, however, the added complexity did not produce a substantially improved estimation and this modification was dropped.

Motion Hypothesis Generation

At this stage of the algorithm we have estimated segmented regions. For the first iteration of the first frame, the image is initially segmented into non-overlapping blocks. For subsequent frames, the algorithm is initialized with the regions from the previous frame.

For each region, the six affine motion parameters are estimated from the dense motion vectors using a least-squares approximation. These parameters are the average motion hypotheses for the region layers.

Motion Hypothesis Refinement

Each motion hypothesis describes a possible motion in the image and can be thought of as a point in a six dimensional space. K-means clustering is used to refine initial hypotheses into K representative motion hypotheses. These hypotheses will be used in the next step of the WA algorithm to segment the image.

The k-means clustering algorithm begins with K initial cluster centers that represent the mean motion models. The cluster membership of each hypothesis is determined by finding the nearest cluster center. Once the membership has been determined, new cluster means are determined and the algorithm is repeated until convergence.

We implemented several modifications to the k-means algorithm to improve the clustering efficiency and accuracy.

One problem is that the value of K is not known explicitly. Another problem is that both empty clusters and largely varying clusters do not accurately represent motion in the image. Therefore the algorithm must adapt to remove empty clusters and divide clusters that are too large. To solve these problems empty clusters are removed during each k-means iteration. To keep the number of means from continually getting smaller, a minimum K value is input to the clustering algorithm. If the number of clusters is less than this minimum, the cluster with the largest variance is split into two. This cluster splitting is repeated until the number of means reaches the minimum.

The cluster means from the previous k-means iteration are used to initialize the current k-means algorithm. For the first iteration, the six dimensional space covered by the initial motion hypotheses is divided evenly in each direction to generate initial mean guesses.

Region Segmentation

The output of the k-means clustering is the list of possible motion models. The projected coordinates of each pixel using the dense motion vectors is compared to the projected coordinates obtained with the possible affine parameter hypotheses. The hypothesis with

the least error is chosen as the representative model. A layer map that describes the motion model membership for each pixel is then constructed.

An error threshold has been added to this step so that pixels with large motion error are left unclassified. These keeps outlier pixels from affecting the average motion parameter calculation. This threshold is determined by calculating the variance across the motion estimates for each layer. Pixels whose estimates are more than N standard deviations away from the mean are removed. This threshold, N, can be changed for each WA iteration so that the allowed error is reduced as the motion hypotheses improve. Currently we are using N=1 for all iterations.

Region Splitting

In this step disconnected regions within each layer are separated, because generally objects that undergo the same motion are connected. This allows for a separate motion hypothesis for each unconnected region. If the regions still fall in the same layer, they will be merged in the next region segmentation step, thus restoring the cohesion between distinct areas with similar motions. The region splitting also produces more motion hypotheses for the k-means, which improves the clustering results.

Region Filtering

Small regions do not provide good least squares motion estimates. Therefore, layers with small regions are removed. The minimum region size is an algorithm parameter currently set at 256 pixels.

Region Generation

A layer map now describes the motion regions. This map is inputted to the motion hypothesis generator to start the next WA iteration.

Figure 2 shows two original frames, and Figure 3 displays the region maps generated by the algorithm at several iteration steps.

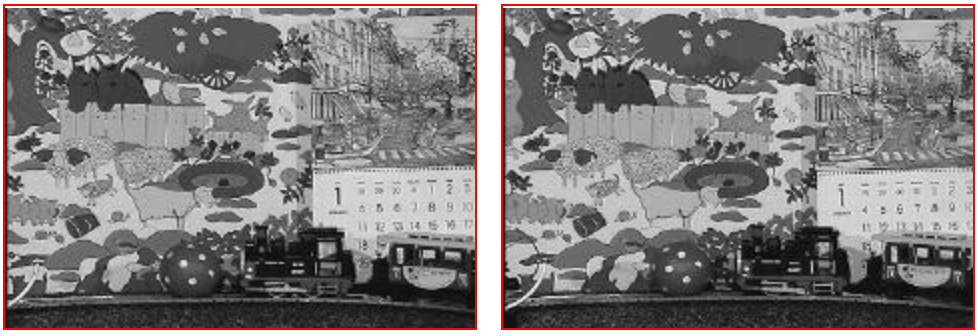

Figure 2. Two frames from the original sequence

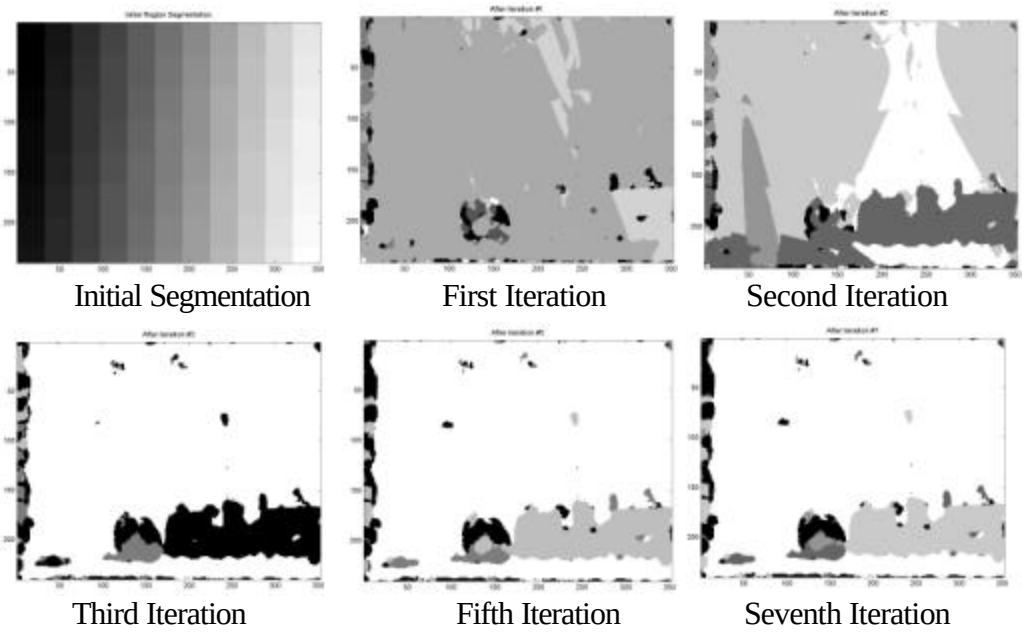

Figure 3. Generated region maps for several iterations

The Whole Enchilada

The previous sections describe the building blocks of the WA algorithm. Some clarification is needed on how we combine these building blocks to segment video data.

For each pair of frames, the dense motion vectors are computed, the segmentations calculated, and the affine parameters estimated for each segment. The dense motion vectors are calculated once using the phase correlation method. Determining the region segmentation and affine parameters is an iterative procedure. In our implementation, five iterations appeared to be sufficient for reasonable convergence. The number of iterations could be set adaptively by monitoring the convergence of either the region map or the MSE of the motion estimates.

For the first pair of frames, the region map is initialized to blocks and the k means initialized with a lattice. For all subsequent frame pairs, the previous region map and motion estimates are used for initialization.

Layer Synthesis

Once all the motion data for all frames is computed, they are merged together in the synthesis step. Because the relationship between the layer numbers is not consistent between frames, we chose to merge the largest layer from each frame assuming that that will represent the background layer. This turned out to be an appropriate assumption for

our input videos. We unfortunately did not have time to implement a more elegant method of associating layers across frames.

Back-projecting each region from the second frame onto the first frame performs the layer accumulation. This establishes a common image frame for both sets of data. Because the difference between frames is small, data from numerous frames will project to the same pixel location in the common image frame. Starting with the first pair of frames, we use the computed affine motion parameters to project the second frame onto the first. For the next pair of frames, we need to compute the total projection from the third frame to the first frame. The accumulated projection is stored in a single affine transformation matrix and updated simply by right-multiplying it by the newest projection. The downside of this method is that small frame-to-frame motion estimation errors can be accumulated over a long period of time and produce noticeable alignment artifacts in the final image. In order to reduce noise, preserve edges, and help remove motion artifacts, a temporal median filter is applied to the final accumulated image. That is, each pixel in the final image is the median of all the values covering that pixel from each frame.

Results

The basic panoramic functionality of our implementation is shown in Figures 4 and 5. A vertical panorama is also shown in Figure 6.

Figure 4. Four example frames from a video sequence panning from left to right.

Figure 5. The synthesized background image from the panning video.

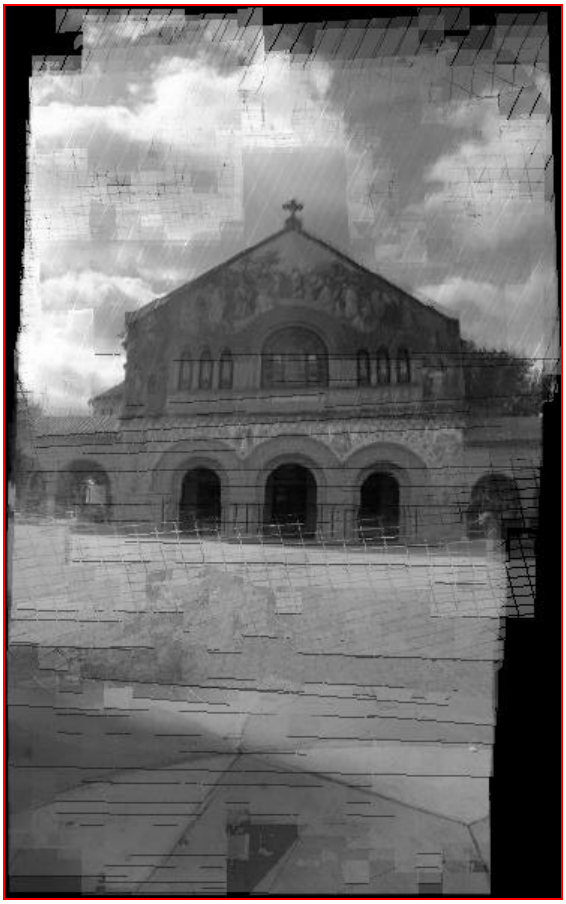

Figure 6. A panorama created from a vertical video pan.

Our implementation also removes moving objects from a given scene. This is first demonstrated by a video sequence that remains still but contains a couple walking from left to right across the scene. Figure 7(a) shows an original frame from the video sequence and 7(b) shows the segmented background region for that frame. Figure 8 shows the resulting background panorama.

Figure 7(a) on the left shows the original video frame and
7(b) on the right shows the segmented background.

Figure 8. The resulting synthesized frame without foreground objects.

Figures 9 and 10 show another example of removing a moving object during a camera pan. Figures 9 (a) and (b) are two frames from the original video sequence with a cyclist. Notice how he is removed in the resulting panorama.

Figure 9(a) and (b). Two original frames with cyclist.

Figure 10. The resulting panorama without cyclist.

Conclusions

Overall, the algorithm accumulates the background well, forming nice panoramic images. Moving objects in the video sequence are neatly removed from the panoramic image. While the algorithm performs well, the numerous algorithm parameters could be refined further for future improvements.

One possible improvement is to include the color information in the dense motion estimation. While this would triple the computational cost, the importance of this stage may justify the increased complexity if the dense motion estimation is significantly improved.

Improved methods for finding the corresponding layers across frames, would allow more flexibility in processing the results of our implementation. For example, moving objects could be extracted separately or replaced. The registration of layers between two frames could be performed manually, or by finding the maximum cross correlation between the layer combinations.

References

[1] J.Y.A. Wang, E.H. Adelson. Representing moving images with layers. In *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, volume 3, pages 625-638, September 1994.

[2] J.Y.A. Wang, E.H. Adelson. Spatio-temporal segmentation of video data. In *Proceedings of the SPIE: Image and Video Processing II*, volume 2182, San Jose, February 1994.

[3] Wang, Ostermann, Zhang. *Video Processing and Communications*. Prentice Hall, 2002.

Work Division

While the overwhelming majority of the work on this project was performed working together, each of us was responsible for primarily for some parts of the code. The phase correlation and synthesis code was Augusto's while the region segmentation and motion parameter estimation code was Taly's.