Building mosaics from video using MPEG motion vectors

Ryan C. Jones, Daniel DeMenthon, David S. Doermann
Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
rjones,daniel,doermann@cfar.umd.edu

Abstract

In this paper we present a novel way of creating mosaics from an MPEG video sequence. Two original aspects of our work are that (1) we explicitly compute camera motion between frames and (2) we deduce the camera motion directly from the motion vectors encoded in the MPEG video stream. This enables us to create mosaics more simply and quickly than with other methods.

1 Introduction

The mass digitization of video has elevated automated storage and retrieval to a grand challenge. Video sequences can store a vast amount of useful information, but redundancy between individual frames is a problem when analyzing, browsing, or searching video. Presenting the video sequence in a compact manner is a difficult challenge because eliminating redundancy could also eliminate content. The selection of static keyframes to represent a shot sequence is commonly used for indexing as well as for presentation of retrieval results. This technique is insufficient for revealing much of the content in a video sequence. In some domains it may be more appropriate to collect all the frames of a shot into one image, called a mosaic, and present the static mosaics to the user. This method overcomes the redundancy problem while revealing camera motion content more readily than other techniques. The use of mosaics for video browsing has been investigated in [8], [9], and [10]. By constructing mosaics, more efficient representations can be used in various other tasks and applications, including video analysis, editing and manipulation, surveillance, digital libraries, interactive low-bit rate video transmission, video conferencing, and high-resolution still images.

One novel feature of our technique is that rather than using computation intensive image processing techniques to align frames of a video sequence, we estimate the camera motion by averaging motion vectors encoded in the MPEG compression scheme. The method is therefore faster than methods that rely on image processing.

2 Previous Work

To build a mosaic one must be able to align frames from a sequence and then integrate them into one image. Typically, researchers accomplish frame alignment as follows: A model for frame to frame transformation is assumed; they then solve for unknowns in the model by matching points between the frames. By contrast, we compute camera motion from MPEG motion vectors and invoke the geometry of perspective projection to generate a pixel mapping that will align frames with a reference frame, usually the first frame in a sequence. Morimoto and Chellappa [7] also construct mosaics using camera motion computation, but find the camera motion by tracking feature points.

Most techniques use 2D transformations (affine, projective, or quadratic) to align frames. 2D transformations are effective on static scenes such as a view of a city skyline. One widely used technique is hierarchical direct registration [1, 3, 8]. This method compares the image intensities of successive frames at different scales to find frame to frame transformations. Optical flow has also been used as a basis to compute frame transformation parameters [6, 9]. The optical flow is matched with a frame transformation model to calculate displacements between frames in a video sequence.

After the motion parameters are estimated, the frames are aligned and integrated into one image. Overlapping pixels are either averaged or dropped entirely. The averaging technique includes (1) averaging all the intensity values (but this usually creates ghost images of objects not correctly aligned), (2) applying a temporal filter based on the temporal domain of the images or a predefined weighted temporal median or average, which can show the progression of moving objects [3, 9]. Methods to select one pixel of the overlapping images include (1) selecting the pixel with the most recent information [2, 3], (2) selecting the pixel that has best resolution and quality [2, 3].

3 System Overview

A video file typically contains several shots. Our system builds a separate mosaic for each shot so that the user can be presented with a sequences of mosaicked images as a summary. To do this, our system requires modules for segmentation into shots, camera motion estimation, frame alignment, frame integration, and a goodness heuristic to judge the quality of the final mosaic. Shot change detection provides a file containing the indexes of the frames that begin a new shot, and is obtained by the detection techniques described in [5]. The camera motion parameters are com-

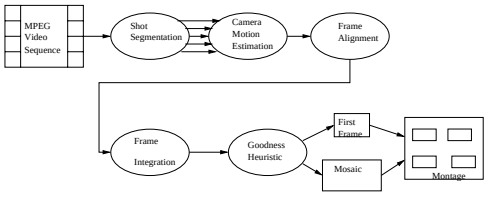

Figure 1: System diagram.

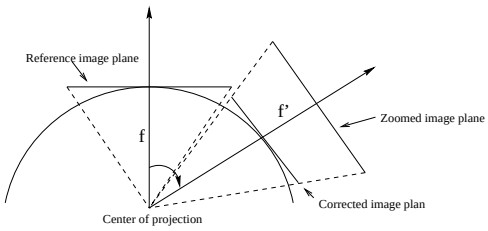

Figure 2: Cylinder diagram.

puted for each shot using the MPEG motion vectors (see Appendix), frames from a shot are aligned (Section 3.1), integrated into a static mosaic (Section 3.2), and a montage of all the mosaics is assembled into a single image (Section 4). This representation gives a preview of the video that can be browsed quickly. The components of the system are shown in Figure 1.

3.1 Frame Alignment

Our system aligns frames directly from camera motion estimates, bypassing any computationally intensive registration of frames. To determine the parameters needed to align different frames from a video sequence, the camera motion is computed by using the technique described in the Appendix. The technique is based on averaging the motion vectors from the macro blocks within each frame of an MPEG sequence. From the camera motion parameters the displacement between two consecutive frames in a sequence can be computed. For each successive pair of frames the pan, tilt, and zoom differences are found. To find the displacement between an initial reference frame and any other frame, the angular displacements (pan, tilt) between consecutive frames are summed up, while scale changes are performed to correct for zoom changes.

In our work the first frame of a sequence is used as the reference frame. This is a good selection if there are clean breaks between shots; otherwise, if fades are detected for shot transitions, a later frame should be used because the fade may produce a false detection of camera motion which could misalign subsequent frames in the sequence. To simplify the alignment, the focal length of the reference frame is used for successive frames. Thus individual corrected frames are projected on a common cylindrical surface (Figure 2). The details of this computation are provided in the Appendix. The mosaic of a sequence is the visual representation of the unfolding of the frames of the sequence onto a flat plane.

3.2 Frame Integration

Once the frames have been aligned on the cylinder (Figure 2), the next step is the selection of pixels to be put into the resulting mosaic. To integrate multiple frames into a single

Figure 3: Panoramic view from a control tower, constructed from a 4-second clip.

Figure 4: Panoramic view of a conference room sequence of 600 frames. Black lines illustrate the boundaries between areas of selected frames. In this mosaic every tenth frame was considered for mosaicking.

mosaic image, the first step is to find the size of the mosaic. The minimal bounding box is found around the upper right hand point and lower left hand point of each the frame in the sequence. Overlapping pixels can be filtered in two ways. The overlapping pixel that is closest to its frame center can be used in the mosaic, on the basis that pixels closer to frame centers tend to be less distorted than those toward the edges. For each new frame, frame pixels that belong to the area of overlap with the mosaic and are close to the frame center are used to replace previous mosaic pixels. This is good for static scenes where the camera moves over scenery, but moving objects may be removed. To show the progression of moving objects in a shot, a second option is to average all the overlapping pixels together. The appearance that is produced is that of the object fading into the background; its position at the begining of a clip is nearly transparent and its position at the end of a clip is nearly opaque (Figure 5).

3.3 Goodness Heuristic

Not every video shot produces a useful mosaic. The motion parameters are prone to error; moving objects sometimes confuse the camera motion computations (see Appendix). A heuristic is used to determine whether it is better to present the mosaic of a sequence or an individual keyframe. The heuristic value is determined by the square root of the sum of squared differences of luminance intensity values at the adjoining pixels along the boundaries between frames (Figure 4). If areas of the mosaic corresponding to different frames don't match well, there will be large discontinuities at the boundaries between these areas. Above a certain threshold of discontinuity, the mosaic becomes confusing and is less useful than individual keyframes.

4 Results

To evaluate the performance of this mosaicking technique we ran it on several video clips from our in-house database

Figure 5: The left mosaic was created with pixel dropping filtering applied on a 80-frame sequence, while the right mosaic created by the temporal averaging technique.

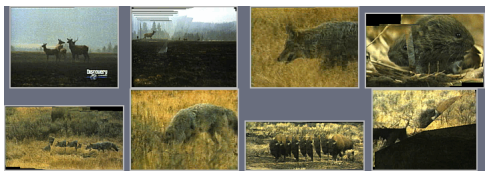

Figure 6: Montage of a 60-second video clip.

that demonstrate the advantages and disadvantages of our system. The length of the video sequences ranged from 3 seconds to 2 minutes. The example in Figure 3 shows how good the results can be when there is little or no movement of objects in the video sequence. In the video sequence that produced this result there were 117 frames, all of which were considered when creating the mosaic. The mosaic created in Figure 4 came from a video sequence in which the camera moved left to right followed by a zoom-in. One can see the zoom by the bounding boxes on the left side. For this video sequence the computation of the zoom parameter from the MPEG motion vectors was accurate enough to produce a nearly seamless integration of video frames. The correlations of the MPEG encodings used to produce the motion vectors gave better results in some parts of the sequence than others.

While camera motion is revealed in a mosaic, whether objects moving in the shot are present may not be revealed. By using an averaging technique for image integration, the progression of an object can be detected. In Figure 5, the mosaics are constructed from a sequence in which the camera pans left to right while tracking a player backing up from the ball. While this action is evident with a pixel averaging technique, it would not be noticed if the frames were integrated by selecting one of the overlapping pixels.

One way of browsing a long video clip is to examine representative frames throughout a sequence. While content is revealed, camera motion is not. Our system can produce a montage of all the mosaics that reveals both content and camera motion (Figure 6). This allows users to browse the video sequence quickly. This type of representation allows users to get a better sense of the action that is present in a video sequence than by looking at representative frames. The montage clearly reveals the presence of any panning and tilting present in the video sequence. Zooming is harder to detect because all zoomed frames are scaled to be neatly juxtapose with a representative frames. If the field of view does not change from the representative frames the zoom will only enhance what is present in the first frame. After careful examination a zoom sequence can be detected in the second mosaic on the first row. The deers on the right are sharper than those in the center. Video shots that did not produce good alignments were represented by the keyframes. For example in Figure 6 the third image of the first row in the montage is a keyframe. The mosaic was discarded in favor of the first frame of the sequence.

In Figure 7, an observer can detect that most of the motion is present in the first row. In this row the second mosaic contains black space. Black space will be present in a mosaic when the camera motion combines panning and tilting. The third (from left) mosaic is a pan. The height was shortened to keep the width the same as that of the keyframe. Mosaicking has problems with handling fades between shots. Without a clean shot break, the faded in/out frames will blur the final mosaic if pixels from these frames are selected during integration. The second mosaic of Figure 7 handles a case that other mosaic techniques may have problems with, a forward moving camera. In this scene a camera flies over a canyon. The later frames are reduced and projected to the same image plane as the first (Figure 8).

Figure 7: Montage of a 2-minute clip.

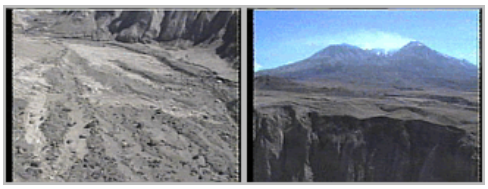

Figure 8: 10th & 275th frames of the 3rd shot in Figure 7.

5 Conclusion

By summarizing sequences of video frames into one mosaic image, more information can be viewed at once than by browsing keyframes. Long panning and tilting sequences become easier and faster to understand when viewing mosaics than when viewing the video sequentially or with keyframes. Our technique for constructing mosaics is novel in two respects. First, we explicitly compute the camera motion between frames and are able to construct a mosaic image because we know where the image planes of the camera are in space at the times when the frames are taken. The mosaic construction consists of projecting the pixels from the individual image planes onto a common cylindrical surface which is flattened to display the mosaic (Figure 2). Second, we use the motion vector information contained in the MPEG encoding of the video to compute the camera motion between frames, including the zoom factor. By using the MPEG vectors to determine pan, tilt, and zoom, no other image processing is needed to align the frames. Therefore this method is faster at creating mosaics than methods that rely on image processing techniques. Ultimately, this method will be useful for browsing and indexing large video databases.

6 Appendix: Camera motion from MPEG motion vectors

We define a vector $\mathbf{mm'}$ as the image motion vector corresponding to the motion of the world point from M to M'.

Its coordinates are $dx = x' - x$, $dy = y' - y$. It can be shown [4] that these quantities are approximately related to the pan (p), tilt (t) and zoom factor $\zeta = f'/f$ of the camera by

$$dx = (\zeta - 1)x - \zeta f p, \quad dy = (\zeta - 1)y + \zeta f t \tag{1}$$

Generally the focal length f for the first image is unknown. We are able to retrieve the products fp and ft rather than p and f. These are the arcs traced by an arm of length f rotating by the pan angle p and the tilt angle t. For cylindrical mosaicking, these arcs are precisely the quantities that are needed, rather than the corresponding angles.

In an MPEG clip the motion vectors $\mathbf{m}_i \mathbf{m}'_i$ are known, as well as their positions m_i at the centers of macroblocks. Therefore we can write equations such as Eqs. 1 *for each macroblock for which a motion vector is defined.*

If we sum Eqs. 1 over the N points of a symmetric grid of macroblocks, the positive and negative terms cancel out in the sum $\sum_N x_i$ and we obtain

$$fp = -\frac{\sum_N dx_i}{N\zeta} = -\frac{\overline{dx}}{\zeta}, \quad ft = \frac{\sum_N dy_i}{N\zeta} = \frac{\overline{dy}}{\zeta} \tag{2}$$

Above we used the fact that $\frac{1}{N}\sum_N dx_i$ is the mean of the motion vector component dx over the grid of macroblocks, denoted by $\overline{dx}$. We use this notation for all means.

To eliminate the pan and tilt terms and preserve the terms containing ζ, we multiply the expression of dx in Eqs. 1 by and odd function $f(x)$, and the expression of dy by an odd function $f(y)$, then we sample and sum the results over the grid of N macroblocks. We obtain

$$\zeta = 1 + 0.5\left(\frac{\overline{f(x)dx}}{\overline{xf(x)}} + \frac{\overline{f(y)dy}}{\overline{yf(y)}}\right) \tag{3}$$

In practice, we use $f(x) = \sin(kx)$. The factor k is selected to give little or no weight to the motion vectors at the edges of the image, which tend to be noisy.

The above derivations are correct only if the summation is performed over a symmetric grid. In practice there are grid points where the motion vector coordinates dx_i and dy_i are not available: they may be outliers that belong to a moving object (see below), or to an unreliable region whose DCT coefficients show that there is little texture. More often, motion vectors may not be known because the corresponding macroblocks are *intracoded*: the MPEG encoding process does not define a motion vector for them. If we only ignored these macroblocks when we computed sums, we would no longer be summing over a symmetric grid. To maintain summation over a symmetric grid, we must also ignore the macroblocks that are symmetric to these discarded macroblocks in the image.

The zoom factor ζ is the ratio between the focal length f' of the present frame and the focal length f of the previous frame. Therefore, if we want to find the change of zoom over several frames, we need to compose zoom factors between pairs of frames by multiplication. For example, the zoom factor between frame 1 and frame 3 is $\zeta_{1,3} = \zeta_{1,2}\zeta_{2,3}$.

If we find a pan arc ($f_1 p_{1,2}$) between frame 1 and frame 2, and an arc ($f_2 p_{2,3}$) between frame 2 and frame 3, we cannot just add these two arcs, because they correspond to two different radii f_1 and f_2. Instead, we have to normalize the arcs to the same focal length, for example the first focal length f_1. We can use the zoom ratios to perform this normalization. The normalized pan and tilt arcs between frame 1 and frame i can be expressed as

$$f_1 p_{1,i} = \sum_i \frac{(fp)_i}{\zeta_{1,i}}, \quad f_1 t_{1,i} = \sum_i \frac{(ft)_i}{\zeta_{1,i}} \tag{4}$$

with $\zeta_{1,i} = \prod_i \zeta_i$.

Once an initial camera motion estimate has been obtained, we compute the field of motion vectors that corresponds to this camera motion using Eqs. 1, in order to detect the outlier motion vectors that are not well explained by this camera motion. These outliers correspond to regions that do not move consistently with the majority of the motion vectors. We then repeat the camera motion calculation without using the outliers. We iterate over these steps until only consistent motion vectors contribute to the calculation, which generally occurs in three or four iterations.

References

[1] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierachical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pages 237–252, 1992.

[2] P.J. Burt, M. Hansen, and P. Anandan. Video mosaic displays. In *Proceedings of SPIE,* Volume 2736, 1996.

[3] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *Proceedings of the International Conference on Computer Vision*, pages 22–30, 1995.

[4] R.C. Jones, D. DeMenthon, and D.S. Doermann. Building mosaics from video using mpeg motion vectors. University of Maryland, College Park, Language and Media Processing Laboratory No. 035, 1999.

[5] V. Kobla, D.S. Doermann, K-I. Lin, and C. Faloutsos. Compressed domain video indexing techniques using DCT and motion vector information in MPEG video. In *Proceedings of the SPIE conference on Storage and Retrieval for Image and Video Databases V,* Volume 3022, pages 200–211, 1996.

[6] S. Mann and R.W. Picard. Video orbits of the projective group; a simple approach to featureless estimation of parameters. MIT Media Laboratory Perceptual Computing Section No. 338, 1995.

[7] C. Morimoto and R. Chellappa. Fast 3D stabilization and mosaic construction. In *Proceedings of the IEEE Conference on on Computer Vision and Pattern Recognition*, pages 660–665, 1997.

[8] Y. Taniguchi, A. Akutsu, and Y. Tonomura. PanoramaExcerpts: Extracting and packing panoramas for video browsing. In *Proceedings of the ACM Multimedia Conference*, pages 427–436, 1997.

[9] L. Teodoiso and W. Bender. Salient video stills: Content and context preserved. In *Proceedings of the ACM Multimedia Conference*, pages 39–46, 1993.

[10] Y. Tonomura, A. Akutsu, K. Otsuji, and T. Sadakata. Videomap and VideoSpaceIcon for anatomizing video content. In *Proceedings of INTERCHI*, pages 131–138, 1993.