Pixel Matching and Motion Segmentation in Image Sequences

Narendra Ahuja and Ram Charan

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign, Urbana, Illinois, U.S.A. 61801
e-mail: ahuja@stereo.ai.uiuc.edu, ramm@stereo.ai.uiuc.edu

Abstract. This paper presents a coarse-to-fine algorithm to obtain pixel trajectories in a long image sequence and to segment it into subsets corresponding to distinctly moving objects. Much of the previous related work has addressed the computation of optical flow over two frames or sparse feature trajectories in sequences. The features used are often small in number and restrictive assumptions are made about them such as the visibility of features in all the frames. The algorithm described here uses a coarse scale point feature detector to form a 3-D dot pattern in the spatio-temporal space. The trajectories are extracted as 3-D curves formed by the points using perceptual grouping. Increasingly dense correspondences are obtained iteratively from the sparse feature trajectories. At the finest level, which is the focus of this paper, all pixels are matched and the finest boundaries of the moving objects are obtained.

Keywords: Motion Segmentation, Perceptual Grouping, Pixel Matching, Triangulation, Feature Matching, Optical Flow.

1 Introduction

This paper describes a component of our work aimed at interpretation of image sequences. Given an image sequence containing an arbitrary number of rigid objects in motion, the objectives of the overall work are to identify feature points in the scene, obtain spatially dense trajectories of those points, segment moving objects, compute image flow at each pixel, and derive a qualitative description of the scene structure and dynamics from the image sequence. Such qualitative interpretation of the image sequence is useful for a variety of applications such as traffic scene analysis, biological image analysis and aerial image understanding. The focus of this paper is on the detection of pixel flow trajectories. Next section reviews some related previous work. Section 3 summarizes the steps of coarse-to-fine detection of sparse feature trajectories. Section 4 then gives the details of the algorithm for finding pixel flow trajectories which is the objective of this paper. Section 5 presents experimental results and Section 6 presents concluding remarks.

2 Previous Work

Much previous work has addressed feature correspondences between two frames which are used for 3D motion and structure estimation. The work on optical flow has been concerned with the detection of pixel correspondences across two frames closely separated in time. Both of these areas have seen a significant amount of activity and we will not attempt to provide the long list of references here. For motion analysis from long image sequences, several researchers have addressed the problem of feature correspondence in the past. Sethi and Jain[1] formulate this problem as an optimization problem and propose an iterative algorithm which they call the Greedy Exchange algorithm. Sethi et al.[2] propose a relaxation algorithm for feature point matching where the formation of smooth trajectories over space and time is favored. This method requires the correct initial correspondence and was used on very few feature points. Rangarajan and Shah [3] have proposed a noniterative polynomial time approximation algorithm by minimizing a proximal uniformity cost function. Cheng and Aggarwal[4] propose a two stage hybrid approach to the trajectory finding problem. The first stage extends the trajectories and the second one attempts to correct any errors. Debrunner and Ahuja[5] uses a two stage method to finding trajectories. The first step computes short feature paths of constant velocity. The second step deals with joining the feature paths into trajectories.

This paper exploits known trajectories of features for finding pixel trajectories. In the next section, we first briefly present our approach to coarse-to-fine feature trajectory detection which is based on perceptual grouping.

3 Feature point matching and segmentation

This section summaries the first two steps of the algorithm to analyze a single batch of frames.

3.1 Sparse Correspondences

The goal of the first step is to find correspondences of sparsely located feature points and segment the moving objects. We have no knowledge of number of moving objects or their motions in the scene. The only assumptions made here are that they are rigid objects and are moving smoothly. A perceptual grouping technique is used to achieve this goal. Feature points are detected in each image. The images in a batch are stacked to form a 3D dot pattern. The 3D Voronoi tessellation defined by the points is constructed. The batch size must be chosen carefully. If the number of frames in the batch is small, then the Voronoi tessellation may not represent the 3D structure well due to lack of data along time axis. A large batch size, on the other hand, may contain frames in which a moving object enters or exits the visual field which results in trajectories that last for only part of the batch. This makes their detection more difficult. Further, it significantly adds to the computational load. Since in the algorithm used in this

work, the batch analysis is used to estimate final correspondences for a central pair of frames, it is sufficient to ensure that the Voronoi structure associated with the dots in several central frames are correct, i.e., are not affected by lack of image frames.

The result of this step is a curve segment joining the feature points in all frames. In general, some feature points are missed and some new feature points appear from frame to frame. Therefore, the length of a curve segment or trajectory varies. The feature point locations along a trajectory in the two central frames are considered as correspondences. Such correspondences may be used by any of a number of motion and structure algorithms [6, 7] that require point feature correspondences.

Once the feature point correspondences are known, the feature points in each frame are segmented into different moving objects based on similarity of motion. Local adjacency among points is made explicit through the Delaunay triangulation when a Delaunay edge connects a point with its Voronoi neighbors. Segmentation is then achieved by identifying Delaunay edges connecting points belonging to different objects as well as those inside a single object. In general two independently moving objects differ in the magnitude and direction of their 3D motion. However 2D direction alone is a strong basis to discern if two points belong to the same or different objects, and in fact is the stronger cue for motion boundary perception in human vision. Edge identification is done by comparing the motion vectors at its two vertices.

3.2 Dense Correspondences

The second step performs matching of finer level features with the help of coarsest level matches already identified through perceptual grouping as described above. The finer level features are more densely distributed and therefore they improve the accuracy of detected moving object shapes relative to those segmented at the coarsest level (Fig. 1). A sequence of coarse-to-fine matching steps described in the following paragraph is iterated to the finest level of detected features, yielding the highest density of feature matches. The motions of denser features are predicted based on the known motions of nearby, coarser level features.

The coarser level features near a detected fine level feature may belong to one or more differently moving objects (Fig. 2). Therefore the detected feature may have any of these motions. Accordingly all candidate motions are considered. For a new fine level feature, each of the available motion estimates predicts a different matching location in the next frame. Only one of these is correct and to be selected. Around each such predicted match, feature points are tested to identify those whose neighbors gray levels are well correlated with those of the fine feature point being examined. Since the estimates derived from coarse level are more approximate due to lower feature density, the newly identified candidate matches serve as more accurate alternative motion estimates. Selection among these candidates is now performed by enforcing the spatial continuity of motion.

The unique matched pairs selected for fine level feature points comprise denser correspondences than the coarse level correspondences inherited from the coarser level. These finer level correspondences can again be segmented into distinctly moving objects in the same way as done at the coarser level. The resulting segmentation follows the object boundaries more accurately. These fine level features along with the segmentation are the final result of the iteration. The coarse to fine motion estimation and segmentation is continued at increasingly fine spatial scales until the feature detector no longer gives useful new features.

4 Pixel Correspondences

The method for computation of pixel matches is similar to the method for dense point-features correspondences. Once the finest-level features are found and matched, the remaining pixels are matched using raw intensity information. This is done using intensity correlation, which results in the pixel correspondence for the whole image, as discussed in Sections 4.1 and 4.2, or pairwise image flow, as also computed by the algorithms in [8, 9, 10, 11, 12]. After this step, the motion field is segmented to obtain the boundaries of the moving objects, as discussed in Section 4.3. An attempt is made to integrate the information present in the motion edges and intensity edges to obtain better estimates of the scene structure.

4.1 Finding Candidate Matches for Pixels

Consider a pair of frames (frame 1 and frame 2) in which the finest-level features have been matched. The Delaunay triangulation is computed for the matched points in frame 1 as shown in Fig. 1. The 2D motion of every pixel in this triangle is interpolated from the three vertex motions. The three vertices of a triangle may belong to one, two, or three different moving objects in the scene as shown in Fig. 2. Therefore, all of the pixels in a triangle will have one, two, or three motions for computing the candidates for matching.

As shown in Fig. 3, for each pixel P with coordinates (x, y) and an estimated 2D motion (emx, emy), an $r \times r$ window is selected, centered at the location $(x + emx, y + emy)$ in the next frame. Gray-level correlation is computed at the location $(x + emx, y + emy)$ and at its eight neighboring pixels. The pixels for which this correlation is above some threshold are also considered as candidate matches of P. This is repeated for each possible motion of P to generate all possible candidates for matching. From these candidates, the five with the highest correlations are retained to select the best match in the next step.

4.2 Obtaining the Best Pixel Match

The candidate matches are used by a relaxation algorithm to find the best match for each pixel. The support for a certain candidate match for a pixel is computed

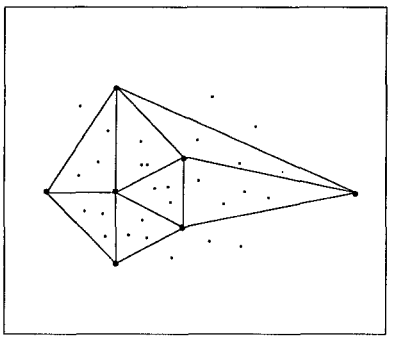

Fig. 1. Vertices of the triangles are the matched, coarse level feature points, and the other dots are new, finer level feature points.

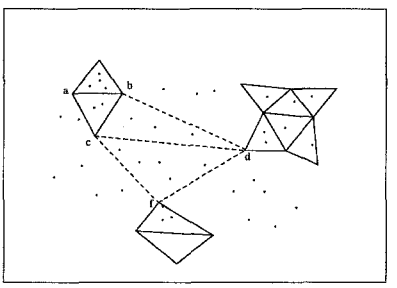

Fig. 2. Triangle *abc* has all vertices from same object, triangle *bcd* has its vertices from two objects and triangle *cdf* has its vertices from three objects.

from the four adjacent pixels, analogous to the Voronoi neighbors in case of feature points. Eventually, two or more pixels may be matched to the same pixel in the next frame. To avoid such situations, the algorithm is used for finding matches from frame 2 to frame 1 also. If a pixel i in frame 1 matches a pixel j in frame 2 while initiating matching from frame 1, then pixel j in frame 2 should match pixel i in frame 1 when matching is initiated in frame 2. Pixel matches not satisfying this two-way constraint are discarded. Near the boundary of moving objects, matches will not be obtained for those pixels corresponding to the scene points that are visible in one frame but not in the other frame. This yields thick bands of unmatched pixels comprising self-occlusion regions of a moving object.

4.3 Segmentation of Moving Objects

The boundaries of the moving objects are obtained based on the similarity of the motion field. However, the detected object boundaries will have errors whenever the motion estimates of pixels are erroneous. This will happen whenever, for example, the number of features in an image part is sparse, leading to rather large triangles. Therefore, the estimates of candidate matches of points within the triangle (for finer-level features or pixels) will contain large errors because

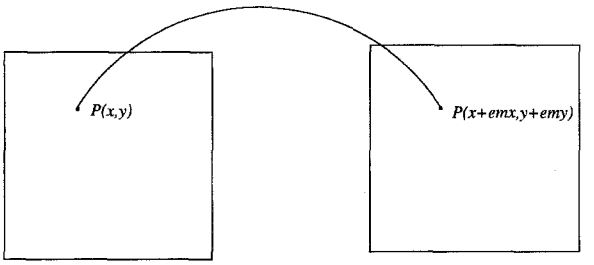

Fig. 3. Pixel P located at (x, y) in frame 1 and its estimated location at $(x + emx, y + emy)$ in frame 2.

the estimates are based on linear interpolation of the vertex motions. This will propagate errors down to both feature matching at the finest scale and pixel matching. It is discussed later as how intensity structure can be used to help overcome some of these shortcomings.

The pixel motion estimates yield a motion field whose discontinuities would ideally correspond to the boundaries of objects moving differently. However, the discontinuities in the estimated motion field are noisy and are jagged versions of object boundaries. The jaggedness is caused by the remaining errors in pixel motion estimates, which are particularly hard to eliminate in the vicinity of the object silhouettes where the object surfaces gradually turn away from the viewer. This is because the image intensity variations of a scene point with motion and the perspective compression of a surface patch are particularly severe under such conditions, resulting in severe correlation-based matching errors.

To address these problems, it is assumed that the object (motion) boundaries coincide with intensity edges. This assumption is valid for a large fraction of real scenes. Consequently, any motion boundary almost coinciding with an intensity edge is interpreted as a noisy version of the intensity edge that is assumed to be the motion and object boundary. Accordingly, the algorithm searches for all intensity edges and tests for similarity of motion estimates on its two sides. If the motion estimates on the two sides are similar, then the intensity edge is merely a marking on the object. If not, the edge is tested as a motion edge.

To this end, intensity edge contours are also obtained which yield regions of similar gray levels. Every pixel is given a label that corresponds to its region. To detect the boundary pixels of the moving objects, a small window centered around an edge pixel is considered. Average motion vectors are computed on the two sides separated by the intensity edge and a measure of similarity between them is computed. If they are not similar then the edge pixel is interpreted as on a motion boundary. By repeating this procedure for all of the intensity edge pixels, the pixels on the motion boundaries of the moving objects are detected.

Another approach to obtain motion boundaries is to merge similar motion regions. For a pair of neighboring regions, average motion vectors are computed for these two regions by considering all the pixels within a small neighborhood of the intensity edge they share.

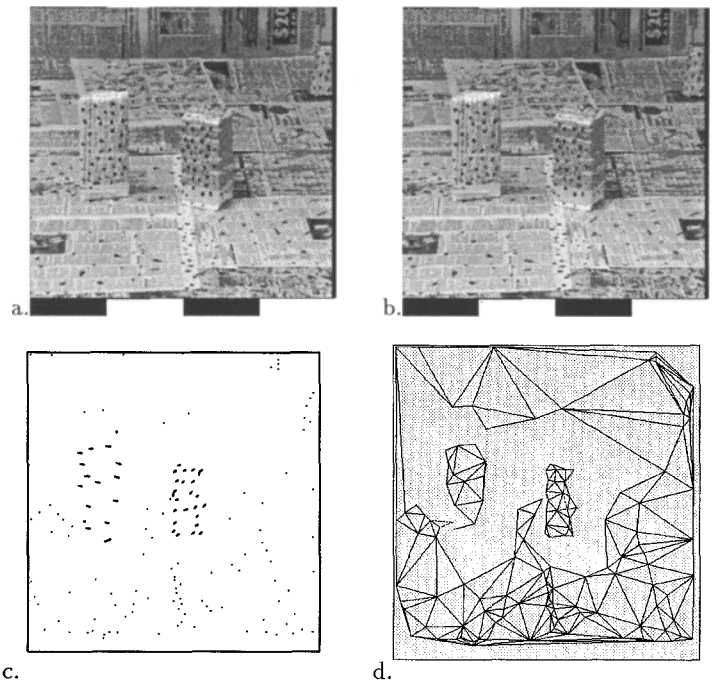

Fig. 4. (a) One frame in an image sequence. (b) Point features detected in the frame. (c) Feature matching at a coarse level. (d) Motion segmentation at the coarse level.

5 Experimental Results

Experimental results obtained by the algorithm presented are shown in this section for two different image sequences. One of these was obtained in our laboratory and the other one shows an outdoor scene.

For the first example a sequence was taken by moving objects with known motion between successive frames as shown in Fig. 4. Two objects are moving. Fig. 4a is the 3rd frame in the sequence and Fig. 4b shows point features detected in this frame. Point correspondences at a coarser level from perceptual grouping between frame-3 and frame-4 are shown in Fig. 4c. The segmentation of moving objects in frame-3 is shown in Fig. 4d. Point correspondences at a denser level are shown in Fig. 5a and segmentation of moving objects is shown in Fig. 5b. Finally, pixel matches are obtained for every pixel in the image and Fig. 5c shows the motion field (for every 5th row and 5th column pixel) between frame 3 and frame 4. Fig. 5d shows the segmentation of motion field.

For the second example the camera is moving while acquiring the image sequence. Fig. 6a is frame 3 in the sequence and Fig. 6b shows the motion field at pixel level.

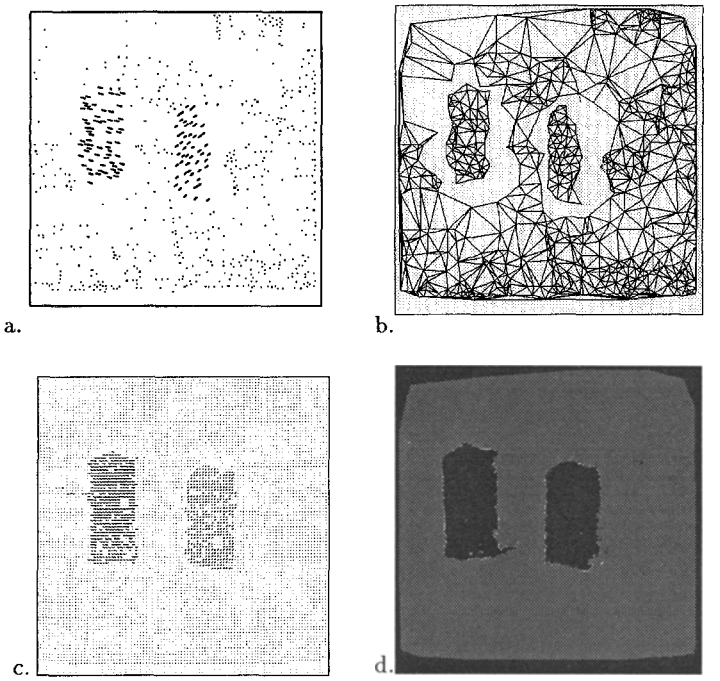

Fig. 5. (a) Feature matching at a dense level. (b) Motion segmentation at the dense level. (c) Motion field (every 5th row and 5th column). (d) segmentation of motion field at pixel level.

6 Conclusions

In this paper, an algorithm is proposed to obtain flow of image points across a sequence. Features as well as pixels are segmented into different moving objects. Experiments were conducted with both laboratory image sequences as well as natural sequences. Intensity maxima and minima were used as features. Other feature point detector may be used to obtain the feature points required for this algorithm.

Perceptual grouping yields reliable correspondences. Although they are sparse, most correspondences found are correct. Inside homogeneous regions, sometimes the pixels are not matched uniquely. When an object also has a motion boundary with little change in texture/gray level across it, the boundary may not be found accurately. We have tried to integrate the intensity edge information with the motion field to more accurately locate the boundaries of the moving objects. Some of the pixels near the boundary of a moving object that are within the stationary background are not matched correctly.

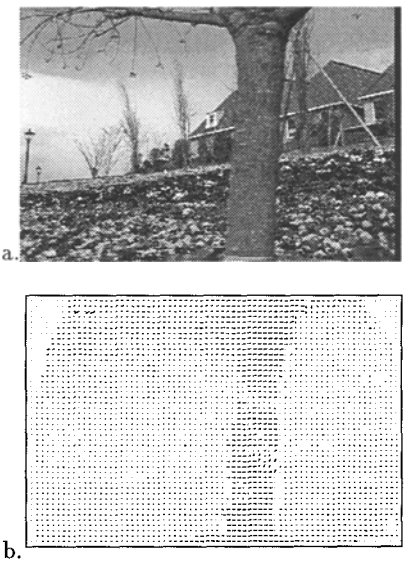

Fig. 6. (a) Frame 3 in the sequence. (b) Motion field (every 5th row and 5th column).

References

1. I. K. Sethi and R. Jain, "Finding trajectories of feature points in a monocular image sequence," *IEEE Transactions on Pattern Analysis and Machine intelligence*, vol. PAMI-9, January 1987.
2. I. K. Sethi, V. Salari, and S. Vemuri, "Feature point matching using temporal smoothness in velocity," in *Pattern Recognition Theory and Applications* (P. A. Devijver and J. Kittler, eds.), pp. 119–131, New York: Springer-Verlag, June 1986.
3. K. Rangarajan and M. Shah, "Establishing motion correspondences," *CVGIP: Image Understanding*, vol. 54, pp. 56–73, July 1991.
4. C. L. Cheng and J. K. Aggarwal, "A two-stage hybrid approach to the correspondence problem via forward searching and backward correcting," in *Proceedings of the International Conference on Pattern Recognition*, pp. 173–179, 1990.
5. C. Debrunner and N. Ahuja, "Motion and structure factorization and segmentation of long multiple motion image sequences," in *European Conference on Computer Vision*, pp. 217–221, 1992.
6. J. K. Aggarwal and Y. F. Wang, "Analysis of a sequence of images using point and line correspondences," in *Proceedings of the International Conference on Robotics and Automation*, 1987.
7. J. Weng, N. Ahuja, and T. Huang, "Motion and structure from point correspondences: A robust algorithm for planar case with error estimation," in *Proceedings of the International Conference on Pattern Recognition*, 1988.
8. J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 236–242, 1992.

9. J. H. Duncan and T. C. Chou, "The detection of motion and computation of optical flow," in *Proceedings of the International Conference on Computer Vision*, pp. 374–382, 1988.
10. D. Heeger, "Model for the extraction of image flow," *Journal of the Optical Society of America*, pp. 1455–1471, 1987.
11. B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–204, 1981.
12. A. Singh, *Optical Flow Computation: A Unified Perspective*. Los Alamitos: IEEE Computer Society Press, 1992.