Motion Based Segmentation

Shmuel Peleg* and Hillel Rom†

Department of Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, ISRAEL

Abstract

Images of scenes with moving objects, taken from a moving camera, have several motion components. The dominant image motion is usually generated by the camera motion which affects the stationary background. Moving objects contribute to other motion components. An iterative method is described for segmenting image sequences into independently moving regions while computing the motion parameters of each region. In each iteration, image points are classified to regions based on their consistency with the different motion estimates, and motion estimates are then updated using the obtained regions. The motion estimates and the segmentation improve with every iteration, and the iteration stops when a stable segmentation is obtained. Accurate motion parameters are recovered for each segment. The proposed process is performed directly on grey-level images, and does not require detection of special feature points and the computation of point correspondence. It is also faster and more robust than optical flow based segmentation methods.

1 Introduction

The segmentation of moving objects in image sequences becomes harder when the camera itself is moving, and the motion parameters of the camera and of each object are unknown. Common approaches to motion based segmentation use optical flow [2]. The optical flow is computed at every image point, and is then used for segmentation [1, 11, 10, 13]. Adiv [1] shows that given the optical flow, segmentation of the scene into independently moving planar objects is possible. He partitions the flow field into connected segments of flow vectors where each segment is consistent with a rigid motion of a planar surface. Segments are then groups under the hypothesis that they are induced by a single rigidly moving object.

Optical flow based methods have all the drawbacks associated with the computation of optical flow. There is very little motion information in the small windows used for optical flow computation, resulting in inaccuracies at motion boundaries, within uniform regions, and in the presence of noise. Increasing the window size reduces the ambiguity, but there is a larger chance that the window will cover more than one motion.

We propose to combine segmentation with motion computation to overcome the problem of several motions in one large region. While computing motion on large regions, thus reducing noise and ambiguity, the segmentation assures that only relevant points will participate in motion computation.

Analysis of camera motion relative to a static scene has been extensively treated [5, 7, 8, 9, 12]. Recovering this uniform motion does not require the computation of optical flow at every image point. Negahdaripour and Horn [12] present a method for recovering the observer's motion relative to a planar surface directly from the spatial and temporal derivatives of the image brightness. Horn and Weldon [7] follow this approach, and using a least squares method determine the observer's motion in the special cases of known depth, pure rotation, or known rotational motion component.

We use a method based on [7] for recovering uniform motion of a set of pixels to perform motion-based segmentation, in conjunction with an iterative clustering process. This process converges from initial motion estimates for each object to final accurate motion parameters. Segmentation is obtained by clustering all pixels based on their agreement with the motion parameters of the different objects. The problems of the optical flow based methods are avoided by using larger regions. The motion recovery method requires the knowledge of the depth of the scene. Therefore, the segmentation process is currently limited to two special, but important, cases: when the depth of the scene is either constant or otherwise known.

2 Recovering Uniform Motion

In this section we consider the problem of recovering the camera motion relative to a rigid unknown world. We present the *brightness change constraint equation* of Negahdaripour and Horn [12], and following Horn and Weldon [7] we use a least squares method to solve the equation for the motion parameters in the case of known depth.

---

*Part of this work was done while this author was with David Sarnoff Research Center, Princeton, NJ 08543.

†Current Address: Institute of Robotics and Intelligent Systems, University of Southern California, Los Angeles, California 90089.

This research was supported by a grant from the Israeli National Council for Research and Development.

CH2898-5/90/0000/0109$01.00 © 1990 IEEE

2.1 Constraint Equation

Under perspective projection, a scene point $P = (X, Y, Z)$ is imaged at the picture coordinates $p = (x, y) = (X/Z, Y/Z)$, where the world (X, Y) coordinates are parallel to the image (x, y) coordinates, and the optical axis is along the Z axis.

When the camera moves with translational velocity $\tau = (U, V, W)$ and rotational velocity $\omega = (A, B, C)$, the *optical flow* (x_t, y_t), is given by [10]:

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} Axy - B(x^2+1) + Cy + \frac{-U+xW}{Z} \\ -Bxy + A(y^2+1) - Cx + \frac{-V+yW}{Z} \end{pmatrix}. \quad (1)$$

The brightness of the image changes with the motion due to change in factors such as lighting or orientation. However, it is reasonable to assume that the brightness of a small patch does not change by motion (refer to [7] for a discussion on this assumption). Under this assumption on the image brightness, I, we have $dI/dt = 0$. Expansion of the total derivative of I using the chain rule results in

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0, \quad (2)$$

which can be rewritten as

$$x_t I_x + y_t I_y = -I_t. \quad (3)$$

Substituting from Equation (1) for x_t and y_t and rearranging (refer to [7] for details) we get the *brightness change constraint equation* of Negahdaripour and Horn [12] which can be written in compact form as

$$v \cdot \omega + \frac{s \cdot \tau}{Z} = -I_t, \quad (4)$$

where

$$s = \begin{pmatrix} -I_x \\ -I_y \\ xI_x + yI_y \end{pmatrix}; \; v = \begin{pmatrix} +I_y + y(xI_x + yI_y) \\ -I_x - x(xI_x + yI_y) \\ yI_x - xI_y \end{pmatrix}. \quad (5)$$

2.2 Motion Computation

The camera's motion (τ, ω) and the scene depth $Z(x, y)$ can be computed from an image region by solving Equation (4) using the measurements of s, v, and I_t as defined in Equations (5). Computing both camera motion and scene depth is impossible in the general case, but this can be done in some special cases. We solve the equations in the case of known depth. This case, and a few others, are also presented in [7].

Motion is computed by finding the parameters (ω, τ) which minimize the total deviation from the brightness change constraint equation (4) in the region. This deviation is defined as the sum over all region points:

$$E(\omega, \tau) = \sum_{xy} [I_t + v \cdot \omega + (1/Z)s \cdot \tau]^2. \quad (6)$$

The deviation can be minimized by differentiating $E(\omega, \tau)$ with respect to ω and τ, and setting the results to zero. A set of two vector equations is obtained. This set is actually a set of six linear equations with six unknowns which could be solved for the six motion parameters $(\omega, \tau) = ((A, B, C), (U, V, W))$.

$$\begin{aligned} \left[\sum_{xy} \frac{1}{Z}^2 ss^T\right]\tau + \left[\sum_{xy} \frac{1}{Z} sv^T\right]\omega &= -\sum_{xy} I_t \frac{1}{Z} s \\ \left[\sum_{xy} \frac{1}{Z} vs^T\right]\tau + \left[\sum_{xy} \frac{1}{Z} vv^T\right]\omega &= -\sum_{xy} I_t v \end{aligned} \quad (7)$$

2.3 Iterative Refinement

It may be impossible to compute the camera motion using Equations (7), as the brightness change constraint equation (4) holds only for small rotations and translations. For two images I_1 and I_2 recorded at a discrete time interval, the displacements between the images might not be sufficiently small, and the motion recovery method described above could fail. To ensure the accuracy of computed motion in most general cases, motion computation is iterated in two different ways.

To overcome the difficulties presented by large motions, a multi-resolution approach is used. From each input image, a sequence of reduced resolution images is generated by repeatedly blurring the image, followed by subsampling by a factor of two. Each image is therefore represented at several levels of resolution. For an input image of size 256×256, for example, reduced resolution images of sizes 128×128, 64×64, and 32×32 are generated. The registration is first performed between the two lowest resolution representations of the input images. At this low resolution level, all displacements are small with respect to the pixel size. Under these conditions motion computation is more accurate. The obtained transformation coefficients are used to warp one of the next higher resolution images towards the other. The resulting two higher resolution images have smaller pixel displacements, allowing the residual motion to be computed. The total motion is now applied to warp the next higher resolution images, and the process continues until the two highest resolution images are registered.

To increase accuracy, the following iterations are performed at every resolution level [9]:

1. Initially assume no motion between the frames.
2. Compute approximations to the motion parameters by solving Equations (7). Add the computed motion to the existing motion estimate.
3. Warp I_1 towards I_2 using the current motion estimates, and return to Step 2 with the warped image I_1.

The warped version of I_1 gets closer to I_2 at every iteration, and as the residual values of ω and τ computed in Step 2 get smaller they become more accurate. The process is stopped when the corrections to ω and τ approach zero.

It should be noted that of all the coefficients used in Equations (7), only I_t changes from one iteration to the next. Therefore, 36 of the 42 coefficients are computed

only once. Only six coefficients, depending on I_t, are recomputed at each iteration.

The above method recovers the camera motion relative to a static world. A modification of the algorithm enables the recovery of the camera motion even when some small moving objects are present in the scene. This is achieved by multiplying the contributions of each pixel in Equation (7) with weights which are inversely proportional to the brightness difference, I_t, between the two images. In the present algorithm, weights used for every pixel (x, y) are $1/(1+|I_t(x,y)|)$. When the two images are almost registered, a large brightness difference usually corresponds to moving objects, and these points will have a smaller effect on the computation of motion parameters.

The ability of the algorithm to recover the camera motion in a non static world is an important feature which is used by the segmentation algorithm presented in the next section.

3 Motion Based Segmentation

Image sequences will be segmented into independently moving objects, using the registration algorithm for recovering uniform motion described in Section 2.

Let I_1 and I_2 be two consecutive frames in an image sequence. For simplicity, we assume that the scene consists of one moving object, and that the background is moving due to camera motion. Let m_0 and m_1 denote the *motion parameters* of the background and the object respectively. Each motion parameter consists of two vectors, the translation vector $\tau = (U, V, W)$ and the rotation vector $\omega = (A, B, C)$.

3.1 Iterative Segmentation

Step 1. Deriving Initial Motion Estimates

The registration algorithm of Section 2 is first applied to frames I_1 and I_2. Assuming that the moving object is much smaller than the background, the resulting motion parameter is an estimate for m_0, the background's motion. We will denote this estimate by m_0^0 (the superscript being the iteration number).

I_1 is then warped towards I_2 using the motion estimate m_0^0. The warped I_1 is now aligned with I_2, and the difference between the two images is computed. High absolute values in the difference image are therefore a coarse approximation to the location of the independently moving objects. The largest region having high difference is identified, and motion parameters are computed for this region by applying the registration algorithm to compute m_1^0 - an initial estimate to the motion of the moving object.

Step 2. Segmenting Using Motion Estimates

For every pixel location p in frame I_1, m_0^0 and m_1^0 are used with Equation (1) to calculate the two possible locations of p in the frame I_2: $m_0^0(p)$ and $m_1^0(p)$ respectively.

If p is part of the object which moved m_1, then its grey-level $I_1(p)$ in frame I_1 should be similar to the grey-level $I_2(m_1^0(p))$ at location $m_1^0(p)$ in frame I_2. On the other hand, if p is in the background which moved m_0, then $I_1(p)$ should be similar to the grey-level $I_2(m_0^0(p))$ at location $m_0^0(p)$ in frame I_2.

Let $R(p)$ be the ratio between the grey-level differences when using motion m_1^0 and the grey-level differences when using motion m_0^0:

$$R(p) = \frac{|I_1(p) - I_2(m_1^0(p))|}{|I_1(p) - I_2(m_0^0(p))|}. \quad (8)$$

Pixel p will be classified as *background*, moving motion m_0^0, when $R(p) \gg 1$, and as *object*, moving m_1^0, when $R(p) \ll 1$. Pixel p is classified as *undecided* when $R(p) \approx 1$.

In actual implementations, the differences in Equation (8) are not single pixel differences, but are taken on a 3×3 local neighborhood of the pixel. Although this limits the accuracy of the segmentation to be within one pixel around the edges of the object, it has been found to be necessary when applying the algorithm to noisy images. After classification, object pixels are smoothed by morphological opening and closing [6] to eliminate random noise.

Step 3. Calculating New Motion Estimates

Motion is computed separately for the region classified as object and for the region classified as background. The two new motion parameters computed by the registration algorithm, m_0^1 and m_1^1, are improved over m_0^0 and m_1^0, since the segmentation in Step 2 improves the separation of the scene into object and background.

Steps 2 and 3 above are repeated using m_0^i and m_1^i as the initial estimates to compute m_0^{i+1} and m_1^{i+1} and so on. The process is terminated when there is no (or very little) change in the values of the motion parameters from one iteration to the next. After termination of the iterations an accurate segmentation of the image into object and background is obtained, as well as an accurate estimate of their motion.

The algorithm can be generalized to include scenes with more than one object by using more motion parameters, and by applying the algorithm to $m_0^0, m_1^0, m_2^0 \ldots$ to compute $m_0^1, m_1^1, m_2^1 \ldots$ and so on. In this case Equation (8) is being applied between the two motions which cause the lowest temporal grey-level differences.

3.2 Experimental Results

Two examples for motion segmentation are shown. One is on synthetic random noise images and the other is on a real scene. In both cases the algorithm has been applied to two consecutive images from the sequence, in which the background and the object have translated and rotated in different directions.

Figure 1 presents a synthetic example: image (a) is a random noise image of size 128×128. Image (b) is a copy of (a) translated by $(-4, 1)$ with a square of size 25×25, representing the object, translated by $(-2, -2)$. The motion parameters are therefore $(\tau, \omega) = ((-4, 1, 0), (0, 0, 0))$ for the background and $((-2, -2, 0), (0, 0, 0))$ for the central square. The motion parameters recovered by the algorithm after two

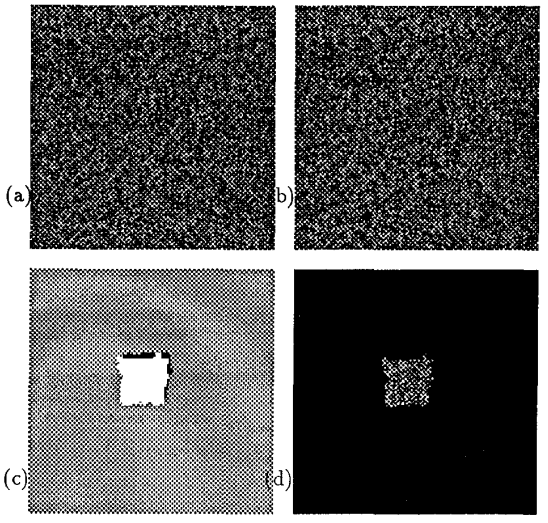

Figure 1: Motion segmentation on random noise images.
a-b) Original sequence images.
c) Segments after 2 iterations: object is white, background is grey and occluded regions are black.
d) The grey-level of object regions.

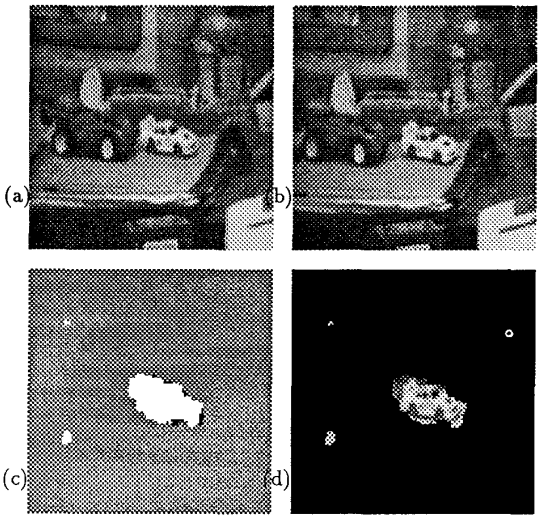

Figure 2: Motion segmentation of an imaged scene.
a-b) Original sequence images.
c) Segments after 2 iterations: object is white, background is grey and occluded regions are black.
d) The grey-level of object regions.

iterations were $((-3.97, 0.98, 0.00), (0.00, 0.00, 0.09))$ and $((-2.24, -1.83, -0.01), (0.00, 0.00, 0.21))$ respectively (rotation angles are in degrees). The segmentation as shown in Figure 1 is accurate within one pixel.

Figure 2 presents an example on a real scene: (a) and (b) are the first and second images. The camera has moved between the two images, and the white car has moved independently. The motion parameters found by the algorithm after two iterations were $((-6.48, 2.09, 0.00), (0.00, 0.00, 0.00))$ for the background, and $((0.60, -1.59, 0.05), (0.00, -0.00, 2.94))$ for the object (rotation is in degrees). As the background regions around the car are smooth, some of their pixels are also classified as an object in Figure 2.

3.3 Detecting Occluded Areas

Regions that are exposed in one frame and occluded in another frame cause special problems in segmentation and motion estimation. Consider the case of a background region in frame I_1 which becomes occluded by an object in frame I_2. Pixels in this region do not have matching locations in frame I_2. Therefore they either remain unclassified, or each is classified randomly. In order to have accurate classification, these regions should be identified, and either not joined to any other region, or joined to the background.

Figure 3 presents a simple one-dimensional example for such occlusion. Let X_i denote background pixels and O_i denote object pixels. Let us assume that values at all pixels are unique so that matching is trivial.

Figure 3.a-b are the sections of the two frames, I_1 and I_2. As shown, the background is stationary, and the object has moved 2 pixels to the right.

Locations 7 and 8 in frame I_1 have no match in frame I_2, and cannot be classified correctly (Figure 3.c). If occlusion is not detected, they may be classified as either object or background, making the boundary inaccurate.

To improve the segmentation, the inverse problem is examined: finding the motion from I_2 to I_1. From the discontinuity of the detected motion values between locations 8 and 9 (Figure 3.d) and from the fact that the relative motion between the object and the background is 2, we learn that there should be a discontinuity of motion between locations 6 and 7 of frame I_1 and that locations 7 and 8 in I_1 are occluded in I_2. Due to noise, we might also get discontinuities around locations 4 and 5, but in this case analysis of the motion values will reveal a contradiction.

The full segmentation process, taking occlusions into account, is summarized as follows: Motion segmentation is first performed from frame I_1 to frame I_2. The unclassified regions of frame I_1, due to lack of preferred direction when using Equation (8), are marked as suspect occluded region in frame I_2. The motion segmentation from frame I_2 to frame I_1 is then computed. Using the motion discontinuities from frame I_2 to frame I_1, the exact occluded areas

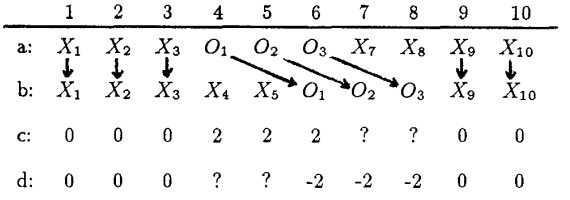

Figure 3: A one-dimensional example for occlusion (see text). a) Frame I_1; b) Frame I_2. c) Translation from I_1 to I_2. d) Translation from I_2 to I_1.

are marked, giving more accurate segmentation. The segmentation results in Figures 1 and 2 include the application of the above process.

4 Concluding Remarks

An algorithm has been presented for segmentating image sequences into independently moving objects, based only on motion information. The algorithm is fast, accurate and relatively immune to noise. Segmentation is an important ingredient even when only motion computation is of interest. It has been found that the predominent motion can be computed accurately using the method described in Section 2 if sufficient iterations are performed. However, to compute the other motions in the scene, the effect of the prominent motion must be removed by segmentation. Without segmentation, the effects of the predominent motion mask out the other motions. A method for removing the effect of one motion without requiring segmentation is described in [4].

Experiments with the algorithm show that it gives accurate results. The rotation angles, however, must be relatively small due to the use of Taylor series approximation in the analysis. In the case of image sequences taken at video rate this constraint is usually met. The segmentation found by the algorithm could be incorrect, but such error can be detected by checking the error after transforming a detected segment in the first frame to its location in the second using the associated motion parameters.

The presented experimental results were obtained under the assumption of constant depth of the scene (i.e. 2-D scenes). This case covers applications such as aerial or satellite images, and some microscopical images. This method should also work with 3-D scenes if the depth of the scene is known. The depth information may be obtained either directly from a range finder or using the disparity information provided by a stereo system [3]. Our method will fail in more complicated 3-D cases. This is due to the inherent ambiguity in the brightness change constraint equation. In these cases a more general method, based on the optical flow of every point in the image, is probably needed.

A method for detecting regions which are occluded in one image and revealed in another has also been presented. This method was used for refining the segmentation around the boundaries of the objects.

References

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):384–401, July 1985.

[2] P. Anandan. A unified perspective on computational techniques for the measurment of visual motion. In *International Conference on Computer Vision*, pages 219–230, London, May 1987.

[3] S.T. Barnard and M.A. Fischler. Computational stereo. *ACM Computing Surveys*, 14(4):553–572, December 1982.

[4] J.R. Bergen, P.J. Burt, R. Hingorani, and S. Peleg. Transparent-motion analysis. In *European Conference on Computer Vision*, Antibes, France, April 1990.

[5] E. De Castro and C. Morandi. Registration of translated and rotated images using finite fourier transforms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(5):700–703, September 1987.

[6] R.M. Haralick, S.R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(4):532–550, July 1987.

[7] B.K.P. Horn and E.J. Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76, June 1988.

[8] T.S. Huang and R.Y. Tsai. Image sequence analysis: Motion estimation. In T.S. Huang, editor, *Image Sequence Analysis*, pages 1–18. Springer-Verlag, Berlin, 1981.

[9] D. Keren, S. Peleg, and R. Brada. Image sequence enhancement using sub-pixel displacements. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 742–746, Ann Arbor, Michigan, June 1988.

[10] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of The Royal Society of London B*, 208:385–397, 1980.

[11] D.W. Murray and B.F. Buxton. Scene segmentation from visual motion using global optimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(2):220–228, March 1987.

[12] S. Negahdaripour and B.K.P. Horn. Direct passive navigation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(1):168–176, January 1987.

[13] A.M. Waxman and K. Wohn. Contour evolution, neighborhood deformation and global image flow: Planar surfaces in motion. *International Journal of Robotics Research*, 4(3):95–108, Fall 1985.