Tracking Based Motion Segmentation under Relaxed Statistical Assumptions

King Yuen Wong

Minas E. Spetsakis

Technical Report CS-2003-09

Oct. 14 2003

Department of Computer Science

4700 Keele Street North York, Ontario M3J 1P3 Canada

Tracking Based Motion Segmentation under Relaxed Statistical Assumptions

King Yuen Wong, Minas E. Spetsakis

Department of Computer Science, Centre of Vision Research, York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada

Abstract

Many Computer Vision algorithms employ the sum of pixel-wise squared differences between two patches as a statistical measure of similarity. This silently assumes that the noise in every pixel is independent. We present a method that involves a much more general noise model with relaxed independence assumptions but without significant increase in the computational requirements. We apply this technique to the problem of motion segmentation that uses tracking to estimate the motion of each region and then we employ our statistic to classify every pixel as part of a segment or the background. We tested several versions of the algorithm on a variety of image sequences (indoor and outdoor, real and synthetic, constant and varying lighting, stationary and moving camera, one of them with known ground truth) with very good results.

Keywords: Motion Segmentation, Tracking, Varying Light, Optical Flow, Hypothesis Testing, Mahalanobis.

1. Introduction

Motion segmentation refers to the partition of pixels having similar optical flow into groups such that each group corresponds to the motion of the projection of an independently moving object. Motion segmentation could be done solely using optical flow but this is a chicken and egg problem. Accurate optical flow computation requires precise knowledge of the motion boundaries and motion boundaries require segmentation.

Previous motion segmentation techniques [30, 7, 33] assume either that the flow is precomputed usually by dividing the image into regions, computing flow in each region and then merging the regions with similar flow. The goodness of the segments produced is limited by the accuracy of the initial optical flow computation step. An extension to these methods is to use Expectation Maximization (EM) for merging optical flow into regions [5, 32, 14] but the number of segments which is required as input to the EM algorithm cannot be obtained easily. Another category of motion segmentation techniques [8, 17, 20] perform motion segmentation by iteratively estimating optical flow, warping the images according to flow and building up a model of the moving object.

A tracking algorithm measures and predicts the motion of a moving object over time. Contours [18, 28] corresponding to the silhouette of moving objects are commonly used feature for tracking. The coherence of a moving region [12, 6] corresponding to the projection of a surface of the moving object is another good basis for tracking. Color [9, 19] of a moving object is also frequently used in tracking. Instead of tracking attributes belonging

Fax: (416) 736-5872.

E-mail addresses: kywong@cs.yorku.ca (K. Y. Wong), minas@cs.yorku.ca (M. E. Spetsakis).

The support of NSERC (App. No. OGP0046645) and CITO is gratefully acknowledged.

to the moving object, an orthogonal tracking approach is to find the moving objects in a dynamic scene by performing image difference on the image frames with known background [35]. In all of the above approaches, an initial representation of the to-be-tracked object or its background is given to the tracker as input and the role of the tracker is to measure and predict the motion of the moving object representation over time.

Meyer and Bouthemy [24] tracked the motion of regions computed by a motion segmentation algorithm over time assuming a model for the motion and change of shape of the regions. They used Kalman filtering to merge the prediction of the model with the actual measurements from the motion segmentation algorithm. This model can be best described as tracking based on motion segmentation whereas ours is best described as motion segmentation based on tracking.

In this paper, we present a novel and efficient motion segmentation and tracking algorithm that segments an image sequence into regions corresponding to the motion of projection of independent moving objects. The algorithm can work with manually or automatically selected features for tracking and warps one image frame towards the next using the computed optical flow on the features. Pixels moving consistently with the features have small difference between the two frames and vice versa. The major contribution of the paper is that we come up with fairly accurate and efficient statistics (pixel-wise and patch-wise) that model the noise between the aligned frames. A pixel-wise statistic assumes the independence of noise between the neighboring pixels and is an appropriate model under many circumstances. But in reality, the noise in neighboring pixels is correlated. This effect is prominent when an image sequence is taken under varying lighting conditions. We develop a patch-wise statistic to model such dependence. We are able to compute the patch-wise statistic efficiently by introducing to Computer Vision the Sherman-Morrison-Woodbury identity which is a little used numerical analysis technique for inversion of the covariance matrix involved in the patch-wise statistic. By applying the identity, we are able to reduce the cost of computation from $O(k^6)$ to constant time where the size of the patch is $k \times k$. As an added advantage, the Sherman-Morrison-Woodbury identity could have applications to other Computer Vision problems. We present the results of running our algorithm using a variety of image sequences (indoor and outdoor, real and synthetic, constant and varying lighting, stationary and moving background).

2. Overview of the Approach

The basic idea for tracking based segmentation is to select a feature point (we do it both manually and automatically) that belongs to a particular object and track a small seed region around it. The tracking will give us the affine flow u_a between images I_N and I_{N-1}. If we warp I_{N-1} by u_a we obtain image ${}^{u_a}I_{N-1}$. The segmentation is now seemingly easy. We take the difference between I_N and ${}^{u_a}I_{N-1}$ and pixels whose difference is small should belong to the object and the rest should not. Unfortunately this kind of classifier cannot be as simple as this for many reasons:

- There are several kinds of random noise that corrupt the images.
- While many kinds of motion seem to be affine, in practice they are only approximately affine.
- Two different parts of the image might have identical color or texture and be aligned accidentally.
- Effects like lack of texture or change in illumination.

The problem with all the above is that they lead to a complex and expensive noise model. The most important contribution of this paper is that we explicitly model most of these forms of noise and propose a statistic for this model that is computationally efficient. We also developed a Maximum Likelihood Estimator (MLE) that fits the model parameters and tested our algorithm using both the MLE fitted parameters and a set of empirical parameters that were common to all sequences.

The experiments were done on a wide variety of sequences that included real and synthetic sequences, indoor and outdoor, with and without change in illumination, with stationary and hand-held cameras, short and long sequences, etc. We went as far as using our algorithm to facilitate the solution of the optical flow problem (by detecting the motion boundaries and incorporating the tracking information to reduce inter-frame motion) to compare our results with the ground truth of a standard sequence (Yosemite sequence).

In the rest of this section, we describe the major components of the algorithm:

(1) Feature (seed region) selection.

(2) Tracking by fitting of successively more refined flow models to the seed region.

(3) Elaborate but efficient noise model for motion segmentation. We provide both a pixel-wise model and a patch-wise model.

(4) Application of the algorithm to the computation of optical flow.

2.1. Feature Selection

In most cases, the initial tracking region, which we call seed region in this paper, is provided to the tracking module by the user or by another module. Since we might be interested in complete segmentation, we need to be able to identify enough seed regions to segment most of the image.

In the feature selection step, if we are doing it automatically, we extract potential features for tracking by identifying "corners" in an image frame. In most literature, the term "corner" means features that can be tracked reliably from frame to frame and not only points of maximal curvature. Unfortunately many points that have rich enough texture to be corners are not suitable because they straddle a motion boundary.

We detect corners in an image frame by the corner detection algorithm proposed by Tomasi and Kanade [31] with some speed-up modifications made by Benedetti and Perona [3]. The corner detection algorithm finds feature points that have good localization in all directions. Tomasi and Kanade argue that these are pixels whose smallest eigenvalue of the matrix M [22] is bigger than a threshold λ_t where

$$M = \begin{bmatrix} E_{xx} & E_{xy} \\ E_{xy} & E_{yy} \end{bmatrix},$$

$E_{xx} = \int I_x^2$, $E_{xy} = \int I_x\, I_y$, $E_{yy} = \int I_y^2$ over a small region and I_x, I_y are the spatial derivatives of the image. Benedetti and Perona speed up the method by evaluating the characteristic polynomial $P(\lambda_t)$ of the above matrix M.

$$P(\lambda_t) = (E_{xx} - \lambda_t)(E_{yy} - \lambda_t) - E_{xy}^2$$

where λ_t is a parameter specifying the corner strength. All points whose $E_{xx} - \lambda_t$, $E_{yy} - \lambda_t$ and $P(\lambda_t)$ positive are classified as corners, Among the identified corners, we select the the N strongest corners whose $P(\lambda_t)$ are the N biggest and randomly pick one of them. This will be the center of the small seed region (10×10 pixels in our experiments) used for tracking.

Once a seed region is instantiated from a randomly selected corner feature, we run our tracking and segmentation algorithm to segment out a region whose pixels move in a way consistent with the seed region. We classify a corner as good feature for tracking in subsequent frames if

(1) The segment output from motion segmentation/tracking step around the seed region overlaps significantly with the seed region itself. In our experiments, we set the overlap threshold to be 75%.

(2) The segment is not very small. We discard segments that are less than 1% of the area of the image.

(3) The segment does not overlap significantly with segments found so far. We discard segments that overlap more than 90% with the existing segments.

When a good feature and its associated segment is found, we keep track of all the corner features that are within the segment. For subsequent frames, if the tracked feature generates a segment that does not satisfy all of the above criteria as a good feature for tracking, we generate seed regions around other corner features inside the segment in an attempt to continue the tracking/segmentation of the region. In this way, we can still track the motion of a segment if some of its corner features become occluded during its motion trajectory.

If the seed region was selected manually, we keep track of it without replacing it with any other seed region.

2.2. Seed Region Tracking

We track the motion of the seed region by fitting successively a uniform integer flow model, a uniform subpixel flow model and an affine flow model to the tracked region. We compute the integer flow of the seed region R by minimizing its Sum of Squared Difference (SSD) between the $N-1^{th}$, N^{th} image frames I_{N-1}, I_N:

$$SSD(\vec{u},\alpha) = \sum_{\vec{x}\ in\ R} (I_{N-1}[\vec{x}] - \alpha\ I_N[\vec{x}+\vec{u}])^2 \tag{2.1}$$

The role of parameter α is to compensate for the light changes. We minimize $SSD(\vec{u},\alpha)$ with respect to α analytically.

$$\frac{\partial SSD(\vec{u},\alpha)}{\partial \alpha} = 0$$

which leads to

$$\alpha = \frac{\sum_{\vec{x}\ in\ R} I_{N-1}[\vec{x}]\ I_N[\vec{x}+\vec{u}]}{\sum_{\vec{x}\ in\ R} I_N[\vec{x}+\vec{u}]\ I_N[\vec{x}+\vec{u}]}\,.$$

The integer optical flow of the region $\vec{u}_{Int}$ is taken to be the $\vec{u}$ that gives the minimum SSD given α in Eq. (2.1).

$$\vec{u}_{Int} = \min_{\vec{u}\ in\ \vec{u}_{\max}} SSD(\vec{u})$$

where $\vec{u}_{\max}$ is the maximal inter-frame motion in pixel, and we do this using search.

We compute subpixel flow of the region by first shifting I_{N-1} with the integer flow $\vec{u}_{Int}$ and then finding the subpixel displacement $\vec{u}_s$ that yields the minimum Sum of Squared Difference.

$$SSD(\vec{u}_s \mid \vec{u}_{Int}) = \min_{\vec{u}_s\ in\ W} \Big(\sum_{\vec{x}\ in\ R} (I_{N-1}[\vec{x}] - \alpha\ I_N[\vec{x}+\vec{u}_{Int}+\vec{u}_s])\Big)^2$$

We apply subpixel shifts $\vec{u}_s = (u_{s,x}, u_{s,y})$ in 9 directions, namely NW, N, NE, W, E, SW, S, SE, (0,0) and compute their SSDs. For example, in the NW direction, $u_{s,x} = -\,subpixel, u_{s,y} = -\,subpixel$. After that, we shift I_{N-1} by the subpixel flow that yields the minimal SSD using cubic interpolation. The above procedure is repeated for successively smaller amount of subpixel shifts. We found empirically the best sequence of values for *subpixel* is $0.75, 0.75^2, 0.75^3 \cdots$. Many other algorithms like variants of the gradient method, can do at least as good a job but this one gives us a bound on subpixel accuracy.

If the region for computing affine flow is too small, small linear deformation such as shrinkage or rotation would be indiscernible. Conversely, if the region for computing affine flow is too big, we have a higher possibility of inclusion of a motion boundary. Therefore, we compute affine flow in a region R_a that is larger than the initial seed region R and it is equal to the seed region R expanded several times (4 in our experiments) but excluding pixels that were not part of the same segment in previous frames to avoid inclusion of discontinuities. The affine flow $\vec{u}_a$ for a region R_a is defined by six parameters $u_x, u_y, v_x, v_y, u_0, v_0$

$$\vec{u}_a = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \vec{x} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}. \tag{2.2}$$

Plugging Eq. (2.2) into the optical flow equation and forming the SSD we get

$$\sum_{all\ \vec{x}\ in R_a} \left(\Delta\vec{I}[\vec{x}] + \nabla I[\vec{x}] \cdot \vec{u}_a \right)^2 \tag{2.3}$$

where

$$\Delta\vec{I}[\vec{x}] = I_{N-1}[\vec{x}] - \alpha\ I_N[\vec{x}+\vec{u}_{Int}+\vec{u}_s]$$

and $\nabla I[\vec{x}]$ is the gradient of the average of I_N and I_{N-1} [16]. We apply standard least squares to compute the six affine parameters by solving the normal equation from Eq. (2.3). We do least squares on the R_a instead of the seed region R alone because R_a encodes information from motion history of the tracked object. This way we can have a larger amount of data on which to do least squares without including pixels that are unlikely to belong to the object.

One of the hardest problems related to the differential technique is taking spatial and temporal derivatives. We did not have many difficulties in this case because we have already shifted the images so the residual flow is small. Then we apply the method to the segmented area of the image which is consistent with small flow and do least squares on a rather large area. The process can be repeated to increase the accuracy. In our experiments we repeated it twice.

2.3. Motion Segmentation

After obtaining the affine motion between the last two frames we can warp I_{N-1} towards I_N and then subtract them and square the difference. It can be seen that the parts of the image (presumably the tracked object) that move in a way consistent with the computed affine motion will have small squared difference and the rest of the image will have mostly large squared difference so a simple thresholding should be sufficient. Unfortunately, there are many reasons that this is not always true:

(1) There is a certain amount of noise in any image like digitization noise, random white noise, noise due to resampling or down-sampling (aliasing) etc. In our experiments these forms of noise had a combined standard deviation as high as 4.5, which is substantial.

(2) We approximate a rather complex optical flow with affine flow. While the approximation is quite good, the standard deviation of the error in flow is around 0.2 pixels, this can induce substantial error in the sum of squared difference and give rise to false negatives.

(3) When we warp image I_{N-1} using the affine flow we might align two totally different regions that happen to have identical texture or color. This can give rise to a large number of false positives.

(4) While we compensate for the change of illumination in the tracking and correct it, this change of illumination we compute is the "average" over the seed region and might not be appropriate for pixels outside the seed region. The situation is even worse when the change of illumination is not uniform (as when the object goes behind a shadow).

We develop two different statistics for the image model. A pixel statistic that measures how well each pixel in the image model $\hat{I}_N$ predicts the motion of the tracked object in the N^{th} frame. A patch statistic that measures how well a small image patch centered at each pixel predicts the motion of the tracked object in the N^{th} frame. The pixel statistic is simple and hence fast. The patch statistic is slower than the pixel statistic but can account for light illumination change across successive frames.

2.3.1. Pixel statistic

We compute the squared difference between pixels in the image model and an image frame aligned by the computed optical flow. Pixels following the motion of the tracked image model have small squared difference and hence can be identified by simple thresholding. However, the image model obtained from a pair of frame is not very reliable due to noise. We improve the accuracy of the image model by considering the trajectory of the tracked object over time. Consider the following summation,

$$\sum_{i=1}^{N-1} ({}^{N}I_i - I_N)^2 \tag{2.4}$$

where ${}^{N}I_i$ is the i^{th} image frame aligned with the N^{th} frame using the optical flow derived during tracking. The summation in Eq. (2.4) is the pixel-wise SSD between the last image I_N and the previous images ${}^{N}I_i$ properly aligned with the last image. All the pixels that are tracked correctly (under reasonable assumptions) should have the same intensity as in the last image so all their SSDs should be small. The pixels in areas that are not tracked correctly should have significantly higher SSD. Computing Eq. (2.4) directly is a time consuming task because we have to keep all the image frames and align all of them using the computed optical flows. Therefore, we need

a more efficient way of computing Eq. (2.4). Expanding it we notice,

$$\sum_{i=1}^{N-1} ({}^{N}I_i - I_N)^2 =$$

$$\sum_{i=1}^{N-1} {}^{N}I_i{}^2 - 2(\sum_{i=1}^{N-1} {}^{N}I_i)I_N + I_N^2(N-1) \tag{2.5}$$

Dividing equation (2.5) by $N-1$ gives:

$$\frac{\sum_{i=1}^{N-1} ({}^{N}I_i - I_N)^2}{N-1} = \tag{2.6}$$

$$\hat{I^2}_N - 2\,\hat{I}_N\,I_N + I_N^2 = t_{stat}$$

where $\hat{I}_N = \sum_{i=1}^{N-1} {}^{N}I_i \,/\, (N-1)$, $\hat{I^2}_N = \sum_{i=1}^{N-1} {}^{N}I_i{}^2 \,/\, (N-1)$ are first and second moments of the images aligned with the N^{th} image frame. In order to avoid any error in the optical flow computation haunting us forever, we let the influence of the past images decay exponentially in the computation of the moments. So

$$\hat{I}_N = h\,{}^{uv}\hat{I}_{N-1} + (1-h)\,I_N \tag{2.7}$$

$$\hat{I^2}_N = h\,{}^{uv}\hat{I^2}_{N-1} + (1-h)\,I_N^2$$

where h is the history coefficient, a decimal number between 0.0 and 1.0 arbitrating the relative weight between information from the previous model and the N^{th} frame, $\hat{I}_{N-1}$, ${}^{uv}\hat{I^2}_{N-1}$ are the first and second moments of the previous image model aligned to the N^{th} frame.

We use pixel-wise thresholding to identify pixels belonging to the tracked object. The pixels whose tracking statistics in Eq. (2.6) are less than their threshold values, are classified as belonging to the tracked object. Due to noise, even pixels that belong to the tracked object, might have the statistic given by Eq. (2.6) significantly different from zero. The noise can be attributed to two sources: Noise due to incorrect tracking which we call motion noise and noise due to the camera that we call camera noise. If the tracking statistic for a pixel is significantly higher than the noise then this pixel does not belong to the tracked object. We compute the value of the threshold (t_{value}) for the track statistics in Eq. (2.6) as follows:

$$t_{value} = (\sigma_c^2 + \sigma_f^2) \cdot z \tag{2.8}$$

where σ_c^2 denotes the variance of the camera noise, σ_f^2 gives the variance of the motion noise and z is a constant parameter that models the "significantly higher". In all our experiments, we set $z = 3.0$. In simple terms, the quantity t_{value} gives us an idea of the variance of the intensity of the aligned images. If the quantity from Eq. (2.6) is smaller than its threshold value from Eq. (2.8) then we have good reason to believe that the corresponding pixel belongs to the tracked object. If it is higher then most likely does not.

Motion noise models the noise resulting from incorrect computation of optical flow (u and v). The error can be due to insufficient search (for example, we searched up to 0.75^2 pixels only) or due to insufficient modeling (for example, underlying model is more complex than affine). According to the optical flow equation:

$$I_{N-1,\,x} \cdot \Delta u + I_{N-1,\,y} \cdot \Delta v + I_{N-1,\,t} = 0$$

where $I_{N-1,\,x}$ and $I_{N-1,\,y}$ are the x and y derivatives of I_{N-1} and $I_{N-1,\,t} = I_{N-1} - I_N$. Hence, the variance of the intensity difference becomes:

$$\begin{aligned} Var(I_{N-1,\,t}) &= Var(I_{N-1,\,x} \cdot \Delta u + I_{N-1,\,y} \cdot \Delta v) \\ &= (I_{N-1,\,x}^2 + I_{N-1,\,y}^2) \cdot \sigma_{uv}^2 \end{aligned} \tag{2.9}$$

where σ_{uv}^2 is the variance of Δu and Δv. We assume that the errors Δu and Δv are independent, which is quite reasonable because this is the error in the model and not the uncertainty in the optical flow that is usually very unisotropic due to the aperture problem. Therefore,

$$\sigma_f^2 = (I_{N-1,\,x}^2 + I_{N-1,\,y}^2) \cdot \sigma_{uv}^2$$

The camera noise models the white Gaussian noise generated by electronic circuit of a camera or by other means not directly related to motion. We assumed random noise only, which we calculated empirically to be of zero mean and of variance σ_c^2 equal to 1 for our 8 bit webcam quality camera.

Intuitively, as the tracking algorithm runs for more frames, the moving object being tracked should be kept aligned by the optical flow computed in each successive pair of frames. So, we classify those pixels whose tracking statistics t_{stat} (from Eq. (2.6)) are less than their corresponding threshold values t_{value} (from Eq. (2.8)) as belonging to the tracked object and vice versa.

$$^{(N)}I_{track} = \begin{cases} 1 & \text{if } t_{stat} \le t_{value} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

2.3.2. Patch statistic

The drawback of the pixel-wise squared difference is that it implicitly assumes noise in neighboring pixels is independent which is often not true. However, the pixel-wise statistic is sufficient in many situation where the correlation between neighboring pixels is small and hence the noise can be considered safely as independent. One situation where the independence assumption is inadequate is when the lighting conditions change from frame to frame. Then the noise in neighboring pixels becomes correlated assuming the pixels within a patch are under the same lighting conditions. Another situation where noise is correlated is when the tracking is uniformly inaccurate such that pixels within the same patch drift by the same amount. In what follows, we describe a model that addresses both situations.

So we identify pixels that follow the same motion model as the seed region by evaluating a statistic on ΔI the image difference between the N^{th} image frame and an image model of the tracked region $\hat{I}$ aligned by the optical flow by examining small $k \times k$ image patches. The image model is the same as in Eq. (2.7). We rearrange the pixels of each image patch into a vector and we take the difference between corresponding vector patches in the image model and the current image

$$\Delta I = {}^{uv}\vec{\hat{I}}_{N-1} - \vec{I}_N \tag{2.11}$$

where the left superscript ^{uv} denotes warping by uv, ${}^{uv}\vec{\hat{I}}_{N-1}$ is a vector formed from a patch of the aligned image model $\hat{I}_{N-1}$ and $\vec{I}_N$ is a vector from a patch extracted from image frame I_N. If the tracking was perfect, the noise was absent and the illumination did not change, ΔI would be zero. But instead it is

$$\begin{aligned} \Delta I = \Delta\vec{n} + Diag({}^{uv}\hat{I}_{N-1,\,x})\,\Delta\vec{u}_\alpha + Diag({}^{uv}\hat{I}_{N-1,\,y})\,\Delta\vec{v}_\alpha + {}^{uv}\hat{I}_{N-1,\,x}\,\Delta u + \\ {}^{uv}\hat{I}_{N-1,\,y}\,\Delta v + {}^{uv}\vec{\hat{I}}_{N-1}\,\Delta l + \Delta f \end{aligned} \tag{2.12}$$

where $\Delta\vec{n}$ is the pixel-wise noise, a k^2 vector of random independent variables, $Diag({}^{uv}\hat{I}_{N-1,\,x})$, $Diag({}^{uv}\hat{I}_{N-1,\,y})$ are diagonal matrices whose diagonals contain the elements of the vector ${}^{uv}\hat{I}_{N-1,\,x}$, ${}^{uv}\hat{I}_{N-1,\,y}$ respectively. ${}^{uv}\hat{I}_{N-1,\,x}$ and ${}^{uv}\hat{I}_{N-1,\,y}$ are vectors from a patch of the x and y derivatives of ${}^{uv}\vec{\hat{I}}_{N-1}$ and $\Delta\vec{u}_\alpha$, $\Delta\vec{v}_\alpha$ are the random vectors denoting the pixel-wise error in flow in the image frames. Up till now, the random vectors are pixel-wise measures and are of nature similar to the camera and motion noise in the pixel statistic. Next, we are going to describe the random variables defined for patches. Δu, Δv are random variables representing the residual flow in patches Δl, Δf are random variables that model respectively the percentage and absolute change of light intensity [25] over patches between the image frames I_{N-1} and I_N.

In other words, we model seven kinds of noise in this image patch. A white noise $\Delta\vec{n}$ that is attributed to digitization and electronic noise in camera and anything that can be considered independent and identically

distributed (i.i.d). This is the same as the camera noise in the pixel case. Two components of pixel-wise motion noise $\Delta\vec{u}_\alpha$, $\Delta\vec{v}_\alpha$ that come from either independent motion of image details within an image patch (for example, the motion of individual blowing leaves of a tree) or more commonly the result of aliasing in images. Again this is similar to the motion noise in the pixel case but does not include the motion noise that is common to the whole patch and in our experience is dominated by aliasing noise which may be due to resampling of the images or warping and we found empirically to be proportional to the derivatives of the images. On the other hand, we also model two components of patch-wise motion noise Δu, Δv that arises from the error in the tracking within an image patch which can be attributed to the fact that real-world objects only moves with an approximately affine motion model. Also, there are two patch-wise noise components Δl, Δf that come from the change of illumination that we model as affine [25]. Of the seven random variables, $\Delta\vec{n}$, $\Delta\vec{u}_\alpha$, $\Delta\vec{v}_\alpha$ are vectors, the rest being scalar.

Since we represent the image patches as vectors of length k^2, Eq. (2.12) can be rewritten in vector and matrix notation:

$$\Delta I = \vec{n} + \mathbf{U}\,\vec{u} \tag{2.13}$$

where $\vec{n}$ is the sum of the camera and pixel-wise motion noise $(\Delta\vec{n} + Diag({}^{uv}\hat{I}_{N-1,\,x})\,\Delta\vec{u}_\alpha + Diag({}^{uv}\hat{I}_{N-1,\,y})\,\Delta\vec{v}_\alpha)$, $\mathbf{U}$ is a $k^2 \times 4$ matrix whose columns store the three k^2 vectors ${}^{uv}\hat{I}_{N-1,\,x}$, ${}^{uv}\hat{I}_{N-1,\,y}$, ${}^{uv}\vec{\hat{I}}_{N-1}$, from Eq. (2.12) and a column of 1s and $\vec{u}$ is $[\Delta u\ \Delta v\ \Delta l\ \Delta f]^T$ which we call motion noise. Eq. (2.13) defines a noise model for any image patch. This model is accurate for patches that are on the tracked object, but very inaccurate if the patch is not on the object. So we perform a χ^2 test for every patch in the image to decide which pixels belong to the object.

One issue we have to resolve before the evaluation of the χ^2 statistic, is how to estimate the statistical parameters of the seven random variables we introduced in the noise model in Eq. (2.12). We will describe how we estimate these parameters at the end of this section. Another issue is the high dimensionality of the image patch vector which makes the computation and inversion of the covariance matrix $\mathbf{C}_{\Delta\mathbf{I}}$ of ΔI very time consuming. ΔI is a vector with k^2 elements so its covariance matrix has k^4 elements and needs about $k^{(2)\,3} = k^6$ operations to be inverted. This is prohibitive for even small k, because we have to do it on every pixel. In the next paragraphs we show how to reduce the cost by several orders of magnitude.

The covariance $\mathbf{C}_{\Delta\mathbf{I}}$ of the ΔI for each successive pair of frames is

$$\mathbf{C}_{\Delta\mathbf{I}} = Cov(\vec{n} + \mathbf{U}\cdot\vec{u})$$

Assuming independence of camera noise $\vec{n}$ and the motion noise $\vec{u}$,

$$= Cov(\vec{n}) + Cov(\mathbf{U}\cdot\vec{u})$$

$$= Cov(\vec{n}) + \mathbf{U}\,Cov(\vec{u})\,\mathbf{U}^T$$

$$= \mathbf{C}_\mathbf{n} + \mathbf{U}\,\mathbf{C}_\mathbf{u}\,\mathbf{U}^T$$

$\mathbf{C}_\mathbf{n} = Cov(\vec{n})$ is a $k^2 \times k^2$ diagonal matrix with each diagonal entry equal to the corresponding element from the vector $\vec{1}_{k^2}\sigma_n^2 + {}^{uv}\vec{\hat{I}}_{N-1,\,d}\,\sigma_a^2$, where $\vec{1}_k^2$ is a k^2 vector of 1s, ${}^{uv}\vec{\hat{I}}_{N-1,\,d}$ is a vector from a patch of the sum of squares of the x and y derivatives of ${}^{uv}\vec{\hat{I}}_{N-1}$ and σ_n^2, σ_a^2 are scalar constants representing the variances of the camera and aliasing noise.

$$\mathbf{C}_\mathbf{u} = Cov(\vec{u}) = \begin{bmatrix} \sigma_u^2 & 0 & 0 & 0 \\ 0 & \sigma_v^2 & 0 & 0 \\ 0 & 0 & \sigma_l^2 & 0 \\ 0 & 0 & 0 & \sigma_f^2 \end{bmatrix}$$

where the scalars σ_u^2, σ_v^2, $\sigma_l^{2,}$ σ_f^2 denote the variance of random variables Δu, Δv, Δl, Δf. If we had a different model for the motion noise then $\mathbf{C}_\mathbf{u}$ might not be diagonal but this would have only a minimal effect in the total cost of computation.

The ΔI for pixels undergoing small displacement with light intensity change across successive frames as described by Eq. (2.11), should be small. We could use a Weighted Sum of Squared Differences, which properly done is a straight χ^2 statistic, but this does not take into account the interdependence between the pixels in the same patch. So we have to identify those pixels whose motion follows the computed flow model by computing their Mahalanobis distance [11, 23] which is defined as

$$D_m^2 = ({}^{uv}\vec{\hat{I}}_{N-1} - \vec{I}_N)^T \; \mathbf{C}_{\Delta\mathbf{I}}{}^{-1} \; ({}^{uv}\vec{\hat{I}}_{N-1} - \vec{I}_N) \tag{2.14}$$

where $\mathbf{C}_{\Delta\mathbf{I}} = \mathbf{C}_\mathbf{n} + \mathbf{U}\,\mathbf{C}_\mathbf{u}\,\mathbf{U}^T$. The Mahalanobis distance takes into account of the inter-dependence between the parameters and standardizes each parameter to unit variance by using the covariance matrix. For an image patch of $k \times k$ pixels, using any standard matrix inversion algorithm takes $O((k^2)^3) = O(k^6)$ to invert $\mathbf{C}_{\Delta\mathbf{I}}$. The Sherman-Morrison-Woodbury (SMW) identity, a little known numerical analysis tool becomes extremely useful in our problem [13, 27, 34]. It can be used when we have a low rank update on a matrix that we can easily invert. In this problem matrix $\mathbf{C}_\mathbf{n}$ is a diagonal matrix and $\mathbf{U}\,\mathbf{C}_\mathbf{u}\,\mathbf{U}^T$ is a rank-4 update to $\mathbf{C}_\mathbf{n}$. Applying the SMW we get

$$\mathbf{C}_{\Delta\mathbf{I}}{}^{-1} = \mathbf{C}_\mathbf{n}{}^{-1} - \mathbf{C}_\mathbf{n}{}^{-1}\mathbf{U}(\mathbf{C}_\mathbf{u}{}^{-1} + \mathbf{U}^T\mathbf{C}_\mathbf{n}{}^{-1}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{C}_\mathbf{n}{}^{-1} \tag{2.15}$$

We note that $\mathbf{C}_\mathbf{s} = (\mathbf{C}_\mathbf{u}{}^{-1} + \mathbf{U}^T\mathbf{C}_\mathbf{n}{}^{-1}\mathbf{U})$ is a 4×4 matrix, hence the time required for its inversion is just $O(4^3)$, a constant, and the cost of computing the inverse $\mathbf{C}_\mathbf{n}{}^{-1}$ of the diagonal matrix $\mathbf{C}_\mathbf{n}$ is $O(k^2)$ which as we will see can be amortized over neighboring patches. If we use simpler notation for the columns of the matrix $\mathbf{U}$ and denote them by the vectors $\vec{I}_x$, $\vec{I}_y$, $\vec{I}$, $\vec{1}$ then $\mathbf{C}_\mathbf{s}$ can be written as:

$$\mathbf{C}_\mathbf{s} = \begin{bmatrix} \frac{1}{\sigma_u^2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_v^2} & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_l^2} & 0 \\ 0 & 0 & 0 & \frac{1}{\sigma_f^2} \end{bmatrix}$$

$$+ \begin{bmatrix} \vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_x & \vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_y & \vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I} & \sum \vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1} \\ \vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_y & \vec{I}_y\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_y & \vec{I}_y\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I} & \sum \vec{I}_y\,\mathbf{C}_\mathbf{n}{}^{-1} \\ \vec{I}\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_x & \vec{I}\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_y & \vec{I}\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I} & \sum \vec{I}\,\mathbf{C}_\mathbf{n}{}^{-1} \\ \sum \vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1} & \sum \vec{I}_y\,\mathbf{C}_\mathbf{n}{}^{-1} & \sum \vec{I}\,\mathbf{C}_\mathbf{n}{}^{-1} & \sum \mathbf{C}_\mathbf{n}{}^{-1} \end{bmatrix}$$

For each pixel, we need to evaluate product such as $\vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_x$ or $\vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_y$. Direct evaluation of these products is quite expensive even for small k because it requires $2\,k^2$ multiplications and $k^2 - 1$ additions per patch but we can reduce the cost by reusing the computation from adjacent patches. We demonstrate the procedure using $\vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_y$ as example. We store the x, y derivatives of I in images I_x, I_y. respectively.

In the following, we show how to compute the product of $\vec{I}_x\,\mathbf{C}_\mathbf{n}{}^{-1}\,\vec{I}_y$ in more detail. We assume the patches are centered at a point $[x, y]$ and each patch is of size k which is usually an odd number. Since $\mathbf{C}_\mathbf{n}$ is diagonal

$\mathbf{C}_\mathbf{n}[i, j] = 0$ if $i \neq j$

and

$$\mathbf{C}_\mathbf{n}[i, i] = \sigma_n^2 +$$

$$\sigma_a^2\left(I_x{}^2\left[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\right] + I_y{}^2\left[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\right]\right)$$

assuming the patch is arranged on the diagonal in row-major order. If we set the image I_n

$$I_n = \sigma_n^2 + \sigma_a^2(I_x{}^2 + I_y{}^2)$$

where $I_x{}^2$, $I_y{}^2$ are the pixel-wise squares of the x, y derivatives of the image I, then

$$\mathbf{C_n}[i,i] = I_n\Big[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\Big]$$

and

$$\mathbf{C_n}^{-1}[i,i] = \frac{1}{I_n\Big[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\Big]}.$$

The vectors $\vec{I}_x$ and $\vec{I}_y$

$$\vec{I}_x[i] = I_x\Big[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\Big]$$

$$\vec{I}_y[i] = I_y\Big[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\Big]$$

are patches of the images I_x, I_y respectively, again in row-major order. Then

$$\vec{I}_x \mathbf{C_n}^{-1} \vec{I}_y = \sum_i \sum_j I_x[i]\ \mathbf{C_n}^{-1}[i,j]\ I_y[j] = \sum_i \vec{I}_x[i]\ \mathbf{C_n}^{-1}[i,i]\ \vec{I}_y[i]$$

$$= \sum_{i=1}^{k^2} I_x\Big[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\Big]\ I_n{}^{-1}\Big[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\Big]$$
$$I_y\Big[y - \lfloor k/2 \rfloor + \lfloor i/k \rfloor, x - \lfloor k/2 \rfloor + i\%k\Big]$$

$$= \sum_{m=0}^{k} \sum_{n=0}^{k} I_x\Big[y_o - m, x_o - n\Big]\ I_n{}^{-1}\Big[y_o - m, x_o - n\Big]\ I_y\Big[y_o - m, x_o - n\Big]$$

where $m = \lfloor k/2 \rfloor - \lfloor i/k \rfloor$, $n = \lfloor k/2 \rfloor - i\%k$. Therefore,

$$\vec{I}_x\ \mathbf{C_n}^{-1}\ \vec{I}_y = \sum_{m=0}^{k} \sum_{n=0}^{k} I_x\Big[y_o - m, x_o - n\Big]\ I_n{}^{-1}\Big[y_o - m, x_o - n\Big]\ I_y\Big[y_o - m, x_o - n\Big] = \sum_{m=0}^{k} \sum_{n=0}^{k} I_{xny}\Big[y_o - m, x_o - n\Big]$$

where $I_{xny} = \dfrac{I_x\ I_y}{I_n}$. So the whole computation involves the pixel-wise multiplication of the images I_x, I_y and $I_n{}^{-1}$ and convolution with a $k \times k$ uniform kernel. The result is an image every pixel of which is the $\vec{I}_x\ \mathbf{C_n}^{-1}\ \vec{I}_y$ for the patch centered on the pixel. The uniform convolution kernel is separable and if we implement it as a running sum it requires only five operations per patch: two additions and two subtractions and one multiplication. The same can be done with the other dot products and calculate the elements of matrix $\mathbf{C_s}$. Since we use multiple overlapping patches centered at every pixel of the image, we have as many 4×4 matrices as pixels and we perform a constant amount of computation per pixel independent of patch size.

From Eq. (2.14) and Eq. (2.15), we have:

$$D_m^2 = (\Delta I)^T (\mathbf{C_n}^{-1} - \mathbf{C_n}^{-1}\ \mathbf{U}\ \mathbf{C_s}^{-1}\ \mathbf{U}^T\ \mathbf{C_n}^{-1})(\Delta I)$$

We apply Cholesky factorization and get $\mathbf{C_s}^{-1} = \mathbf{L}^{-T}\ \mathbf{L}^{-1}$ where $\mathbf{L}$ is a 4×4 upper triangular matrix which is easy to invert so

$$D_m^2 = \Delta I^T\ \mathbf{C_n}^{-1}\ \Delta I - (\Delta I^T \mathbf{C_n}^{-1}\ \mathbf{U}\ \mathbf{L}^{-T})\ (\mathbf{L}^{-1}\ \mathbf{U}^T\ \mathbf{C_n}^{-1} \Delta I)$$

$$= \Delta I\ \mathbf{C_n}^{-1}\ \Delta I - \|\ \mathbf{L}^{-1}\ \mathbf{U}^T\ \mathbf{C_n}^{-1}\ \Delta I\ \|^2 \qquad (2.16)$$

The product $\mathbf{U}^T\ \mathbf{C_n}^{-1}\ \Delta I$ results in a vector of length 4

$$\begin{bmatrix} \vec{I}_x \\ \vec{I}_y \\ \vec{I} \\ \vec{1} \end{bmatrix} \mathbf{C_n}^{-1}\,\Delta I = \begin{bmatrix} \vec{I}_x\,\mathbf{C_n}^{-1}\,\Delta I \\ \vec{I}_y\,\mathbf{C_n}^{-1}\,\Delta I \\ \vec{I}\,\mathbf{C_n}^{-1}\,\Delta I \\ \mathbf{C_n}^{-1}\,\Delta I \end{bmatrix}$$

and we use the same technique to reduce it to 4 convolutions. Hence, following Eq. (2.16), the computation of the inverse of the covariance matrix needs a constant number of operations per pixel, independent of k. The cost could increase if we used a motion noise model that has more than four components. The main change in the cost is associated with the construction and inversion of matrices $\mathbf{C_s}$ and U.

The Mahalanobis distance should be small for pixels that are following the motion of the seed region but we have to define how small is small and we do it in a probabilistic way. The sum of the squares of k^2 independent and identically distributed normal variables $N(0,1)$ is a Chi-Square distribution with mean k^2 and variance $2k^2$ [29]. The Mahalanobis distance can be thought of (after proper change of coordinates) as a sum of the squares of independent normal variables. Therefore, the Mahalanobis distance between two $k \times k$ image patches follows a Chi-Square distribution with k^2 degrees of freedom, mean k^2 and variance $2\,k^2$ for appropriately estimated parameters of σ_u^2, σ_v^2, σ_l^2, σ_f^2, σ_n^2, σ_a^2.

The Mahalanobis distance is a good measure of the consistency of the tracking but its variance can be higher than expected if the image noise is not Gaussian and in most real images it is only approximately Gaussian. We can reduce the effect of non-Gaussianity considerably by performing temporal smoothing of Mahalanobis

$$d_{t,N} = h\ {}^{N}d_{t,N-1} + (1-h)\ D_m^2$$

where ${}^{N}d_{t,N-1}$ is the previous smoothed Mahalanobis distance aligned with the current N^{th} frame, D_m^2 is the Mahalanobis distance from the current (N^{th}, $N-1^{th}$) frames and h is a decimal coefficient between 0.0 and 1.0 arbitrating the relative importance of history information. As more frames are temporally integrated, $d_{t,N}$ becomes more Gaussian-like according to the Central Limit Theorem. Therefore, the segmentation result improves as information from more frames is temporally integrated.

So, we can segment out pixels that move consistently with the seed region by thresholding with $d_{t,N}$ to obtain the binary image ${}^{(N)}I_{track}$ needed in

$${}^{(N)}I_{track} = \begin{cases} 1 & \text{if } d_{t,N} \le c_{value} \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

The value of c_{value} can be taken from a Chi-Square table [29]. For example, for $k^2 = 25$, the degree of freedom is 25 and using the level of confidence 0.995, then $t_{stat} = 46.9$. We prefer higher levels of confidence to reduce the number of false negatives.

2.4. Estimation of the Model Parameters

We estimate the model parameters by using a Maximum Likelihood Estimator (MLE). We select manually a few seed regions ($\kappa = 5$ was enough for all our experiments) that fall within the projection of an independently moving object. For each selected seed region, we compute its uniform integer flow using straight search. For properly aligned patches around these seed regions, we assume that they follow a multidimensional Gaussian distribution of mean equal to 0 and covariance equal to $\mathbf{C}_k$

$$N(\Delta I; 0, \mathbf{C}_k) = \frac{1}{(2\Pi)^{k/2}|\mathbf{C}_k|^{1/2}}\, e^{\frac{-\Delta I^T \mathbf{C}_k^{-1} \Delta I}{2}}.$$

The covariance matrix $\mathbf{C}_k$ is a function of the model parameters:

$$\mathbf{C}_k = C_k(\sigma_n^2, \sigma_a^2, \sigma_u^2, \sigma_v^2, \sigma_l^2, \sigma_f^2) = \sigma_n^2\mathbf{1} + \sigma_a^2\mathbf{D}_k + \mathbf{U}_k\ \mathbf{C}_u\ \mathbf{U}_k{}^T$$

where $\mathbf{1}$ is the identity matrix, $\mathbf{D}_k = Diag({}^{uv}\vec{\hat{I}}_{N-1,\,d})$ is a $k^2 \times k^2$ diagonal matrix with each diagonal element equal to the corresponding element from the vector ${}^{uv}\vec{\hat{I}}_{N-1,\,d}$ which is formed by a patch equal to $\vec{I}_x{}^2 + \vec{I}_y{}^2$ and

$\mathbf{U}_k = [\vec{I}_x \ \vec{I}_y \ \vec{I} \ \vec{1}]$. The likelihood L is a compound function $L(C_k(\sigma_n^2, \sigma_a^2, \sigma_u^2, \sigma_v^2, \sigma_l^2, \sigma_f^2))$. We compute the maximum likelihood estimator of these parameters using Davidon-Fletcher-Powell (DFP) [26] algorithm given an initial guess of the value of these variances. DFP is a variable metric method for finding the minimum of the negative of the log likelihood and we use the following derivative formulae:

$$\frac{\partial L}{\partial \sigma_n^2} = \frac{1}{2} \sum_{all\ \kappa} tr(\mathbf{C}_k{}^{-1}) - (\mathbf{C}_k{}^{-1} \Delta I_k)^2$$

$$\frac{\partial L}{\partial \sigma_a^2} = \frac{1}{2} \sum_{all\ \kappa} tr(\mathbf{C}_k{}^{-1} \mathbf{D}_k - \mathbf{C}_k{}^{-1} \Delta I_k \Delta I_k^T \mathbf{C}_k{}^{-1} \mathbf{D}_k)$$

For $\sigma_u^2 = \mathbf{C}_u[1,1]$, $\sigma_v^2 = \mathbf{C}_u[2,2]$, $\sigma_l^2 = \mathbf{C}_u[3,3]$, $\sigma_f^2 = \mathbf{C}_u[4,4]$,

$$\frac{\partial L}{\partial \mathbf{C}_u[i,i]} = \frac{1}{2} \sum_{all\ \kappa} tr(\mathbf{U}_k{}^T[*,i] \mathbf{C}_k{}^{-1} \mathbf{U}_k[*,i]) - (\mathbf{U}_k{}^T[*,i] \mathbf{C}_k{}^{-1} \Delta I_k)^2$$

where $\mathbf{U}_k[*,i]$ is the ith column of $\mathbf{U}_k$. One of the reasons that we prefer the analytical computation of the derivatives is that we have higher accuracy and do not need to worry about the step for the numerical derivatives.

Estimating the parameters is extremely important because it removes one degree of uncertainty from the design and evaluation of the algorithm. In the experimental section we perform experiments using both the MLE estimate we just described and a set of empirically derived parameters that were common to all experiments. The empirically derived parameters worked well for most image sequences in our experiments that included experiments with different cameras, different lighting, etc. The MLE estimates worked very well for all sequences.

2.5. Postprocessing

We apply postprocessing to ${}^{(N)}I_{track}$ in Eq. (2.10) or Eq. (2.17) to reduce the number of incorrectly identified moving regions. Such false positives are usually very small, disconnected, have little or no texture or even not moving. Any form of flow computation is very problematic in such regions and we perform postprocessing to alleviate noise in these regions.

In the following three figures, we show the input image frames, the tracked regions before and after this postprocessing step.

Figure 2.1: Input image frames

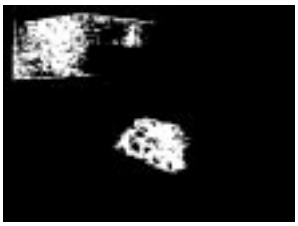


Figure 2.2: Before postprocessing

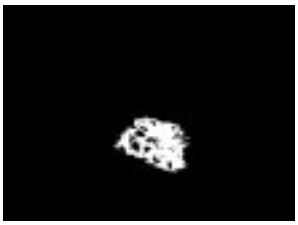


Figure 2.3: After postprocessing

Postprocessing involves two major steps. First, we compute pixels that are moving using image frame difference and store the result in $\Delta\,\mathrm{Im}$. Second, we extend the conjunction ($\Delta\,\mathrm{Im}$ AND ${}^{(N)}I_{track}$) by adding to it those positive pixels in ${}^{(N)}I_{track}$ that are in the direct connected neighborhood of the conjunction.

2.5.1. Finding the moving regions

For each successive pair of image frames (lets say Im_{N-1} and Im_N), we compute the image difference as follows:

$$ImDiff = \mathrm{Im}_N - \mathrm{Im}_{N-1} \tag{2.18}$$

If there is no motion of any kind in successive frames, then *ImDiff* should be bounded by a multiple of the camera white noise. The mean of the noise distribution is 0.0 and the variance equals to σ_c^2. From the theory of hypothesis testing in statistics [21], a sample from a normal distribution $N(\mu, \sigma^2)$ has a probability of 0.9978 belonging in the range $\mu - 3\sigma$ to $\mu + 3\sigma$.

$$\Delta\,\mathrm{Im} = \begin{cases} 1 & \text{if } ImDiff < -3\sigma_c \text{ or } ImDiff > 3\sigma_c \\ 0 & \text{otherwise} \end{cases}$$

After that, we perform size filtering on $\Delta\,\mathrm{Im}$. We keep only those blobs of connected regions in $\Delta\,\mathrm{Im}$ of size bigger than a threshold T_s. We set the size $T_s = 20$ pixels in our experiments.

2.5.2. Region growing

$\Delta\,\mathrm{Im}$ contains patches for all independently moving objects in a pair of successive image frames. Since we are interested in tracking the motion of the moving object whose projection contains the seed region, we need to eliminate all moving patches unrelated to the tracked object. We perform this motion filtering by a bitwise conjunction between $\Delta\,\mathrm{Im}$ and ${}^{(N)}I_{track}$ and save the result as ${}^{(N)}I_{filter}$.

${}^{(N)}I_{filter} = \Delta\,\mathrm{Im}$ and ${}^{(N)}I_{track}$

One drawback of image difference technique in the detection of moving objects is that it can only capture moving regions with large image frame difference. However, a region can have a small *ImDiff* even if it is the projection of a moving object due to the aperture problem [15] or flat intensity. We address this shortcoming by extending ${}^{(N)}I_{filter}$ with ${}^{(N)}I_{track}$ using region growing and use the merged image ${}^{(N)}I_{merge}$ as the output of our tracking system.

if $({}^{(N)}I_{filter}(x, y) \equiv 1$ or

${}^{(N)}I_{filter}(x, y) \equiv 0$ and ${}^{(N)}I_{track}(x_n, y_n) \equiv 1)$

then ${}^{(N)}I_{merge}(x, y) = 1$

else ${}^{(N)}I_{merge}(x, y) = 0$

where (x_n, y_n) is any of the four direct neighbors (NW, N, NE, W) of (x, y).

2.6. Optical Flow

One application and test-bed of our motion segmentation and tracking algorithm is the computation of optical flow. Two difficult issues in optical flow are motion boundaries and large inter-frame motion. By warping the

image using the affine flow derived from our tracking algorithm and computing optical flow within a segment, we avoid the motion discontinuities. Although big inter-frame motion could be handled by following a hierarchical scheme [1], optical flow computation in an image pyramid has strengths and limitations that make it complementary to our tracking and segmentation method. For example, a finely textured object which is conspicuous at full image resolution might be indiscernible at a lower image resolution. A more serious issue is that a hierarchical approach would not work well for tracking objects that are of size comparable to the inter-frame motion. Consider for instance an object that is about 60 pixels in each dimension and moves by about 10 pixels per frame. If we use a three or four level pyramid to reduce the flow to about one pixel the size of the object will be 4 or 8 pixels in each dimension which is comparable to the size of the Lucas and Kanade [22] regions. Such an object might be missed by a hierarchical scheme. Luckily the hierarchical scheme can be relatively easily incorporated into our method if one wants to take advantage of its proven record.

To compute optical flow, we warp I_{N-1} with the computed affine motion parameters to bring it close to I_N and then apply the standard Lucas and Kanade algorithm [22] with temporal support of two frames to compute the residual flow. After that we combine residual flow with the affine flow to compute the total flow. The Lucas and Kanade algorithm works really well in this situation because the residual flow is small (around 1 pixel per frame) and there is no discernible motion boundary within the region. We report one experiment on a synthetic image using this technique.

3. Experiments

In this section, we present the results of running the segmentation and tracking algorithm on several real and synthetic image sequences, some with moving background. We show the results of segmentation using both the pixel and patch statistic.

In some experiments we generate several random seed regions, and then we track and segment the objects around the seed regions. We highlight the seed regions in the output segments. During tracking, if the segment obtained from tracking of the seed region containing the feature point is inappropriate, the algorithm tries to continue the tracking/segmentation using another seed region derived from other feature points that fall on the original segment. Depending on the number of seed regions that we maintain in the sequence and the complexity of the scene we can cover substantial portion of the image. We show a few input frames, one from the beginning, one from the middle and one from the end of the sequence, then we show one or more segments from the same frames.

In other experiments we manually select the seed regions for tracking and segmentation. If the seed region does not span a motion boundary then the tracking was very accurate for all the sequences we tried with 25 to 115 frames. We need the experiments with manually selected seed regions to perform empirical analysis of the algorithm decoupled from the seed region selection method.

3.1. Pixel statistic

In the first experiment, we show the result of running our algorithm on the “zoo animal” sequence. The animals are on a large piece of paper, the camera is stationary and the paper is moved by hand. Some of the animals shake. We show the result of two segments generated from randomly selected corner features: One segment corresponds to the moving hippopotamus and the other shows the moving paper/background. The seed regions used for tracking are relocated by the algorithm in later frames in order to track the motion of the segment as long as possible.

Frames 5, 15, 27 of the zoo animal sequence.

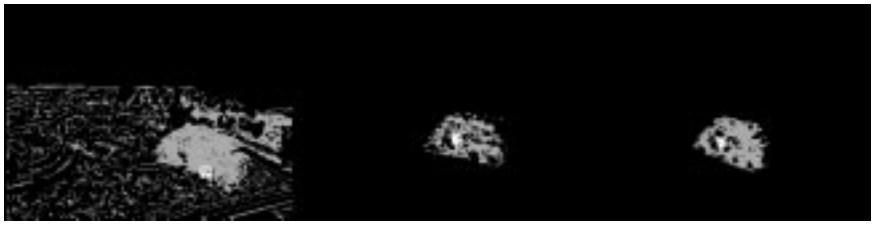
Segment corresponding to the hippopotamus.

Segment corresponding to the background.

Figure 3.1: The toy animals sequence.

In the second experiment, we show the result of running our algorithm on the truck image sequence which shows a turning toy truck in a moving background. The image sequence is taken by a hand-held camera that attempts to follow the motion of the toy truck. We show two segments output by our algorithm corresponding to the turning truck and moving background. The seed regions used for tracking are generated randomly. For the truck segment, the seed region used in the first frame is replaced by another seed region within the same segment in later frames because the original seed region does not satisfy the criteria for tracking in later frame. We show the residual horizontal, vertical flow as a gray scale image. The residual flow computed is small and never exceeded 2 pixels, and the mean square average is rarely above 0.5 pixels. We do not use needlemaps to display flow because they provide rather little accuracy for so small regions. The reason is simple. A typical segment is 50-80 pixels in each dimension and the needles in the needlemap cannot be more than 3-4 pixels, which provide little resolution. After that, we show stabilized versions of the image, where the object that is associated with a segment appears stationary. We superimpose a reference grid on the stabilized image. We measured the drift on the stabilized image with the mouse and it was less than 1 pixel in 28 frames in the center of the initial seed region. The purpose of showing the stabilized image with a reference grid superimposed is to quantify the accuracy of the tracking.

Frames 1, 15, 28 of the turning truck sequence.



Segment corresponding to the turning truck with seed region highlighted.

Residual horizontal flow as gray scale image.

Residual vertical flow as gray scale image.

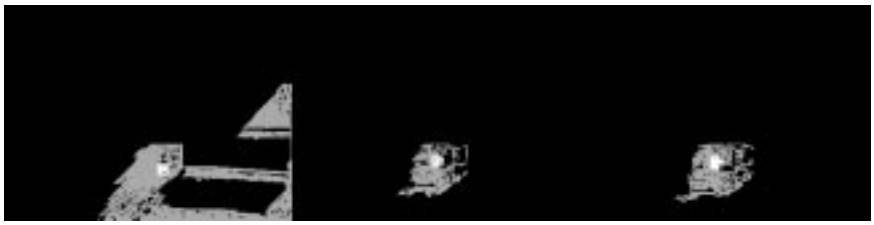
Motion stabilized frames with grid.

Segment corresponding to the moving background.

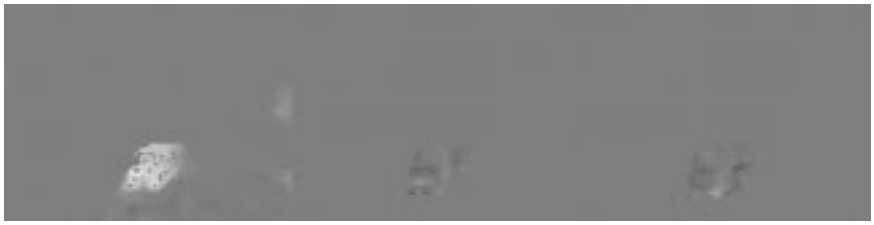
Figure 3.2: The input image frames show a toy truck with a hippopotamus on it taken from a hand-held camera that attempts to follow the toy truck.

In the third experiment, we show the result of running our algorithm on the approaching train sequence. The toy train is moving towards the stationary camera. We show that our algorithm is able to find a seed region that segments out the approaching train. The seed region is relocated in later frames of the sequence. In addition, we show the residual flow as a gray scale images.

Image sequence at frame 1, 14, 26.



Segment corresponding to the train.



Residual horizontal flow as gray scale image.

Residual vertical flow as gray scale image.

Figure 3.3: The input image frames show a toy train moving towards the camera.

In the fourth experiment, we show the result of running our algorithm on the Hamburg taxi sequence. We show the segment corresponding to the turning taxi from a randomly generated seed region.

Input sequence at frames 0, 9, 18.

Segment corresponding to the turning taxi.

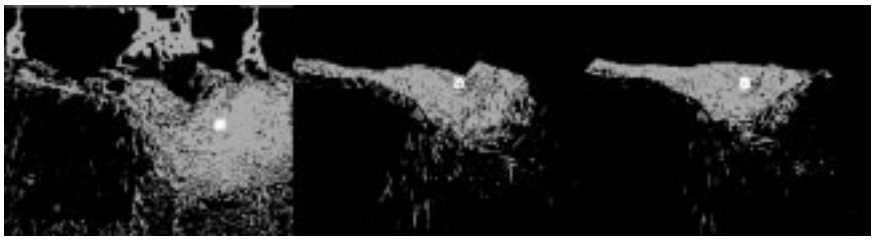
Figure 3.4: The input image frames show the Hamburg taxi sequence.

In the fifth experiment, we show the result of running our algorithm on the Yosemite sequence which is a synthetic fly through sequence from the Yosemite valley by Lynn Quann at SRI. We show various segments output by our algorithm from randomly selected seed regions. We compute the optical flow on these segments and show the mean square flow error and the mean angular error [2] for one of the segments between the optical flow computed by our algorithm and the ground truth of the Yosemite sequence. For comparison, we also show the mean

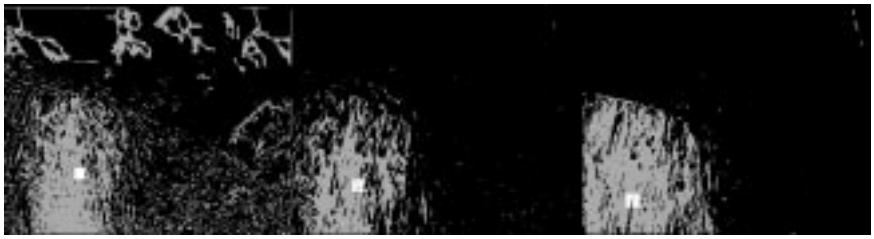
square error and the mean angular error between the optical flow computed using the pure Lucas and Kanade [22] without segmentation and alignment method and the ground truth. The error on both algorithms is computed on the same region and both use the same parameters (same derivatives, etc) for the Lucas and Kanade algorithm.

The Lucas and Kanade algorithm does not produce good results on this sequence because the images have areas of rather little texture and have large inter-frame motion. Our algorithm is not affected by the large inter-frame motion and is minimally affected by the areas of no texture since they are deemed to move consistently with the tracked feature.

Input sequences at frames 2, 8, 15.

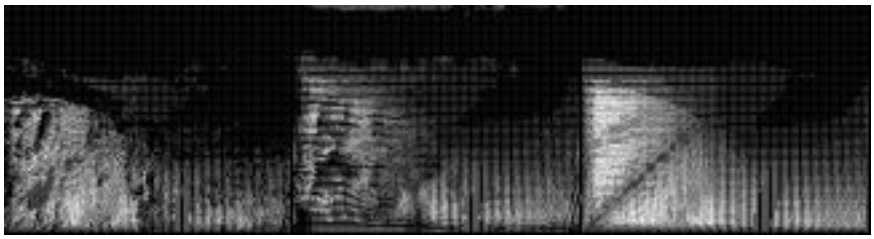


Segment corresponding to the distant mountain slope.

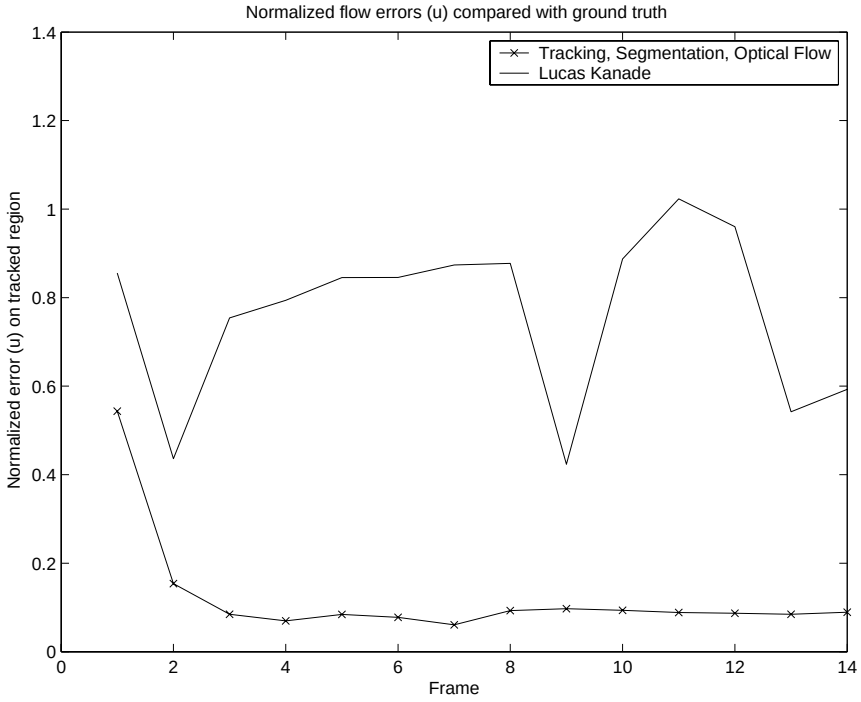


Segment corresponding to the left cliff.

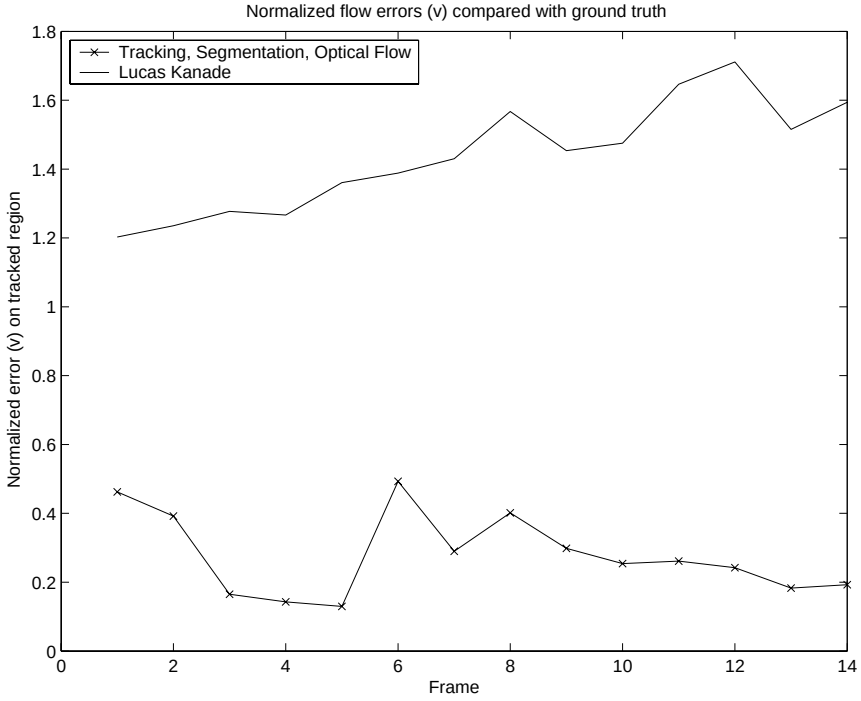
Segment corresponding to the middle valley.



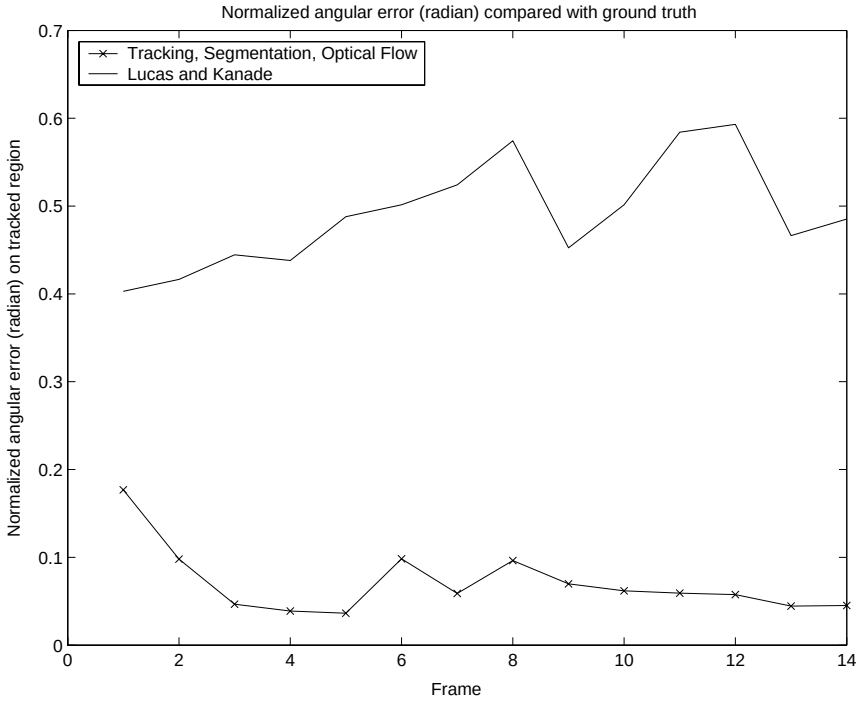
Needle maps of our algorithm, Lucas & Kanade and the ground truth.



Mean square flow error (u) on the left cliff segment compared with the ground truth.



Mean square flow error (v) on the left cliff segment compared with the ground truth.



Mean angular error (radian) on the left cliff segment compared with the ground truth.

Figure 3.5: The input image frames show the Yosemite sequence. Mean square/angular error of flow estimated by our algorithm and the Lucas and Kanade algorithm on the left cliff segment shown above. The ground truth is from Black's [4] web site.

In all experiments, the segmentation result improves over time because the algorithm needs a few frames to build the model of the tracked object. We run our experiments on a Linux PC with a 3.0 GHz Pentium IV CPU that delivers 1164 SPEC CINT2000 [10]. For input image frames of size 320×240, our motion segmentation and tracking algorithm using pixel statistic takes 0.28 seconds per frame on on the Pentium IV.

3.2. Patch statistic

For the image model using patch statistic, the image patch size is 5×5 pixels. We highlight the seed region in the output segmented images. The default (empirical) standard deviation $\vec{\sigma}_{def}$ of the various kind of noises used in the individual experiments are: $\sigma_n = 2.75$, $\sigma_a = 0.08$, $\sigma_u = \sigma_v = 0.2$, $\sigma_l = 0.001$, $\sigma_f = 0.5$. These $\vec{\sigma}_{def}$ parameters are determined empirically. We also compute an optimized parameters $\vec{\sigma}_o$ for each sequence. These $\vec{\sigma}_o$ parameters are estimated by using a Maximum Likelihood Estimator (MLE) for a few number of seed regions within the tracked region of an image sequence.

In the sixth experiment, we repeat the first experiment using the patch statistic. The output segments are very similar to the segment obtained using pixel statistic. We show the result of tracking on the hippopotamus for comparison.

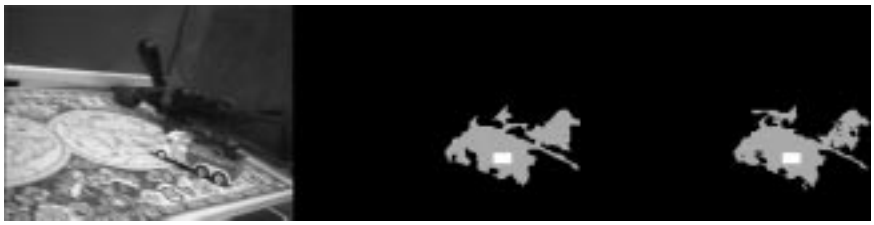
Frame 5 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

Frame 15 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

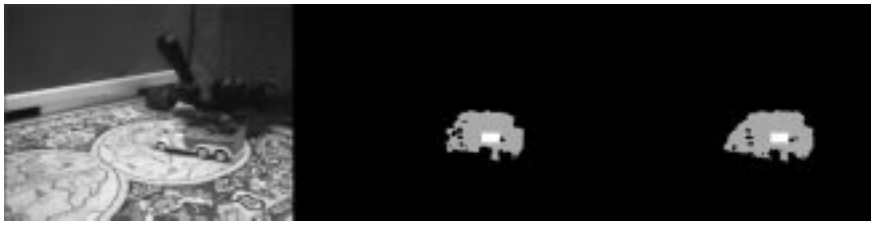
Frame 27 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

Figure 3.6: The toy animals sequence. The optimized $\vec{\sigma}_o$ parameters are: $\sigma_n = 0.97$, $\sigma_a = 0.0002$, $\sigma_u = 0.32$, $\sigma_v = 0.05$, $\sigma_l = 0.01$, $\sigma_f = 1.03$.

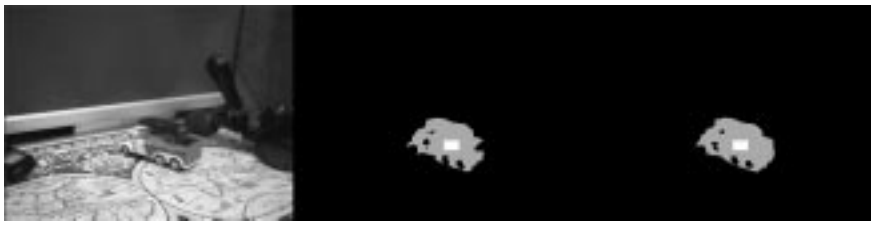
In the seventh experiment, we repeat the second experiment using the patch statistic. The output segments are very similar to the segment obtained using pixel statistic. We show the result of tracking on the truck for comparison.



Frame 1 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.



Frame 13 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

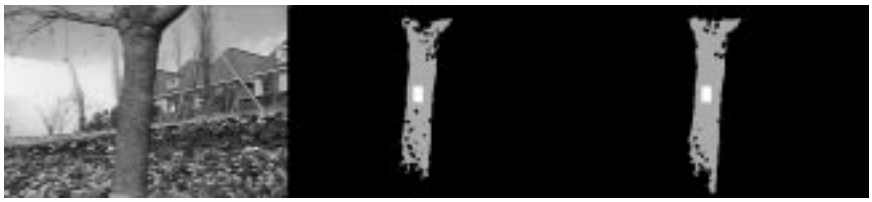


Frame 28 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

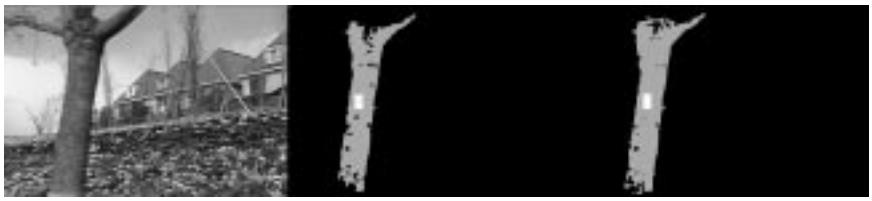
Figure 3.7: Turning truck in a moving background under constant lighting condition. The truck and the hippopotamus on top of it are segmented out. The wheels and the highly specular "engine block" of the truck are not segmented. The optimized $\vec{\sigma}_o$ parameters are: $\sigma_n = 1.18$, $\sigma_a = 0.055$, $\sigma_u = 0.26$, $\sigma_v = 0.24$, $\sigma_l = 0.05$, $\sigma_f = 6.45$.

In the next experiment, we run the algorithm on the Flower Garden sequence [4] that has a tree in the foreground and a flower garden and houses in the background.

Frame 1 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.



Frame 18 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.



Frame 29 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

Figure 3.8: Flower Garden image sequence. The moving tree in the foreground of the flower garden is segmented out by tracking a seed region on the tree trunk. The optimized $\vec{\sigma}_o$ parameters are: $\sigma_n = 1.28$, $\sigma_a = 0.23$, $\sigma_u = 0.128$, $\sigma_v = 0.359$, $\sigma_l = 0.0078$, $\sigma_f = 0.45$.

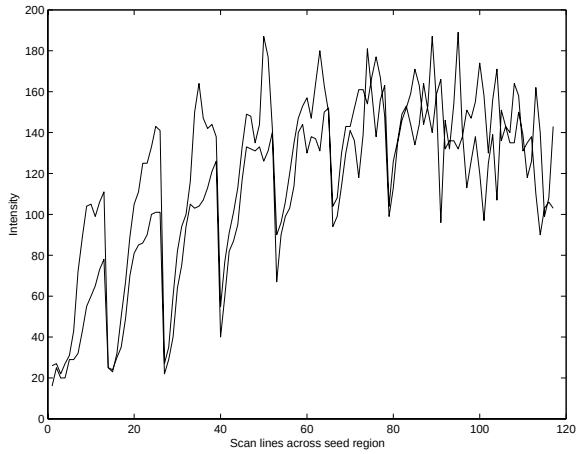
In the next experiment, the same toy truck is moving diagonally from the lower left hand corner to the upper right hand corner and is taken by a panning camera. A table lamp is placed near the right hand side of the images so that the truck goes from dark areas to bright areas. The biggest illumination change is between frames 9 and 10 where the seed region at the side of the truck undergoes up to about 40% change in intensity. The average change is roughly 10%.

Close-up of frame 9

Close-up of frame 10

Close-up of frame 12

Output from tracker for frame 9 and 10.

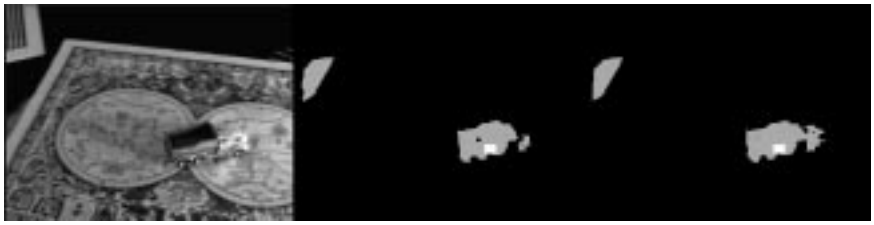


Scan lines across seed region in frame 9 and 10

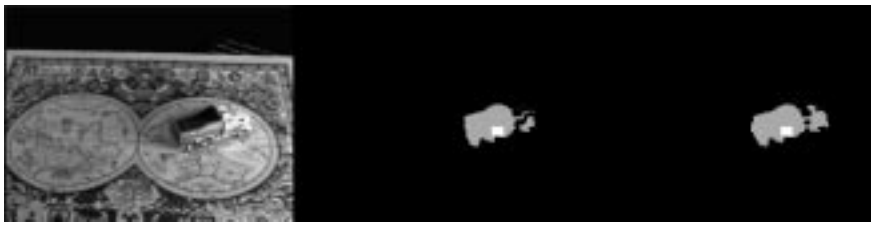
Figure 3.9: Close-up for frame 9 and 10 to illustrate the change in light intensity. The highly specular engine is not part of the segment.

The side panel of the truck is relatively dark in frame 1 whereas it is much brighter in frame 41. The image sequence shows a moving truck in a non-stationary background under varying lighting conditions.

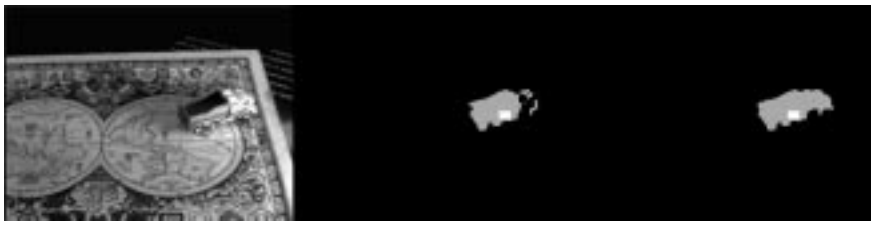
Frame 1 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.



Frame 11 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.



Frame 21 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.



Frame 31 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

Frame 41 and its output from tracker using default $\vec{\sigma}_{def}$ and optimized $\vec{\sigma}_o$.

Figure 3.10: Moving truck in a dynamic background under varying lighting conditions. The optimized $\vec{\sigma}_o$ parameters are: $\sigma_n = 2.585$, $\sigma_a = 0.0437$, $\sigma_u = 0.397$, $\sigma_v = 0.268$, $\sigma_l = 0.000002$, $\sigma_f = 2.245$.

The segmentation result improves over time because the algorithm needs a few frames to build the model of the tracked object. For input image frames of size 320×240 and image patch size of 5×5, our motion segmentation and tracking algorithm using patch statistic takes 1.32 seconds per frame on a Pentium IV 3.0 GHz PC compared with 0.28 seconds per frame using pixel statistic. Therefore, running our motion segmentation and tracking algorithm using patch statistic is about 5 times slower than using pixel statistic. However, running the algorithm using patch statistic can segment and track object in image frames under varying lighting conditions. Using pixel statistic, the algorithm can only segment and track object in image frames under constant lighting condition.

4. Conclusion

We describe a novel and efficient algorithm that performs motion segmentation and tracking using corner features. The algorithm automatically selects a good feature and segments out a region whose motion is consistent with the motion of the selected feature. We perform motion segmentation by an integrated segmentation and tracking process. We fit an affine optical flow model on the tracked region and align the image model representing the tracked object with the current frame. We segment out those pixels moving consistently with the seed region by thresholding either a pixel or patch statistic involving the image difference between the aligned image model and the current frame. In order to account for varying lighting conditions in the image frames, we use an image flow model that incorporates both percentage change and absolute change in image intensity. We have presented the results of running our algorithm on several real and synthetic image sequences. The method can obtain clear outlines of objects that undergo approximately affine motion, under many different conditions on images taken in our laboratory or standard image sequences.

References

[1] P. Anandan, “A Computational Framework and an Algorithm for the Measurement of Visual Motion,” *International Journal of Computer Vision* **2**(1989). pp. 283-310

[2] J. L. Barron, D. J. Fleet, and S. S. Beauchernin, “Performance of Optical Flow Techniques,” in *International Journal of Computer Vision,* (1994) pp. 43-77.

[3] A. Benedetti and P. Perona, “Real-time 2-D feature detection on a reconfigurable computer,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* (1998) pp. 586-593.

[4] Black, "Michael J Black: Image Sequences," in *http://www.cs.brown.edu/people/black/images.html*, (2002)

[5] M. Black and A. Jepson, "Mixture models for optical flow computation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, (1993) pp. 760-761.

[6] M. Black and A. Jepson, "EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," in *International Journal of Computer Vision*, (1998) pp. 1-29.

[7] G. D. Borshukov, G. Bozdagi, Y. Altunbasak, and A. M. Tekalp, "Motion Segmentation by Multistage Affine Classification," in *IEEE Transactions on Image Processing*, (1997) pp. 1591-1594.

[8] P. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. Lee, A. Leung, J. Lubin, and H. Shvaytser, "Object Tracking with a Moving Camera," in *Proceedings of Workshop on Visual Motion*, (1989) pp. 2-12.

[9] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, (2000) pp. 142-149.

[10] Standard Evaluation Corporation, "Submitted CPU2000 results," in *http://www.spec.org/osg/cpu2000/results/cint2000.html*, (2003)

[11] B. S. Everitt, *Cluster Analysis*, John Wiley & Sons, New York (1974).

[12] G. Hager and P. Belhumeur, "Efficient Region Tracking with Parametric Models of Geometry and Illumination," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1998) pp. 1025-1039.

[13] William W. Hager, "Updating the Inverse of a Matrix," *SIAM Review* **31**(2)(1989). pp. 221-239

[14] J. Heikkila, P. Sangi, and O. Silven, "Camera Motion Estimation from Non-Stationary Scenes Using EM-Based Motion Segmentation," in *Proceedings of 15th International Conference on Pattern Recognition*, (2000) pp. 370-374.

[15] B. K. P. Horn, *Robot Vision*, McGraw-Hill Book Company, New York (1986).

[16] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow - a Retrospective," *Artificial Intelligence* **59**(1993). pp. 81--87

[17] M. Irani, B. Rousso, and S. Peleg, "Computing Occluding and Transparent Motions," *International Journal of Computer Vision* **12**(1)(1994). pp. 5-16

[18] M. Isard and A. Blake, "Contour tracking by stochastic propagation of condition density," in *Proceedings of European Conference on Computer Vision*, (1996) pp. 343-356.

[19] M. Isard and A. Blake, "ICONDENSATION: Unifying low-level and high level tracking in a stochastic framework," in *Proceedings of European Conference on Computer Vision*, (1998) pp. 893-909.

[20] A. Jepson, D. Fleet, and T. F. El-Maraghi, "Robust Online Appearance Models for Visual Tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, (2001) pp. 415-422.

[21] R. Larsen and M. Marx, *Introduction to Mathematical Statistics and its Applications*, Prentice Hall (1986).

[22] B. D. Lucas and T. Kanade, "An Iterative Technique of Image Registration and Its Application to Stereo," in *Proceedings of 7th International Joint Conference on Artificial Intelligence*, (1981) pp. 674-679.

[23] P. C. Mahalanobis, "On Tests and Measures of Groups Divergence I," in *Journal of the Asiatic Society of Benagal*, (1930) pp. 541.

[24] F. Meyer and P. Bouthemy, "Region-Based Tracking Using Affine Motion Models in Long Image Sequences," in *CVGIP:Image Understanding*, (1994) pp. 119-140.

[25] S. Negahdaripour and C. M. Yu, "A generalized brightness change model for computing optical flow," in *Proceedings of Fourth International Conference on Computer Vision*, (1993) pp. 2-11.

[26] W. Press, *Numerical recipes in C: the art of scientific computing*, Cambridge University Press (1992).

[27] J. Sherman and W. J. Morrison, “Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or a Given Row of the Original Matrix.,” *Ann. Math. Statist.* **20**(1949).

[28] H. Sidenbladh, M. Black, and D. Fleet, “Stochastic Tracking of 3D Human Figures using 2D Image Motion,” in *Proceedings of European Conference on Computer Vision*, (2000) pp. 702-718.

[29] H. Stark and J. W. Woods, *Probability and Random Processes with Applications to Signal Processing*, Prentice Hall (2002).

[30] A. Strehl and J. K. Aggarwal, “A New Bayesian Relaxation Framework for the Estimation and Segmentation of Multiple Motions,” in *Proceedings of 4th IEEE Symposium on Image Analysis and Interpretation*, (2000)

[31] C. Tomasi and T. Kanade, “Detection and Tracking of Point Features,” in *Technical Report CMU-CS-91-132, Carnegie Mellon University*, (1991)

[32] N. Vasconcelos and A. Lippman, “Empirical Bayesian EM-based Motion Segmentation,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, (1997) pp. 527-532.

[33] J. Y. A. Wang and E. H. Adelson, “Representing moving images with layers,” in *IEEE Transactions on Image Processing*, (1994) pp. 625-638.

[34] M. Woodbury, “Inverting Modified Matrices,” *Memorandum Rept. 42, Statistical Research Group*, Princeton University, (1950).

[35] Y. Ye, J. Tsotsos, E Harley, and K Bennet, “Tracking a person with pre-recorded image database and a pan, tilt and zoom camera,” in *Machine Vision and Applications*, (2000) pp. 32-43.