Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Vision and Image Understanding 101 (2006) 45–64
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

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, Ont., Canada M3J 1P3

Received 15 August 2003; accepted 6 July 2005
Available online 3 October 2005

Abstract

We present a novel and efficient motion segmentation and tracking algorithm that follows the shift and align paradigm. We introduce two statistical tests to evaluate the similarity of aligned image pixels or patches and we use them to determine the spatial extend of each segment. The one statistical test is fast and accurate when the noise is moderate and the other employs a sophisticated noise model involving the Mahalanobis distance to handle correlated noise. Direct computation of the Mahalanobis distance is prohibitively expensive so we apply the Sherman–Morrison–Woodbury identity and amortization to reduce the cost by several orders of magnitude. We tested both 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 is the partitioning of pixels having similar optical flow into groups such that each group corresponds to the motion of the projection of a moving object. Motion segmentation would be a much easier problem if optical flow was available, but this is a chicken and egg problem because optical flow would be more accurate if segmentation, and thus the motion boundaries, were available.

Previous motion segmentation techniques [41,8,45] work 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 clustering for merging optical flow into regions [24,44,18,42,2,13,49,50,37,15]. Another category of motion segmentation techniques perform motion segmentation by iteratively estimating piecewise constant or affine optical flow, warping the image and estimating the support of the segments [9,21,25,47,48]. Level set [11,30] is also becoming a popular approach for motion segmentation.

A tracking algorithm measures and predicts the motion of a moving object over time. Contours [22,35] corresponding to the silhouette of moving objects are commonly used feature for tracking. The coherence of a moving region [16,7,46] corresponding to the projection of a surface of the moving object can provide a powerful constraint and has been used for tracking. Color [10,23,26] of a moving object is often easily distinguished from the surroundings and can be combined with sophisticated statistical or three-dimensional techniques to perform tracking accurately and efficiently. Subspace representation [6,53] of the tracked object uses techniques in the principal components analysis (PCA) family to model the possible deformations of the object and in conjunction with other statistical techniques like particle filtering recover the motion model of the object. Instead of tracking attributes belonging to the moving object, an orthogonal tracking approach is to find the moving objects in a dynamic scene by performing

* Corresponding author. Fax: +1 416 736 5872.
E-mail addresses: kywong@cs.yorku.ca (K.Y. Wong), minas@cs.yorku.ca (M.E. Spetsakis).

1077-3142/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2005.07.001

image difference on the image frames with known or stationary background [52]. 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 [31] tracked the motion of regions computed by a motion segmentation algorithm over time assuming a model for the motion and deformation of the regions and then used the Kalman filter 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 objects (or parts of objects) having distinct motion. The algorithm starts with manually or automatically selected features, tracks them by computing their motion, warps one image towards the next using this computed motion and then subtracts the two images. Pixels having motion consistent with the computed motion have small difference between the two frames and vice versa. The major contribution of the paper is that we introduce fairly accurate and efficient statistics, one pixel-wise and the other patch-wise, that provide a measure of the noise in the image difference. The pixel-wise statistic assumes temporal independence of noise in the differences between pairs of images. It does not need a spatial independence assumption because it works on a single pixel difference at a time. This allows us to design a simple and computationally efficient algorithm but a more robust statistic should take into account the noise dependence and benefit from it rather than avoiding it since in reality, the noise in neighboring pixels is very often correlated. This effect is prominent when an image sequence is taken under varying lighting conditions or when the tracking produces imperfect alignment between the images. To model such dependence we developed a patch-wise statistic, along with an efficient algorithm to compute it using the Sherman–Morrison–Woodbury identity which is a little used numerical analysis technique for inversion of matrices similar to the covariance matrix involved in our 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 can have applications to other Computer Vision problems that use similar covariance matrices [53,14].

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) with excellent results. We also use our tracking and segmentation algorithm as preprocessing to optical flow computation and we can improve the performance of these algorithms especially when the inter-frame motion is large.

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 an affine flow u_a between images I_N and I_{N-1} which is an adequate model for the seed region and its surrounding area. 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 because there are several kinds of random noise that corrupt the images, like change in illumination, camera shot noise, digitization noise, imperfect alignment, etc.

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 introduce two statistics to compute a measure of the noise, one that avoids and one that explicitly models most of these forms of noise. The pixel-wise statistic does not need a spatial independence assumption so it needs to model only camera shot noise and pixel-wise misalignment, and thus it is computationally inexpensive. The patch-wise statistic explicitly models all these forms of noise and we designed a computationally efficient and exact algorithm to compute it. We also developed a maximum likelihood estimator (MLE) to fit the model parameters.

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 also used our algorithm as a preprocessing step to facilitate the solution of the optical flow problem.

In the rest of the paper, we describe the major components of the algorithm:

- Feature (seed region) selection.
- Tracking by fitting of successively more refined flow models to the seed region.
- Elaborate but efficient noise model for motion segmentation. We provide both a pixel-wise model and a patch-wise model.
- Postprocessing.
- Application of the algorithm to the computation of optical flow.
- Experiments on segmentation and tracking, using both statistics as well as experiments with optical flow.

3. 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 may straddle motion boundaries.

We detect corners in an image frame by the corner detection algorithm proposed by Tomasi and Kanade [43]. The corner detection algorithm finds feature points that have good localization in all directions. Among the N identified corners, we randomly pick one of them. This becomes 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

- 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%.
- The segment is not very small. We discard segments that are less than 1% of the area of the image.
- 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 are found, we keep track of all the corner features that are within the segment. If the tracked feature generates in a subsequent frame a segment that does not satisfy all of the above criteria, we generate seed regions around other corner features inside the segment in an attempt to continue the tracking/segmentation of the region. 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.

4. 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, Nth image frames I_{N-1}, I_N

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

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

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

The integer optical flow of the region $\vec{u}_{\mathrm{int}}$ is taken to be the $\vec{u}$ that gives the minimum SSD given α in Eq. (1).

$$\vec{u}_{\mathrm{int}} = \min_{\vec{u} \in \vec{u}_{\max}} \mathrm{SSD}(\vec{u}),$$

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

To compute the subpixel flow of the region, we first shift I_{N-1} with the integer flow $\vec{u}_{\mathrm{int}}$ to reduce interframe motion and then find the subpixel displacement $\vec{u}_{\mathrm{s}}$ that yields the minimum sum of squared differences. This can be done by either searching or any gradient based optical flow technique.

Next we compute affine flow. To do this we need to enlarge the seed region because affine deformations such as shrinkage and rotation are indiscernible in a small region and thus hard to compute. On the other hand, if we enlarge the seed region too much 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 expanded several times (4 in our experiments) but we exclude pixels that were not part of the same segment in the final segmentation result obtained from the previous pair of frames to avoid inclusion of discontinuities.

The affine flow $\vec{u_{\mathrm{a}}}$ for a region R_{a} is defined by six parameters u_x, u_y, v_x, v_y, u_0, and v_0

$$\vec{u_{\mathrm{a}}} = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \vec{x} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}. \quad (2)$$

Plugging Eq. (2) into the optical flow equation and forming the SSD we get

$$\sum_{\vec{x} \in R_{\mathrm{a}}} \left(\Delta \vec{I}[\vec{x}] + \nabla I[\vec{x}] \cdot \vec{u_{\mathrm{a}}} \right)^2, \quad (3)$$

where

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

and $\nabla I[\vec{x}]$ is the gradient of the average of I_N and αI_{N-1} [20]. We apply standard least squares to compute the six affine parameters by solving the normal equation from Eq. (3).

One of the hardest problems related to differential flow techniques is taking spatial and temporal derivatives. We did not have many difficulties because our incremental method uses the results of the previous stages of the computation to reduce the interframe motion and avoid boundaries.

5. Motion segmentation

After obtaining the affine motion between the last two frames one could warp I_{N-1} towards I_N and then

subtract them and square the difference. One would expect 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:

- There is a certain amount of noise in any image like digitization noise, random white noise, etc. In measurements we did on one of our cameras these forms of noise had a combined standard deviation as high as 4.5, which is substantial. In most of our experiments this noise had a standard deviation of about 1–3.
- We approximate a rather complex optical flow with affine flow. While the approximation is quite good, the standard deviation of the error in the flow was empirically found to be about 0.2 pixels and this can induce substantial error in the sum of squared difference and give rise to false negatives.
- When we warp image I_{N-1} using the affine flow we might align two totally different regions not belonging to the object that happen to have identical texture or color. This can give rise to a large number of false positives.
- 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).

In the next section, we describe two different statistics to perform segmentation. A pixel statistic that measures how well each pixel in the image model $\hat{I}_N$ matches the corresponding pixels in the previous images. And a patch statistic that measures how well a small image patch centered at each pixel matches the motion of the tracked object in the $N-1$th frame. The pixel statistic is simple and hence fast. The patch statistic is slower but can account for effects like illumination change across successive frames.

5.1. Pixel statistic

It is clear from the above discussion that one cannot do a simple threshold of the squared differences of the aligned images. We obviously need to average more than one squared difference but we would prefer to avoid taking the average over a patch, since this assumes that the noise in the pixels within the patch is independent. The alternative is to take the average over time for the same pixel and this is where we need the assumption of temporal independence of noise. This temporally averaged squared difference is

$$\Delta\hat{I}_N^2 = \frac{1}{N-1}\sum_{i=1}^{N-1}\left(\frac{{}^N I_i - I_N}{\sigma_p}\right)^2, \tag{4}$$

where ${}^N I_i$ is the ith image frame corrected for intensity gain by multiplying with α and aligned with the Nth frame using the affine optical flow derived during tracking, and σ_p^2 is the variance of the residual noise in the difference between the aligned frames. We derive this variance later in this section. The summation in Eq. (4) is the normalized 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. (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. (4). Expanding it we notice

$$\Delta\hat{I}_N^2 = \frac{1}{N-1}\left(\sum_{i=1}^{N-1}\frac{{}^N I_i^2}{\sigma_p^2} - 2\frac{\left(\sum_{i=1}^{N-1}{}^N I_i\right)}{\sigma_p}\frac{I_N}{\sigma_p} + \frac{I_N^2(N-1)}{\sigma_p^2}\right) \tag{5}$$

$$= \hat{I^2}_N - 2\hat{I}_N\frac{I_N}{\sigma_p} + \frac{I_N^2}{\sigma_p^2}, \tag{6}$$

where $\hat{I}_N = \frac{\sum_{i=1}^{N-1}{}^N I_i}{(N-1)\sigma_p}$, $\hat{I^2}_N = \frac{\sum_{i=1}^{N-1}{}^N I_i^2}{(N-1)\sigma_p^2}$ are first and second moments of the images aligned with the Nth image frame. To avoid keeping all the aligned images and their squares, we use an exponential sliding window in the computation of the moments. So

$$\begin{aligned}\hat{I}_N &= h\,{}^N\hat{I}_{N-1} + (1-h)I_N,\\ \hat{I^2}_N &= h\,{}^N\hat{I^2}_{N-1} + (1-h)I_N^2,\end{aligned} \tag{7}$$

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 Nth frame, ${}^N\hat{I}_{N-1}$, ${}^N\hat{I^2}_{N-1}$ are the first and second moments of the normalized aligned images. Using these values for $\hat{I}_N$ and $\hat{I^2}_N$ the mean of the statistic $\Delta\hat{I}_N^2$ is

$$E\left(\Delta\hat{I}_N^2\right) = 1$$

and the variance is approximately

$$\mathrm{Var}\left(\Delta\hat{I}_N^2\right) \approx 2\frac{1-h}{1+h}.$$

We use thresholding to identify pixels belonging to the tracked object. The pixels whose tracking statistics in Eq. (6) are less than threshold are classified as belonging to the tracked object. The threshold t_{value} is set equal to

$$t_{\text{value}} = 1 + z\sqrt{2\frac{1-h}{1+h}}, \tag{8}$$

where z is a constant. To get an idea of what z should be we use hypothesis testing and define the null hypothesis H_0 to

be that the pixel in question moves with the computed flow and thus the image difference is noise with variance σ_p^2 and the alternative hypothesis H_1 is that the pixel does not move with the computed flow. If we choose 0.01 level of significance, then z is 3. This should give 1% false negatives. If this error rate is inappropriate we can decrease the level of significance by choosing a larger z. In our experiments $z = 4$.

We estimate the value of σ_p^2 as a sum of two variance components

$$\sigma_p^2 = \sigma_c^2 + \sigma_f^2,$$

where σ_c^2 denotes the variance of the camera noise and σ_f^2 is the variance of the motion noise. The camera noise represents the white noise generated by the electronic circuits of the camera and the motion noise is generated by the discrepancy between the affine flow model and the real optical flow. These two components of noise are produced by two unrelated processes so they can be considered independent. The camera noise variance σ_c^2 was empirically set to 1 and this worked fine for all the images we tried using the pixel statistic. The variance of the motion noise can be calculated with the help of the optical flow equation

$$\hat{I}_{N,x} \cdot \Delta u + \hat{I}_{N,y} \cdot \Delta v + \hat{I}_{N,t} = 0,$$

where $\hat{I}_{N,x}$ and $\hat{I}_{N,y}$ are the x and y derivatives of $\hat{I}_N$ and $\hat{I}_{N,t} = \hat{I}_N - {}^{\mathrm{N}}\hat{I}_{N-1}$ is the noise due to discrepancy between the actual and the affine flow and Δu and Δv are the flow residuals after warping with the affine flow, similar to [36]. Hence, the variance of $\hat{I}_{N,t}$ is

$$\begin{aligned}\sigma_f^2 &= \mathrm{Var}(\hat{I}_{N,t}) = \mathrm{Var}(\hat{I}_{N,x}\Delta u + \hat{I}_{N,y}\Delta v) \\ &= (\hat{I}_{N,x}^2 + \hat{I}_{N,y}^2)\sigma_{uv}^2, \end{aligned} \tag{9}$$

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 these are the errors in the model and not the uncertainty in the optical flow which is usually very anisotropic due to the aperture problem. So, we classify those pixels whose tracking statistics t_{stat} (from Eq. (6)) are less than their corresponding threshold values t_{value} (from Eq. (8)) as belonging to the tracked object

$${}^{N}I_{\mathrm{track}} = \begin{cases} 1 & \text{if } \Delta\hat{I}_N^2 \leqslant t_{\mathrm{value}}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

5.2. Patch statistic

The obvious choice for a patch statistic is the sum of squared differences, but this would require a noise independence assumption between neighboring pixels. In the pixel-wise statistic, we avoided the noise independence question by defining the statistic on individual pixels but we have to confront it now if we want the method to be applicable in harder situations. One such situation 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.

Let $\Delta\vec{I}$ be the image difference between the Nth image frame and an image model of the tracked region $\hat{I}$ aligned by the optical flow and corrected for light intensity gain. We identify pixels that follow the same motion model as the seed region by evaluating a statistic on the image difference $\Delta\vec{I}$. This statistic is defined on small $k \times k$ image patches. As it will become apparent later we need to use vector notation so 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

$$\Delta\vec{I} = {}^{N}\vec{\hat{I}}_{N-1} - \vec{I}_N, \tag{11}$$

where the left superscript N denotes warping by affine flow to align it to the Nth frame and corrected for intensity gain by multiplying with α, ${}^{N}\hat{I}_{N-1}$ 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, $\Delta\vec{I}$ would be zero. But instead it is

$$\begin{aligned}\Delta\vec{I} = {} & \Delta\vec{n} + \mathrm{Diag}({}^{N}\vec{\hat{I}}_{N-1,x})\Delta\vec{u}_\alpha + \mathrm{Diag}({}^{N}\vec{\hat{I}}_{N-1,y})\Delta\vec{v}_\alpha \\ & + {}^{N}\vec{\hat{I}}_{N-1,x}\Delta u + {}^{N}\vec{\hat{I}}_{N-1,y}\Delta v + {}^{N}\vec{\hat{I}}_{N-1}\Delta l + \Delta f, \end{aligned} \tag{12}$$

where $\Delta\vec{n}$ is the pixel-wise noise, a k^2 vector of random independent variables, $\mathrm{Diag}({}^{N}\vec{\hat{I}}_{N-1,x})$, $\mathrm{Diag}({}^{N}\vec{\hat{I}}_{N-1,y})$ are diagonal matrices whose diagonals contain the elements of the vector ${}^{N}\vec{\hat{I}}_{N-1,x}$, ${}^{N}\vec{\hat{I}}_{N-1,y}$, respectively. ${}^{N}\vec{\hat{I}}_{N-1,x}$ and ${}^{N}\vec{\hat{I}}_{N-1,y}$ are vectors from a patch of the x and y derivatives of ${}^{N}\hat{I}_{N-1}$ and $\Delta\vec{u}_\alpha$, $\Delta\vec{v}_\alpha$ are the random vectors denoting the pixel-wise error in flow in the image frames. The above three random vectors represent the pixel-wise independent noise and are of the same nature as the camera and motion noise in the pixel statistic. They are similar to other models used in visual motion, either implicitly or explicitly [36]. Next, we describe the random variables defined for patches. Δu, Δv are random variables representing the residual flow of the whole patch, Δl, Δf are random variables that model respectively the proportional and additive change of the light intensity [32] over patches between the image frames I_{N-1} and I_N.

In other words, we model the noise using seven terms. A white noise $\Delta\vec{n}$ that is attributed to digitization and electronic noise in the camera and anything that can be considered independent and identically distributed (i.i.d). The next component is the pixel-wise motion noise which is associated with the next two terms involving $\Delta\vec{u}_\alpha$, $\Delta\vec{v}_\alpha$. This noise comes from either independent motion of image details within an image patch (for example, the motion of

individual blowing leaves of a tree) or from aliasing and interpolation error. Aliasing is present in cameras where the lens projects on the image plane more detail than can be recorded by the CCD and interpolation noise is generated by warping the image using less than perfect interpolation algorithms. These three kinds of noise appeared to have similar statistical properties and we found empirically their combined effect to be proportional to the derivatives of the images. Given their apparent similarity we treat them together for economy. Although one can potentially derive a mathematical expression relating the aliasing or interpolation noise between neighboring pixels, the relation appears too complex to incorporate into our model so we treat it as independent noise. Our experiments did not produce any indication that this is a strong assumption.

The next two terms model the patch-wise motion noise Δu, Δv, the discrepancy between the average real motion of the patch and the motion associated with the affine model of the segment. We do not need an affine model for Δu, Δv since we now operate on a small $k \times k$ patch and the images are already aligned. The last two terms associated with the random variables Δl, Δf represent the change of illumination component of our model. The change of illumination affects the image intensity non linearly, but an affine approximation is adequate [32].

Of the seven random variables, $\Delta\vec{n}$, $\Delta\vec{u}_\alpha$, $\Delta\vec{v}_\alpha$ are vectors, the rest being scalar. One can notice that in Eq. (12) we have products like ${}^N\vec{\hat{I}}_{N-1,x}\Delta\vec{u}_\alpha$ where ${}^N\vec{\hat{I}}_{N-1,x}$ is treated as a deterministic noise free quantity, thus ignoring second order noise terms. But three reasons contribute to keep the effects of the second order terms small. First, the noise is mostly small hence the higher order terms are even smaller. Second, the noise is assumed independent so the effect of these terms is minimized when we take the expected values. Third, we used the model of the image ${}^N\hat{I}_N$ instead of the image since it is an average of past images and the noise has decreased.

Since we represent the image patches as vectors of length k^2, Eq. (12) can be rewritten in vector and matrix notation:

$$\Delta\vec{I} = \vec{n} + \boldsymbol{U}\vec{u}, \tag{13}$$

where $\vec{n}$ is the sum of the camera and pixel-wise motion noise $(\Delta\vec{n} + \mathrm{Diag}({}^N\vec{\hat{I}}_{N-1,x})\Delta\vec{u}_\alpha + \mathrm{Diag}({}^N\vec{\hat{I}}_{N-1,y})\Delta\vec{v}_\alpha)$, $\boldsymbol{U}$ is a $k^2 \times 4$ matrix whose columns store the three k^2 vectors ${}^N\vec{\hat{I}}_{N-1,x}$, ${}^N\vec{\hat{I}}_{N-1,y}$, ${}^N\vec{\hat{I}}_{N-1}$, from Eq. (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. (13) defines a noise model for an image patch difference. This model applies to patches with motion similar to the motion of the tracked object, but not to patches moving differently. We can then use a statistical test to decide for every patch if it follows our model. If it does we can assume that the motion of this patch is consistent with the motion of the tracked object. If it does not follow the model, then it moves differently. The method of choice is a χ^2 test on the Mahalanobis distance [12,29] which is defined as

$$D_m^2 = \Delta\vec{I}^T \boldsymbol{C}_{\Delta\vec{I}}^{-1} \Delta\vec{I}, \tag{14}$$

where $\boldsymbol{C}_{\Delta\vec{I}}$ is the covariance of $\Delta\vec{I}$. 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.

There are several issues that we need to address before we can apply the χ^2 test. The first issue is the estimation of the statistical parameters of the seven random variables we introduced in Eq. (12). We will describe this at the end of this section. A second issue is the high cost incurred by the computation and inversion of the covariance matrix $\boldsymbol{C}_{\Delta\vec{I}}$ of $\Delta\vec{I}$ due to the high dimensionality of $\Delta\vec{I}$, the image patch vector. $\Delta\vec{I}$ 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 derive the covariance matrix and show how to reduce the cost by several orders of magnitude.

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

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

Assuming that these two noise terms are independent

$$\begin{aligned}\boldsymbol{C}_{\Delta\vec{I}} &= \mathrm{Cov}(\vec{n}) + \mathrm{Cov}(\boldsymbol{U}\cdot\vec{u}) \\ &= \mathrm{Cov}(\vec{n}) + \boldsymbol{U}\mathrm{Cov}(\vec{u})\boldsymbol{U}^T = \boldsymbol{C}_n + \boldsymbol{U}\boldsymbol{C}_u\boldsymbol{U}^T,\end{aligned} \tag{15}$$

where $\boldsymbol{C}_n = \mathrm{Cov}(\vec{n})$ is a $k^2 \times k^2$ diagonal matrix with each diagonal entry equal to the corresponding element from the vector $\vec{1}_{k^2}\sigma_n^2 + {}^N\vec{\hat{I}}_{N-1,d}\sigma_a^2$, $\vec{1}_{k^2}$ is a k^2 vector of 1s, ${}^N\vec{\hat{I}}_{N-1,d}$ is a vector from a patch of the sum of squares of the x and y derivatives of ${}^N\vec{\hat{I}}_{N-1}$ and σ_n^2, σ_a^2 are scalar constants representing the variances of the camera noise and pixel-wise motion noise. We assume that $\boldsymbol{C}_u$ has the form

$$\boldsymbol{C}_u = \mathrm{Cov}(\vec{u}) = \begin{bmatrix} \sigma_u^2 & 0 & 0 & 0 \\ 0 & \sigma_v^2 & 0 & 0 \\ 0 & 0 & \sigma_l^2 & 0 \\ 0 & 0 & 0 & \sigma_f^2 \end{bmatrix},$$

where the scalars σ_u^2, σ_v^2, σ_l^2, σ_f^2 denote the variance of random variables Δu, Δv, Δl, Δf, respectively. If we had a different model for the motion noise then $\boldsymbol{C}_u$ might not be diagonal but this would have only a minimal effect in the total cost of computation. The form of the covariance matrix $C_{\Delta\vec{I}}$ is not uncommon [14,53].

Our next goal is to avoid the direct inversion of $\boldsymbol{C}_{\Delta\vec{I}}$ which takes $O((k^2)^3) = O(k^6)$. We can do this if we notice that $\boldsymbol{C}_{\Delta\vec{I}}$ is the sum of the diagonal matrix $\boldsymbol{C}_n$ and the rank-4 matrix $\boldsymbol{U}\boldsymbol{C}_u\boldsymbol{U}^T$ (since $\boldsymbol{U}$ is a $k^2 \times 4$ matrix). In other words $\boldsymbol{C}_{\Delta\vec{I}}$ is a rank-4 update to a diagonal matrix. Such matrices can be inverted by using the Sherman–Morrison–Woodbury (SMW) identity [17,34,51], a little known numerical analysis tool. This identity is also known as the Matrix Inversion Lemma and has been used scantly in Computer Vision [38,54]. The SMW identity has many different forms, and the one suitable for symmetric matrices is

$$\boldsymbol{C}_{\Delta\vec{I}}^{-1} = \boldsymbol{C}_n^{-1} - \boldsymbol{C}_n^{-1}\boldsymbol{U}(\boldsymbol{C}_u^{-1} + \boldsymbol{U}^T\boldsymbol{C}_n^{-1}\boldsymbol{U})^{-1}\boldsymbol{U}^T\boldsymbol{C}_n^{-1}. \quad (16)$$

We note that $\boldsymbol{C}_s = (\boldsymbol{C}_u^{-1} + \boldsymbol{U}^T\boldsymbol{C}_n^{-1}\boldsymbol{U})$ 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 $\boldsymbol{C}_n^{-1}$ of the diagonal matrix $\boldsymbol{C}_n$ 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 $\boldsymbol{U}$ and denote them by the vectors $\vec{I}_x$, $\vec{I}_y$, $\vec{I}$, $\vec{1}$ then $\boldsymbol{C}_s$ can be written as

$$\boldsymbol{C}_s = \begin{bmatrix} \frac{1}{\sigma_u^2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_v^2} & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_I^2} & 0 \\ 0 & 0 & 0 & \frac{1}{\sigma_f^2} \end{bmatrix} + \begin{bmatrix} \vec{I}_x^T\boldsymbol{C}_n^{-1}\vec{I}_x & \vec{I}_x^T\boldsymbol{C}_n^{-1}\vec{I}_y & \vec{I}_x^T\boldsymbol{C}_n^{-1}\vec{I} & \sum\vec{I}_x^T\boldsymbol{C}_n^{-1} \\ \vec{I}_y^T\boldsymbol{C}_n^{-1}\vec{I}_x & \vec{I}_y^T\boldsymbol{C}_n^{-1}\vec{I}_y & \vec{I}_y^T\boldsymbol{C}_n^{-1}\vec{I} & \sum\vec{I}_y^T\boldsymbol{C}_n^{-1} \\ \vec{I}^T\boldsymbol{C}_n^{-1}\vec{I}_x & \vec{I}^T\boldsymbol{C}_n^{-1}\vec{I}_y & \vec{I}^T\boldsymbol{C}_n^{-1}\vec{I} & \sum\vec{I}^T\boldsymbol{C}_n^{-1} \\ \sum\vec{I}_x^T\boldsymbol{C}_n^{-1} & \sum\vec{I}_y^T\boldsymbol{C}_n^{-1} & \sum\vec{I}^T\boldsymbol{C}_n^{-1} & \sum\boldsymbol{C}_n^{-1} \end{bmatrix}.$$

For each pixel, we need to evaluate product such as $\vec{I}_x^T\boldsymbol{C}_n^{-1}\vec{I}_x$ or $\vec{I}_x^T\boldsymbol{C}_n^{-1}\vec{I}_y$. Direct evaluation of these products is quite expensive even for small k because it requires 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 $\vec{I}_x^T\boldsymbol{C}_n^{-1}\vec{I}_y$ as example in Appendix A.

The next step is to compute the Mahalanobis distance. From Eq. (14) and Eq. (16), we have

$$D_m^2 = (\Delta\vec{I})^T(\boldsymbol{C}_n^{-1} - \boldsymbol{C}_n^{-1}\boldsymbol{U}\boldsymbol{C}_s^{-1}\boldsymbol{U}^T\boldsymbol{C}_n^{-1})(\Delta\vec{I}). \quad (17)$$

We invert $\boldsymbol{C}_s$ using Cholesky factorization and get $\boldsymbol{C}_s^{-1} = \boldsymbol{L}^{-T}\boldsymbol{L}^{-1}$ where $\boldsymbol{L}$ is a 4×4 upper triangular matrix. Substituting into Eq. (17)

$$\begin{aligned} D_m^2 &= \Delta\vec{I}^T\boldsymbol{C}_n^{-1}\Delta\vec{I} - (\Delta\vec{I}^T\boldsymbol{C}_n^{-1}\boldsymbol{U}\boldsymbol{L}^{-T})(\boldsymbol{L}^{-1}\boldsymbol{U}^T\boldsymbol{C}_n^{-1}\Delta\vec{I}) \\ &= \Delta\vec{I}^T\boldsymbol{C}_n^{-1}\Delta\vec{I} - ||\boldsymbol{L}^{-1}\boldsymbol{U}^T\boldsymbol{C}_n^{-1}\Delta\vec{I}||^2. \end{aligned} \quad (18)$$

The product $\boldsymbol{U}^T\boldsymbol{C}_n^{-1}\Delta\vec{I}$ is a vector of length 4

$$\begin{bmatrix} \vec{I}_x^T \\ \vec{I}_y^T \\ \vec{I}^T \\ \vec{1}^T \end{bmatrix}\boldsymbol{C}_n^{-1}\Delta\vec{I} = \begin{bmatrix} \vec{I}_x^T\boldsymbol{C}_n^{-1}\Delta\vec{I} \\ \vec{I}_y^T\boldsymbol{C}_n^{-1}\Delta\vec{I} \\ \vec{I}^T\boldsymbol{C}_n^{-1}\Delta\vec{I} \\ \vec{1}^T\boldsymbol{C}_n^{-1}\Delta\vec{I} \end{bmatrix}$$

and we use the same technique as described in the appendix to reduce the computation to 4 convolutions. Hence, 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 $\boldsymbol{C}_s$ and $\boldsymbol{U}$.

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. The Mahalanobis distance of two k^2 vectors under Gaussian assumption follows the χ^2 distribution with k^2 degrees of freedom and has mean k^2 and variance $2k^2$ [40]. Since the noise in most real images is only approximately Gaussian, the behaviour of the Mahalanobis distance will deviate from the ideal and to reduce the effect of non-Gaussianity we perform temporal smoothing of the 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 Nth frame, D_m^2 is the Mahalanobis distance from the current (Nth, $N-1$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 [40] and thus our χ^2 test becomes more reliable. In our experience there was a noticeable improvement in the segmentation using this temporal smoothing.

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

$${}^{(N)}I_{\text{track}} = \begin{cases} 1 & \text{if } d_{t,N} <= c_{\text{value}}, \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

The value of c_{value} can be taken from a χ^2 table [40]. For example, for $k^2 = 25$ degrees of freedom using 0.995 level of confidence, the threshold is $t_{\text{stat}} = 46.9$. We prefer higher levels of confidence to reduce the number of false negatives.

5.3. Estimation of the model parameters

There are two ways to estimate the model parameters. The first is to start from the first principles and compute the shot noise, the interpolation noise, motion noise, etc. This is clearly a very challenging approach. The second way is to consider the model as a mathematical abstraction and compute the parameters that best fit the observed data. This is the approach followed in this paper. Although this process dissociates the parameters from their physical meaning, it produces a tighter fitting model.

We use maximum likelihood estimation (MLE) to fit our parameters. We select manually a few seed regions ($\kappa = 5$ seed regions of size 5×5 was enough for all our experiments since we only have to estimate 7 parameters) 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 zero mean multidimensional Gaussian distribution with covariance equal to $\boldsymbol{C}_k$

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

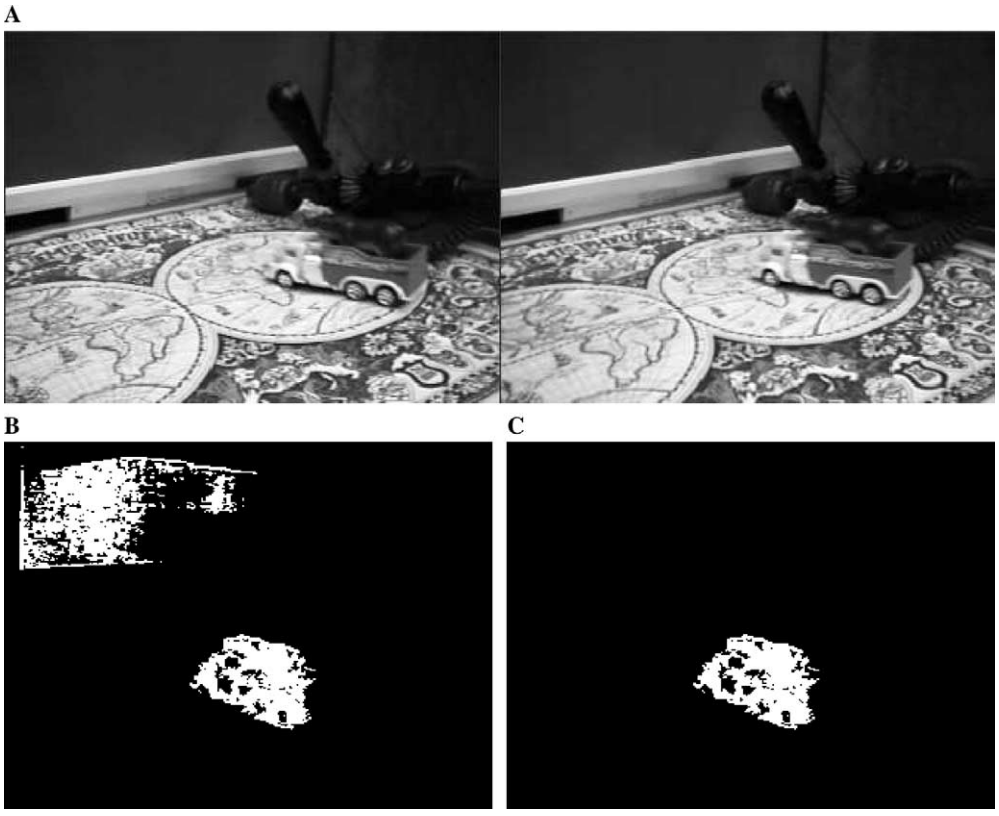

Fig. 1. Postprocessing. (A) Input image frames. (B) Before postprocessing. (C) After postprocessing.

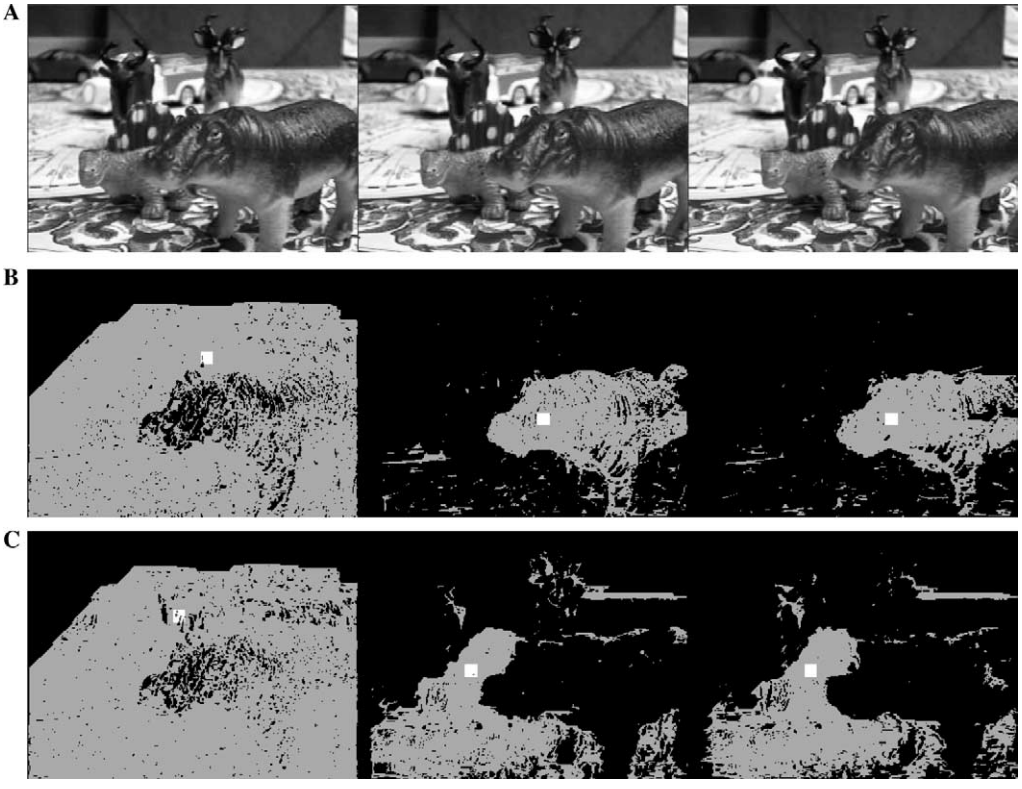

Fig. 2. The zoo sequence using the pixel statistic. (A) Original frames 5, 15, and 27. (B) Segments corresponding to the hippopotamus for frames 5, 15, and 27. (C) Segment corresponding to the background for frames 5, 15, and 27.

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

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

where $\mathbf{1}$ is the identity matrix, $\boldsymbol{D}_k = \mathrm{Diag}({}^N\vec{\hat{I}}_{N-1,d})$ is a $k^2 \times k^2$ diagonal matrix with each diagonal element equal to the corresponding element from the vector ${}^N\vec{\hat{I}}_{N-1,d}$ which is formed by a patch equal to $\vec{I}_x^2 + \vec{I}_y^2$ and $\boldsymbol{U}_k = [\vec{I}_x \vec{I}_y \vec{1}]$. The likelihood L is a compound function $L(C_k(\sigma_n^2,\ \sigma_a^2,\ \sigma_u^2,\ \sigma_v^2))$. We compute the maximum likelihood estimator of these parameters using Davidon–Fletcher–Powell (DFP) [33] algorithm given an initial guess of the values of these variances. DFP is a variable

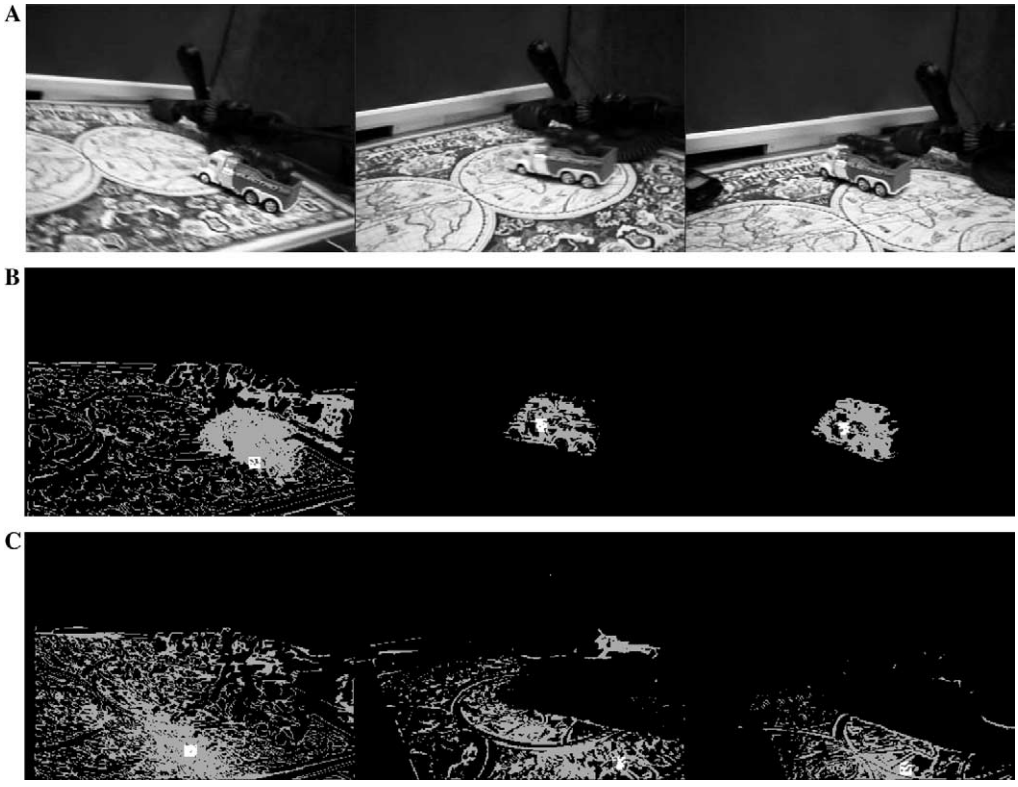

Fig. 3. The toy truck sequence with the hippopotamus using the pixel statistic. The sequence is taken by a hand-held camera that attempts to follow the truck. (A) Frames 1, 15, and 28 of the turning truck sequence. (B) Segment corresponding to the turning truck with seed region highlighted for frames 1, 15, and 28. (C) Segment corresponding to the moving background for frames 1, 15, and 28.

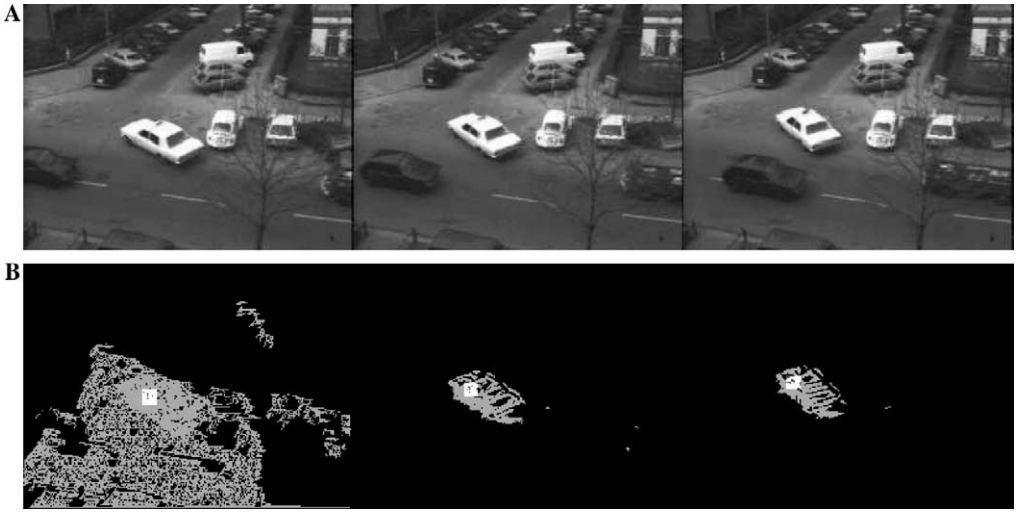

Fig. 4. The input image frames show the Hamburg taxi sequence using the pixel statistic. (A) Input sequence at frames 0, 9, and 18. (B) Segment corresponding to the turning taxi for frames 0, 9, and 18.

metric method for finding the minimum of the negative of the log likelihood and we use the following derivative formulae:

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

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

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

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

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.

6. Postprocessing

We apply postprocessing to ${}^{(N)}I_{\text{track}}$ in Eq. (10) or Eq. (19) 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 Fig. 1, we show the input image frames, the tracked regions before and after this postprocessing step.

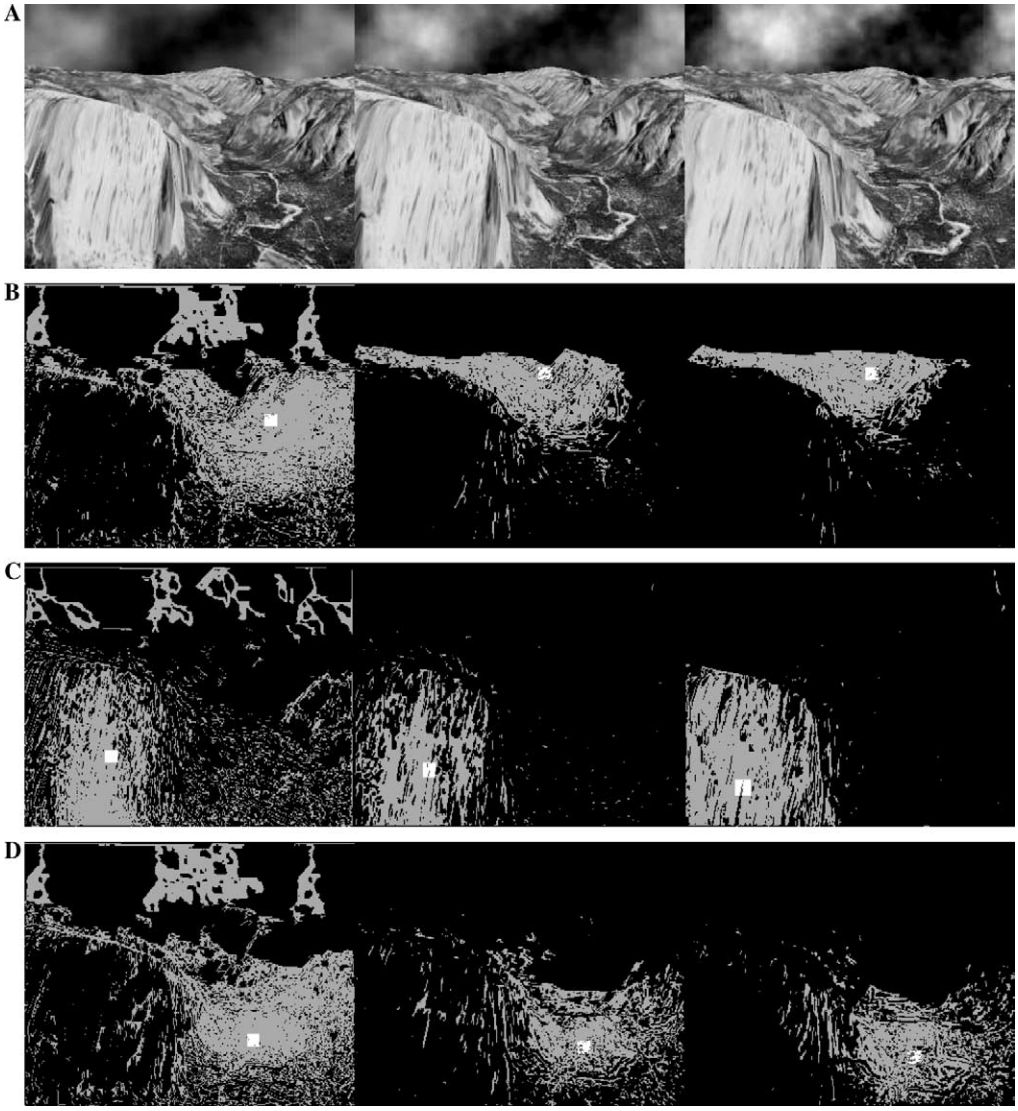

Fig. 5. The input image sequence show the Yosemite sequence using the pixel statistic. (A) Input sequences at frames 2, 8, and 15. (B) Segment corresponding to the distant mountain for frames 2, 8, and 15. (C) Segment corresponding to the left cliff for frames 2, 8, and 15. (D) Segment corresponding to the middle valley for frames frames 2, 8, and 15.

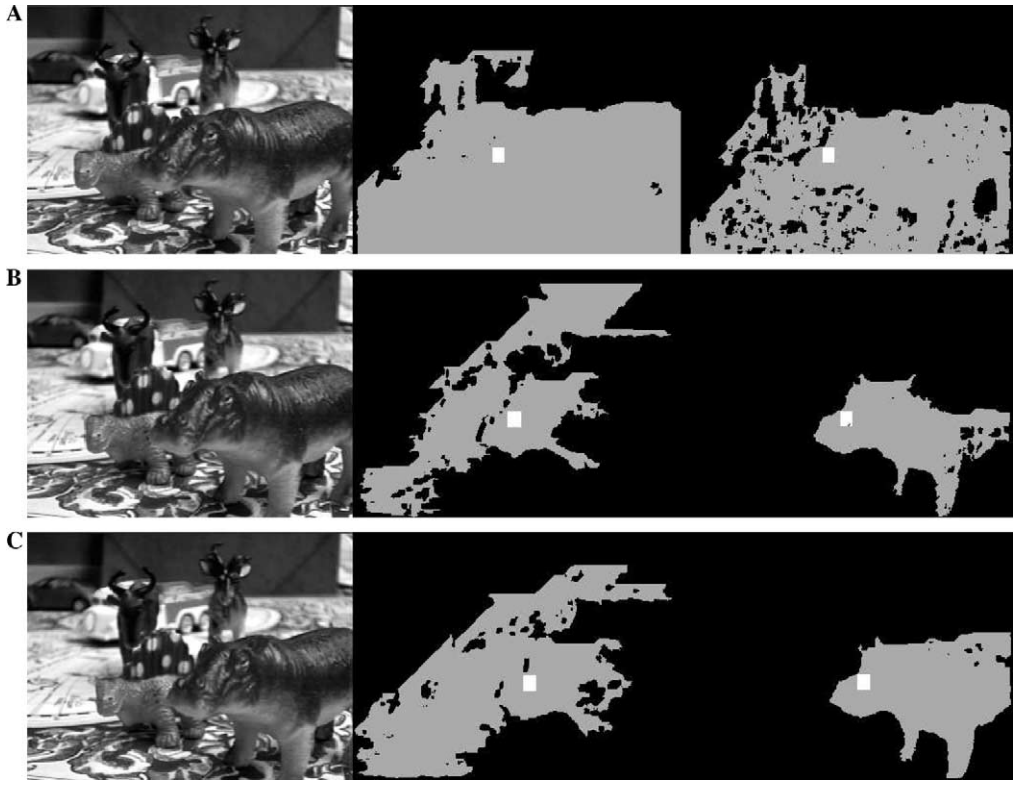

Fig. 6. The toy animal sequence using the patch statistic. The optimized parameters are: $\sigma_n = 0.97$, $\sigma_a = 0.0002$, $\sigma_u = 0.32$, $\sigma_v = 0.05$, $\sigma_l = 0.01$, $\sigma_f = 1.03$. (A) Frame 5 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$. (B) Frame 15 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$. (C) Frame 27 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$.

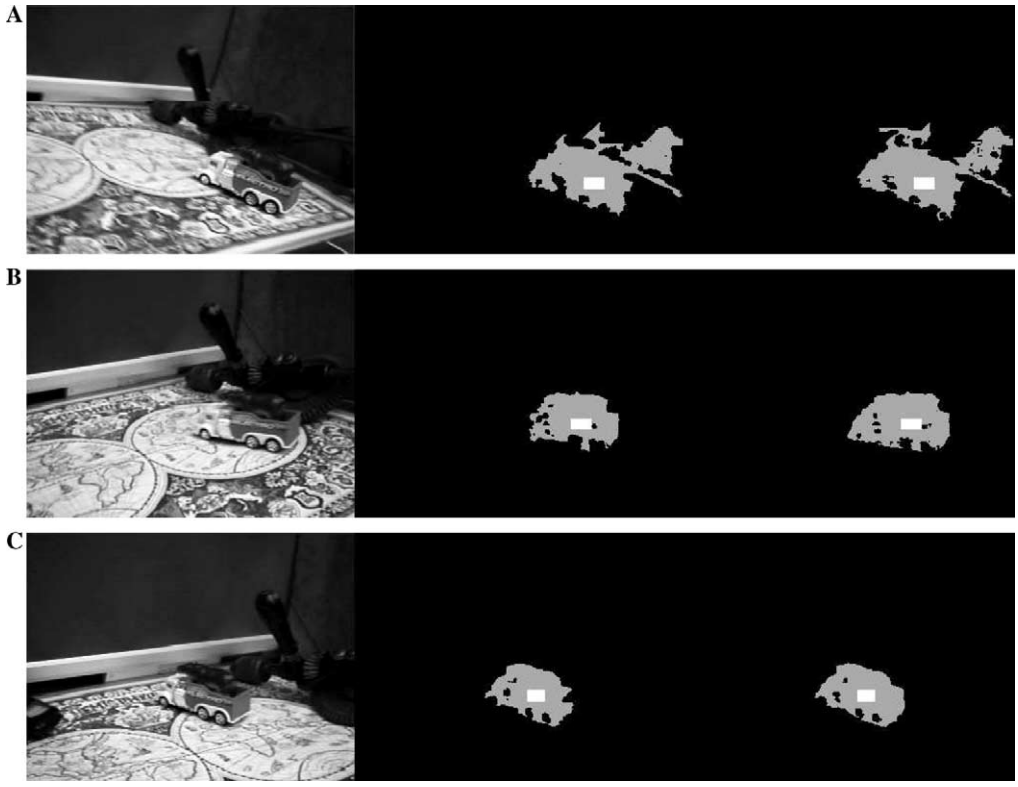

Fig. 7. The toy truck sequence with the hippopotamus using the patch statistic. The truck and the hippopotamus on top of it are segmented out. The wheels and the specularities on the engine block of the truck are not part of the segment. The optimized $\vec{\sigma}_0$ parameters are: $\sigma_n = 1.18$, $\sigma_a = 0.055$, $\sigma_u = 0.26$, $\sigma_v = 0.24$, $\sigma_l = 0.05$, $\sigma_f = 6.45$. (A) Frame 1 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$. (B) Frame 13 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$. (C) Frame 28 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$.

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

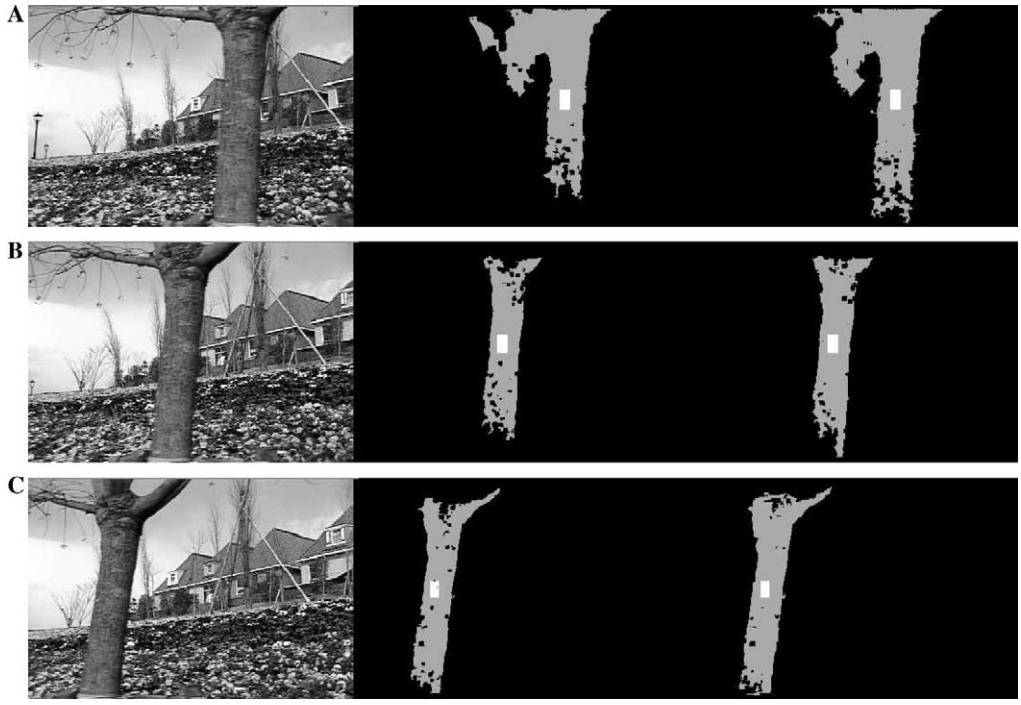

Fig. 8. Flower Garden sequence. The optimized $\vec{\sigma}_0$ parameters are: $\sigma_n = 1.28$, $\sigma_a = 0.23$, $\sigma_u = 0.128$, $\sigma_v = 0.359$, $\sigma_l = 0.0078$, $\sigma_f = 0.45$. (A) Frame 1 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$. (B) Frame 18 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$. (C) Frame 29 and its output from tracker using default $\vec{\sigma}_{\text{def}}$ and optimized $\vec{\sigma}_0$.

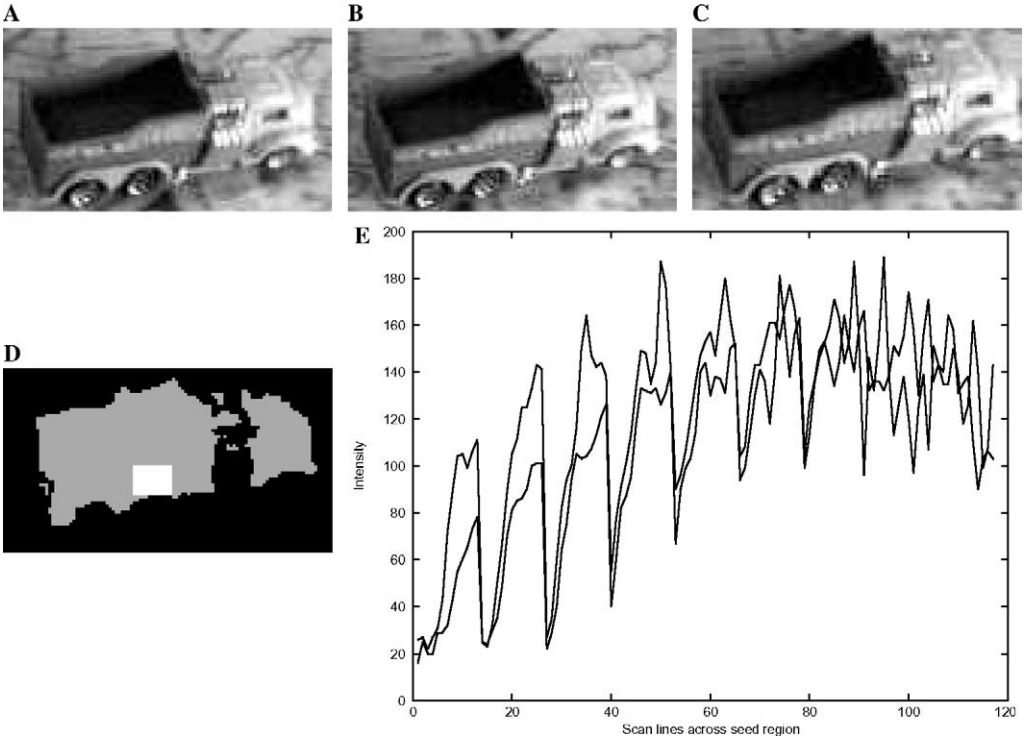

Fig. 9. The toy truck sequence with varying light. (A) Close-up of frame 9. (B) Close-up of frame 10. (C) Close-up of frame 12. (D) Output from tracker for frame 9 and 10. (E) Scan lines across seed region in frame 9 and 10.

6.1. 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:

$$\Delta I = \mathrm{Im}_N - \mathrm{Im}_{N-1}. \tag{20}$$

If there is no motion of any kind in successive frames, then ΔI 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 [27], a sample from a normal distribution $N(\mu,\sigma^2)$ has a probability of 0.9978 belonging in the range $\mu - 3\sigma$ to $\mu + 3\sigma$.

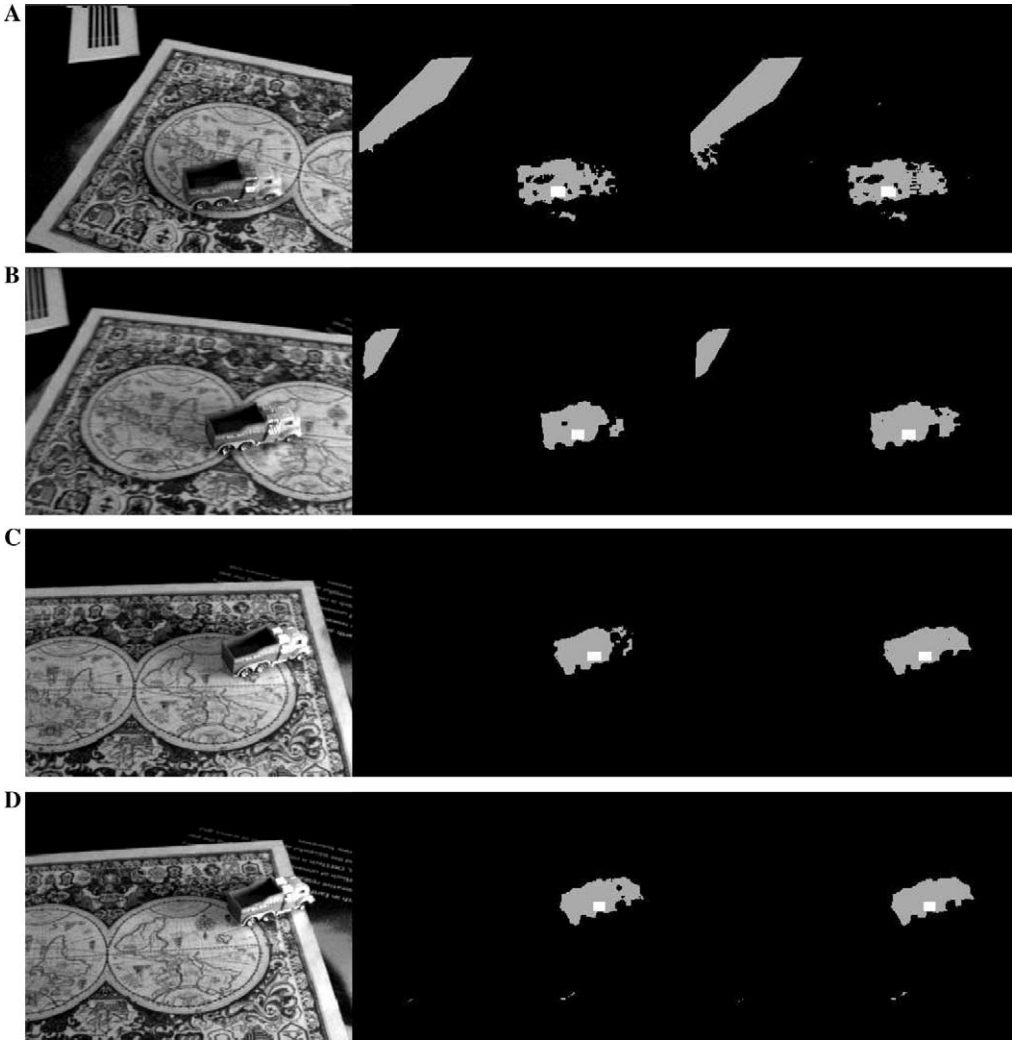

Fig. 10. Results using the toy truck sequence with varying light. The optimized $\vec{\sigma}_0$ parameters are: $\sigma_n = 2.585$, $\sigma_a = 0.0437$, $\sigma_u = 0.397$, $\sigma_v = 0.268$, $\sigma_l = 0.000002$, $\sigma_f = 2.245$. (A) Frame 1 and its output from tracker using default $\vec{\sigma}_{\mathrm{def}}$ and optimized $\vec{\sigma}_0$. (B) Frame 11 and its output from tracker using default $\vec{\sigma}_{\mathrm{def}}$ and optimized $\vec{\sigma}_0$. (C) Frame 31 and its output from tracker using default $\vec{\sigma}_{\mathrm{def}}$ and optimized $\vec{\sigma}_0$. (D) Frame 41 and its output from tracker using default $\vec{\sigma}_{\mathrm{def}}$ and optimized $\vec{\sigma}_0$.

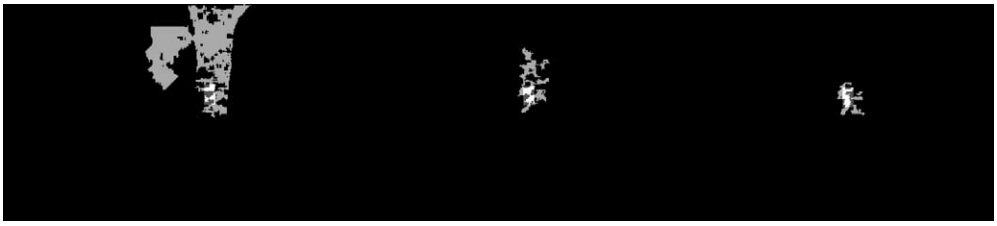

Fig. 11. Flower Garden sequence with the pixel-wise motion component disabled. As expected the performance degrades. The parameters are: $\sigma_n = 1.28$, $\sigma_a = 1e-6$, $\sigma_u = 0.128$, $\sigma_v = 0.359$, $\sigma_l = 0.0078$, $\sigma_f = 0.45$. Output from tracker for frames 1, 3, and 5 with σ_a close to zero.

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

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

6.2. Region growing

ΔIm contains patches for all moving objects in a pair of successive images. 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 ΔIm and ${}^{(N)}I_{\text{track}}$ and save the result as ${}^{(N)}I_{\text{filter}}$.

${}^{(N)}I_{\text{filter}} = \Delta\text{Im}$ and ${}^{(N)}I_{\text{track}}$.

One drawback of image difference technique in the detection of moving objects is that it can only capture moving regions with large image difference. However, a region can have a small ΔIm even if it is the projection of a moving object due to the aperture problem [19] or flat intensity. We address this shortcoming by extending ${}^{(N)}I_{\text{filter}}$ with ${}^{(N)}I_{\text{track}}$ using region growing and use the merged image ${}^{(N)}I_{\text{merge}}$ as the output of our tracking system

if ${}^{(N)}I_{\text{filter}}(x,y) == 1$ or ${}^{(N)}I_{\text{filter}}(x,y) == 0$ and ${}^{(N)}I_{\text{track}}(x_n,y_n) == 1$
then ${}^{(N)}I_{\text{merge}}(x,y) = 1$
else ${}^{(N)}I_{\text{merge}}(x,y) = 0$,

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

7. 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 interframe motion. By warping the image using the affine flow derived from our tracking algorithm and computing optical flow within one segment at a time, we avoid both the motion discontinuities and the large interframe motion. Although large interframe motion could be handled using 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 interframe 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 can be missed by most flow algorithms. 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 a standard algorithm like Lucas and Kanade [28] or Black and Anandan [5] to compute the residual flow. After that we combine residual flow with the affine flow to compute the total flow. Most flow algorithms work 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 segment. We re-

Fig. 12. Moving truck in a dynamic background under varying lighting conditions with the varying illumination components disabled. Again the performance degrades. The parameters are: $\sigma_n = 2.585$, $\sigma_a = 0.0437$, $\sigma_u = 0.397$, $\sigma_v = 0.268$, $\sigma_l = 1e-6$, $\sigma_f = 1e-6$. (A) Output from tracker for frames 1, 11, and 21 setting σ_l, σ_f close to zero. (B) Output from tracker for frames 31, 41 setting σ_l, σ_f close to zero.

port experiments on one synthetic and one real image sequence using this technique trying two different optical flow algorithms.

8. 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 the 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 does not satisfy the criteria of Section 3, 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 accu-

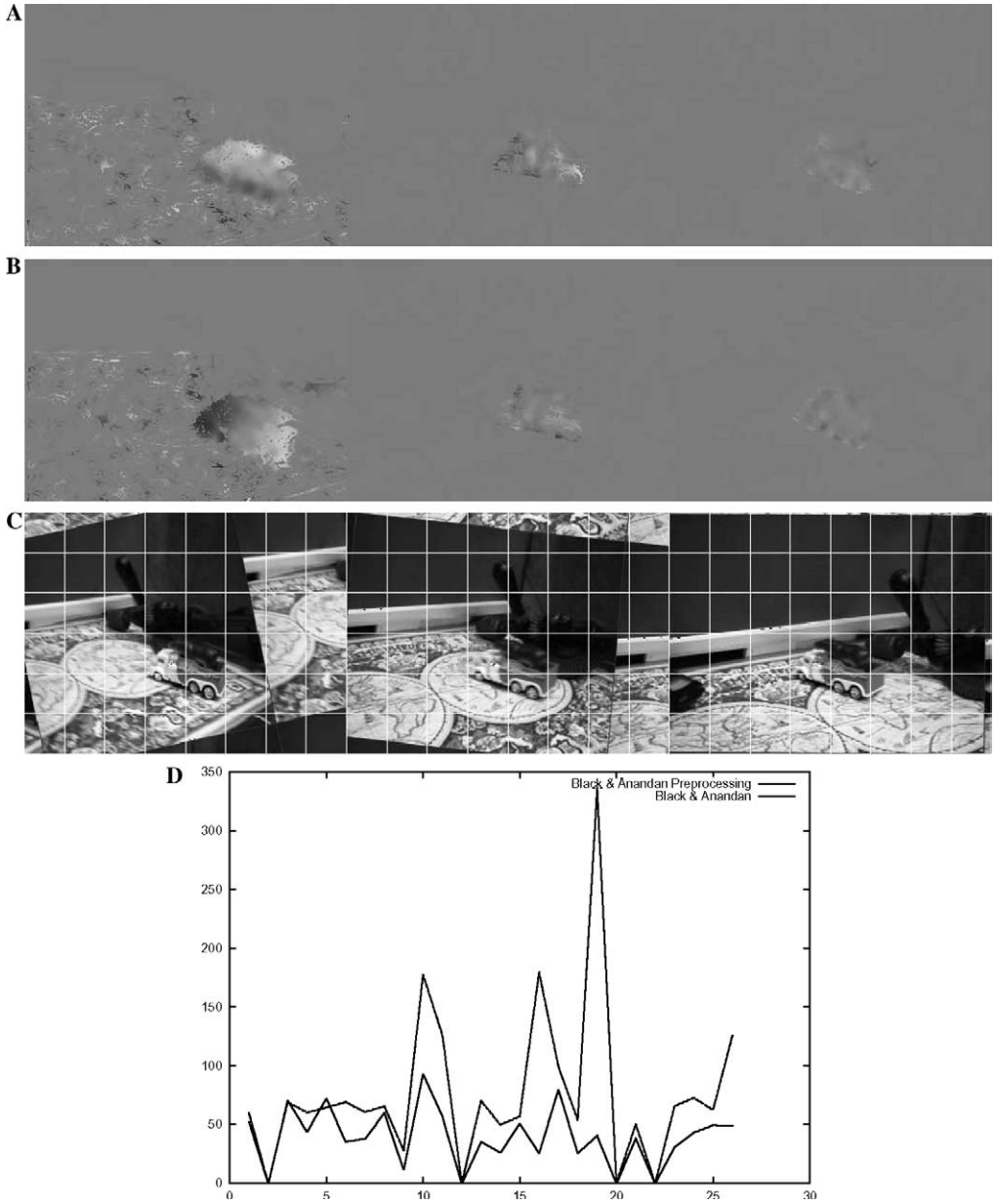

Fig. 13. Residual flow between aligned images on truck segment. (A) Residual horizontal flow as gray scale image for frames 1, 15, and 28. (B) Residual vertical flow as gray scale image for frames 1, 15, and 28. (C) Motion stabilized frames with grid for frames 1, 15, and 28. (D) SD between aligned images on the truck segment.

rate for all the sequences we tried with 25–115 frames. We need the experiments with manually selected seed regions to perform empirical analysis of the algorithm decoupled from the seed region selection method.

In all the experiments the quality of the segmentation improves in successive frames. The improvement is more pronounced when using the pixel statistic where the method needs to compute the first and second moments of the image and less so when using the patch statistic where the method integrates information temporally to reduce the effects of the noise.

8.1. Pixel statistic

In the first experiment (Fig. 2), we show the result of running our algorithm on the "zoo" sequence. The animals are on a large piece of paper, the camera is stationary and the paper is moved by hand. Two of the animals, the ram and the bison, 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 to track the motion of the segment as long as possible.

In the second experiment (Fig. 3), we show the result of running our algorithm on the toy truck sequence with the hippopotamus which shows a turning toy truck in a moving background. The image sequence is taken by a handheld camera that attempts to follow the motion of the toy truck. We show two segments computed by our algorithm corresponding to the truck and moving background. The seed regions used for tracking are generated randomly. The seed region that gave rise to the truck segment was replaced by another seed region within the same segment in the middle of the sequence when it stopped satisfying the criteria for tracking, i.e., the seed region was straddling a motion boundary.

In the third experiment (Fig. 4), 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.

In the fourth experiment (Fig. 5), we show the result of running our algorithm on the Yosemite sequence which is a synthetic fly through sequence from the Yosemite valley created by Lynn Quann at SRI. We show various segments produced by our algorithm using randomly selected seed regions as starting points.

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 [39]. For input image frames of size 320×240, our motion segmentation and tracking algorithm using pixel statistic takes 0.28 s per frame on the Pentium IV.

8.2. Patch statistic

All the experiments using the patch statistic the patch size is 5×5 pixels. We highlight the seed region in the out-

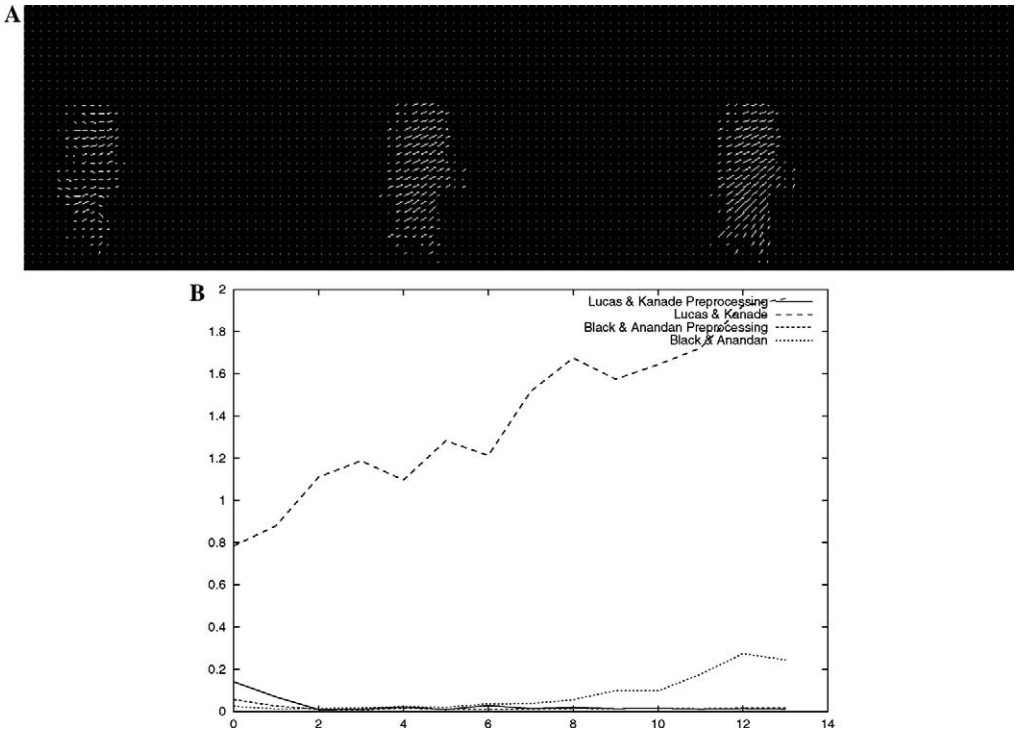

Fig. 14. Yosemite sequence needle map and its mean square flow error (u). (A) Needle maps of Lucas and Kanade, Lucas and Kanade with preprocessing and ground truth on the cliff segment. (B) Mean square flow error (u) on the left cliff segment compared with the ground truth.

put segmented images. The default (empirical) array of standard deviations $\vec{\sigma}_{\text{def}}$ of the various components of noise used in the individual experiments are: $\sigma_n = 2.75$, $\sigma_a = 0.08$, $\sigma_u = \sigma_v = 0.2$, $\sigma_l = 0.001$, $\sigma_f = 0.5$. The $\vec{\sigma}_{\text{def}}$ parameters were determined empirically and used in all sequences. We also computed a set of optimized parameters $\vec{\sigma}_0$ for each sequence using the maximum likelihood estimator (MLE) (Section 5.3) using a small number of seed regions within the tracked region of an image sequence. For all the sequences we run the algorithm with both the default parameters and the corresponding optimized ones.

In the fifth experiment (Fig. 6), we apply the patch statistic to the zoo sequence from the first experiment. The output segments are very similar to the segment obtained using pixel statistic. We show the result of tracking on the hippopotamus segment for comparison. The quality of the results improves in successive frames as the Mahalanobis distance is smoothed temporally. The quality of the segmentation using the optimized parameters is much higher than using the default ones.

In the sixth experiment (Fig. 7), we apply the patch statistic on the toy truck sequence from the second experiment. The output segments are similar to the segment obtained using pixel statistic but are much more accurate with fewer holes. We show the result of tracking on the truck for comparison. Most of the holes are on the hubs of the wheels and the specular engine block behind the cabin and neither the hubs nor the specularities on the engine block move with the same affine as the rest of the truck. The quality of the segmentation using the optimized and the default parameters was almost the same and although the optimal illumination change parameters were rather large, the method worked almost as well with significantly smaller parameters.

In the seventh experiment (Fig. 8), we run the algorithm using the patch statistic on the Flower Garden sequence [4] that has a tree in the foreground and a flower garden and houses in the background. The translating tree in the foreground of the flower garden is segmented out by tracking a seed region on the tree trunk. The results with the two sets of parameters are practically identical.

In the eighth experiment (Figs. 9 and 10), we use the toy truck sequence with varying light. The truck is moving diagonally from the lower left corner to the upper right corner and is taken by a panning camera. A table lamp is placed near the right 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 large change in intensity. The average change is roughly 15 gray level units (0–255 scale). In Fig. 9 we show a close-up of the truck for frames 9, 10, and 12, a close-up of the computed truck segment and a plot of the scanlines of the two images that shows the change in illumination. The optimal model parameters $\vec{\sigma}_0$ indicate that the change of illumination is mostly additive. The highly specular engine block is not part of the segment.

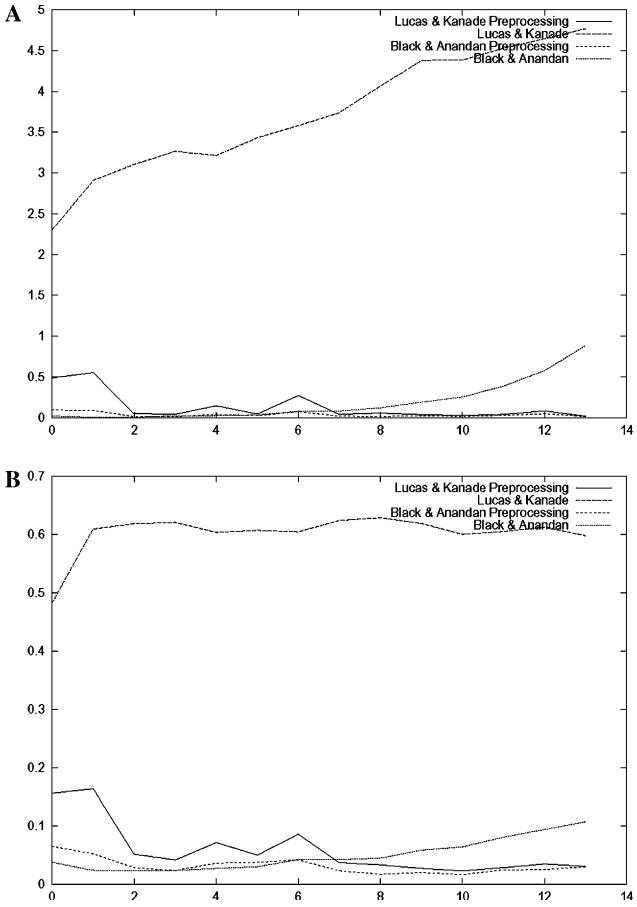

Fig. 15. Yosemite mean square flow error (v) and mean angular error (radian) with and without segmentation and alignment preprocessing. (A) Mean square flow error (v) on the left cliff segment compared with the ground truth. (B) Mean angular error (radian) on the left cliff segment compared with the ground truth.

We timed the execution our motion segmentation and tracking algorithm using patch statistic for images of size 320×240 pixels. It takes 1.32 s per frame on a Pentium IV 3.0 GHz PC compared with 0.28 s per frame using pixel statistic.

8.3. Significance of model components

The noise model we use for the patch statistic is rather complex, so the natural question to ask is if we need all the noise components. We address this question with two experiments. In the first experiment we also show the importance of the pixel-wise motion components and their associated parameter σ_a by disabling it[1] and rerunning the experiment. The resulting segments (Fig. 11) are clearly degraded with significant part of the tree missing. The tracking failed altogether after frame 5.

In the second experiment, we show the importance of the change of illumination component of the model and the asso-

[1] To disable them we set close to zero. We do not set them to exactly zero to avoid division by zero in the inversion of $\boldsymbol{C}_{\text{u}}$.

ciated parameters σ_l, σ_f by disabling them and rerunning the experiment. The resulting segments (Fig. 12) are again degraded with significant part of the truck missing.

Clearly the performance of the tracking and segmentation algorithm degrades if we omit components. When we were developing our model the pixel-wise motion component was the last one to add and without this the flower garden and the Yosemite sequence did not work. While the need for the varying illumination component was obvious, the need for the pixel-wise motion component became obvious only after analyzing our experiments.

8.4. Optical flow

One way to judge the quality of the results of our algorithm is to use them in an application. The obvious choice is optical flow, for the reasons we mentioned above, but also because there is substantial experience in evaluating optical flow algorithms.

In the first experiment, we use the toy truck sequence with the hippopotamus. We compute the residual horizontal and vertical flow between the stabilized images on the truck segment and show the results as a gray scale image (Fig. 13). The residual flow computed is small and never exceeded 2 pixels, and the mean square average over multiple frames is less than 0.5 pixels. We do not use needlemaps to display flow because they provide rather little accuracy for so small flow and small region. 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 toy truck remains stationary. We superimpose a grid on the stabilized image for reference. 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.

Since there is no ground truth available for this real image sequence we quantify the accuracy of the optical flow computed, by evaluating the squared differences (SD) between the previous image frame warped by the computed flow and the current frame. This allows us to compare two different versions of an optical flow algorithm and although small differences in the SD do not necessarily determine the relative performance, large differences are very indicative. For this experiment we use the hierarchical robust flow by Black[2] and Anandan [5] with and without preprocessing the images using our algorithm and compare the results.

In the preprocessing step, we employ the motion segmentation and tracking algorithm to compute the affine motion of the truck segment and use this motion to warp one of the two images to reduce the large interframe motion. Then we show the graph of the SD for the results of Black and Anandan with and without the preprocessing. The peaks of the SD for Black and Anandan without the preprocessing correspond to interframe motion of about 10 pixels near the tracked region. The truck is about 30×90 pixels.

In the previous experiment, the sequence was particularly difficult for any algorithm that uses a hierarchical scheme to handle large interframe motion because the interframe motion is large and the object is relatively small. For the second experiment we used Lynn Quann's Yosemite sequence which has ground truth. The segments are relatively large and the interframe motion is not excessive so it is well suited for hierarchical schemes. We computed the optical flow on one of the segments (left cliff) of the Yosemite sequence with and without the preprocessing (Figs. 14 and 15). We compared the accuracy of the optical flow computed on this segment using four methods: multi-stage Lucas and Kanade with warping after the each step [28] as well as hierarchical robust flow by Black and Anandan [5], both with and without preprocessing by our tracking and segmentation algorithm. We evaluate the accuracy of the computed optical flow on these segments by comparing the mean square flow error and the mean angular error [3] for one segment using the above methods with the ground truth. The use of preprocessing offers a substantial improvement for the Lucas and Kanade algorithm. The Black and Anandan algorithm saw no real improvement and the two versions were within a couple of tenths of a pixel from each other. It is worth noticing that Lucas and Kanade with preprocessing is comparable to Black and Anandan with or without preprocessing.

9. Conclusion

We described a novel and efficient algorithm that performs motion segmentation and tracking using corner features. The core of the algorithm are two statistics that can be used to determine if a small patch is being tracked correctly. The first is a pixel based statistic that computes a temporal average of the squared differences of aligned pairs of images. It is fast and quite accurate but cannot handle highly correlated noise. The second is a patch based statistic that computes the Mahalanobis distance between two aligned images and can handle correlated noise. Since direct computation of the Mahalanobis distance is prohibitively expensive, we developed an exact algorithm that provides speedup of several orders of magnitude using the SMW identity and amortization. We used a noise model that has seven parameters, but the method can be easily adapted for different models. We estimated the parameters of the model using maximum likelihood estimation with a small number of samples. The patch based statistic proved very flexible and robust in the presence of correlated noise.

Beyond the design of the tracking and segmentation algorithm, this paper includes two more contributions. The first contributions is the treatment of correlated noise based on a model with a minimal number of approximations. Indeed the only place that our model is approximate is Eq. (12) in Section 5.2, which is a first order approximation commonly used in the optical flow literature. This

[2] We are grateful to the authors of Ref. [5] for letting us use their code.

model is a definite improvement over the standard sum of squared differences which, while fine for sharp and distinct feature points, it is inadequate for the rest of the image. In the future we intend to apply the same technique to other problems in computer vision.

The other contribution is that we provided an alternative to hierarchical optical flow estimation. Our algorithm can handle very large interframe motion since it tracks using search and as a result it can compute an affine approximation of the flow and warp one of the two images to reduce interframe motion. Using our algorithm as preprocessing, flow can be computed in images with small objects undergoing large interframe motion.

We 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.

Appendix A

In this appendix, we show how to compute products like $\vec{I}_x^{\,T}\boldsymbol{C}_n^{-1}\vec{I}_y$ in constant time with respect to k using amortization, i.e., to compute the above product for one particular pixel we reuse as much as possible the intermediate results of the computation for the neighboring pixels.

Let the x, y derivatives of I be the images I_x, I_y respectively. The patches are centered at a point $[x,y]$ and each dimension of the patch is k which is usually an odd number. Since $\boldsymbol{C}_n$ is a diagonal matrix

$$\boldsymbol{C}_n[i,j] = 0 \quad \text{if } i \neq j$$

and

$$\boldsymbol{C}_n[i,i] = \sigma_n^2 + \sigma_a^2(I_x^2[y - \lfloor k/2\rfloor + \lfloor i/k\rfloor, x - \lfloor k/2\rfloor + i \bmod k] + I_y^2[y - \lfloor k/2\rfloor + \lfloor i/k\rfloor, x - \lfloor k/2\rfloor + i \bmod k]),$$

where $0 \leqslant i,j \leqslant k^2$. We assume 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

$$\boldsymbol{C}_n[i,i] = I_n[y - \lfloor k/2\rfloor + \lfloor i/k\rfloor, x - \lfloor k/2\rfloor + i \bmod k]$$

and

$$\boldsymbol{C}_n^{-1}[i,i] = \frac{1}{I_n[y - \lfloor k/2\rfloor + \lfloor i/k\rfloor, x - \lfloor k/2\rfloor + i \bmod k]}.$$

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

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

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

$$\vec{I}_x^{\,T}\boldsymbol{C}_n^{-1}\vec{I}_y = \sum_i \sum_j I_x[i]\,\boldsymbol{C}_n^{-1}[i,j]\,I_y[j] = \sum_i \vec{I}_x[i]\,\boldsymbol{C}_n^{-1}[i,i]\,\vec{I}_y[i]$$
$$= \sum_{m=0}^{k}\sum_{n=0}^{k} I_x[y_o - m, x_o - n] I_n^{-1}[y_o - m, x_o - n] I_y \times [y_o - m, x_o - n],$$

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

$$\vec{I}_x\boldsymbol{C}_n^{-1}\vec{I}_y = \sum_{m=0}^{k}\sum_{n=0}^{k} I_x[y_o - m, x_o - n] I_n^{-1}[y_o - m, x_o - n] I_y \times [y_o - m, x_o - n]$$
$$= \sum_{m=0}^{k}\sum_{n=0}^{k} I_{xny}[y_o - m, x_o - n],$$

where $I_{xny} = \frac{I_x I_y}{I_n}$. 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 $\vec{I}_x\boldsymbol{C}_n^{-1}\vec{I}_y$ 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 products and calculate the elements of matrix $\boldsymbol{C}_s$. There is a constant number of such products and since each takes a constant amount of time to compute the amount of computation per pixel is also constant independent of patch size.

References

[1] P. Anandan, A computational framework and an algorithm for the measurement of visual motion, Int. J. Comput. Vision 2 (1989) 283–310.

[2] S. Ayer, H. Sawhney. Layered representation of motion video using robust maximum likelihood estimation of mixture models and MDL encoding, in: Proceedings of Fifth International Conference on Computer Vision, 1995, pp. 777–784.

[3] J.L. Barron, D.J. Fleet, S.S. Beauchernin, Performance of optical flow techniques, Int. J. Comput. Vision (1994) 43–77.

[4] M. Black. Michael J black: Image sequences. http://www.cs.brown.edu/people/black/images.html, 2002.

[5] M. Black, P. Anandan, A framework for the robust estimation of optical flow, in: Fourth International Conference on Computer Vision, 1993, pp. 231–236.

[6] M. Black, P. Anandan, The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields, Comput. Vision Image Understand. 63 (1) (1996) 75–104.

[7] M. Black, A. Jepson, Eigentracking: robust matching and tracking of articulated objects using a view-based representation, Int. J. Comput. Vision 26 (1) (1998) 63–84.

[8] G.D. Borshukov, G. Bozdagi, Y. Altunbasak, A.M. Tekalp, Motion segmentation by multistage affine classification, IEEE Trans. Image Process. 6 (1997) 1591–1594.

[9] P. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. Lee, A. Leung, J. Lubin, H. Shvaytser, Object tracking with a moving camera, in: Proceedings of Workshop on Visual Motion, 1989, pp. 2–12.

[10] D. Comaniciu, V. Ramesh, P. Meer, Real-time tracking of non-rigid objects using mean shift, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, 2000 pp.142–149.

[11] D. Cremers, S. Soatto. Variational space-time motion segmentation, in: Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003, pp. 886–893.
[12] B.S. Everitt, Cluster Analysis, Wiley, New York, 1974.
[13] B. Frey, N. Jojic. Estimating mixture models of images and inferring spatial transformation using the em algorithm, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1999, pp. 416–422.
[14] B. Frey, N. Jojic, A. Kannan, Learning appearance and transparency manifolds of occluded objects in layers, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2003.
[15] A. Gruber, Y. Weiss, Multibody factorization with uncertainty and missing data using the em algorithm, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2003, pp. 707–714.
[16] G. Hager, P. Belhumeur, Efficient region tracking with parametric models of geometry and illumination, IEEE Trans. Pattern Anal. Mach. Intell. (1998) 1025–1039.
[17] W.W. Hager, Updating the inverse of a matrix, SIAM Rev. 31 (1989) 221–239.
[18] J. Heikkila, P. Sangi, O. Silven, Camera motion estimation from non-stationary scenes using EM-based motion segmentation, in: Proceedings of 15th International Conference on Pattern Recognition, vol. 1, 2000, pp. 370–374.
[19] B.K.P. Horn, Robot Vision, McGraw-Hill Book Company, New York, 1986.
[20] B.K.P. Horn, B.G. Schunck, Determining optical flow—a retrospective, Artif. Intell. 59 (1993) 81–87.
[21] M. Irani, B. Rousso, S. Peleg, Computing occluding and transparent motions, Int. J. Comput. Vision 12 (1994) 5–16.
[22] M. Isard, A. Blake, Contour tracking by stochastic propagation of condition density, in: Proceedings of European Conference on Computer Vision, 1996, pp. 343–356.
[23] M. Isard, 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.
[24] A. Jepson, M. Black, Mixture models for optical flow computation, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1993, pp. 760–761.
[25] A. Jepson, D. Fleet, T. F. El-Maraghi. Robust online appearance models for visual tracking, in: IEEE Conf. on Computer Vision and Pattern Recognition, vol. 1, 2001, pp. 415–422.
[26] Q. Ke, T. Kanade, A subspace approach to layer extraction, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 1, 2001, pp. 8–14.
[27] R. Larsen, M. Marx, Introduction to Mathematical Statistics and its Applications, Prentice Hall, Englewood Cliffs, NJ, 1986.
[28] B.D. Lucas, T. Kanade, An iterative technique of image registration and its application to stereo, in: Proc. 7th Internat. Joint Conf. on Artificial Intelligence, 1981, pp. 674–679.
[29] P.C. Mahalanobis, On tests and measures of groups divergence i, J. Asiatic Soc. Benagal (1930).
[30] A. Mansouri, J. Konrad, Multiple motion segmentation with level sets, IEEE Trans. Image Process. (2003) 201–220.
[31] F. Meyer, P. Bouthemy, Region-based tracking using affine motion models in long image sequences, Comput. Vision Graph. Image Process. (1994) 119–140.
[32] S. Negahdaripour, C. Yu, A generalized brightness change model for computing optical flow, in: Proceedings of Fourth International Conference on Computer Vision, 1993, pp. 2–11.
[33] W. Press, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, 1992.
[34] J. Sherman, 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. (1949) 20.
[35] H. Sidenbladh, M. Black, D. Fleet, Stochastic tracking of 3d human figures using 2d image motion, in: Proc. Eur. Conf. Computer Vision, 2000, pp. 702–718.
[36] E.P. Simoncelli, E.H. Adelson, D.J. Heeger, Probability distributions of optical flow, Proc. Comput. Vision Pattern Recognit. (1991) 310–315.
[37] P. Smith, T. Drummond, R. Cipotia, Layered motion segmentation and depth ordering by tracking edges, IEEE Trans. Pattern Anal. Mach. Intell. (2004) 479–494.
[38] H. Spies, B. Jahne, J. L. Barron, Regularised range flow, in: European Conference on Computer Vision, vol. 2, 2002, 785 –799.
[39] Standard. Submitted cpu2000 results. http://www.spec.org/osg/cpu2000/results/cint2000.html, 2003.
[40] H. Stark, J.W. Woods, Probability and Random Processes with Applications to Signal Processing, Prentice Hall, Englewoods Cliffs, NJ, 2002.
[41] A. Strehl, J. K. Aggarwal, A new bayesian relaxation framework for the estimation and segmentation of multiple motions, in: Proc. 4th IEEE Symposium on Image Analysis and Interpretation, 2000.
[42] H. Tao, H. S. Sawhney, R. Kumar, Dynamic layer representation with applications to tracking, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2000, pp. 2134–2141.
[43] C. Tomasi, T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.
[44] N. Vasconcelos, A. Lippman, Empirical bayesian EM-based motion segmentation, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1997, pp. 527–532.
[45] J.Y.A. Wang, E.H. Adelson, Representing moving images with layers, IEEE Trans. Image Process. 3 (1994) 625–638.
[46] J. Wills, S. Agarwal, S. Belongie, What went where, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2003, pp. 37–44.
[47] K.Y. Wong, M.E. Spetsakis, Motion segmentation and tracking, in: Proc. 15th Internat. Conf. on Vision Interface, 2002, pp. 80–87.
[48] K.Y. Wong, M.E. Spetsakis, Tracking, segmentation and optical flow, in: Proc. 16th Internat. Conf. on Vision Interface, 2003, pp. 57–64.
[49] K.Y. Wong, M.E. Spetsakis, Motion segmentation by EM clustering of good features, in: Proceedings of Second IEEE Workshop on Image and Video Registration, 2004.
[50] K.Y. Wong, L. Ye, M.E. Spetsakis, EM clustering of incomplete data applied to motion segmentation, in: Proc. British Mach. Vision Conference, 2004, pp. 237–246.
[51] M. Woodbury, Inverting Modified Matrices, Memorandum Rept. 42, Statistical Research Group, Princeton University, NJ, 1950.
[52] Y. Ye, J. Tsotsos, E. Harley, K. Bennet, Tracking a person with pre-recorded image database and a pan, tilt and zoom camera, Mach. Vision Appl. 12 (2000) 32–43.
[53] T. Balch. Z. Khan, F. Dellaert, A rao-blackwellized particle filter for eigentracking, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2004, pp. 980–986.
[54] L. Zelnik-Manor, Micchal Irani, Multi-view subspace constraints on homography, in: Proc. Seventh IEEE Internat. Conf. on Computer Vision, vol. 2, 1999, pp. 710 – 715.