Motion Segmentation and Tracking

King Yuen Wong
Minas E. Spetsakis
Dept. of Computer Science
York University
4700 Keele Street
Toronto, ONTARIO
CANADA, M3J 1P3
Email: kywong@cs.yorku.ca, minas@cs.yorku.ca

Abstract

This paper presents a novel tracking based motion segmentation algorithm. The tracking is done by fitting successively more elaborate models of optical flow on the tracked region and the segmentation is done by extracting the regions of the image that are consistent with the computed model of flow. The method can track objects in image sequences with moving background, taken by a hand-held camera, tolerate up to 30 pixels interframe motion and takes 0.3 seconds per frame pair of size 320 x 240 pixels on a 500 Mhz Sun Blade 100 workstation.

Keywords: Motion Segmentation, Tracking, Optical Flow

1. Related Work

A tracking algorithm measures and predicts the motion of a moving object over time. Contours [8, 13] corresponding to the silhouette of moving objects are commonly used feature for tracking. The coherence of a moving region [5, 1] corresponding to the projection of a surface of the moving object is another good basis for tracking. Color [3, 9] of a moving object is also frequently used in tracking. 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 image difference on the image frames with known background [15]. 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 [11] 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.

Burt [2] identified and tracked the object by successively computing optical flow on a region, fitting an affine model on the computed flow, detecting outliers and warping the images. In a sense they fit two models on the optical flow data. One for the object and one for the background.

2. Algorithm

Our motion segmentation and tracking system automatically segments and tracks a region corresponding to the projection of a moving object given its input seed window. For every successive pair of image frames, the tracking stage fits successively more elaborate optical flow models on the tracked seed window and the segmentation stage detects the pixels whose changing intensity patterns are consistent with the model of optical flow. This is done by aligning all previous frames to the last one, computing a sum of squared differences statistic and thresholding. Aligning all previous frames is a computation intensive task, so in order to speed up the computation, we propose a statistic that encodes the history of the intensity pattern. We also propose a method to compute the per pixel threshold that takes into account the local image variation, the anticipated camera noise and the desired goodness of fit.

2.1. Input to system

The main input to the system is a window for the first image frame of the image sequence representing a region within the to-be-tracked object and its maximum interframe motion in pixels. The input window is the projection of a surface within the tracked object and hence its boundary must be inside the moving object. In our experiments, we use seed window of size 10×10 pixels. A small seed

The support of NSERC (App. No. OGP0046645) and CITO (Communications and Information Technology Ontario) is gratefully acknowledged.

window is preferred because the flow in a small window can be easily considered uniform. It also allows us to search a bigger area for possible matches and still run the whole algorithm in about 0.3 seconds.

2.2. Optical flow computation

We compute the optical flow of the region we track in two stages. First, we compute integer/subpixel optical flow on the tracked seed window using least sum of squared differences. Second, we perform affine flow computation on the motion segmented region derived from integer/subpixel flow.

2.2.1. Integer optical flow computation

For every successive pair of image frames (Im_{N-1}, Im_N), we compute integer optical flow, u and v, for the tracked window using Least Sum of Squared Differences (LSSD).

$$LSSD\,(u,v) = \min_{(u,v)} \sum_{\substack{x_{\min}..x_{\max} \\ y_{\min}..y_{\max}}} (\mathrm{Im}_{N-1}(y+v, x+u) - \mathrm{Im}_N(y,x))^2$$

using straight search. Since the seed region is small, we can search a large area like $-30..30$ in each dimension.

2.2.2. Subpixel optical flow computation

We compute subpixel optical flow for the tracked window by first aligning Im_{N-1} with the optical flow obtained from the integer optical flow step. We perform LSSD between Im_N and the shifted Im_{N-1}. We apply subpixel shifts ($adjU$, $adjV$) in 9 directions, namely NW, N, NE, W, E, SW, S, SE, (0, 0) and compute their SSDs. For example, in the NW direction, $adjU = -subpixel, adjV = -subpixel$. After that, we shift Im_{N-1} by the subpixel flow that yields the minimal SSD. The above procedure is repeated for successively smaller amount of subpixel shifts. We found empirically the best sequence of values for *subpixel* is $0.75, 0.75^2, 0.75^3 \cdots$. Although shifting by $0.5, 0.5^2, \cdots$ seems more reasonable, in our experiments we get better results by using the $.75, .75^2 \cdots$ sequence of steps. Many other algorithms can do at least as good a job but this one gives us a bound on subpixel accuracy.

2.2.3. Affine flow computation

In order to account for rotational and perspective motion, we compute affine optical flow. Under the affine flow model, the optical flow (u, v) at each point (x,y) is a function of six affine motion parameters (u_0 , u_x, u_y, v_0, v_x, v_y):

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \qquad (2.1)$$

We use the output of the constant flow model to do a preliminary segmentation (see next section) to decide on which region to apply the affine flow computation. We grow the original seed window a constant number of times (4 in our experiments) and AND it with the segmented region.

We compute the six affine motion parameters by minimizing the SSDs for all pixels belonging to the region. For any successive pair of image frames im_{N-1}, im_N,

$$SSD = \sum_{all\ x,y} {}^{(N-1)}I_{track}(x,y) \cdot [\mathrm{Im}_{N-1}^{(u,v)}(x,y) - \mathrm{Im}_N(x,y)]^2$$

where ${}^{(N-1)}I_{track}$ represents the region on which we compute affine optical flow, $\mathrm{Im}_{N-1}^{(u,v)}$ is Im_{N-1} shifted by (u, v). Using a differential approach, as in [7], Im_{N-1} can be expressed in terms of Im_N and its x, y derivatives

$$\mathrm{Im}_{N-1}^{(u,v)}(x,y) \approx \mathrm{Im}_{N-1}(x,y) + \mathrm{Im}_{N-1,x}(x,y) \cdot u + \mathrm{Im}_{N-1,y}(x,y) \cdot v$$

Using this expression for Im_{N-1} and Eq. (2.1), the SSD can be expressed as follows:

$$\begin{aligned} SSD = \sum_{all\ x,y} {}^{(N-1)}I_{track}[&\Delta \mathrm{Im}_N(x,y) \\ &+ \mathrm{Im}_{N-1,x}(x,y)u_0 + \mathrm{Im}_{N-1,x}(x,y)xu_x + \mathrm{Im}_{N-1,x}(x,y)yu_y \\ &+ \mathrm{Im}_{N-1,y}(x,y)v_0 + \mathrm{Im}_{N-1,y}(x,y)xv_x + \mathrm{Im}_{N-1,y}(x,y)yv_y]^2 \end{aligned} \qquad (2.2)$$

and

$$\Delta \mathrm{Im}_N(x,y) = \mathrm{Im}_{N-1}[x,y] - \mathrm{Im}_N[x,y]$$

We apply standard least squares to solve for the six affine parameters by deriving and solving the normal equation from Eq. (2.2).

One of the hardest problems related to the differential techniques is taking derivatives both spatial and temporal. 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 areas of the image which is consistent with small flow and do least squares on a rather large area.

2.3. Segmentation of the tracked object

We segment out the tracked object by thresholding the sum of the squared differences between the last image and all the images aligned with the last image. Regions corresponding to the projection of the tracked object should have small intensity difference between the aligned images and the last image. Therefore, we classify those regions with small intensity difference as belonging to the tracked object. In the next section we show how to do it efficiently and how to set the thresholds.

2.3.1. Statistics of the tracked object

Consider the following summation,

$$\sum_{i=1}^{N-1} ({}^{(N)}\mathrm{Im}_i - \mathrm{Im}_N)^2 \tag{2.3}$$

where Im_i is the ith image, for $i = 1..N$ and ${}^{(N)}\mathrm{Im}_i$ is the ith image aligned with the Nth image using the optical flow information from the tracking. The summation in Eq. (2.3) is the pixelwise SSD between the last image Im_N and the previous images ${}^{(N)}\mathrm{Im}_i$ 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. So it is a simple matter of thresholding to segment out the region that is tracked correctly. But there are two problems to solve: how to compute the statistics and how to select the threshold.

2.3.2. Computing the statistics

Computing Eq. (2.3) 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 would need a more efficient way of computing Eq. (2.3). Expanding it

$$\sum_{i=1}^{N-1} ({}^{(N)}\mathrm{Im}_i - \mathrm{Im}_N)^2 = \tag{2.4}$$

$$\sum_{i=1}^{N-1} {}^{(N)}\mathrm{Im}_i{}^2 - 2\Big(\sum_{i=1}^{N-1} {}^{(N)}\mathrm{Im}_i\Big)\mathrm{Im}_N + \mathrm{Im}_N^2(N-1)$$

Dividing equation (2.4) by $N-1$ gives:

$$\frac{\sum_{i=1}^{N-1} ({}^{(N)}\mathrm{Im}_i - \mathrm{Im}_N)^2}{N-1} = \tag{2.5}$$

$${}^{(N)}\hat{\mu}_2 - 2\,{}^{(N)}\hat{\mu}_1\,\mathrm{Im}_N + \mathrm{Im}_N^2 = t_{stat}$$

where $\hat{\mu}_1$ and $\hat{\mu}_2$ are the first and second moments of the images and the left superscript ${}^{(N)}$ means aligned with the last image. In this way, we only need to store and align two statistics ($\hat{\mu}_1$, $\hat{\mu}_2$) using the computed optical flow for every tracking step. However, Eq. (2.5) still has a major shortcoming, namely, if we compute optical flow incorrectly in any tracking step, the mistake would be incorporated into the tracked region and haunt us forever. We address this problem by using a recursive expression that lets the influence of the past images decay exponentially into our tracking statistics. Specifically, for each successive pair of image frames Im_{N-1}, Im_N, we compute the first moment (${}^{(N)}\hat{\mu}_1$) of the tracked object as a weighted sum of the translated first moment ${}^{(N)}\hat{\mu}_1^t$ and Im_N.

$${}^{(N)}\hat{\mu}_1 = \alpha \cdot {}^{(N)}\hat{\mu}_1^t + (1-\alpha)\cdot \mathrm{Im}_N \tag{2.6}$$

and

$${}^{(N)}\hat{\mu}_1^t = \mathit{shift}\ {}^{(N-1)}\hat{\mu}_1\ \mathit{by}\ (u, v) \tag{2.7}$$

where (u, v) is the optical flow computed from the tracked moving region using the $N-1$ and N image frames. The base case for ${}^{(1)}\hat{\mu}_1^t$ is:

$${}^{(1)}\hat{\mu}_1^t = \mathrm{Im}_1$$

Parameter α is a decimal number between 0 and 1.0 that represents the relative weighting between the projected trajectory of the moving object based on history and the information from the current frame. A high value of α (for example > 0.8) places more emphasis on the projected trajectory based on history whereas a low value of α suggests an almost memoryless motion trajectory model. In other words, we can balance historical and current information by varying α. In our experiments $\alpha = 0.8$.

Similarly, we store the second moment (${}^{(N)}\mu_2$) of the tracked object as a weighted sum of the squared of ${}^{(N)}\hat{\mu}_2^t$ and Im_N.

$${}^{(N)}\hat{\mu}_2 = \alpha \cdot {}^{(N)}\hat{\mu}_2^t \cdot {}^{(N)}\hat{\mu}_2^t + (1-\alpha)\cdot \mathrm{Im}_N \cdot \mathrm{Im}_N \tag{2.8}$$

and

$${}^{(N)}\hat{\mu}_2^t = \mathit{shift}\ {}^{(N-1)}\hat{\mu}_2\ \mathit{by}\ (u, v) \tag{2.9}$$

The base case for ${}^{(1)}\hat{\mu}_2^t$ is:

$${}^{(1)}\hat{\mu}_2^t = \mathrm{Im}_1 \cdot \mathrm{Im}_1$$

The *shift* operation on the statistics is done using warping with quadratic interpolation.

2.3.3. Computation of threshold

We use pixelwise thresholding to identify pixels belonging to the tracked object. The pixels whose tracking statistics in Eq. (2.5) 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 statistics given by Eq. (2.5) 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. (2.5) as follows:

$$t_{value} = (\sigma_c^2 + \sigma_f^2)\cdot z \tag{2.10}$$

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. (2.5) is smaller than its threshold value from Eq. (2.10) 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.

2.3.4. Motion noise

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:

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

where $\mathrm{Im}_{N-1,x}$, $\mathrm{Im}_{N-1,y}$ are the x, y derivative of im_{N-1} and $\mathrm{Im}_{N-1,t} = \mathrm{Im}_{N-1} - \mathrm{Im}_N$. Hence, the variance of the intensity difference becomes:

$$Var(\mathrm{Im}_{N-1,t}) = Var(\mathrm{Im}_{N-1,x} \cdot \Delta u + \mathrm{Im}_{N-1,y} \cdot \Delta v) \qquad (2.11)$$

$$= (\mathrm{Im}_{N-1,x}{}^2 + \mathrm{Im}_{N-1,y}{}^2) \cdot Var(uv)$$

where $Var(uv)$ is the variance of the optical flow. Therefore,

$$\sigma_f^2 = (\mathrm{Im}_{N-1,x}{}^2 + \mathrm{Im}_{N-1,y}{}^2) \cdot Var(uv)$$

In Eq. (2.11) we assume that the errors in u and v 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.

2.3.5. Camera noise

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 equal to 1 for our 8 bit webcam quality camera.

2.4. Classification of the tracking object

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. (2.5)) are less than their corresponding threshold values t_{value} (from Eq. (2.10)) as belonging to the tracked object and vice versa.

$${}^{(N)}I_{track} = \begin{cases} 1 & \text{if } t_{stat} \le t_{value} \\ 0 & \text{otherwise} \end{cases} \qquad (2.12)$$

2.5. Postprocessing

We apply postprocessing to ${}^{(N)}I_{track}$ 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.

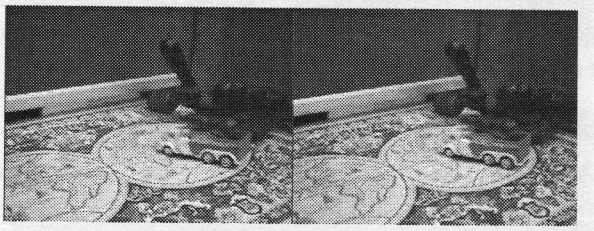


Figure 2.1: Input image frames

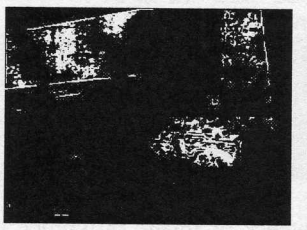


Figure 2.2: Before postprocessing

Figure 2.3: After postprocessing. The seed window is highlighted.

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:

$$ImDiff = \mathrm{Im}_N - \mathrm{Im}_{N-1} \tag{2.13}$$

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 [10], a sample from a normal distribution $N(\mu, \sigma^2)$ has a probability of 0.9987 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 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.

2.5.2. Region growing

ΔIm 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 input track window, we need to eliminate all moving patches unrelated to the tracked object. We perform this motion filtering by a bitwise conjunction between ΔIm and ${}^{(N)}I_{track}$ and save the result as ${}^{(N)}I_{filter}$.

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

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 [6] 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 ${}^{(N)}I_{track}(x_n, y_n) \equiv 1)$

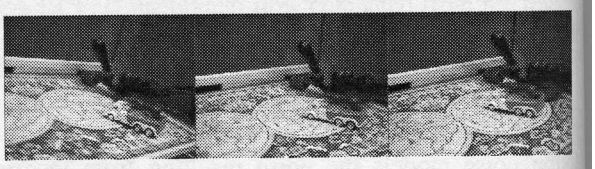
then ${}^{(N)}I_{merge}(x, y) = 1$

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

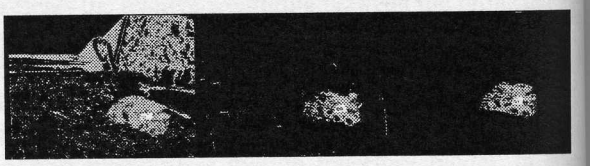
where (x_n, y_n) is any of the four direct neighbors (NW, N, NE, W) of (x, y).

3. Experiments

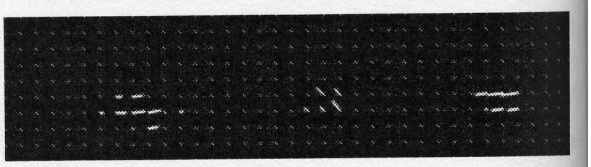
In this section, we present the results of running the tracking system on several real image sequences with moving background. In all experiments we show the image sequence and ${}^{(N)}I_{merge}$ where we highlight the original tracking region.



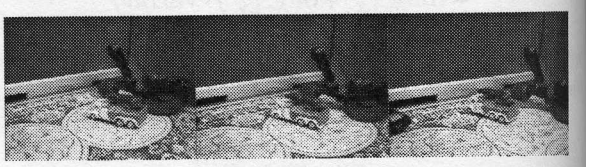
Frames 3, 8, 14 of the truck sequence



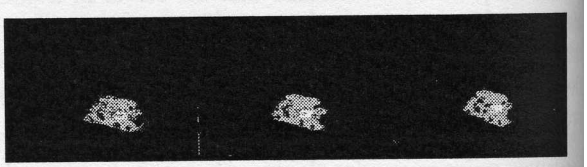
Output from the tracker for frames 3, 8, 14



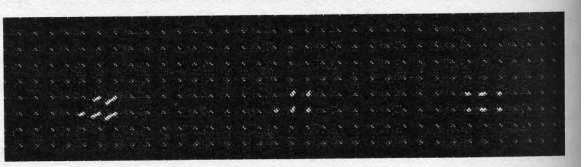
Optical flow computed for frames 3, 8, 14



Frames 17, 23, 29 of the truck sequence



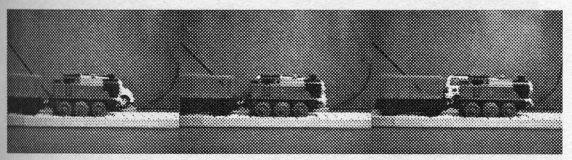
Output from the tracker for frames 17, 23, 29



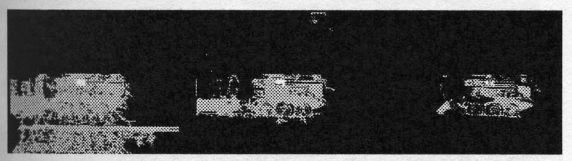
Optical flow computed for frames 17, 23, 29

Figure 3.1: The input image frames show a toy truck with a hippopotamus on it in a moving background. The toy truck is moving from the lower right end corner of the picture towards the upper left hand corner. The sequence is taken by a hand-held camera that attempts to follow the toy truck. We show the image sequence at frame 3, 8, 14, 17, 23, 29 and the output from our tracker for the corresponding frames.

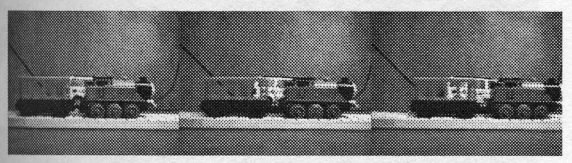
We only show a few selected frames because the sequences are long (25 to 100 frames). In general, the result improves over time. As more frames are processed, we have more history information at our disposal for motion segmentation. We run our experiments on a Sun Blade 100 workstation with 256 MB memory and a 500 Mhz UltraSPARC-IIe CPU that delivers 165 SPECint2000 integer and 163 SPECfp2000 floating point performance [4]. We implement our tracking system using MediaMath [14] The system is written with a combination of MediaMath script, C and MediaLib library from Sun Microsystems. MediaLib [12] is a library that takes advantage of the Single Instruction Multiple Data (SIMD) instruction set of the SPARC processor to accelerate multimedia applications. For input image frames of size 320×240, our tracking system takes 0.3 seconds per frame on Sun Blade 100.



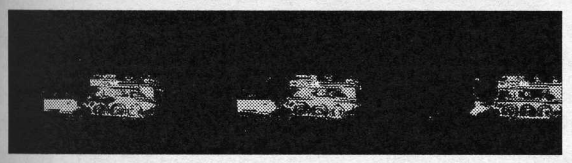
Frame 1, 5, 9 of the translational train sequence



Output from the tracker for frames 1, 5, 9

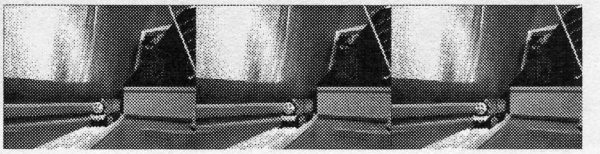


Frame 11, 15, 19 of the train sequence

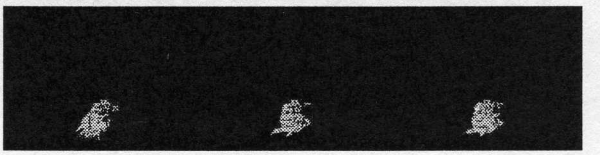


Output from the tracker for frames 11, 15, 19

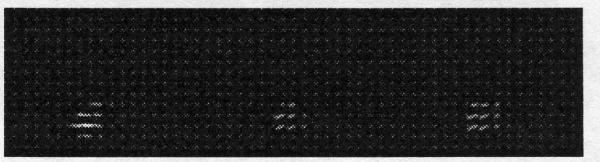
Figure 3.2: The input image frames show a toy locomotive engine going from left to right. We show the image sequence at frames 1, 5, 9, 11, 15, 19 and the output from our tracker for the corresponding frames. The outputs of the tracker shows false positives for frame 1 in areas below the railway track due to the reflection of the train on the table. There are fewer false positives in later frames.



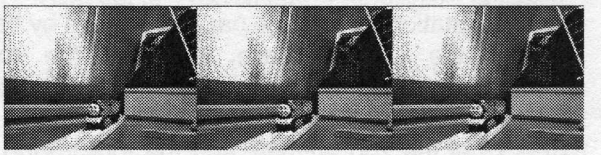
Frame 1, 5, 9 of the approaching train sequence



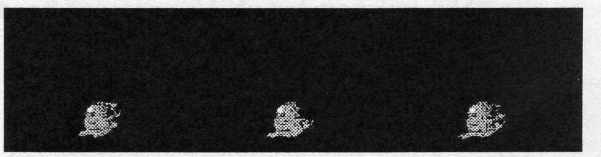
Output from the tracker for frames 1, 5, 9



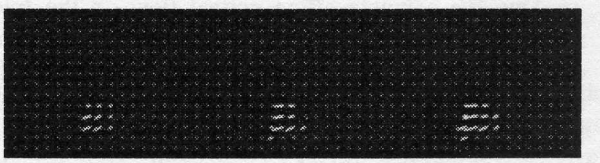
Optical flow computed for frames 1, 5, 9



Frame 11, 15, 19 of the train sequence

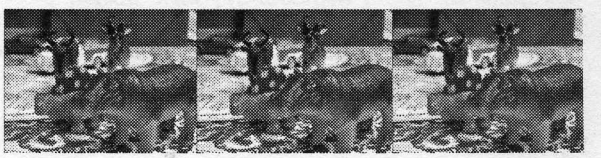


Output from the tracker for frames 11, 15, 19

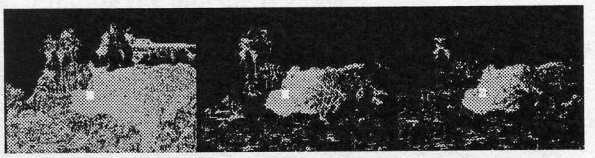


Optical flow computed for frames 11, 15, 19

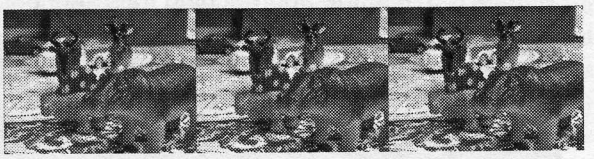
Figure 3.3: The input image frames show a toy train moving towards the camera. We show the image sequence at frame 1, 5, 9, 11, 15, 19 and the output from our tracker for the corresponding frames.



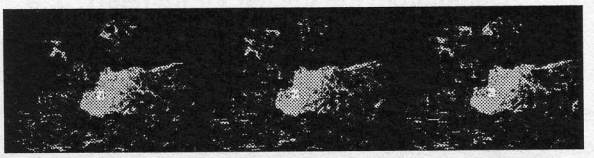
Frames 1, 5, 9 of the zoo animals sequence



Output from the tracker for frames 1, 5, 9



Frames 11, 15, 19 of the zoo sequence



Output from the tracker for frames 11, 15, 19

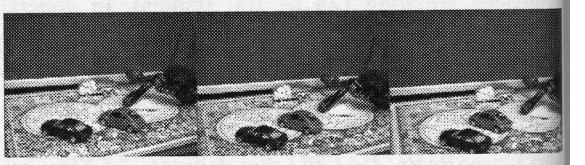
Figure 3.4: The input image frames show toy animals. The animals are on a large piece of paper, the camera is stationary and the paper is moved by hand. Some of the animals shake. We show the image sequence at frames 1, 5, 9, 11, 15, 19 and the output from our tracker at the corresponding frames. For frame 1, all the zoo animals and the foreground are segmented out by our tracker. In later frames, only the head of the hippopotamus is segmented because the tracking region is somewhere on its head.

4. Conclusion

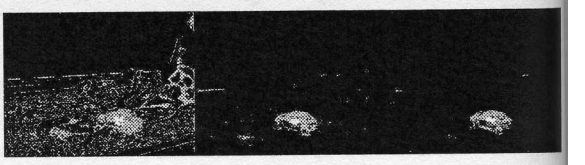
We describe a new algorithm that performs motion segmentation and tracking simultaneously given an input seed window. We compute affine optical flow on the tracked region and align all the previous image frames with the current frame. Pixels belonging to the projection of the tracked object have small sum of square differences between the registered image frames and the current input frame. Motion segmentation is achieved by pixelwise thresholding on the registered image. Aligning all the image frames with the current flow is computationally expensive, so we propose statistics involving the first and second moments of the registered images to speed up this motion segmentation based on tracking process. We showed the result of running our algorithm on real image sequences with moving background. For image frames of 320 x 240 pixels, our tracker runs 3 frames per second on an entry level 500 Mhz Sun Blade 100 workstation.

References

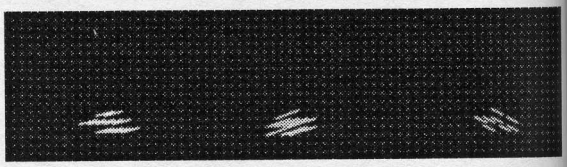
1. M. Black and A. Jepson, "EigenTracking: Robust Matching and Tracking of Articulated Objects Using



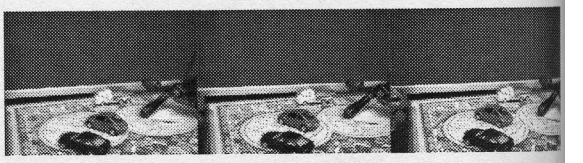
Frame 1, 5, 9 of the moving car sequence



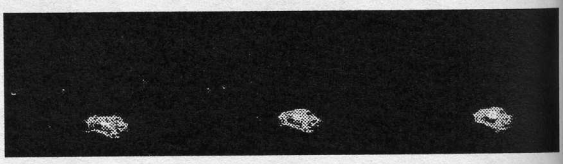
Output from the tracker for frames 1, 5, 9



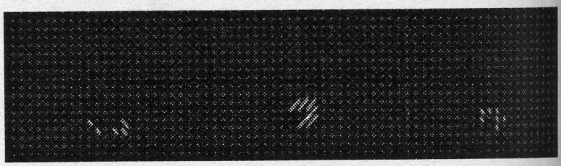
Optical flow for frames 1, 5, 9



Frame 11, 15, 19 of the moving car sequence

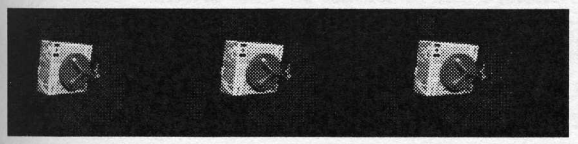


Output from the tracker for frames 11, 15, 19



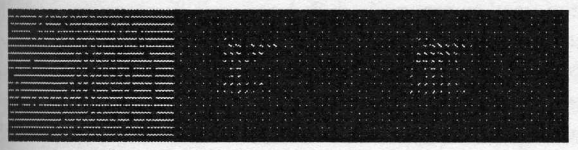
Optical flow for frames 11, 15, 19

Figure 3.5: The input image frames show a toy c going diagonally from lower right corner to upper l corner . We show the image sequence at frame 1, 9, 11, 15, 19 and the output from our tracker for t corresponding frames.

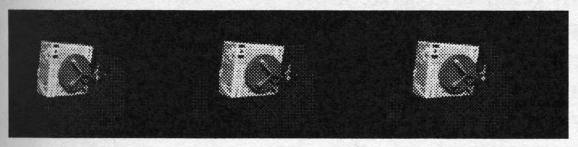


Frames 1, 3, 5 of the satellite sequence

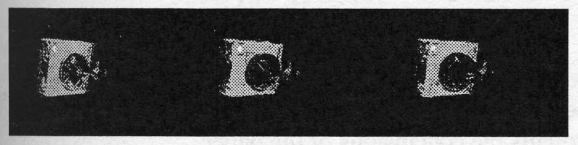
Output from the tracker for frames 1, 3, 5



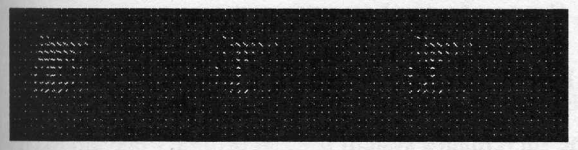
Optical flow for frames 1, 3, 5



Frames 6, 8, 10 of the satellite sequence



Output from the tracker for frames 6, 8, 10



Optical flow for frames 6, 8, 10

Figure 3.6: The input image frames show a satellite as it turns and approaches the camera (Courtesy of MD Robotics). We show the image sequence for frames 1, 3, 5, 6, 8, 10 and the output from our tracker for the corresponding frames. The parts of the satellite that move at approximately the same speed as the tracked region are segmented out.

a View-Based Representation," pp. 1-29 in *International Journal of Computer Vision*, (1998).

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

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

4. Standard Evaluation Corporation, "Submitted CPU2000 results," in *http://www.specbench.org/osg/cpu2000/results*, (2000).

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

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

7. B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence* **17** pp. 185-204 (1981).

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

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

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

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

12. Sun Microsystems, *MediaLib User's Manual Part Number: 802-7799-06,* Sun Microsystems (1999).

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

14. M. E. Spetsakis, "MediaMath: A research environment for vision research," pp. 118-126 in *Vision Interface*, (1994).

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