Tracking, Segmentation and Optical Flow

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

Abstract

Two difficult issues in optical flow are motion discontinuities and large interframe motion. We present an algorithm that addresses both issues by first performing feature tracking and motion segmentation and then warping one of the images to reduce the interframe motion and avoid the motion discontinuities. The algorithm consists of three major phases: 1) feature selection, 2) feature tracking and segmentation 3) optical flow. We used the Lucas and Kanade algorithm to compute the flow. The experiments on real images as well as synthetic images with ground truth showed that this method is very accurate.

Keywords: Motion, Segmentation, Tracking, Optical Flow

1. Introduction

Motion segmentation is defined as the decomposition of an image sequence into coherently moving objects. Being able to do segmentation would greatly simplify the solution of problems like optical flow because we would then be able to work on regions that do not contain discontinuities.

The computation of optical flow is ill-posed due to the *aperture problem* [9]. The most common solution for aperture problem assumes the flow is smooth [10, 17] but the smoothness assumption is violated at motion boundaries of moving objects. If we could segment the image, then we would be able to avoid the motion boundaries. A statistical technique called Expectation Maximization (EM) has been applied successfully to cluster pixels into segments and thus detect boundaries [5]. These motion segmentation techniques assume either that the flow is precomputed usually by dividing the image into regions, computing flow in each region and then merging the regions with similar flow [19, 6, 21] or that the interframe motion is small enough to use the optical flow equation directly [5]. The goodness of the segments produced is limited by the accuracy of the initial optical flow computation step. The main drawback of EM algorithms is that they require the number of segments as input. Although there are techniques that can help estimating the number of clusters (AIC, BIC etc) [14], what seems to work well in practice is to include an extra cluster for all the outliers and work on small areas where the number of segments can be assumed fixed [13].

Another approach to motion segmentation is to identify line or corner features in an image frame and attempt to cluster feature points into regions by following the trajectories of the feature points over time. This can be done by either using active contours [18] or SVD decomposition of the motion measurement matrix into shape and motion matrices which can then be used to segment the image by rearranging the rows and columns of the matrices [8, 11, 15]. The approach we favour in this paper is to track the object and build up a model of the object over time [13, 12, 7, 22].

2. Overview of the Algorithm

The basic steps of the algorithm are as follows:

(1) Select a feature point.

(2) Track the feature and segment assuming various models for flow.

(3) Compute optical flow on the segment.

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

(4) Repeat the above procedure to obtain several segments.

If during tracking and segmentation the feature point proves inappropriate we discard it. The options after that are either to continue with fewer points or select another point in its place. The tracking of a feature point might fail for several reasons such as the point falls outside the image, the segment it detects does not include the feature point itself (usually because the feature is in an area that moves erratically) or the seed region centered on the feature point overlaps very little with the segment (usually due to eminent occlusion).

2.1. Feature Selection

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

We detect corners in an image frame by the corner detection algorithm proposed by Tomasi and Kanade [20] with some speed-up modifications made by Benedetti and Perona [3]. The corner detection algorithm finds feature points that have good localization in all directions. Tomasi and Kanade argue that these are pixels whose smallest eigenvalue of the matrix M [17] is bigger than a threshold λ_t where

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

$E_{xx} = \int I_x^2$, $E_{xy} = \int I_x\, I_y$, $E_{yy} = \int I_y^2$ over a small region and I_x, I_y are the spatial derivatives of the image. Benedetti and Perona speed up the method by using some properties of the characteristic polynomial of the above matrix. Among the points that exceed the threshold, we select N strongest (their smallest eigenvalues are largest) and randomly pick one of them. This will be the center of the small seed region (10×10 pixels in our experiments) used for tracking.

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

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

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

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

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

2.2. Tracking and Segmentation.

Our motion segmentation and tracking algorithm [22] iterates between tracking and segmentation with more elaborate models of flow. Given a seed region R and a pair of successive image frames Im_{N-1}, Im_N, we find the integer displacement $\vec{u}_{R_i}$ that yields the minimal Sum of Squared Differences (SSD) between the two frames using straight search

$$\vec{u}_{R_i} = \min_{\vec{u}\ in\ \vec{u}_{\max}} \left(\sum_{\vec{x}\ in\ R} (\mathrm{Im}_N[\vec{x}] - \mathrm{Im}_{N-1}[\vec{x} + \vec{u}])^2 \right)$$

where $\vec{u}_{\max}$ is the maximum interframe motion in pixels.

We compute the subpixel flow $\vec{u}_{R_s}$ of the seed region R by first shifting Im_{N-1} with the integer flow $\vec{u}_{R_i}$ and then finding the subpixel displacement $\vec{adj}$ that yields the minimum SSD

$$\vec{u}_{R_s} = \min_{\vec{adj}\ in\ \vec{u}_{\max}} \left(\sum_{\vec{x}\ in\ R} (\mathrm{Im}_{N-1}[\vec{x}] - \mathrm{Im}_N[\vec{x} + \vec{u}_{R_i} + \vec{adj}]) \right)^2$$

We apply subpixel shifts $\vec{adj} = (adjU, adjV)$ 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 using cubic interpolation. The above procedure is repeated for successively smaller amount of subpixel shifts. We found empirically the best sequence of values for *subpixel* is $0.75, 0.75^2, 0.75^3 \cdots$. Many other algorithms can do at least as good a job but this one gives us a bound on subpixel accuracy.

The third tracking model assumes affine flow

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

We compute the six affine motion parameters by minimizing the SSDs for all pixels belonging to the region ${}^{(N-1)}I_{track}$ that is the bitwise AND between the enlarged seed region and the segment from the previous step. In this way, we have a larger area to compute the affine flow but we exclude pixels that are not part of the segment in previous frames to avoid inclusion of motion discontinuities. 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 (u, v) is the total flow computed in the previous steps and $\mathrm{Im}_{N-1}^{(u,v)}$ is the image Im_{N-1} aligned by the total flow (u, v). Using a differential approach, as in [10], 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:

$$\begin{aligned} SSD = \sum_{all\ x,y} {}^{(N-1)}I_{track} [&\Delta \mathrm{Im}_N(x,y) \\ &+ \mathrm{Im}_{N-1,x}(x,y)\ u_0 + \mathrm{Im}_{N-1,x}(x,y)\ x\ u_x + \mathrm{Im}_{N-1,x}(x,y)\ y\ u_y \\ &+ \mathrm{Im}_{N-1,y}(x,y)\ v_0 + \mathrm{Im}_{N-1,y}(x,y)\ x\ v_x + \mathrm{Im}_{N-1,y}(x,y)\ y\ v_y]^2 \end{aligned} \tag{2.2}$$

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).

After computing the motion parameters of the seed region, we segment out a region whose pixels are moving in a way consistent with the computed motion parameters by pixelwise thresholding. We warp the previous image Im_{N-1} by the computed flow (u, v) thereby aligning Im_{N-1} with the current image Im_N. Pixels moving consistently with the computed motion model of the seed region have small Sum of Squared Differences. The problem is that many pixels that do not have the desired motion might have small squared difference by coincidence and as a result the segmentation would be noisy. We solve this by adding the squared differences between the current image and all the previous images

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

where Im_N is the current, or N^{th}, image and ${}^{(N)}\mathrm{Im}_i$ is the i^{th} image aligned with the N^{th} image. After expansion and simplification, this gives a much simpler expression

$$= \mathrm{Im}_N^2 - 2\ {}^{(N)}\hat{\mu}_1\ \mathrm{Im}_N + {}^{(N)}\hat{\mu}_2 = t_{stat} \tag{2.4}$$

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 current image. We update the first and second moments by an expression that make the history components of these moments decay exponentially. The constant α is a decimal number between 0 and 1 that arbitrates the relative importance between current and history components:

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

$${}^{(N)}\hat{\mu}_1^t = warp\ {}^{(N-1)}\hat{\mu}_1\ by\ (u, v)$$

Due to the presence of noise, the statistics t_{stat} would be non-zero even for perfectly registered image frames. We attribute the noise to two sources: 1) Camera noise 2) Motion noise. Camera noise models the white noise generated by a camera and can be modeled by a Gaussian distribution of zero mean and variance equal to σ_c^2. In all our experiments $\sigma_c^2 = 1$. Motion noise represents the noise due to the error in motion computation. From 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 Δu and Δv are the errors in the flow, $\mathrm{Im}_{N-1,x}$, $\mathrm{Im}_{N-1,y}$ are the x, y derivatives of Im_{N-1} and $\mathrm{Im}_{N-1,t} = \mathrm{Im}_{N-1} - \mathrm{Im}_N$. We omitted the left superscript (N) because every image is assumed aligned to the current frame. Hence, the variance of the intensity difference becomes:

$$\begin{aligned} Var(\mathrm{Im}_{N-1,t}) &= Var(\mathrm{Im}_{N-1,x} \cdot \Delta u + \mathrm{Im}_{N-1,y} \cdot \Delta v) \\ &= (\mathrm{Im}_{N-1,x}{}^2 + \mathrm{Im}_{N-1,y}{}^2) \cdot Var(uv) \end{aligned} \tag{2.5}$$

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. (2.5) 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.

We set the threshold (t_{value}) for the tracking statistics in Eq. (2.4) [16] to

$$t_{value} = (\sigma_c^2 + \sigma_f^2) \cdot z \tag{2.6}$$

where z is a constant parameter that represents the level of confidence. In all our experiments, we set $z = 3.0$. The segmentation, which is represented by a binary image is

$$^{(N)}I_{track} = \begin{cases} 1 & \text{if } SSD \le t_{value} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

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 eliminate these regions. First, we identify moving pixels using image frame difference and store the result in Δ Im. Second, we extend the conjunction (Δ 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.

2.3. Optical Flow

The last step for every region is to compute optical flow. We warp I_{N-1} with the computed affine motion parameters to bring it close to I_N and then apply the standard Lucas and Kanade algorithm [17] with temporal support of two frames to compute the residual flow. After that we combine residual flow with the affine flow to compute the total flow. The Lucas and Kanade algorithm works really well in this situation because the residual flow is small (around 1 pixel per frame) and there is no discernible motion boundary within the region.

Historically, optical flow computation using differential approach assumes small interframe motion [10]. Although big interframe motion could be handled by following a hierarchical scheme [1], optical flow computation in an image pyramid has strengths and limitations that make it complementary to our tracking and segmentation method. For example, a finely textured object which is conspicuous at full image resolution might be indiscernible at a lower image resolution. A more serious issue is that a hierarchical approach would not work well for tracking objects that are of size comparable to the 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 is comparable to the size of the Lucas and Kanade [17] regions. Such an object might be missed by a hierarchical scheme. Luckily the hierarchical scheme can be relatively easily incorporated into our method if one wants to take advantage of its proven record.

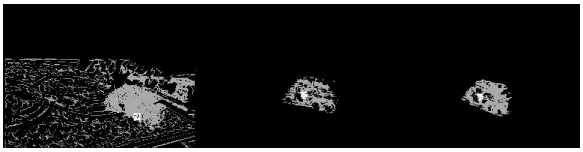
Another difficulty with optical flow algorithms is the presence of discontinuities or motion boundaries. Our motion segmentation and tracking algorithm addresses this issue by performing motion segmentation first.

3. Experiments

In this section, we present the results of running the segmentation and optical flow algorithm on several real image sequences, some with moving background. In general, the segmentation result improves over time because as more frames are processed, we have more history information available for motion segmentation. In all the experiments we generate several random seed regions, we track and segment them and then compute optical flow on them. 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.

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

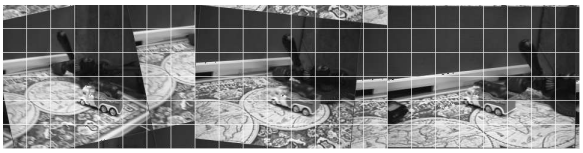
Frames 1, 15, 28 of the turning truck sequence.



Segment corresponding to the turning truck with seed region highlighted.

Residual horizontal flow as gray scale image.

Residual vertical flow as gray scale image.



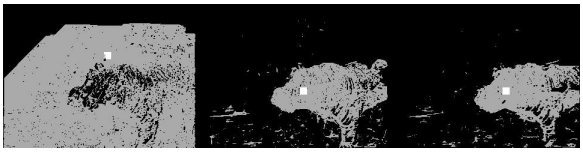
Motion stabilized frames with grid.

Segment corresponding to the moving background.

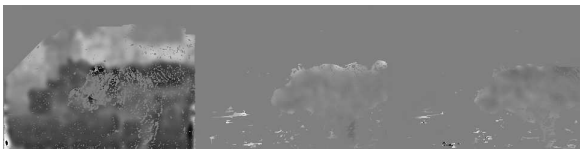
Figure 3.1: The input image frames show a toy truck with a hippopotamus on it taken from a handheld camera that attempts to follow the toy truck.

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

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

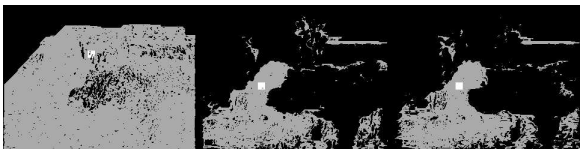


Segment corresponding to the hippopotamus.



Residual horizontal flow as gray scale image.

Residual vertical flow as gray scale image.



Segment corresponding to the background.

Figure 3.2: The toy animals sequence.

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

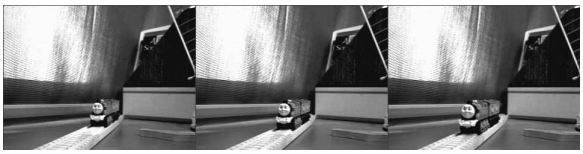
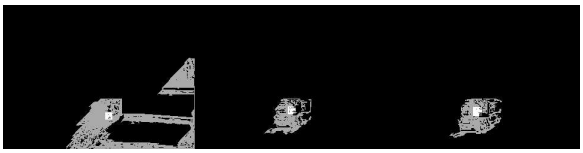


Image sequence at frame 1, 14, 26.



Segment corresponding to the train.

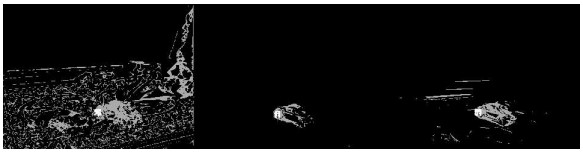
Residual horizontal flow as gray scale image.

Residual vertical flow as gray scale image.

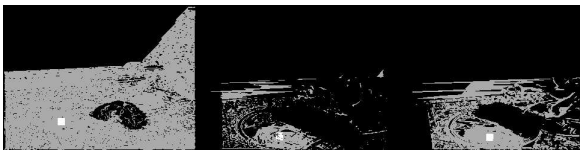
Figure 3.3: The input image frames show a toy train moving towards the camera.

In the fourth experiment, we show the result of running our algorithm on the toy car sequences which shows a toy car going diagonally from lower right hand corner to upper left hand corner. We show two segments generated from randomly selected seed regions: One is the moving car and the other is the background.

Image sequence at frame 2, 15, 27



Segment corresponding to the central moving car.

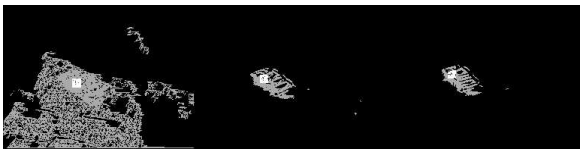


Segment corresponding to the background.

Figure 3.4: The input image frames show a toy car going diagonally from lower right corner to upper left corner.

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

Input sequence at frames 0, 9, 18.

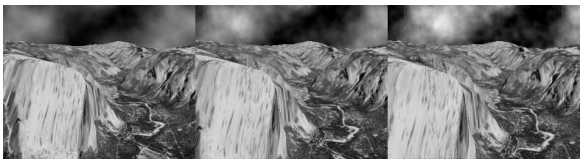


Segment corresponding to the turning taxi.

Figure 3.5: The input image frames show the Hamburg taxi sequence.

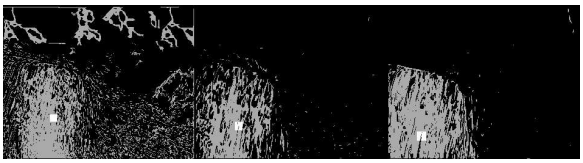
In the sixth experiment, we show the result of running our algorithm on the Yosemite sequence which is a synthetic fly through sequence from the Yosemite valley by Lynn Quann at SRI. We show various segments output by our algorithm from randomly selected seed regions. We show the mean square flow error and the mean angular error [2] for one of the segment between the optical flow computed by our algorithm and the ground truth of the Yosemite sequence. For comparison, we also show the mean square error and the mean angular error between the optical flow computed using the pure Lucas and Kanade [17] without segmentation and alignment method and the ground truth. The error on both algorithms is computed on the same region and both use the same parameters (same derivatives, etc) for the Lucas and Kanade algorithm.

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

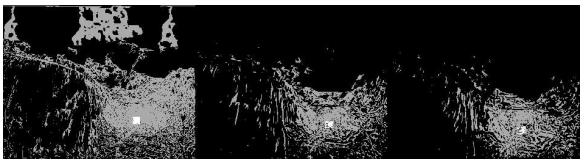


Input sequences at frames 2, 8, 15.

Segment corresponding to the distant mountain slope.

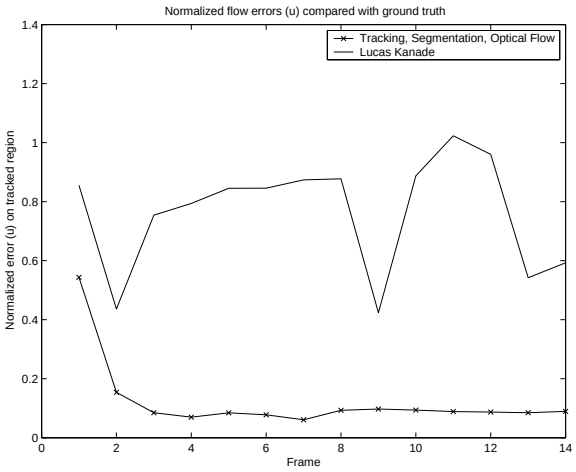


Segment corresponding to the left cliff.

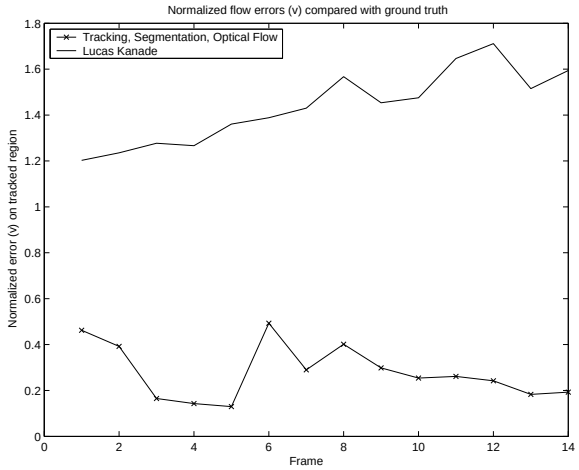


Segment corresponding to the middle valley.

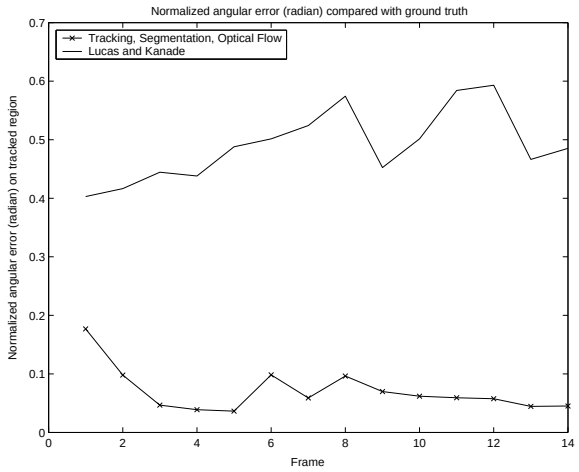
Needle maps of our algorithm, Lucas & Kanade and the ground truth.



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



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



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

Figure 3.6: The input image frames show the Yosemite sequence. Mean square/angular error of flow estimated by our algorithm and the Lucas and Kanade algorithm on the left cliff segment shown above. The ground truth is from Black's [4] web site.

4. Conclusion

We describe a new algorithm that estimates optical flow using feature tracking and segmentation. The algorithm automatically selects a good feature and segments out a region whose motion is consistent with the motion of the selected feature. We perform motion segmentation by an iterative segmentation and tracking process. After that, we compute the residual flow between the image warped by the flow derived during the segmentation step and the next image frame. By following this logic, we identify the region of support for optical flow computation and at the

same time, avoid the problem of computation of optical flow across motion boundaries.

References

1. P. Anandan, “A Computational Framework and an Algorithm for the Measurement of Visual Motion,” *International Journal of Computer Vision* **2** pp. 283-310 (1989).
2. J. L. Barron, D. J. Fleet, and S. S. Beauchernin, “Performance of Optical Flow Techniques,” pp. 43-77 in *International Journal of Computer Vision*, (1994).
3. A. Benedetti and P. Perona, “Real-time 2-D feature detection on a reconfigurable computer,” pp. 586-593 in *Proccedings of IEEE Conference on Computer Vision and Pattern Recognition*, (1998).
4. Black, “Michael J Black: Image Sequences,” in *http://www.cs.brown.edu/people/black/images.html*, (2002).
5. M. Black and A. Jepson, “Mixture models for optical flow computation,” pp. 760-761 in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, (1993).
6. G. D. Borshukov, G. Bozdagi, Y. Altunbasak, and A. M. Tekalp, “Motion Segmentation by Multistage Affine Classification,” pp. 1591-1594 in *IEEE Transactions on Image Processing*, (1997).
7. 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).
8. J. Costeira and T. Kanade, “A Multi-body Factorization Method for Motion Analysis,” pp. 159-179 in *International Journal of Computer Vision*, (1998).
9. B. K. P. Horn, *Robot Vision*, McGraw-Hill Book Company, New York (1986).
10. B. K. P. Horn and B. G. Schunck, “Determining Optical Flow,” *Artificial Intelligence* **17** pp. 185-204 (1981).
11. N. Ichimura, “Motion Segmentation Based on Factorization and Discriminant Criterion,” pp. 600-605 in *Proceedings of seventh IEEE International Conference on Computer Vision*, (1999).
12. M. Irani, B. Rousso, and S. Peleg, “Computing Occluding and Transparent Motions,” *International Journal of Computer Vision* **12**(1) pp. 5-16 (1994).
13. A. Jepson, D. Fleet, and T. F. El-Maraghi, “Robust Online Appearance Models for Visual Tracking,” pp. 415-422 in *IEEE Conference on Computer Vision and Pattern Recognition*, (2001).
14. K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*, Artifical Intelligence Laboratory, Department of Computer Science, Gunma University, Japan (1995).
15. K. Kanatani, “Motion Segmentation by Subspace Separation and Model Selection,” pp. 586-591 in *Proceedings of Eighth International Conference on Computer Vision*, (2001).
16. R. Larsen and M. Marx, *Introduction to Mathematical Statistics and its Applications*, Prentice Hall (1986).
17. B. D. Lucas and T. Kanade, “An Iterative Technique of Image Registration and Its Application to Stereo,” pp. 674-679 in *Proceedings of 7th International Joint Conference on Artificial Intelligence*, (1981).
18. S. M. Smith and J. M. Brady, “ASSET-2: Real-Time Motion Segmentation and Shape Tracking,” pp. 814-820 in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1995).
19. A. Strehl and J. K. Aggarwal, “A New Bayesian Relaxation Framework for the Estimation and Segmentation of Multiple Motions,” in *Proceedings of 4th IEEE Symposium on Image Analysis and Interpretation*, (2000).
20. C. Tomasi and T. Kanade, “Detection and Tracking of Point Features,” in *Technical Report CMU-CS-91-132, Carnegie Mellon University*, (1991).
21. J. Y. A. Wang and E. H. Adelson, “Representing moving images with layers,” pp. 625-638 in *IEEE Transactions on Image Processing*, (1994).
22. K. Y. Wong and M. E. Spetsakis, “Motion Segmentation and Tracking,” pp. 80-87 in *Proceedings of 15th International Conference on Vision Interface*, (2002).