Snakes and Spiders

Brendan McCane
Department of Computer Science
University of Otago, Dunedin, New Zealand
mccane@cs.otago.ac.nz

Abstract

Intensity information is a strong cue for segmentation but on its own cannot be used to distinguish between accidental and non-accidental alignments in a scene, thus resulting in incorrect segmentations. However, motion information can be used to distinguish between accidental and non-accidental alignments. In this paper an integrated method using both intensity and motion information for the segmentation and tracking of objects in a sequence is presented. The method is based on an extension to active contours (snakes) called spiders.

1. Introduction

This paper deals with the problems of motion tracking and object segmentation in an integrated common framework. The techniques presented here are based on the observation that segmentation is easier if features have already been successfully tracked over several frames, and tracking is easier if segmentation has already been performed. This suggests an integrated approach to both problems.

Feature points on a single rigid object are often connected by quite strong edges and this can be used as a useful cue for segmentation. However, accidental alignment may also cause feature points on separate objects to be connected by a strong edge, so any segmentation using edge strength between feature points will not be able to discriminate between real connections and accidental ones. On the other hand, tracking of moving objects over an image sequence will eventually lead to any accidental alignments becoming non-aligned. Therefore, it seems reasonable to expect that reliable segmentation (and therefore tracking) is best achieved using a combination of intensity and motion cues. It is this conjecture which is addressed in this paper.

Motion based segmentation techniques typically come in two flavours: region based methods (usually based on optical flow); and feature based methods. This paper describes a system of the latter type. Combined techniques are also possible and Paragios and Deriche [6] describe a promising technique using an extension of geodesic active contours for segmentation and tracking in a video surveillance application. Although it appears to work very well, their method relies on relatively static backgrounds common in surveillance applications. In contrast, the technique presented in this paper does not rely on static backgrounds and is therefore likely to be more useful for mobile robotic applications.

Smith and Brady [8] describe ASSET-2, a real time motion segmenter and tracker. ASSET-2 utilises corner points which are matched between frames and clustered to produce an object segmentation. The object clusters are then used to improve the reliability of tracking in future frames. The limitation of this system is that until reliable clusters are formed, each feature must be tracked individually. Of course, it is more difficult to track individual features robustly. The system described in this paper does not suffer from this limitation since features are initially tracked dependent on their neighbours according to the intensity information linking any two of them. In other words, corner points linked by strong edges will attempt to remain at a similar distance in subsequent frames.

2. System Overview

The system uses both corner and edge information to perform segmentation and tracking. The basic steps of the process are as follows:

1. Extract feature points using the Harris corner detector [3].
2. Determine neighbours of each extracted point using a Delaunay triangulation and construct a feature point graph from this information.
3. Assign strengths to each graph edge via the minimum cost path between the two points in the image. The minimum cost path is one which has a high image

edge strength and is determined using the livewire algorithm described in Mortensen and Barrett [5].

4. Perform intensity segmentation by keeping only strong edges based on a threshold of the edge strength.
5. For all subsequent frames, use the Spider tracker to track the corner points and continually break those edges which fall below the edge strength threshold. Also break those edges whose strength changes significantly from one frame to the next. A large change in edge strength is indicative of an unstable edge which should be removed.

3. The Spider Tracker

The spider tracker forms the heart of the system. A spider is a generalisation of a snake [1, 4] from a 1D linking of features to a 2D linking of features. It is called a spider because it is a tree structure which bears some similarities to a spider with many legs. A tree was chosen for efficiency reasons. A dynamic programming approach to minimisation of the style of Amini *et al.*[1] can be applied to a tree with the same cost complexity per feature as a snake. A single path from the root of the tree to a leaf is equivalent to a single snake, so each node has only a single predecessor, and therefore the dynamic programming approach can be applied in a straightforward manner. The optimal value function of the energy of a spider then is:

$$S(v_p) = \alpha E_{ext}(v_p) + \sum_{v_c \in C(v_p)} \overset{min}{v_c} \left[S(v_c) + \beta E_{int}(v_c, v_p) \right], \tag{1}$$

where the subscripts p and c indicate parent and child nodes respectively, $C(v_p)$ represents the set of children of node v_p, and the minimum for v_c is calculated over a local neighbourhood for each iteration. So, instead of minimising over a single successor as for a snake, we need to minimise over multiple successors as if we were minimising multiple snakes. Figure 1 shows pictorially what happens for two root node positions in the case of a simplified tree with just three nodes where each vertex has only two possible new positions. For each possible position of the root node, we find the minimum energy locations of its children - the energy of the root node in this position is then the sum of the minimum energies of the children and the external energy of the root node as indicated in Equation 1. The minimum energy locations of the children are found in a similar way if they are not leaf nodes, or are simply the minimum external energy locations in the case of leaf nodes. In Figure 1, the solid line represents the minimum energy configuration and the dashed lines other positions that were tested. This process occurs in a recursive manner so that we calculate the best position of each child node prior to each parent. The minimum energy position of the spider is then calculated by setting the root node to the position of minimum energy and choosing the position of each child node given the position of the root. This change filters down the tree to all leaf nodes. In the case of Figure 1, if we choose position -1 for the root node, then we should choose position -1 for the left child and 0 for the right child as the optimal positions. The process would then be repeated until a locally optimal position was found.

As for the case of snakes [1], each iteration has complexity of $O(nm^2)$ for n feature points and m neighbours using a first order internal energy term. This is not true for a more general graph structure whose worst case complexity would be $O(nm^n)$.

In the case of corner features, the external energy is given by the cornerness field defined by the Harris corner detector [3]. The internal energy attempts to preserve the distances between the feature points weighted by the strength of the edge joining the features:

$$E_{int}(v_c, v_p) = w_l * |l(v_c) - l(v_p) - \bar{d}| \tag{2}$$

where w_l is the weight of the edge joining v_c and v_p and is given by the livewire joining the two points, $l(v_c)$ is the location (image position) of the feature v_c, and $\bar{d}$ is the time-decaying average of the term $|v_c - v_p|$. This has the effect of maintaining the approximate shape of the spider so that any changes are gradual as would be the case for a 3D object changing orientation slowly.

The spider is the minimum spanning tree, in terms of inverse edge strength, of a graph of connected features (in this case corner features). The graph is constructed from the Delaunay triangulation of the features in the initial frame. The Delaunay triangulation is performed simply to get neighbourhood information about the features and to avoid the added cost of calculating a livewire edge between all features in the scene. The reason for choosing a minimum spanning tree should be obvious. Since a tree is needed for efficient execution of the spider tracker, the minimum spanning tree guarantees that the most strongly connected feature points will affect each other during tracking. And, since dynamic programming guarantees convergence to a locally optimal solution, it is unlikely that the choice of root node of the tree will affect the locally optimal position of the spider.

Although it is the spider which performs the tracking function of the system, information concerning connectedness of features, and therefore object segmentation, is maintained. Figure 2 shows the initial frame of the taxi sequence along with the Delaunay graph after edge strength segmentation and the corresponding minimum spanning tree. Notice that five clusters have already formed, but there are still many spurious connections which could not be resolved via intensity information alone, with the current parameter set. Also notice that the features connected by the strongest edges are those which are evident in the minimum spanning tree.

4. Results

The spider tracker has been tested on several real sequences with promising results obtained thus far. Figure 3 shows the spider graph for Frames 5, 12 and 19 of the taxi sequence (Frame 0 is shown in Figure 2). Due to the corner detecting parameters and the lack of contrast, very few corners appear towards the bottom of the images, hence the technique has little or no chance of correctly segmenting the lower two cars. As mentioned previously, Frame 0 contains many spurious intensity edges. By Frame 5, many of these edges have been deleted - especially note that the single feature extracted on a pedestrian in the top left of the sequence has been successfully segmented out from the rest of the scene. Some spurious edges still remain between the parked Volkswagen and the moving car, which are still evident in Frame 12, but are successfully deleted by Frame 19. Of course, the segmentation is not completely successful especially among the stationary objects, but the prominent moving objects, most notably the taxi and the pedestrian have been successfully segmented. The two features on the moving car on the bottom right of the scene are not successfully tracked however, due to the lack of contrast in that area of the scene and fast motion of that car.

Figure 4 shows the spider graph for Frames 0, 5, and 15 of the ambulance sequence. Again, spurious edges exist in Frame 0, some are deleted by Frame 5, and all are deleted by Frame 15, leaving a correct segmentation of the two moving vehicles in the sequence.

At current speeds, the technique takes approximately 30 seconds to process a single frame on a Celeron 366MHz Intel processor.

5. Conclusion

In this paper I have introduced an integrated technique to perform segmentation and tracking based on intensity and motion information. The key contributions of this work include the development of the spider tracker — an extension of active contours — and its application to segmentation and tracking of real sequences. Results from two sequences have been displayed to demonstrate the utility of the technique.

The system is still undergoing development and there are limitations which I intend to address in the future. Future enhancements include: developing a sensible method for adding in new corner features without disrupting current segmentations; predicting positions of the features in the next frame using a Kalman filter; including multi-scale methods to account for large motions; investigating the use of multiple hypothesis methods such as presented in Galvin *et al.*[2] for handling strong occlusions; improving the efficiency of the technique; investigating automatic techniques such as those discussed in Shahraray and Anderson [7] for setting parameters — currently there are 8 parameters to set including four livewire parameters, one corner extraction parameter, and three spider parameters.

6. Acknowledgements

Thanks to Jonathan Richard Shewchuk from the University of California at Berkeley for providing the Delaunay triangulation code free of charge (`http://www.cs.cmu.edu/~quake/triangle.html`).

References

[1] Amir A. Amini, Terry E. Weymouth, and Ramesh C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, September 1990.

[2] Ben Galvin, Brendan McCane, and Kevin Novins. Virtual snakes for occlusion analysis. In *Computer Vision and Pattern Recognition*, volume 2, pages 294–299, Fort Collins, Colorado, USA, 1999. IEEE Computer Society.

[3] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference (AVC88)*, pages 147–151, 1988.

[4] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[5] Eric N. Mortensen and William A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60:349–384, 1998.

[6] Nikos Paragios and Rachid Deriche. Unifying boundary and region-based information for geodesic active tracking. In *Computer Vision and Pattern Recognition*, volume 2, pages 300–305, Fort Collins, Colorado, 1999. IEEE Computer Society.

[7] B. Shahraray and D. Anderson. Optimal estimation of contour properties by cross-validated regularisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):600–610, 1989.

[8] S.M. Smith and J.M. Brady. Asset-2: Real-time motion segmentation and shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):814–820, 1995.

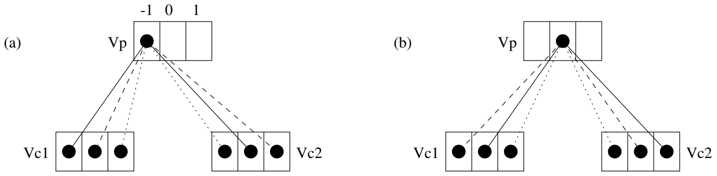
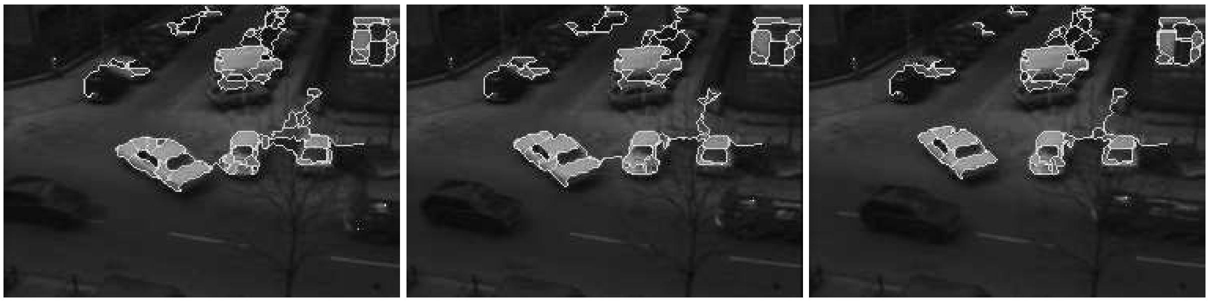


Figure 1. Finding the locally optimal position of the spider.

(a) A superimposed spider graph

(b) Minimum spanning tree

Figure 2. The spider graph and its spanning tree after edge strength segmentation.

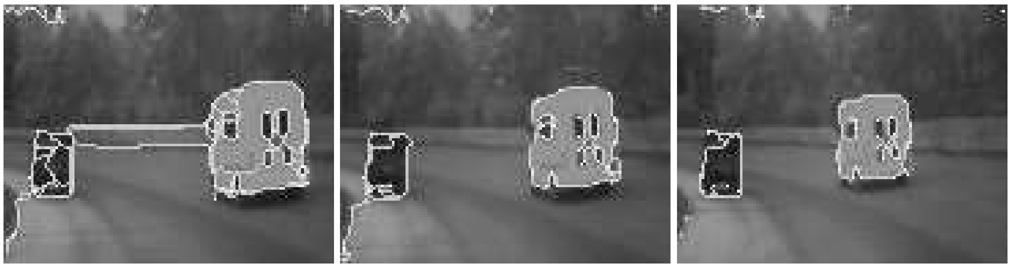


(a) Frame 5 of the taxi sequence

(b) Frame 12 of the taxi sequence

(c) Frame 19 of the taxi sequence

Figure 3. Three Frames of the taxi sequence with the spider graphs superimposed on the scene



(a) Frame 0 of the ambulance sequence

(b) Frame 5 of the ambulance sequence

(c) Frame 15 of the ambulance sequence

Figure 4. Three Frames of the ambulance sequence.