Object Tracking in Image Sequences using Point Features

P. Tissainayagam and D. Suter
raj@tns.nec.com.au and d.suter@eng.monash.edu.au
Dept. of Electrical and Computer Systems Engineering
Monash University, Clayton, Vic. 3168, Australia

Abstract

This paper presents an object tracking technique based on the Bayesian Multiple Hypothesis Tracking (MHT) approach. Two algorithms, both based on the MHT technique are combined to generate an object tracker. The first MHT algorithm is employed for contour segmentation (based on an edge map). The second MHT algorithm is used in the temporal tracking of a selected object from the initial frame. An object is represented by key feature points that are extracted from it. The key points (mostly corner points) are detected using information obtained from the edge map. These key points are then tracked through the sequence. To confirm the correctness of the tracked key points, the location of the key points on the trajectory are verified against the segmented object identified in each frame. The results show that the tracker proposed can successfully track simple identifiable objects through an image sequence.

Key words: Object tracking, Key points, Multiple Hypothesis Tracking, Contour segmentation, Edge grouping.

1 Introduction

The primary purpose of this paper is to track a selected object (as opposed to a single point feature) from the initial frame through the image sequence. The process is an attempt to extend the point feature tracking introduced in [13, 14] to object tracking. In this case, key points from the object are selected using a curvature scale space technique [11] to represent that object. The key points are temporally tracked and are validated against the object contour (obtained by grouping edge segments) in each frame. The tracking technique involves applying the MHT algorithm in two stages: The first stage is for contour grouping (object identification based on segmented edges) and the second stage is for temporal tracking of key features (from the object of interest). For the contour grouping process, we employed the algorithm developed by Cox et al [6], and for the key point tracking procedure we used the tracker introduced by the authors in [13]. Both algorithms combine to provide an object tracker.

The set of image contours produced by objects in a scene, encode important information about their shape, position, and orientation. Image contours arise from discontinuities in the underlying intensity pattern, due to the interaction of surface geometry and illumination. A large body of work, from such areas as model-based object recognition and contour motion flow, depend critically on the reliable extraction of image contours. Reliable image contours are necessary to identify an object with certainty, which in turn is necessary for tracking the object over a period of time in a sequence of images. We use the term 'object' for a group of edge segments that form a recognisable object (identified as belonging to the same object). The object will be identified by an enclosed (*or* near-enclosed) contour.

This paper is organised as follows: Section 2 gives a brief description of the Multiple Hypothesis Tracking (MHT) approach relating to edge segmentation. Section 3 shows how the multiple hypothesis approach can be used for object recognition. In section 4 we briefly show the process to extract key points from an object, and the MHT approach for tracking key point features through an image sequence. Section 5 provides the object-tracking framework employed using methods described in section 3 and 4. Section 6 gives results obtained from experiments. Section 7 gives a general discussion, and finally section 8 provides the conclusion.

2 Multiple Hypothesis Framework for Contour Grouping

This section briefly describes the multiple hypothesis approach in relation to contour segmentation. The details of which are discussed in [6-8].

Fig. 1 outlines the basic operation of the MHT algorithm for contour grouping. At each iteration, there are a set of hypotheses (initially null), each one representing a different interpretation of the edge points. Each hypothesis is a collection of contours, and at each iteration each contour predicts the location of the next edgel as the algorithm follows the contour in unit increments of arc length. An adaptive search region is created about each of these predicted locations as shown in Figure 2 [6]. Measurements are extracted from these surveillance regions and matched to predictions based on the statistical Mahalanobis distance. This matching process reveals ambiguities in the assignment of measurements to contours. This procedure provides an associated ambiguity matrix (Ω) for each global hypothesis from which it is necessary to generate a set of legal assignments. As a result, the hypothesis tree grow another level in depth, a parent hypothesis generating a series of hypotheses each being a possible interpretation of the measurements. The probability of each new hypothesis is calculated based on assumptions described in [6, 8]. Finally, a pruning stage is invoked to constrain the exponentially growing hypothesis tree. This completes one iteration of the algorithm.

In the following sections we briefly describe the contour-grouping algorithm employed, and the key point selection and tracking process used. Both these methods are based on the multiple hypothesis approach.

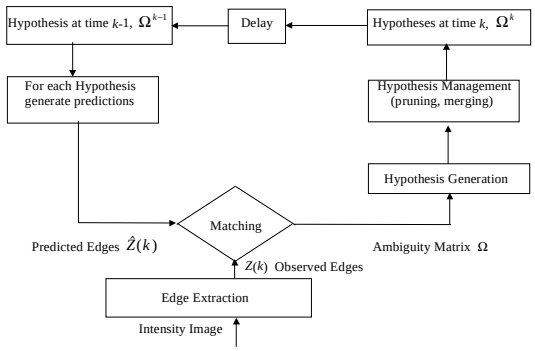


Figure 1: Outline of the multiple hypothesis algorithm for edge grouping

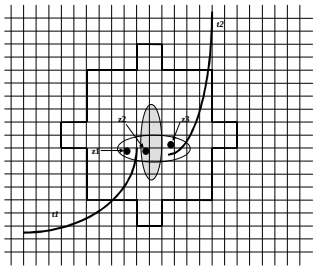


Figure 2: Predicted contour locations, a surveillance region and statistical Mahalanobis (elliptical) regions for a situation with two known contours (t1 and t2) and three new measurements (z1, z2 and z3).

3 Object Recognition

3.1 Contour Segmentation

The contour grouping problem examined in this paper, involves assigning edge pixels produced by an edge detector [4, 5] to a set of continuous curves. Associating edge points with contours is difficult because the input data (from edge detectors) is noisy; there is uncertainty in the position of the edge, there may be false and / or missing points, and contours may intersect and interfere with one another. There are four basic requirements for a successful contour segmentation algorithm. First, there must be a mechanism for integrating information in the neighbourhood of an edgel to avoid making irrevocable grouping decisions based on insufficient data. Second, there must be a prior model for the smoothness of the curve to base grouping decisions on. This model must have an intuitive parameterisation and sufficient generality to describe arbitrary curves of interest. Third, it must incorporate noise models for the edge detector, to optimally incorporate noisy measurements, and detect and remove spurious edges. And finally, since intersecting curves are common, the algorithm must be able to handle these as well. The algorithm due to Cox et.al. [6-8] is one which has a unified framework that incorporates these four requirements, and we will use this algorithm for contour segmentation.

The contour grouping is formulated as a Bayesian multiple hypothesis 'tracking' problem (as in [12]). The algorithm has 3 main components. A dynamic contour model that encodes the smoothness prior, a measurement model that incorporates edge-detector noise characteristics, and a Bayesian hypothesis tree that encodes the likelihood of each possible edge assignment and permits multiple hypothesis to develop in parallel until sufficient information is available to make a decision.

A key step in assigning probabilities to segmentation hypothesis is the computation of the likelihood that a given measurement originated from a certain contour. This likelihood computation depends on two things: a dynamic model that describes the evolution of the curve in the image, and a measurement model that describes how curves produce edgels. In this formulation, the curve state vector is $[x \;\; \dot{x} \;\; y \;\; \dot{y}]^T$ (where (x, y) are the position in a Cartesian coordinate) and its dynamics are described by a linear noise-driven acceleration model common in the tracking literature [1, 2]. The autocorrelation of the white Gaussian acceleration noise can be varied to model curves of arbitrary smoothness. Thus the tip (end point) of the contour as a function of arc length, u, is $(x(u), y(u))$ and has tangent $(\dot{x}(u), \dot{y}(u))$. Since many edge detectors provide gradient information, it is assumed that the entire state vector is available for measurement (a good edge detector such as Canny [5], Boie-Cox algorithm [4] etc. which provide both position and coarse gradient information (horizontal, vertical, and two diagonals) is employable for this application). A Kalman filter is then employed to estimate curve state and predict the location of edgels. These predictions are combined with actual measurements to produce likelihoods.

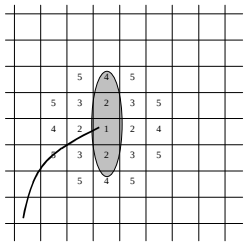


Figure 3: A contour, its surveillance region (labelled 1 – 5) and its validation region.

Once the location of a given curve has been predicted by the Kalman filter and discretized to image coordinates, a surveillance region is employed to extract measurements. A surveillance region is an adaptive variable sized window that travels with the tip of the contour and is used to extract measurements from the edge map. Every iteration, each contour searches for edge points in a series of circles of increasing diameter centred at the predicted contour endpoint. The search halts as soon as at least one measurement is found, or the maximum search radius is reached. The size of the surveillance region determines the distance the curve must travel in that time period, and is reflected in the step size for the curve. The use of a set of windows of increasing size ensures that no more than one measurement from the given contour will be found in a single time period.

The search for measurements takes place after the prediction phase of the state estimator generates an extrapolated endpoint location, (x, y), for the contour. This location determines the discrete image coordinates,

(x_i, y_j), at which the surveillance region is centred. If there is no edge at the predicted location, concentric circles (see Fig. 3), of radius $1, \sqrt{2}, 2, \sqrt{5}$, are searched for edgels (the radii define discrete pixel neighbourhoods). These surveillance regions are labelled 1 to 5 in Fig. 3 (It should be noted that the surveillance region of a contour is not equivalent to its validation region, which is defined by the Mahalanobis distance and is depicted in Fig. 3 as an ellipse). It is these measurements that form segmentation hypothesis whose probabilities are computed. See [6] for details.

3.2 Contour Merging

The grouped contours resulting from the above mentioned process still might have breaks and gaps between segments of the same object. A further refinement process can be employed to merge segments to form identifiable objects. A merging technique is employed by using a distance test (eg: Mahalanobis distance) applied to the end points of contours (assuming a non-closed contour). In this case the multiple contours can be merged to recover the correct segmentation, compensating for the incorrect initial conditions. Two contours with state estimates $\hat{x}_i$ and $\hat{x}_j$ at common boundary are merged if $dx'_{ij}\, \mathrm{T}^{-1}_{ij}\, dx_{ij} \le \delta$, where $dx_{ij} = \hat{x}_i - \hat{x}_j$. T_{ij} is the covariance, and δ is obtained from χ^2 tables or set appropriately as a threshold. This test is applied after the algorithm produces an initial segmentation. The procedure resolves many ambiguities left by the contour segmentation algorithm. A simpler algorithm can also be used simply by using the end-point positions and derivatives of the end-point positions of each curve (produced by the edge detector) which can be quicker.

4 Temporal Tracking of Key Feature Points

In this section we discuss the process to extract key points from the object of interest and we also discuss the procedure to track them temporally.

4.1 Extracting Key Feature Points from Objects

In order to temporally track the object of interest, key points from the object are extracted to represent the object. The key point extraction method should ensure that only true corner points (or any clearly identifiable and definable points) are extracted. Extraction of multiple points within a small region should be avoided (eg: in a curved object, ideally only 1 point should be selected from the curved portion) for good tracking. Since contour grouping (discussed in the previous section) is based on an edge-map, it is desirable that key points should also be selected from the same edge map. Such a process will be efficient and will eliminate the requirement to employ a separate corner detection algorithm. Because of these limitations, we cannot effectively use any of the standard corner extraction algorithms [11] (these calculate corner values directly from the raw image). Instead we have employed a method called the curvature scale space technique [11], which selects key points directly from an edge map efficiently. In the next section the curvature scale space technique is discussed in brief.

4.2 The Curvature Scale Space Algorithm (CSS)

The CSS technique is suitable for recovering invariant geometric features (curvature zero-crossing points and / or extrema) of a planar curve at multiple scales. To compute it, a curve Γ is first parameterised by the arc length parameter *u*:

$$\Gamma(u) = (x(u), y(u))$$

An evolved version Γ_σ of Γ can then be computed. Γ_σ is defined by:

$$\Gamma_\sigma(u) = (\chi(u,\sigma), \gamma(u,\sigma)),$$

where

$$\chi(u,\sigma) = x(u) \otimes g(u,\sigma) \qquad \gamma(u,\sigma) = y(u) \otimes g(u,\sigma),$$

where $\otimes$ is the convolution operator and $g(u,\sigma)$ denotes a Gaussian of width σ (σ is also referred to as the *scale* parameter). The process of generating evolved versions of Γ as σ increases from zero to infinity is referred to as the evolution of Γ. This technique is suitable for removing noise from, and smoothing a planar curve as well as gradual simplification of its shape. In order to find curvature zero-crossings or extrema from evolved versions of the input curve, one needs to compute the curvature accurately and directly on an evolved version Γ_σ. Curvature κ on Γ_σ is given by [11]:

$$\kappa(u,\sigma) = \frac{\chi_u(u,\sigma)\gamma_{uu}(u,\sigma) - \chi_{uu}(u,\sigma)\gamma(u,\sigma)}{\left(\chi_u(u,\sigma)^2 + \gamma(u,\sigma)^2\right)^{1/2}}$$

where

$$\chi_u(u,\sigma) = x(u) \otimes g_u(u,\sigma) \qquad \chi_{uu}(u,\sigma) = x(u) \otimes g_{uu}(u,\sigma)$$

$$\gamma_u(u,\sigma) = y(u) \otimes g_u(u,\sigma) \qquad \gamma_{uu}(u,\sigma) = y(u) \otimes g_{uu}(u,\sigma)$$

4.3 CSS Key Point Detection Method

4.3.1 Brief Overview

The corners (key points) are defined as the local maxima of the absolute value of curvature. At a very fine scale, there exist many such maxima due to noise on the digital contour. As the scale is increased, the noise is smoothed away and only the maxima corresponding to the real corners remain. The CSS detection method finds the corners at these local maxima.

As the contour evolves, the actual locations of the corners change. If the detection is achieved at a large scale the localisation of the corners may be poor. To overcome this problem, local tracking is introduced in the detection. The corners are located at a high scale σ_{high}, assuring that the corner detection is not affected by noise. σ is then reduced and the same corner points are examined at lower scales. As a result, location of corners may be updated. This is continued until the scale is very low and the operation is very local. This improves localisation and the computational cost is low, as curvature values at scales lower than σ_{high} do not need to be computed at every contour point but only in a small neighbourhood of the detected corners.

There are local maxima on the evolved contours due to rounded corners or noise. These can be removed by introducing a threshold value *t*. The curvature of a sharp corner is higher than that of a rounded corner. The final stage to the candidate corner declaration is that each local maximum of the curvature is compared to its two neighbouring local minima. The curvature of a corner point should be double the curvature of a neighbouring extremum. This is necessary since if the contour is continuous and round, the curvature values can be well above the threshold value *t* and false corners may be declared.

4.3.2 CSS Detection Process

The CSS key point detection process can be given by the following steps:

1. Utilise an Edge detector (such as Canny [5] or Boie-Cox [4] etc.) to extract edges from the original image.
2. Extract the edge contours from the edge image:
   - Fill gaps in the edge contours
   - Find the T-junctions and mark them as T-corners
3. Compute the curvature at highest scale σ_{high} and determine the corner candidates by comparing the maxima of curvature to the threshold *t* and the neighbouring minima.
4. Track the corners to the lowest scale to improve localisation.
5. Compare the T-corners to the corners found using the curvature procedure, and remove corners which are very close.

The details of the CSS process can be found in [11].

4.4 Tracking Point Features

The MHT-IMM (MHT coupled with a multiple model Kalman filter, as discussed in [13]) algorithm can be applied for tracking key point features through an image sequence. The measurements for the tracking filter in this case will be the key features extracted (from and near the object of interest) from every frame of a given image sequence (key points are searched within a region of interest surrounding the estimated object centroid). The object centroid position is initially calculated in the first frame by taking the mean of the sum of object key point positions. In the subsequent frames the object centroid is estimated using the MHT-IMM tracker. The extracted measurements are then matched to predictions based on the *Mahalanobis* distance.

The advantage in using the key point tracking algorithm is that we can verify each of the temporally translated key points on the object (selected) against the likely contour of that object in every frame. By doing so, we examine to see whether the object as a whole is tracked correctly (the process for doing this is explained in the next section). In the next section we give the procedure involved in combining the point feature tracking algorithm and the contour grouping algorithm for object tracking.

5 Object Tracking

This section shows how the contour grouping and key point feature tracking procedures combined can be applied for object tracking in image sequences. One cycle of the algorithm recursion is displayed in Fig. (4).

For every frame in an image sequence we first apply the contour segmentation algorithm. This process will group segments of edges that are likely to be from the same object. The result of such a process applied to our test sequences are given in colour Figures 5(a) – 7(a). The procedure as seen from these figures, fail at high curvature contour regions, or is unable to bridge a gap in edgels extracted. As a result, contours from the same object are often broken or separated. To overcome this limitation we applied the contour merging algorithm, which resulted with recognisable object contours (colour Figs. 5b – 7b).

Once the object contours are categorised separately, we can now track a selected object (selection of object can be automated by using a snake type algorithm (eg: Gsnake [9, 10]) or any other suitable algorithm) from the initial frame through the sequence. To track the selected object, we first select some key features (points) from the object (these are selected using the edge map information and then applying the CSS algorithm) as discussed section (4).

The key features of the object are extracted in every frame and the object centroid calculated (this is the mean position of the sum of key points of the object contour). The key points (and the centroid) from the first frame are now tracked through the sequence using the MHT-IMM algorithm (as discussed in [13, 14]). The tracking process is achieved by predicting the object centroid position in the following frame, and then searching a region of interest surrounding the centroid to look for the key points, this process is followed by matching the key points to a grouped object contour within that region. This procedure will provide trajectories for every key point of the object. Each trajectory point is validated against a grouped contour in each frame. By imposing a distance threshold between the tracked key points and the key points on the segmented contour (in each frame), we can verify whether the points have been tracked to an acceptable level of precision. If an acceptable number of key points tracked are identified to lie on or near the object contour (that is passing the threshold test) in each of the frames, we conclude that the object has been tracked successfully. If a key point fails the threshold test, then that point will not be considered as part of that feature trajectory any further.

6 Results

Image Sequence (frames length)	**Number of key points selected**	**Attempted number of points tracked** (& percentage)		**Features tracked for more than 2/3's of the seq. length**	
UMASS (11)	8	8	100 %	8	100 %
PUMA (30)	4	4	100 %	4	100 %
Outcones(20)	12	10	83.3%	10	100 %

Table 1: Object tracking statistics for the 3 test image sequences considered.

The 3 sequences considered give a variety of scenarios to test our algorithm. In all 3 cases the tracking results are promising (see Figures 5 – 7). Table 1 provides quantitative performance values for the object tracker. For the UMASS lab sequence 100% of the key-points selected as forming the object (posters) in the first frame are successfully tracked for the entire sequence length.

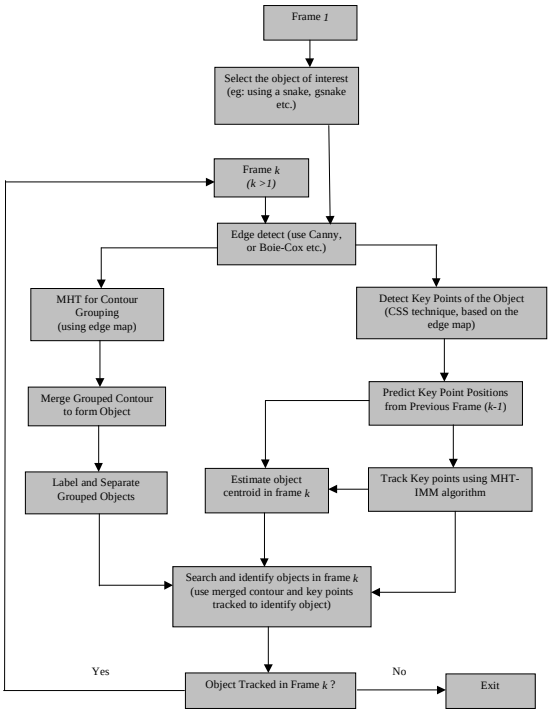
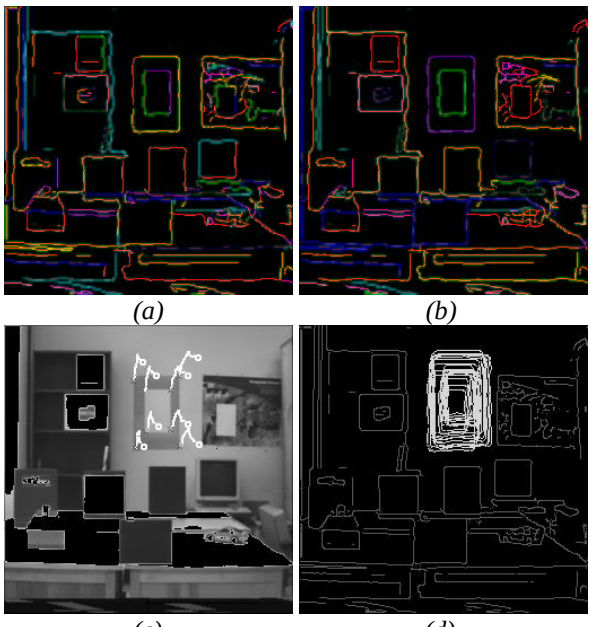


Figure 4: Overview (1 cycle) of the object tracker.

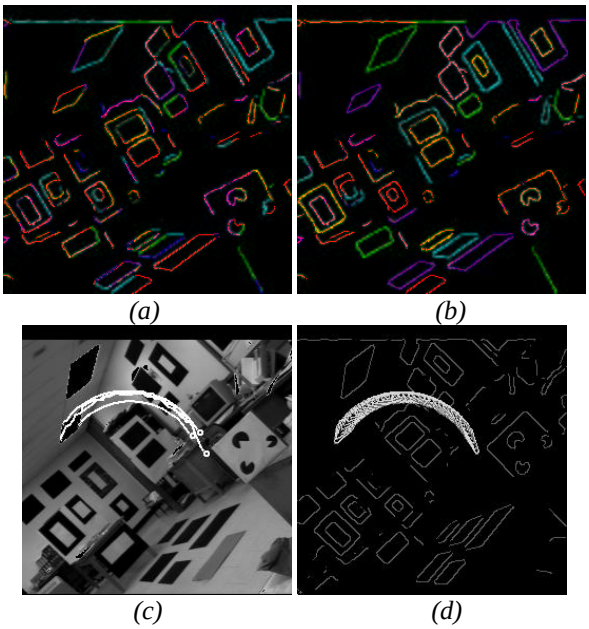
Similar observations can be made for the PUMA sequence. Finally, a multiple object example is demonstrated. For the outdoor cone sequence, 4 cones are considered as part of an object. As the result suggests, 10 out of the 12 key corner features are tracked successfully for more than 2/3's of the sequence length.



(a) *(b)*

(c) *(d)*

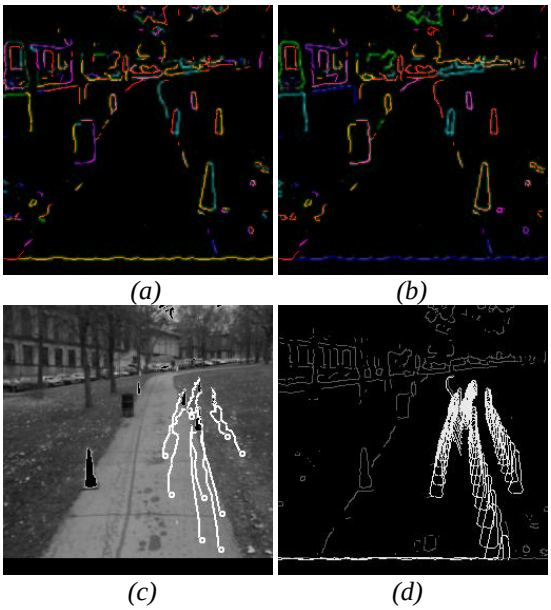
Figure 5: UMASS lab sequence result.



(a) *(b)*

(c) *(d)*

Figure 6: PUMA lab sequence result.



(a) *(b)*

(c) *(d)*

Figure 7: Outdoor cone sequence result.

Figures 5-7 (colour figures): (a) Contours grouped by applying the contour segmentation algorithm based on the edge map obtained (one frame of the sequence is displayed). Each segmented contour (grouped edges) is shown with a different colour. (b) Result after the application of segment merging algorithm (observe that the segments that are identified as forming the same object are merged together in most instances). (c) The trajectory of the key points by applying the MHT-IMM algorithm. The 'x' shows the start of the trajectory while the little white circle indicates the end of trajectory. (d) The identified object trajectory. The white contours (identified as belonging to the same object in each frame) are superimposed on the first frame of the sequence to show the motion of the object.

7 Discussion

Colour figure 5(a) – 7(a) shows the result of applying the contour segmentation algorithm. It can be seen that the segmentation algorithm fails to group segments of the same edge around sharp curves. Since the algorithm scans the edge image by "walking" along the contours, it may encounter a new contour at any point along its length. When tracking begins in the interior of a curve, it is usually partitioned, erroneously into two or more segments sharing common boundary points. As a result of this, contours belonging to the same object can be grouped as separate objects. To overcome this limitation we applied the contour-merging algorithm (as described in section 4) which provided better results (colour figures 5(b) - 7(b)). It can be clearly seen that most of the segments belonging to the same object have now been grouped together successfully (the quality of the segmentation also depends on the thresholds that are used for both algorithms [6-8]).

For the PUMA sequence, the window on the top left corner of frame 1 (see Figure 6) was tracked through the sequence. The result of the tracking is given in Figure 6(d) and the corresponding trajectories of the key points are given in Figure 6(c). From visual inspection the results are promising. Similar results are observed for the UMASS lab sequence (Fig. 5(c, d)), despite the short irregular translation of the posters (top right corner of frame 1). The qualitative results are supported by quantitative results presented in Table 1.

Figure (7) shows the result of the outdoor cone sequence. In this case, multiple objects are tracked (4 cones on the right). Each cone is treated as a separate entity, while all 4 cones combine to form a 'grouped-object'. Each of the key points from the 4 cones are tracked and matched to the segmented shape (the 4 cones). Apart from the last frame, where the cone in the front gets segmented with the road, 83% of the key points have been tracked correctly, thus successfully tracking the 4 cones.

8 Conclusion

In this paper we have shown how the multiple hypothesis technique can be used for rigid object tracking in image sequences. The contour of object tracked is achieved by first applying the MHT approach to group segments of the same object. This process is followed by applying the contour merging algorithm to identify recognisable object contours. Then by selecting key point features of this object, temporal tracking (matching) of key points is achieved by using the MHT again. The validity of the trajectory of the key points is verified by inspecting whether the key points were lying on or near the contour of the tracked object (searched within a region of interest). The results are promising for objects that are not occluded and can be recognised clearly in every frame.

One of the main drawbacks of the system is that the contour grouping process can break down due to occlusion of the object being tracked. The MHT can predict possible trajectory for the key points despite the occlusion [7] and thus retain the trajectory (as shown in [13, 14]). But the contour segmentation and grouping process will fail, as it considers only the edge map to group contours. As a result the object tracker fails in its primary purpose. The tracking process presented may also fail for deforming objects. This is because the key point tracking phase will not be robust enough to track unexpected deformation of object contours.

Recognising and tracking objects using point features as presented in this paper is possible for relatively simple objects (as demonstrated in the results). For complex objects the process is inefficient, and can lead to errors in object identification and tracking. A more versatile method of object tracking will require an object contour to be represented using a parameterised curve, such as using Snakes, Deformable templates, or using B-splines (see [3, 9, 10]).

References

[1] Y. Bar-Shalom and X. R. Li, Estimation and Tracking: principles, techniques, and software, *Artech House*, Boston, MA, 1993

[2] Y. Bar-Shalom and Y. Fortmann, Tracking and Data Association, volume 179 of *Mathematics in Science and Engineering*, AP, Boston, 1988

[3] A. Blake and M. Isard, Active contours, *Springer*, 1998

[4] R. A. Boie and I. J. Cox, "Two dimensional optimum edge detection using matched and wiener filters for machine vision", In *ICCV*, pp.450-456, June 1987

[5] J. Canny, "Computational Approach to Edge Detection", *PAMI*, Vol.8(6), pp.679-698, 1986

[6] I. J. Cox, J. M. Rehg, and S. Hingorani, "A Bayesian Multiple Hypothesis Approach to Contour Grouping and Segmentation", *IJCV*, vol. 11(1), pp.5-24, 1993

[7] I. J. Cox and S. L. Hingorani, "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking", *PAMI*, vol. 18, no. 2, pp.138-150, Feb. 1996

[8] I. J. Cox, J. M. Rehg, and S. Hingorani, "A Bayesian Multiple Hypothesis Approach to Contour Grouping", *ECCV*, pp.72-77, 1992

[9] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active contour models", *IJCV*, pp.321-331, 1988

[10] K. F. Lai and R. T. Chin, "Deformable contours – modeling and extraction", *IEEE Trans. PAMI*, Vol.17(11), pp.1084-1090, 1995

[11] F. Mokhtarian and R. Suomela, "Robust image corner detection through curvature scale space", *PAMI*, Vol.20(12), pp.1376-1381, 1998

[12] D. B. Reid, "An Algorithm for Tracking Multiple Targets", *IEEE Trans. on Automatic Control*, Vol.24, no. 6, pp.843-854, Dec. 1979

[13] P. Tissainayagam and D. Suter, "Visual Tracking with Automatic Motion Model Switching", *International Journal of Pattern Recognition*, Vol(34), pp.641-660, 2001.

[14] P. Tissainayagam and D. Suter, "Tracking multiple object contours with automatic motion model switching", *ICPR-2000*, pp.1146-1149.