Tracking with Multiple Cameras for Video Surveillance

M.K. Bhuyan[†] **Brian C. Lovell**[†‡] **Abbas Bigdeli**[†]

[†] National ICT
Queensland Research Laboratory, Brisbane QLD 4000, Australia.
`manas_kb@hotmail.com, Abbas.Bigdeli@nicta.com.au`

[‡] School of ITEE,
The University of Queensland, Brisbane Qld 4072, Australia.
`lovell@itee.uq.edu.au`

Abstract

The large shape variability and partial occlusions challenge most object detection and tracking methods for non-rigid targets such as pedestrians. Single camera tracking is limited in the scope of its applications because of the limited field of view (FOV) of a camera. This initiates the need for a multiple-camera system for completely monitoring and tracking a target, especially in the presence of occlusion. When the object is viewed with multiple cameras, there is a fair chance that it is not occluded simultaneously in all the cameras. In this paper, we developed a method for the fusion of tracks obtained from two cameras placed at two different positions. First, the object to be tracked is identified on the basis of shape information measured by MPEG-7 ART shape descriptor. After this, single camera tracking is performed by the unscented Kalman filter approach and finally the tracks from the two cameras are fused. A sensor network model is proposed to deal with the situations in which the target moves out of the field of view of a camera and reenters after sometime. Experimental results obtained demonstrate the effectiveness of our proposed scheme for tracking objects under occlusion.

1. Introduction

In recent years, automated processing of surveillance video has gained a lot of attention. Over the past several years, a significant number of papers have reported progress on detecting and tracking humans. However, because of the inherent complexity of the task, problems are still far from being solved, even with the use of a single static camera. Video based tracking can be regarded as the problem of extracting the information about a moving object from a sequence of image frames taken over time. It is the act of searching for a target within a scene and following its position, frame by frame.

The tracking problem can be treated in a number of ways depending on the type of features to be tracked. The contour of the target object can be used as one of the features to be tracked. The contour can be modelled as curves. Again, the prior knowledge of the object dynamics can be used to enhance the tracking performance. Moreover, stochastic dynamic models are required to describe the likely motion and appearance of the target. Stochastic dynamic models along with the input image sequence contains all the information about the position, orientation and velocity of the object. This information can be termed as the state of an object. After this, suitable algorithms are used to keep updating the state by making use of the probability function. The performance of a tracking system is affected by many factors like noise, variations in the background and foreground intensities, occlusion etc. Overall, efficient tracking of an object is quite a challenging problem in Computer Vision.

Single camera tracking is suited only for certain local environments and even simple surveillance tasks demand the use of multiple cameras [1]. A single camera cannot provide adequate coverage of the environment due to its limited field of view. So, it is desirable to have multiple cameras to provide robustness against occlusion. Multiple cameras provide a more complete history of the motion of a target in an environment. Most of the current surveillance applications still treat multiple cameras as a set of single cameras *i.e.,* there is no additional information gained from the use of multiple cameras. But, multiple cameras provide more complete history of a person's actions in an environment.

In multiple cameras based tracking, one important point is to establish a correspondence between different views [2]. Single camera tracking is a correspondence problem from frame to frame over time. But, tracking with multiple cameras is a correspondence problem between tracks of an object seen from different view points at the same time instant. Multiple-camera tracking involves tracking with a single camera at a lower level and then combining the tracks obtained from each of the cameras to get the final track of the object [10].

Multiple camera tracking has not received much attention in the field of computer vision until very recently. Utsumi *et al.* proposed a feature matching approach in which the color or other features of the objects are matched to establish tracking [9]. Kelly *et al.* proposed an approach based on 3D environment, where each camera is calibrated for tracking [6]. Some approaches try to establish time correspondence between non-overlapping fields of view [5].

In this paper, we proposed a method to identify moving objects in a scene based on shape information. We used MPEG-7 Angular Radial Transformation shape descriptor for this purpose. Next, an unscented Kalman filter (UKF)-based track fusion algorithm is developed for tracking targets in a nonlinear multi-sensor system. In a multi-sensor system where each sensor processes its own measurements and keeps its tracks separately, a question is how to decide whether tracks coming from different sensors represent the same target. This is a track-to-track associating problem. When tracks are decided to be from the same target, then the next problem is how to combine (fuse) the track estimates together. This is a track fusion problem. Track fusion is an important part in multi-sensor fusion. In our system, state fusion method is employed to solve this problem. In this, a group of unscented Kalman filters are used to obtain individual sensor-based state vector and variance estimates. Based on the online estimates of the state variances with each sensor, an optimum weight factor is obtained for each state. We now describe this proposed scheme in more detail in the section to follow.

2. Proposed Method

As shown in Figure 1, we used MPEG-7 Angular Radial Shape descriptor to distinguish different objects in a scene on the basis of shape information. The shape dissimilarity measure between the object and the stored shapes in the knowledge base is used to select an object in a scene. Next, we used the B-spline technique to get mathematical representation of the object to be tracked. We used two cameras (simplest case of multiple camera tracking) to track the object and subsequently fused the tracks obtained from the trackers associated with each of the cameras to obtain a final track. Tracking relies on predicting the next state of an object, based on the present state. In our approach with two cameras, the present states of the object, as given by the individual trackers, are fused. The resulting fused state is provided to all the trackers for predicting their next state. The fusion of the data collected from different trackers requires the determination of the accuracy of their respective tracks so that they can be fused in a weighted manner. The state estimate of the object predicted by the occluded camera contains more error than that predicted by the non-occluded camera. Making no distinction between the information given by different trackers could lead to filter instabilities and erroneous estimates especially when one of the trackers is malfunctioning (possibly due to occlusion in its camera). This may even lead to a result which is worse than that with a single camera. In our proposed method, the fusion process is such that the fused estimate is more biased by accurate state estimates and almost unaffected by inaccurate ones. The responsiveness of the fusion process to the state estimates from each of the trackers can be adjusted by the error covariance matrix. The state estimate given by the tracker, the trace of whose corresponding error covariance matrix has smaller value, is given more weightage and vice versa. Our work is aimed towards obtaining a framework for dealing with the situations in which the target moves out of the FOV of one of the cameras for a certain period of time. When this happens, the tracker associated with that camera keeps updating the state. This results in tracking error since the tracker generates an estimate of the state even though it has lost track of the target. If this erroneous in-

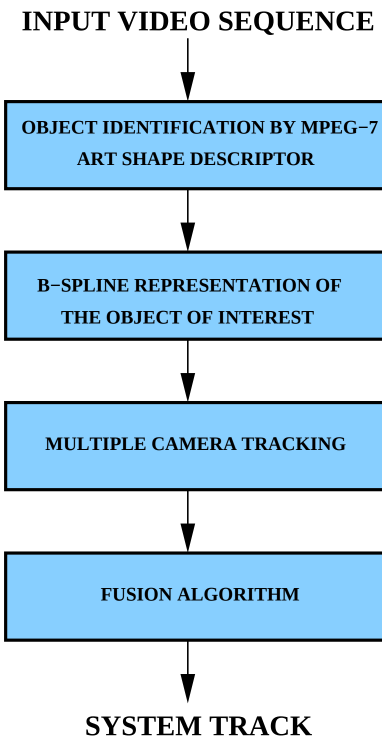
INPUT VIDEO SEQUENCE
OBJECT IDENTIFICATION BY MPEG−7 ART SHAPE DESCRIPTOR
B−SPLINE REPRESENTATION OF THE OBJECT OF INTEREST
MULTIPLE CAMERA TRACKING
FUSION ALGORITHM
SYSTEM TRACK

Figure 1. Proposed scheme for the overall system.

formation is fused with the information from other trackers, the performance of the tracking system degrades. Though fusion is performed in such a way that less weightage is given to the state estimate predicted by erroneous tracker, its effect is not avoided to the full extent.

As shown in Figure 2, the fusion algorithm is present at a base station. In this model, communication between the base station and each of the sensor nodes is bidirectional. Further, mutual communication is not allowed between the individual sensor nodes. The base station is used to keep track of the information about the cameras in which the object is visible, and in which, it is not visible, at each time instant. Moreover, each camera can determine whether the object is visible in the other camera. The sensor nodes communicate this information to the base station. When the base station comes to know that the object is out of the FOV of a camera, it stops using the information obtained from the corresponding sensor node since it will be erroneous. Whenever an object moves out of the FOV of a camera and again reenters, the updating filter of the camera should be reinitialized with the current state of the object. This task is accomplished by the base or central station.

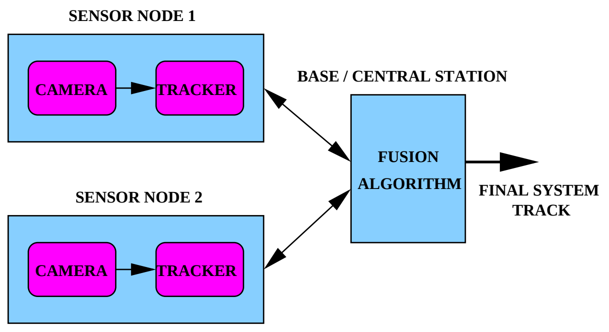

Figure 2. Object tracking method

2.1. Object identification based on MPEG - 7 ART shape descriptor

In our proposed method, MPEG-7 region based shape descriptor is used to find the dissimilarity (in shape) between moving object in a scene and the stored shapes in the knowledge base. If the shape dissimilarity lies below a particular threshold, then the object can be identified on the basis of the shape information. The region-based shape descriptor expresses pixel distribution within a 2-D object region; it can describe complex objects consisting of multiple disconnected regions as well as simple objects with or without holes [3]. Consequently, an ART based descriptor was recently adopted by MPEG-7. Conceptually, the descriptor works by decomposing the shape into a number of orthogonal 2-D basis functions (complex-valued), defined by the Angular Radial Transform (ART) [7]. The normalized and quantized magnitudes of the ART coefficients are used to describe the shape.

From each shape, a set of ART coefficients F_{nm} is extracted, using the following formula:

$$F_{nm} = \; < V_{nm}(\rho,\theta), f(\rho,\theta) >$$

i.e.,

$$F_{nm} = \int_0^{2\pi}\int_0^1 V^*_{nm}(\rho,\theta), f(\rho,\theta)\rho d\rho d\theta \qquad (1)$$

where $f(\rho,\theta)$ is an image function in polar coordinates and V_{nm} is the ART basis function that are separable along the angular and radial directions, that is,

$$V_{nm}(\rho,\theta) = \frac{1}{2\pi}\exp(jm\theta)R_n(\rho) \qquad (2)$$

$$R_n(\rho) = \begin{cases} 1 & \text{if } n = 0 \\ 2\cos(\pi n\rho) & \text{if } n \neq 0 \end{cases} \qquad (3)$$

Descriptor representation

The ART descriptor is defined as a set of normalized magnitudes of complex ART coefficients. Twelve angular and three radial functions are used $(n < 3, m < 12)$. For scale normalization, ART coefficients are divided by the magnitude of ART coefficient of order $n = 0, m = 0$. Therefore, discarding the normalized ART co-efficient of order $n = 0, m = 0$, which is unity, we have 35 coefficients in all. To keep the descriptor size to a minimum, quantization is applied to each coefficient using four bits per coefficient. Hence, the default region-based shape descriptor has total 140 bits.

Shape similarity measurement by ART shape descriptor

The distance (or dissimilarity) between two shapes described by the ART descriptor is calculated using an $L - 1$ norm, for example, by summing up the absolute differences between ART coefficients of equivalent order $(L = 2)$.

$$Dissimilarity = \sum_i \parallel M_d[i] - M_q[i] \parallel \qquad (4)$$

Here, the subscript d and q represent image in the database and query image, respectively and M is the array of ART descriptor values.

2.2. Video based tracking with shape representation by B-spline technique

The first step of the proposed tracking system is to obtain a representation of the shape of the object as a curve. B-spline representation [4] is one of the most frequently used methods for this purpose. A probability function can be obtained by using stochastic dynamic models to describe the likely motion of the object and this gives information about the initial state of the object. The tracking algorithm has to update the state at each time instant. For updating of the target state, the unscented Kalman filter approach is used in our method.

Unscented Kalman filter

The unscented Kalman filter is a recursive Minimum Mean Square Error (MMSE) estimator [8]. The UKF incorporates the latest observations into a prior updating routine. The UKF does not approximate the process and observation models rather approximate the distributions of the state random variable. In the UKF, the state distribution is represented by a Gaussian random variable (GRV), and is specified using a minimal set of chosen sample points. These sample points completely capture the true mean and covariance of the GRV. When propagated through the true nonlinear system, they capture the posterior mean and covariance accurately to the second order for any nonlinearity, with errors only introduced in the third orders and more. The process described above is called Unscented Transformation (UT), which is a method for calculating the statistics of a random variable that undergoes a nonlinear function. The unscented Kalman filter algorithm is able to accurately predict the state and co-variance of a system and is easier to implement. In this method, a small number of carefully chosen sample points are propagated at each time step. These sample points provide compact parameterizations of the underlying distribution. This is more robust than the tracking algorithms based on Kalman filter.

2.3. Object tracking with multiple cameras

In our proposed method, single camera tracking is performed first to implement the multiple-camera tracking system. In this method, low-level tracking is performed at each of the cameras using the unscented Kalman filter algorithm. Then, a fusion algorithm is employed to fuse the individual tracks and yield the final system track. To take advantage of multiple cameras, it is necessary to establish correspondence between different views. This problem is termed as the consistent labeling problem in multiple cameras. In this, all views of the same object is given the same label. The correct instant where the correspondence is to be established is the instant when a new view is seen. In a multiple-camera tracking system, at each time instant, the object to be tracked may be visible in some of the cameras and not visible in the others. So, the tracking system has to find out the set of cameras in which the object is visible, at each time instant. To accomplish this task, the concept of field of view (FOV) lines is used.

Concept of FOV lines

The field of view lines are the edges of the footprint of a camera as seen in other cameras. The computation of these lines is required to find correspondence between different views and tracks. It also provides information about the set of cameras in which that object will be visible for each new view. A view-event is a time instant when an object enters or leaves the field of view of a camera during tracking. Ambiguity in labeling of an object arises when an object enters the FOV of a camera. So, the boundaries of FOV of a camera are very important in consistent labeling problem. The cameras are assumed to have overlapping FOVs but it is allowed for two cameras FOVs not to overlap with each other, as long as they are linked with cameras in between. This is a reasonable assumption since, if an object disappears completely from all the cameras and then reappears in a camera; it will be treated as a new object. Figure 3 shows the FOV lines of two cameras C^1 and C^2. The FOV of a camera is a rectangular pyramid in space with tip at the center of projection of the camera and its four sides passing through the lines on the image plane. Let S denotes one of the above mentioned lines. A portion of the FOV line of a camera is seen in another camera if they have overlapping FOVs. If $L^{i,S}$ is an FOV line of C^i from side S, then $L_j^{i,S}$ is the FOV line of side S of C^i in C^j. In Figure 3, there are three objects in the ground plane covered by the fields of view of two cameras C^1 and C^2. Object 3 is visible in both cameras whereas Object 1 is visible in C^1 only and Object 2 is visible in C^2 only. A portion of the FOV lines of C^2 are visible in C^1 and a portion of the FOV lines of C^1 are visible in C^2. The camera C^1 should have the information about whether an object visible in C^1 is also visible in C^2 or not. This information can be obtained with the help of FOV lines as described in the following section.

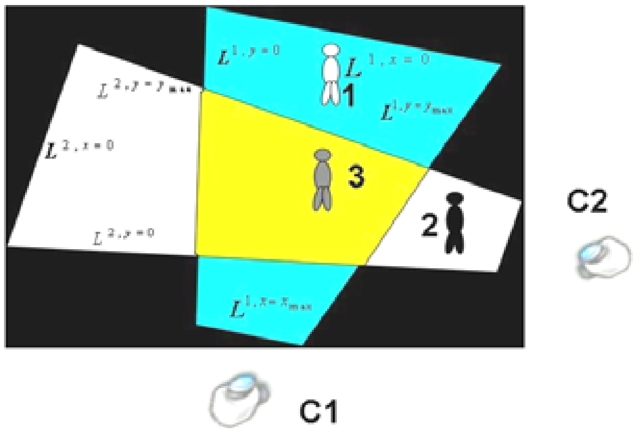

Figure 3. Field of View Lines of two cameras

Consistent labelling of objects

Let the object seen in C^i be denoted as O_k^i. This is the local label of the view of the object given by the single-camera tracking module. O^i denotes the set of all objects visible in C^i. For example, $O^i = \{O_1^i, O_3^i\}$ means object 1 and object 3 are currently visible in C^i. The location of a view is given by a single point (x, y) that is $(x_k^i, y_k^i) = P(O_k^i)$, where $P(.)$ denotes the location of the object.

The labeling task is to establish equivalences of the form $O_m^i \rightarrow O_n^j$, where O_m^i is the object with label m in camera C^i and O_n^j is the object with label n in camera C^j. If the FOV lines are known, each line $L_j^{i,S}$ divides the image into two parts, the part which is inside the FOV of C^i and the other which is outside. The function $L_j^{i,S}(x, y)$ returns greater than 0 if the point (x, y) lies on the side of $L_j^{i,S}$ that should be visible in C^i and less than 0 if it should not be visible in C^i and equal to 0 if it is on the line.

Now, we can determine whether the current object O_k^i in C^i will be visible in another camera or not. The set of cameras C in which the current new view will be visible is given by

$$C_i(k) = \{j / L_i^{j,s}(P(O_k^i)) > 0 \forall \{x / L_i^{j,x} \exists C^i\}\} \quad (5)$$

If $C = \Phi$(empty set), the view at (x, y) in C^i is that of a new object that is not currently seen in any other camera. If C is not empty, at least one of the existing views must correspond to the current view. The camera in which the corresponding object will be found is a member of C. This is a generalized method for the case of multiple objects tracking with multiple cameras. This can be easily extended to single object tracking, since, in this simple case, it is sufficient to determine whether an object is visible in each camera, at each time instant.

2.4. Track fusion algorithm

This is the final step of our proposed system. Once the tracking is performed at each of the cameras, the individual tracks are to be fused/combined to obtain the final system track. In this step, a sensor model can be used to fuse the tracking results from each camera. Each camera and its associated tracker can be treated as the constituents of a sensor node and a fusion algorithm can be developed to fuse the information from each sensor node. As shown in Figure 2, the central node possesses the fusion algorithm to combine the tracks obtained from each sensor node.

Theoretically, the best tracking performance can be achieved by fusing the measurements from the sensors directly. But, direct fusion has one major disadvantage. It requires large volumes of raw data to be processed for long distances on the network. This is not feasible for a network with a large number of sensors. The cost of communications increases with the number of sensors in the system. Hence, we have employed a hierarchical structure. In this structure, the fusion system has no direct access to the sensor data. The sensor data are processed locally to form sensor tracks and finally these are fused to give system tracks. Track fusion is needed to associate the sensor tracks. This association generates an improved target estimate in the scene. In general, the concept of track fusion is technically more complex than measurement fusion. The state estimates of sensor tracks cannot be treated like sensor measurements. Moreover, these estimates can not be fused with a centralized tracking algorithm. This is because while sensor measurement errors are independent across sensors, the errors in target state estimates associated with the tracker's outputs are correlated with one another.

Architectures for track fusion algorithm

There are two processing architectures for track fusion algorithm. These two architectures are described below.

- ***Sensor-to-sensor track fusion:***

  In this method, the state estimates from different sensor tracks are (propagated to a common time) associated and fused to obtain the state estimate for the system track. As shown in Figure 4, the previous state estimate of the system track is not used in this process. With this architecture, there is no requirement to deal with the problem of correlated estimation errors. As it is a memory less operation, the errors in track state estimation fusion are not propagated in time. But, as the past processing results are discarded, this approach is less efficient.

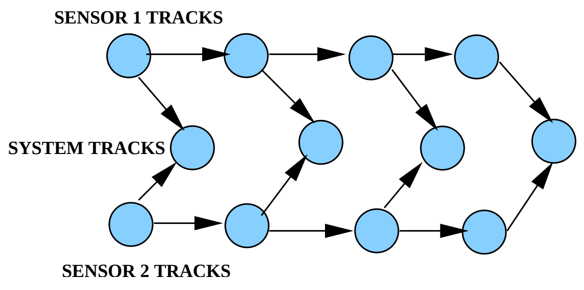

Figure 4. Sensor to sensor track fusion.

- ***Sensor-to-system track fusion:***

  In this approach, whenever a set of sensor tracks are received, the state estimates of the system tracks are extrapolated to the time of sensor tracks and fused with the newly received sensor tracks. This process is repeated when another set of sensor tracks is received.

This architecture has to deal with the problem of correlated estimation errors. In Figure 5, the sensor tracks in A and the system tracks in B, all depend on C and hence, the errors in A and B will be correlated. Also, any errors in system tracks due to past processing errors in fusion will affect future fusion performance.

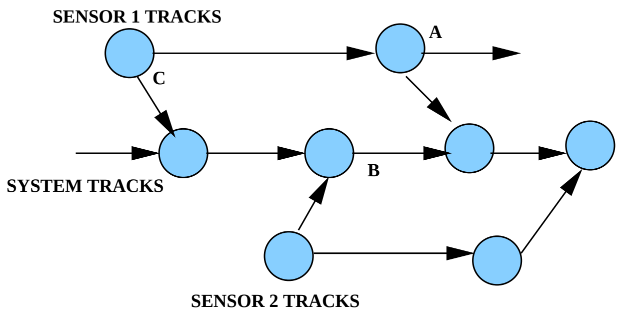

Figure 5. Sensor to System track fusion.

Some technical issues in track fusion algorithm

The inputs to track fusion are the sensor tracks formed from local measurements and are represented by position and velocity estimates and their error covariance matrices. Consider two tracks i and j with state estimates and error covariance matrices $\hat{X}_i$ and $\hat{X}_j$, P_i and P_j respectively. The estimate fusion problem is to find out the best fused estimate $\hat{X}$ and error covariance matrix P. If it is sensor to sensor fusion architecture, the two tracks may be sensor tracks and if it is sensor to system fusion architecture, they may be a system track and a sensor track.

The fused estimate can be expressed as a linear combination of the track estimates *i.e.* $\hat{X} = A\hat{x_i} + B\hat{x_j}$. The matrices A and B are chosen to optimize some criterion like the minimum variance. Two algorithms, namely, basic convex combination and linear combination with cross covariance, are available for linearly combining the track estimates depending on whether the cross covariance between the track estimates is considered.

- ***Basic convex combination:***

  In this, the cross co-variance between the two track state estimates can be ignored. Corresponding to this condition, the fusion algorithm is given as

$$\hat{x} = P_j(P_i + P_j)^{-1}\hat{x}_i + P_i(P_i + P_j)^{-1}\hat{x}_j$$

$$= P(P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j) \quad (6)$$

$$P = P_i - P_i(P_i + P_j)^{-1}P_i$$

$$= P_i(P_i + P_j)^{-1}P_j = (P_i^{-1} + P_j^{-1})^{-1} \quad (7)$$

  One advantage of this algorithm is that it is simple to implement. In the case of sensor to sensor track fusion, this algorithm will be almost optimal if there is no process noise. But, if sensor to system track fusion is employed or if there is process noise, this algorithm is only suboptimal.

- ***Linear combination with cross covariance:***

  In this case, the cross covariance between the two states cannot be ignored. Accordingly, the fusion algorithm is represented as

$$\hat{x} = \hat{x}_i + (P_i - P_{ij})(P_i + P_j - P_{ij} - P_{ji})^{-1}(\hat{x_j} - \hat{x_i}) \quad (8)$$

$$P = \hat{P}_j + (P_i - P_{ij})(P_i + P_j - P_{ij} - P_{ji})^{-1}(P_j - P_i) \quad (9)$$

  The cross covariances P_{ij} and P_{ji} can be computed from the observation matrices and the Kalman filter gains. The advantage of this algorithm is that it is robust to common process noise. The main disadvantage is that more amount of information is needed to compute the cross covariance. The method used to integrate the information passed by the sensors depends on whether the sensors provide competitive information or complementary information. In the first case, each sensor provides identical information. This redundancy of the sensor readings helps in improving the reliability of the network. Also, the noise in the signals can be detected and removed. The basic principle is that the noise in different sensor signals tend to be uncorrelated while the signals of interest are correlated. So, if the information from the sensors is combined in an effective manner, the noise can be reduced significantly, giving comparatively good results. The method of complementary information integration is implemented only when partial information is available from each sensor. In case of our proposed fusion process, the information provided by the cameras comes under competitive information since all the cameras give the information about the track of the object in the scene.

3. Experimental results

First, we showed that the object of interest can be located in the scene based on the shape information. As shown in Figure 6, after determining moving objects in the scene, shape information of the objects are derived on the basis of MPEG-7 ART shape descriptor and shape similarity measure (with predefined threshold) is used to compare the observed shape with the stored one. After detecting the object of interest in the scene, the object can be tracked in the subsequent frames. The black strip shown in the images is used to create an artificial occlusion for experimentation. First, we have performed face tracking and vehicle tracking against a cluttered background with a single camera. After this, the same experiment is performed with two cameras. Figure 7 shows the results for these two cases.

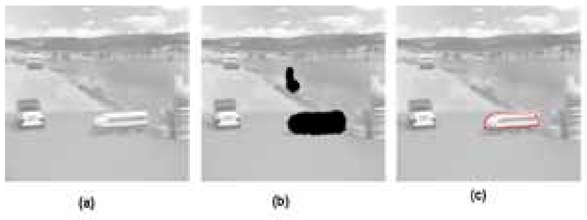

Figure 6. Object identification.

It is evident that in frame number 24, the object is occluded and the single camera lost its track. But in case of two cameras tracking, even though the object is occluded in one of the cameras, the other camera provided true information and finally the fusion algorithm used it to correct the track of the occluded camera.

4. Conclusion

In this paper, we have proposed an algorithm for identification of moving object based on MPEG-7 ART shape descriptor. Finally, we have developed a method for video based tracking with multiple cameras and subsequently demonstrated that multiple camera based tracking enhances the tracking performance as compared to the single camera tracking, especially in the presence of occlusion. One important aspect of the proposed system is the allocation of weights to the track estimates obtained from each of the cameras. If the track estimates generated by each of the cameras are not weighted properly during the process of fusion, it may lead to degradation of the performance. Single camera tracking is adequate in ideal cases with simple background and minor occlusion. So, there is no need to bear the complexity of using multiple cameras in these cases. Future work includes implementation of the base station model and development of a background motion compensation model.

5. Acknowledgement

This project is supported by a grant from the Australian Government Department of the Prime Minister and Cabinet. NICTA is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

[1] K. Chan, R. Saha, and Y. Bar-Shalom. On optimol track-to-track fusion. *IEEE Transaction on Aerospace Electronic Systems*, 33(4):1271–1276, 1997.

[2] C. Y. Chong, S. Mori, W. H. Barker, and K. C. Chang. Architectures and algorithms for track association and fusion. *IEEE AES Systems magazine*, pages 5–13, January 2000.

[3] B. Erol and F. Kossentini. Local motion descriptors. *Proc. IEEE Workshop on Multimedia Signal Processing*, pages 467–472, 2001.

[4] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, 1989, ISBN 0133361659.

[5] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking across multiple cameras with disjoint views. *Proc. Ninth IEEE International Conf. Computer Vision*, pages 952–957, October 2003.

[6] P. Kelly, A. Katkere, D. Kuramura, S. Moezzi, S.Chatterjee, and R. Jain. An architecture for multiple perspective interactive video. *Proc. ACM Multimedia*, pages 201–212, 1995.

[7] B. S. Manjunath, P. Salembier, and T. Sikora. *Intoduction to MPEG-7, Multimedia Content Description Interface*. John Wiley and Sons Ltd., England, 2002.

[8] D. Ponsa, A. Lopez, J. Serrat, F. Lumbreras, and T. Graf. Multiple vehicle 3D tracking using an unscented kalman filter. *Proc. eighth International IEEE Conf. Intelligent Transportation Systems*, pages 1108–1113, September 2005.

[9] A. Utsumi and J. Ohya. Multiple camera based human tracking using non synchronous observations. *Proc. Asian Conf. Computer Vision*, pages 1034–1039, January 2000.

[10] H. Yang and J. Zhang. An unscented kalm an filter-based multisensor track fusion algorithm. *Proc. International Conf. Instrumentation and Measurement Technology*, pages 527–530, May 2005.

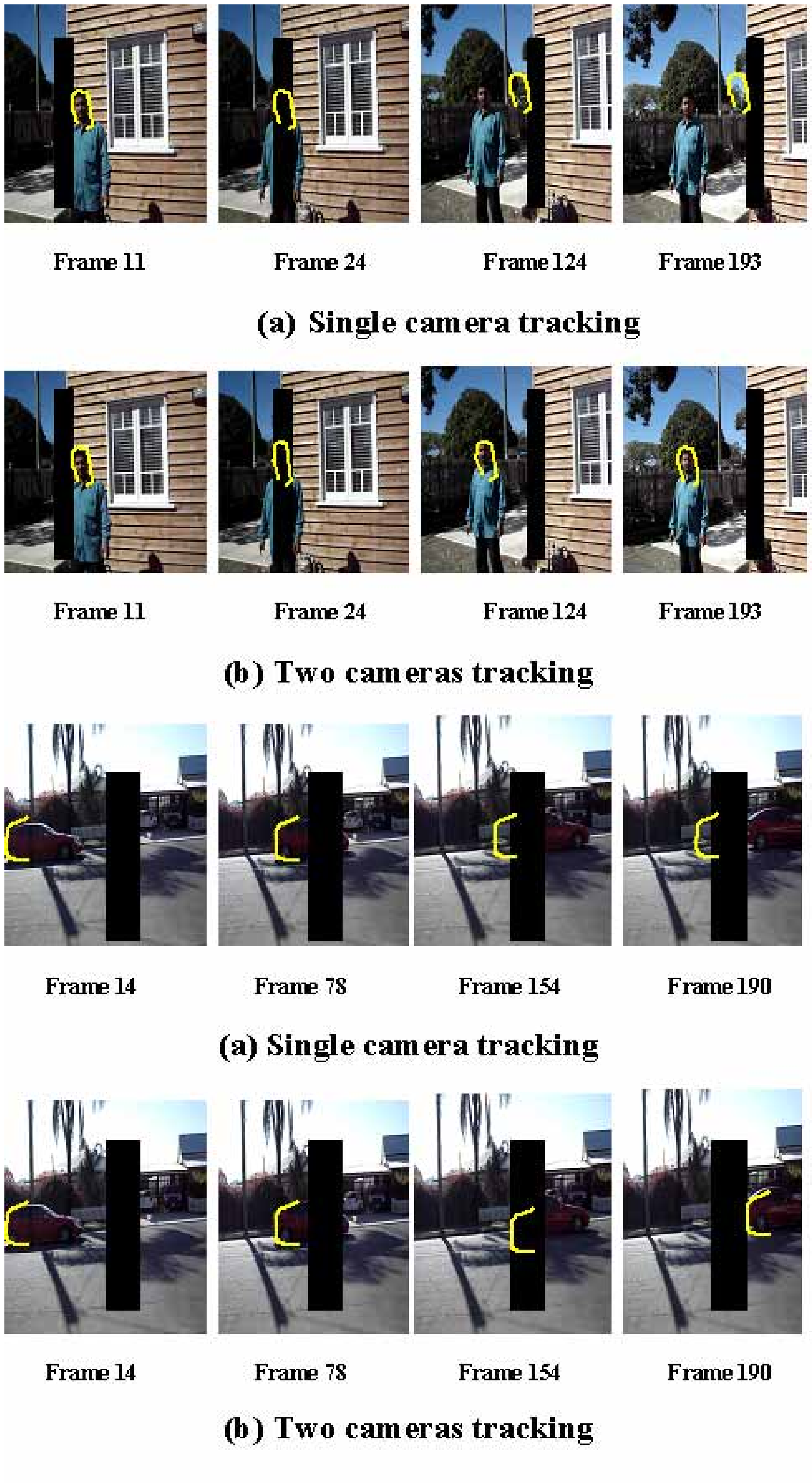

Figure 7. Comparison of tracking performance: Single vs Two cameras tracking.