# Cooperative Multitarget Tracking With Efficient Split and Merge Handling

Pankaj Kumar, *Member, IEEE*, Surendra Ranganath, Kuntal Sengupta, and Huang Weimin

***Abstract*—For applications such as behavior recognition it is important to maintain the identity of multiple targets, while tracking them in the presence of splits and merges, or occlusion of the targets by background obstacles. Here we propose an algorithm to handle multiple splits and merges of objects based on dynamic programming and a new geometric shape matching measure. We then cooperatively combine Kalman filter-based motion and shape tracking with the efficient and novel geometric shape matching algorithm. The system is fully automatic and requires no manual input of any kind for initialization of tracking. The target track initialization problem is formulated as computation of shortest paths in a directed and attributed graph using Dijkstra's shortest path algorithm. This scheme correctly initializes multiple target tracks for tracking even in the presence of clutter and segmentation errors which may occur in detecting a target. We present results on a large number of real world image sequences, where upto 17 objects have been tracked simultaneously in real-time, despite clutter, splits, and merges in measurements of objects. The complete tracking system including segmentation of moving objects works at 25 Hz on 352 $\times$ 288 pixel color image sequences on a 2.8-GHz Pentium-4 workstation.**

***Index Terms*—Data association, dynamic programming (DP), Kalman filtering, multitarget tracking (MTT), shape matching, split and merge.**

## I. Introduction

FOR identifying object specific behaviors, detection of the moving objects by itself is usually not sufficient. In most cases, along with motion information, trajectory analysis and interaction of the target with other elements of the scene are important for robustly inferring behavior. Therefore, it is important that objects are reliably and continuously tracked, and their identity (ID) is preserved even if they split or merge with other targets in the field of view (FoV) of the camera. In the presence of several objects, the problem is one of multitarget tracking (MTT) where targets and observations/measurements need to be matched from frame to frame in a video sequence. MTT has usually been formulated as an optimal feature estimation and data association problem, e.g., in the radar community [1], [2], and we follow this paradigm. MTT has two aspects; one is data association where the measurements from the video frames need to be properly associated with objects, and the other is a filtering scheme to use the associated measurements to optimally update object track. The filtering process is usually handled by optimal filters such as Kalman filter [3], [4], grid-based filter or extended Kalman filter [4], interacting multiple models [5] or particle filters [6].

The popular solutions for this problem are joint probabilistic data association filter (JPDAF) [7] and multiple hypothesis tracking (MHT) [8], [9]. Both these methods are based on the following assumptions: 1) an object will generate one measurement in each frame and 2) a measurement could have originated from at most one object. Both assumptions usually fail when tracking objects with visual sensors. Due to occlusions or errors in segmentation, an object can split into multiple blobs and yield more than one measurement for a single object. Often, objects can overlap to give rise to only one blob from the several objects that have merged. Fig. 1 shows examples of splits and merges. These situations occur frequently in visual data and, hence, object splits and merges must be explicitly taken into account for accurate tracking in dense tracking environments.

Table I gives a comparative overview of works on visual MTT. Some of the important considerations of MTT algorithms are whether splits and merges are considered, the data association method used, track initialization, and computational complexity. The last is important for tracking algorithms to work in real-time when tracking several objects. Cox and Hingorani [11] efficiently implementated Reid's MHT algorithm [8] for tracking targets with visual sensors. However, they tracked point features and did not address the problem of data association in the presence of splits and merges. Medioni *et al.* [10] proposed an approach based on graph theory for tracking multiple targets which was similar to Reid's MHT. Their algorithm considered splits only, and they used gray level correlation between objects and segmented blobs to detect and handle splits. Algorithms have also been proposed for tracking people [17], [18] with splits and merges, and they are based on a 2-D model of humans. Groups of people have been tracked in [16] with split and merge but there is no systematic discussion on handling complex merges and splits which arise while tracking large number of objects. A few works on MTT where the core problem of data association in the presence of splits and merges is addressed are Genovesio and Olivo-Marin [12] and García, *et al.* [19], [20].

The idea in [12] was to synthesize a set of *virtual measurements* from actual measurements obtained from splitting and merging objects, for use in the JPDA filter which requires one to one matching between objects and measurements. The virtual measurements are obtained by using gating and predicted states

Manuscript received February 7, 2006; revised June 9, 2006. This paper was supported in part by the NUS grant R-263-000-214-112. This paper was recommended by Associate Editor E. Steinbach.

P. Kumar is with the School of Computer Science University of Adelaide, Adelaide, SA 5005, Australia.

S. Ranganath is with the Department of Electrical and Computer Engineering, National University of Singapore, 117576 Singapore (e-mail: elesr@nus.edu.sg).

K. Sengupta is with the AuthenTec Inc., Melbourne, FL 32902-2719 USA.

H. Weimin is with the Institute for Infocomm Research, 119613 Singapore.

Color versions of Figs. 1, 2, 4, 6, and 8–12 are available at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2006.885715

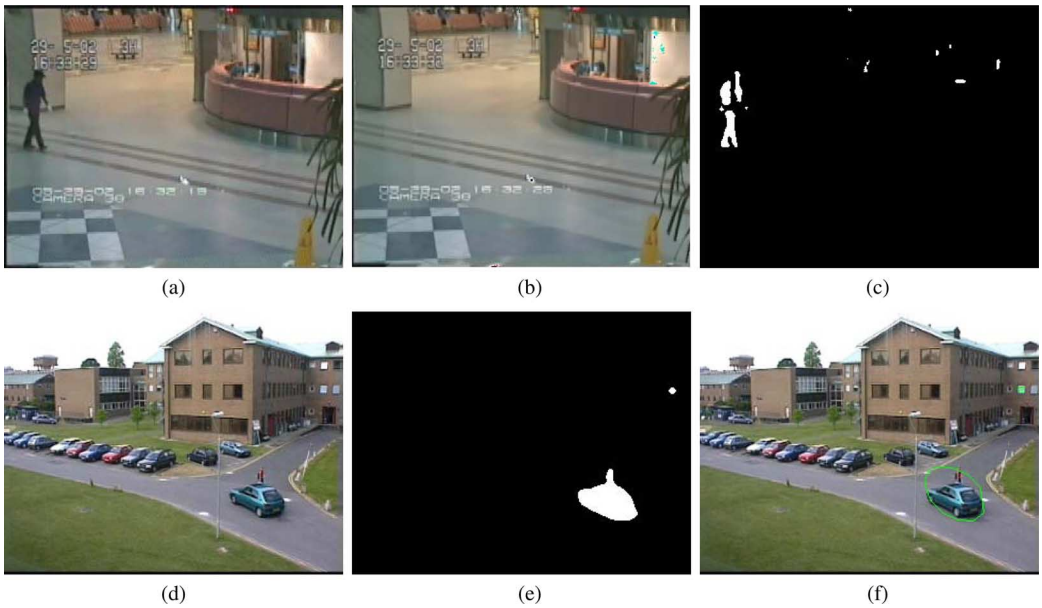


Fig. 1. (a)–(c) A case of splitting when the foreground object color is similar to the background. The black clothing of the person is quite similar to the black color on the edges of the walls, as seen in the background model (b). This causes splitting of the object into multiple blobs as seen in (c). (d) Merging of two targets (a person and a car) and the corresponding segmentation shown in (e) is obtained after foreground/background segmentation. (f) Shows that only one bounding ellipse is obtained as measurement for the two targets.

TABLE I
A SUMMARY OF RELATED WORKS IN THE AREA OF VISUAL MTT WITH SPLITS AND MERGES

| References | Data Association | Split Merge Handling | Track Initialization | Real Time | Average numbers of tracks |
|---|---|---|---|---|---|
| Medioni *et al.* [10] | Graph based | Split only | Automatic | No | 5-6 |
| Cox & Hingorani [11] | Mahalanobis & cross correlation test | Not discussed | Manual | No | 50-150 |
| Genovesio & Olivo-Marin [12] | Virtual measurements & JPDAF | Yes | Not discussed | Not Mentioned | 3-4 |
| Tao *et al.* [13] | Motion layer correspondence using Gaussian priors | No | Automatic | Yes | 4-5 |
| Javed & Shah [14] | Intensity Correlation | Occlusions considered | Not discussed | Yes | 4-5 |
| Paragios & Deriche [15] | Not discussed | No | Manual | No | 2-3 |
| McKenna *et al.* [16] | Support map and bounding box | Heuristic | Automatic | Yes | 3-4 |
| Haritaoglu *et al.* [17], [18] | Correlation based correspondence | Object Template | Automatic | Yes | 4-5 |
| García *et al.* [19] | Fuzzy Heuristic & JPDAF | Yes | Not discussed | Yes | 4-5 |
| Kumar *et al.* [This paper] | DP based shape matching | Yes | Automatic graph based | Yes | 10-17 |

of the objects. Subsequently, all possible feasible associations between target tracks and the virtual measurements are evaluated to find the association that maximizes the joint association probability. The number of feasible associations grows rapidly with the number of targets, increasing the computational load to find the optimal association. However, no efficient scheme was proposed to find the optimal associations; this would be necessary for real-time implementation in dense tracking environments (e.g., traffic scenes). Moreover, their formulation of overlapping tracks and conflicting measurements did not consider situations that may occur in traffic scenarios, for example, several initially separated, tracked vehicles coming to a stop at a traffic light. These may merge into a single elongated blob/measurement, linearly chaining the gating ellipses of the objects. Fig. 2 shows an example of this. Four different vehicles have merged to give rise to one blob. The resulting blob measurement lies outside the gating ellipse of object $T_4$ (which is usually computed using the innovation covariance matrix), and hence $T_4$ will lack measurement to update its track. When the prediction and measurement error is small, the gating ellipse will be small but as the error in prediction and measurement increases, the gating region will increase. In visual tracking, objects can merge quite quickly, while it may take longer for the gating region to increase in size. This may result in tracks being terminated for lack of matching measurements.

García, *et al.* [19], [20] propose an MTT system that avoids the combinatorial complexity of evaluating matches for data association in the presence of splits and merges by using a fuzzy logic rule-based system. The fuzzy system integrates different heuristics computed from gated blobs and target tracks to compute "confidence levels" that are used to weight each gated blob's contribution to update the target track and its rectangular gate. They showed good tracking results from using their system in an airport scenario which consisted of taxiing aircraft and other ground operation vehicles. Real-time operation was shown with examples of two or three moving objects. Though not explicitly mentioned in the works, the system may be able to track a larger number of objects in real-time.

Our approach has been to find an efficient computational solution to the data association problem in the presence of object splits and merges. We have done this by formulating a novel geometric match measure which allows decomposing the matching

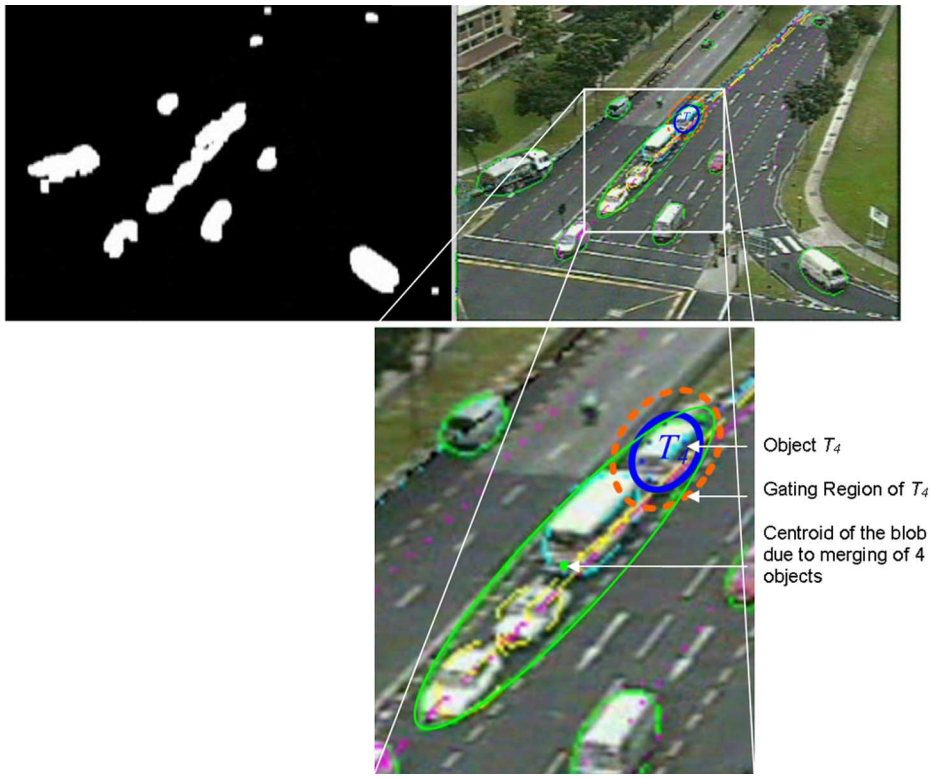



Fig. 2. Problem that can arise when several objects merge during visual tracking. The merges may lead to the blob measurement vector lying outside the gating region of several of the targets that have merged.

problem into several subproblems and facilitates the use of dynamic programming (DP) for data association. This allows the overall MTT system to run in real time (25 fps) as a cooperative effort between the target matching algorithm and Kalman filter based tracking, even when tracking a large number of targets simultaneously (we have tracked up to 17). In addition, a graph based object track initialization algorithm is introduced to ensure automatic, accurate, and stable initialization of the trackers in our MTT algorithm.

DP is a computationally efficient divide-and-conquer method [21] where the optimal solution to a problem is obtained by combining optimal solutions to subproblems. It is a powerful technique for nonlinear matching as shown by Martelli [22], and has long been used to great advantage to solve problems efficiently in computer vision. For example [23]–[25] used it for character recognition. [26] used it for contour based object matching and [27] used the same technique to recognize objects such as scissors, metal rings etc. in images. Amini *et al.* [28] used DP to find object contours by minimizing snake energy in the presence of hard constraints and Geiger *et al.* [29] used DP for active contour matching and tracking. Ueda and Mase proposed another method of matching contours in [30], where they imposed a cost for the deformation of the contour to make contour detection robust. Hospital *et al.* [31] proposed a contour matching scheme for recognition of a sequence of linear segments in contours. Other examples of using DP for pattern matching can be found in [32] and [33].

The rest of the paper is organized as follows. In Section II we propose an object modeling and matching algorithm which facilitates the use of DP strategy to efficiently compute the data association of targets in the presence of multiple splits and merges. Section III presents a cooperative MTT scheme based on the target matching method and Kalman filtering. A new algorithm for automatic initialization of target tracking is presented in Section IV. Finally, results and conclusions are discussed in Sections V and VI, respectively. The appendix contains the details of Kalman filter equations for tracking.

## II. Feature Extraction and Object Matching

We use our active background modelling and foreground segmentation scheme to segment moving foreground objects in the camera's FoV [34]. This method has features for shadow and highlight removal and can also adapt to gradual changes in the background. The extracted foreground regions are enclosed within their convex hulls to remove concavities. If there are small connected regions lying within the convex hull of a larger connected region, then the smaller regions are ignored and only the larger region is considered. The convex hulls are approximated by an ellipse using the algorithm in [35]. Each foreground region in a frame is called a *blob* from which the following features are extracted.

1) Centroid of the ellipse, $X_c$.
2) $J$ angularly equidistant control points $X_1, X_2, \ldots, X_J$ on the ellipse (we used $J = 12$ in our experiments).
3) The normalized, $L$ bin histogram of the $Y$, $Cr$, $Cb$ channels of the blob, $H_1, H_2, \ldots, H_L$. This histogram is computed from the foreground pixels inside ellipse.

The $n$th blob in a frame is represented as

$$B^n = \left\{b^n_{X_c}, b^n_{X_1}, b^n_{X_2}, \ldots, b^n_{X_J}, b^n_{H_1}, b^n_{H_2}, \ldots, b^n_{H_L}\right\}. \quad (1)$$

The $m$th *object/target* being tracked by the Kalman filter is represented as

$$T^m = \left\{t^m_{X_c}, t^m_{X_1}, t^m_{X_2}, \ldots, t^m_{X_J}, t^m_{H_1}, t^m_{H_2}, \ldots, t^m_{H_L}, t^m_{V_c}, t^m_s\right\}. \tag{2}$$

This is the same representation as for blobs with the addition of two more features, viz., $t^m_{V_c}$, the velocity of the centroid and $t^m_s$, a parameter to measure scale change of the object.

### A. Match Measures

Three match measures $D_S$, $D_X$, and $D_H$ are discussed here for matching objects with blobs based on shape and color information. The control points of the $m$th object $T^m$ are $t^m_{X_1}, t^m_{X_2}, \ldots, t^m_{X_J}$. These control points form polygon $Poly_{T^m}$ and enclose area $A_{T^m}$. Similarly, the control points of the $n$th blob, $B^n$ form polygon $Poly_{B^n}$ and enclose area $A_{B^n}$. $(A_{T^m} \bigcap A_{B^n})$ is the common area between the polygons $Poly_{T^m}$ and $Poly_{B^n}$. Two match measures $D_S$ and $D_X$ can now be defined for matching the shape of a blob $B^n$ with object $T^m$. $D_S$ is a simple shape matching measure that is described here for the purpose of discussion—to comparatively assess and illustrate our new shape matching measure $D_X$ that is better suited to the task of matching objects when there are splits and merges. $D_S$ is defined as

$$D_S(B^n, T^m) \triangleq \frac{\sum_{j=1}^{J} d_s^2\left(b^n_{X_j}, Poly_{T^m}\right)}{(A_{T^m} + A_{B^n})}$$
$$d_s(b^n_{X_j}, Poly_{T^m}) \triangleq \text{Shortest distance of } b^n_{X_j} \text{ from Poly}_{T^m}. \tag{3}$$

The denominator normalizes the match measure with respect to the sum of the areas of the object and blob. The match measure $D_X$ is defined in (4), as shown at the bottom of the page. The computation of $d_s$ and $d_x$ is explained by the illustration in Fig. 3. Some of the properties of $D_X$ are the following.

1) Nonnegative: $D_X(B^n, T^m) \geq 0$.
2) Nonsymmetric: $D_X(B^n, T^m) \neq D_X(T^m, B^n)$.
3) $D_X(A, B) = 0 \wedge D_X(B, C) = 0 \Rightarrow D_X(A, C) = 0$.
4) $D_X(A, C) = 0 \wedge D_X(B, C) = 0 \nRightarrow D_X(A, B) = 0$.

A blob $B^n$ is a match with object $T^m$ when the match measure $D_X(B^n, T^m) = 0$. This happens when $B^n$ is spatially coincident with object $T^m$ or lies entirely within it. However, $B^n$ can be considered a match with $T^m$ when $D_X(B^n, T^m)$ is less than a small threshold, which is not critical in practice. The presence of $\digamma$ in the denominator of (4) is to ensure that $B^n$ and $T^m$ can match only when they overlap. When there is no overlap between $B^n$ and $T^m$, $\digamma = 0$ and $D_X(B^n, T^m) = \infty$, thereby preventing a match. In other words blob and object overlap is necessary for a match and subsequent updating of object track using blob features.

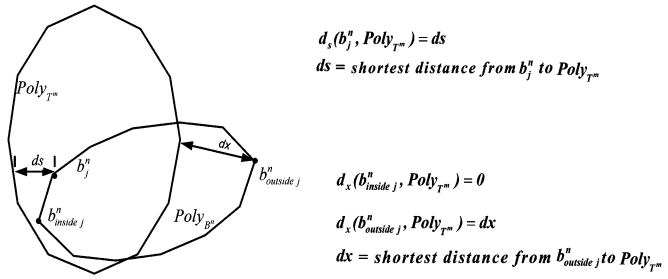



Fig. 3. Illustration to explain the computation of distances $d_x$ and $d_s$.

The match measure $D_H$ for matching a blob $B^n$ and an object $T^m$ is defined as

$$D_H(B^n, T^m) \triangleq \sum_{l=1}^{L} \left| b^n_{H_l} - t^m_{H_l} \right| \tag{5}$$

and is a measure of similarity of two patterns with respect to intensity and color information. Each bin $H_l$ has three subbins corresponding to $Y$, $C_r$, and $C_b$ channels and $\left|b^n_{H_l} - t^m_{H_l}\right|$ is the sum of the absolute differences of the subbins in bin $H_l$. The value of $D_H$ when there is a match and when there is not, is usually found to differ by an order of 10. Hence it is easy to set an empirical threshold to decide whether a blob and an object match by the $D_H$ measure. We also tried the Bhattacharyya coefficient as measure of histogram similarity and found the results to be similar to sum of absolute difference. Therefore, for the sake of simplicity and efficient computation we chose to use sum of absolute difference.

### B. Object Matching With $D_s$

Here we consider matching objects with blobs using $D_S$ in the presence of merging and splitting. Let the set of objects being tracked be represented by $T^1, T^2, \ldots, T^m, \ldots, T^M$. A video frame can consist of several blobs, listed as $B^1, B^2, \ldots, B^n, \ldots, B^N$. A blob $B^n$ can arise from the following:

1) a single object;

$$D_X(B^n, T^m) \triangleq \frac{\sum_{j=1}^{J} d_x^2\left(b^n_{X_j}, Poly_{T^m}\right)}{(A_{T^m} + A_{B^n})\digamma}$$
$$d_x\left(b^n_{X_j}, Poly_{T^m}\right) \triangleq \begin{cases} 0, & \text{if } b^n_{X_j} \text{ is inside } Poly_{T^m} \\ \text{shortest distance of } b^n_{X_j} \text{ from } Poly_{T^m}, & \text{otherwise} \end{cases}$$
$$\digamma \triangleq \begin{cases} 0, & \text{if } A_{T^m} \bigcap A_{B^n} = 0 \\ 1, & \text{if } A_{T^m} \bigcap A_{B^n} > 0 \end{cases} \tag{4}$$

2) multiple objects merging together;
3) part of an object which has split into multiple blobs;
4) part of an object which has simultaneously merged with other objects and has also split to give rise to more than one blob.

The merging of two or more objects is expressed with an operator $\oplus$, i.e., $T^1 \oplus T^2 \oplus T^3 \oplus T^4$ denotes the merging of the four objects $T^1$, $T^2$, $T^3$, and $T^4$. The synthesized object $\overline{T}$ formed by merging of the objects will have a new convex hull, which is obtained from the points on the convex hull of the individual objects. Given $M$ objects, the total number of different ways in which a new synthesized object can be formed is $2^M - 1$. For example, for $M = 4$ the different possibilities for $\overline{T}$ are $\{T^1, T^2, T^3, T^4, T^1 \oplus T^2, T^1 \oplus T^3, T^1 \oplus T^4, T^2 \oplus T^3, T^2 \oplus T^4, T^3 \oplus T^4, T^1 \oplus T^2 \oplus T^3, T^1 \oplus T^2 \oplus T^4, T^1 \oplus T^3 \oplus T^4, T^2 \oplus T^3 \oplus T^4, T^1 \oplus T^2 \oplus T^3 \oplus T^4\}$. Notationally, any possible synthesized object $\overline{T}$ formed from merges can be written as

$$\overline{T} = T^{m(1)} \oplus T^{m(2)} \oplus \cdots \oplus T^{m(p)} \oplus \cdots \oplus T^{m(P)} \tag{6}$$

where $P$ of the existing objects have merged and $m(p)$ denotes the index of the objects used in synthesis of $\overline{T}$. For each $\overline{T}$ the match measure $D_S(\overline{T}, B^n)$ can be computed to find the best match such that

$$\hat{P}, \hat{m}(p) = \arg \min_{P, m(1), \ldots, m(p)} D_S(\overline{T}, B^n). \tag{7}$$

Though the formulation of $D_S$ as a match measure was intuitive and simple, it does not possess properties that can be exploited to allow efficient computation of $\hat{P}$ and $\hat{m}(p)$; all possible merges of objects have to be exhaustively considered to find the optimal solution. Hence, solving the optimization problem expressed in (7) with $D_S$ as the match measure is computationally prohibitive to solve in real time. The order of computation to find the optimal match when targets merge is $O((2^M - 1) \times N)$.

Likewise, the optimal solution to the problem of splitting using the $D_S$ match measure can be addressed by exhaustively merging the $N$ blobs, $B^n$ in all possible combinations and computing their match measure with the different objects. The new synthesized pattern formed by merging different blobs is $\overline{B} = B^{n(1)} \oplus B^{n(2)} \oplus \cdots \oplus B^{n(p)} \oplus \cdots \oplus B^{n(P)}$. The problem is to compute all possible $\overline{B}$ by changing $P$, the number of blobs, and $n(p)$ the indexes of the blobs. Again due to lack of suitable properties in $D_S$, computing the optimal solution in case of splits requires a computational complexity of $O((2^N - 1) \times M)$. Sometimes the total number of blobs can be quite large due to the presence of clutter, making $2^N$ a very large number. The total complexity to handle both splits and merges is $O((2^M - 1) \times N) + O((2^N - 1) \times M)$.

Next we show that the new match measure $D_X$, which has the properties 1–4 listed in Section II-A and the property that $D_X(B^n, T^m) = 0$ for a match (both, when $B^n$ is coincident with $T^m$, or is within it), can be exploited to implement the optimization using DP and solve the problem efficiently in $O(M \times N)$. This resulting computational savings to solve the data association problem makes real-time implementation possible for MTT.

### C. Dynamic Programming Strategy for Efficient Matching With $D_X$

The use of DP requires that the problem is decomposable into subproblems such that the optimal solution of these subproblems can be combined together to find the optimal solution for the main problem. The properties of the distance function $D_X$ facilitates the use of the DP strategy. To illustrate the idea, we consider Fig. 4 where four objects $T^{m(1)}$, $T^{m(2)}$, $T^{m(3)}$, and $T^{m(4)}$ in frame $k$ are being tracked with Kalman filter based trackers (or other trackers which predict and estimate the states of the objects). The predictions for the objects' shape and position in the next frame $k+1$ are denoted as $\hat{T}^{m(1)}$, $\hat{T}^{m(2)}$, $\hat{T}^{m(3)}$, and $\hat{T}^{m(4)}$. These predicted objects merge to give rise to a new synthesized object $\overline{T}$ in frame $k+1$. If a blob $B^n$ is indeed due to the merger of these four objects in frame $k+1$ then $\overline{T}$ will coincide with $B^n$ and hence $D_X(\overline{T}, B^n) = 0$. Now if $D_X$ is computed separately for $\hat{T}^{m(1)}$, $\hat{T}^{m(2)}$, $\hat{T}^{m(3)}$, and $\hat{T}^{m(4)}$ with $B^n$, then each of them will also be equal to zero. This is because all the control points of these predicted objects lie within the polygon formed by the control points of $\overline{T}$ which is a match with $B^n$. Therefore, the problem of matching when objects merge to form a new synthesized object $\overline{T}$ is decomposed to the problem of finding all objects $T^{m(1)}, T^{m(2)}, \ldots, T^{m(P)}$, which individually match the blob, $B^n$, by the match measure $D_X$. The match measure $D_X(\hat{T}^m, B^n)$ is designed to give a match when polygon $Poly_{\hat{T}^m}$ lies within $Poly_{B^n}$ or is coincident with it. This property is lacking in $D_S$ and hence we cannot apply DP with $D_S$ to improve the efficiency of solving the matching problem.

As discussed in Section II-B when the objects split to give rise to more blobs than the objects in the scene, then the same DP strategy as above can be applied by exchanging the roles of $T^m$ and $B^n$ in the DP table.

### D. Rules for Developing the DP Table

The details of the DP strategy for optimally solving the matching problem in the presence of multiple splits and merges is explained with the rules for computing the DP table. The DP method used here to efficiently compute the matches for data association is somewhat similar to that used for string matching, and both follow the basic principle of DP, i.e., *computing the partial solutions, storing them and then combining the optimal partial solutions to compute the final optimal solution*. Since our method based on DP is similar to that used for string matching, we first illustrate using an example of string matching.

The string matching problem computes minimal cost deletion, insertions and reversals for converting a string to another string [36]. An application of this is in spell checking where the method not only finds if a word is misspelt but also suggests alternate correct words for substitution. Let a misspelt word "$B_{\mathrm{str}}$" be "becaase" and the correct word "$T_{\mathrm{str}}$" be "because." The DP table for the solution to this problem can be constructed as shown in Table II (the details of constructing such DP tables can be found in [37], for example). A table of width $length(B_{\mathrm{str}}) + 1$ and height $length(T_{\mathrm{str}}) + 1$ is

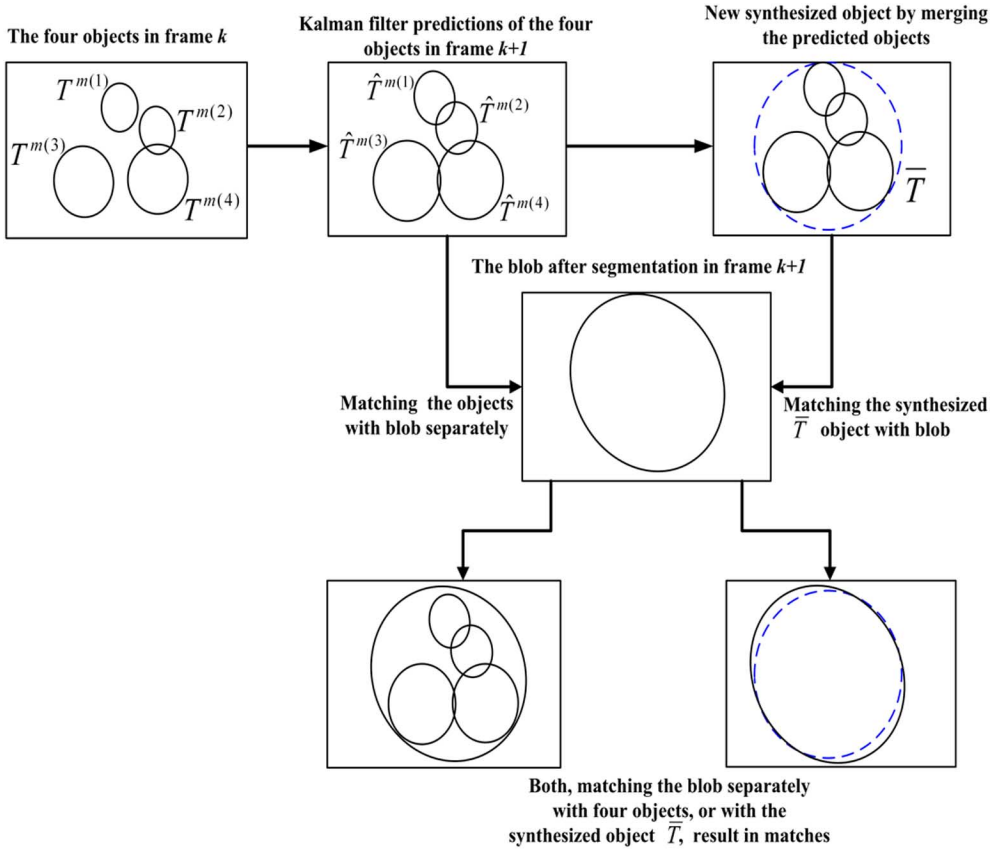



Fig. 4. Object matching principle using $D_X$. A synthesized object $\overline{T} = \hat{T}^{m(1)} \oplus \hat{T}^{m(2)} \oplus \hat{T}^{m(3)} \oplus \hat{T}^{m(4)}$ is a match with blob $B^n$ by the measure $D_X$ when $\hat{T}^{m(1)}$, $\hat{T}^{m(2)}$, $\hat{T}^{m(3)}$, and $\hat{T}^{m(4)}$ all individually match with $B^n$. This decomposition enables the use of DP strategy to speed up computations.

TABLE II
DP TABLE FOR MATCHING TWO STRINGS "BECAUSE" AND "BECAASE" AND FINDING THE MINIMUM NUMBER OF CHANGES TO CONVERT THE HORIZONTAL STRING TO THE VERTICAL STRING

$y \rightarrow$

| $x \downarrow$ | | | $b$ | $e$ | $c$ | $a_1$ | $a_2$ | $s$ | $e$ |
|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $b$ | 0 | | $\nwarrow$ 1 | $\leftarrow$ 1 | $\leftarrow$ 1 | $\leftarrow$ 1 | $\leftarrow$ 1 | $\leftarrow$ 1 | $\leftarrow$ 1 |
| $e$ | 0 | | $\uparrow$ 1 | $\nwarrow$ 2 | $\leftarrow$ 2 | $\leftarrow$ 2 | $\leftarrow$ 2 | $\leftarrow$ 2 | $\leftarrow$ 2 |
| $c$ | 0 | | $\uparrow$ 1 | $\uparrow$ 2 | $\nwarrow$ 3 | $\leftarrow$ 3 | $\leftarrow$ 3 | $\leftarrow$ 3 | $\leftarrow$ 3 |
| $a$ | 0 | | $\uparrow$ 1 | $\uparrow$ 2 | $\uparrow$ 3 | $\nwarrow$ 4 | $\nwarrow$ 4 | $\leftarrow$ 4 | $\leftarrow$ 4 |
| $u$ | 0 | | $\uparrow$ 1 | $\uparrow$ 2 | $\uparrow$ 3 | $\uparrow$ 4 | $\uparrow$ 4 | $\uparrow$ 4 | $\uparrow$ 4 |
| $s$ | 0 | | $\uparrow$ 1 | $\uparrow$ 2 | $\uparrow$ 3 | $\uparrow$ 4 | $\uparrow$ 4 | $\nwarrow$ 5 | $\leftarrow$ 5 |
| $e$ | 0 | | $\uparrow$ 1 | $\uparrow$ 2 | $\uparrow$ 3 | $\uparrow$ 4 | $\uparrow$ 4 | $\uparrow$ 5 | $\nwarrow$ 6 |

used. $\text{score}(x, y)$ stores the numerical score for each cell and $\text{arrow}(x, y)$ stores an arrow symbol(e.g, $\boxed{\uparrow}$) for each cell.

for $x \leftarrow 1$ to $\text{length}(T_{\text{str}})$ $\text{score}[x, 0] \leftarrow 0$
  for $y \leftarrow 0$ to $\text{length}(B_{\text{str}})$ $\text{score}[0, y] \leftarrow 0$
  for $x \leftarrow 1$ to $\text{length}(T_{\text{str}})$
    do for $y \leftarrow 1$ to $\text{length}(B_{\text{str}})$
      do if $T_{\text{str}}(x) = B_{\text{str}}(y)$
        then $\text{score}[x, y] \leftarrow \text{score}[x-1, y-1] + 1$
          $\text{arrow}[x, y] \leftarrow \boxed{\nwarrow}$
        else if $\text{score}[x-1, y] \geq \text{score}[x, y-1]$
          then $\text{score}[x, y] \leftarrow \text{score}[x-1, y]$
            $\text{arrow}[x, y] \leftarrow \boxed{\uparrow}$
          else $\text{score}[x, y] \leftarrow \text{score}[x, y-1]$
            $\text{arrow}[x, y] \leftarrow \boxed{\leftarrow}$

After the construction of the table, string matching and correction can be done. For this, one has to move along the arrows starting from the right and bottom most cell of the DP table. The score of this cell gives the total number of character matches found between strings "$T_{\text{str}}$"and "$B_{\text{str}}$." The arrow $\boxed{\nwarrow}$ indicates a match and the next cell to move to is the cell pointed to by the arrow in the present cell. In this table, starting from the location $(e, e)$ one moves along the diagonal arrow $\boxed{\nwarrow}$ and reaches location $(u, a_2)$, which is filled with $\boxed{\uparrow 4}$. The symbol $\boxed{\uparrow}$ at location $(u, a_2)$ is the indicator for the step of insertion and the letter to be inserted comes from the row location. Thus 'u' has to be inserted in $B_{\text{str}}$ at this location in the process of changing it to $T_{\text{str}}$. After this, moving along the arrows one reaches location $(c, a_1)$, which is filled with $\boxed{\leftarrow 3}$. The symbol $\boxed{\leftarrow}$ is the indicator for the step of deletion and the character to be deleted

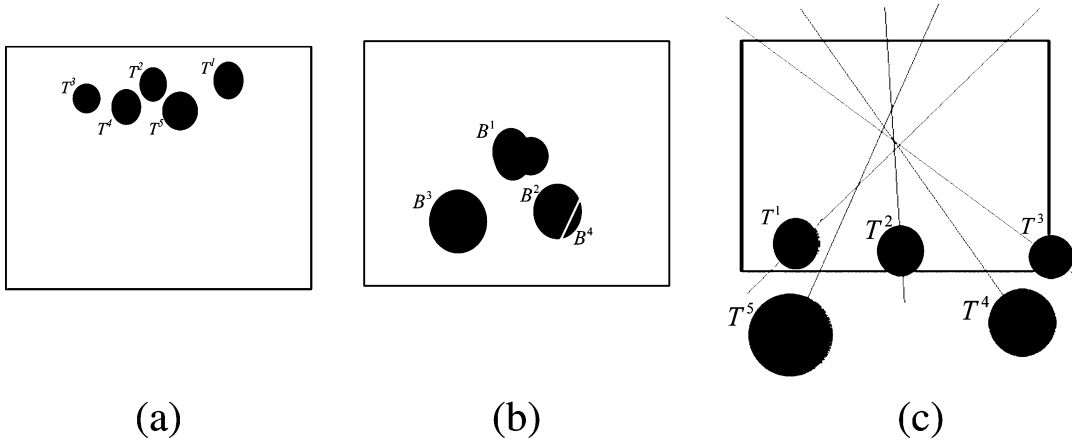



Fig. 5. (a) Five objects $T^1$, $T^2$, $T^3$, $T^4$, $T^5$ being tracked. (b) Objects $T^1$, $T^2$, and $T^3$ have merged to give rise to blob $B^1$ and objects $T^4$, $T^5$ have crossed paths giving rise to blobs $B^2$, $B^3$, respectively. Also object $T^4$ has split to give rise to two blobs, $B^2$ and $B^4$. (c) Tracks of the objects to clearly visualize their paths, and how merges and splits occur.

comes from the column location. Thus 'a' has to be deleted from $B_{\text{str}}$ in the process of making it the same as $T_{\text{str}}$.

We construct and use a similar DP table to obtain the optimal matches for data association given a set of predicted objects $T^1, T^2, \ldots T^M$ (here $\hat{T}^m$ are simply written as $T^m$ for clarity of notation) and the segmented blobs $B^1, B^2, \ldots B^N$ in a frame. We specify the indexes of the blobs and arrange the objects in the DP table by enumerating them sequentially as they occur from left to right and top to bottom in image frames. First, we state the rules for building the DP table and then illustrate its use for matching with an example. Following are the rules for moving in the DP table from any cell $(m, n)$, when we are solving for the case when multiple targets merge to give rise to one measurement.

1) Arrange the different objects being tracked $(T^1, T^2, \ldots T^M)$ vertically and the blobs $(B^1, B^2, \ldots B^N)$ horizontally for construction of the DP table. Initialize the score of all cells in the zeroth row and column of the DP table to zero.
2) Scan the cells in the DP table from left to right and top to bottom. Initial value of $m$ and $n$ is zero. At a location $(m+1, n+1)$, compute the match measures $D_X(T^{m+1}, B^{n+1})$ and $D_H(T^{m+1}, B^{n+1})$. If both match measures indicate a match then increment the score of cell $(m, n)$ and store it in cell $(m+1, n+1)$ along with a diagonal arrow symbol $\boxed{\nwarrow}$.
3) At a cell $(m+1, n+1)$ if $D_X$ indicates a match but $D_H$ does not, then this is considered as an indication of merge. Increment by one the score at cell $(m, n)$ and store it at $(m+1, n+1)$ with a diagonal arrow $\boxed{\nwarrow}$, and a label $\mathcal{M}$ to indicate that other matches through $D_X$ can be found in this column, as two or more objects have merged to give rise to a single blob, $B^{n+1}$. Scan this column to find all the possible matches with the measure $D_X$ and label the corresponding cells as $\mathcal{M}$.
4) If cell $(m+1, n+1)$ is not a match then examine cell $(m, n+1)$. If the score at this cell is greater than or equal to the score at cell $(m+1, n)$ then store this value at cell $(m+1, n+1)$ with a left arrow $\boxed{\uparrow}$ to indicate the cell where it came from. Otherwise, store the value from cell $(m+1, n)$ in $(m+1, n+1)$ with an up arrow $\boxed{\leftarrow}$ to indicate the cell from where the value came.
5) The above steps are repeated until the bottom right corner of the table is reached. Then from the bottom right of the table, moving along the arrow signs, the optimal matches are found along the diagonal arrows.

TABLE III
FORWARD DP TABLE FOR HANDLING MERGES OF OBJECTS. AS $T^1$, $T^2$, AND $T^3$ MERGE TO GIVE RISE TO $B^1$, ALL THREE OBJECTS INDIVIDUALLY MATCH WITH $B^1$ AND THEY ALL LIE IN THE SAME COLUMN INDICATING A MERGE. $B^3$ MATCHES WITH $T^5$, WHICH IS AN EXAMPLE OF ONE-TO-ONE MATCH AS THERE IS NO SPLIT OR MERGE OF THE OBJECT

$n \rightarrow$

| $m \downarrow$ | | $B^1$ | $B^2$ | $B^3$ | $B^4$ |
|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 |
| $T^1$ | 0 | $\nwarrow 1\mathcal{M}$ | $\leftarrow 1$ | $\leftarrow 1$ | $\leftarrow 1$ |
| $T^2$ | 0 | $\nwarrow 1\mathcal{M}$ | $\uparrow 1$ | $\uparrow 1$ | $\uparrow 1$ |
| $T^3$ | 0 | $\nwarrow 1\mathcal{M}$ | $\uparrow 1$ | $\uparrow 1$ | $\uparrow 1$ |
| $T^4$ | 0 | $\uparrow 1$ | $\leftarrow 1$ | $\leftarrow 1$ | $\leftarrow 1$ |
| $T^5$ | 0 | $\uparrow 1$ | $\leftarrow 1$ | $\nwarrow 2$ | $\leftarrow 2$ |

The DP table computed according to the above rules finds the matches of the blobs in the current frame with the existing objects. In the case where objects split to give rise to multiple blobs, the problem is solved by using a reverse DP table. The rules for computing the reverse DP table are similar, with the roles of $T$ and $B$ interchanged in the previous procedure.

Data association using the DP table in the presence of splits and merges is illustrated with an example. Fig. 5 (a) shows a frame in which five objects $T^1$, $T^2$, $T^3$, $T^4$, and $T^5$, are being tracked. Fig. 5(b) shows the blobs in a subsequent frame when the objects $T^1, T^2, T^3$ have merged to give one blob $B^1$ and objects $T^4$ and $T^5$ have crossed each other to give rise to blobs $B^2$ and $B^3$, respectively, and object $T^4$ has split to give rise to two blobs $B^2$ and $B^4$. Fig. 5(c) shows the actual tracks of the five objects to visualize their motion and how they gave rise to the different merges and splits. Table III shows the forward DP table obtained for matching the blobs to the objects. The

TABLE IV
REVERSE DP TABLE FOR HANDLING SPLITTING. $T^4$ HAS SPLIT INTO TWO BLOBS $B^2$ AND $B^4$ AND HENCE BOTH MATCHES APPEAR IN THE SAME COLUMN TO INDICATE THE SPLIT

$m\rightarrow$

| $n\downarrow$ | | $T^1$ | $T^2$ | $T^3$ | $T^4$ | $T^5$ |
|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 |
| $B^1$ | 0 | $\leftarrow 0$ | $\leftarrow 0$ | $\leftarrow 0$ | $\leftarrow 0$ | $\leftarrow 0$ |
| $B^2$ | 0 | $\leftarrow 0$ | $\leftarrow 0$ | $\leftarrow 0$ | $\nwarrow 1\mathscr{S}$ | $\leftarrow 1$ |
| $B^3$ | 0 | $\leftarrow 0$ | $\leftarrow 0$ | $\leftarrow 0$ | $\uparrow 1$ | $\uparrow 1$ |
| $B^4$ | 0 | $\leftarrow 0$ | $\leftarrow 0$ | $\leftarrow 0$ | $\nwarrow 1\mathscr{S}$ | $\nwarrow 2$ |

matches of $T^1$, $T^2$, and $T^3$ lie in one column indicating merge of these three targets to form one blob $B^1$. To handle the cases of splits the reverse DP table shown in Table IV is computed.

This efficient DP-based algorithm uses the $D_X$ and $D_H$ match measures to match objects to blobs in each frame and thus solve the data association problem. Once this is done, the object features need to be updated. This is done cooperatively with Kalman filters associated with each object. This is described in Section III.

## III. COOPERATIVE MTT

The above method of detecting merges and splits can be used in conjunction with any state estimation filter which can predict the location and shape of the objects being tracked. Here we briefly describe how two Kalman filters are used cooperatively with our efficient data association technique to track the position and shape states of each object. The separation of the position and shape is intentional because the object features are updated differently when the objects merge and split. Thus our system can be classified as a multiple model system [38]. This feature also makes our system more robust and efficient compared to other systems which use just one filter per track to track the object.

The position and motion of an object are tracked through the centroid of the ellipse modeling the object. With the fast processing rate of 25 fps we use a constant velocity model for the objects. The shape of the object is tracked through the $J$ control points representing the shape of the object. The motion of these control points is approximated by an affine motion model, which has been widely used in computer vision [39], [40]. The change of scale of the objects as they move away from or towards the camera is accounted for by the parameter $t_s^m$ of the $m$th object. Kalman filter details are given in the appendix.

The DP based algorithm uses the two match measures $D_X$ and $D_H$ to match the blobs $B^1, B^2, \ldots, B^N$ with the objects $T^1, T^2, \ldots, T^M$. $D_X$ is used in the algorithm to find the occurrence of splits and merges while $D_H$ combined with $D_X$ is used to infer one to one matches between objects and blobs when objects do not split or merge.

For one to one matches (with $D_H$ and $D_X$) the object features for shape, position and motion are updated by Kalman filter *estimates*. The shape features of objects which undergo splits are updated by Kalman filter *predictions* but their position and velocity are updated by Kalman filter *estimates*. The measurement for position and velocity update is the area weighted mean of all centroids of the different blobs which match the object with $D_X$. The shape, position and motion features of objects, which have merged to give rise to a new blob are updated by their Kalman filter *predictions*. The occurrence of simultaneous merges and splits is detected by occurrence of symbol $\mathcal{M}$ in the forward DP table and symbol $\mathcal{S}$ in the reverse DP table. In such a case the objects' position, velocity, and shape parameters are updated by the Kalman filter *predictions*.

From the above discussion we conclude that there are three types of matches possible. For each of these, a different method is used for updating the target parameters.

1) The objects which have not undergone merge or split, match their corresponding blob with both match measures $D_H$ and $D_X$. The motion, position, and shape attributes of these objects are updated by Kalman filter *estimates*.
2) For matches where an object has split into multiple blobs, the shape feature of the object is updated by Kalman filter *predictions* but position and motion are updated by Kalman Filter *estimates*.
3) The shape, position and motion features of objects, which have merged or have undergone simultaneous merges and splits are updated by their Kalman filter *predictions* only.

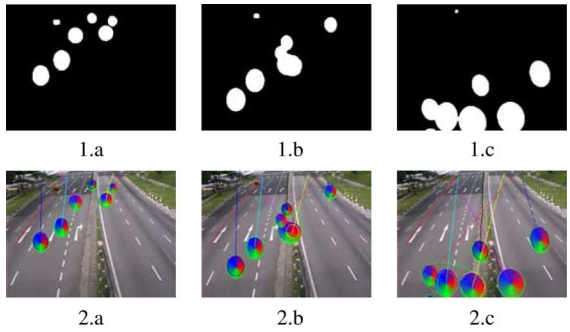



Fig. 6. Imges 1.a–1.c show the segmentation results for objects in the FoV and images 2.a–2.c show the tracking results. In image 1.a seven objects are being tracked. Appearance of a new object in the top right of image 2.b is correctly initialized for tracking. Images 1.b and 2.b show a case where four objects have merged into one blob. In the video all eight objects were properly tracked even as they underwent multiple merges and splits. Note that all the objects here are similarly colored so that a correspondence, template, or color histogram based tracking is likely to fail.

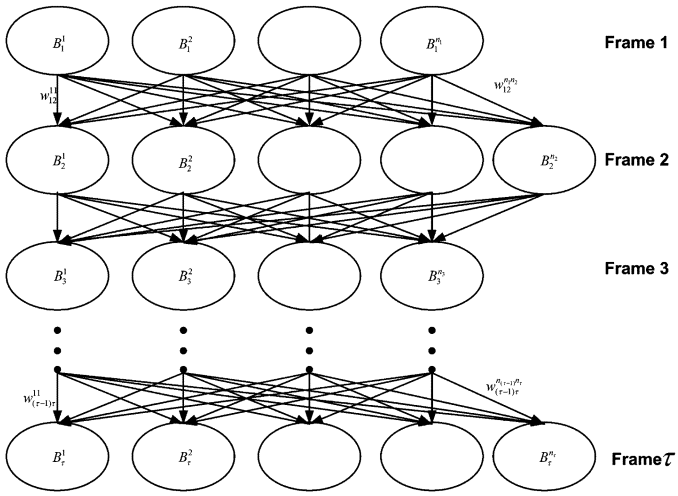



Fig. 7. Structure of the weighted directed graph used for initialization of tracking.

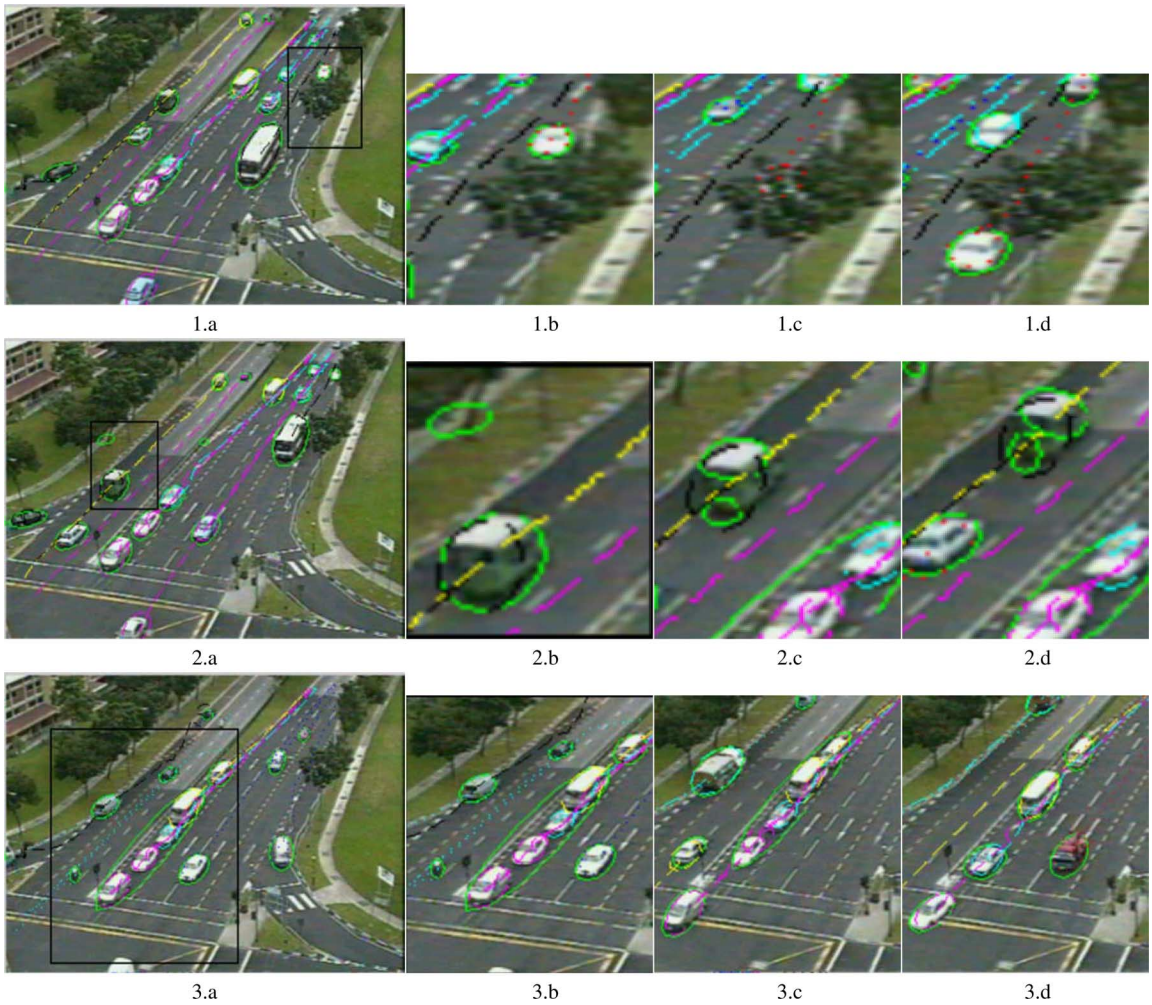



Fig. 8. Tracking results for vehicles. The black rectangle in images (1.a–3.a) are the ROIs. The corresponding images in (b-d) are zoomed versions of the ROIs at different time instants. Images 1.a–d show tracking results when an object gets completely occluded. The white car shown tracked with red dotted ellipse got completely occluded by the tree, but it was still tracked properly. Images 2.a-d show the tracking result when an object splits into multiple blobs due to segmentation errors. The grey van shown being tracked with black ellipse split into multiple blobs, but it was still tracked as a single vehicle. Images 3.a-d show the tracking result when 4–5 targets merge into one blob. Images 3.a and 3.b show tracking results in frame 635. In images 3.c (frame 733) and 3.d (frame 820) it can be seen that the proper tracks were maintained for each of the five targets which merged to give rise to one or two blobs. The color of the object tracks remain the same before the merge and after they separate.

The result of tracking as a cooperative effort between data association and Kalman filter based tracking is shown on a test image sequence in Fig. 6. The video was made by overlaying artificial objects on a real video. These images show the algorithm's ability to handle multiple merges and splits. In spite of many instances of merges, the objects' position and shape have been accurately tracked.[1]

## IV. Track Initialization

The results in Fig. 6 have been produced with the inclusion of a scheme to automatically initialize the tracking of a new object which appears in the camera FoV. Accurate initialization of the position, motion, and shape parameters of an object is an important step, which must be accomplished in the presence of clutter. There are two types of track initializations that need to be handled: 1) the initial bootstrapping stage and 2) when tracking is in progress. Weighted directional graphs are very useful for initializing MTT systems as they provide means to incorporate both spatial and temporal information of the objects in decision making. Graphs for object tracking and track initialization have been used in [10] and [41], respectively. In [41], the number of paths to be found is predetermined or user specified, while we determine the number of paths automatically from the graph. Furthermore, our system can handle the effects of missed detection or object merges by initializing tracks only when reliable measurements are available consistently over the past $\tau$ frames. This requirement makes the initialization accurate and the tracker stable.

Automatic initialization of object tracks for bootstrapping is done by using a weighted graph of the blobs in the first $\tau$ frames, as shown in Fig. 7. The attributes of each node in the graph are: 1. the frame number, 2. the centroid, 3. shape parameters, and 4. color histogram of the blob. Edges $E$ are present between nodes in successive frames as shown in Fig. 7. There are no edges between nodes whose difference in frame number is greater than

[1]Complete video with tracking results for demonstration is available at http://perception.i2r.a-star.edu.sg under "Cooperative tracking."

one. The weights of these edges is the sum of match measures $D_H$ and $D_X$ between the blobs

$$W_{(k-1)k}^{n_{(k-1)}n_k} = D_H\left(B_{(k-1)}^{n_{(k-1)}}, B_k^{n_k}\right) + D_X\left(B_{(k-1)}^{n_{(k-1)}}, B_k^{n_k}\right). \tag{8}$$

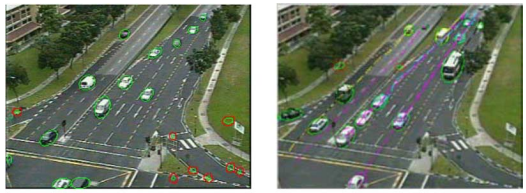
This definition of the weights of the edges ensures that the weights are nonnegative. The use of the graph and Dijkstra's shortest path algorithm inhibits clutter from getting initialized as objects for tracking. Clutter does not occur consistently and, therefore, with a sufficient number of layers in the graph we can isolate new consistent blobs from an object and initialize them for tracking. For bootstrapping, the nodes with frame number 1 are considered as source nodes and nodes with frame number $\tau$ are considered as destination nodes. For each source node the shortest path to all destination nodes is computed using Dijkstra's algorithm. The shortest path between a source and destination node in a graph is a directed simple path from source to destination with the property that no other such path has a lower weight. We have used Dijkstra's algorithm because its time complexity of $VE\log_d V$ (where $V$ is the number of nodes in the graph, $E$ is the number of edges in the graph and $d = E/V$) is less than the time complexity of Floyd's algorithm, which is $V^3$. Amongst these shortest paths, different paths have different sum-of-weights. From these paths, the path with smallest sum-of-weights is called the *path of least sum-of-weights*, and is considered to be a valid object track. The nodes of this path are then removed from the graph to yield a new graph with reduced number of nodes and edges. The same process is repeated for the new graph until there is no node from any *one* of the intermediate frames, the source or the destination frame in the graph or the sum of weights of the *path of least sum-of-weights* at any iteration is greater than a heuristic threshold. The heuristic threshold on the *least sum-of-weights* is chosen such that only targets which do not undergo splits and merges during the initial $\tau$ frames are initialized for tracking. If targets undergo splits and merges then the computed *least sum-of-weights* for their shortest path in the graph will be greater than the heuristic threshold and hence they will not be initialized for tracking. Thus, the initialization of tracking takes place only when reliable measurements are available in the last $\tau$ frames. We have used $\tau = 10$ in our experiments.

The second problem is the initialization of tracking for new objects, which enter the FoV or appear in the FoV due to resolution of occlusion, while tracking of other objects is in progress. To solve this problem, an attributed graph of blobs which have no match with the objects currently being tracked is maintained. *The path of least sum-of-weights* from the source nodes to the destination nodes is computed as described earlier. The source nodes are from the first layer formed by the unmatched blobs in frame $(k - \tau + 1)$, where $k$ is the current frame number. The destination nodes are the unmatched blobs in frame $k$. A new object track is confirmed by finding a *least sum-of-weights path* whose weight is less than the heuristic threshold. All the nodes of this path are removed from the attributed graph.

## V. Results

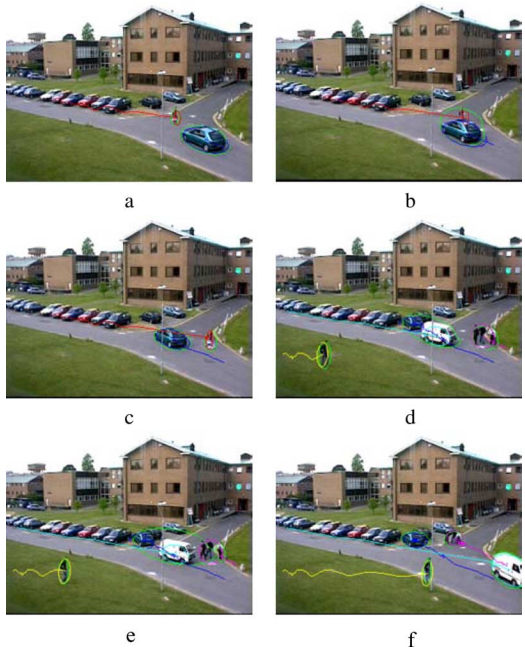
We show the results of tracking for both articulated and nonarticulated objects. In all the results the green ellipses indicate the bounding ellipse of the blobs obtained in every frame. An object and its track is shown with same color and style. We have tried to use different color and style for different objects. First, in Fig. 8, we show tracking results in a video obtained from a camera used for monitoring traffic. In the sequence there are: 1) instances when vehicles got partially or completely occluded [Fig. 8.1(a)–(d) shows an example of occlusion]; 2) cases of splits [Fig. 8.2(a)–(d) shows an example of splitting]; 3) instances of merges [Fig. 8.3(a)–(d) shows an example of four vehicles merging]; and 4) several instances of clutter (Fig. 9 shows examples of clutter).

The split and merge handling algorithm helps in maintaining the ID of objects even when they undergo splits and merges. The Kalman filter based tracker does a good job of tracking the vehicles even when they get partially or completely occluded. The graph based tracking algorithm correctly initializes the tracking of potential targets and not of clutter which occur many times in



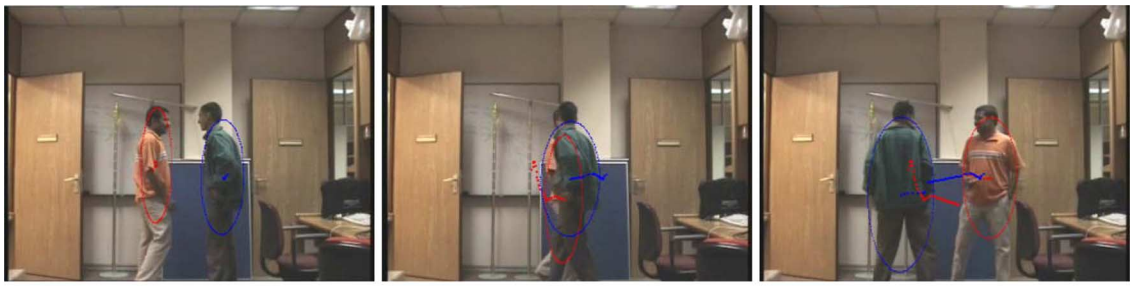
a b

Fig. 9. (a) Some examples of clutter where measurements are obtained even when there are no foreground objects at that location. We have highlighted the clutter by drawing boundaries around them. (b) Shows that clutter is not initialized for tracking.



a b c d e f

Fig. 10. Tracking results for both vehicles and pedestrians in an image sequence taken from the PETS2001 data set. Images (a)–(c) show tracking results when a pedestrian and a car merge and then separate. Images (d)–(f) show the tracking results for the difficult case of simultaneous merge and split of objects. In this case, the objects have been properly tracked.



a b c

Fig. 11. Tracking results for articulate objects in an indoor scene. The two persons are exchanging positions and both of them have been tracked properly in spite of the merge that occurs due to occlusion.

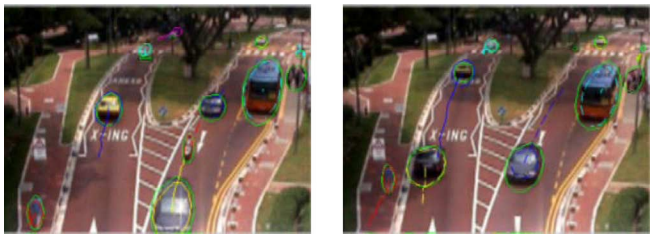


Fig. 12. Tracking of multiple objects with different types of vehicles and pedestrians in a difficult situation.

the segmentation results.[2] Simultaneous tracking of upto 17 targets in real time was achieved. In several different cases of splits, merges, and occlusion the tracking continued robustly without errors. Robust tracking (prediction) under occlusion continued reliably because the motion of the vehicle was learned properly before occlusion and when the vehicle reappeared after occlusion, the state of the object corresponded well with blob measurements. Also as the object is well in the interior of the frame, more frames without matching measurements can be tolerated before the object track is terminated, than when the object is near the frame boundary. There was also no incorrect initialization of targets in spite of the presence of clutter in the segmentation results.

In Fig. 10, we show the tracking results in an image sequence from the PETS2001 data set. Here also the merges and splits have been correctly handled and all the objects have been correctly tracked. In this test sequence a simultaneous occurrence of merge and split was also correctly handled.[2] Fig. 11 shows the tracking results for a video of articulated objects. It is difficult to track articulated objects because their shape change is difficult to track and their motion is difficult to learn and predict. However, here also we have been able to track the two persons correctly after they merge and separate.[2] Fig. 12 shows the tracking results for a large variety of objects in a video which includes pedestrians, motor-bike, cars, and buses.

Our system has been implemented in VC++ using the OpenCV library, to process PAL video digitized to a frame size of $352 \times 252$, on a 2.8-GHz Pentium IV workstation running Windows XP Professional operating system. We have used some look-up tables to speed up computations in the foreground segmentation algorithm, but no special hardware. We have been able to track 17 vehicles simultaneously at 25 fps. At present this system is being used to detect different types of behavior in traffic videos e.g., potential accident between pedestrians and vehicles [42].

[2]The complete tracking videos are available at http://perception.i2r.a-star.edu.sg under "Cooperative Tracking."

## VI. Conclusion

In this paper, we have addressed the problem of multiple target tracking in the presence of splits and merges that occur due to segmentation errors, similarity of objects' color with the background, objects occluding each other, or splits due to resolution of occlusion. We also presented a robust track initialization scheme for target track initialization for MTT. The robust real time tracking algorithm uses a novel geometric shape matching measure whose properties allow the use of DP to efficiently handle multiple splits and merges and maintain the ID of objects. The DP-based algorithm for associating the objects with the measured blobs significantly reduces the computation cost from exponential order to polynomial order, allowing development of a real time tracking system. When this data association technique is combined with Kalman filter based tracking, it is possible to preserve the labels of the objects even when they cross each other, or get completely or partially occluded by background or foreground objects. The method works well when the position and motion of the objects are predictable. The system will fail if the objects' motion changes significantly during occlusion. A weighted directional graph based technique was proposed to initialize the target tracks. The use of graphs helps incorporating spatial and temporal information together for decision making. The use of the graph and Dijkstra's shortest path algorithm inhibits clutter from getting initialized as objects for tracking. Clutter does not occur consistently and, therefore, graphs with a sufficient number of layers are able to isolate new consistent blobs and initialize them as objects for tracking. Results have shown that the system can handle very complex merging and splitting in cases where the motion and shape of the objects can be learned and reliably predicted. We have achieved real-time tracking of up to 17 targets simultaneously in real life videos.

## APPENDIX

Following [4], the standard Kalman filter equations are given below. We have used upper case bold for matrices and lower case bold for vectors.

$\boldsymbol{x}_k = \boldsymbol{A}_k\boldsymbol{x}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_{k1} + \boldsymbol{w}_{k-1}$ State equation
$\boldsymbol{z}_k = \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k$ Measurement equation
$p(\boldsymbol{w}) \sim N(0, \boldsymbol{Q})$ Process noise distribution
$p(\boldsymbol{v}) \sim N(0, \boldsymbol{R})$ Measurement noise distribution
$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_{k-1}$ State prediction equation
$\boldsymbol{P}_k^- = \boldsymbol{A}_k\boldsymbol{P}_{k-1}\boldsymbol{A}_k^{\boldsymbol{T}} + \boldsymbol{Q}$ Error covariance prediction equation
$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}_k^{\boldsymbol{T}}\left(\boldsymbol{H}_k\boldsymbol{P}_k^-\boldsymbol{H}_k^{\boldsymbol{T}} + \boldsymbol{R}\right)^{-1}$ Kalman gain equation
$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_k^-\right)$ State estimation equation
$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^-$ Error covariance estimation equation

In the above equations $\boldsymbol{A}_k$, $\boldsymbol{B}_k$, and $\boldsymbol{H}_k$ are the state, control and measurement matrices, respectively. $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, and $\boldsymbol{z}_k$ are the state, control and measurement vectors at step $k$. $\boldsymbol{w}_k \sim N(0, \boldsymbol{Q})$ is Gaussian process noise with covariance matrix $\boldsymbol{Q}$ and $\boldsymbol{v}_k \sim N(0, \boldsymbol{R})$ is the Gaussian measurement noise with covariance matrix $\boldsymbol{R}$.

$\hat{\boldsymbol{x}}_k^-$ and $\hat{\boldsymbol{x}}_k$ represent the state prediction and state estimate given measurement $\boldsymbol{z}_k$, respectively, at step $k$.

$\boldsymbol{P}_k^-$ and $\boldsymbol{P}_k$ are the covariance matrices of the *a priori* and *a posteriori* estimation errors, respectively, and $\boldsymbol{K}_k$ is the Kalman gain.

*A) Position and Motion Tracking:* For position and motion tracking the state vector of the $m$th object $T^m$ is $\boldsymbol{x}_k^m = \left[t_{x_{c_k}}^m, t_{y_{c_k}}^m, t_{u_{c_k}}^m, t_{v_{c_k}}^m\right]^T$, where $[t_{x_{c_k}}, t_{y_{c_k}}]^t$ is the centroid of the ellipse bounding the object and $[t_{u_{c_k}}, t_{v_{c_k}}]^T$ is its time derivative. The measurement vector for the $n$th blob $B^n$ is taken as:

$$\mathbf{z}_k^n = \left[b_{x_{c_k}}^n, b_{y_{c_k}}^n, \left(b_{x_{c_k}}^n - b_{x_{c_{k-1}}}^n\right), \left(b_{y_{c_k}}^n - b_{y_{c_{k-1}}}^n\right)\right]^T.$$

With these definitions we have $\boldsymbol{A}_k = \boldsymbol{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, and $\boldsymbol{H}_k = \boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, and $\boldsymbol{u}_k = \underline{0}$.

We estimate the process noise covariance matrix $\mathbf{Q}$ using the matching blob measurements obtained from the graph based initialization algorithm. When initializing an object's track by this method we have the last $\tau(\tau = 10)$ consistent blobs. From these measurements we compute the object's average velocity, and then using this constant velocity we compute the position of the object in each frame starting from the initial position. The error in position and velocity from the constant velocity model and those from the actual measurement is used to estimate the process noise covariance matrix $\boldsymbol{Q}$. The measurement noise covariance matrix $\boldsymbol{R}$ is estimated by a one time off-line process for an imaging setup. For this we use short video clips taken at different times of the day from a fixed camera. Manual segmentation of the objects is done on these video clips. The errors in the features from the manually segmented objects and features obtained by background segmentation is used to estimate $\boldsymbol{R}$. Thereafter, this $\boldsymbol{R}$ is used for tracking of all the objects in the operational videos from that imaging setup.

*B) Shape Tracking:* The shape of the $m$th object $T^m$ is modeled by the $J$ control points (in our experiments we used $J = 12$) on the ellipse bounding the object. The shape of $T^m$ is tracked through these control points. The motion of these control points is approximated by affine motion. Under this assumption the position of the $j$th control point $t_{X_{j_k}}^m$ at time instant $k$ can be written as

$$\begin{aligned} t_{X_{j_k}}^m &= t_{X_{c_{k-1}}}^m + t_{s_{k-1}}^m \cdot \left(t_{X_{j_{k-1}}}^m - t_{X_{c_{k-1}}}^m\right) + t_{v_{c_{k-1}}}^m \\ t_{s_k}^m &= t_{s_{k-1}}^m. \end{aligned}$$

The state vector for shape tracking is

$$\boldsymbol{x}_k^m = \left[t_{x_{1_k}}^m, t_{y_{1_k}}^m, t_{x_{2_k}}^m, t_{y_{2_k}}^m, \ldots t_{x_{J_k}}^m, t_{y_{J_k}}^m, t_{s_k}^m\right]^T$$

and measurement vector is

$$\boldsymbol{z}_k^n = \left[b_{x_{1_k}}^n, b_{y_{1_k}}^n, b_{x_{2_k}}^n, b_{y_{2_k}}^n, \ldots b_{x_{J_k}}^n, b_{y_{J_k}}^n, b_{s_k}^n\right]^T$$

where $b_{s_k}^n$ is computed as

$$b_{s_k}^n = \frac{\sum_{j=1}^{j=J}\sqrt{\left(b_{x_{j_k}}^n - b_{x_{c_k}}^n\right)^2 + \left(b_{y_{j_k}}^n - c_{y_{c_k}}^n\right)^2}}{\sum_{j=1}^{j=J}\sqrt{\left(b_{x_{j_{k-1}}}^n - b_{x_{c_{k-1}}}^n\right)^2 + \left(b_{y_{j_{k-1}}}^n - b_{y_{c_{k-1}}}^n\right)^2}}.$$

$\boldsymbol{A}_k^m$ is a $(2J+1)\times(2J+1)$ diagonal matrix, whose $(2J+1)$ diagonal elements are $\left\{t_{s_{(k-1)}}^m, t_{s_{(k-1)}}^m, \ldots, t_{s_{(k-1)}}^m, 1\right\}$; $\boldsymbol{B}_k^m$ is $(2J+1)\times 2$ matrix

$$B_k^m = \begin{bmatrix} t_{x_{c_{(k-1)}}}^m, & t_{u_k}^m \\ t_{y_{c_{(k-1)}}}^m & t_{v_{(k-1)}}^m \\ t_{x_{c_{(k-1)}}}^m, & t_{u_{(k-1)}}^m \\ t_{y_{c_{(k-1)}}}^m & t_{v_{(k-1)}}^m \\ \vdots, & \vdots \end{bmatrix}$$

and $\mathbf{u}_{(k-1)}^m$ is a $2\times 1$ vector $= \begin{bmatrix} (1 - t_{s_{(k-1)}}^m) \\ 1 \end{bmatrix}$. $\boldsymbol{H}_k$ is a $(2J+1)\times(2J+1)$ identity matrix. The process noise covariance matrix $\boldsymbol{Q}$ and measurement noise covariance matrix $\boldsymbol{R}$ are estimated by a procedure similar to that followed in the filter for position and motion tracking.

## ACKNOWLEDGMENT

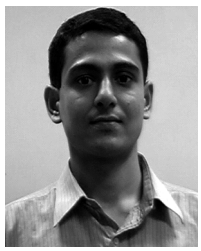
The authors would like to thank the anonymous reviewers for their detailed suggestions and comments which greatly improved the presentation of the paper.

## REFERENCES

[1] Y. Bar-Shalom, "Tracking methods in a multitarget environment," *IEEE Trans. Autom. Control*, vol. AC-23, no. 8, pp. 618–626, Aug. 1978.

[2] S. Blackman, *Multiple-Target Tracking With Radar Application*. Norwood, MA: Artech House, 1986.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82(D), pp. 35–45, 1960.
[4] G. Welch and G. Bishop, An Introduction to the Kalman Filter UNC-Chapel Hill, 2004, Tech Report TR 95-041.
[5] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A survey," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 1, pp. 103–123, Jan. 1998.
[6] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/nonGaussian bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.
[7] C. Rasmussen and G. D. Hager, "Probabilistic data association methods for tracking complex visual objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 560–576, Jun. 2001.
[8] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automatic Control*, vol. AC-24, no. 12, pp. 843–854, Dec. 1979.
[9] I. J. Cox, "A review of statistical data association techniques for motion correspondence," *Int. J. Comput. Vis.*, vol. 10, no. 1, pp. 53–66, 1993.
[10] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia, "Event detection and analysis from video streams," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 873–889, Aug. 2001.
[11] I. Cox and S. Hingorani, "An efficient implementation of Reid's mulitiple hypothesis tracking algorithm and its evaluation for visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, pp. 138–150, Feb. 1996.
[12] A. Genovesio and J. -C. Olivo-Marin, "Split and merge data association filter for dense multitarget tracking," in *Proc. 17th Int. Conf. Pattern Recognit.*, Aug. 2004, vol. 4, pp. 677–680.
[13] H. Tao, H. S. Sawhney, and R. Kumar, "Object tracking with Bayesian estimation of dynamic layer representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 75–89, Jan. 2002.
[14] O. Javed and M. Shah, "Tracking and object classification for automated surveillance," in *Proc. Eur. Conf. Comput. Vis.*, 2002, pp. IV: 343 ff–IV: 343 ff.
[15] N. Paragios and R. Deriche, "Geodesic active regions for motion estimation and tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 3, pp. 266–280, Mar. 2000.
[16] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people," *Comput. Vis. Image Understand.*, vol. 80, pp. 42–56, 2000.
[17] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Who, when, where, what: A real time system for detecting and tracking people," in *Proc. 3rd Int. Conf. Autom. Face Gesture Recognit. (FG'98)*, Apr. 1998, pp. 222–227.
[18] ——, "W4s: A real time system for detecting and tracking people in 2.5d," in *Proc. 5th Eur. Conf. Comput. Vis.*, Jun. 1998, pp. 877–892.
[19] J. García, J. M. Molina, J. A. Besada, and J. I. Portillo, "A multitarget tracking video system based on fuzzy and neuro-fuzzy techniques," *J. Appl. Signal Process. Special Issue on Advances in Intelligent Vision Systems: Methods and Applications*, no. 14, pp. 2341–2358, 2005.
[20] J. García, J. A. Besada, J. M. Molina, and J. Portillo, "Fuzzy data association for image-based tracking in dense scenarios," in *Proc. IEEE Int. Conf. Fuzzy Systems*, Honolulu, HI, May 2002, pp. 902–907.
[21] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
[22] A. Martelli and U. Montanari, "Optimal smoothing in picture processing: An application to fingerprints," in *Proc. IFIP Congr.*, 1971, vol. 71, pp. 173–178.
[23] V. A. Kovalevsky, *Image Pattern Recognit.*. New York: Springer-Verlag, 1980.
[24] ——, "Sequential optimization in pattern recognition and pattern description," in *Proc. IFIP Congr. 68*, 1968, pp. 1603–1607.
[25] H. Yamada, "Contour dynamic programming matching method and its application to handprinted chinese character recognition," in *Proc. 7th Int. Conf. Pattern Recognit.*, 1984, pp. 389–392.
[26] H. Yamada, C. Merritt, and T. Kasvand, "Recognition of kidney glomerulus by dynamic programming matching method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 5, pp. 731–737, Sep. 1988.
[27] H. Yamada and T. Kasvand, "Dynamic programming matching method for recognition of occluded, reflective and transparent objects with unconstrained background and illumination," in *Proc. 8th Int. Conf. Pattern Recognit.*, Oct. 1986, pp. 95–98.
[28] A. A. Amini, S. Tehrani, and T. Weymouth, "Minimizing the energy of active contours in the presence of hard constraints," in *Proc. 2nd Int. Conf. Comput. Vis.*, Tarpon Springs, FL, 1988, pp. 95–99.
[29] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, "Dynamic programming for detecting, tracking, and matching deformable contours," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 3, pp. 294–302, Mar. 1995.
[30] N. Ueda and K. Mase, "Tracking moving contours using energy minimizing elastic contour models," in *Proc. Eur. Conf. Comput. Vis.*, Santa Margherita, Italy, 1992, vol. 588, pp. 453–457.
[31] M. Hospital, H. Yamada, T. Kasvand, and S. Umeyama, "3-D curve based matching method using dynamic programming," in *Proc. Int. Conf. Comput. Vis.*, 1987, pp. 728–732.
[32] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. C-22, no. 1, pp. 67–92, Jan. 1973.
[33] M. Levine and D. Ting, "Intermediate level picture interpretation using complete two dimensional models," *Comput. Graph. Image Process.*, vol. 16, pp. 185–209, 1981.
[34] P. Kumar, S. Ranganath, and W. Huang, "Queue based fast background modelling and fast hysteresis thresholding for better foreground segmentation," in *Proc. 2003 Joint Conf. 4th ICICS and PCM*, Singapore, Dec. 2003, pp. 2A2.5–2A2.5.
[35] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 476–480, May 1999.
[36] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Doklady Akademii Nauk SSSR*, vol. 4, no. 165, pp. 845–848, 1965.
[37] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1992, pp. 314–319.
[38] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A survey," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 1, pp. 103–123, Jan. 1998.
[39] R. Cipolla and A. Blake, G. Sandini and S. Margherita, Eds., "Surface orientation and time to contact from image divergence and deformation," in *Proc. 2nd Eur. Conf. Comput. Vis.*, May 1992, pp. 187–202.
[40] Q. Zheng and R. Chellappa, "Automatic feature point extraction and tracking in image sequences for unknown camera motion," in *Proc. Int. Conf. Comput. Vis.*, Berlin, Germany, May 1993, pp. 335–339.
[41] J. K. Wolf, A. M. Viterbi, and G. S. Dixon, "Finding the best set of k-paths through a trellis with application to multitarget tracking," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 25, no. 3, pp. 287–296, Mar. 1989.
[42] P. Kumar, S. Ranganath, H. Weimin, and K. Sengupta, "Framework for real-time behavior interpretation from traffic video," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 3, pp. 43–53, Mar. 2005.



**Pankaj Kumar** (M'03) received the B.Tech degree in electrical and computer engineering from the Indian Institute of Technology, Delhi, India, and the M.Eng. and Ph.D. degrees from the National University of Singapore, Singapore.

He worked as a Research Fellow at the Institute of Infocomm Research, Singapore, from 2003 to 2006. Currently, he is working as a Research Fellow at the University of Adelaide, South Australia. His research interests include computer vision, artificial intelligence, behavior analysis, multicamera tracking, and video surveillance.

**Surendra Ranganath** received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, the M.E. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in electrical engineering from the University of California, Davis.

From 1982 to 1985, he was with the Applied Research Group at Tektronix, Inc., Beaverton, OR, where he was working in the area of digital video processing for enhanced and high-definition TV.

From 1986 to 1991, he was with the medical imaging group at Philips Laboratories, Briarcliff Manor, NY. In 1991, he joined the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, where he is currently an Associate Professor. His research interests are in digital signal and image processing, computer vision, and neural networks with focus on human-computer interaction and video understanding applications.

**Kuntal Sengupta** received the B.Tech degree from the Indian Institute of Technology, Kanpur, India, in 1990, and the M.S. and Ph.D. degrees from The Ohio State University, Columbus, in 1993 and 1996, respectively.

From 1996 to 1998, he worked as a Researcher at the Advanced Telecommunications Research (ATR) Laboratories, Kyoto, Japan. From 1998 to 2002, he was an Assistant Professor in the Electrical and Computer Engineering Department, National University of Singapore, Singapore. Presently, he is a Senior Algorithm Scientist with AuthenTec Inc., Melbourne, FL. His present research interests are in biometrics, HCI, video analysis, and multimodal fusion.

Dr. Sengupta received the Siemens Best Paper Award at IEEE CVPR 1993.

**Huang Weimin** received the B.Eng. degree in automation, and the M.Eng and Ph.D. degrees in computer engineering from Tsinghua University, Beijing, China, in 1989, 1991, and 1996, respectively.

He is a Research Scientist with the Institute for Infocomm Research, Singapore. He has worked on research of handwriting signature verification, biometrics authentication, and audio/video event detection. His current research interests include pattern recognition, image processing, computer vision, human-computer interaction, and statistical learning.