Globally Optimum Multiple Object Tracking

Ismail Oner Sebe, Suya You, Ulrich Neumann
Integrated Media Systems Center
University of Southern California, Los Angeles, CA
90089-0781
{iosebe | suyay | uneumann}@graphics.usc.edu

ABSTRACT

Robust and accurate tracking of multiple objects is a key challenge in video surveillance. Tracking algorithms generally suffer from either one or more of the following problems, excluding detection errors. First, objects can be incorrectly interpreted as one of the other objects in the scene. Second, interactions between objects, such as occlusions, may cause tracking errors. Third, globally-optimum tracking is hard to achieve since the combinatorial assignment problem is NP-Complete. We present a modified Multiple-Hypothesis Tracking algorithm, MHT, for globally optimum tracking of moving objects. The system defines five states for tracked objects: appear, disappear, track, split, and merge, and these states cover all the interactions of object pairs. After the detection of objects in the current frame, a resemblance matrix is computed for every object pair. We convert the two-dimensional resemblance matrix into a three-dimensional state-likelihood structure and use a MHT technique to solve the state-assignment problem in 3D. This prevents incorrect assignments due to local minima in the assignment process. Moreover, the method models occlusion cases with the split and merge states. Finally, this method approximates a globally optimum state assignment in polynomial time complexity.

Keywords: Real-time tracking, multiple objects tracking, multiple hypothesis tracking, occlusion tracking, combinatorial assignment.

1. INTRODUCTION

Visual tracking is becoming a key feature of surveillance systems. Some recent research examples are the Distributed Interactive Video Array (DIVA) developed at UCSD [Hall 02], the VideoFlashlight system developed at Sarnoff Corporation [Kumar 00], and the Video Surveillance and Monitoring (VSAM) project at CMU [Kanade 98]. Tracking is increasingly important, in part due to the rapid increase in the number of video cameras within surveillance systems. Unfortunately, large arrays of video cameras become unmanageable since operators are limited in the number of images they can observe effectively. However, automatic detection and tracking of moving objects from video can address this problem. Tracking techniques range from simple Kalman filters to Markov Chain Monte Carlo (MCMC) methods for human tracking [Tao 04]. Regardless of the surveillance method used, tracking continues to be problematic. Many of the current algorithms, several of which are surveyed in the next section, address sub-problems such as people and car tracking.

This paper presents a real-time object tracker that handles object interactions such as occlusions. We abstract the tracking problem to a matching of detected objects. Our formulation does not assume multiple cameras, knowledge of camera parameters, or ground plane position. Furthermore, the method is not limited to a specific object type such as humans or vehicles. Our tracking formulation can be used with any detection method (such as foreground extraction) or matching criterion. This allows each part of the tracking system to be modeled as separate and independent blocks. Our method differs from other tracking solutions in that we explicitly model interaction among or between objects and then handle these situations in a global sense.

The remainder of this section presents a survey of the relevant methodologies. Section 2 describes the assignment problem and its mathematics. Section 3 describes our algorithm, and various test results are given in section 4 along with a discussion of the algorithm behavior.

1.1 Tracking literature

Tracking has been one of the main topics in computer vision in the last couple of decades. Although a summary of the whole area of tracking is not possible, we group and summarize some of the related approaches.

Tracking algorithms can be divided into the methods that model backgrounds, such as [Harveille 01], [Elgammal 02], [Toyama 99], [Stauffer 99], [McKenna 99], and the ones that do not, such as [Isard 98], [Comaniciu 00,03]. Methods that do not model backgrounds generally rely on the foreground statistics or properties. For example, [Isard 98] learns the movement behavior of the foreground objects and uses this movement to track the objects. [Comaniciu 00, 03] uses the color histogram of the foreground object in order to distinguish it from the background. Although these methods are applicable in some cases, many tracking problems require explicit modeling of the background. A Mixture of Gaussians (MOG) over the RGB color space is a powerful method to estimate the background when the camera is fixed [Harveille 01], [Stauffer 99]. An HSV color space for MOG is also presented in [McKenna 99]. [Kang 03] extended this method to handle arbitrary camera motion. [Elgammal 02] uses a non-parametric kernel density estimator for background estimation. An extensive survey on background estimation and a combined method is presented in [Toyama 99].

Once the foreground and background are separated, the foreground is split into groups using criteria such as previous observation and connectivity. We classify these methods, according to the prior knowledge they impose over the objects they track, into the three following groups: full-model based, semi-model based, and non-model based approaches. Full-model based algorithms use an explicit 3D model of the objects, allowing the system to search for the transformation that maps the model into the image sequence. These systems can only track modeled objects, commonly humans. The system developed by [Remagnino 97] uses a static model for vehicles and a dynamic model for people. Although this system can track people and vehicles, it is still limited by the available models. [Wren 97] uses a part-based human body model to track and classify people. Their "PFinder" method tracks one person at a time and distinguishes between poses. PFinder is also used for automatic sign language conversion to English over a limited vocabulary.

Semi-model based systems track a specific type of object using an approximate model. A common example of this is the use of ellipses for people. [Zhao 04] developed a full-body people tracker in this category, combining a head-tracker, human segmentation from foreground blobs, shadow removal, and movement analysis to aid tracking. The algorithm finds and tracks people under occlusion, and classifies their actions, such as walking, running, or standing. Another full-body human detector is presented in [Elgammal 02], employing color distributions of foreground objects and ellipse-fitting in foreground blobs. [Kang 03] tracks full body people using an appearance model of the human body. The head, torso, and legs are modeled with respect to the centroid of the detected blob and tracked appropriately. [Chen 01] tracks human faces by estimating their position with ellipses, employing a joint probability data association filter (JPDAF)-based hidden markov models (HMM) for tracking the contour of the ellipse. [Tao 00] uses ellipse models to track vehicles in aerial images using generalized expectation maximization (EM) algorithm in a maximum a posteriori probability (MAP) estimation framework.

Non-model based tracking is generally used for tracking various types of objects at the same time. This group of trackers makes no prior assumptions about the shape, appearance, or type of objects in the scene. Clearly, this generality often leads to less-robust tracking. The previous two groups use models to convert the tracking problem into likelihood maximization. Lacking any models, the tracker either requires manual foreground labeling, or identifies foreground objects as an arbitrary shape. [Comaniciu 00, 03] uses a manual labeling of the foreground and minimizes a matching function with a mean-shift algorithm. The method handles non-rigid object movement as well as camera movement. [Isard 01] uses statistically-learned foreground and background models to do segmentation and uses these arbitrary masks for tracking. The system requires modeling the background and foreground, which are not always available or practical. [McKenna 99] uses a rectangular box that changes size to track arbitrary objects. The system uses a matching method in HSV color space. [Koller 94] developed a non-model based tracker that models objects with their convex hulls. Although the system is non-model based, it is applied to a specific set of objects, cars on a highway, and requires camera parameters and ground plane location. Although this algorithm is designed for a specific purpose, it resembles our system in how it handles the tracking of objects.

A quick review of the above systems quickly reveals that none of these methods is suitable for automatic tracking of

different groups of objects at the same time, such as people, cars, bikes, etc. Although a fully general tracker is a difficult problem, a method that can quickly be modified to accommodate a wide range of objects appears feasible. With this motivation, we developed a multiple object tracker using a modified MHT algorithm (Combinatorial MHT, or CMHT). The algorithm estimates a resemblance matrix between the already tracked objects and the newly detected objects. A state-likelihood matrix is calculated from this resemblance matrix. There are five object states: appear, disappear, track, split, and merge. Lastly, the CMHT method approximates the globally optimum assignment in polynomial time.

2. Combinatorial Assignment Problem

In this section, we present the assignment problem and various proposed solutions. In Section 3, we further extend the assignment problem to handle additional states of tracking such as appear, disappear, split and merge. Multiple objects tracking problem is a more general form of the assignment problem.

2.1 Assignment problem

Assignment problem can be stated as follows: "Given ***n*** known (e.g. previously tracked) objects and ***n*** new (e.g. currently detected) objects, find the best matching of this n-tuple for a given similarity measure". We will first state the problem mathematically and then analyze its complexity.

Function '***f*** ', resemblance (similarity) function, is a scaled matching criterion. For example, ***f*** (o_1, o_2) returns one if two objects are exactly same, and zero if they are completely different. A resemblance matrix ***R*** is estimated using resemblance function on every object pair:

$$R(i, j) = f(g_i^1, g_j^2), \tag{1}$$

where g_i^1 stands for the i^{th} previously tracked object and g_j^2 stands for the j^{th} newly detected object. Resemblance matrix R is an n_xn square matrix. A set of assignments (matches) is valid if every object in group one has a distinct match in group two, i.e. the mapping is a bijection, one-to-one and onto. Such a mapping corresponds to a permutation matrix. The indices of a permutation matrix are either zero or one, and there is exactly one one in every row and column. An example for resemblance matrix and permutation matrix is as follows:

0.9	0.2	0.3
0.1	0.8	0.7
0.1	0.5	0.2

(a) R_{3x3}

1	0	0
0	0	1
0	1	0

(b) P_{3x3}

1	0	0
1	0	0
0	0	1

(c) P_{3x3}

Figure 1: (a) is a sample 3x3 resemblance matrix. (b) is a valid permutation matrix. (c) is an invalid permutation matrix

The optimum assignment problem is then formulated as follows:

$$\arg\max_{P} \left(\sum_{\forall i, j} R(i, j).P(i, j)\right) \tag{2}$$

,subject to P is a permutation matrix. There are n! different permutation matrices for a n-pair. Furthermore the assignment problem is incapable of adding or deleting objects from tracking. In assignment problem, every object has only one possible state: "track". For assignment problem to model appearing and disappearing of the objects as well as tracking, the sum of the rows and columns of matrix P should be allowed to be not only one but also zero.

Another way of stating this problem is "how to choose a subset of size *h* out of the group one of size *n* and a corresponding subset of size *h* out of the group two of size *m*, such that these two subsets form an optimum matching". This gives us (***h***) matches, (***n***-***h***) disappeared objects and (***m***-***h***) appearing objects. In this new extended assignment problem, objects have three possible states: "track", "appear", and "disappear". The total search space for this case

is $\binom{m}{h}.\binom{n}{h}.h!$.

2.2 Suggested solutions for combinatorial assignment problem
Assignment problem is a well-known combinatorics problem. In this subsection, we summarize some of the proposed solutions to the assignment problem.

[Gary 79] uses linear programming approach to search for the optimum assignment by maximizing the sum in (2) with respect to P being a permutation matrix. Although this may solve the basic assignment problem, it is computationally not efficient and not extendible to more complicated tracking problems.

Another algorithm is the Softassign method [Gold 96, Guo 01]. Softassign is an iterative method that enforces the constraints of a permutation matrix on the resemblance matrix. The proposed method rescales the sums of the rows and the columns of R to one so that matrix R will converge to a permutation matrix. This algorithm has a very low asymptotic computational complexity. To our best knowledge, there has not been any method that generalizes these constraints to handle more complicated assignment problems that allows objects to split and merge.

Finding the optimum assignment by searching over all possibilities is computationally too extensive. Multiple hypothesis tracking (MHT) however is capable of finding **sub**-optimal solutions to the combinatorial assignment problem in polynomial time [Reid 79, Cox 96]. The proposed algorithm is successfully applied to the tracking of multiple corners from video [Cox 96]. This method approximates the optimum solution by tracking trees, pruning (N-scan back) and spatial division. MHT is similar to running multiple greedy search algorithms at different greed levels. The algorithm is able to jump over local minimums via choosing some sub-optimal hypotheses rather than the most optimum one at every step. Moreover, their system is able to not only track corners but also able to add and remove corners from the tracking.

In this paper, we extend this idea to track arbitrary objects even under occlusion. This adds two new states to the tracking algorithm, split and merge. Merge happens when one object occludes another and split happens when one blob becomes two blobs. The state of the objects enforces the system to choose either the estimate from the current state or the data from the current frame. For example, if an object is in the merging state then the data extracted from the image is discarded and the estimate from the current state is used instead. The addition of split and merge states increases the complexity of the overall algorithm. In particular, complexity is $\binom{m}{h}.\binom{n}{h}.(h!+2\binom{h}{2}!h!)$. Additional states introduce the second term in parenthesis and are asymptotically larger than the first term. We present a method to approximate the global optimum via changing our two dimensional **R** matrix into a three dimensional state-likelihood matrix **C** and use a modified MHT algorithm (combinatorial MHT, or CMHT) to search for the solution in polynomial time.

3. Tracking with Combinatorial MHT

3.1 Overall method
Our tracking algorithm uses an object detection algorithm outlined in [Sebe 04]. The detection algorithm does not use any prior 3D model or make any particular appearance assumptions. Although it may be good idea to model both the world and the objects for better tracking, this causes the tracker to loose its generality. This section will explain the tracking of the detected objects. First, resemblance matrix is estimated between the detected objects and the previously tracked objects. Second, the state-likelihood matrix C is calculated. Third, a globally optimum assignment is found using CMHT. Last, the assignments are run through a finite state machine (FSM) per object for temporal coherence.

3.2 Resemblance function
Each object is modeled by their shape, size and velocity. Objects' shapes are modeled by their convex hulls, which are extracted from the video using background subtraction and connected component analysis. Size of an object is

inferred from its convex hull and velocity is estimated using the motion vector estimation method explained in [Sebe 04]. Resemblance function $\boldsymbol{f}$ is obtained by fusing all these cues together:

$$f(i,j)=\frac{\frac{4*Int(O_i,O_j)}{S_i+S_j}\left(\frac{\sqrt{S_i*S_j}}{S_i+S_j}+2\lambda\frac{\sqrt{\|V_i\|*\|V_j\|}}{\|V_i\|+\|V_j\|}*\left(\frac{\vec{V_i}\bullet\vec{V_j}}{\|V_i\|*\|V_j\|}+1\right)\right)}{1+\lambda}, \tag{3}$$

Int() function finds the overlapping area between two objects. This function enforces the continuity of the tracker, i.e. no overlap in the image suggests no resemblance. We also compare the relative size of the objects, size and direction of their motion vectors. One can and should use a different function that is specific to problem at hand. Our proposed tracking algorithm will work with any function that is scaled between zero and one. The entries of the R_{nxm} matrix are filled by using (3).

3.3 Search space conversion

2D resemblance matrix R represents the interaction between the detected and previously tracked objects. We use this information to make inferences about the state of the objects. 3D state-likelihood matrix C is estimated combining R with the following rules:

- A newly detected object should be classified as "appear" if it does not resemble any previously tracked objects.
- A previously tracked object should be classified as "disappear" if it does not resemble any newly detected objects.
- A newly detected object should be assigned to a particular previously tracked object if they resemble only to each other and no other objects. "track".
- A previously tracked object should be classified as "split" if it resembles to two new objects but nothing else and also these two new objects should not resemble any other previously tracked object.
- A newly detected object should be classified as "merge" if it resembles to two tracked objects but nothing else and also these two tracked objects should not resemble any other new object.

These five statements are converted to mathematical expressions and applied to every pair of objects (i.e. each entry in the R matrix) to create the C matrix. C matrix entries correspond to the likelihoods of states for object pairs/triplets. These values are clipped to [0, 1] interval. One can assign a threshold parameter per state, where these parameters can be statistically learned or manually entered. Unfortunately this may result in conflicting states, such as all the states can fail or more than one state can be satisfied at the same time. Instead we choose the state assignment of the objects using CMHT algorithm, which will be explained in the next subsection.

First state is "appear". If a new object does not resemble any previously tracked object than the sum of its corresponding resemblance should be small (Another option is using "max()" instead of sum). Since this entity is only a function of new objects we copy the same result to the whole row. The likelihood of an object to "appear" is:

$$C(i,:,A)=1-\sum_{l=1}^{m}R(i,l), \tag{4}$$

Second state is "disappear". If a tracked object does not resemble any new object than the sum of its corresponding resemblance should be small. This entity corresponds the sum over columns (Again max() operator can be used instead of the sum). The likelihood of an object to "disappear" is:

$$C(:, j, D) = 1 - \sum_{k=1}^{n} R(k, j)\,, \tag{5}$$

Third state is "track". This state will be chosen if the positive clue (resemblance of the pair) is considerably larger than the negative clue (resemblance of these objects to other objects). The likelihood of an object pair to be assigned as "track" is:

$$C(i, j, T) = R(i, j) - \sum_{k=1, k\neq i}^{n} R(k, j) - \sum_{l=1, l\neq j}^{m} R(i, l)\,, \tag{6}$$

Fourth state is "split". This state is for a previously tracked object splitting into two new objects. This is a relation between three objects. Since we have only the pairs and not the triplets in our data structure, we take a first order approximation in our search at this level. Fixing the pair we search for the extra new object (there is only "m-1" of them) that gives us the largest likelihood when flagged as "split" using (7). This third element is saved in an extra matrix called "S". The likelihood of a triplet to be assigned as "split" is:

$$C(i, j, S) = R(i, j) + R(i, j') - \sum_{l=1, l\neq j, l\neq j'}^{m} R(i, l) - \sum_{k=1, k\neq i}^{n} R(k, j) - \sum_{k=1, k\neq i}^{n} R(k, j'), \qquad \mathrm{S}(i, j) = j'\,, \tag{7}$$

Fifth and last state is "merge". Two previously tracked objects can merge and form a new object. This generally happens due to occlusion. As before a search for the third element is performed and saved into the "merge" matrix M. The likelihood of a triplet to be assigned as "merge" is:

$$C(i, j, M) = R(i, j) + R(i', j) - \sum_{k=1, k\neq i, k\neq i'}^{n} R(k, j) - \sum_{l=1, l\neq j}^{m} R(i, j) - \sum_{l=1, l\neq j}^{m} R(i', j), \qquad \mathrm{M}(i, j) = i'\,, \tag{8}$$

The storage space required is O(nm), since state-likelihood matrix C has a size of $n_x m_x 5$, and split matrix S and merge matrix M are of size $n_x m$. The computational complexity required to create these matrices is $O(nm(n+m)) \sim O(n^3)$.

Once these matrices are created, we can search for the optimum assignment using the modified MHT algorithm which will be explained in detail in the next subsection.

3.4 Combinatorial multiple hypothesis tracking (CMHT)

CMHT approximates the optimum assignment without enumerating all the possibilities, which is inefficient since the number of possibilities is $\binom{m}{h}.\binom{n}{h}.(h!+2\binom{h}{2}!\,h!)$. For example, if there are eight new objects and eight previously tracked objects, then the number of possible assignment is around seventeen million. Exhaustive search is not feasible if real-time performance is expected. Instead we develop an MHT algorithm that is able to approximate the solution in polynomial time.

Our method is an iterative one. A set of assignments is referred as a "path", where we add one more match to the path at every iteration. Only best k-hypotheses will succeed to the next iteration. First step is the creation of one hypothesis per object. This creates in (n+m) new hypotheses and best k of them is chosen, one for each path. In the next iteration, per path we remove the every object that is already assigned. A new set of hypotheses per remaining objects are created and the best k hypotheses are chosen locally. This is an important step since if we skip this elimination at this level, then the number of elements is k(n+m) instead of k^2. This first elimination does not change the best k hypotheses since if a hypothesis is not among the best k in one of the branches; it will not be in the best k when all branches are combined. This is applied to every path and k^2 hypotheses are obtained (there are k paths and we create k new hypotheses for every path). We choose the best k out of these and add appropriate assignments to the path.

Every path is composed of a set of object assignments. For example, $\{\{g^1_{2,}\ g^2_3\}, \{g^2_4\}, \{g^1_{1,}\ g^2_4, g^2_5\}\}$ is one path and stands for object two from newly detected objects is tracked as object three from the previously tracked object, object four is disappeared and etc. The program terminates if the current best hypothesis already has all the objects assigned.

Pseudo-code	Computational Complexity
1. Create R_{nxm} matrix using resemblance function f.	$O(n^2)$
2. Create C_{nxmx5} from R_{nxm}.	$O(n^3)$
3. Current k-best is NULL.	O(1)
4. While (best solution is not final)	O(n)
4.1. for every k path	O(k)
4.1.1. Remove already assigned objects for this path.	O(d)
4.1.2. Find the best hypothesis for every remaining object.	$O(n^2)$
4.1.3. Choose the k-best out of these.	O(kn)
4.2. Choose k-best out of k^2 hypothesis.	$O(k^2lgk)$
4.3. Eliminate paths that correspond to the same set of assignments.	$O(n^2)$
4.4. Best solution is the best of the k-best	O(1)
5. Return best solution as the assignment.	O(1)

Figure 2: Pseudo-code and computational complexity for CMHT algorithm

The MHT algorithm when applied to combinatorial problems requires some additional work to function properly. One of these is the elimination of the paths that have the same assignments. Different paths do not necessarily mean different set of assignments, since assignments can appear in different order. This may seem to be rare but in the specific case of object matching, this is very common. Good matches pop themselves to the top of the search space at the local k-best search. For example, let's have two good matches, a_1 and a_2, and four mediocre matches, b_1, b_2, b_3 and b_4 and allow two hypotheses at a time. During the first iteration, two good assignments, a_1 and a_2, will be chosen for each path, $\{a_1\}$ and $\{a_2\}$. At the second iteration, the path that has a_1 will choose a_2 $\{\{a_1\}, \{a_2\}\}$, and the path that has a_2 will choose a_1 $\{\{a_2\}, \{a_1\}\}$. Although the two paths are separate the matches that are made are same. Removal of the same assignments requires a comparison of k^2 matches, whom each have d assignments (d for depth of the path in the search). This operation is of complexity $O(k^2d^2)$. Since the depth of the tree can be (n+m) in the worst case. The complexity is $O(n^2k^2)$, but since the k is a fixed parameter such as five or ten in the program, this elimination has only $O(n^2)$ complexity. A pseudo-code of the proposed tracker is shown in figure 2. The computational complexity of the program is $(O(n^3)+O(n).O(k).\ O(n^2))\sim O(n^3)$. During the derivation we choose n=m for simplicity.

3.5 Temporal coherence for CMHT

CMHT method is optimum between frames but not over time (a suboptimal assignment in one frame may result in better spatio-temporal tracking). This can be prevented by creating and tracking multiple hypotheses per object (currently not implemented). Instead in order to prevent sudden or inappropriate changes in the states of our objects, we used a finite state machine (FSM), see figure 3.

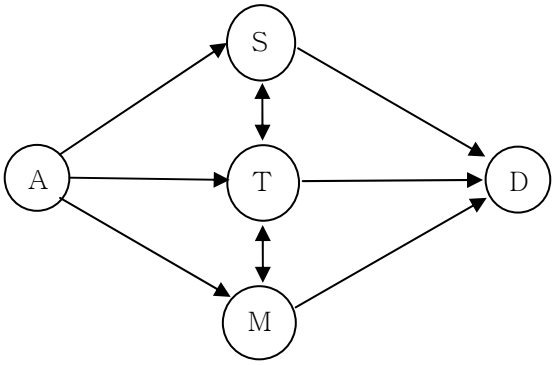


Figure 3: Transition diagram of an object states

If CMHT returns an invalid state transition, the FSM will discard this and use the own properties of the object to estimate its location. Also each link has a threshold that will prevent transitions to be done abruptly. The thresholds are minimum number of frames that particular transition needs to be requested by CMHT. For example, if the threshold for the A→T link is four, then an object will go from appear state to track state only if this is requested four times in a row. The threshold values can be either learned from a learning data or set manually. For example, if splitting of an object is very unlikely, higher thresholds should be set for split transitions.

4. Experiments and Discussion

CMHT is applied to various video streams, ranging from synthetically created video sequences to real world data sequences. Our tracker is an un-optimized C++ code that runs on a 2.6GHz Pentium IV computer. The average speed of the tracker is around twenty frames per second on this particular machine on 360x240 RGB images when the number of objects is around ten. Image reading from the hard drive takes approximately thirty percent of the total computational time.

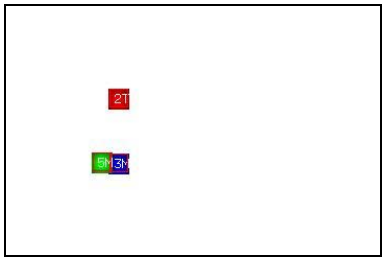
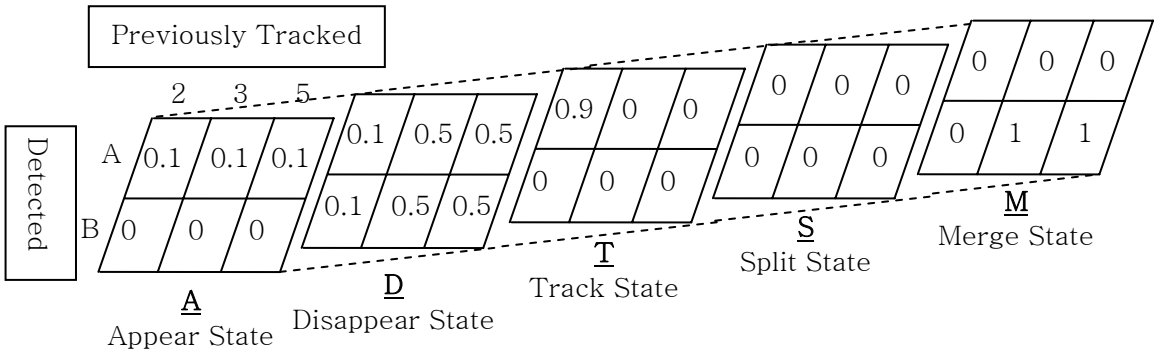


Figure 4: Sample frame from a synthetic sequence, where object three and object five are going under occlusion.

We will first demonstrate how our system is able track multiple objects under occlusion when the detection is perfect, i.e. synthetic sequence. The test synthetic sequence is a multiple objects scenario, where the scene has various number of full and semi occlusion cases. Detected object's convex hull is overlaid in red onto the actual image and also each object has its own ID and state overlaid as a number and a letter respectively. One snapshot of the tracker can be seen in figure 4. For example, for figure 4, object two is tracked and object three and object five are merging.

	#2	#3	#5
A	0.94	0	0
B	0	0.52	0.52

(a) Resemblance Matrix R_{2x3}



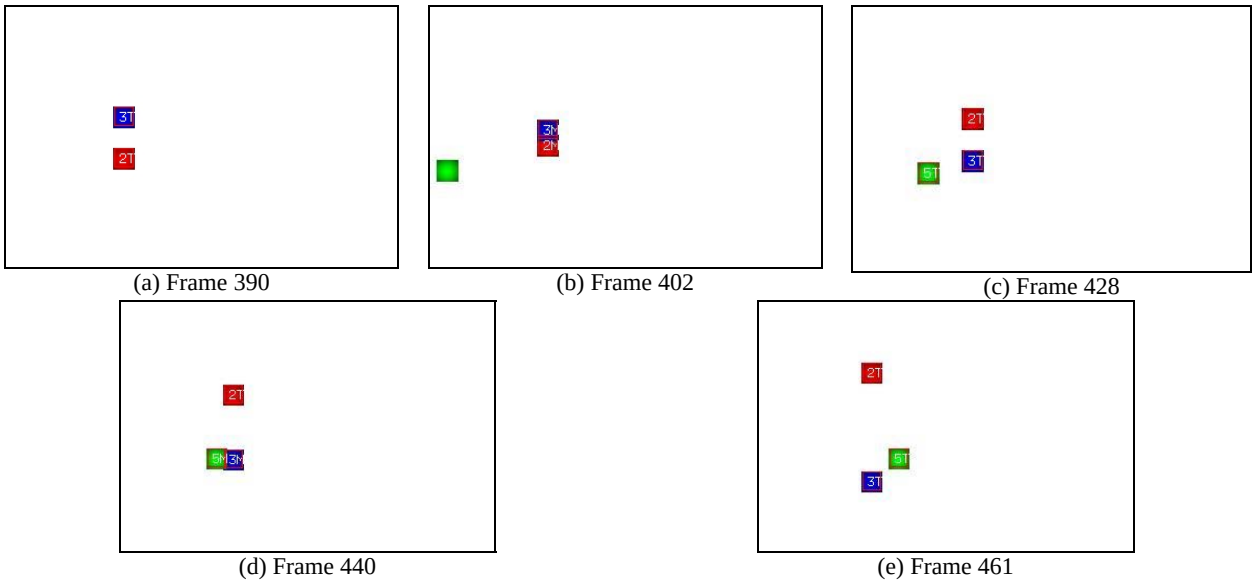
(b) State-likelihood matrix C_{2x3x5}

Figure 5: Resemblance and state-likelihood matrices for a particular case in Figure 4

For the case seen in figure 4, the corresponding resemblance matrix R and state likelihood matrix C is shown in figure 5a and 5b respectively. There are three history objects and 2 blobs (A and B) in the current frame. Matrix R is of size 2_x3 and C is of size 2_x3_x5. Merging of an object with another object is forbidden if their resemblance is too small. Every entry of matrix C is the likelihood of a state for the given object pair. The corresponding result found by CMHT

to this particular case is A is object two and B is a merge of object three and object five. This was expected since the corresponding likelihood values are 0.9 and 1, respectively. This shows how converting R matrix into a state-likelihood matrix C helps to solve our assignment problem.

Figure 6 shows a sequence where objects are going through a couple of occlusions (one full and one semi-occlusion). Red object going from bottom to top has ID two, blue object going from top to bottom has ID three, and green object going from left to right has ID five.



(a) Frame 390 (b) Frame 402 (c) Frame 428

(d) Frame 440 (e) Frame 461

Figure 6: Synthetic sequence results under multiple occlusions

As can be seen from figure 6, objects are successfully tracked even under occlusion. This synthetic sequence shows that the tracker can track objects when the segmentation is perfect and all the assumptions are satisfied. The synthetic sequences are good for ground truth testing, but real world data has a lot of challenges, e.g. imperfect segmentation.

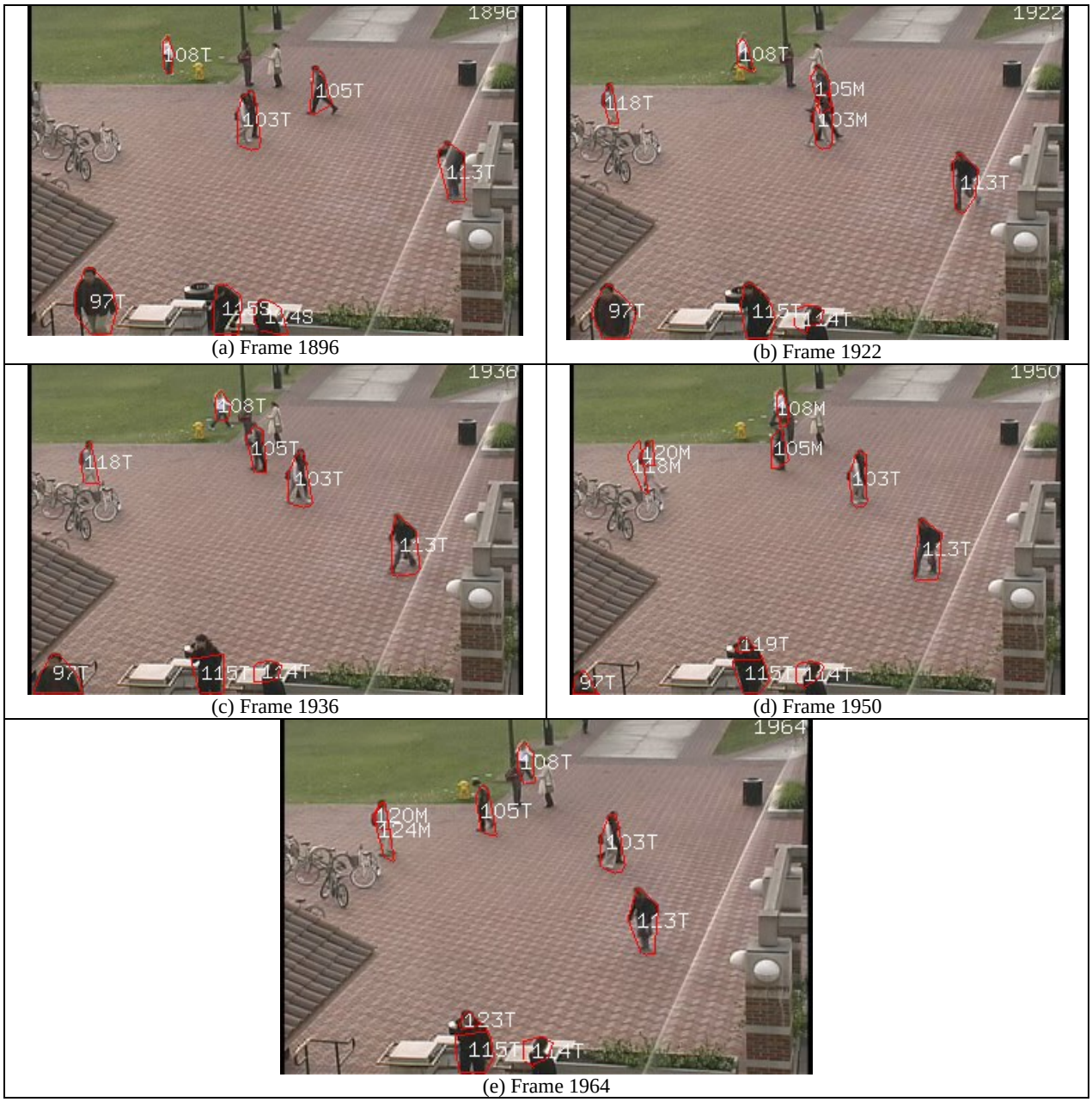
In order to show the generality of our algorithm, we tested our system on real world sequences as well. We used an outdoor scene[1], where a fairly large number of moving objects have numerous occlusions. The video is composed of 1500 RGB frames of size 360x240. We selected five frames from this video sequence in order to show the results of the successful tracking even under occlusion. In figure 8a, on the top middle part of the image, three people are walking, object 103, object 105 and object 108, all in tracking state. In figure 8b, object 103 occludes object 105. Their ids are successfully kept and their corresponding states are changed to "merge". Figure 8c shows the result right after they get out of occlusion and the ids are still correct. In figure 8d, object 105 occludes object 108. Again the ids and states are correctly estimated. Finally figure 8e shows the result right after occlusion ended. Another successful tracking can be seen on the right side of the image, object 113. One can ask why the two persons standing in the middle of the action are not detected. They are not detected because they stand at the same location motionless for a long time and become a part of the background.

However tracking is not always perfect, for example object 118 in figure 8b (top left side of the image) could not be tracked due to the segmentation errors. The main problem with that particular object is the color of that person's clothing is very close to the background image. This causes the object to weave in and out of foreground segmentation. We consider such problems as detection errors rather than tracking errors.

On the other hand, it is important to analyze when our proposed algorithm fails to track. We identified three main cases. First, when the assumptions made about the objects are violated. Most of the problems of this kind are due to

[1] We thank IRIS group at USC for this video sequence.

the violation of constant speed assumption or faulty motion vector estimates. When two objects go under occlusion, if they exit the occlusion state with very different velocities, then the tracker fails. Second, the use of only one hypothesis per object causes problems. Currently we are working on adding multiple hypotheses in the case of merge state (occlusion). One such hypothesis is the stopping of both objects; this happens if two people meet and stop for possibly a chat. Another is one object going in the direction of the other; this is happens when two people meet and one joins the movement of the other. Solely adding these hypotheses does not resolve our problem since the answer to the ambiguity can not be found in one frame but instead takes many frames (e.g. it is difficult to decide if one stopped vs. passing still). This can be solved with a similar method mentioned in [Cox 96]: N-scan back. N-scan back allows MHT to have multiple hypotheses for the same object and choose the best one over time. This can be seen as having hypotheses on not only on that frame but also in time (one more dimension in the hypotheses generated).



(a) Frame 1896

(b) Frame 1922

(c) Frame 1936

(d) Frame 1950

(e) Frame 1964

Figure 7: Real world data sequence, Leavy. Multiple humans are tracked even under various occlusions

Lastly, our formulation handles only two objects at a time. If three or more objects start occluding each other the program drops one of the objects and choose the more probable two. This can be solved via recursive formulation of the CMHT but we found this case to be rare in outdoor surveillance, since the cameras generally placed in high places. Another solution to this problem is to create new states that handle the merging of more objects. Although this solves the problem in theory, it will be very difficult to stabilize the system since there will be very few data per object in the case of three or more object occlusion.

5. Conclusion

In this paper, we have presented a modified multiple hypothesis tracking algorithm for globally optimum multiple object tracking. Our non-model based method does not require any information about the particular objects to be tracked, such as humans or vehicles. Furthermore, it does not need any specific camera configuration or knowledge about the camera parameters or ground plane knowledge. The main contribution of this paper is the abstraction of the tracking problem into a polynomial time search algorithm. Objects are governed by five states: appear, disappear, track, split and merge. A state-likelihood matrix is estimated using the similarity of the detected objects and tracked objects. Combinatorial multiple hypothesis tracking (CMHT) approximates globally optimum assignment over this state-likelihood matrix. In our formulation, occlusions are explicitly modeled, which enables the system to act ahead of time in such cases. This particular property differentiates our tracker from more conventional trackers such as the Kalman filter or particle filters. We presented results of our tracker both on synthetic and real world video sequences. As future work, we are working on tracking multiple hypotheses per object over time. This will model various other movements of the objects under occlusion, such as two people meeting rather than passing by.

ACKNOWLEDGEMENT

This work was supported by the National Geospatial Intelligence Agency (NGA) under a NGA University Research Initiative (NRUI) program. We thank the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, for their support and facilities.

REFERENCES

[Chen 01] Y. Chen, Y. Rui, and T. Huang, “JPDAF-Based HMM for Real-Time Contour Tracking”, Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. I, pp. 543-550, 2001.

[Comaniciu 03] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No: 5, May 2003.

[Comaniciu 00] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean-shift”, Proc. IEEE Conf. Computer Vision and Pattern Recognition, Vol. II, pp. 142-149, June 2000.

[Cox 96] I. Cox and S. Hingorani, “An efficient implementation of Reid’s multiple hypothesis tracking algorithm and its evaluation for purpose of visual tracking”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 2, pp. 138-150, February 1996.

[Elgammal 02] A. Elgammal, R. Duraiswami, D. Harwood, L.S. Davis, “Background and foreground modeling using non-parametric kernel density estimation for visual surveillance”, Proc. of The IEEE, Vol. 90, 1151-1163, 2002

[Gary 79] M. R. Gary and D. S. Johnson. Computers and Intractability. Freeman, 1979

[Gold 96] S. Gold, A. Rangarajan, “Softmax and softassign: Neural network algorithms for combinatorial optimization”, Journal of Artificial Neural Networks, pages 381-399, Aug. 1996

[Guo 01] Z. Guo, J. Wanshou, L. Deren, “A new algorithm of automatic relative orientation”, Proc. 22nd Asian Conference on Remote Sensing, 5-9 November 2001, Singapore.

[Hall 02] B. Hall, M. Trivedi, “A novel graphical interface and context aware map for incident detection and monotoring”. 9th World Congress on Intelligent Transport Systems. October 2002.

[Harville 01] M. Harville, G. Gordon, J. Woodfill, “Foreground Segmentation using Adaptive Mixture Models in Color and Depth”. Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video, July 2001.

[Isard 98] M. Isard, and A. Blake, “CONDENSATION - Conditional Density Propagation for Visual Tracking”, IJCV, 29, 1, 1998.

[Isard 01] M. Isard and J. MacCormick, “BraMBLe: A Bayesian Multiple-Blob Tracker,” ICCV 2001.

[Kanade 98] T. Kanade, R. Collins, A. Lipton, P. Burt and L. Wixson, “Advances in cooperative multi-sensor video surveillance”. Proc. Of DARPA Image Understanding Workshop. Vol. 1, pp. 3-24. 1998.

[Kang 03] J. Kang, I. Cohen, and G. Medioni, “Continous tracking within and across camera streams”, Proc. IEEE Conf. Computer Vision and Pattern, CVPR 2003, Madison, Wisconsin, June 16-22, 2003.

[Koller 94] W. Koller, J. Malik, “Robust multiple car tracking with occlusion reasoning”. In Proc. Third European Conference on Computer Vision, May 2-6, 1994, pp. 189-196, LNCS 800, 1994.

[Kumar 00] R. Kumar, H.S. Sawhney, Y. Guo, S. Hsu, S. Samarasekera, “3D manipulation of motion imagery”. ICIP 2000. September 2000.

[McKenna 99] S. McKenna, Y. Raja, and S. Gong, “Tracking color objects using adaptive mixture models”, Image and Vision Computing Journal, vol. 17, pp. 223-229, 1999.

[Reid 79] D. Reid, “An algorithm for tracking multiple targets”, IEE Transactions on Automatic Control, vol. AC-24, pp. 843-854, 1979.

[Remagnino 97] R. Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A. Worrall, K. Baker, “An integrated traffic and pedestrian Model-based Vision System”. Proceedings of BMVC97, vol. 2, 8-11th September, pp 380-389, 1997.

[Sebe 04] I.O. Sebe, S. You, U. Neumann, “Dynamic objects modeling and 3D visualization”, ASPRS04, May 2004, Colorado

[Stauffer 99] C. Stauffer and W.E.L Grimson, "Adaptive background mixture models for real-time tracking", CVPR 1999.

[Tao 00] H. Tao, H.S. Sawhney, R. Kumar, “Dynamic layer representation with applications to tracking”, in Proc. IEEE conf. on Computer Vision and Pattern Recognition, CVPR 2000, June 2000, vol. 2, pp 134-141.

[Toyama 99] K. Toyama, J. Krumm, B. Brumitt, B. Meyers (1999). Wallflower: Principles and Practice of Background Maintenance. ICCV 1999. pp 255-261.

[Wren 97] C. Wren, A. Azarbayejani, T. Darell, and A. Petland, “PFinder: Real-time tracking of Human Body”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, pp. 780-785, 1997.

[Zhao 04] T. Zhao, R. Nevatia, “Tracking multiple humans in complex situations”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No: 9, September 2004.