This document is published in:

Journal of Heuristics, August 2011, 17 (4), pp 415-440.

DOI: 10.1007/s10732-010-9140-4

Boosting video tracking performance by means of Tabu Search in intelligent visual surveillance systems

Ivan Dotu · Miguel A. Patricio · A. Berlanga · J. García · José M. Molina

Abstract In this paper, we present a fast and efficient technique for the data association problem applied to visual tracking systems. Visual tracking process is formulated as a combinatorial hypotheses search with a heuristic evaluation function taking into account structural and specific information such as distance, shape, color, etc.

We introduce a Tabu Search algorithm which performs a search on an indirect space. A novel problem formulation allows us to transform any solution into the real search space, which is needed for fitness calculation, in linear time. This new formulation and the use of auxiliary structures yields a fast transformation from a blob-to-track assignment space to the real shape and position of tracks space (while calculating fitness in an incremental fashion), which is key in order to produce efficient and fast results. Other previous approaches are based on statistical techniques or on evolutionary algorithms. These techniques are quite efficient and robust although they cannot converge as fast as our approach.

This work was supported in part by Projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, CAM CONTEXTS (S2009/TIC-1485) and DPS2008-07029-C02-02.

I. Dotu
Brown University, Box 1910, Providence, RI 02912, USA
e-mail: idotu@cs.brown.edu

M.A. Patricio (✉) · A. Berlanga · J. García · J.M. Molina
Universidad Carlos III de Madrid Avda. Universidad Carlos III, 22, 28270 Colmenarejo, Madrid, Spain
e-mail: mpatrici@inf.uc3m.es

A. Berlanga
e-mail: aberlan@ia.uc3m.es

J. García
e-mail: jgherrer@inf.uc3m.es

J.M. Molina
e-mail: molina@ia.uc3m.es

Keywords Video-tracking · Tabu search · Data association

1 Introduction

Intelligent Visual Surveillance (IVS) systems are becoming a key component in ensuring security in buildings, commercial areas, public transportation, parking areas, etc. (Regazzoni et al. 1998a, 1998b; Ferryman et al. 2000). The primary aims of these systems are the real-time monitoring of persistent and transient objects and the understanding of their activity within a specific environment.

Intelligent Visual Surveillance Systems track all the targets moving within their local field of view (Castanedo et al. 2006; Patricio et al. 2007a). The real-world video object trackers often face the Multi-target Joint Estimation (MTJE) problem. MTJE is concerned with the estimation of the number of objects in a scene, together with their instantaneous locations, kinematic states, and any other characteristics required. These entities, which characterize the target, are referred to as a *track state*. Track-state vectors with position and kinematic estimates (which can be referred to the camera plane) are used for tracking, and are typically complemented with attributes defining the target's extension, shape, color, and identification. A frequently used representation of the target state $\mathbf{x}$ corresponding to the i-th object is (Han et al. 2004) $\mathbf{x}_i \in \Re^{\,d}$, $\mathbf{x}_i = [\,x,\ y,\ \dot{x},\ \dot{y},\ w,\ h,\ s]$, which correspond to the centroid $(x,\ y)$, velocity $(\dot{x},\ \dot{y})$, extent (w—width and h—height) and scale (s) of the object, respectively. The state dimension, d is set to 7, in this case.

The inference of the real state of a certain environment, based on the information available in the sequence of images, can be addressed as an estimation problem with statistical techniques. The multi-target tracking problem consists in estimating the number of objects in a scene, together with their dynamic state (location, speed, attitude, size, etc.). For each time instant, $t[k]$, there exists as a set of $N[k]$ objects, $E[k] = \{O_1[k], \ldots O_{N[k]}[k]\}$, where each object is defined by a set of characteristics at this instant. The description of the individual objects is expressed in a vector state space, $\mathbf{x}_i[k] \in \Re^d$, $1 \leq i \leq N[k]$.

In video processing, each detected object is observed through a set of compact regions (blobs), formed by adjacent detected binary pixels at this instant:

$$Z_i[k] = \{b_{i,1}[k], \ldots b_{i,M_i}[k]\}$$

where M_i is the number of blobs that are due to i-th object. The problem is that both $N[k]$ and the correspondences among blocks and objects (i subindex) are unknown, they must be estimated from the observed data. The global estimator in the multitarget case is denoted as $\hat{\mathbf{X}}[k]$, which includes both the number of targets and their state in the scene at time instant $t[k]$, $\hat{\mathbf{X}}[k] = \{\hat{\mathbf{x}}_1[k], \ldots \hat{\mathbf{x}}_{N_k}[k]\}$, where $\hat{\mathbf{x}}_i[k] \in \Re^d$.

We can characterize a combinatorial method as the "hard" association of the sequence of measurements to all the tracks, based on certain cost criterion. The data association process determines which observations correspond to which objects, so that each measurement is used to update the appropriate track.

In the simplest case, a Nearest Neighbor (NN) strategy determines blob-to-track correspondence, heuristically assigning tracks to the closest observations, without

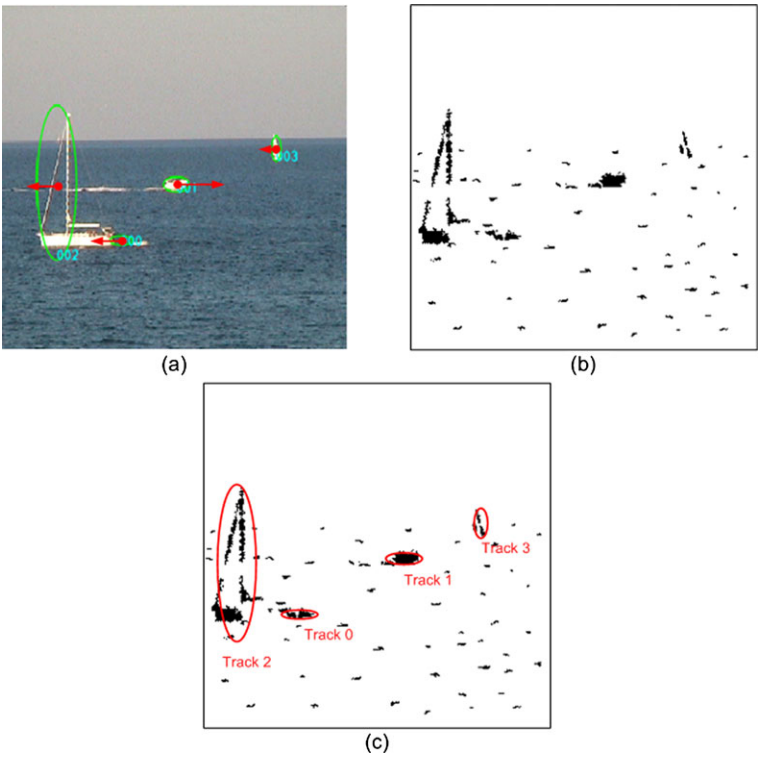

Fig. 1 (**a**) Four tracks (from 0 to 3) and their current state (position, size and velocity) at frame $k-1$; (**b**) Observation/detection of moving objects at frame k, we call blob to each connected group of detected pixels; (**c**) Association output at frame k

dealing with ambiguous correspondences among different tracks, or considering the subsequent frames. However, in real-world environments, the tracking system should be able to handle complex movements and interactions among targets. Therefore, the one-to-one constraint is normally removed, and the correspondence of multiple blobs to multiple tracks is evaluated, accounting for fragmented or merged blobs which can contribute to each track. In order to assess such correspondence, an evaluation function, which takes into account not only residuals between centroids but also structural information about objects, has to be defined.

In Fig. 1, four targets are represented by their track-state vector (position, size and velocity) at frame $k-1$. At time k, a new frame is captured from the camera. First, a detector of moving objects must generate a list of blobs that are found in the frame; this list must contain information about the position and size of each blob. A blob is a connected set of pixels in an image. The pixels contained in a blob must be locally distinguishable from pixels which are not part of the blob (see Fig. 1b). Next, the association process will solve the problem of blob-to-track multi-assignment, where several (or no) blobs may be assigned to the same track and where, simultaneously, several tracks can overlap and share common blobs. Thus, the association problem we must solve corresponds to the decision regarding the most proper grouping of

blobs and their assignment to each track for every processed frame. The blob deleting process eliminates those blobs which have not been associated to any track, since they are considered to be noise. Finally, the track updating process, updates the track-state vectors with the information obtained from the association output (Fig. 1c).

The Kalman filter is the most popular estimation technique to compute track state vectors at frame k, combining the information in the current observation with the prediction from a previous frame at $k-1$. However, the Kalman filter solves the particular problem of a single state vector updated with a single measurement at each frame; that is, the state of a single object is estimated. Unfortunately, tracking multiple objects is a much more difficult problem. It deals with an unknown number of active objects as sources of measurements (blobs), and the statistical model requires both continuous variables to describe each target state and discrete variables to describe the correspondences between objects and observations. The strategy of converting a *N-multi-target* problem into a collection of N single-target problems (and then applying a Kalman-type estimation solution) requires decisions interleaved with the state estimation processes from a joint solution of data association.

Recently, Patricio et al. (2007b) have shown how to formulate the association process as a search problem. This new approach consists of a search across the hypothesis space defined by the possible associations among tracks and detections (blobs), carried out for each frame of a video sequence. The full data association problem in visual tracking is formulated as a combinatorial hypothesis search with a heuristic evaluation function taking into account structural and specific information estimated by a filter method, usually a Kalman filter. Obviously, since the process must guarantee real time performance, the search must have a time limit. They proposed several evolutionary computation techniques, which proved efficient when searching in this hypothesis space (Patricio et al. 2007b).

In this work, we propose the use of a metaheuristic named *Tabu Search* (Glover and Laguna 1993) for carrying out the data association problem. This method introduces a novel approach to achieve the necessary incrementality[1] in this particular problem. Metaheuristics and Local Search techniques have been widely used to solve real-life problems, especially scheduling problems (it would be impossible to name these works so we refer the reader to this review (Grunert et al. 2005)). These techniques, have been applied to other real-life engineering problems, however, there are a few works related to video-tracking problem. The most frequently reported applications of Tabu Search or local search we are aware of are: Kincaid and Laba (1998) for acoustic control, Pisinger et al. (2003) for VLSI design, Cordon and Damas (2006) for image registration, and Huwer and Niemann (1998) for second object tracking. This last reference presents a similar problem to that shown in this paper, although it focuses on different issues, and it does not even describe the implemented local search algorithm. In order to assess the performance of *Tabu Search*, the authors have modeled the problem of video-tracking to be solved by other algorithms and have made a comparative analysis, as can be seen in the experimental section.

[1] Incrementality is the ability to evaluate quickly (in constant or linear time) a function using a previous calculation as a starting point.

2 Related work

The performance of a video tracking system implementation depends on two strictly coupled subtasks: the data association method used for assigning observations to tracks and the model selected to estimate the movement of an object (Patricio et al. 2008). Tracking algorithms can be formulated as the correspondence of detected objects represented by tracks across frames. The correspondence methods can be divided into heuristic and probabilistic methods. Probabilistic methods consider the object measurement and the uncertainties to establish the correspondence, while heuristic methods use qualitative motion heuristics to constrain the problem.

2.1 Probabilistic methods

In the case of probabilistic methods, the target-tracking community has usually formulated the total process as an optimal state estimation and a data association problem, with a Bayesian approach to recursively filter the observations coming from the sensors. Only in the case that a single target appears, with no false alarms, is there no association problem and optimal Bayesian filters can be applied, such as Kalman filters under ideal conditions, or suboptimal Bayesian filters like Multiple Models (IMM) (Yeddanapudi et al. 1997) for realistic maneuvering situations and Particle Filters (PF) (Arulampalam et al. 2002; Djuric et al. 2003) in non-Gaussian conditions.

In general, the tracking system should handle complex motions and interactions such as passing, occlusions and stopping. Each real object may generate multiple observations and the problem of searching for an optimal or near-optimal hypothesis requires a deeper analysis. A number of statistical data association techniques have been developed for this purpose. The problem can also be extended across multiple scans (frames) to search for the best hypothesis for the associations, using a tree of open hypotheses to delay assignment decisions until more evidence is available. Multi-scan, multiple-target algorithms commonly used include Multi-Hypothesis Trackers (MHT) and Joint Probabilistic Data Association (JPDA) (Blackman and Popoli 1999). The Probabilistic Multiple Hypotheses Tracking algorithm (Ruan and Willett 2004), and also artificial neural networks (Shams 1996) have been developed in order to reduce the computational complexity of JPDA and MHT.

Recently, data association algorithms have also received wider attention by the computer vision community. For instance, the JPDA filter has been applied to vision 3-D reconstruction (Chang and Aggarwal 1991). However, the direct formulation of these algorithms (JPDA and MHT) presents important limitations. The first one is that they assume that a target can generate, at most, one measurement per scan and, conversely, a measurement could have originated from, at most, one target. For instance, Cox (Cox and Hingorani 1996) proposes an implementation of the MHT algorithm with visual sensors (although objects are simplified to points), not considering the problem of data complexity. The full problem of data association in the visual context is a correspondence of multiple blobs to multiple tracks. It is needed to remove the one-to-one constraint (when we have only a single measurement for a target) and extend the algorithms to consider multiple fragmented or merged blobs updating each

track. Besides, the evaluation function, such as a distance considering only residuals between centroids disregards structural information about the objects. Thus, it is necessary to build an extended function to evaluate the assignment hypotheses considering structural information as well. In the second place, the required computational burden may be too heavy with large numbers of targets and observations.

2.1.1 Mean-shift and particle filtering

In this section, we describe the probabilistic method based on an implementation of the Mean-Shift algorithm together with a Particle Filtering algorithm, which is one of the most powerful tracking system in the vision community (Chen et al. 2005). The Mean-Shift algorithm was proposed by Comaniciu and Meer (2002) a s a n i m - age segmentation technique. The algorithm finds clusters in the joint spatial & color space. It is initialized with a large number of hypothesized cluster centers, which are chosen, randomly, from a particular image. Then, each cluster center is moved to the mean of the data lying inside the multidimensional ellipsoid centered on the cluster center. Several iterations where clusters may get merged exist. Mean-shift clustering has been used in various applications, such as edge detection, image regularization (Comaniciu and Meer 2002), and tracking (Comaniciu et al. 2003).

The Kalman filter (Xiao-Rong and Bar-Shalom 1995) is the most extended probabilistic approach within the vision community for video tracking processes. It states that measurements obtained from video cameras invariably contain noise. Moreover, the object motions can undergo random perturbations, for instance, maneuvering vehicles. The Kalman filter performs the tracking by taking the measurement and the model uncertainties into account during the object state estimation. It uses the state space approach to model several mobile object properties such as position, velocity, and acceleration. Broida and Chellappa (1986) use the Kalman filter to track a point in noisy images. Beymer et al. (1999) use a Kalman filter for predicting the object's position and speed in a 3D space. One limitation of the Kalman filter is the assumption that the state variables are regularly distributed. Kitagawa (1987) proposed the Particle Filtering algorithm in order to overcome this limitation. The Particle Filtering algorithm is a powerful tool for approximating non-Gaussian probability distributions, and it has been applied in video tracking as well.

2.2 Heuristic methods

On the other hand, heuristic algorithms define a cost of associating each object in frame $k-1$ to a single object in frame k using a set of motion constraints. Minimization of the correspondence cost is formulated as a combinatorial optimization problem.

Genetic Algorithms (Gas) have been previously applied to the data association problem in the radar context by Angus et al. (1993) (on a single scan scenario), and by Hillis (1997) to deal with the multi-scan data association problem. In Patricio et al. (2007b), in order to achieve a fast video process, the performance of a family of very efficient evolutionary computation algorithms is analyzed. This novel approach is called Estimation of Distribution Algorithms (EDAs) (Larraaga and Lozano 2001).

EDAs are a class of Evolutionary Algorithms (EA) which do not implement crossover or mutation operators. The new population is generated by sampling the probability distribution, which is estimated using information from several selected individuals from previous generations. The estimation of the joint probability distribution associated to the selected individuals is the most complex task, and there is a wide range of different attempts to approach this issue. For instance, UMDA (Mühlenbein 1997) assumes linear problems with independent variables; PBIL (Cestnik 1990) uses a vector of probabilities instead of a population, also with independent variables, MIMIC (de Bonet et al. 1997) searches for permutations of variables in each generation in order to find the probability distribution using the Kullback-Leibler distance (two-order dependencies); FDA (Mühlenbein and Mahnig 1999) factorizes the joint probability distribution by combining an evolutionary algorithm and simulated annealing, etc.

In the following, we briefly describe some aspects of these algorithms.

2.2.1 Estimation of distribution algorithms

EDAs (Larraaga and Lozano 2001) present several features which appear to be suitable for dealing with problems that require a very efficient search. They usually use small population samples and carry a small number of iterations (compared to other more classical approaches within the Evolutionary Computation field). The main difference between EDAs and a classic EA is that the former perform a search for the probability distribution describing the optimal solutions, while the latter implements a search procedure which, through selection, crossover and mutation, provides a solution to the problem itself. However, they share the necessity for codification of the solutions by means of (usually) binary chains ('individuals', in the EA terminology) and the definition of a merit measurement that allows for search guidance: the so called 'fitness function'. The EDAs do not need to implement any operators to manipulate the individuals (such as mutation, selection, crossover, etc.), since the search is performed directly on the distribution which describes all possible individuals. EDA algorithms replace an evolving population by a vector that codifies the joint probability distribution corresponding to the best solutions. The crossover and mutation operators are replaced by rules that update the probability distribution. Also, EDAs have a great advantage over the evolutionary algorithms: they can express the interactions between variables by means of the associated joint probability distribution.

In our work, we have implemented three different EDA approaches in order to compare our Tabu Search approach: Univariate Marginal Distribution Algorithm (UMDA), Population-Based Incremental Learning (PBIL) and Compact Genetic Algorithm (CGA). In the UMDA algorithm (Mühlenbein 1997) the joint probability distribution is estimated as the relative frequencies of the variables' values stored in a data set. Independence between the variables is assumed, and theoretically it has been demonstrated that UMDA works almost perfectly with linear problems and rather well when the dependencies are not very significant. PBIL (Population-Based Incremental Learning) (Baluja 1994) mixes the search applied by genetic algorithms with competitive learning. It applies a Hebbian rule to update the vector of probabilities. The CGA (Harik et al. 1999) simulates the performance of a canonical genetic

algorithm with uniform crossover. It is the simplest EDA, and it only needs one parameter: the population size.

3 Formulation of the association process as a search problem based on a probabilistic approach

A Bayesian framework to compute the best estimation, $\mathbf{X}[k]$, inferable from available measurements, $Z[k]$, is the one targeted at obtaining the maximum a posteriori probability of state conditioned to the whole set of observations:

$$\hat{\mathbf{X}}[k] = \arg\max_{\mathbf{X}[k]} P(\mathbf{X}[k]|Z[k], Z[k-1], \ldots, Z[0]) \tag{1}$$

The classic inference formulation applies the Bayes theorem to formulate the problem of updating the conditional *pdf* (probability density function) in a recursive way, depending on the *pdf* computed at frame $k-1$:

$$\begin{aligned} &P(\mathbf{X}[k]|Z[k], Z[k-1], \ldots, Z[0]) \\ &\quad = \frac{1}{c} P(Z[k]|\mathbf{X}[k]) P(\mathbf{X}[k]|Z[k-1], \ldots, Z[0]) \\ &\quad = \frac{1}{c} P(Z[k]|\mathbf{X}[k]) \\ &\qquad \times \int [P(\mathbf{X}[k]|\mathbf{X}[k-1]) P(\mathbf{X}[k-1]|Z[k-1], \ldots, Z[0])] d\mathbf{X}[k-1] \end{aligned} \tag{2}$$

where the integral in the joint problem would extend over the whole space of predicted states, $P(\mathbf{X}[k]|\mathbf{X}[k-1])$, using the recursively estimated pdf at last time $k-1$, $P(\mathbf{X}[k-1]|Z[k-1], \ldots, [Z[0])$, and c is a normalization term independent of $\mathbf{X}$.

The multi-target tracking problem is usually divided into two related problems: data association and state estimation. Data association decides the correspondences to match objects and observations. Then, once the association is decided, one applies the Kalman filter to estimate each target's state as conditioned to this decision. Only when there is an unambiguous way of matching observations to objects, can the N multi-target problem be decomposed into N independent single-target tracking problems solvable with the Kalman filter.

In order to include the association variables in the problem, we can assume that each track is estimated with a Kalman filter, so that the global estimator at a certain time is totally equivalent to the chain of association decisions taken with the sequence of observations:

$$\hat{\mathbf{X}}[k] = KF([A[k], A[k-1], \ldots, A[0], Z[k], Z[k-1], \ldots, Z[0]])$$

where the assignment matrix $A[k] = \{a_{i,j}[k]\}$ is defined as $a_{i,j}[k] = 1$ if blob $b_i[k]$ is assigned to track $\hat{\mathbf{x}}_j[k]$; and $a_{i,j}[k] = 0$ otherwise. The special case, where $j = 0$, represents the assignment of blobs to the 'null track' at the current time. This is used to discard or initialize objects (see Fig. 2).

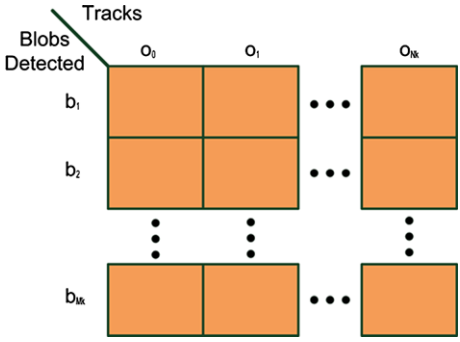

Fig. 2 Assignment matrix representation. Note that its dimension is $M[k] \times (1 + N[k])$. First column is added, for the special case where a blob is not assigned to any track ('null track')

Under these conditions, the pdf of the posterior estimator is determined by the pdf of the sequence of association matrices $P(\mathbf{X}[k]|Z[k]) = P(A[k], A[k-1], \ldots, A[0]|Z[k], Z[k-1], \ldots, Z[0])$.

So, the optimal estimation under this formulation is equivalent to finding the sequence of association matrices, $g_A[k]$, which maximizes the posterior probability $P(\mathbf{X}|Z)$.

Now, the Bayes theorem can be applied to get a recursive formulation of the data association problem and to compute a recursive expression of the function to optimize:

$$\begin{aligned} g_A[k] &= P(A[k], A[k-1], \ldots, A[0]|Z[k], Z[k-1], \ldots, Z[0]) \\ &= \frac{P(Z[k]|A[k], A[k-1], \ldots, A[0], Z[k-1], \ldots, Z[0])\, P(A[k]|A[k-1], \ldots, A[0], Z[k-1], \ldots, Z[0])}{P(Z[k]|Z[k-1], \ldots, Z[0])} \\ &\quad \times g_A[k-1] \\ &= cP(Z[k]|A[k], \mathbf{X}[k-1])\, P(A[k]|\mathbf{X}[k-1])\, g_A[k-1] \end{aligned} \tag{3}$$

If we assume a uniform distribution for prior probability of association $P(A[k]|\mathbf{X}[k-1])$ (all tracks have the same probability of being updated), the optimization problem would consist in finding the optimum sequence of associations to maximize all detection likelihoods:

$$\begin{aligned} \hat{\mathbf{X}}[k] = \arg \max_{A[k],\ldots,A[1]} [&P(Z[k]|A[k], \mathbf{X}[k-1])\, P(Z[k-1]|A[k-1], \mathbf{X}[k-2]), \ldots, \\ &P(Z[1]|A[1], X[0])] \end{aligned} \tag{4}$$

However, this joint optimization of the whole sequence of association matrices is not possible; its complexity increases at an exponential rate with time. A practical approach (single hypothesis optimization) is the sequential optimization of association decisions, where decision at frame $k-1$ is propagated for time k, and the search space reduces to the size of matrix $A[k]$ for each processed frame:

$$\hat{\mathbf{X}}[k] = \arg \max_{A[k]} \left[P(Z[k]|A[k], \hat{\mathbf{X}}[k-1]) \right] \tag{5}$$

The data association requires from a discrete optimization at each frame over the likelihood of current observations conditioned to all previous assignments, sum-

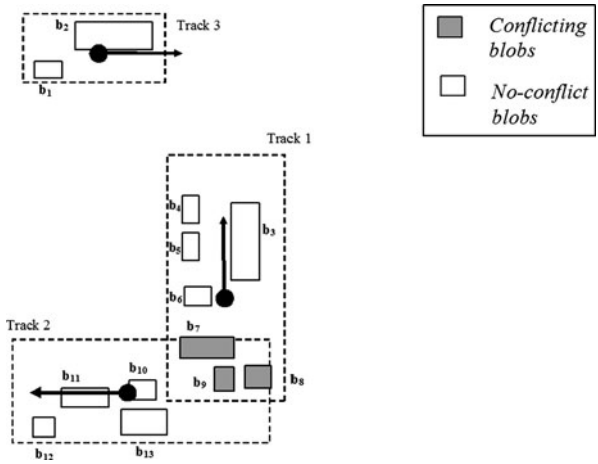

Fig. 3 Optimal assignment in a hypothetical situation with conflicting blobs (i.e., there are blobs that belong to two or more tracks). The three bounding boxes with *dotted lines* represent the prediction of the three tracks according to the Kalman filter

marized by system state at previous time $k-1$, $\hat{\mathbf{X}}[k-1]$. This state represents the estimated number of targets and vector states at the time $\hat{\mathbf{X}}[k-1] = \{\hat{\mathbf{x}}_1[k-1], \ldots, \hat{\mathbf{x}}_{N[k-1]}[k-1]\}$, $\hat{\mathbf{x}}_i[k-1] \in \Re^d$.

The optimal resolution of binary values for $A[k]$ can be represented as a search along the association hypotheses. Generally, the number of hypotheses is given by the expression:

$$N_H = 2^{M[k]\times(1+N[k])} \tag{6}$$

where $N[k]$ and $M[k]$ are the number of tracks and blobs at frame k, respectively. N_H includes the possibility of labelling each blob as not belonging to any track, the 'null track'. In the case of a severe overlap where all blobs can be assigned to all tracks, the expression takes on the following form: $N_H = (1+N[k])^{M[k]}$.

Consequently, the assignment problem may require the evaluation of a very large number of combinations to determine the optimal solution, implying the need for efficient search methods.

Figure 3 illustrates an optimal assignment in a hypothetical situation with three tracks and 13 blobs. The three bounding boxes with dotted lines represent the prediction of the three tracks according to the Kalman filter. In this case, there are three moving objects (in the association problems objects are considered as tracks, thus, from now on, we will use both terms interchangeably). Objects labeled as 'Track 2' and 'Track 3' overlap and, therefore, they share the blobs labeled as 'b7' , 'b8' and 'b9'. In this toy, scenario we can observe that some blobs clearly belong to a certain track (or object), but some others are conflicting, meaning that they could belong to one or more tracks. The representation provided in Fig. 3 shows an occlusion between tracks 1 and 2, thus, conflicting blobs must belong to both tracks at the same time, which is why we define this is an optimal assignment.

In Fig. 4, we see the same scenario but in this case we assume that tracks 1 and 2 are farther away and there is no occlusion. Also, that if track 1 corresponds to a larger object, all previously conflicting blobs belong to it.

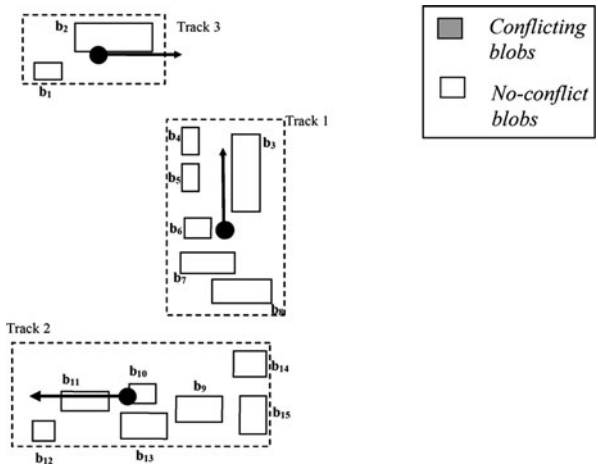

Fig. 4 Optimal assignment in a hypothetical situation without conflicting blobs

3.1 A constraint-based local search formalization of the problem

Let us now formalize the specific problem we will be dealing with from the constraint-based local search standpoint: Given a set of blobs with information about their width, length and centroid (measured in a 2-D grid of pixels) and a set of tracks with the same *predicted* information, assign blobs to tracks in such a way that the distance to the prediction for each track is minimized. Additionally, we can access a hypothesis matrix that discards some blob-to-track combinations, reducing, thus, the search space.

The association problem is thus defined as:

- Given a set of blobs $B = b_1, b_2, \ldots, b_n$ where every blob b_i is defined by its width $b_i.w$, its length $b_i.l$ and its centroid $(b_i.x, b_i.y)$ (measured in a 2-D grid of pixels).
- Given a set of tracks $T = t_1, t_2, \ldots, b_m$ where every track t_i is defined by its width $t_i.w$, its length $t_i.l$ and its centroid $(t_i.x, t_i.y)$ as well.
- An assignment matrix X where there is a $X_{b,t}$ for each blob and track representing whether blob b is considered to belong to track t or not.
- A set of constraints represented by a hypothesis matrix H where $H_{b,t}$ is a boolean value allowing or disallowing the possibility of blob b to belong to track t.
- A cost function f which represents the similarity between the potential assignment explored by the search algorithm and the prediction of every target j according to the Kalman filter in every frame k, $\hat{x}_j[k|k-1]$. The cost function also considers those assignments that leave confirmed tracks with no updating blobs.

The cost function Let $O_j[k]$ be the set of blobs assigned to track j by a search algorithm for frame k. These blobs are represented by their bounding boxes, specifically by their centroid pixel coordinates, width and height (x_j^e, y_j^e, w_j^e and h_j^e). They correspond to those blobs with index i such that $A_k[ij] = 1$. Track j contains the prediction provided by the Kalman filter, $\hat{x}_j[k|k-1]$ represented by its centroid pixel coordinates, width and height (x_j^u, y_j^u, w_j^u and h_j^u).

Let d_j be the normalized distance between the proposal and predicted track j:

$$d_j = \frac{|x_j^u - x_j^e|}{w_j^u} + \frac{|y_j^u - y_j^e|}{h_j^u} \tag{7}$$

Let s_j be the normalized size similarity between the proposal and predicted track j:

$$s_j = \frac{|w_j^u - w_j^e|}{w_j^u} + \frac{|h_j^u - h_j^e|}{h_j^u} \tag{8}$$

Let I be the foreground image that we are processing. We define $I(x, y)$ as true, if and only if the pixel (x, y) is in the bounding box of the proposal for track j. We define the Density Ratio, dr_j, of track j as:

$$dr_j = \frac{\text{Cardinality}\{I(x, y)\}}{w_j^e \times h_j^e} \tag{9}$$

this measurement informs us on how many detected pixels are in the bounding box of proposal for track j. This measurement is better the greater its value.

We also incorporate two penalty ratios to the cost function, corresponding to the probabilities of false positive detection (PFA), and to the probabilities of true positive missing (PD):

- *ap* (assignment penalty). A penalty value is added to the cost function every time a blob is assigned to a distant track.[2]
- *lt* (lost track). A high value is added every time no blob is assigned to one track.

The cost function, f_i , of an assignment i is:

$$f_i = \sum_{j=1}^{N_{k-1}} (d_j + s_j - dr_j) + ap + lt \tag{10}$$

where, N_{k-1} represents the number of tracks in the frame $k - 1$.

Our objective is thus:

- to determine an assignment matrix X which, satisfying all the constraints in H, minimizes the cost function f.

Note that the problem which arises here is that the cost function is not calculated directly. We first need to assign blobs to tracks; then we have to calculate the shape of every track and, finally, we must find the distance from the produced tracks to their prediction. The cost function is the addition (sum) of the distances from all the tracks to their predictions.

[2]By "distant track" we mean when the Euclidean distance between the centroids of the track and the blob exceeds a threshold U.

4 The algorithm: Tabu Search

In this section, we will review all the aspects related to the Tabu Search algorithm, namely the modeling, incrementality issues, neighborhood, memory structure, and, finally, the sketch of the algorithm.

4.1 Modeling

Modeling is a key feature of any algorithm, and especially within the local search framework. Different representations of the same problem define different search spaces as well. In this paper, we use a modeling approach which associates a variable $x[b,t]$ for each pair of blob b and track t. The value of $x[b,t]$ in a given solution, denoted as $\sigma(x[b,t])$ will be 1 if blob b is considered to belong to track t and 0 if it is not.

This is to allow the possibility of a blob belonging to two different tracks. At this level of information, when two tracks collide, we are unable to decide whether the blob belongs to one track or another. Also, forcing a blob to belong to one single track in these situations, could result in the disappearance of one of the objects in the collision. This is obviously not desirable.

In our specific problem there are no constraints on track membership; all combinations are allowed, and thus, we only need to consider the optimization function. As previously stated, the optimization criteria is to minimize the distance to a prediction. However, the prediction is defined in terms of track characteristics. In a straightforward approach, we would need to:

1. Determine which blobs belong to which tracks from the set of $x[b,t]$ variables.
2. Calculate the characteristics of each potential track enclosing the assigned blobs.
3. Calculate the distance from each potential track to its prediction.
4. Add all the distances to compute total distance to prediction.

However, in order to implement a metaheuristic which uses local search on its core for this problem, we need a procedure to calculate the cost of a solution (distance to prediction) in constant time, i.e., achieve incrementality.

4.2 Achieving incrementality

An assignment σ to all the variables has an associated cost $f(\sigma)$ which corresponds to the distance to its prediction. The problem consists of finding the solution σ which minimizes this distance. This is computed in terms of shape and position. We have just seen that this is a process with many steps, which is not suitable for a local search algorithm. However, we can overcome this obstacle by introducing several additional structures:

4.2.1 Track information

The first auxiliary structure is one that captures the information related to the tracks. For each track, we maintain several characteristics:

– **Number** of blobs currently assigned to it.
– **Centroid** of the track in a 2-D matrix of pixels.[3]
– **Height** in pixels.
– **Width** in pixels.

Note, that a track can have 0 blobs assigned to it. In this case, height and width will be set to 0 and centroid to the $(0, 0)$ position. This will penalize the scenario where a track has no associated blobs, but will not forbid it, since it is possible for a track to have disappeared from the scene.

Moreover, this very same information for each blob is available to the algorithm. This information is provided by other modules of the complete system introduced in Sect. 1.

4.2.2 Distance information

We also maintain information regarding the distance from each track to its prediction. Centroid, length, and width relative distance to the prediction is held for each track, along with the total distance (addition of the distances of all tracks). This is standard information in any local search algorithm. Taking this information into account, we can compare it to the new information (after a local move) and quickly find the actual gain in the cost function. Nevertheless, centroid and shape information cannot be calculated directly from a $x[b, t]$ set of variables. This is the key modeling feature which is captured in the following additional structures.

In the following subsections we introduce the auxiliary structures and procedures that allow us to recalculate the centroid, length and width of track after a local move is attempted. Note that for a given track whose distance to the prediction is known (as detailed in Sect. 3.1) f_t, we can easily calculate the new value of the fitness function:

$$f = f + (f'_t - f_t)$$

where f'_t is the new distance to the prediction calculated using precisely the new centroid, length and width of the track.

4.2.3 Blobs information

For each blob b, we maintain several features related to each track t:

– **North:** This represents how many pixels blob b adds to track t from the centroid toward a virtual north coordinate.
– **South:** This represents how many pixels blob b adds to track t from the centroid toward a virtual south coordinate.
– **East:** This represents how many pixels blob b adds to track t from the centroid toward a virtual east coordinate.
– **West:** This represents how many pixels blob b adds to track t from the centroid toward a virtual west coordinate.

[3] $(0, 0)$ position is assumed to be at the upper-left corner of the axis.

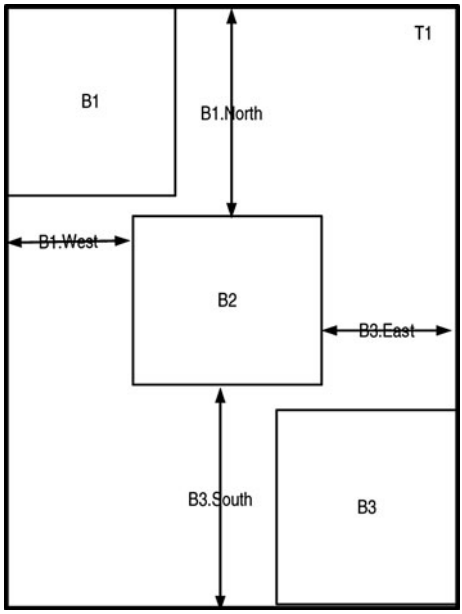

Fig. 5 Information necessary for each blob in order to achieve incrementality

This information corresponds to how many pixels the track would lose if the blob was missing in each coordinate's direction respectively. Figure 5 depicts a scenario where all those features are shown.

This information is calculated initially and then updated after each move. The important feature is its use when calculating the future cost of performing a tentative move. Let us now detail how we can use this information to calculate potentially new characteristics of a track (position and shape) after a tentative move in constant time:

The neighborhood will be formally introduced in the following subsection, but, at a more abstract level, we can distinguish between two different moves: (1) removing a blob from a track, or (2) adding a blob to a track.

4.2.4 Removing a blob from a track

When removing a blob b from a track t, we need to subtract its added shape and position from that track. If that blob is the only blob within the track, then the track becomes empty, and thus, all track characteristics are set to 0.

However, if other blobs still belong to the track, we can recalculate the cost in the following manner:

1. Recalculate x axis coordinate:

$$t_{x'} = t_x + 0.5 * (b,t)_{west} - 0.5 * (b,t)_{east}$$

2. Recalculate y axis coordinate:

$$t_{y'} = t_y + 0.5 * (b,t)_{north} - 0.5 * (b,t)_{south}$$

3. Recalculate width:

$$t_{w'} = t_w - (b,t)_{west} - (b,t)_{east}$$

4. Recalculate height:

$$t_{h'} = t_x - (b,t)_{north} - (b,t)_{south}$$

5. Recalculate distance using the structures mentioned in Sect. 4.2.2.

Where we use t_* to denote characteristic $*$ of track t (and $t_{*'}$ potential characteristic $*$ of track t) and $(b,t)_\lambda$ to denote coordinate λ of blob b with respect to track t.

4.2.5 Adding a blob to a track

When adding a blob b to a track t we have to add its information to the track unless it is already contained in the current shape of the track. If the track is empty then its information will be set to that of the blob we are adding.

However, if track t is not empty, its new information after potential adding blob b is calculated as follows:

1. Recalculate width:

$$t_{w'} = t_w + \max(0, b_{right} - (t_x + 0.5 * t_w)) + \max(0, (t_x - 0.5 * t_w) - b_{left})$$

2. Recalculate height:

$$t_{h'} = t_h + \max(0, b_{bottom} - (t_y + 0.5 * t_h)) + \max(0, (t_y - 0.5 * t_h) - b_{top})$$

3. Recalculate x axis coordinate:

$$t_{x'} = \min(b_{left}, t_x - 0.5 * t_w) + 0.5 * t_{w'}$$

4. Recalculate y axis coordinate:

$$t_{y'} = \min(b_{top}, t_y - 0.5 * t_y) + 0.5 * t_{y'}$$

5. Recalculate distance using the structures mentioned in Sect. 4.2.2.

In these calculations, we use $\min(a,b)$ and $\max(a,b)$ to denote the minimum and the maximum (respectively) out of two real numbers: a and b, and:

- b_{top} to denote y axis coordinate value for the upper extreme of the blob ($b_y - 0.5 * b_h$)
- b_{bottom} to denote y axis coordinate value for the lower extreme of the blob ($b_y + 0.5 * b_h$)
- b_{left} to denote x axis coordinate value for the leftmost position of the blob ($b_x - 0.5 * b_w$)
- b_{right} to denote x axis coordinate value for the rightmost position of the blob ($b_x + 0.5 * b_w$)

4.3 The initialization phase

The initialization phase is known to have a significant impact on the efficiency of a Tabu Search algorithm (see Dotu and Van Hentenryck 2005; Harvey and Winterer 2005). In this case, we implemented two different initialization methods:

- **Randomly** assign 0 or 1 to the $x[b, t]$ variables. This is equivalent to randomly assigning blobs to tracks. Whenever a blob is not allowed to belong to a certain track (given by the hypothesis matrix mentioned in Sect. 3.1), the corresponding $x[b, t]$ variable is set to 0.
- **Copy** the hypothesis matrix, i.e., set to 1 all the allowed blob-to-track combinations.

The former method is used when the problem presents a low blob-track ratio, i.e., when there will not be many blobs belonging to each track. On the other hand, the second method will be more efficient in scenarios where there is a high blob-track ratio, i.e., when each track is composed of many different blobs.

4.4 The neighborhood

The neighborhood used consists of flipping the value of a $x[b, t]$ variable. Notice that we can always flip a variable when it is set to 1. However, we can only flip a variable set to 0 if the corresponding blob-to-track combination is allowed by the hypothesis matrix.

The set of swaps is thus defined as

$$\mathcal{S}(\sigma) = \{\langle b, t\rangle \mid \sigma(x[b, t]) = 1 \text{ or } \sigma(x[b, t]) = 0 \text{ and } b \in h[t]\}$$

where we consider $h[t]$ to be the set of blobs allowed to belong to track t.

4.5 The memory structure

The Memory structure consists of an array Tabu which maintains a tuple $\langle b, t\rangle$, where b is the blob, t is the track and the value stored in $\text{Tabu}(\langle b, t\rangle) = i$ represents the first iteration where the assignment of blob b to track t can be flipped again.

The Tabu tenure, i.e., the time τ a pair of blob and track $\langle b, t\rangle$ stays in the list, is dynamic: It is randomly generated in the interval [4, 100]. In other words, each time a blob in a track $\langle b, t\rangle$ is flipped, a random value τ is drawn uniformly from the interval [4, 100] and the pair $\langle b, t\rangle$ is Tabu for the next τ iterations ($\text{Tabu}(\langle b, t\rangle) = \tau$).

At iteration k, flipping the value of a pair $\langle b, t\rangle$ is Tabu, which is denoted by $\text{Tabu}\langle b, t\rangle$) if the Boolean expression $\langle b, t\rangle \in \text{Tabu}$ & $\langle b, t\rangle \leq k$ holds. As a consequence, for the complete assignment σ and iteration k, the neighborhood consists of the set of moves $\mathcal{S}^t(\sigma, k)$ defined as

$$\mathcal{S}^t(\sigma, k) = \{\langle b, t\rangle \in \mathcal{S}(\sigma) \mid \text{Tabu}(\langle b, t\rangle) \leq k\}$$

4.5.1 Aspiration criteria

Tabu Search usually introduces a mechanism called aspiration criteria by which a move which is Tabu can be perform if and only if the resulting cost function is better than the best solution found so far. Thus, the actual neighborhood is defined as follows:

$$\mathcal{S}^{t*}(\sigma, k) = \{\langle b, t\rangle \in \mathcal{S}(\sigma) \mid \text{Tabu}(\langle b, t\rangle) \leq k \text{ or } f(\{\langle b, t\rangle) < f^*\}$$

where $f(\langle b, t\rangle)$ denotes the cost of the tentative solution after the move and f^* is the cost of the best solution found so far.

4.6 The Tabu Search algorithm

We are now ready to present our metaheuristic. The algorithm, depicted in Fig. 6, is a Tabu Search with a restarting component. Lines 2–5 perform the initializations. In particular, the initial solution is generated randomly (or copied from the hypoth-esis) in line 2, while lines 4 and 5 initialize the iteration counter k, and the stability counter s. The best solution found so far σ^* is initialized to σ.

The core of the algorithm is given in lines 6–19. They repeatedly select a local move to perform until a maximum number of iterations is reached. The key idea is to select the best move in the neighborhood $\mathcal{S}^t(\sigma, k)$, i.e., the one that minimizes the optimization function. If there are several moves minimizing the optimization function one is randomly chosen. Observe that the expression $f(\sigma[\text{move}])$ represents the value of the optimization function obtained after a move (where a move is defined

```
1.   Tabu()
2.       σ ← random (or copied) configuration;
3.       σ* ← σ;
4.       k ← 0;
5.       s ← 0;
6.       while k ≤ maxIt
7.           select move = arg min{f(σ[move]) | move ∈ S^t(σ, k)};
8.           τ ← RANDOM([4,100]);
9.           Tabu_N ← Tabu_N ∪ {⟨move, k + τ⟩};
10.          σ ← σ[move];
11.          if f(σ) < f(σ*) then
12.              σ* ← σ;
13.              s ← 0;
14.          else if s > maxStable then
15.              σ ← random (or copied) configuration;
16.              s ← 0;
17.          else
18.              s++;
19.          k++;
```

Fig. 6 Tabu Search for blobs-to-tracks assignment problem

by the pair $\langle b, t \rangle$ to flip). The corresponding Tabu list is updated in line 9, and the new solution is computed in line 10, where we consider σ[move] to be the effect of performing the move in the current solution. Lines 11–13 update the best solution, while lines 14–16 specify the restarting component.

The restarting component simply reinitializes the search from a random (or copied) configuration whenever the best solution found so far has not been improved upon for maxStable iterations. Note that the stability counter s is incremented in line 18 and reset to zero in line 13 (when a new best solution is found) and in line 16 (when the search is restarted).

Notice that choosing a copied initialization will not necessarily yield the same local moves after each restart. This is due to the random component included both in the Tabu list and in the selection of the best move.

5 Several real life scenarios

The evaluation of our algorithm is tested against three different scenes (see Fig. 7). These datasets are from two different sources: the publicly available CVBASE dataset (University of Ljubljana Machine Vision Group 2001) and a DV camcorder. The datasets are quite diverse in their technical aspects, and the quality of the image sequences differs radically from poor to excellent along with their pixel resolutions.

5.1 Maritime scenes (BOAT)

The videos were recorded in an outdoor scenario using a DV video-recorder. The videos have a high quality, with a resolution of 720×480 pixels with 15 fps. The videos feature several boats in an outdoor environment lit by the sun. The videos are very interesting due to the complex segmentation of maritime scenes. The sea has continuous movement, which contributes to the creation of a great number of noisy blobs.

5.2 Squash tournament (SQUASH)

These videos are from the CVBASE dataset and were filmed during a tournament of amateur players. The videos were recorded using a S-VHS video-recorder, from a birds-eye view with wide angle lens. The videos were digitized to digital video format with 25 fps, with a resolution of 384×576 and M-JPEG compression. The selected video is a zenithal record of two players playing squash. They are in close proximity to each other, wear similar clothing, carry faster movements, and constantly cross in front of each other, all making for a challenging sequence.

5.3 Handball match (HANDBALL)

This video is also from the CVBASE dataset and it has the same characteristics as the squash tournament sequences. The players do not leave the court during the match, they constantly cross in front of each other, with occlusions and disocclusions, and the number of objects (players) per track is quite high. These conditions also make for a challenging sequence.

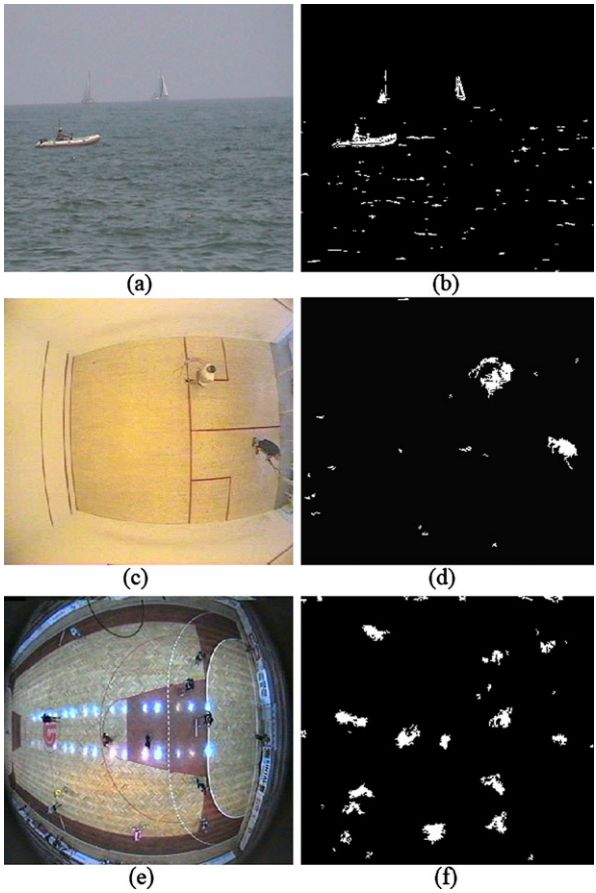

Fig. 7 Three scenes used in the evaluation process. Original images are depicted in the first column and their corresponding segmented images in the second column. Segmented images are the input of the tracking algorithms

6 Evaluation of algorithms

6.1 Evaluation metrics

Tracking methods can be evaluated on the basis of whether they generate correct mobile object trajectories. A qualitative comparison of tracking algorithms can be based on the ability to maintain the number of targets during the video sequence and to provide an optimal solution to the cost function minimization problem used for establishing correspondence (Yilmaz et al. 2006). Therefore, the metrics that allow us to provide formal comparisons among the algorithms tested are:

Tracks per Frame (TPF). It is used to compare the behavior of the tracking algorithms in terms of continuity of the tracks. An optimal tracker would have to obtain the value referred as 'ideal'. When the obtained value is below this 'ideal' value, it means that the tracker lost the continuity of the tracks (merge effect) and, conversely, when it is over the 'ideal' value, the tracker had an excess of tracks (split effect). The standard deviation of TPF allows for discriminating between behaviors with very similar averages but worse quality (greater deviation).

Frames per Second (FPS). This is the rate of processed images by the tracking algorithms; high values imply a greater capacity of processing.
Lost Track Probability (LTP). This determines the probability of losing a track on a given frame. Note that this measure has also been used in other frameworks (Kan and Krogmeier 1996).

6.2 Results

The system described here has been implemented in C++ under Microsoft Visual Studio, using the 'visual surveillance' algorithms incorporated in the Open Source Computer Vision Library (OpenCV). The system was tested on an AMD Athlon 64 Processor 3200+ with 2.01 GHz and 1 Gb of RAM. The OpenCV 'visual surveillance' algorithms use the pipeline structure depicted in Fig. 8. The input data for the pipeline is the image of the current frame and the output data is the information regarding track position and size. The 'FG/BG Detection' module performs foreground/background segmentation for each pixel; the 'Blob Entering Detection' module uses the result of 'the FG/BG Detection' module to detect new blob objects which entered the scene on each frame; and the 'Blob Tracking' module is initialized by the 'Blob Entering Detection' results and tracks each new blob that enters. This pipeline structure allows us to exchange different algorithms for the 'Blob Tracking' module, and to maintain the same execution conditions, i.e. using the same 'FG/BG Detection' and 'Blob Entering Detection' modules.

In Table 1, we show the six 'Blob Tracking' modules implemented in this work. Our intention is to compare the performance of our approximation with other 'Blob Tracking' modules, so we have fixed the same 'FG/BD Detection' and 'Blob Entering Detection' modules for every evaluation. This way, we can compare the results of the six different tracking modules. Regarding the 'FG/BD Detection' module, we have selected the OpenCV implementation of the adaptive background mixture models for real-time tracking (Stauffer and Grimson 1999). These five methods are described in

Fig. 8 The OpenCV 'visual surveillance' algorithms

Table 1 'Blob Tracking' algorithms used in the evaluation section

Algorithm	Description
CGA	Compact Genetic Algorithm (Harik et al. 1999)
UMDA	Univariate Marginal Distribution Algorithm (Mühlenbein 1997)
PBIL	Population-Based Incremental Learning (Cestnik 1990)
MSPF	Particle Filtering based on Mean-Shift weight (Chen et al. 2005; Comaniciu and Meer 2002; OpenCV)
GA	Genetic Algorithm (Goldberg 1989)
Tabu	Tabu Search Algorithm (Glover and Laguna 1993)

Table 2 Parameters of 'Blob Tracking' algorithms used in the evaluation section

Algorithm	Parameters
CGA	Learning Rate = 2000, Maximum number of generations = 10000
UMDA	Population = $8(\mu) + 100(\lambda)$, Maximum number of generations = 1000
PBIL	Learning Rate = 0.25, Maximum number of generations = 5000
MSPF	Number of particles = 100, Percent of particles which use velocity feature = 0.8, Size variation = 0.01, Position variation = 0.8
GA	Population = 100, Crossover Prob. = 0.8, Mutation Prob. = 0.01, Elitism = 0.2, Maximum number of generations = 5000

Table 3 Measures of quality of the algorithms applied to SQUASH

	Mean TPF (ideal = 2)	sd TPF	FPS	LTP
Tabu	1.90	0.25	14.56	0.003
UMDA	1.87	0.25	14.40	0.003
CGA	1.88	0.24	14.22	0.003
PBIL	1.89	0.23	12.31	0.002
GA	1.87	0.26	6.64	0.004
MSPF	1.90	0.24	5.74	0.005

the related work section and all of them, except the Particle Filtering—Mean Shift algorithm (MSPF), have been implemented in this work following their references. Finally, we have used the MSPF algorithm implemented in the OpenCV library.

In this work, we have implemented several Evolutionary Computation algorithms. Specifically, we have modeled the video-tracking problem using two families of evolutionary algorithms: Genetic Algorithm and Estimation of Distribution Algorithms (CGA, UMDA and PBIL). All of them have been implemented similarly to the Tabu Search algorithm, where the association consists of finding the appropriate values for the assignment matrix (see Fig. 2). To be able to use the evolutionary algorithm techniques, the assignment matrix is codified as a string of bits. The initial individuals are not randomly generated, but are fixed to solutions in which each blob is assigned to the closest object to assure an effective search. So, the search is performed over combinations starting from this solution to optimize the heuristic (similar to cost function, see Sect. 3.1) after any of this initial configuration is changed. The four algorithms have been implemented following the cited references: canonical GA (Goldberg 1989), CGA (Harik et al. 1999), UMDA (Mühlenbein 1997), and PBIL (Cestnik 1990). The parameter of the algorithms used in the experimentation are shown in Table 2.

In the following tables, we present the quality measurements of the EDAS, GAs, Particle Filtering and Tabu Search applied to the SQUASH (Table 3), BOAT (Table 4) and HANDBALL (Table 5) sequences.

Table 4 Measures of quality of the algorithms applied to BOAT

	Mean TPF (ideal = 3)	sd TPF	FPS	LTP
Tabu	2.983	0.067	4.88	0.000
CGA	2.983	0.067	4.29	0.002
UMDA	2.934	0.167	4.26	0.020
PBIL	2.983	0.067	4.04	0.019
MSPF	2.983	0.067	2.33	0.000
GA	2.932	0.166	2.21	0.024

Table 5 Measures of quality of the algorithms applied to HANDBALL

	Mean TPF (ideal = 14)	sd TPF	FPS	LTP
Tabu	12.321	1.684	7.40	0.10
CGA	10.769	1.095	0.81	0.20
PBIL	11.639	1.220	0.66	0.20
MSPF	12.920	0.523	0.56	0.22
UMDA	7.353	1.014	0.31	0.71
GA	12.403	1.821	0.04	0.43

All of the tables above show results ordered by a decreasing number of FPS. When all approaches are similar in terms of efficiency, the most important measure is the FPS, since it corresponds to the real time constraints.[4]

The first two tables show results for easy problems, and we can see that they are all comparable in terms of efficiency. It is hard to reproduce the actual behavior of the algorithms without watching the video scene itself; however, the measures used in this paper can give us an approximate idea of their quality. TPF shows how many tracks appear on every frame, which should be close to the actual number of tracks. However, this does not mean that these tracks are the real tracks we would like to target (noise instead of objects). Thus, this measure should be taken into account along with the LTP measure. This measure tells us how likely it is that we will lose a track using a given algorithm.

Turning to the third table, which corresponds to the most complicated scene, we can see that all of the algorithms perform similarly in most of the performance measures, although the differences are now more significant than in the other scenarios. In particular, we can clearly see how Tabu Search outperforms the rest of the approaches in terms of FPS while it compares with the best algorithms in terms of efficiency. This is very important because it means that now, with Tabu Search, we are able to perform video tracking for complicated scenes (namely more than 10 tracks and around 70 blobs) in real time. Note that all the previous approaches were able to process less than 1 frame per second, while Tabu Search is capable of processing more than 7.

[4]Real time is achieved when FPS is more than 5.

It is worth further examining the results in this last table. It might be surprising to see how Tabu Search outperforms the rest of the techniques, but it can be easily explained. Tabu Search can handle many more frames per second than the other approaches, which was not the case in the other examples (there were not many objects). The reason is that this last scenario contains a much larger number of blobs and tracks. Finding the optimal assignment matrix for the first scenarios was not very computationally demanding, while in this last case, more computation power is required. This is due to the explosion in the size of the search space. Thus, in this last case, Tabu Search is a much powerful technique (it has a greater greedy component) and given the auxiliary structures that allow incremental calculation of the fitness function, Tabu Search can solve the association problem much faster. This yields a better ratio of frames per second. The larger the search space, the larger the difference in performance.

7 Conclusions and future work

We have presented a Tabu Search algorithm, along with a novel technique for achieving incrementality, in order to solve the association problem in video tracking. The main contributions of our work are:

- Showing that local search techniques can be very useful for solving real life problems, especially when real time is a hard constraint.
- Introducing a novel technique for achieving incrementality when the search space is not the same as the solution space, i.e., the space needed to calculate the fitness of a candidate solution.
- Solving the association problem in video tracking more efficiently and much faster than with previous approaches. Real time is now possible for complicated video scenes.

Nevertheless, we are interested in doing further research on this problem. The main features we would like to address in the near future are:

- Improving the fitness function by adding other components such as color, texture, etc.
- Developing a self-tuning version of the algorithm so it can choose the right parameters for every instance of a problem. For example, increase the maximum number of iterations depending on the number of blobs and tracks, or switch to a different initialization depending on the blobs-to-tracks ratio.
- Trying a different strategy where Tabu Search is applied to each track separately. This would need several Tabu Search runs for each problem (as many as the number of tracks), but we could focus our efforts on the most problematic tracks (by increasing the allowed number of iterations in these cases) while relaxing the computation time spent on the easiest ones.
- Developing a hybrid evolutionary strategy where Tabu Search can be used as mutation operator.

Acknowledgements We would like to thank Pascal Van Hentenryck for his help and encouragement. We would also like to thank the reviewers for their useful comments.

References

Angus, J., Zhou, H., Bea, C., Becket-Lemus, L., Klose, J., Tubbs, S.: Genetic algorithms in passive tracking. Technical report, Claremont Graduate School, Math. Clinic Report (1993)

Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Trans. Signal Process. **50**(2), 174–188 (2002) [see also IEEE Transactions on Acoustics, Speech, and Signal Processing]

Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, CMU-CS, Pittsburgh, PA (1994)

Beymer, D., Konolige, K.: Real-time tracking of multiple people using continuous detection (1999)

Blackman, S.S., Popoli, R.: Design and Analysis of Modern Tracking Systems. Artech House, Norwood (1999)

Broida, T.J., Chellappa, R.: Estimation of object motion parameters from noisy images. IEEE Trans. Pattern Anal. Mach. Intell. **8**(1), 90–99 (1986)

Castanedo, F., Patricio, M.A., Garcia, J., Molina, J.M.: Extending surveillance systems capabilities using bdi cooperative sensor agents. In: VSSN '06: Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks, pp. 131–138. Assoc. Comput. Mach., New York (2006)

Cestnik, B.: Estimating probabilities: a crucial task in machine learning. In: ECAI, pp. 147–149 (1990)

Chang, Y.L., Aggarwal, J.K.: Neural network optimization for multi-target multi-sensor passive tracking. In: Proc. IEEE Workshop on Visual Motion, pp. 268–273 (1991)

Chen, T.P., Haussecker, H., Bovyrin, A., Belenov, R., Rodyushkin, K., Kuranov, A., Eruhimov, V.: Computer vision workload analysis: case study of video surveillance systems. Intel Technol. J. **9**(2), 109–118 (2005)

Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)

Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 564–575 (2003)

Cordon, O., Damas, S.: Image registration with iterated local search. J. Heuristics **12**(1–2), 73–94 (2006)

Cox, I.J., Hingorani, S.L.: An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. IEEE Trans. Pattern Anal. Mach. Intell. **18**(2), 138–150 (1996)

de Bonet, J.S., Isbell, C.L., Jr., Viola, P.: MIMIC: Finding optima by estimating probability densities. In: Mozer, M.C., Jordan, M.I., Petsche, T. (eds.) Advances in Neural Information Processing Systems, vol. 9, p. 424. MIT Press, Cambridge (1997)

Djuric, P.M., Kotecha, J.H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M.F., Miguez, J.: Particle filtering. IEEE Signal Process. Mag. 19–38 (2003)

Dotu, I., Van Hentenryck, P.: Scheduling social golfers locally. In: CP-AIOR-05 (2005)

Ferryman, J.M., Maybank, S.J., Worrall, A.D.: Visual surveillance for moving vehicles. Int. J. Comput. Vis. **37**(2), 187–197 (2000)

Glover, F., Laguna, M.: Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific, Oxford (1993)

Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison–Wesley, Reading (1989)

Grunert, F., Funke, B., Irnich, S.: Local search for vehicle routing and scheduling problems: review and conceptual integration. J. Heuristics **11**(4), 267–306 (2005)

Han, M., Xu, W., Tao, H., Gong, Y.: An algorithm for multiple object trajectory tracking. In: CVPR 2004: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 01, pp. 864–871 (2004)

Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Trans. Evol. Comput. **3**(4), 287 (1999)

Harvey, W., Winterer, T.: Solving the MOLR and social golfers problems. In: CP-05 (2005)

Hillis, D.B.: Using a genetic algorithm for multi-hypothesis tracking. In: ICTAI '97: Proceedings of the 9th International Conference on Tools with Artificial Intelligence, p. 112. IEEE Computer Society, Washington (1997)

Huwer, S., Niemann, H.: 2d-object tracking based on projection-histograms. In: 5th European Conference on Computer Vision, pp. 861–876 (1998)

Kan, W.Y., Krogmeier, J.V.: A generalization of the pda target tracking algorithm using hypothesis clustering. Sign. Syst. Comput. **2**, 878–882 (1996)

Kincaid, R.K., Laba, K.E.: Reactive Tabu Search and sensor selection in active structural acoustic control problems. J. Heuristics **4**(3), 199–220 (1998)

Kitagawa, G.: Non-Gaussian state-space modeling of nonstationary time series. J. Am. Stat. Assoc. **82**, 1032–1063 (1987)

Larraaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic, Norwell (2001)

Mühlenbein, H.: The equation for response to selection and its use for prediction. Evol. Comput. **5**(3), 303–346 (1997)

Mühlenbein, H., Mahnig, T.: The factorized distribution algorithm for additively decompressed functions. In: 1999 Congress on Evolutionary Computation, pp. 752–759. IEEE Service Center, Piscataway (1999)

OpenCV. intel.com/technology/computing/opencv/index.htm

Patricio, M.A., Carbó, J., Pérez, O., García, J., Molina, J.M.: Multi-agent framework in visual sensor networks. EURASIP J. Adv. Signal Process. **2007**, 98639 (2007a). doi:10.1155/2007/98639, 21 pp.

Patricio, M.A., García, J., Berlanga, A., Molina, J.M.: Video tracking association problem using estimation of distribution algorithms in complex scenes. In: Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach: First International Work-Conference on the Interplay Between Natural and Artificial Computation. Lecture Notes in Computer Science. Springer, Berlin (2007b)

Patricio, M.A., Castanedo, F., Berlanga, A., Perez, O., Garcia, J., Molina, J.M.: Computational intelligence in visual sensor networks: improving video processing systems. In: Computational Intelligence in Multimedia Processing: Recent Advances. Studies in Computational Intelligence, vol. 96, pp. 351–377. Springer, Berlin (2008)

Pisinger, D., Faroe, O., Zachariasen, M.: Guided local search for final placement VLSI design. J. Heuristics **9**(3), 269–295 (2003)

Regazzoni, C.S., Vernazza, G., Fabri, G. (eds.): Highway Traffic Monitoring. Kluwer Academic, Norwell (1998a)

Regazzoni, C.S., Vernazza, G., Fabri, G. (eds.): Security in Ports: the User Requirements for Surveillance System. Kluwer Academic, Norwell (1998b)

Ruan, Y., Willett, P.: Multiple model pmht and its application to the benchmark radar tracking problem. IEEE Trans. Aerosp. Electron. Syst. **40**(4), 1337–1350 (2004)

Shams, S.: Neural network optimization for multi-target multi-sensor passive tracking. In: Proceedings of the IEEE, vol. 84, pp. 1442–1457 (1996)

Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2. IEEE Computer Society, Los Alamitos (1999)

University of Ljubljana Machine Vision Group. Cvbase '06 workshop on computer vision based analysis in sport environments, found at url: http://vision.fe.uni-lj.si/cvbase06/ (2001)

Xiao-Rong, L., Bar-Shalom, Y.: Multitarget-Multisensor Tracking. Principles and Techniques (1995)

Yeddanapudi, M., Bar-Shalom, Y., Pattipati, K.: IMM estimation for multitarget-multisensor air traffic surveillance. In: Proceedings of the IEEE, vol. 85, pp. 80–96 (1997)

Yilmaz, A., Javed, O., Shah, M.: Object tracking: a survey. ACM Comput. Surv. **38**(4), 13 (2006)