A Framework for Inter-camera Association of Multi-target Trajectories by Invariant Target Models

Shahar Daliyot[1,2] and Nathan S. Netanyahu[1,3]

[1] Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel
[2] Verint Systems Ltd., Herzliya 46733, Israel
[3] Center for Automation Research, University of Maryland, College Park, MD 20742, USA

Abstract. We propose a novel framework for associating multi-target trajectories across multiple non-overlapping views (cameras) by constructing an invariant model per each observed target. Ideally, these models represent the targets in a unique manner. The models are constructed by generating synthetic images that simulate how targets would be seen from different viewpoints. Our framework does not require any training or other supervised phases. Also, we do not make use of spatiotemporal coordinates of trajectories, i.e., our framework seamlessly works with both overlapping and non-overlapping field-of-views (FOVs) as well as widely separated ones. Also, contrary to many other related works, we do not try to estimate the relationship between cameras that tends to be error prone in environments like airports or supermarkets where targets wander about different areas, stop at times, or turn back to their starting location. We show the results obtained by our framework on a rather challenging dataset. Also, we propose a black-box approach based on Support Vector Machine (SVM) for fusing multiple pertinent algorithms and demonstrate the added value of our framework with respect to some basic techniques.

1 Introduction

Tracking targets across multiple cameras, also known as the "handover" problem, is an important problem in computer vision, in general, and visual surveillance, in particular, where common applications are motivated by large surveillance systems installed in complex compounds. The goal is to maintain the identities of targets traveling across cameras, disappearing and later reappearing at a different location. The task of target tracking in a single view (single camera) has been studied quite extensively (see, e.g., [1,3,18]) and is not the focus of this work. We assume, in fact, that this fundamental building block is given. Tracking analysis of a single view produces a set of trajectories representing the target motions in a given FOV, independently from other views. Our goal is to associate trajectories belonging to the same target from multiple, independently-analyzed views.

To make aforementioned associations between trajectories, we compute a signature for each trajectory. This signature consists of models representing the target that originated the trajectory. Ideally, trajectory signatures belonging to the same target are more similar than those belonging to different targets. We obtain associations based on construction of invariant features. As opposed to common feature extraction techniques (e.g., SIFT [13]) that search for invariant features with respect to scale, rotation, etc., we explicitly promote invariance to any feature extracted from an observed target. We do that by generating synthetic images that simulate how the target would be seen from different viewpoints. This approach is different from various common approaches utilized in related work, whereby probability models evaluate inter-camera relationships to predict the associations between trajectories.

Our approach does not rely on spatiotemporal coordinates, color cues, and similar information that is usually exploited in other related work. Instead, by using our innovative scheme of invariant signature models constructed from a target's trajectory, we employ, essentially, an alternative technique for invariant feature extraction to finding trajectory association. Our solution can then be fused nicely with other, more common approaches. Indeed, we present also an SVM-based black-box fusion scheme that can easily be used to combine multiple pertinent algorithms and demonstrate its added value.

This paper is organized as follows. Section 2 provides an overview of related research in this area. Section 3 presents our framework for trajectory association. Section 4 presents our black-box approach for fusing a variety of pertinent algorithms. In Section 5 we present our experimental results. Section 6 makes concluding remarks.

2 Related Work

Recently, there has been a growing interest in tracking targets between blind regions, i.e., non-overlapping regions, which give rise to target disappearances at certain time intervals. In [4] a system for multi-camera tracking in "blind" regions is described. In order to match targets between different cameras, the authors assume that targets move in a fairly constant velocity. They utilize a known camera topology to construct a probabilistic prediction of where and when a target, disappearing from one view, will appear in a nearby view. This assumption does not hold in a supermarket-like environment where people wander about the different aisles, stop at times, or turn back to their starting location. In [9] a different approach is taken, whereby the camera topology and path probabilities are learned during a training phase. This method is constrained to a small number of cameras. It is quite cumbersome and even infeasible to have such a training phase when dealing with large scale premises with hundreds of cameras. On the other hand, an unsupervised method for learning the topology of a camera network is presented in [14]. The system learns "Entry" and "Exit" zones in each camera view according to a training dataset that contains a large number of sample trajectories. Then, a probabilistic graph representing links between these zones is learned based on the training dataset, using expectation maximization (EM) methods. The resulting graph is used to predict the location where a disappeared target might reappear. This unsupervised approach of learning a prediction model might suffer from poor accuracy in a large camera network with substantial amount of targets moving around, constantly disappearing and reappearing. The model might set high probabilities to links between "Entry" and "Exit" zones based on trajectories that are assumed to represent the same target but which in fact do not.

A very common approach of handling the problem of tracking across multiple disjoint views is based on color histogram comparisons (see, e.g., [8,10,11,15,17]). Issues such as differences in illumination, pose, and internal parameters between different cameras are discussed in [10]. The authors show that all brightness transfer functions, from a given camera to another, lie in a low-dimensional subspace which can be learned by supervised methods using labeled learning sets. In [11] the authors deal with the problem of tracking across multiple cameras by combining location prediction and appearance model matching. The location prediction is based on Kalman filters (KF), and the appearance model is constructed from multiple color distribution components, each of which is obtained by partitioning the blobs (representing the detected target) into their polar representation. This localization of color properties provides an advantage over a simple color model, whereby a single histogram represents the entire target.

The main task of our work is to find associations between trajectories of targets seen from several arbitrary viewpoints. Approaching the problem is often based on feature matching techniques, which have been studied extensively (see, e.g., [2,7,13,15,16]). The methods presented in these papers extract features that are invariant to affine transformations (i.e., scale, rotation, and shear), noise, illumination, etc. Such features can then be matched, regardless of the viewpoint of a target or an object. This insight serves as an inspiration for our target association scheme.

Fusion methods have also been introduced extensively in this domain. In [10] the authors combine space-time cues with an appearance matching scheme based on a Maximum Likelihood (ML) estimation framework. In order to overcome appearance changes between cameras, a brightness transfer function is learned during a training phase. Space-time cues are also learned during this training phase. The ML estimation framework is then used to assign correspondences between tracked targets. In [12] the authors combine three different feature types (the color histogram, covariance matrix, and Histogram of Oriented Gradients (HOG) features) using the Multiple Instance Learning (MIL) method. Similarity measures based on these features, among a set of automatically collected training samples, form a "feature pool". The MIL boosting algorithm is then applied to select discriminative features from this pool and their corresponding weighted coefficients. We propose a simple, yet effective black-box approach for fusing a variety of pertinent algorithms without any knowledge of the internal structure of the fused algorithms. This is in contrast to the above approaches whereby fusion is accomplished within the core of the algorithms, requiring the utilization of heuristics which makes it more complex.

3 Our Trajectory Association Framework

To associate trajectories across multiple views, we will compute a signature for each trajectory. These signatures will be used subsequently to compare between trajectories and obtain associations. Subsections 3.2–3.3 describe the process of signature computation and comparison.

3.1 Model Construction

We refer to the constructed model that describes, ideally, the target of a trajectory in a unique manner as an “invariant feature”. The following paragraphs describe in detail the model construction.

ROI Selection. Contrary to the approach taken in many related works, we compute the descriptor over a specific region of interest (ROI), rather than the entire region where the target is captured. We will be interested specifically in regions containing a persons head. This is motivated by several reasons: (1) The head includes facial features that identify the target uniquely. (2) The head forms a fairly rigid surface relative to other moving parts of the human body like the arms. This helps when comparing descriptors computed from different viewpoints of the same target. (3) The head is the topmost body part, making it less prone to occlusions. (4) In practice, the region we use includes not only the head but also the upper body part that captures slightly areas from the neck and shoulders. This upper body part is still fairly rigid, yet often includes distinctive, informative characteristics about the target, e.g., texture of shirt, type of collar, tie, scarf, etc.

We extract the head region automatically from the bounding ellipse of the target in the frame obtained by a visual tracker. Calculating a bounding rectangle of the head region relative to the bounding ellipse is based on fixed coefficients learned a prior, as illustrated in Figure 1.

Descriptor Computation. Once a region is determined, a descriptor is computed from the image data in the region. We use a Histogram of Oriented Gradients (HOG) proposed in [7] to represent the data in the region. Namely, we compute a HOG feature by concatenating 8-bin orientation histograms calculated in 6×6 cells over the region, resulting in a row vector of size 288.

We note that the specific HOG feature we chose as a descriptor is not necessarily optimal for the task of trajectory association. Other types of descriptors could be used coherently with our framework. The type of descriptor used is not as important as the invariance required from the constructed model to changes in scale, rotation, viewpoint, and to some extent, also perspective distortions. This construction is the foundation of our framework as described henceforth.

Invariant Model Construction. Computing a single descriptor for the head region in the original frame (as captured by the camera) is not enough to match

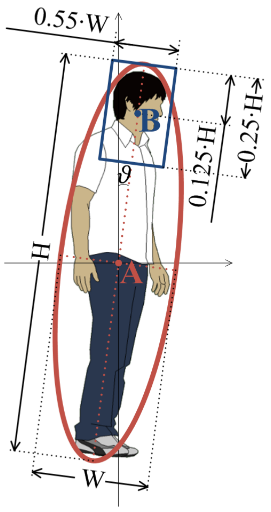

Fig. 1. Positioning of rectangle representing region of head relative to bounding ellipse of target; height and width of bounding rectangle are set with respect to lengths of major and minor axes of ellipse ($0.25H$ and $0.55W$, respectively); center of rectangle (point B) is located on major axis; rectangle (aligned with major and minor axes of ellipse) is tilted by an angle of ϑ

this instance of the head with other instances of the same target, taken from different viewpoints. Therefore, we simulate, as much as possible, how a given target would be seen from different viewpoints. We do that by applying various affine transformations to the ROI in the original frame and then computing the corresponding descriptors of these transformed regions. We realize that it is not possible to construct a full 3D model of a face given the data quality we deal with. Still, applying rather "small" affine transformations provides various simulated viewpoints of the head.

An affine transformation can be represented by the seven parameters: scale-x, scale-y, translation-x, translation-y, shear-x, shear-y and rotation. We denote these parameters by S_x, S_y, T_x, T_y, R_x, R_y, and ϑ, respectively. Each of the seven parameters can be expressed by a 3×3 transformation matrix of homogeneous coordinates. These parameters can be combined to form a variety of affine transformations by using matrix composition. Recall that the head region is captured by a rectangle, i.e., it can be represented by a 3×3 affine transformation matrix (with respect to a reference 1×1 rectangle centered at the origin) that is composed of several transformation matrices of homogeneous coordinates. Let G be that rectangular head region centered at point $C = (C_x, C_y)$ with width W, height H, and counter-clockwise rotation ϑ' with respect to C. The composition corresponding to G is given by

$$\begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\vartheta & -\sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & R_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ R_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

where

$$S_x = W \ ; \ S_y = H \ ; \ T_x = C_x \ ; \ T_y = C_y \ ; \ R_x = 0 \ ; \ R_y = 0 \ ; \ \vartheta = \vartheta' \tag{2}$$

We compute $N \gg 1$ descriptors, corresponding essentially to N different affine transformations applied to the original head region. It is important that these transformations be diversified, albeit rather slightly, to avoid an intense distortion of the information contained in the relevant region. Having experimented with various values, we picked eventually $N = 1000$. (A larger number of transformations did not yield significantly better results.) We thus generate at random 1000 affine transformations as follows. Each affine transformation is associated with a 7-element list of the form $(\Delta S_x, \Delta S_y, \Delta T_x, \Delta T_y, \Delta R_x, \Delta R_y, \Delta\vartheta)$, where each element is chosen at random from the range $[-0.2, +0.2]$, i.e., each random generation corresponds to a perturbation of the given image by an affine transformation with the following parameters:

$$S_x = 1 + \Delta S_x \ ; \ S_y = 1 + \Delta S_y \ ; \ T_x = \Delta T_x \ ; \ T_y = \Delta T_y$$

$$R_x = \Delta R_x \ ; \ R_y = \Delta R_y \ ; \ \vartheta = \Delta\vartheta \tag{3}$$

(with ϑ in radians). We end up with a list containing 1000 descriptors and a list containing corresponding representations of 1000 affine transformations. That is, for each i, $1 \leq i \leq 1000$, descriptor d_i corresponds to the set of coefficients of affine transformation t_i. Eventually, associations are determined by an iterative algorithm; at each iteration, given a descriptor d, it looks for the affine transformation t that "best fits" this descriptor. We have considered, among others, perspective transformations and non-linear transformations, but settled eventually for a simpler linear mapping according to the standard least squares approach. Let D be the domain of descriptors and let T be the domain of affine transformations. We want to determine a linear mapping $L : D \to T$. Since the output $t \in T$ consists of seven parameters, there is a need to determine seven linear mappings, $L_1, \ldots, L_7$, each of which corresponds to a single parameter of the ultimate affine transformation. Finding these linear mappings requires solving an overdetermined system of linear equations. Each of these seven linear mappings is essentially a list of 288 coefficients corresponding to the elements of a descriptor. The linear least squares method finds these coefficients, such that the error over the entire data (1000 instances in our case) is minimized. Once the seven linear mappings are determined, we concatenate them to form the final 288×7 matrix L. Given a descriptor d, the corresponding affine transformation t can be computed by:

$$\left[t\right]_{1\times 7} = \left[d\right]_{1\times 288} \cdot \left[L\right]_{288\times 7} \tag{4}$$

We can construct also the model that represents the head of a target in the frame using the data computed. The model consists of the following four items: (1) The vertical and horizontal gradient images calculated from the frame during the computation of the HOG feature. (2) *model-region*: the head region extracted from the bounding box of the target. (3) *model-descriptor*: a single descriptor computed for the *model-region*. (4) The linear mapping matrix L.

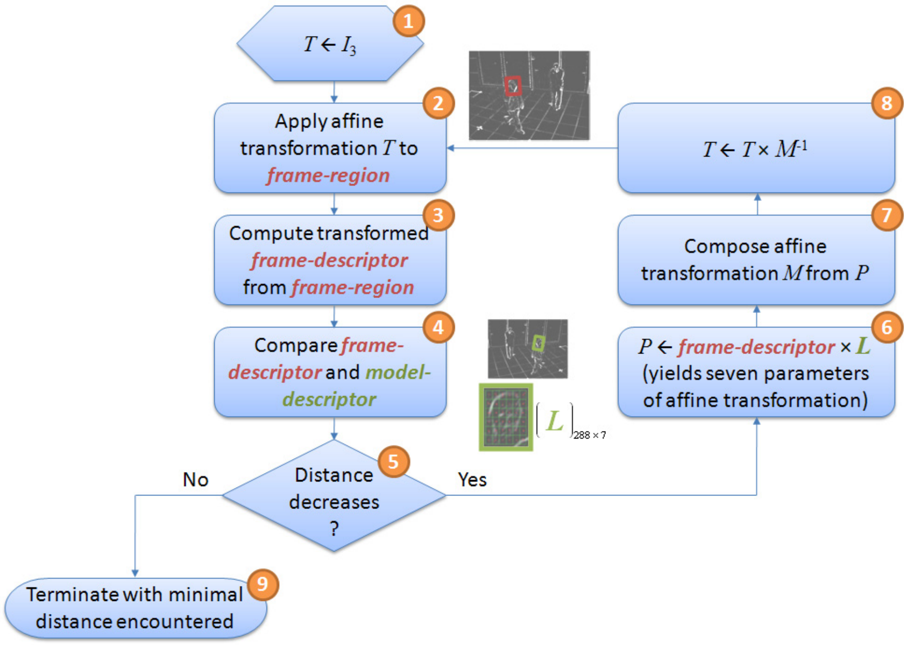
$T \leftarrow I_3$
Apply affine transformation T to *frame-region*
Compute transformed *frame-descriptor* from *frame-region*
Compare *frame-descriptor* and *model-descriptor*
Distance decreases ?
No
Yes
Terminate with minimal distance encountered
$P \leftarrow$ *frame-descriptor* × L (yields seven parameters of affine transformation)
Compose affine transformation M from P
$T \leftarrow T \times M^{-1}$
$[L]_{288 \times 7}$
1
2
3
4
5
6
7
8
9

Fig. 2. Flowchart of iterative descriptor-to-model comparison

3.2 Signature Computation

Note that a model represents only a single target in a single frame. The purpose of a signature is to represent an entire trajectory, which spans usually multiple frames. A signature is a collection of models computed from frames of the trajectory. Through-out the frames of a trajectory, the target is captured from different viewpoints and in different poses. However, the target may look very similar in many frames. We want to keep only the frames in which the target looks different with respect to other frames of the trajectory. We start with an empty signature and append to it models from each frame that introduces new data not yet encountered. Starting with the first frame, the signature is still empty; hence, a model is always computed from this frame and added to the signature. Examining consecutive frames, the head in each frame is compared to models already appended to the signature in previous frames. If a head model is "similar enough" to any of the previous models, the frame in question is skipped. On the other hand, if the head is "distinct enough" from all previous models, a new model is computed for the current frame and appended to the signature.

Descriptor-to-Model Comparison. A fundamental element of the signature construction algorithm is the ability to compare a targets head in a given frame to models, which already reside in the signature. That is, to compute the distance between a given head and a model. Once we compute this distance, a simple threshold can be used to decide whether the head in question is "similar enough" to any of the models in the signature. We start by computing a single descriptor for the head region in a given frame. We will refer to this head region as *frame-region* and to the corresponding descriptor computed for that region as *frame-descriptor*. To compute the aforementioned distance, we use an iterative approach. The iterative descriptor-to-model comparison approximates the *frame-descriptor* to the *model-descriptor*. This is done by iteratively applying an (inverse) affine transformation to the *frame-region*, re-computing a descriptor for the resulting region and comparing it to the *model-descriptor*. Computing the (inverse) affine transformation that should be applied to the *frame-region* is done using the linear mapping matrix L and the descriptor obtained at each iteration. Figure 2 presents a flowchart of the iterative descriptor-to-model comparison. At the first iteration, the algorithm computes in Step 3 the *frame-descriptor* from the *frame-region*. Initially, the affine transformation T applied to the *frame-region* in Step 2 is the identity matrix, so that there is no effect on the region. In consecutive iterations, however, the *frame-region* will be transformed, yielding a so-called transformed *frame-descriptor*. In Step 4, the transformed *frame-descriptor* is compared to the *model-descriptor*, yielding a distance between the two. (The distance is the angle (in degrees) between the two unit vectors.) If the distance decreases, the algorithm continues to Step 6. Otherwise, the algorithm terminates, returning the minimal distance encountered so far. Steps 6-8 comprise the essence of our framework. First, the transformed *frame-descriptor* is multiplied by the linear regression matrix of the model, resulting in an affine transformation represented by a vector of seven parameters. Recall that the linear mapping matrix was calculated from corresponding pairs of affine transformations and descriptors. Intuitively, the product of an arbitrary descriptor and the linear regression matrix of a model yields an affine transformation that should approximate in some sense the descriptor of the model head to that arbitrary descriptor. Namely, the affine transformation obtained should approximate the *model-descriptor* to the *frame-descriptor*. However, since we want to approximate actually the *frame-descriptor* to the *model-descriptor*, we need to apply the inverse affine transformation to the *frame-region*. Steps 7 and 8 do exactly that. In Step 7 an affine transformation is composed out of the seven parameters obtained by the multiplication, and in Step 8 the inverse of this affine transformation is applied to the affine transformation T that process adjusts the (inverse) affine transformation applied to the *frame-region*, as long as the distance between the *frame-descriptor* and the *model-descriptor* continues to decrease.

Horizontal Flipping. We exploit the fact that a human head is usually symmetric along the horizontal axis to enhance a signature computation. In addition to comparing the descriptor representing a head region in a given frame to the models contained in the signature, we will compare also a descriptor derived from a horizontally-flipped image of the frame. The final distance between the head in a given frame and a head model is defined as the minimum distance between the *model-descriptor* and the two descriptors, i.e., the one computed for the original frame and that computed for the horizontally-flipped frame. This enables to reduce the number of models contained in the signature, thereby reducing computation time and space. Also, we exploit this flipping element, while comparing between signatures, to improve significantly the total matching rate (see next subsection).

3.3 Signature Comparison

Before presenting the algorithm for signature comparison, *CompareSignatures*, we define an algorithm for comparing models, *CompareModels*. In the previous subsection we presented an algorithm for comparing a single descriptor to a model, the iterative descriptor-to-model comparison. We later obtained an enhanced algorithm, *ModelWithHorzFlip*, using horizontal flipping. We utilized this enhanced algorithm for comparing also between models. Given two models, *model1* and *model2*, *CompareModels* compares the two models and returns a scalar representing the distance between them. The algorithm is symmetric, i.e., *model1* is compared to *model2* and vice versa. First the algorithm calls *CompDescToModelWithHorzFlip* with input information of the frame-region from *model1* and *model2* as the *model-descriptor*. This means that transformed descriptors are computed from the gradient images of *model1* and approximated to the *model-descriptor* contained in *model2* using the linear mapping matrix of *model2*. Then, the opposite comparison is performed, i.e., transformed descriptors computed from the gradient images of *model2* are approximated to the *model-descriptor* contained in *model1*. The distance between the two models is defined as the average between the values returned by the two comparisons performed.

We now sketch the *CompareSignatures* algorithm. Given two signatures, *sig1* and *sig2*, all models of *sig1* are compared to all models of *sig2* using the *CompareModels* algorithm. The minimum distance among all comparisons is returned as the distance between the two signatures. Final trajectory associations are obtained based on these distances by comparing them, for example, against a specified threshold.

4 Fusing Multiple Algorithms

The insight of establishing an improved outcome based on the fusion of multiple algorithms has long been recognized. Fusion methods have been introduced extensively in the domain of object matching across different views (see, e.g., [10,12]). We propose a simple, yet effective black-box approach for fusing a variety of matching algorithms. We let every algorithm run independently and return a distance (or grade) that is assigned to each pair of trajectories among a set of labeled training samples. Once the results from all algorithms are available, we train a Support Vector Machine (SVM) module [5] using the labeled training samples based on the distances obtained by the different algorithms. Samples of the SVM training set are compiled from all possible pairs of trajectories from the labeled training samples. Each sample has several features that are captured by the distances provided by the different algorithms for that specific sample. (Note that a sample represents a pair of trajectories; hence a distance or a grade is assigned to this pair by each algorithm.) The number of sample features is equal to the number of different algorithms that are fused. The sample class is assigned a value according to ground truth; if the two trajectories in the sample represent the same target, the class is set to 1, and otherwise to 0. The outcome of the training process is an SVM module that accepts a list of distances obtained by the different algorithms for a given pair of trajectories. The output of the SVM module is one of two possible classes, 1 and 0, as defined above. In order to assess the quality of the SVM module that was constructed, we used the well-known K-fold cross-validation technique (with $K = 10$). The accuracy was stable across all 10 repetitions, i.e., the variance between the 10 results was very low. This indicates that the module is well-balanced and coherent. Results are presented in the next section.

Using a labeled training set should not imply there is a need for a training phase or some supervised procedure each time the fusion is invoked with a new camera network. The labeled set is utilized to train an SVM module on the outputs of the fused algorithms. Theoretically, the same labeled dataset can be utilized repeatedly for different invocations and different algorithms. Of course, using a training set created from actual data of a given camera network is likely to yield better results. Note that the fusion is adapted in a modular manner upon the introduction of a new algorithm. One need only train a modified SVM module with the new additional data obtained by that algorithm, which is an unsupervised procedure that can be handled automatically.

The above is a "black-box" approach since no knowledge of the internal structure of the fused algorithms is needed. The only requirement from an additional fused algorithm is a coherent output of the distance (or grade) between all trajectory pairs. This is a rather elementary requirement from an algorithm whose purpose is to find associations between trajectories.

5 Experimental Results

All experiments were done using a 30-minute recorded scene of six cameras installed at the lobby of a facility according to the model shown in Figure 3. The views of the six cameras are shown in Figure 4. The scene includes both congested and sparse scenarios. A visual tracker was used to analyze separately the recorded video of each of the six cameras and produce trajectories of detected targets. Trajectories of 12 targets in the scene were manually labeled to form a dataset that serves as ground-truth. The 12 targets produced 98 trajectories in the scene. Each of the 98 trajectories was compared to all trajectories (including itself), yielding distances per each trajectory pair, i.e., ($\frac{1}{2} \times 98 \times 99 =$) 4851 entries altogether.

Figure 5 presents precision vs. recall results for our trajectory association framework before and after the horizontal flipping. (Precision and recall are determined

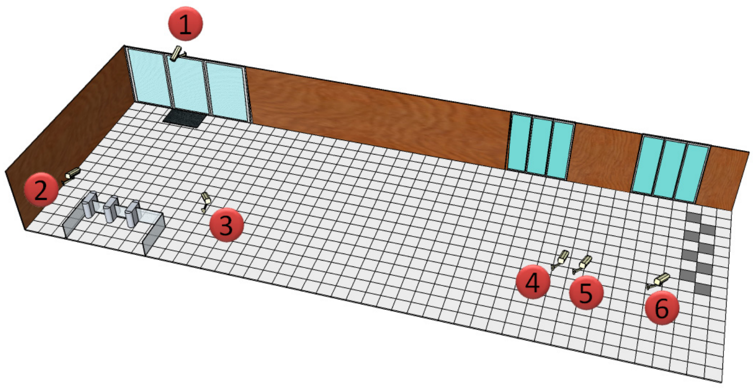

Fig. 3. Model of lobby showing positions of six cameras partially covering the area

by the number of "true-positives", "false-positives", and "false-negatives" returned by our algorithm over all trajectory pairs.) The optimal accuracy of 82% is quite impressive given the challenging dataset we experimented with. The accuracy is given by the F-measure (also known as the F_1-score), which is the harmonic mean of precision and recall, i.e.,

$$F\text{-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

As mentioned in the Introduction, our algorithm for trajectory association does not consider spatiotemporal coordinates, color cues, or similar information that is exploited in other related work. Instead, it introduces a new type of information based on invariant features. This approach lends itself to further improvement by fusing it with other more common approaches. To validate this assumption, we implemented two additional basic algorithms and examined the results due to fusion of all three. The first additional algorithm is based on color histograms. That is, the distance between two trajectories is based on the Euclidean distance between the color histograms computed from the blobs of the relevant targets. Color histograms were constructed based on the three RGB channels of the frames. We allocate 16 bins to each of the three channels. We then concatenate them and normalize the resulting 48-bin histogram. The accuracy obtained by this algorithm is 76%. The second additional algorithm is based on spatiotemporal information. All trajectories from all cameras are projected into a common coordinate system and a grade is assigned to each pair of trajectories as follows. Two trajectories recorded at the same time frame but in distant locations are assigned a grade of -1, meaning they belong to different targets. This is based on the fact that a target cannot be in two different locations at the same time. A grade in the range $(0, 1]$ is assigned to two trajectories recorded at the same time frame that are in close vicinity. The exact grade is computed based on the spatial distance between the two trajectories. Finally, a grade of 0 is assigned to all other trajectory pairs, indicating the algorithm does not have sufficient

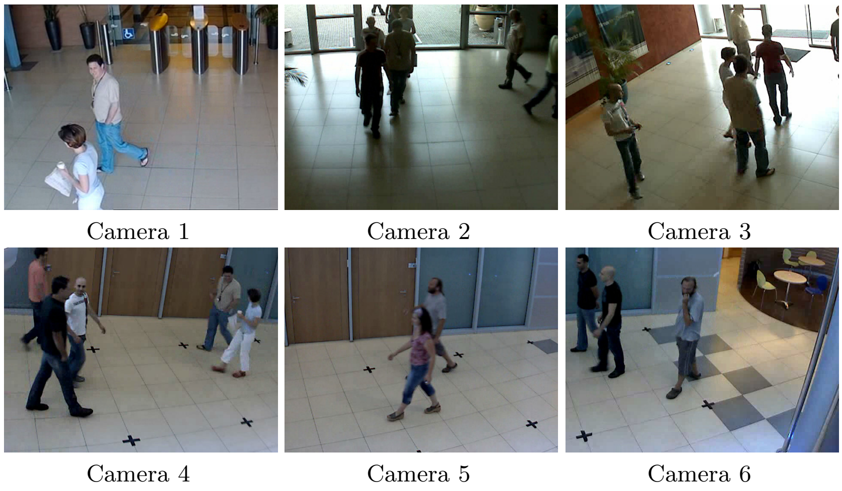

Fig. 4. Views of six cameras positioned according to model presented in Figure 3

information to obtain a reliable decision. The accuracy obtained by this algorithm for our dataset is 35%. This poor outcome is a result of the inability of the algorithm to compare between trajectories that are not adjacent in time or space. When executed only on a subset of the trajectories that do have adjacent spatiotemporal information it produces an impressive accuracy of 95%.

We fused our algorithmic framework with the color-based and spatiotemporal-based algorithms using our black-box technique. Fusing the three algorithms is pretty straightforward; we let every algorithm run independently and yield its own results. A sample (point in the graph) provided as input to the SVM module represents a pair of trajectories and consists of the distances/grades obtained by the three algorithms for that pair. A 10-fold cross validation procedure was executed to evaluate the accuracy of the SVM module. Figure 6 presents the final point classification according to the support vectors that were computed by the SVM module in one of the 10 iterations of the cross validation procedure. Precision, recall, and accuracy (F-measure) were calculated during each iteration and then averaged to yield the ultimate measures of the fusion. The bottom-line averages obtained for precision, recall, and accuracy are 94%, 83%, and 88%, respectively. Recall that the accuracies obtained when running the three algorithms independently were 35%, 76%, and 82% by the spatiotemporal-based algorithm (run on the entire dataset), the color-based algorithm, and our framework, respectively. Clearly, the bottom-line 88% accuracy obtained by the fusion comprises an improvement in comparison to each of the three independent algorithms. Evidently, the fact that each algorithm obtains trajectory associations based on a different type of information causes a significant mutual improvement.

Having established the added value of our framework relatively to the above basic approaches, it would be of interest, of course, to carry out a more comprehensive study with respect to various recent methods, e.g., [6,19].

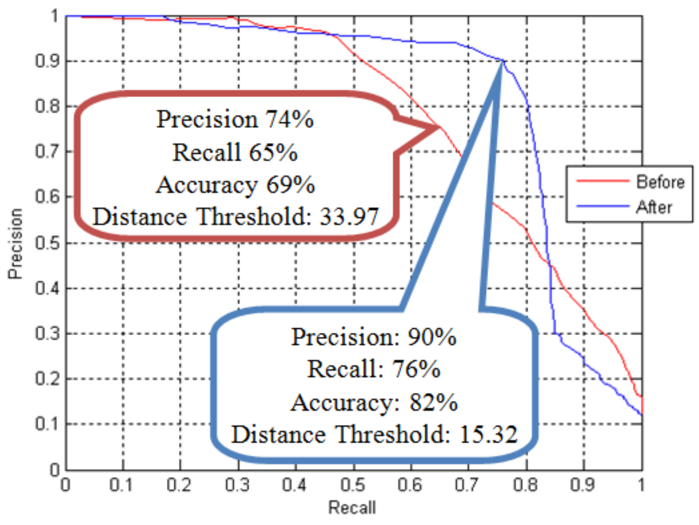

Fig. 5. Precision vs. recall before (red) and after (blue) horizontal flipping; note gain in accuracy of over 10%

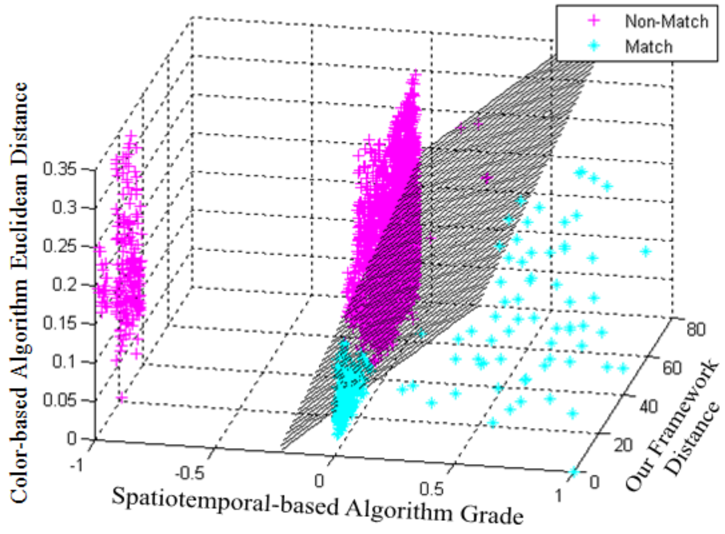

Fig. 6. Final classification of all pairs according to SVM module fusing our framework, color-based, and spatiotemporal-based algorithms: Pairs with positive classification, i.e., matches (cyan), pairs with negative classification, i.e., non-matches (magenta), and the planar separator computed by the SVM module

6 Conclusions

We have presented a novel basic framework for constructing an invariant descriptive model of a target for trajectory association across multiple views. Unlike many other related works, our system does not assume a constant velocity of moving targets and does not require a learning phase to construct a statistical model of target motions. Instead, we showed how to construct an invariant model for a target to represent it uniquely across multiple views. The model is constructed by generating synthetic target images that simulate how a target would be seen from different viewpoints. A regression analysis is then used to capture this simulated model in a compact and efficient manner. Next, an iterative convergence scheme is employed to compare between constructed models and find trajectory correspondences. We demonstrated the relatively high success rate of this approach, especially when fused with other, more common methods. Specifically, we examined fusion with two common techniques based on color histograms and spatiotemporal information. We showed that fusing the results from these common techniques with those of our framework yields an evident gain in accuracy. To carry out the fusion we introduced an effective black-box technique based on SVM. Our fusion technique cancels the need for sophisticated or heuristic methods to combine the outputs of multiple association algorithms.

The system was tested using real-world surveillance videos from a network of six cameras with nearly 100 trajectories. To the best of our knowledge, this is a much larger dataset than most datasets used in previous related work. Still, our system manages to yield a remarkable accuracy of close to 90%.

Acknowledgements. We would like to thank Dr. Shmuel Kiro for his fruitful technical discussions and suggestions and Oren Golan of Verint Systems for his consistent administrative support. Also, the support of the VULCAN Consortium under the Israeli Ministry of Industry, Trade and Labor is gratefully acknowledged.

References

1. Avidan, S.: Ensemble tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 261–271 (2007)
2. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
3. Changjiang, Y., Duraiswami, R., Davis, L.: Fast Multiple Object Tracking via a Hierarchical Particle Filter. In: Proceedings of the Tenth IEEE International Conference on Computer Vision, pp. 212–219 (2005)
4. Chilgunde, A., Kumar, P., Ranganath, S., Weimin, H.: Multi-Camera Target Tracking in Blind Regions of Cameras with Non-Overlapping Fields of View. In: Proceedings of the British Machine Vision Conference, pp. 397–406 (2004)
5. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning, 273–297 (1995)

6. D'Angelo, A., Dugelay, J.: People Re-idnetification in Camera Networks Based on Probabilistic Color Histograms. In: Proceedings of the SPIE Conference on Visual Information Processing and Communication, vol. 7882 (2011)
7. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 886–893 (2005)
8. Gilbert, A., Bowden, R.: Tracking Objects Across Cameras by Incrementally Learning Inter-camera Colour Calibration and Patterns of Activity. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 125–136. Springer, Heidelberg (2006)
9. Javed, O., Rasheed, Z., Shafique, K., Shah, M.: Tracking across Multiple Cameras with Disjoint Views. In: Proceedings of the Ninth IEEE International Conference on Computer Vision, pp. 952–957 (2003)
10. Javed, O., Shafique, K., Rasheed, Z., Shah, M.: Modeling Inter-camera Space-time and Appearance Relationships for Tracking across Non-overlapping Views. Computer Vision and Image Understanding, 146–162 (2008)
11. Kang, J., Cohen, I., Medioni, G.: Continuous Tracking within and across Camera Streams. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 267–272 (2003)
12. Kuo, C.-H., Huang, C., Nevatia, R.: Inter-camera Association of Multi-target Tracks by On-Line Learned Appearance Affinity Models. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 383–396. Springer, Heidelberg (2010)
13. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 91–110 (2004)
14. Makris, D., Ellis, T.J., Black, J.: Bridging the Gaps Between Cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 205–210 (2004)
15. Mazzeo, P.L., Spagnolo, P., D'Orazio, T.: Object Tracking by Non-overlapping Distributed Camera Network. In: Blanc-Talon, J., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2009. LNCS, vol. 5807, pp. 516–527. Springer, Heidelberg (2009)
16. Mikolajczyk, K., Schmid, C.: An Affine Invariant Interest Point Detector. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part I. LNCS, vol. 2350, pp. 128–142. Springer, Heidelberg (2002)
17. Montcalm, T., Boufama, B.: Object Inter-camera Tracking with Non-overlapping Views: A New Dynamic Approach. In: Proceedings of the Canadian Conference on Computer and Robot Vision, pp. 354–361 (2010)
18. Porikli, F., Tuzel, O., Meer, P.: Covariance Tracking Using Model Update Based on Means on Riemannian Manifolds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 728–735 (2006)
19. Zheng, W., Gong, S., Xiang, T.: Person Re-identification by Probabilistic Relative Distance Comparison. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 649–656 (2011)