Path Recovery of a Disappearing Target in a Large Network of Cameras

Amir Lev-Tov
Efi Arazi School of Computer Science
The Interdisciplinary Center
Herzliya 46150, Israel
amir.levtov@gmail.com

Yael Moses
Efi Arazi School of Computer Science
The Interdisciplinary Center
Herzliya 46150, Israel
Yael@idc.ac.il

ABSTRACT

A large network of cameras is necessary for covering large areas in surveillance applications. In such systems, gaps between the fields of view of different cameras are often unavoidable. We present a method for path recovery of a single target in such a network of cameras. The solution is robust, efficient, and scalable with the network size. It is probably the first that can cope with hundreds of cameras and thousands of objects. The spatio-temporal topology of the network is assumed to be given. In addition, an algorithm for computing features that can be used to match the appearance of the object at different time steps is assumed to be available. Due to low video quality and limitations of the computed features, possible confusion between the target and other objects can occur. The suggested method overcomes this challenge using a new modified particle filtering framework that produces at each time step a small set of candidate solutions represented by states. Each state consists of an object location and identity. Since invisible locations are explicitly modeled by states, the detection of disappearing and reappearing targets is inherent in the algorithm. A second phase recovers the path using a dynamic programing algorithm on a layered graph that consists of the computed candidate states. A synthetic system with hundreds of cameras and thousands of moving objects is generated and used to demonstrate the efficiency and robustness of the method. The results depend, as expected, on the network topologies and the confusion level between objects. For challenging cases our method obtained good results.

1. INTRODUCTION

A large network of cameras is necessary for covering a large area in surveillance applications. In such systems, gaps between the fields of view (FOVs) of different cameras are often unavoidable. We address the problem of path recovery of a single target in a large scale network of cameras with relatively large gaps between their FOVs. We consider a scene where objects similar to the target are visible and can be confused with it. The major challenges addressed are efficiency and scalability. Our solution allows online handover tracking between cameras or offline processing of a large number of videos. For the latter, our solution requires processing only a small amount of video data.

1.1 Previous Work

Tracking objects in a network of cameras can be regarded as an extension of single camera tracking when there are overlapping regions in their FOVs. These regions are used for tracking and handover (e.g., [15, 17, 8]). In such cases, stitching techniques can also be applied to obtain a single FOV (e.g., [21]). Alternatively, objects can be tracked on a common ground plane, when such a plane is available (e.g., [13, 3, 4, 18, 14]). Such settings can also be regarded as an extension of single camera tracking. However, the cameras in a large network typically have gaps between their FOVs, and a common ground plane is often not available.

Methods that address the problem of tracking in a network of cameras with non-overlapping FOVs often make use of the spatial or temporal topology of the network (e.g., [19, 23, 22, 2, 12]). Several studies (e.g., [11, 23, 19]) regard the handover problem as path recovery in an *observation graph* where each node corresponds to the identity representation of an object observed by one of the cameras at a given time (e.g., a color histogram). The edges and their weights represent the likelihood that two nodes are successive observations of the same object. The likelihood is defined, for example, by the similarity of two observations and the spatial and temporal information about the camera topology. A multi-path recovery is then solved by optimizing over the paths' likelihood in this graph. These methods differ in the chosen likelihood computation as well as the optimization methods used. Nevertheless, none of these methods is scalable. This is due to the exhaustive search on the entire observation graph inherent in all of them.

1.2 Proposed Method

We suggest an efficient and scalable method for recovering the path of a target in a large network of cameras with gaps between their FOVs. As in previous studies, a variation of an observation graph is considered, and the likelihood between nodes is based on network topology and similarity between identities. In contrast to previous methods, our method is scalable and directly handles gaps. The spatial and temporal topology of the network is assumed to be given (e.g., by [16, 5, 20]). This topology is used not only for computing this likelihood, but also for optimizing the search. The construction of the graph is based on a local tracking algorithm that can produce, for each observed object, a feature vector and on a similarity function between a pair of feature vectors. Such an algorithm is assumed to be available (e.g., [24]) and considered as a black box; the better it is, the less confusion is expected between objects and the easier the task is. A new measure for evaluating the possible confusion between objects is suggested and used for measuring the task's difficulty.

The suggested method consists of two phases, and can be easily implemented by a distributed system. The first is a novel probabilistic online multi-hypothesis tracker that can be regarded as a modified particle filter (PF) (A modified PF framework rather than a Kalman filter is chosen due to the expected noise in our observation model, and the non-linearity of the process). The new state space in the modified PF allows to directly model hidden locations. Because tracking is performed in this new state space, the detection of disappearing and reappearing objects is inherent in the algorithm. This is in contrast to classical PF algorithms that overcome short time occlusions only through system robustness to errors and noise (e.g., [9, 13]). This phase significantly reduces the size of the observations graph and defines the weight of each node in it. For each time step, it produces a small set of candidate states, used as an input graph layer for the second phase. The second phase is a shortest path algorithm applied to the graph defined by the first phase.

1.3 Data Simulation and Testing

The suggested method is a theoretical approach for allowing scalability of tracking in a large network of cameras. Even so, we performed a set of experiments in order to test it. Testing its performance on a real large-scale system would clearly involve many practical issues that are orthogonal to the one we study here. This could very well obscure the evaluation of the specific concerns addressed in this work. Therefore, we generate a framework for producing large scale data. This allows us to define the spatio-temporal topology of the network, including gap distribution, as well as the confusion level between a target and other objects moving in the scene. The simulated data allow us to take one step forward and better understand the challenges of large systems without being constrained by the limitations of existing local tracking solutions. This is in contrast to many existing studies for path recovery, which focus much attention on local tracker results. Indeed, these algorithms were tested on only a few cameras with relatively few objects in motion (e.g., [2] with 3 cameras, and [19] with 20 cameras and 10 people). To the best of our knowledge, none of these studies has demonstrated scalability.

A collection of data sets with hundreds of cameras and thousands of objects were used to test our method. The data sets have different confusion levels and spatio-temporal topologies. Our results demonstrate the efficiency of our probabilistic algorithm and its scalability. In addition, they show the advantages of directly modeling the gaps between cameras. Furthermore, the necessity for a second phase is demonstrated by showing that the probabilistic algorithm alone is insufficient, and the second phase is required for a more reliable solution.

1.4 Main Contribution

Our method is the first to be scalable to hundreds and even thousands of cameras, while existing methods have been applied only to a small number of cameras and objects moving in the scene. The new modified particle filtering allows tracking in a state space with no order relation. This enables the direct modeling of gaps, where the detection of appearing and disappearing targets is inherent in the algorithm. The combination of two phases achieves efficiency using the probabilistic phase and robustness using the deterministic phase. Finally, the proposed measure of confusion between a target and other objects can be used for estimating the difficulty of tracking as well as evaluating local tracking methods.

The rest of the paper is organized as follows: In Sec. 2 we review the PF fundamentals. We define our formal terminology in Sec. 3 and our method in Sec. 4. A discussion on complexity issues is presented in Sec. 5. The confusion measure is defined in Sec. 6 and experimental results are presented in Sec. 7. Finally, we conclude in Sec. 8.

2. PARTICLE FILTERS

Here we briefly review the PF framework for propagating state density over time ([7, 1]). The state density, X_t, is represented by a set of particles $\{x_t\}$ with weights $\{w(x_t)\}$. Each particle is in fact a hypothesis of the system's state. PF efficiently approximates a Bayesian filter and can model general distributions, particularly of non-linear processes with non-Gaussian noise.

The particles' distribution approximates the state density $P(X_t|Z_t)$, where X_t represents the current state (e.g., the current location of a tracked object), and Z_t the current observation. The computation is performed using the prior $P(X_{t-1}|Z_{t-1})$ from the previous time step followed by a prediction process that yields a predicted prior for the current time step. The particles representing this prior are weighted by the current observation density, $P(Z_t|X_t)$, and the new distribution becomes the approximation for the new prior $P(X_t|Z_t)$. Ignoring the constant observation prior $P(Z_t)$, the computation follows formally from Bayes' rule:

$$\begin{aligned} P(X_t|Z_t) &\propto P(Z_t|X_t)P(X_t|Z_{t-1}) = \\ &P(Z_t|X_t)\int P(X_t|X_{t-1})P(X_{t-1}|Z_{t-1})dX_{t-1}. \end{aligned} \tag{1}$$

All computations are given the previous states $X_{1..t-1}$ and $Z_{1..t-1}$, but depend only on the last state, using Markovian assumptions. The larger the number of particles, the better approximation we get for the Bayes filter. This iterative process requires a prior at time $t = 0$, $P(X_0)$, which is the initialization.

Our method reimplements the steps used in the PF to operate in a new state space. This state space lacks an additive operation and an order relation. Thus, a moment cannot be calculated from the resulting density. Instead, the maximum likelihood hypotheses are chosen. The diffusion and prediction steps are not implemented at each time step by an additive operation as in the classic PF, and our resampling procedure distinguishes between two sets of particles and resamples from one of them. Our weighting step evaluates according to the observations only part of the particles, but the normalization considers all of them.

3. PROBLEM FORMULATION

3.1 Network Topology

A weighted directed graph, the *network graph*, represents the topology of our camera network $G_{mc} = \langle C, E, W^p \rangle$. The nodes $C = \{C_i\}_{i=1}^n$ represent cameras, and the edges, E, represent *adjacency* of cameras (direct transitions between cameras, without passing through other cameras' FOVs). Self edges represent that a target might remain in the same camera's FOV. The edge weight $W^p(C_i, C_j)$ is taken to be the probability of transition between the cameras C_i and C_j. It captures the *a priori* knowledge of the system regarding the behavior of targets in choosing their path. For each node C_i, the sum of outgoing edge weights is normalized to be 1. We assume that the transition of a target from one camera to another is independent of its previous location. Therefore, G_{mc} can be regarded as a *Markov chain* representing the object locations over the tracked path. Nevertheless, further knowledge about the environment's dynamics, such as higher order transition model, can be used or calculated (e.g., [6]), and thus, modeling better the targets' expected paths and the overall accuracy of tracking.

In our system we assume that the cameras might have large gaps between their FOVs and the time required for a target to move between different cameras can vary. We assume that a set of time delay probability density functions (*pdf*s) between all pairs of adjacent cameras is given. Such information can be obtained from other studies such as [5]. A *pdf* between adjacent cameras C_i and C_j is denoted by $T_{i,j} : \mathbb{R}^+ \to [0, 1]$. In our algorithm we consider time as a sequence of discrete segments. Together, $T_{i,j}$ and W^p form the dynamic model of the system.

3.2 Observations

Most existing single camera tracking algorithms are based on computing a set of features for each of the tracked objects. The appearance of objects at different time steps are then matched by a function that compares these features. The features may include color histogram, gait, velocity, height, gender, and so forth. We assume that such features and their matching function are available to our method by some existing algorithm. In addition, without loss of generality we assume that the features representing the observed object can be summarized by a feature vector, and denoted by $\vec{f} \in \mathbb{R}^d$. Denote by F_t the set of all feature vectors observed by the system at a given time t, and by $F = \bigcup_{t=1}^{T} F_t$ the set of feature vectors in overall time of tracking.

For comparing feature vectors, f_1 and f_2, the *Mahalanobis distance* is used in our experiments, assuming that a covariance matrix, Σ, of a typical feature vector is known:

$$d_M(f_1, f_2) = \sqrt{(f_1 - f_2)\Sigma^{-1}(f_1 - f_2)^T}. \tag{2}$$

The probability P_s of f_1 and f_2 to represent the same object, is then defined by:

$$P_s(f_1, f_2) = e^{-\frac{1}{2}d_M^2(f_1, f_2)}. \tag{3}$$

3.3 Object State

The location and appearance of an object in the system is modeled by a state, $s \in S = \{(\ell, f) : \ell \in L, f \in F\}$, where $f \in F$ is the feature vector, and $\ell \in L = C \cup H$ is the location of the object. For a visible object $\ell \in C$, and for a hidden one, $\ell \in H$. Each hidden location contains the object's destination camera C_i, as well as its expected time of arrival, ta. The range of ta is determined by the time delay functions $T_{i,j}$, and it takes discrete values. Let Δ_i be the maximal time of arrival to a given camera, C_i. Then H is defined by:

$$H = \{(C_i, ta) \mid C_i \in C, 1 \le ta \le \Delta_i)\}. \tag{4}$$

The feature vector may vary over time. When the object is not visible, we consider the last visible feature vector. The sets of particle states and observation states at time t are denoted by $\{z_t\}$ and $\{x_t\}$, respectively.

4. THE METHOD

Given the initial state of the target, $x_0 = (C_0, f_0)$, the task is to recover the sequence of states that best represents the path of the target in the network over time. The initial state can be determined manually or automatically, depending on the application. Our method recovers, at each time step, the target's location (visible or hidden), and its feature vector representation along the path. The first phase of the method is a probabilistic online algorithm that selects a set of the k most probable states at each time step (k-best). Using the k-best states, the second phase recovers the full path of the target using a deterministic algorithm. Note that the second phase requires only the last step of the first phase. Hence, our algorithm is online.

4.1 First Phase: Particle Filter

A set of N state hypotheses (so called particles) are spread over the state space S, modeling our belief about the current target's state (location and feature vector). As in the classic PF algorithm, propagating this belief in S, using a dynamic model and an observation model, provides certainty information about the target's state at each time step. Given the initial state, x_0, the particles in the set are initialized to reside at the target's origin camera C_0, where each particle possesses the target feature vector f_0, and is equally weighted.

4.1.1 Prediction

At each time step a prediction is applied to the set of particles $\{x_t\}$. The prediction models the expected state of a particle in the next step, $x_t \sim P(X_t|X_{t-1})$. The new location is determined by the dynamic model of moving targets in the world, given by the *network graph*, G_{mc}, and the time delay probability density functions $\{T_{i,j}\}$. Here we use G_{mc} to perform a random walk of a single step from the corresponding camera state. The time of arrival, ta, is sampled from the respective $T_{i,j}$. No gap is indicated by $ta = 1$, whereas $ta > 1$ implies a hidden state. The prediction of a hidden particle is performed by only decreasing ta by one. If $ta = 1$, it enters a non-hidden state, according to the destination camera.

In our implementation, the predicted feature vector remains unchanged. In future work it can be modified to use additional knowledge about the network, such as the photometric transformations between cameras (e.g., [10]).

4.1.2 Observation

The observation is a set of system states $Z_t \subseteq C \times F_t$. A system state, $z_t = (\ell, f) \in Z_t$ consists of the feature vector, f, produced by the local tracker at camera ℓ. Clearly, no observation can produce a hidden state. Hence, only $\ell \in C$ locations are considered.

4.1.3 Evaluation & Update

In PF terms, the observation model is defined as $P(Z_t|X_t = x_t) = w(x_t)$, where $w(x_t)$ is the weight of the particle. For assigning a weight to a visible particle, $x_t = (C_x, f_x)$, we first define the probability of an observation $z_t = (C_z, f_z)$ to support that particle. An observation can support a particle only if it is observed by the particle's camera, that is, $C_x = C_z$. In this case, the support depends on the probability of the two feature vectors to represent the same object, and on the transition probability from the particle's camera at the previous time step, $C_{x_{t-1}}$, to the current camera, C_{x_t} (defined by G_{mc}). Formally, we obtain:

$$P_h(x_t = z_t) = \alpha W^p(C_{x_{t-1}}, C_{x_t}) + (1 - \alpha)P_s(f_{x_t}, f_{z_t}). \tag{5}$$

In addition, f_z is also required to be similar to f_0 (the target's feature vector in the initial state). Putting it all together, we obtain:

$$P(x_t = z_t|f_0) = \begin{cases} 0 & C_{x_t} \neq C_{z_t} \\ \begin{array}{l}\beta P_h(x_t = z_t) + \\ (1 - \beta)P_s(f_0, f_{z_t})\end{array} & C_{x_t} = C_{z_t} \end{cases} \tag{6}$$

where $0 < \alpha, \beta < 1$. In our implementation and experiments we used $\alpha, \beta = 0.5$.

The weight of a visible particle is defined by the maximum probability over the support of all observations:

$$w(x_t) = \max_{z_t \in Z_t} P(x_t = z_t|f_0). \tag{7}$$

The associated observation of the particle x_t is the observation for which this maximum is obtained, z_t^*.

A particle x_t for which $w(x_t) > 0$ is updated with its associated observation, z_t^*, with probability P_{update}. In our experiments we used $P_{update} = 0.5$. As a result, our current knowledge is updated, and hypotheses that are not real observations are also maintained. This allows us to also model an occlusion within a camera's FOV.

Since the feature vectors are compared only between particles and observations in the same camera, it is sufficient to perform tracking only in the particles' set of cameras. For further improving efficiency, a maximum of N_{fc} most frequent cameras from this set are considered. Consequently, the complexity of our algorithm is significantly reduced, since both tracking and vector comparisons can be computationally expensive.

4.1.4 *Normalization*

The weight of a visible particle is determined by the support of observations (Eq. 7). The initial weight of a hidden particle is taken to be its weight when it entered the hidden state. Hidden particles cannot be supported directly by observation. However, the weight of visible particles may reflect the weight of hidden ones: lack of support for visible particles increases the support for hidden ones and vice versa. The weights of all particles are normalized to sum up to 1 and denoted by $\hat{w}(x_t)$. By normalizing the weights of all particles, the weights of the hidden particles are correctly affected.

4.1.5 *Output*

Our current distribution of particles models our belief $P(X_t|Z_t)$ about the tracked state. In the classic PF, the set of particles represents the *pdf* in a continuous space. Therefore, the location at a given time is chosen by, for example, the expectation. In our case, no order relation between the states exists nor is any additive operation defined, and thus no moment can be used. We therefore use the normalized weight of the particles to rank the k distinct most probable particles, $\{x_t^i\}_{i=1}^k$. For a hidden particle, the i^{th} output is the i^{th} particle itself, x_t^i. For a visible particle, the associated observation, z_t^{*i} is chosen (the one that maximizes Eq. 7). This set of k states is the input to the second phase of our algorithm, as described in Sec. 4.2.

4.1.6 *Resampling and Diffusion*

A resampling procedure is applied to the set of particles, generating a new subset of hypotheses. This is done by a factored sampling as in a classic PF, according to the particles' weights. Similar random sampling is used also in the diffusion and prediction steps. This step causes particles with high weights to be duplicated, thus increasing the probability to correctly detect the next state solution in their region. On the other hand, particles with low weights are more likely to be discarded.

An outcome of the resampling step is a generation of duplicated states. In the classic PF, duplications are avoided by adding noise. However, our state space lacks an additive operation. Hence, a diffusion procedure is redefined. To diffuse the location component of a state, a *Markov chain* denoted by $G_{mcd} = \langle V, E, W^d \rangle$ is used. The G_{mcd} is similar to G_{mc} but has higher probabilities for staying in the same camera state. Thus, most of the particles do not change their location component. In our experiments, the reflexive transition probabilities of G_{mcd} are set to be twice that of G_{mc}'s. The feature vectors may randomly change to one of the vectors produced from observed objects in the camera, according to their similarities to f_0. Whether a change takes place is randomly chosen with probability P_{switch} (in our experiments, $P_{switch} = 0.5$).

At this point a new iteration of the algorithm begins, in which a prediction step is applied on the new generation of particles.

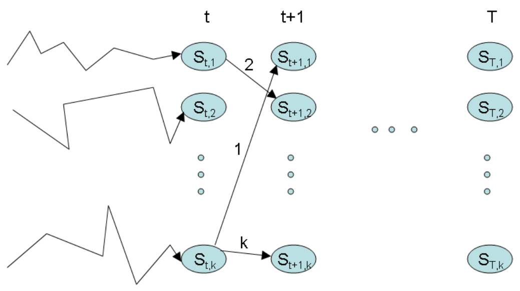


Figure 1: The layered graph of the second phase. Each node at layer t represents one of the k most likely states computed at time t. Edges in the graph represent legal transitions between the states. The weight of an edge is the priority of the state it is directed to. The trajectories on the left are the shortest paths to the corresponding states until time step t.

4.2 Second Phase: Path Reconstruction

The outcome of the first phase is a set of k most likely states prioritized from 1 to k, for each time step. The correct state is not necessarily the most probable state. In particular, the most probable states do not necessarily represent a legal path according to the given network topology. The goal of the second phase is to choose the optimal path given the sets of k-best states obtained for each time step. The desired optimal path consists of the successive highest priority states with legal transitions between them, defined by the network topology. The optimization is therefore performed on the sum of the states' priorities.

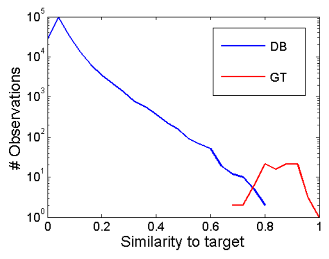
To compute the optimal path, a layered graph is constructed where the nodes in layer t consist of the k-best states computed at time t. An edge between two nodes reflects a legal transition from the corresponding states, defined by the spatial and temporal network topology. In particular, it depends on the adjacency of its cameras' nodes and the expected time delay, in case of hidden locations. The weight of an edge is the priority of the state it is directed to, based on the results of the first phase. (Fig. 1 is a sketch of such a graph.) Using this formulation, the optimal path is computed using an online shortest path algorithm in a layered graph. Note that two layers might be disconnected due to errors. In this case, the highest prioritized node from the next level is chosen. This choice allows poor results to be avoided in case of disconnection, while maintaining the benefits of the reconstruction as a method.

The second phase operates only on the last data gathered from the PF and has the advantage of being online. Effectively, we get an efficient multi-hypotheses path tracker due to dynamic programming.

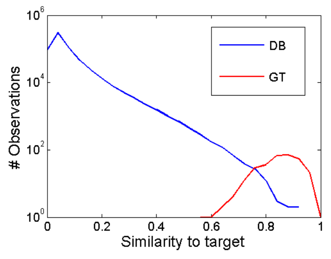
5. COMPLEXITY & EFFICIENCY

One of the main goals of the proposed method is efficiency. An efficient method allows the recovery of a path in a large network of cameras. The efficiency of our method depends on the predefined number of particles, N. A larger N allows better handling of a difficult scene, defined by the network topology, its average connectivity degree, the number of cameras, the size of the gaps, and the number of observations.

At each time step, the number of cameras the local tracking algorithm is performed on is bounded by a predefined constant. The number of vector comparisons is linear in the number of time steps,

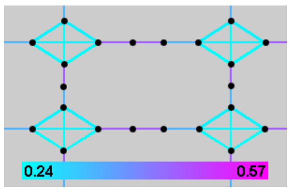


(a)

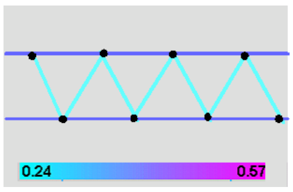


(b)

Figure 2: Similarity distributions of GT and DB (log scaled). (a) High separation results in a low confusion measure of $CM = 0.17$. (b) Lower separation results in a higher confusion measure of $CM = 0.63$.



(a)



(b)

Figure 3: Partial view of the camera networks for the (a) grid and (b) street topologies. The black dots are cameras and edge color represents a probability of transition. The graph shows the probability of moving in one of the directions and self edges are not drawn.

assuming a constant average number of observations at each camera and a constant number of particles. In the worst case, the number of vector comparisons is given by $O(\bar{Z}NT)$, where $\bar{Z}$ is the average number of observations at each camera, and T is the overall number of time steps. The second phase of the algorithm requires only $O(k^2T)$ basic steps, which are cheap integer summations.

For comparison, we next present the complexity of a full search, deterministic method. Let us first consider a simple topology without gaps. A state in this case contains the camera and the feature vector of each observed object. Note that after enough steps, all states must be considered, given that the topology is non-trivial (the camera's out-degree is larger than one). In particular, the tracking must be performed in all cameras, in contrast to the fixed number of cameras considered in our algorithm. Consider a layered graph, where each layer consists of Z nodes that represent the observations from the corresponding time step. The weights of the edges in this graph can be taken to be the similarity between two states (e.g., Eq. 3). The optimal path can be computed using a dynamic programing algorithm similar to the path computed in the second phase of our algorithm. The number of edges here is $Z\bar{Z}dT$, where d is the average degree of a node ($\bar{Z}d$ is the number of neighboring observations for each observation). All edge weights must be considered; hence, the total number of vector comparisons here is $O(Z\bar{Z}dT)$, whereas in our algorithm it is $O(\bar{Z}NT)$. Since $N << Zd$, our method is significantly more efficient than the full search one. Note that in the presence of gaps, the number of edges in the full graph increases, since states may be connected not only to the next layer but also to any other layer within the allowed gap size. In this case, the efficiency of our algorithm further improves over a full search. It should be noted that the complexity of a Bayes filter on the full graph is also $O(Z^2T)$ (or $O(Z\bar{Z}dT)$ with improvements), which is again less efficient than our method.

5.1 Distributed Implementation

The scalability of our algorithm can be further improved by a distributed implementation, assuming each camera has computation power. A camera holds a set of visible particles, and a set of hidden particles which are enrouted to that camera. The local tracking algorithm is performed by each camera and produces the feature vector in its FOV. The particles are locally evaluated according to the observed feature vectors, and the resulting weights are sent to a coordinator (which can be a central server or a leader from a group of cameras). The coordinator normalizes the set of particles, computes the output states and the reconstructed path, and resamples a new set of particles according to the new normalized weights. The new particles are then sent to the corresponding cameras, and diffusion and prediction processes are then applied on them locally. If the diffusion and prediction processes show that the particle should be moved, it is sent to the new camera. After the prediction step ends, tracking of the next time step begins. Each camera that contains particles performs, at each time step, a constant number of vector comparisons. This number depends on $\bar{Z}$ and the average number of particles in each camera. Here, no tracking or vector comparison is performed by the coordinator. Note that the centralized algorithm performs $O(\bar{Z}N)$ vector comparisons at each time step.

6. CONFUSION MEASURE

In this section we present a measure for evaluating the chances of confusion between the target and other objects in the scene. Let F be the set of all observations, f_0 be the target at time $t = 0$, and GT be the set of observations of f_0 over time. Consider a similarity measure s between two feature vectors, where $s : F \times F \to [0, 1]$ (P_s defined in Eq. 3 is an example of such an s). An observation f is a candidate for misclassification as an observation of f_0, if $\exists f' \in GT$ such that $s(f, f_0) \geq s(f, f')$. Using the measure s, we formally define the sample space Ω as the set of observations that are candidates for misclassification of f_0 as:

$$\Omega = \{f \in F : s(f, f_0) \geq \min_{f' \in GT} s(f', f_0)\}. \tag{8}$$

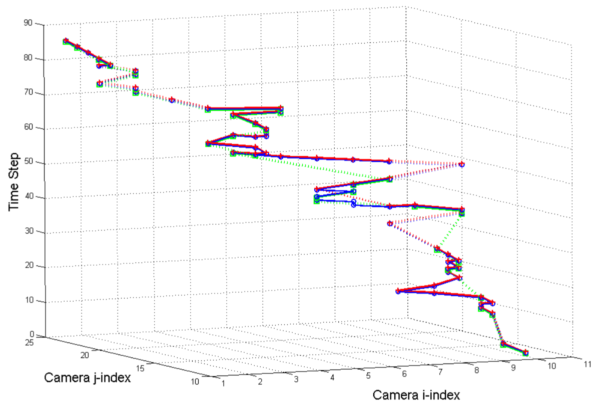
Then, the probability of incorrect classification is:

$$P(Error|f_0) = \frac{|GT^c|}{|\Omega|}. \tag{9}$$

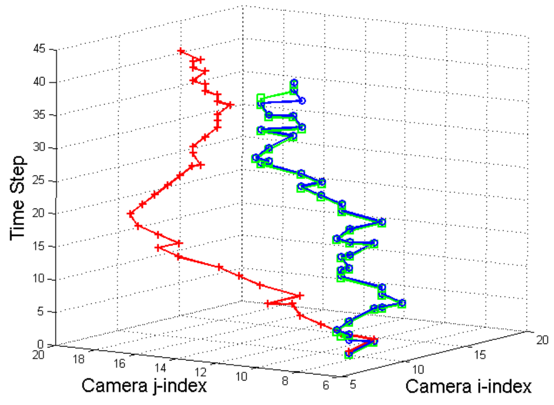
Considering Ω as a non-symmetric sample space, we define $|A| = \sum_{w \in A} s(w, f_0)$, where $A \subseteq \Omega$. Thus, a set whose elements have high similarity values will get a high measure. Given the distribution of element similarities, this can be computed by

$$CM(DB, f_0) = \frac{\int_J f_{DB}(s)s\, ds}{\int_J (f_{GT}(s) + f_{DB}(s))s\, ds}, \tag{10}$$

where f_{GT} is the histogram function of similarities between f_0 and GT, and f_{DB} is the histogram function of similarities between f_0 and $\Omega \setminus GT$. J is the interval $[a, 1]$, where $a = \min\{x \in [0, 1] : f_{GT}(x) > 0\}$. Fig. 2 presents an example of such histograms. Consider the overlapping regions under the blue and the red curves (DB and target representations respectively). These regions represents the possibilities of choosing a non-target and a target, respectively. The higher the region under the blue curve in the overlapping domain, with respect to the region under the red curve, the higher the confusion measure will be. It should be noted that this measure is an average estimation of the confusion complexity of a set of observations, given a target to follow; that is, it is an average



(a)



(b)

Figure 4: Example of tracking results on the Manhattan grid topology, (a) with gaps between cameras' FOVs, and (b) without. The XY surface is the grid plane and the Z axis is the time step. The ground truth path is marked in blue. (a) The results of the best state from the first phase is in green, and the result of the second phase is in red. Gaps between cameras' FOVs are indicated by a dotted line. (b) Comparison to the naive algorithm. Our algorithm is in green and the naive result is in red.

estimation of the difficulty of tracking. It does not take into consideration the dynamics of the scene, i.e., a target might follow a path that contains more similar observations than the average and thus will be more difficult to track.

7. EXPERIMENTS

The power and advantages of our algorithm are likely to be evident only on large networks of cameras. In addition, our method is not based on a specific tracking algorithm. We therefore tested our method by extensive experiments on simulated data.

7.1 Generating the Simulated Data

We implemented our algorithm in Matlab as a simulator of real environments. An environment provides camera network topology (spatial and temporal), and an observations database (ODB). An ODB includes the path of M moving objects and their feature vectors changing over time. Any input from real data tracking results can be used to modify the parameters and noise used by our simulation.

7.1.1 The Topology

Connections for the camera network are generated according to the desired topology. In our experiments we tested two different topological structures, a street and a Manhattan grid. The probabilities G_{mc} are generated for each connection as well as on reflexive edges. In order to avoid a completely uniform transition prior, a small weight is given for going in a certain direction. Fig. 3 presents the transition probabilities of the Manhattan grid and the street topologies. The time delay *pdf*s are generated as normal distributions $T_{i,j}$. In our experiments we define them by $T_{i,j} \sim N(\mu_{i,j}, 0.2\mu_{i,j})$. The mean μ is randomly chosen according to a distribution of mean gaps determined as a scene parameter. We used a distribution with 20% gaps, 50% of them having a mean of 7 time steps and the others having a mean of 3 time steps.

7.1.2 The ODB

The initial observed feature f_i and the origin camera C_i are randomly chosen for each of the M objects. To produce the scene dynamics, M paths over the network are randomly chosen. To do so, a location state is propagated, using the prediction step described in Sec. 4.1 and $T_{i,j}, G_{mc}$, which were already generated. Repeating this process for each object T times provides us with the population dynamics of our scene. One of those objects is chosen to be the target.

To generate the feature vector of each object, we use a multivariate normal distribution of an object feature vector. It is given by a covariance matrix Σ of a typical feature vector in the world, and an object mean of f_i. The feature vectors are changed over time according to Σ and to their first instance at time $t = 0$, which we assume to be the mean, for simplicity. Formally, at each time step t, the new feature vector of an object, f_{z_t}, is changed and defined as a linear combination of $\tilde{f}_{z_{t-1}}$ and f_{z_0},

$$f_{z_t} = \gamma f_{z_0} + (1-\gamma)\tilde{f}_{z_{t-1}}, \tag{11}$$

where $0 < \gamma < 1$ and $\tilde{f}_{z_{t-1}} \sim N(f_{z_{t-1}}, \Sigma)$. By this, the expectation of f_{z_t} is still f_{z_0} but is not completely independent of its instance in the previous time step. In our experiments we used $\gamma = 0.8$.

7.2 Results

The result of our algorithm is a sequence of states. The score of the algorithm is defined by the percentage of correct states along the sequence. Note that when the target is invisible at a given time step, any of the hidden states are considered correct. The first phase of our method results in a sequence of sets of states, each consisting of k-best ranked states. The k-best score is defined to be the percentage of sets along the sequence that contain the respective correct state.

Because our algorithm is probabilistic, it may give different scores for different runs. In each of the results presented below, several tests were executed and the shown scores are the averages of their results.

7.2.1 Grid Topology Experiments

In order to simulate a camera network installed in a city, we generated a Manhattan grid topology. The cameras are connected linearly along a street path with a clique of 4 nodes at the intersec-

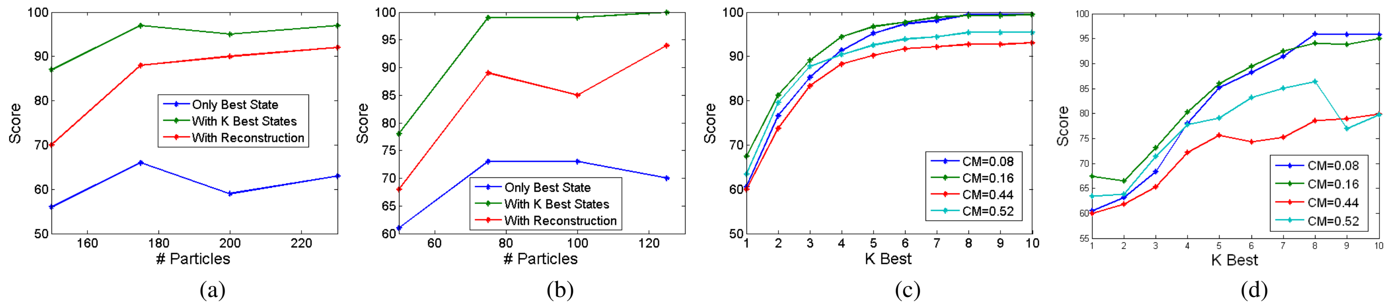


Figure 5: Score as a function of the number of particles for the (a) grid topology and (b) street topology experiments. For the grid topology, k-best score as a function of k for the first phase (c) and the score as a function of k for the second phase (d).

tions, as shown in Fig. 3a. The grid consists of 35×35 cameras with blocks of 5×4 cameras, resulting in $C = 502$ cameras with average degree of 3.7. Disappearance in a gap occurred in 27% of the time steps, and there are 2009 observations with a confusion measure of $CM = 0.17$ (Fig. 2.a).

In the first experiment we used $N = 250$ particles. The resulting scores for the first phase were 63% and 99%, for $k = 1$ and $k = 10$, respectively. The final score obtained, after applying the second phase, was 95%. The scores are averaged over 5 tests with $STD = 5$ in the second phase. We set $N_{fc} = 30$, but the average number of cameras visited at each time step was only 21. Fig. 4(a) presents the true path and the recovered one after the first and the second phases of the algorithm.

Next we tested how the number of particles affected the results, using the same data set. As the number of particles increases, so does the accuracy, as depicted in Fig. 5a, showing a tradeoff between accuracy and efficiency. The graph shows the correctness of the first phase, which yielded a correct estimate among its k-best states, and poor results for using only the best hypothesis of the PF. Moreover, it shows the efficiency of the second phase in using the k-best potential solution to correct the first phase result. The results are averaged over 6 tests.

Next we tested the effect of different confusion measure values on the results as well as the effect of the size of k used in the first phase. To do so, we used the same topology, and the same paths of objects but with different ODBs. The results of the first phase, consisting of the potential score among the k-best, are presented in Fig. 5c, and after the second phase in Fig. 5d. The results demonstrate that, in general, the smaller the confusion measure, the better the results will be. In addition, as expected, the performance of the algorithm improves as a function of k. In our experiments we achieved an optimum value already at about $k = 7$.

To test the efficiency of modeling gaps in our method, we ran our algorithm on the same scene, once with gap modeling and once without it. This experiment was run 5 times, with $N = 200$ and $k = 7$. The resulting scores were 95% with gap modeling, and 72% without it. When we evaluate only the visible locations, then the score of gap modeling decreases to 80% and 70% without it. We conclude that the gap modeling in our algorithm enables us to identify disappearance as well as to improve the tracking results when the target is visible.

7.2.2 Street Topology Experiments

In order to test a camera network installed on a street, we simulate a topology with 500 cameras on both sides of the street, where one camera's FOV is not large enough to identify objects on the other side. In terms of the graph, it is possible for an object to move from a specific camera's FOV to a nearby camera without entering the other's FOV. In other words, each camera has 4 adjacent cameras, as depicted in Fig. 3b. The data consist of 2001 moving objects with a $CM = 0.63$ (Fig. 2.b). The results as a function of the number of particles are presented in Fig. 5b. The results are presented for different values of N and are averaged over 3 test runs with a small STD. The average number of cameras visited at each time step was 14. The street topology is easier than the grid topology: the average node degree here is 5 and the structure is more linear. Indeed, better results are achieved than in the grid experiments.

7.2.3 Comparison to a Naive Algorithm

We compared our algorithm to a naive implementation in which only a single hypothesis is maintained. The tracking prediction is performed uniformly from the previous location and the observation that is most similar to f_0 is chosen from the predicted locations. Testing the naive algorithm in a model with gaps will trivially fail in the first gap. Hence, we compared the two algorithms in a model without gaps. We used a camera network topology of a Manhattan grid, with 178 cameras and an ODB with $CM = 0.86$. A result example is presented in Fig. 4b. Once the target is out of range of the naive tracker, it loses the target and never recovers.

8. CONCLUSION

We presented a new approach for path recovery of a target moving through the FOVs of a large number of cameras and overcome the main challenges of this task: the scale of the problem (number of possible paths), the gaps between the FOVs, and the possible confusion between the target and other moving objects in the scene. A unique feature of our algorithm is the application of a PF-style technique to a discontinuous space. In our approach, the PF is applied to a graph rather than to the more customary camera plane, ground plane, or map. We believe that such an approach may have value in other applications as well.

The efficiency of our algorithm is due to the modified PF algorithm we developed. The quality of the results are due to explicitly modeling and accounting for gaps between FOVs, and the extraction of a set of candidate solutions, rather than a single one as in the classical algorithm. Using this set, together with higher order information, provides a genuine advantage in reconstructing the correct path.

Existing algorithms for path recovery have been tested only on a small set of cameras with a small number of objects. Moreover, most of them are not scalable. To demonstrate the scalability of our algorithm and its robustness, we tested it on a network of hundreds of cameras and thousands of moving objects. The quality of the obtained results is impressive, despite the algorithm's low computational cost.

Acknowledgments

This research was partially supported by the Israel Science Foundation (grant no. 1339/05).

9. REFERENCES

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[2] Y. Cai, W. Chen, K. Huang, and T. Tan. Continuously tracking objects across multiple widely separated cameras. In *ACCV*, 2007.

[3] A. Chilgunde, P. Kumar, S. Ranganath, and H. WeiMin. Multi-camera target tracking in blind regions of cameras with non-overlapping fields of view. In *BMVC*, 2004.

[4] W. Du and J. Piater. Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In *ACCV*, 2007.

[5] R. Farrell and L. Davis. Decentralized discovery of camera network topology. In *ICDSC*, 2008.

[6] R. Farrell, D. Doermann, and L. S. Davis. Learning higher-order transition models in medium-scale camera networks. *IEEE International Conference on Computer Vision*, 0:1–8, 2007.

[7] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proc. F, Radar and Signal Processing*, 140(2):107–113, 1993.

[8] S. Guler, J. M. Griffith, and I. A. Pushee. Tracking and handoff between multiple perspective camera views. *Applied Image Pattern Recognition Workshop*, 2003.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.

[10] O. Javed, K. Shafique, and M. Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *CVPR*, 2005.

[11] V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In *CVPR*, 1999.

[12] H. Kim, J. Romberg, and W. Wolf. Multi-camera tracking on a graph using markov chain monte carlo. In *ICDSC*, 2009.

[13] K. Kim and L. Davis. Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In *ECCV*, 2006.

[14] W. Leoputra, T. Tan, and F. L. Lim. Non-overlapping distributed tracking using particle filter. In *ICPR*, 2006.

[15] K.-C. Lien and C.-L. Huang. Multi-view-based cooperative tracking of multiple human objects in cluttered scenes. In *ICPR*, 2006.

[16] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *CVPR*, 2004.

[17] W. Qu, D. Schonfeld, and M. Mohamed. Distributed bayesian multiple-target tracking in crowded environments using multiple collaborative cameras. *EURASIP J. Appl. Signal Process*, 2007(1):21–21, 2007.

[18] A. Rahimi, B. Dunagan, and T. Darrell. Tracking people with a sparse network of bearing sensors. In *ECCV*, 2004.

[19] B. Song and A. K. Roy-Chowdhury. Stochastic adaptive tracking in a camera network. In *ICCV*, 2007.

[20] C. Stauffer. Learning to track objects through unobserved regions. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, 2005.

[21] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, 1996.

[22] W. Zajdel, A. T. Cemgil, and B. J. A. Krose. Online multicamera tracking with a switching state-space model. In *ICPR*, 2004.

[23] O. J. Zeeshan, O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking across multiple cameras with disjoint views. In *ICCV*, 2003.

[24] T. Zhao, R. Nevatia, and B. Wu. Segmentation and tracking of multiple humans in crowded environments. *IEEE Trans. Patt. Anal. Mach. Intell.*, 30(7):1198–1211, 2008.