Stochastic Adaptive Tracking In A Camera Network

Bi Song, Amit K. Roy-Chowdhury *
Department of Electrical Engineering
University of California, Riverside
{bsong,amitrc}@ee.ucr.edu

Abstract

We present a novel stochastic, adaptive strategy for tracking multiple people in a large network of video cameras. Similarities between features (appearance and biometrics) observed at different cameras are continuously adapted and the stochastically optimal path for each person computed. The following are the major contributions of the proposed approach. First, we consider situations where the feature similarities are uncertain and treat them as random variables. We show how the distributions of these random variables can be learned and how to compute the tracks in a stochastically optimal manner. Second, we consider the possibility of long-term interdependence of the features over space and time. This allows us to adaptively evolve the feature correspondences by observing the system performance over a time window, and correct for errors in the similarity computations. Third, we show that the above two conditions can be addressed by treating the issue of tracking in a camera network as an optimization problem in a stochastic adaptive system. We show results on data collected by a large camera network. The proposed approach is particularly suitable for distributed processing over the entire network.

1. Introduction

As large networks of video cameras are installed, it is essential to develop automated tools for analyzing the data collected from these cameras and summarizing them in a manner that is meaningful to the end user. One of the most basic tasks in this regard is to be able to track objects across the network. This introduces certain challenges that are unique to this particular application scenario, in addition to the existing challenges in tracking objects like pose and illumination variations, occlusion, clutter and sensor noise.

In this paper, we present a stochastic adaptive framework for tracking in a video network whereby errors in feature correspondences are modeled statistically and adapted in time by considering the long-term dependencies between them. We make three *major contributions.*

• First, we take into account the fact that similarity computations in video are intrinsically erroneous. Thus we model the *similarity* between observations at two camera nodes as random variables and show how to learn their probability distributions. Thereafter, we show how to compute the tracks in the camera network in a stochastically optimal manner. This is achieved using dynamic programming methods for finding optimal paths in graphs with stochastic weights [13]. The idea of treating the similarity between two features as a random variable is based on experience that computing feature similarities in video in uncontrolled environments is often a very difficult problem. This model has similarities with models in econometrics, policy planning and network queueing theory.

• Second, we consider long-term (possibly non-Markovian) interdependencies between the features over space and time and use it to correct wrong correspondences. For example, some of the cameras might be seeing objects in shadow thus making feature correspondence difficult; however, by observing other correspondences either earlier or later in time, many of the mistakes can be corrected. This is achieved by considering the variation of a person's appearance and biometrics (gait is considered) within a possible path over a certain period of time. For this purpose, we derive a path smoothness function (PSF) using discriminant analysis methods.

• The above techniques are integrated within the framework of a stochastic adaptive system [11]. By this we mean a stochastic system some of whose parameters are unknown and need to be learned during the system's operation. Our problem of tracking in a camera network by considering errors in feature correspondence and their dependence in space and time lends itself to this framework. The observed features and the similarity between them are represented by a discrete-time stochastic system. Each similarity score is a random variable with a distribution that is assumed to belong to the family of exponential distributions. The param-

*The authors were supported by NSF grants ECS-0622176 and CNS-0551741, ARO grant W911NF-07-1-0485 and CISCO Inc.

eters of this distribution are adapted based on the values of the PSF which are a function of the system's performance. Thus the overall system works by adapting the similarity scores and finding the best expected track of each person. Figure 1 depicts this diagrammatically.

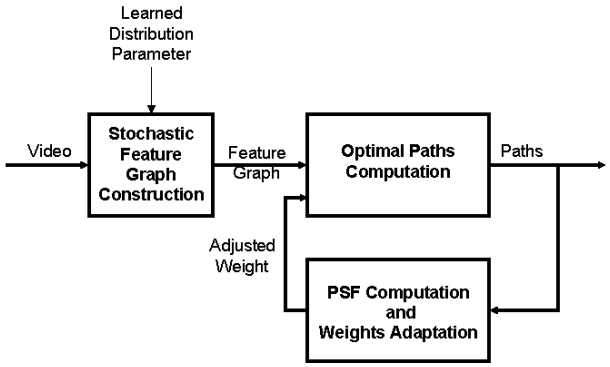

Figure 1. Framework for stochastic adaptive tracking.

We assume that we know a network topology, i.e., connections between cameras and entry/exit points in their view. We shall refer to each entry/exit point as a node. A distribution of the travel time between two nodes is also assumed to be known. A number of recent papers have shown how to obtain them [14, 15, 16, 18] and others [4, 5, 10] have used them for tracking. The similarity computation between features will involve observations at all nodes in a local neighborhood (obtained from the topology). It will be based on appearance features using normalized color [2], identity features using the width vector in gait biometrics [8], and the travel time between two nodes. The stochastic adaptive framework for finding the best tracks will be implemented at certain instants of time by collecting information over a time window up to and including that time instant. We will show that our method is able to track multiple objects over long periods of time in a video network that is spread over a large geographic area.

Our proposed strategy can be implemented in a distributed processing framework. Video collected at each camera will be analyzed locally and only a small amount of information will be transmitted to a central processor, where it will be integrated to obtain the globally optimal solution.

Relation to Previous Work: There have been a few papers in the recent past that deal with networks of video cameras. Particular interest has been focused on learning a network topology [14, 15, 18]. Nodes are defined by entry/exit points or cameras and the goal is to obtain the interconnections between them and the transition times between two nodes. With respect to tracking in a network of cameras, [17] used location and velocity of objects moving across multiple non-overlapping cameras to estimate the calibration parameters of the cameras and the target's trajectory. In [12], the authors used a particle filter to switch between track prediction between non-overlapping cameras and tracking within a camera. In [9], the authors presented a method for tracking in overlapping stationary and pan-tilt-zoom cameras by maximizing a joint motion and appearance probability model. A Bayesian formulation of the problem of reconstructing the path of objects across multiple non-overlapping cameras was presented in [10] using color histograms for object appearance. A graph-theoretic framework for addressing the problem of tracking in a network of cameras was presented in [5]. Adapting the feature correspondence computations by modeling the long-term dependencies between them and then obtaining the statistically optimal paths for each person differentiates our approach from existing ones. It provides a solution that is robust to errors in feature extraction, correspondence and environmental conditions.

2. Problem Formulation

Our problem is to track P people observed over a network of $C_1, ..., C_K$ cameras. This is abstracted as tracking over a collection of nodes, where each node is an entry or exit point (like in [14, 16]). We assume that we know which camera each node can be viewed from (also refereed to as a node belonging to a camera). Mathematically, each node is represented as n_i^c, where the subscript is a node index and the superscript represents the camera it belongs to. For the purposes of this paper, we assume that we can track people within the view of each camera. The cameras are synchronized and thus, each observation can be given a unique time stamp and location information in terms of the node it is observed from.

A network architecture linking the nodes is known. By this we mean that given any pair of nodes $(n_i^c, n_j^d), i \neq j$, we have a link variable $l_{ij} = \{0, 1\}$, where 0 indicates that the two nodes are not linked, i.e., it is not possible to travel between those two nodes without traversing some other node, and 1 indicates that it is possible to travel between these two nodes directly. However, $l = 1$ on a link does not rule out the possibility that a person could have travelled from n_i to n_j through some other node $n_k, k \neq i, j$. Besides this link variable, we also know a distribution of the travel time between two nodes, i.e., $P_\tau(n_i^c, n_j^d), i \neq j$, where $P : \Re^+ \rightarrow [0, 1]$. One of these nodes must be an entry node and the other an exit node.

Observations at each node are represented as feature vectors $\mathbf{F}_{n,t}$, where n indicates the node it is observed at and t the time of observation. The feature vector of choice in this paper is explained in Section 3.1. Each node (i.e., the camera the node belongs to) receives information about the feature vectors from all other nodes that it is linked to. Feature vectors within a time window are stored at that node. Making this precise through an example as shown in Figure 2, let node n_i be linked to nodes (n_j, n_k, n_l, n_m) (we drop

the superscription since the camera identity is not needed). At time t, n_i has feature vectors $\mathcal{F}_{i,t} = \{\mathbf{F}_{n,t_j}, n = (j,k,l,m), t - t_W < t_j \leq t\}$, where t_W is the width of the time window. Note that these observations from neighboring nodes are available only at discrete instants of time, i.e., t_j is a discrete variable. In practice, whenever a person exits a camera view that information is sent by the camera to all the other cameras linked to it based on the network architecture. Thus, (n_j, n_k, n_l, n_m) are all exit nodes and n_i is an entry node.

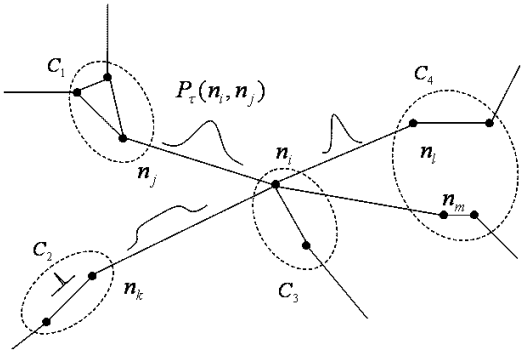

Figure 2. An example of a camera network, node n_i is linked to nodes (n_j, n_k, n_l, n_m) and a distribution of the travel time between them is known.

Overview of Solution Strategy: When a node n_i encounters a new observation at time t, it transforms it into a feature vector $\mathbf{F}_{n,t}$. Then, it computes the similarity of this feature vector with its stored feature vectors, $\mathcal{F}_{i,t}$, obtained from neighboring nodes. The exact procedure for similarity computation is explained in Section 3.1. This is then transformed into a distribution on the similarity between two features. The distribution captures the uncertainty in computing the similarity between two features and is learned during a training phase. For example, if two nodes have very different lighting conditions, the uncertainty in the similarity scores between the features observed at those nodes will be higher. This similarity computation is asynchronous between the different nodes, i.e., each node does it whenever it encounters a new observation.

Using the feature vectors and similarity scores, we create a feature graph $G = (V = \{v_i\}, E = \{e_{ij}\}, \tilde{S} = [\tilde{s}(e_{ij})])$, of $|V| = V$ vertices and $|E| = E$ edges. The vertices are feature vectors $\mathbf{F}_{n,t}$ and the weights on the edges, $\tilde{s}(e_{ij})$, are real-valued random variables with known distribution $p_{\tilde{s}}(s)$, as shown in Figure 3. The tracks of each person can be found by computing the optimal paths in this graph. We show how this can be done using stochastic weights in Section 4.1.

This optimal path computation is based on the principles of dynamic programming and gives the maximum a posteriori (MAP) estimate of the track for each person. However, this is true if the similarity computations in the feature graph are correct. This can be a big assumption due

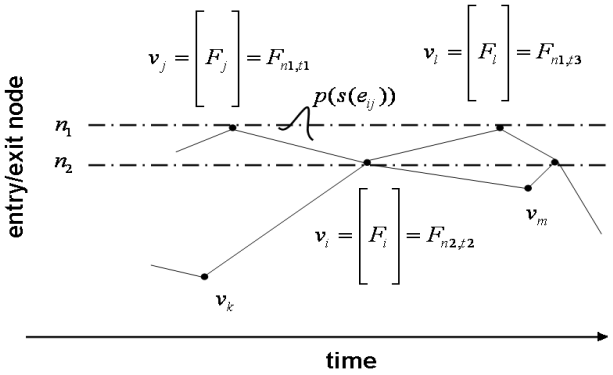

Figure 3. Illustration of feature graph construction. Note the subscripts of F indicating the time and entry/exit node where it is observed.

to the well-known problems of video-based feature correspondence. By considering the distribution of the features over space and time in each hypothesized path, we can infer about the correctness of that path. This leads us to develop a closed-loop, adaptive system whereby the weights along each path are adapted by considering the distribution of features along that path. The optimal path computation and adaptation continue till a suitable stopping criterion is reached. This process ensures that we retain the advantages of dynamic programming while also considering the long-term dependencies in the feature graph. It is explained in detail in Sections 4.2 and 4.3.

3. Computing Stochastic Similarity Scores

Consider the above-mentioned feature graph G. The problem of tracking people across cameras is equivalent to finding the most preferred links between vertices in this graph. To differentiate this feature graph from the network topology, we will use the term node to refer to an entry/exit point and vertex to refer to an observed feature vector.

3.1. Feature Similarity Scores

The feature vector $\mathbf{F}_{n,t}$ can be represented as

$$\mathbf{F}_{n,t} = \begin{pmatrix} F_A \\ F_I \end{pmatrix}_{n,t} \tag{1}$$

where F_A is the appearance feature (normalized color) and F_I is the identity feature (gait width vector [7, 8]). Besides these, we know the travel time between two nodes. Since every feature is observed at a node, this can be easily transformed into the travel time between two feature vectors, i.e., two vertices of the feature graph. Let us denote this travel time between $\mathbf{F}_{n_i,t_1}$ and $\mathbf{F}_{n_j,t_2}$ as $\tau_{n_i,n_j}^{t_1,t_2}$. From $P_\tau(n_i^c, n_j^d)$, we can estimate the feature similarity between $\mathbf{F}_{n_i,t_1}$ and $\mathbf{F}_{n_j,t_2}$ in terms of the travel time as $\mathcal{S}_\tau(\tau_{n_i,n_j}^{t_1,t_2})$. It is reasonable to assume that F_A, F_I and τ are independent random variables. For notational simplicity, we will

drop the explicit dependence on n, t and represent the features as vertices on the graph, i.e., use $F_i, F_{A,i}, F_{I,i}$ and $\tau_{i,j}$, (see Figure 3).

The weight $s(e_{ij})$ is the similarity score between $\mathbf{F}_i$ and $\mathbf{F}_j$, and is computed as the product of the similarities in identity features, appearance features, and the travel time based similarity value explained above, i.e.,

$$s(e_{ij}) = \mathcal{S}_A(F_{A,i}, F_{A,j})\mathcal{S}_I(F_{I,i}, F_{I,j})\mathcal{S}_\tau(\tau_{i,j}). \quad (2)$$

The similarities can be obtained by computing the distance between the feature vectors in appropriate feature spaces as described in [2, 8]. Known geometric and photometric transformations between two corresponding features should be taken into account while computing this distance. Examples include affine warping and brightness transfer functions [6, 7]. These can be learned during the training phase for the network topology.

3.2. Assigning Uncertainty to Similarity Scores

In our formulation, the similarity score s is a realization of a random variable $\tilde{s}$. We now need to compute the distribution of this random variable. If s'_{ij} is the similarity score between two vertices (i, j) in the feature graph obtained as described above, the distribution of $\tilde{s}$ on the edge e_{ij} is modeled as a normal distribution $\mathcal{N}(s'_{ij}, \sigma^2_{ij})$. Thus the distribution on each edge is normal with a mean at the similarity score value obtained from the observations and a variance that will be learned from training data.

Unsupervised Learning of Variance of Similarity Distribution: The confidence that we can assign to an observed similarity score will depend upon a number of factors and is too difficult to be modeled analytically. These include the actual value of the similarity score, the environmental conditions like time of day, the geometric and photometric transformation between the cameras, and so on. Thus we seek to learn a model of the variance of the similarity score during a training phase. This training can continue parallely with the network topology learning phase. Due to the sheer volume of data in a camera network, we seek to develop an unsupervised learning mechanism for this purpose.

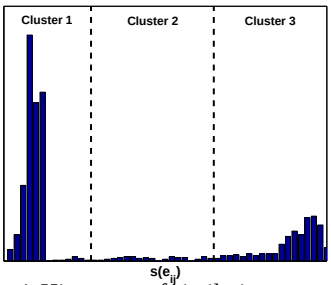

Figure 4. Histogram of similarity scores between v_i and v_j learned during the training phase.

During this learning phase, a large number of similarity scores between two vertices will be collected. These will be clustered and the variance of each cluster computed. During the system operation, the variance σ^2_{ij} is determined based on which cluster s'_{ij} belongs to. As an example, we show the learned histogram of the similarity scores between two vertices in Figure 4. We see that the data can be partitioned into three clusters and we can learn the variance of each cluster. This can be done using K-means algorithm, including automatic estimation of the number of clusters [1].

4. Stochastic Adaptive Strategy for Optimal Path Calculation

Our overall framework for tracking across the cameras consists of the following steps which take place in a loop: (i) finding optimal paths in the stochastically weighted feature graph (Fig. 3) based on the distribution of edge similarity scores, (ii) calculating path smoothness function (PSF) along each hypothesized path, and (iii) adapting the distribution of the edge weights according to the values of the PSF. We will explain each step in detail below.

4.1. Optimal Path in Stochastic Feature Graphs

Solving the optimal path problem in graphs with weights as random variables has been studied in communication networks and decision theory [13]. The most widely used method for this problem was originally proposed by von Neumann and Morgenstern [3], which proposed an utility function and then computed the optimal expected utility. A necessary condition for a valid utility function (hence our weighting function) is that it has to be monotonic, affine linear or exponential (see [13] for a rigorous proof).

Inspired by this method, we define a weighting function on the edge similarity scores. We identify the most preferred set of paths by maximizing the Expected Weighted Similarity (EWS). Thus the weighting function for similarity, s, is defined as

$$w(s) \stackrel{\triangle}{=} exp(s) - 1. \quad (3)$$

The weighting function is designed to give higher weights to correspondences with high similarity scores. Its effect becomes most prominent when the random variable, $\tilde{s}$, has a high variance. Thus, in computing the EWS (expected utility), both the observed similarity and its uncertainty are taken into account through the weighting function (utility function) [1] .

[1] To show that the EWS finds the optimal path by considering not only the similarity scores but also their variance, consider two links with similarity scores of s_1 and s_2, where $s_1 > s_2$. Now, if $p(s) \sim \mathcal{N}(s', \sigma)$, then $E_s[w(s)] = exp(s' + \frac{\sigma^2}{2}) - 1$, where s' is the observed similarity score. Thus the optimal path under the EWS criterion will not necessarily be s_1 (which would be the optimal path in the deterministic case), but will depend upon the similarity scores as well as the variances in these two links.

By introducing the weighting function, the solution to the problem of determining the best set of paths in the feature graph (hence the camera network) is

$$\{\tilde{\lambda}_q\} \triangleq \arg\max_{\lambda_q} \sum_{\lambda_q} \{ \sum_{e_{ij} \in \lambda_q} E_s[w(s(e_{ij}))]\}. \quad (4)$$

This problem can be formulated as the maximum matching problem in a weighted bipartite graph, where the weights of the edges are the EWS scores, $E_s[w(s(e_{ij}))]$. As in [5], the bipartite graph is obtained by splitting each vertex v into v^- and v^+, where the edge connected to v^- represents the path coming into v while the edge connected to v^+ represents the path going out of v.

4.2. Analyzing Features Along A Path

If the similarity scores (edge weights) of the feature graph G were known exactly and assumed to be independent, the tracking problem could be solved optimally in polynomial time by the method described above. This was shown in [5] with the exception that they did not model the uncertainty in the similarity scores. However, it is not uncommon for some of the similarities to be calculated wrongly due to poor lighting conditions or ambiguity of the transition patterns. Even the learned uncertainty model may not be enough to capture the variation. As we show in Figure 5, if the similarity computation is incorrect for one pair of nodes, the overall inferred path may be wrong even if all the other nodes are connected correctly. The exact conditions where such mistakes happen will depend upon the distribution of the similarity scores in the feature graph.

This concern can be addressed if we relax the independence assumption on the correspondences, i.e., we will consider the interdependence of the features and similarity scores over space and time. Intuitively, what we aim to do by this process is to infer incorrect path segments in the tracks obtained from the graph theoretic approach of Sec. 4.1 by considering the variation of the features of the person along these tracks. Based on this inference process we will adapt the similarity scores and recompute the optimal path. This naturally leads to the development of the stochastic adaptive tracking framework.

For this purpose, we define a Path Smoothness Function (PSF). Given an estimated path for the q^{th} person, λ_q, PSF is defined on each edge $e_{ij} \in \lambda_q$. The feature vertices before (in time) e_{ij} on λ_q and those after e_{ij} are treated as two clusters. Let $\{X\}$ be the set of all N feature vertices (i.e., appearance and identity vectors) along the path and let them be clustered into $\{X^{(1)}\}$ and $\{X^{(2)}\}$ with respect to each edge $e_{ij} \in \lambda_q$. Let the mean m of the features in $\{X\}$ be $m = \frac{1}{N}\sum_{x\in\{X\}} x$. Let m_i be the mean of N_i data points of class $\{X^{(i)}\}$, $i = 1, 2$, such that

$$m_i = \frac{1}{N_i} \sum_{x\in\{X^{(i)}\}} x. \quad (5)$$

Let

$$S_T = \sum_{x\in\{X\}} |x - m|^2 \quad (6)$$

and

$$S_W = \sum_{i=1}^{2} S_i = \sum_{i=1}^{2} \sum_{x\in\{X^{(i)}\}} |x - m_i|^2. \quad (7)$$

The PSF for e_{ij} is defined as

$$PSF(e_{ij}) = \frac{|S_T - S_W|}{|S_W|} = \frac{|S_B|}{|S_W|}. \quad (8)$$

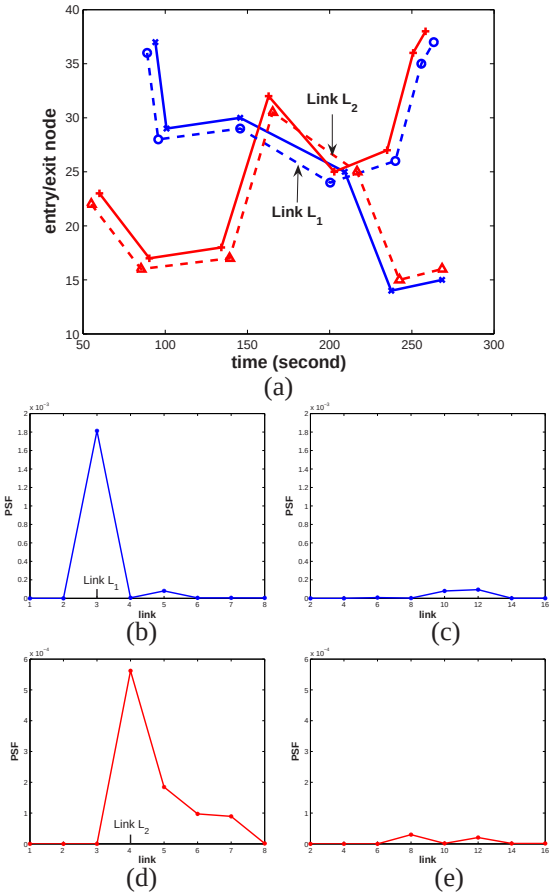

Figure 5. (a): Paths of two people (in blue and red) obtained from the camera network: solid lines are the ground truth and dotted lines are the tracking results using EWS scores only (no adaptation). (b)-(c): PSF values along the incorrect (estimated) and correct (ground truth) path of person 1 respectively. (d)-(e): PSF values along the incorrect and correct path of person 2 respectively. It is clear that PSF has a peak at the wrong link; thus the variance of PSF along wrong path is higher than the variance along correct path.

To make the PSF more accurate, we may consider normalizing all the elements of $\{X\}$ with respect to the geometric and photometric effects between the feature vertices. For example, if $\mathcal{B}_{i,j}$ is the BTF [6] between v_i and v_j and $\mathcal{G}_{i,j}$ is an affine warp (these can be learned during the training phase), we should consider $(\mathcal{G}_{i,j} \circ \mathcal{B}_{i,j})(\mathbf{F}_i)$ and $\mathbf{F}_j$ for this pair of nodes.

Thus the PSF is defined from Fisher's linear discriminant function [1] and measures the ratio of the distance between different clusters, S_B, over the distances between the members within each cluster S_W. If all the feature nodes along a path belong to the same person, the value of PSF at each edge should be low, and thus the variance of PSF over all the edges along the path should also be low. If the feature nodes belonging to different people are connected wrongly, we will get a higher value of PSF at the wrong link, and the variance of PSF along the path will be higher. Thus, the distribution of PSF along a path can be used to detect if there is a wrong connection along that path. An example is shown in Fig. 5, where the PSF is plotted for all edges along one correct and one incorrect path obtained from our camera network.

4.3. Closed-Loop Adaptation of Edge Similarities

Whenever there is a peak in the PSF function for some edge along a path, the validity of the connections between the features along that path is under doubt. We will adjust the weight on this link where the peak occurs by reducing the mean of the distribution of the weights and adjusting the variance based on the learned values for each range of similarity scores (see Section 3.2). For each iteration, the mean can be adjusted as $\mu^{(n)}(s(e_{ij})) = a\mu^{(n-1)}(s(e_{ij})), a \in (0,1)$. Then we will recalculate the optimal paths as described in 4.1 using the new weights. The two steps of each iteration are as follows:

$$s^{(n+1)}(e_{ij}) = f_{adap}\big(s^{(n)}(e_{ij}), \sum_{\lambda_q} Var(PSF(e_{ij} \in \lambda_q^{(n)}))\big),$$

$$\lambda_q^{(n+1)} = \arg\max_{\lambda_q} \sum_{\lambda_q} \{ \sum_{e_{ij} \in \lambda_q} E_s[w(s^{(n+1)}(e_{ij}))]\}, \quad (9)$$

where $f_{adap}(.)$ is the weight adaptation function which adjusts the distribution of edge weights according to the values of PSF. This process of weight adaptation and optimal path computation will continue in a closed-loop till we reach a local minimum of $\sum_{\lambda_q} Var(PSF(e_{ij} \in \lambda_q))$. This process is repeated for each possible path, λ_q.

4.4. Stochastic Adaptive Tracking Algorithm

We are now ready to outline the main steps of the stochastic adaptive tracking algorithm. From practical application considerations, we use a time window $T_j \leq t \leq T_{j+1}$ in which the algorithm runs. Consider the feature vectors $\{\mathbf{F}_{n,t}\}$ which are observed in this time interval $[T_j, T_{j+1}]$.

1. Construct a stochastic weighted graph $G = (V, E, S)$, where the vertices are the feature vectors and distribution of edge weights are set as described in Section 3.

2. Compute the optimal paths, λ_q as mentioned in Section 4.1.

3. Compute the PSF for each $e_{ij} \in \lambda_q$ and adapt the distribution of edge weights according to f_{adap}.

4. Repeat Steps 2 and 3 until a local minimum of $\sum_{\lambda_q} Var(PSF(e_{ij} \in \lambda_q))$ is reached. The final set of optimal paths is given by $\big(s^*(e_{ij}), \lambda_q^*\big)$ as

$$s^*(e_{ij}) = \arg\min_{s} \sum_{\lambda_q} Var(PSF(e_{ij} \in \lambda_q(s))), \quad (10)$$

$$\lambda_q^* = \arg\max_{\lambda_q} \sum_{\lambda_q} \{ \sum_{e_{ij} \in \lambda_q} E_s[w(s^*(e_{ij}))]\}. \quad (11)$$

5. Experimental Results

To evaluate the performance of our system, we will show results on data collected over a large camera network. The network consists of 20 cameras and 50 entry/exist nodes. We consider 10 people moving across the network. Examples of some of the images of the people are shown in Figure 6. Compare this plot with the abstract representation of Figure 3.

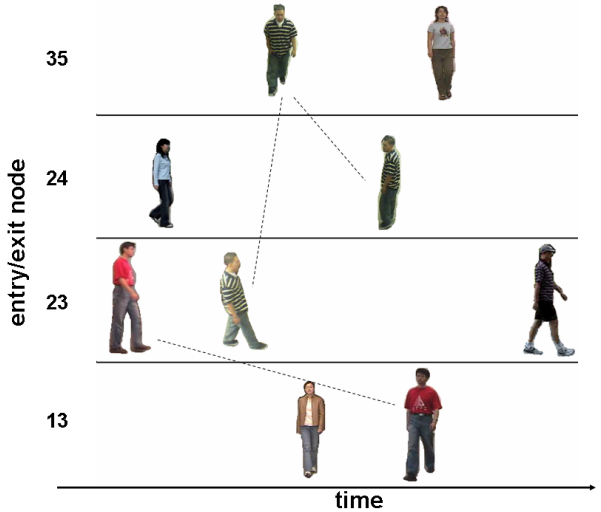

Figure 6. Example of some of the images of the people in the network. The horizontal axis is the time when these features were observed, while the vertical axis is the index of the entry/exit node where the features were observed. The gaps in the plot are because the features are observed asynchronously.

Figure 7 shows the tracking result using optimal path searching using EWS scores only (open-loop framework). The vertices in this graph are the feature vectors observed at different time instances. The paths of different people are shown using different colors, solid lines representing the ground truth and dotted line showing tracking results. Thus, for correct results the colors should match across all the paths. There are three wrong links (circled in Figure 7 – two in the bigger circle, and one in the smaller circle) due to mistakes in feature similarity computation. Details of the wrong link shown in the bigger circle are given in Figure 5. The paths that are estimated wrongly due to these wrong links are highlighted. In Figure 8, we show the correct result by adapting the graph edge weights using the proposed stochastic adaptive strategy. By comparing the colors along the paths highlighted in Figure 7, we can see that these mistakes have been corrected. Both these figures are best viewed on a computer monitor.

We now show some details on the weight adaptation along a wrong link. We show the plot for variation of PSF versus iteration number in Figure 9. When the PSF reaches a local minimum, the estimated paths are correct. It can be seen that for some iterations, the adaptation of some of the path similarity scores (edge weights of the feature graphs) may not change the output of optimal path estimate. The iterations will stop when the variation of PSF starts to increase indicating that it has reached a local minimum.

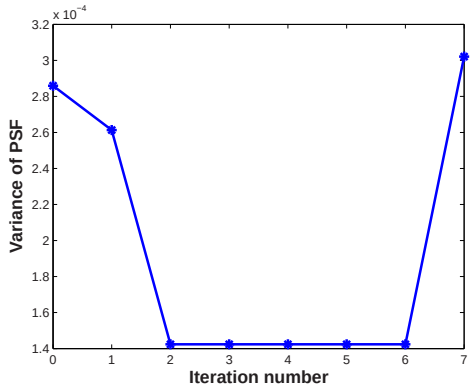

Figure 9. Variation of PSF versus iteration number.

6. Conclusions

In this paper, we proposed a novel stochastic, adaptive framework for tracking multiple people in a large network of video cameras. We considered the uncertainty for feature similarity computation and treated them as random variables. We considered the long-term interdependence of the features over space and time. We derived a path smoothness function (PSF) to correct wrong correspondences. We showed that the above two conditions can be addressed by treating the issue of tracking in a camera network as as an optimization problem in a stochastic adaptive system. We demonstrated the effectiveness of our system by showing results on a real life camera network.

References

[1] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[2] G. D. Finlayson, B. Schiele, and J. L. Crowley. Comprehensive colour image normalization. In *ECCV*, 1998.

[3] P. Fishburn. Utility theory. *Management Science*, 14:335–377, 1968.

[4] T. Huang and S. Russel. Object identification in a bayesian context. In *Proceeding of IJCAI*, 1997.

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

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

[7] A. Kale, A.Roy-Chowdhury, and R. Chellappa. Towards a View Invariant Gait Recognition Algorithm. In *IEEE AVSS*, 2003.

[8] A. Kale, A. Rajagopalan, A. Sundaresan, N. Cuntoor, A. Roy-Chowdhury, A. Krueger, and R. Chellappa. Identification of Humans Using Gait. *IEEE Trans. on Image Processing*, pages 1163–1173, September 2004.

[9] J. Kang, I. Cohen, and G. Medioni. Continuous tracking within and across camera streams. In *IEEE CVPR*, 2004.

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

[11] P. Kumar. A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23(3):329–380, 1985.

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

[13] R. P. Loui. Optimal paths in graphs with stochastic or multidimensional weights. *Communications of the ACM*, 26(9):670–676, 1983.

[14] D. Makris, T. Ellis, and J. Black. The gaps between cameras. In *IEEE CVPR*, 2004.

[15] D. Marinakis, G. Dudek, and D. Fleet. Learning sensor network topology through monte carlo expectation maximization. In *IEEE ICRA*, 2005.

[16] C. Niu and E. Grimson. Recovering non-overlapping network topology using far-field vehicle tracking. In *ICPR*, 2006.

[17] A. Rahimi and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *IEEE CVPR*, 2004.

[18] K. Tieu, G. Dalley, and E. Grimson. Inference of non-overlapping camera network topology by measuring statistical dependence. In *IEEE ICCV*, 2005.

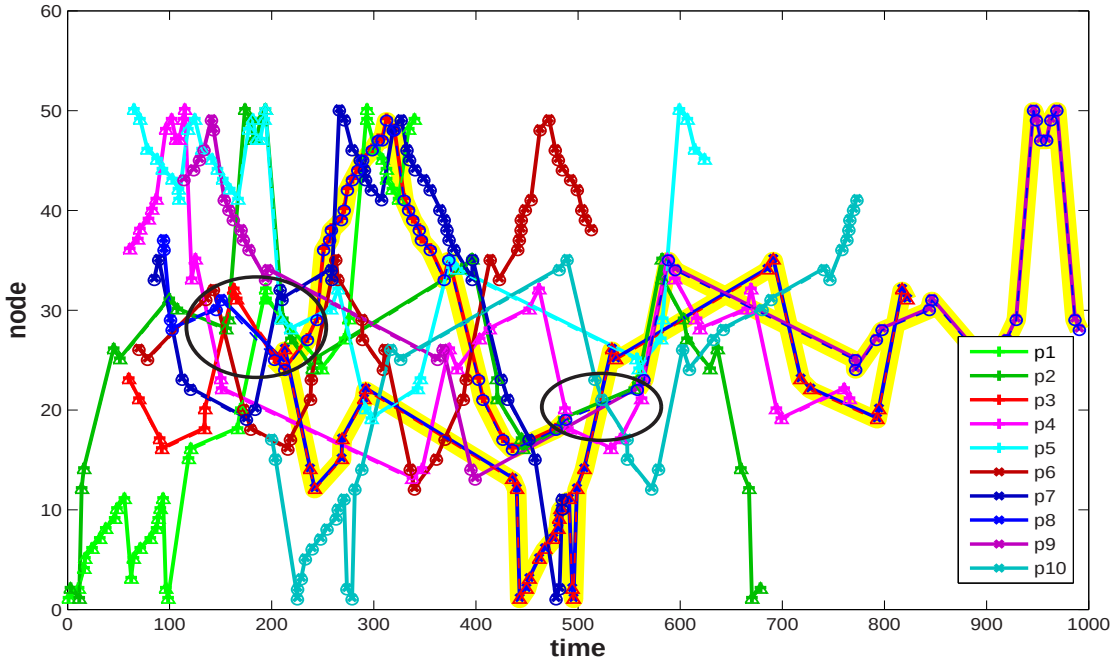

Figure 7. Tracking result using EWS scores only (open-loop framework) in feature graph. Horizontal axis is time, while vertical axis is the entry/exit node where a feature is observed. The paths of different people are shown using different colors, solid lines representing the ground truth and tracking results are showed by dotted lines. The wrong links are circles, while the wrong paths due to these links are highlighted. Details of the wrong link shown in the bigger circle are given in Figure 5. (The figure is best viewed on a color monitor.)

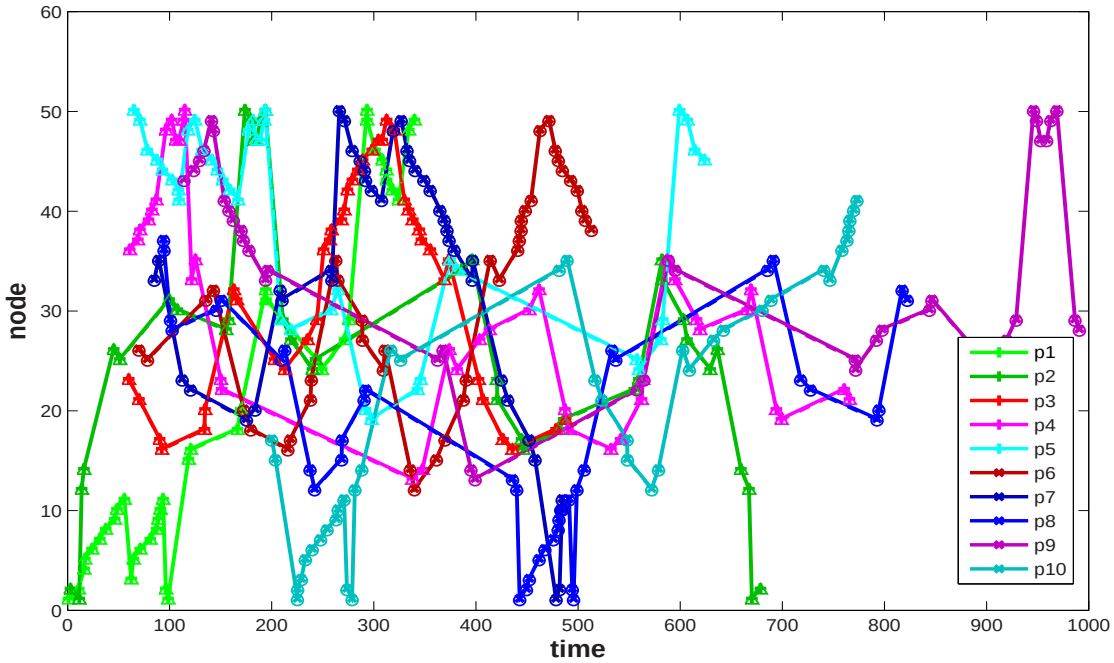

Figure 8. Tracking result using our stochastic adaptive algorithm using the same representation as Figure 7. Compare the color of the paths along the highlighted portions of Figure 7. The colors of the dotted and solid lines match indicating correct path estimation. (The figure is best viewed on a color monitor.)