UNIVERSITY OF CALIFORNIA
RIVERSIDE

Scene Analysis, Control and Communication in Distributed Camera Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Bi Song

June 2009

Dissertation Committee:

Professor Amit K. Roy-Chowdhury, Chairperson
Professor Bir Bhanu
Professor Ertem Tuncel

The Dissertation of Bi Song is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

In the years of working as a Ph.D. student at UCR, many people were instrumental directly or indirectly in shaping up my academic career. I own my gratitude to all those people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

I would like to express my deepest gratitude to my advisor, Dr. Amit K. Roy-Chowdhury, for his guidance and supervision, and for the opportunities he has created for me during my graduate research and career. I thank him for his consistent guidance, suggestion, enlightenment and help in every detail of my research. I had a long way to go to get there, and he has been just what I needed him to be every step of the way.

I would also like to thank Professor Bir Bhanu and Professor Ertem Tuncel for their insightful advices and inputs to this work and for reviewing my dissertation. I thank Professor Matthew Barth and Professor Subir Ghosh for serving as committee members in my qualifying exam. Special thanks to Professor Namrata Vaswani at Iowa State University for her help and guidance.

I would like to acknowledge my mentors during internship, Dr. Kevin Zhou, Dr. Peiya Liu, Mr. Sridharan Palanivelu and Dr. Gianfranco Doretto. Their advice encouragement and friendship, make my experience at Siemens and GE unforgettable in my life.

Many thanks go to my colleagues for rewarding research discussions, and helpful comments on my thesis and numerous manuscripts. It is their friendship and camaraderie have made my years in Riverside the most unforgettable ones in my life. I was very lucky to enter graduate school with Dr. Yilei Xu. We learned so much together during these past five years, experienced some great times and help each other through the hard times. I would like to thank him for everything, congratulate him on his exceptional work. I am also thankful to the past and present IPLers, Dr. Jayanth Nayak, Mr. Cristian Soto, Mr. Luis Gonzalez-Argueta, Ms. Ozgun Bursalioglu, Mr. Min Liu, Ms. Xuechen Chen, Mr. Yang Gao, Mr. Ricky Sethi, Mr. Uttkarsh Gaur, Ms. Ting-Yueh Jeng, Mr. Ahmed Tashrif Kamal, Mr. Chong Ding and Mr. Ufuk Celikcan. I thank my friends at Riverside, Lei Shu, Zheng Fang, Meng Cao, Weihua Zhu, Kezhu Hong, Ning Mi, Rong Wang, Xiaoli Zhou, Rui Li, Xiaotao Zou, Yiming Li, and all other friends.

Most importantly, none of this would have been possible without the love and patience of my family. I would like to express my heart-felt gratitude to my parents for

their constant love, concern, support and strength all these years. Last but not least, I would like to thank my husband, Jingxian Lin, for his continuous moral support. I don't think I would have made it if he hadn't become a part of my life.

To my parents and my husband.

ABSTRACT OF THE DISSERTATION

Scene Analysis, Control and Communication in Distributed Camera Networks

by

Bi Song

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2009
Professor Amit K. Roy-Chowdhury, Chairperson

As networks of video cameras are being installed in many applications, modeling and inference strategies in video networks have captured more and more interest. There are many challenge problems, such as (i) traditional computer vision challenges in tracking and recognition, robustness to pose, illumination, occlusion, clutter; recognition of objects and activities; (ii) aggregating local information to obtain stable, long-term tracks of objects; (iii) cooperative camera control algorithms for multi-resolution target acquisition; (iv) distributed processing and scene analysis and (v) communication in a distributed manner. The overall aim of this thesis is to study the core issues in network-centric processing, control and communication in a multi-camera network, including frameworks for tracking people in video through changes of activities, tracking in a non-overlapping camera network, decentralized control and tracking in a camera network, and distributed video compression.

In our work, we address the problem of tracking in camera networks by dividing it into two parts: tracking people through changes in activities and tracking multiple targets in a network of non-overlapping cameras. We present a novel framework for tracking a long sequence of human activities, including the time instances of change from one activity to the next, using a non-linear switching dynamical feedback system. Also, a multi-objective optimization framework by combining short term feature correspondences across the cameras with long-term feature dependency models is proposed to take care of tracking in a non-overlapping network.

We also deal with the problem of decentralized, cooperative control of a camera

network and distributed multi-target tracking in a such network of self-configuring pan-tilt-zoom cameras. This control cannot be based on separate analysis of the sensed video in each camera. They must act collaboratively to be able to acquire multiple targets at different resolutions. Our research focuses on developing accurate and efficient camera control algorithms in such scenarios using game theory. For tracking the targets as they move through the area covered by the cameras, we propose a special application of the distributed estimation algorithm known as Kalman-Consensus filter through which each camera comes to a consensus with its neighboring cameras about the actual state of the target.

Finally, we present a framework for multi-terminal video compression (MTVC) that exploits the geometric constraints between cameras with overlapping fields of view, and then uses distributed source coding on corresponding points in two or more views. Our proposed method is composed of two parts - a Distributed Motion Estimation (DME) algorithm and a Distributed Source Coding (DSC) algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Networks of video cameras are being installed in many applications, *e.g.*, video surveillance, national and homeland security, assisted living facilities, etc. 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 people performing different activities in a single view as well as track objects across the network. A recent review [35] suggested that tracking long sequences of multiple activities is still a challenging problem. Also, tracking across network introduces certain challenges that are unique to this particular application scenario, in addition to existing challenges in tracking objects like pose and illumination variations, occlusion, clutter and sensor noise. The problem is important for applications in wide area surveillance, digital forensics and video summarization, among others.

Moreover, it is natural to expect that these camera networks would be used to track targets at multiple resolutions, *e.g.*, multiple people, a single person, a face. For efficiency and maximum resource utilization, it is desirable to *actively* control the cameras so as to track the targets based on the requirements of the scene being analyzed. It would be prohibitively expensive to have a static setup that would cater to all possible situations. It is also desirable that the tracking and control mechanism be distributed due to constraints of bandwidth, secure transmission facilities, and difficulty in analyzing a huge amount of data centrally. In such situations, the cameras would have to act as *autonomous agents* and decisions would have to be taken in a distributed manner.

Meanwhile, transmission of video data from multiple sensors over a wireless net-

work requires enormous amount of bandwidth, and could easily overwhelm the system. However, by exploiting the redundancy *between* the video data collected by different cameras, in addition to the inherent temporal and spatial redundancy *within* each video sequence, the required bandwidth can be significantly reduced. As we show in this thesis, it is possible to integrate distributed source coding concepts with video analysis tools that exploit the inherent geometrical constraints between two views.

1.1 Literature Review

A review of recent work in human activity analysis can be found in [35]. Based on the conclusions of [35] and a survey of some of recent representative methods on event analysis [6, 32, 37, 46, 67, 70, 84, 89, 95, 98, 101, 103], we find that most existing methods handle events of short duration with moderate changes in illumination, scene clutter and occlusion. A few techniques have studied the problem of tracking multi-activity sequences using the idea of simultaneous tracking and recognition. In [63], the idea of integrated tracking and recognition of human activities using particle filters was proposed. Simultaneous tracking of moving people and recognition of their activities has also been performed in many applications using a Dynamic Bayesian Network (DBN) model [9, 13, 52, 91]. However, all these methods require a-priori knowledge of the transition probability matrix for switching between different activities. Designing such a transition matrix for uncontrolled environments like video surveillance may be difficult (since the class of possible activities is large and can change arbitrarily from one to another) and the process has to repeat for each application scenario.

Some of the existing methods on tracking in a camera network include [36, 39, 44]. The authors in [74] 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 [49], a particle filter was used to switch between track prediction between non-overlapping cameras and tracking within a camera. In [43], 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 [44] using color histograms for object appearance. A graph-theoretic framework for

addressing the problem of tracking in a network of cameras was presented in [39].

A related work that deals with tracking targets in a camera network with PTZ cameras is [73]. Here, the authors proposed a mixture between a distributed and a centralized scheme using both static and PTZ cameras in a virtual camera network environment. In [18], a solution to the problem of optimal camera placement given some coverage constraints was presented and can be used to come up with an initial camera configuration. A broad framework for the vehicle-target assignment problem using game theory was presented in [2].

There has recently been significant effort in application of DSC (Distributed Source Coding) techniques to video data. A recent method [90] attempts to exploit the redundancy between images available at *different* sensor nodes by independently encoding the images in low resolution and decoding using super-resolution techniques. Another recent work [25, 26] developed a distributed image coding technique for a multi-camera setting. A recent paper considers inter- and intra-sensor correlation for transmission after making some simplifying assumptions about the locations of the sensors and not making explicit use of the geometrical transformations relating the images in various sensors [12]. The work in [96] also addresses multiterminal video coding in its most general sense. For a setting where feedback from the central decoder to one of the encoders is allowed, [21] provides a mechanism for efficient exploitation of inter-sensor redundancy.

1.2 Contributions of the Thesis

In this thesis, we consider some of the core problems in network-centric processing, control and communication in a camera network. Specifically, the thesis makes the following original contributions.

• **Tracking in Multi-Activity Sequences:** Most existing methods, which study tracking long sequences of multiple activities, require a-priori knowledge of the transition probability matrix for switching between different activities. However, designing such a transition matrix may be difficult. In contrast to the open-loop approach, we propose to use change detection measures (e.g., tracking error) to detect transitions between activities, and use them as a feedback signal in a closed-loop system. The main advantages of our approach are: significant computational savings in tracking changes of body posture using

the particle filtering framework; obviating the need to learn a complex transition matrix; and higher efficiency and accuracy in applications where such a transition matrix is not known.

• **Framework for Tracking across Non-Overlapping Camera Network:** We present a robust framework for tracking multiple objects in a *non-overlapping* camera network whereby feature correspondences are modeled statistically and adapted in time by considering the long-term dependencies between them. We pose the problem of tracking in a camera network as a multi-objective optimization problem. 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.

• **Game Theory based Control in Distributed PTZ Camera Network:** We deal with the problem of decentralized, cooperative control of a camera network. We focus on applications where events unfold over a large geographic area and need to be analyzed by multiple cameras or other kinds of imaging sensors. There is no central unit accumulating and analyzing all the data. A camera control framework based on game theoretical ideas allowed for viewing some targets at a high resolution while keeping the entire area under surveillance covered at an acceptable resolution.

• **Distributed Multi-Target Tracking Using Kalman-Consensus Filter:** We present a robust approach to distributed multi-target tracking in a network of self-configuring cameras. A distributed Kalman-Consensus filtering approach was used together with a dynamic network topology for tracking persistently multiple targets across several camera views in an area under surveillance viewed by a dynamic camera network.

• **Multi-Terminal Video Compression:** we develop the framework for a novel multi-terminal video coding (MTVC) algorithm combining distributed source coding (DSC) and computer vision techniques that impose geometrical constraints between the observed views of the two cameras. This lossy compression scheme takes into account the correlation between the video sensor data, and at the same time keeps the communication between the sensors at a minimum.

1.3 Organizations of the Thesis

The thesis is organized along the lines of the previous section. In Chapter 2, we present the close-loop framework for tracking multi-activity sequences with unknown transition probabilities. A multi-objective optimization framework for tracking in a non-overlapping camera network is described in Chapter 3. The decentralized camera network control and tracking algorithm is presented in Chapter 4. Chapter 5 describes the multi-terminal video compression algorithm by integrating distributed source coding with geometrical constraints.

Chapter 2

Closed-Loop Tracing and Change Detection in Multi-Activity Sequences

2.1 Introduction

Video sequences often have people performing multiple activities one after another. For example, Fig. 2.1 shows some frames from an educational Yoga video where the instructor transits continuously between different postures.

A recent review [35] suggested that tracking long sequences of multiple activities is still a challenging problem. A few existing papers like [63, 9, 52] have studied this problem. However, all these methods require a-priori knowledge of the transition probability matrix for switching between different activities. Designing such a transition matrix for uncontrolled environments like video surveillance may be difficult, since the class of possible activities is large and can change arbitrarily from one to another, and the process has to repeat for each application scenario. For a non-overlapping camera network, such a matrix may not be very useful because we do not know what the person may have been doing when not visible. This motivated us to design methods that can track long multi-activity sequences *without* the need to learn a transition probability matrix.

In this chapter, we propose such a method for tracking of human activities consisting of the following steps which take place in a loop: (i) modeling the appearance and

motion of single activity sequences and tracking them, (ii) detecting a change from one sequence to the next, and (iii) automatically reinitializing after the change and starting tracking. A diagram explaining this is shown in Fig. 2.1.

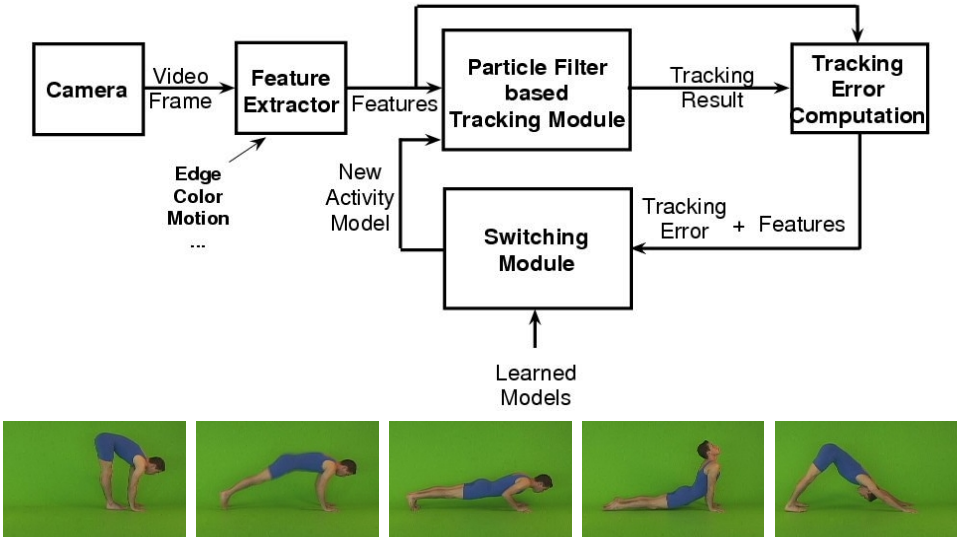


Figure 2.1: Framework for closed-loop tracking and an application scenario.

Activities are represented mathematically using dynamical models defined on the shape of the contour of the human body, motivated by the fact that the shape of the body changes in the course of various activities. The shape representation makes the method insensitive to camera scale changes and translation. Tracking is performed using an auxiliary particle filter [71], incorporating piecewise stationary dynamical models and a nonlinear observation equation that relates the shape, color and motion with the input image. The reason for choosing a particle filter over an extended Kalman filter (EKF) is multimodal observation likelihood due to either background clutter or partial occlusions. Change detection measures, like estimated tracking error (TE) [4], are used to detect when the activity deviates from the current model. When this happens, a new dynamical model is automatically chosen and used to reinitialize the tracking. We show the applicability of this framework to track multi-activity sequences within the view of a single camera, as well as between a collection of non-overlapping cameras.

2.1.1 Relation to Previous Work and Contributions

A review of recent work in human activity analysis can be found in [35]. Based on the conclusions of [35] and a survey of some of recent representative methods on event analysis [6, 32, 37, 46, 67, 70, 84, 89, 95, 98, 101, 103], we find that most existing methods handle events of short duration with moderate changes in illumination, scene clutter and occlusion.

A few techniques have studied the problem of tracking multi-activity sequences using the idea of simultaneous tracking and recognition. In [104], the authors presented methods whereby the identity of a person, based on face recognition, is obtained while tracking the face over the whole sequence. However, the identity of a face in a video sequence is a static parameter which can be estimated by integrating over the entire sequence, whereas activities are inherently dynamic and hence the recognition needs to evolve in time. Simultaneous tracking and recognition was also the theme in [34], but here the authors used color and depth information to create a "plan-view" map based on which tracking is done, and activity recognition was carried out using pose estimates; they did not consider the dynamics inherent in any activity. An integrated approach to tracking and recognition was adopted in [48, 91], though for different kinds of sensors. Video surveillance methods address the problems of tracking and recognition, but usually the tracks are obtained first, followed by recognition [51, 29, 89]. In [63], the idea of integrated tracking and recognition of human activities using particle filters was proposed. Simultaneous tracking of moving persons and recognition of their activities has also been performed in many applications using a Dynamic Bayesian Network (DBN) model [9, 13, 52]. Methods for representing complex activity patterns have been presented using hierarchical hidden Markov models [61] . However, all these methods require a-priori knowledge of the transition probability matrix (or a related function) for switching between different activity models. The method in [41] provides a framework for learning such a transition matrix iteratively, as new data is available. However, our focus is on applications where such a matrix cannot be learned or is not very useful.

In contrast to this open-loop approach, we propose to use change detection measures (e.g., tracking error) to detect transitions between activities, and use them as a *feedback signal* in a system. The main *advantages* of our approach are:
(i) significant computational savings in tracking changes of body posture using the particle

filtering framework,
(ii) obviating the need to learn a complex transition matrix, and
(iii) higher efficiency and accuracy in applications where such a transition matrix is not known.
We show later that this method is significantly more efficient than HMM based methods with uniform transition probabilities (which is the best we can do in situations where more precise information cannot be obtained).

For modeling activities, we use a discrete shape representation which is different from level set representations of shapes such as those described in [38, 75]. The level set approach is theoretically infinite (and in practice large time varying finite) dimensional, and hence defining dynamics on and sampling from such a large dimensional space is computationally expensive. This is overcome through the use of various approximate tracking solutions, as in [38, 75]. Level sets, however, have the advantage that they can adjust to large changes in the shape and topology, which is usually problematic for discrete representations. For large changes in shape, we show that it is possible to overcome this problem for many activity recognition applications by using piecewise stationary dynamical models. Moreover, a discrete representation allows adoption of the overall framework to different descriptions of human body structure (e.g., sampled contours, joint angles, stick figure models, etc.). We describe our approach on the sampled contour (similar to [89]), which can be easily replaced by other features on the image.

2.2 Modeling of Human Activities

We use a composite feature vector comprised of shape, motion and color for tracking. We model human activities by the shape of the contour of the body and its motion (rigid and nonrigid) over time. We model the motion/deformation of a deforming shape as scaled Euclidean motion (i.e., translation, rotation, isotropic scaling) of a "mean shape" plus its non-rigid deformation [97]. The term "shape activity" [89] is used to denote a particular stochastic model for shape deformation. We define a "stationary shape activity" (SSA) as one for which the mean shape remains constant with time and the deformation model is stationary. We use a piecewise stationary shape activity (PSSA) model [88] to represent a shape activity with slowly varying "mean shape" (approximated as piecewise constant).

Fig. 2.2 shows an example of an activity represented using the PSSA model. The overall approach to model human activities is similar to Active Shape Models [10] with the shape being defined in Kendall's shape space. Also, other models for describing human activities, like articulated structures [85, 94], can be easily adopted. However this is not a focus of our work. Our main contribution is in developing a framework for tracking multi-activity sequences without the need for knowing transition probabilities. We adopt this particular activity representation to demonstrate the effectiveness of the method.

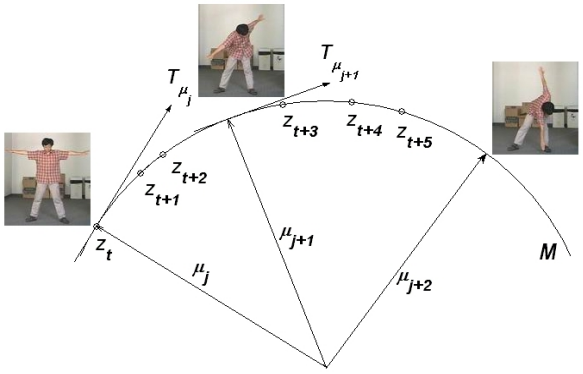


Figure 2.2: Piecewise-Stationary Shape Sequences on the shape manifold which is depicted using a circle ($\mathcal{M}$), instead of a complex $\mathcal{C}^{k-1}$ sphere. Symbols are defined in the text.

2.2.1 Discrete Shape-Based Representation

We use a discrete representation of the shape, z_t^k, for a group of k landmarks on the body contour at time t. Since the shape z_t^k lies on the complex spherical manifold $\mathcal{C}^{k-1}$ [14], we denote its tangent coordinates at a mean shape μ as $v_t(z_t, \mu) = [I_k - \mu\mu^T]z_t$, where I_k is identity (we drop k for notational simplicity) [14]. The tangent space is a linearized local approximation of shape space at μ and Euclidean distance in tangent space is a good approximation to geodesic distance on the shape manifold for points in the vicinity of μ. Let s_t, θ_t, a_t, b_t denote the isotropic scale, rotation, and x and y translation, respectively, in normalizing from the collection of landmarks to the shape space. Let us now define a state vector $X_t = [v_t, s_t, \theta_t, a_t, b_t]$. The configuration of landmark points is related to the shape as $h(X_t) = z_t s_t e^{j\theta_t} + (a_t + jb_t)1_k$, where 1_k is a k dimensional vector with all 1s.

2.2.2 Dynamical Model of Human Activities

The shape dynamics consist of the deformation and global motion of the body contour. Let the mean shape change times be $t_{\mu_1}, t_{\mu_2}, t_{\mu_3}, ...$ (see Fig. 2.2) and the corresponding means be $\mu_1, \mu_2, \mu_3, ...$ Then we have the following dynamics for the *deformation*: between $t_{\mu_{j-1}} \leq t < t_{\mu_j}$, $\mu_t = \mu_{j-1}$, the dynamics is modeled by a stationary Gauss-Markov process, i.e.,

$$v_t(z_t, \mu_{j-1}) = A_{v,j-1} v_{t-1}(z_{t-1}, \mu_{j-1}) + n_t, n_t \sim \mathcal{N}(0, \Sigma_n), \tag{2.1}$$

where $A_{v,j-1}$ models the dynamics of the tangent projection, v, of the shape around μ_{j-1}, and n_t is Gaussian noise with zero mean and covariance matrix Σ_n. At the change time instant, $t = t_{\mu_j}$, $\mu_t = \mu_j$ and so the tangent coordinate v_t needs to be recalculated in the new tangent space, as [88]

$$v_t(z_t, \mu_j) = [I_k - \mu_j {\mu_j}^T] z_t e^{j\theta_t(z_t, \mu_j)}. \tag{2.2}$$

To represent the Euclidean motion of the mean shape, we use a Gauss-Markov model for log-scale, $\log s_t$, and a Markov uniform model for θ_t, as follows:

$$\begin{aligned} \log s_t &= \alpha_s \log s_{t-1} + (1 - \alpha_s)\mu_s + n_{s,t}, \\ \log s_0 &\sim \mathcal{N}(\mu_s, \sigma_s^2), \quad n_{s,t} \sim \mathcal{N}(0, \sigma_r^2) \\ \theta_t &= \alpha_\theta \theta_{t-1} + n_{\theta,t}, \quad n_{\theta,t} \sim Unif(-a, a) \end{aligned} \tag{2.3}$$

which is adopted directly from [89].

2.2.3 Training

Given a training sequence of centered (translation normalized) configurations, $\{Y_t\}_{t=1}^T$, for a particular stationary activity sub-model , the mean shape μ_{j-1} is estimated by Procrustes mean [14] of $Y_t, t_{\mu_{j-1}} \leq t < t_{\mu_j}$. The scale s_t is the Euclidean norm of Y_t. The rotation angle is given by $\theta_t(Y_t, \mu_{j-1}) = -angle((Y_t/s_t)^T \mu_{j-1})$. The shape, z_t, which is defined on configuration, Y_t, and mean shape, μ_{j-1}, is $z_t(Y_t, \mu_{j-1}) = (Y_t/s_t)e^{j\theta_t}$. Then

the tangent coordinates of the shape is [14]

$$v_t(z_t, \mu_{j-1}) \quad = \quad [I_k - \mu_{j-1}\mu_{j-1}^T]z_t. \tag{2.4}$$

Assuming a time invariant Markov model on v_t, we can use $\{v_t\}_{t=1}^T$ to learn its parameters as [69]

$$\begin{aligned} A_v &= R_v(1)\Sigma_v^{-1} \quad where \\ \Sigma_v &= \frac{1}{T}\sum_{t=1}^{T} v_t v_t^T \quad and \quad R_v(1) = \frac{1}{T-1}\sum_{t=2}^{T} v_t v_{t-1}^T \\ \Sigma_n &= \frac{1}{T}\sum_{t=1}^{T}(v_t - A_v v_{t-1})(v_t - A_v v_{t-1})^T. \end{aligned} \tag{2.5}$$

Note that this is done for each stationary activity sub-model.

2.3 Observation Model for Tracking

The observation process is defined by the likelihood distribution, $p(I_t|X_t)$, where X_t is the state vector and I_t is the image observation at t. Our observation models were generated by combining a model of shape with a model of the appearance.

Our shape observation is the edge image, G_t, of I_t. The observation likelihood describes the probability of a set of landmark points, $\Gamma_t \subset G_t$, given the predicted state vector, $X_t = [v_t, s_t, \theta_t, a_t, b_t]$. Let $\hat{Y}_t = h(X_t) = s_t z_t e^{-j\theta_t}$ be the predicted configuration of landmarks based on the dynamical model. It is assumed that a mapping, f, is known that associates each predicted landmark of $\hat{Y}_t$ with a point on the edges. In practice this mapping is set up by searching for the closest edge along the normal of the predicted configuration and this is treated as the observed landmark, Γ_t. Thus the shape observation likelihood is

$$p_s(\Gamma_t|X_t) = \gamma_s \exp\{-\sum_{l=1}^{k} ||q_l - f(q_l, G_t)||^2\}, \tag{2.6}$$

where k is the shape vector dimension, q_l is the l^{th} predicted landmark, i.e., $q_l = \hat{Y}_{t,l}$, $f(q_l, G_t) = \Gamma_t$ is the nearest edge point of q_l along its norm direction (i.e. the normal at q_l to the curve connecting q_l and its neighbor landmarks) and γ_s is a normalizing factor.

To build a color appearance model we sample the color information from the input image I_t over the region covered by the predicted configuration $\hat{Y}_t$ and get the normalized color image [19]. For each predicted landmark, l, of $\hat{Y}_t$, we sample a set of a certain number of pixels, $\{o_n, n = 1...N\}_l$, in an elliptical region whose major axis is aligned with the local normal at l. The color histogram associated with each landmark of $\hat{Y}_t$, ch_l, is calculated over $\{o_n, n = 1...N\}_l$.The appearance observation likelihood is defined as

$$p_a(I_t|X_t) = \gamma_a \exp\{-\sum_{l=1}^{k} ||B(ch_l, CH_l)||^2\}, \tag{2.7}$$

where k is the shape vector dimension, $B(.)$ is the Bhattachayya distance between two color histograms, CH_l is the color histogram associated with the l^{th} landmark of Y_0 (the initialized configuration generated from object of interest in the first frame) and γ_a is a normalizing factor.

The combined observation likelihood is the product of shape and appearance observation likelihoods as

$$p(I_t|X_t) = p_s(\Gamma_t|X_t)p_a(I_t|X_t). \tag{2.8}$$

2.4 Tracking and Change Detection

subsectionTracking using Auxiliary Particle Filters We use an auxiliary particle filter (APF) for tracking [71]. Adaptive particle filters [93, 105] may also be adopted to improve tracking accuracy. The APF with sampling/importance resampling was introduced by Pitt and Shephard [71] as a variant of the standard particle filter to improve sample efficiency. An index K for each particle (called the branch index [30]) is introduced as an auxiliary variable. The sampling is performed in two steps. First draw branch index K according to the marginal auxiliary variable weights $\hat{w}_k = Pr(K = k|I_{1:t})$. Then draw X_t from $p(X_t|X_{t-1}^{(k)})$, the prior density based on the branch indices k. The joint sample set is $\{X_t^{(j)}, K^{(j)}\}$ and the corresponding weights are updated by $w_t^{(j)} \propto w_{t-1}^{K^{(j)}} \frac{Pr(I_t|X_t)Pr(X_t|X_{t-1}^{K^{(j)}})}{Pr(X_t,K^{(j)}|I_t)}$. These are then used to get the filtering (posterior) distribution, $Pr(X_t, K = k|I_{1:t})$.

2.4.1 Detecting Transition Points Between Activities

The activity changes within the course of a long video sequence will cause the PF tuned to the dynamical model of a particular activity to lose track when the activity changes. Thus the tracking error will increase when the activity changes and this can be used to detect the change times. The tracking error (TE) [4] or prediction error is the distance between the current observation and its prediction based on past observations. In our observation model, TE of tracked configuration $\tilde{Y}_t$ is calculated by

$$\begin{aligned} TE(\tilde{Y}_t, I_t) &= TE_{color}(\tilde{Y}_t, I_t) \bullet TE_{shape}(\tilde{Y}_t, I_t), & (2.9) \\ &\triangleq \sum_{l=1}^{k} ||B(ch_l, CH_l)||^2 \cdot \sum_{l=1}^{k} ||q_l - f(q_l, G_t)||^2. & (2.10) \end{aligned}$$

We can also use expected log likelihood (ELL) as described in [81].

2.4.2 Reinitialization to New Activity Model

Once a change has been detected, the next problem is to determine the correct activity from the class of previously learned activity models (i.e., the activity template and its associated dynamics). This is known as the problem of *model switching*, and leads to automatic reinitializaion. Denoting the switching time instance as t_s, and the tracked configuration as $\tilde{Y}_{t_s}$ (which has lost tracking), our reinitialization process is performed automatically as follows:

1. Choose the image at a time before t_s, when TE (or ELL) is low. Let $\tilde{Y}_{ref}$ denote the tracked configuration of this frame. The contour obtained from this configuration $\tilde{Y}_{ref}$ is used as a reference template, $Temp_{ref}$.

2. On the image at time t_s, I_{t_s}, choose a local region around $\tilde{Y}_{t_s}$. Compute the best match of $Temp_{ref}$ in the region, and denote it as $\bar{I}_{t_s} (\subset I_{t_s})$.

3. Use each learned dynamical model in (2.1) and its learned parameters to track starting from $\bar{I}_{t_s}$ until a certain period of time T is reached. This period should be long enough to contain at least half cycle of the activity. (In our experiments, we choose it to be 10 frames.) Denote the tracking results as $\tilde{Y}^m_{t_s:t_s+T}, m = 1, ..., M$, where M is the number of all candidate models.

4. Switch to the model m if $\frac{1}{T+1} \sum_{i=1}^{T} TE_{shape}(\tilde{Y}^m_{t_s+i}, I_{t_s+i})$ is the minimum over

all M activities.

This process can automatically reinitialize in a changing background, when the change in the activity is gradual (which is the case in most videos). Note that by using (2.1) we are incorporating the system matrix, A, in the tracking, change detection and reinitialization phases.

2.4.3 Overall Tracking and Change Detection Algorithm

We now outline the main steps of the tracking and change detection algorithm. For simplicity, let us assume that there are two activities in the sequence, A_1 and A_2. Note that A_1 and A_2 may represent different portions of the same activity - specifically, for those activities in which a non-stationary dynamical model is needed (see Fig. 2.2). For the first frame in A_1, the region of interest is detected based on the application requirements and the corresponding model for the activity is determined as in Section 2.4.2. After this initialization, the algorithm now proceeds as follows.

Track Based on the detected region and the chosen dynamical model, the particle filter is used to track the activity. Measures for determining the accuracy of the tracking algorithm (TE or ELL) are computed for each frame.

Change Detection When the TE or ELL exceeds a threshold for a few consecutive frames, a change is detected.

Model Switching The procedure in Sec. 2.4.2 is used to automatically reinitialize and go back to track.

2.5 Experimental Results

2.5.1 Performance Analysis on Controlled Data

In order to compare with the approaches that assume a known transition probability (e.g., [63, 9, 13, 52]), we use our shape-dynamical representation, but incorporate the transition matrix instead of the switching module in the feedback path. We discuss the comparison on a video consisting of several Yoga postures with continuous changes from one posture to another.

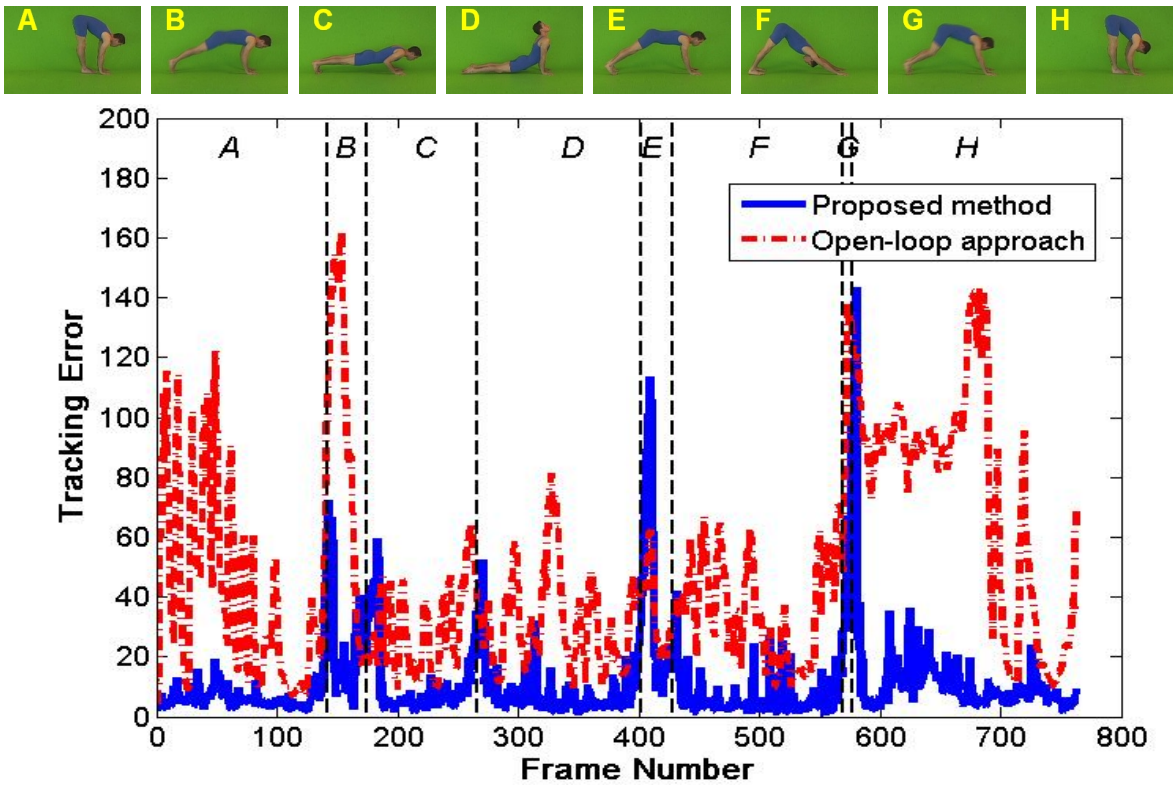


Figure 2.3: Tracking errors of this Yoga sequence using our proposed approach and an approach like [63] using the same number of particles, and with uniform transition probabilities . It is clearly shown that the latter needs much more time to regain track, where the same dynamical models for describing the activities were used and chosen automatically. The actual switching time instances are shown by black dot lines, and the postures for each time interval are shown on the top. The labels A-H show the different postures.

When the accurate transition matrix is unknown, it is reasonable to assume uniform transition probability between activity models. Using the particle filtering framework with a small number of particles, our proposed approach can track through large changes in body posture without knowing the transition probabilities, while an approach like [63] using the same number of particles needs much more time to regain track. Fig. 2.3 shows the tracking errors with both these approaches. In the open-loop framework, the particles are uniformly distributed within several states. When the particle number is small, the number of particles assigned to each state is few, and the correct state can not dominate (we have observed this phenomena when we use less than 50 particles). However, in our approach, all the particles are assigned to the detected state and hence tracking is regained

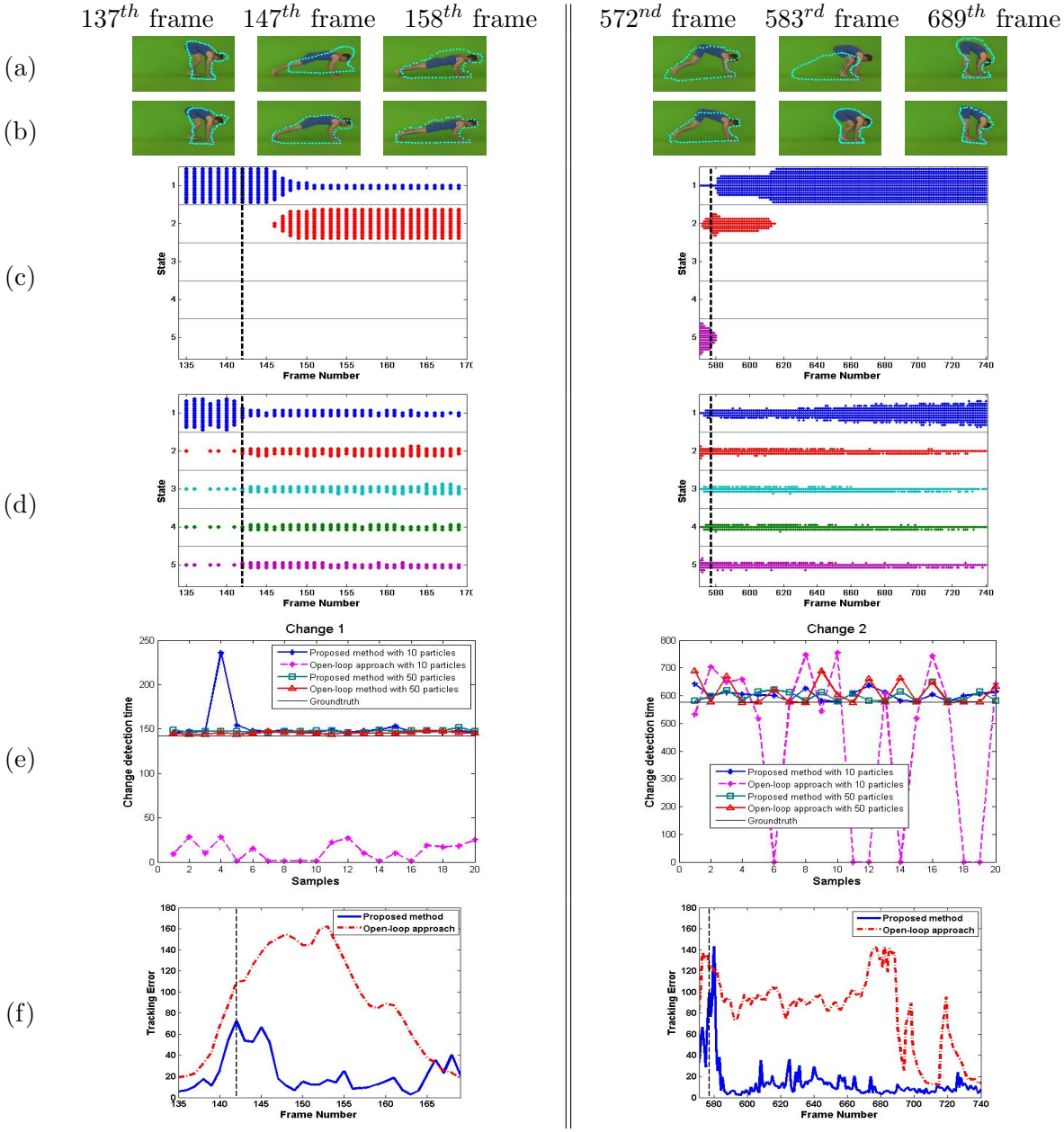


Figure 2.4: (a)-(b) An example of tracking results on a video sequence: (a) Open-loop approach with uniform transition probability (similar to [63]), (b) proposed approach which can track without knowing transition probabilities. Open-loop approach gets back to track after 21 and 117 frames, while our approach is able to maintain tracking continuously. (c)-(d) Average particle distribution on the Yoga sequence: (c) our method, (d) open-loop approach with uniform transition probabilities . (e) Samples of detected change instants. (f) Tracking error plots using the two approaches.

as soon as the correct detection is made.

In order to show the statistics of detected change time instances, we repeat the experiment 20 times for each fixed number of particles. The average detected change instances are listed in Table 2.1. Fig. 2.4 (c)-(d) shows the average distribution of particles in these different scenarios and Fig. 2.4 (e) shows some samples of the change times. The change time instances with the closed-loop approach are close to the ground truth even with a small number of particles. The results show that the closed-loop approach is more accurate especially when the particle number is small. This can be seen for the plots in Fig. 2.3, the tracking results of Fig. 2.4 (a)-(b) and their corresponding TE plots in Fig. 2.4 (f). The Yoga sequence is a very clean data set; the advantages on more noisy data will be seen even if we use a much larger number of particles.

Table 2.1: Average detected change time instances

number of particles	10	15	20	25	30	35	40	45	50	ground truth
closed-loop change 1	152.3	155.3	157.8	148.4	156.3	147.5	159.3	147.1	147.4	142
open-loop change 1	12.5	37.3	106.3	119.5	140.4	140.2	145.1	145	145.5	142
closed-loop change 2	602.6	589.6	595.7	596.4	604.4	602.7	596.9	567.5	598.8	577
open-loop change 2	438.7	492.7	546.6	542.7	578.5	610.7	607.3	595.7	610.4	577

2.5.2 Real-life experiments

We now show some results with video captured in outdoor environments under different conditions. We have three sequences. The first consists of activities around a car (Fig. 2.5(a)). The second is of two people who exchange a box between them in a complex changing background (Fig. 2.5(b)). The third consists of multiple people doing different activities simultaneously (Fig. 2.5(c)). There are a total of eight activities as described in Fig. 2.5.

2.5.3 Discussion of the Results

The output of our closed-loop system is the tracking result, which is equivalent to the tracking error, i.e., the difference between tracking result and observations. The tracking error is calculated as the distance of estimated landmarks to the nearest edge points obtained from the observations. Since the objects in any tracking problem must be

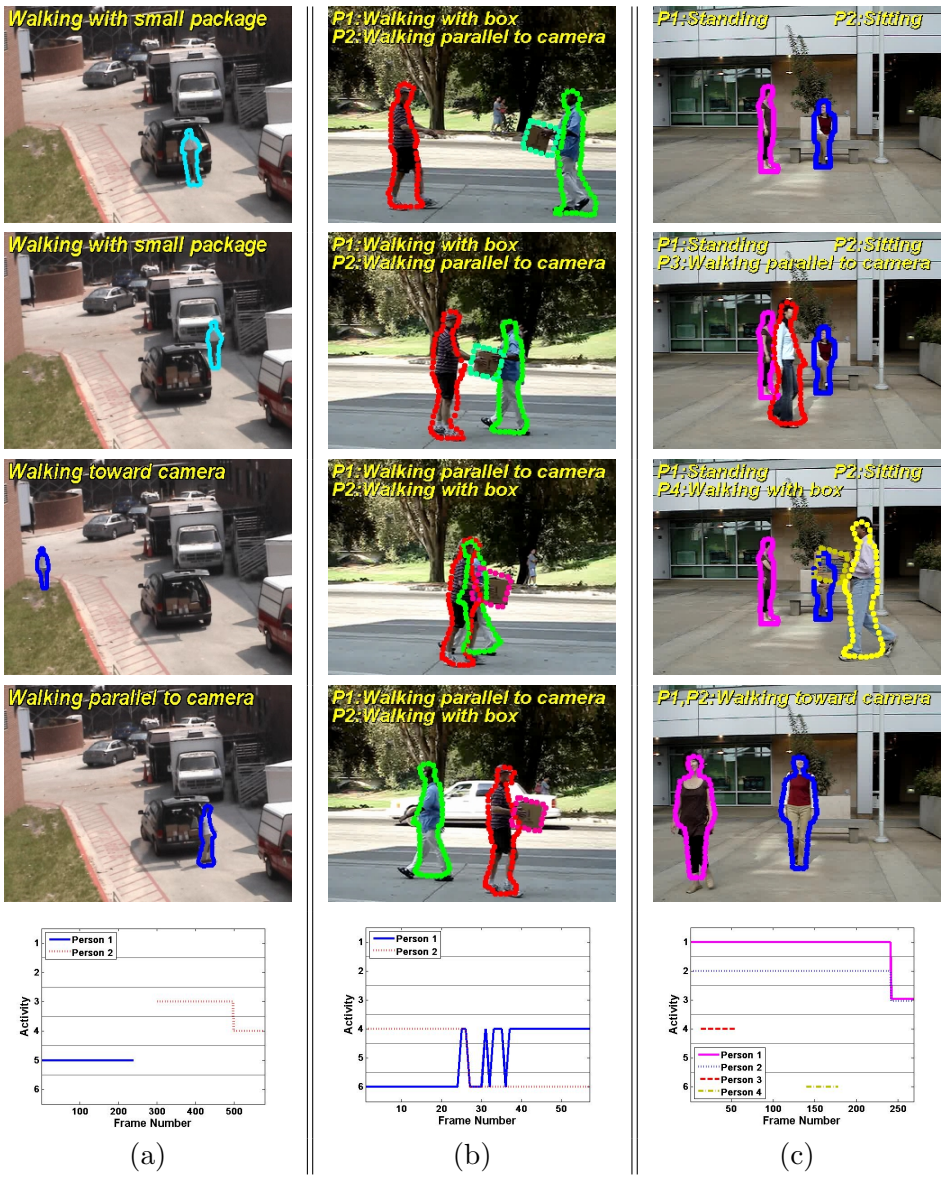


Figure 2.5: Row 1-4: Tracking results with the activity models used to track shown on the top. Row 5: Model switching results. The labels denote: 1: Standing, 2: Sitting, 3: Walking toward camera, 4: Walking parallel to camera, 5: Walking with small package, 6: Walking with box, 7: Walking with backpack, 8: Walking with briefcase. (a):Tracking results on an outdoor sequence. (b): Two person walking towards each other, one person holding a box then giving it to the other person. At the first frame, the activity of person 1 is initialized as "walking with a box", then we split box from person and track them separately. (c): Multiple people doing different activities simultaneously. (The figure is best viewed on a monitor.)

within the range of the image frame (i.e., within view of the camera), the observations are finite. The convergence constraint on the dynamical model in equation (2.1) ($A_{v,j-1}$ is a constant matrix with the magnitudes of its eigenvalues all smaller than one) gives finite estimates of the landmarks. Therefore, the tracking error is finite. If we now make the assumption that the observed activity belongs to one of the learned models, one particular model will have the minimum tracking error, which will be finite, and hence, the overall system will be stable. Moreover, we could show that under a set of reasonable assumptions, the tracking error with a correct model is less than that with an incorrect model (see proof in Appendix A).

2.6 Conclusions

In this chapter, we proposed a novel framework for tracking multi-activity sequences without knowing the transition probabilities between the activities. We demonstrated this for applications using only one camera, as well as in a network of non-overlapping cameras. Using a non-linear dynamical feedback system, we showed that our method is able to continuously track sequences of different activities amidst changing background, occlusions and clutter. Measures were designed to detect changes between activities. We demonstrated the effectiveness of our approach by doing a detailed performance analysis and showing experimental results on real life videos of different activities.

Chapter 3

Tracking in a Non-overlapping Camera Network - A Multi-Objective Optimization Framework

3.1 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 summarize 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 existing challenges in tracking objects like pose and illumination variations, occlusion, clutter and sensor noise. The problem is important for applications in wide area surveillance, digital forensics and video summarization, among others.

In this chapter, we present a robust framework for tracking multiple objects (people) in a *non-overlapping* camera network whereby feature correspondences are modeled statistically and adapted in time by considering the long-term dependencies between them. We pose the problem of tracking in a camera network as a multi-objective optimization

problem. The first objective is to choose the tracks such that they maximize the similarity between the observed features in a local neighborhood. This has been at the core of the approach taken in methods like [39]. This can be solved very efficiently in a dynamic programming framework. However, the method relies on being able to estimate the similarities between features observed at different cameras reliably. This is often not possible, due to the well-known problems in video analysis related to changes in lighting and pose, scene clutter and occlusions. To make the method robust to such errors, we introduce a second optimization objective that minimizes the *long-term* variation in appearance and identity of the tracked objects over the entire camera network. This allows us to correct for errors in tracking based only on the first objective by considering the tracking results obtained over time. This is similar to the reasoning process of a human observer, who on noticing an error in the tracks, may go back, try to locate the source of the error and recompute the tracks.

We show how to design these two objective functions precisely in a stochastic optimization framework. The similarities between the features are modeled as random variables with their probability distributions learned from training data. To model long-term variation in appearance and identity, we propose a novel criterion that is based on discriminant analysis methods [16]. To solve the overall multi-objective optimization problem, we propose a greedy heuristic solution and show that it usually leads to the correct result. This heuristic solution is a stochastic adaptive strategy that finds the optimal tracks by considering the short- and long-term dependencies in the data. A stochastic adaptive system, well-known in the control literature, is a stochastic system some of whose parameters are unknown and need to be learned during the system's operation. 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. We conduct a detailed performance analysis with data captured on practical multi-camera systems with multiple people observed over the network.

We would like to clearly differentiate our work with methods that deal with tracking objects in a network of *overlapping* cameras (e.g., [31]). We concentrate on the problem of non-overlapping cameras. If some of them were overlapping, methods that deal with this scenario can be easily incorporated into the similarity estimation methods, thus making them more reliable.

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. The implementation strategy will be explained in the different sections of the chapter as we describe the details. The amount of data that is transmitted is significantly less than if all the cameras simply transmitted their data (even after compression) to the central processor. We will provide precise numbers in Section 3.6.3.

3.1.1 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 [55, 56, 82, 62]. Based on existing work in this area, 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. The similarity estimation 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 [19] and the travel time between two nodes. In future, we plan to incorporate identity features, e.g., face or gait biometrics.

Some of the existing methods on tracking in a camera network include [36, 39, 44]. The authors in [74] 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 [49], a particle filter was used to switch between track prediction between non-overlapping cameras and tracking within a camera. In [43], 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 [44] using color histograms for object appearance. A graph-theoretic framework for addressing the problem of tracking in a network of cameras was presented in [39].

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.

3.2 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 zone [55, 62] (a small image region; see markers in Fig. 3.13 for examples). We assume that we know which camera each node can be viewed from (also referred to as a node belonging to a camera). Mathematically, each node is represented as n_i^c, where the subscript is a node index (node index is unique regardless of the camera it belongs to) and the superscript represents the camera it belongs to. When we mention local processing at each node, we mean local computations at the camera to which that node belongs. For the purposes of this chapter, 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.

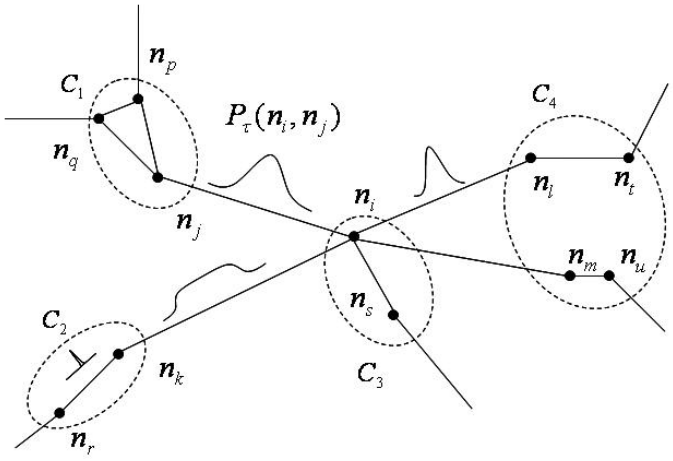


Figure 3.1: An example of a camera network, node n_i is linked to nodes (n_j, n_k, n_l, n_m) and a distribution of the travel time between them is known.

A network architecture [55, 62] 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, $l_{ij} = 1$ 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 belonging

to different cameras, i.e., $P_\tau(n_i^c, n_j^d), i \neq j, c \neq d$, where $P : \Re^+ \rightarrow [0, 1]$. One of these nodes must be an entry node and the other an exit node. Travel time distribution between nodes belonging to the same camera are not needed, as they can be viewed completely and transition between them tracked in the image plane.

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 chapter is explained in Section 3.3.1. 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 Fig. 3.1, 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.

3.2.1 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 estimates the similarity of this feature vector with its stored feature vectors, $\mathcal{F}_{i,t}$, obtained from neighboring nodes. The exact procedure for similarity estimation is explained in Section 3.3.1. This is then transformed into a distribution on the similarity between two features. The distribution captures the uncertainty in estimating 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 is explained in Section 3.3.2. This similarity estimation 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 (see Fig. 3.2 for an example). 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 Fig. 3.2. This

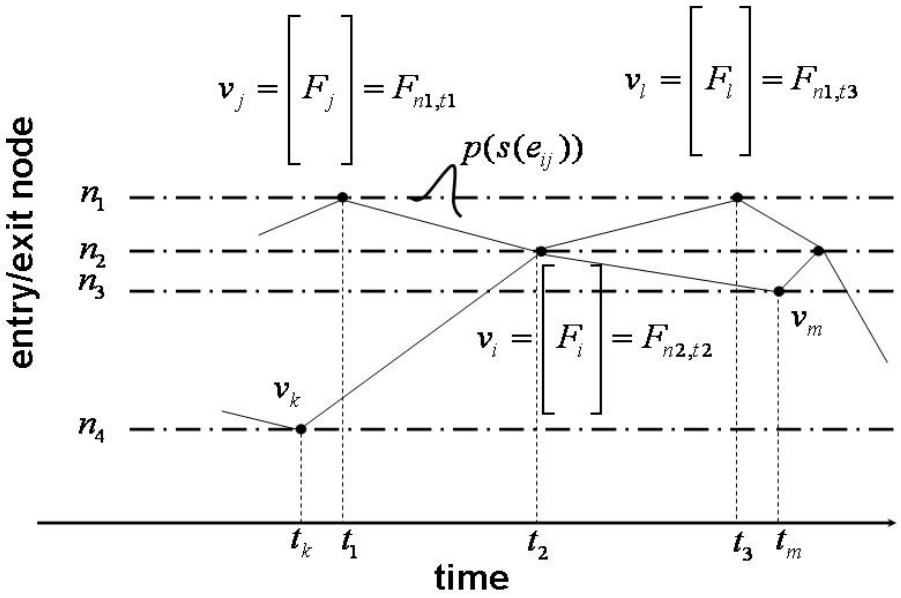


Figure 3.2: Illustration of feature graph construction. Note the subscripts of F indicating the time and entry/exit node where it is observed.

distribution can be learned as explained in Section 3.3.2. The tracks of each person can be found by estimating the optimal paths in this graph. We show how this can be done using stochastic weights in Section 3.3.3.

This optimal path estimation 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 estimations in the feature graph are correct. This can be a big assumption due 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. Thus, the optimal path estimation requires the simultaneous optimization of the local similarities between features along the track for each person and the long-term distribution of the features along that path. This is a *multi-objective optimization* problem. Some trade-off between the criteria is needed to ensure a satisfactory solution. To keep the computational requirements within realistic limits, we propose an adaptive strategy to find a solution of this multi-objective optimization problem. The design of the objective functions and optimization strategy are explained in the Section 3.3 - 3.5.

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

3.3 Computing Optimality in Stochastic Feature Graphs - Objective 1

Consider the above-mentioned feature graph G described in Fig. 3.2 of Section 3.2. The problem of tracking people across cameras is equivalent to finding the most preferred links between vertices in this graph. We will use the term node to refer to an entry/exit point and vertex to refer to an observed feature vector in the graph G.

3.3.1 Feature Similarity Scores

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

$$\mathbf{F}_{n,t} = \Big(F_A\Big)_{n,t} \tag{3.1}$$

where F_A is the appearance feature (normalized color [19] in our case). 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 $\mathrm{S}_\tau(\tau_{n_i,n_j}^{t_1,t_2})$. It is reasonable to assume that F_A 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 $F_i, F_{A,i}$ and $\tau_{i,j}$.

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

$$s(e_{ij}) = \mathrm{S}_A(F_{A,i}, F_{A,j})\mathrm{S}_\tau(\tau_{i,j}). \tag{3.2}$$

The similarities can be obtained by computing the distance between the feature vectors in appropriate feature spaces as described in [19]. Known geometric and photometric transformations between two corresponding features can be taken into account while computing this distance. Examples include affine warping and brightness transfer functions [40, 42]. These can be learned during the training phase for the network topology. We did not need to use any of the transformation in our experiments. We will provide specific examples of the features and similarity estimation in Section 3.6.

3.3.2 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 estimate 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.

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 estimated. During the system operation, the variance σ^2_{ij} is determined based on which cluster s'_{ij} belongs to. In the experiments, we collected training data separate from the testing. In the learning phase, the correspondence of people in different cameras were assumed to be known and this information was used to estimate the transition time distributions. For experiment setup 1, more than 1000 similarity scores between two vertices were collected. For experiment setup 2, the collected similarity scores were more than 2000. As an example, we show the

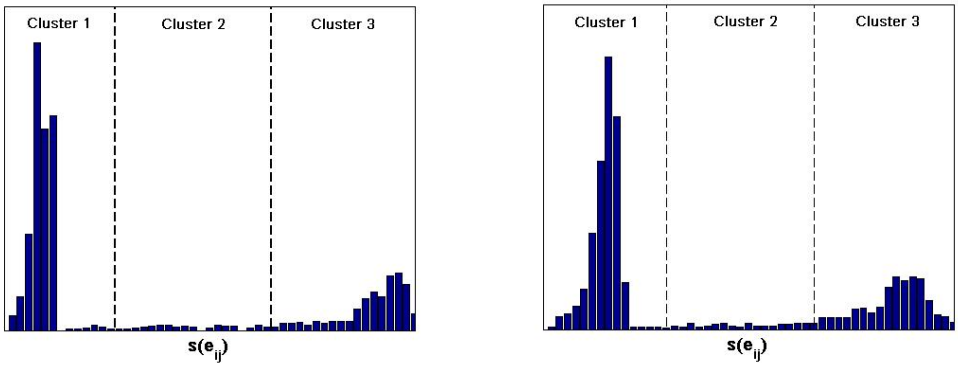


Figure 3.3: Histogram of similarity scores between v_i and v_j learned during the training phase for two node pairs in the real-life networks described in Section 3.6. The left figure is for Data Set 1, while the right is for Data Set 2.

learned histogram of the similarity scores between two vertices in Fig. 3.3. 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 [16].

3.3.3 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 [54]. The most widely used method for this problem was originally proposed by von Neumann and Morgenstern [20], which proposed an utility function and then estimated 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 [54] 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

$$w(s) \stackrel{\triangle}{=} exp(s) - 1. \tag{3.3}$$

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 estimating the EWS (expected utility), both the observed similarity

and its uncertainty are taken into account through the weighting function (utility function) [1] .

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

$$\{\tilde{\lambda}_q\} \stackrel{\triangle}{=} \arg\max_{\lambda_q} \sum_{\lambda_q} \{ \sum_{e_{ij} \in \lambda_q} E_s[w(s(e_{ij}))]\}. \tag{3.4}$$

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 [39], 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.

3.4 Computing Long-term Feature Dependence in A Camera Network - Objective 2

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 [39] 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 estimated 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 Fig. 3.4, if the similarity estimation 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 consider the interdependence of the features and similarity scores

[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.

over space and time. This is a major assumption in the method of [39]. 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. 3.3.3 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 multi-objective optimization based tracking framework, where the second objective function is to measure the similarity of observed features along a path.

3.4.1 Design of Path Smoothness Function

To model the spatio-temporal variation of the observed features along a path, 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

$$m_i = \frac{1}{N_i} \sum_{x\in\{X^{(i)}\}} x. \tag{3.5}$$

Let S_T be the variance of the all observed feature x along the path, i.e.,

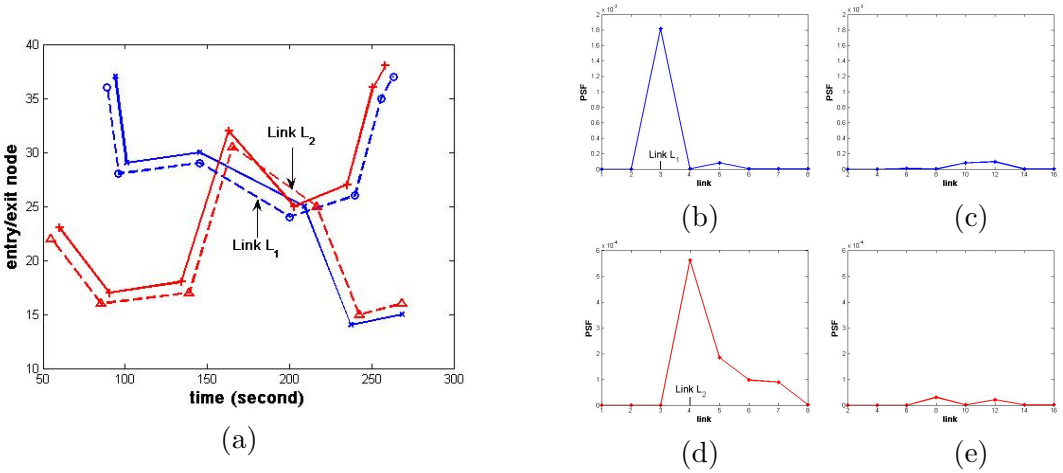
$$S_T = \sum_{x\in\{X\}} |x - m|^2 \tag{3.6}$$

and S_W be the sum of the variances along each sub-path, $\{X^{(1)}\}$ and $\{X^{(2)}\}$, i.e.,

$$S_W = \sum_{i=1}^{2} S_i = \sum_{i=1}^{2} \sum_{x\in\{X^{(i)}\}} |x - m_i|^2. \tag{3.7}$$

The PSF for e_{ij} is defined as

$$PSF(e_{ij}) = \frac{|S_T - S_W|}{|S_W|} = \frac{|S_B|}{|S_W|}. \tag{3.8}$$



(a) (b) (c) (d) (e)

Figure 3.4: (a): Paths of two people (in blue and red) obtained from the camera network in Data Set 1 of Section 3.6.1: 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 the 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 Brightness Transfer Function [40] 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 [2].

Thus the PSF is defined from Fisher's linear discriminant function [16] 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 $e_{ij} \in \lambda_q$ 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. 3.4, where the PSF is plotted for all edges along one correct and one incorrect

[2]The Brightness Transfer Function and an affine warp are possible extensions that may be used in some application scenarios to improve accuracy, but is not used in our experiments.

path obtained from our camera network.

We can now precisely describe the second objective function by analyzing features along a path. In practice, we specify the function in terms of the Path Smoothness Function (PSF). Thus, the second objective function is to minimize

$$H_2(\lambda_q) = \sum_{\lambda_q} Var(PSF(e_{ij} \in \lambda_q^{(n)})). \tag{3.9}$$

3.5 Multi-Objective Optimization for Tracking In a Camera Network

By creating a feature graph, the tracks each person can be found by estimating the optimal paths, λ_q, in this graph. Thus, our first objective function is to maximize

$$H_1(\{\lambda_q\}) = \sum_{\lambda_q} \{ \sum_{e_{ij} \in \lambda_q} E_s[w(s(e_{ij}))] \}. \tag{3.10}$$

The second objective function, just described above, is to minimize

$$H_2(\lambda_q) = \sum_{\lambda_q} Var(PSF(e_{ij} \in \lambda_q^{(n)})). \tag{3.11}$$

Our goal is to find a optimal point λ_q^* of this multi-objective optimization problem such that we maximize $H_1(\lambda_q)$ and minimize $H_2(\lambda_q)$.

3.5.1 Multi-objective Optimization Framework

Many realistic optimization problems require the simultaneous optimization of more than one objective functions, i.e., there is a vector of objectives
$H(x) = [H_1(x), H_2(x), ..., H_m(x)]^T, m \geq 2$ that must be traded off in some way [17]. Note that, because $H(x)$ is a vector, if any of the components of $H(x)$ are competing, there is no unique solution to this problem. Instead, the concept of noninferiority [100], also called Pateto optimality [8, 11] must be used to characterize the objectives. Essentially, a vector $x^* \in \mathcal{C}$ is said to be Pareto optimal for minimizing $H(x)$ if all other vectors $x \in \mathcal{C}$ have a higher value for at least one of the objective functions $H_i(.)$, or else have the same value for all objectives.

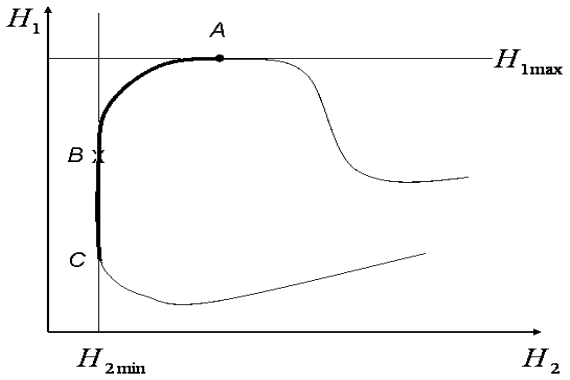


Figure 3.5: Solution space of multi-objective optimization problem.

A conceptual illustration of the multi-objective problem solution space is shown in Fig. 3.5. By maximizing $H_1(\{\lambda_q\})$, we get to the point A. This is determined based on finding the paths in the feature graph which maximize the objective H_1. Under the assumption that the feature similarities are estimated correctly for the most part, A should be close to the correct tracks of each person. By minimizing $H_2(\{\lambda_q\})$, we adapt the similarity scores allowing for errors in their estimation in H_1. This is continued until we reach a local minimum of $H_2(\{\lambda_q\})$, say B. Such a minimum, $H_{2\min}$ exists and is greater than zero based on the similarity measure we defined in Section 3.4, which relies on estimating the variance of the features along each path in a suitable feature space.

3.5.2 A Heuristic Optimization Scheme

Based on the observation that most of the similarity scores are estimated correctly, we propose a heuristic optimization scheme that is computationally efficient and dose not require searching over all possible paths in the feature graph. We initialize the multi-objective optimization with the following set of paths $\tilde{\lambda}_q$ as in (3.4),

$$\{\tilde{\lambda}_q\} \stackrel{\triangle}{=} \arg\max_{\lambda_q} \sum_{\lambda_q} \{ \sum_{e_{ij} \in \lambda_q} E_s[w(s(e_{ij}))]\}.$$

Since it is often reasonable to assume that most of the weights of the feature graph are calculated correctly, the initial point is close to our final desired solution. We now perform an edge weight adaptation scheme, for minimizing H_2 until we reach a local minimum. The adaptation scheme is described below. Note that even if only a few of the weights are

calculated incorrectly, many estimated paths can be incorrect since the effect of a few wrong links will affect many paths.

Whenever there is a peak [3] 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 temporarily set the weight on this link where the peak occurs to be zero, then recalculate the maximum matching paths, λ'_q, of the new feature graph. If $H_2(\{\lambda'_q\}) < H_2(\{\tilde{\lambda}_q\})$, then we accept the new graph, otherwise we decline the changes. We repeat this iteration till a local minimum of $H_2(\{\lambda_q\})$ in (3.9) is reached. This is formally described below.

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 3.3.

2. Estimate the optimal paths, $\tilde{\lambda_q}$ as mentioned in Section 3.3.3.

3. Compute the PSF for each $e_{ij} \in \tilde{\lambda}_q$.

4. Set the weight on the link where the PSF peak occurs to be zero.

5. Recalculate the maximum matching paths, λ'_q, of the new feature graph. If $H_2(\{\lambda'_q\}) < H_2(\{\tilde{\lambda}_q\})$, then we accept the new graph and set $\tilde{\lambda}_q = \lambda'_q$, otherwise we decline the changes.

6. Repeat Steps 4 and 5 until a local minimum of $\sum_{\lambda_q} Var(PSF(e_{ij} \in \tilde{\lambda}_q))$ is reached. The final set of optimal paths is given by $\left(s^*(e_{ij}), \lambda_q^*\right)$ as

$$s^*(e_{ij}) = \arg\min_s \sum_{\lambda_q} Var(PSF(e_{ij} \in \lambda_q(s))), \tag{3.12}$$

$$\lambda_q^* = \arg\max_{\lambda_q} \sum_{\lambda_q} \{ \sum_{e_{ij} \in \lambda_q} E_s[w(s^*(e_{ij}))]\}. \tag{3.13}$$

This strategy generates the correct solution of the multi-objective problem if it can be obtained as the local minimum of a convex function starting with the initial condition in Step 2. In cases when this is not true (a high value of H_2 at the local minimum can be an indication of it), we can initialize with a few of the top values of H_1 and choose the path

[3] The peak is detected if it is above a threshold, which is defined as $E\{PSF(e_{ij} \in \lambda_q)\} + Var(PSF(e_{ij} \in \lambda_q))$.

that minimizes H_2. A more exhaustive search of the solution space may be needed if the initial starting position is very far from the optimal, which should be a very rare occurrence.

3.6 Experimental Results

To evaluate the performance of our system, we will show results on two sets of data collected over large camera networks. This was done using the VideoWeb Lab at UCR which, on completion, will consist of about 80 cameras in indoor and outdoor settings and over wired/wireless channels. The first dataset contains more cameras, but the data is relatively clean. The second set contains slightly fewer cameras, but in an uncontrolled setting.

For people detection and segmentation, we use the methods similar to our previous work for a single camera [80]. Motion in an image is detected and the region is segmented to obtain the features of the person. Currently, in our experiments, we only consider fixed cameras - thus people can be easily extracted by background subtraction. Normalized color histogram [19] of segmented region is calculated as the appearance feature. Similarity scores of each pair of observations are estimated based on appearance and transition time as described in Section 3.3.1. A feature graph is constructed based on similarity scores (see Section 3.2.1) and optimal paths are estimated in this graph (Objective 1, see Section 3.3). Then PSF is computed on each estimated path (Objective 2, see Section 3.4). The feature graph is adapted based on PSF values to get a solution for this multi-objective optimization problem (Section 3.5).

3.6.1 Experiments on Data Set 1

The network consists of 20 cameras and 50 entry/exist nodes. We consider 10 people moving across the network over a period of about 20 minutes. The total video was almost 2GB. Examples of some of the images of the people are shown in Fig. 3.6. The reader should compare this plot with the abstract representation of the feature graph in Fig. 3.2. Some of the features used in the similarity estimation are shown in Fig. 3.7. The feature is the normalized color [19] histogram and the similarity is estimated using the method described in [39].

Fig. 3.8 shows the tracking result using optimal path searching using Objective 1

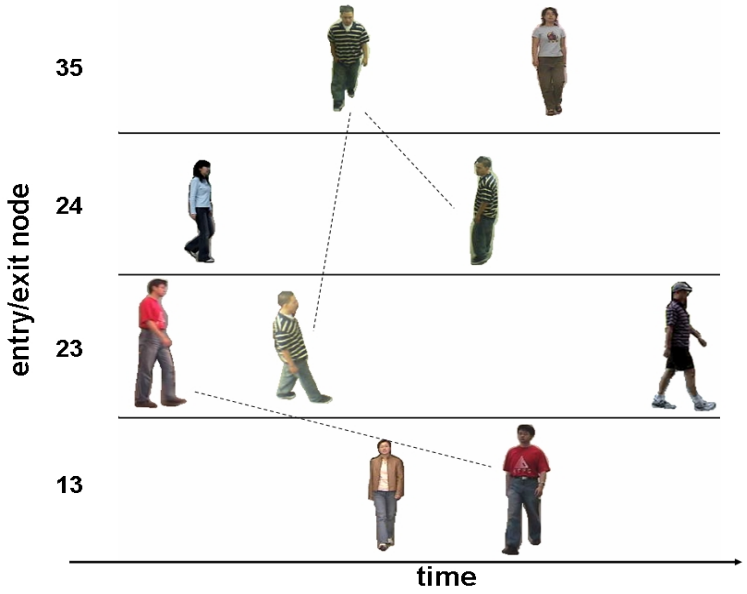


Figure 3.6: Example of some of the images of the people in the network. The horizontal axis is the time when these features were observed, while the vertical axis is the index of the entry/exit node where the features were observed. The gaps in the plot are because the features are observed asynchronously.

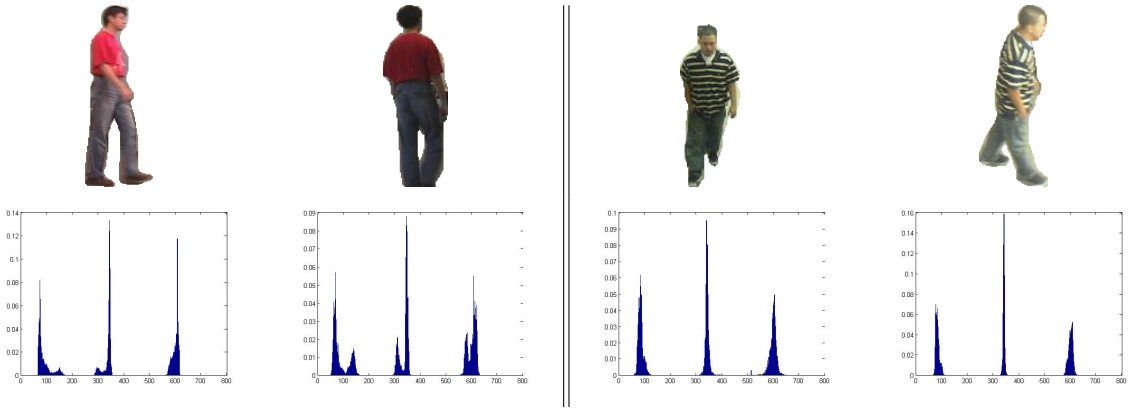


Figure 3.7: Example the features used in the similarity estimation. Images are shown on the top, and the normalized color histograms are shown on the bottom.

only (i.e., there is no adaptation). This is similar to the method in [39]. 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 lines showing tracking results. Thus, for correct results the colors should match across all

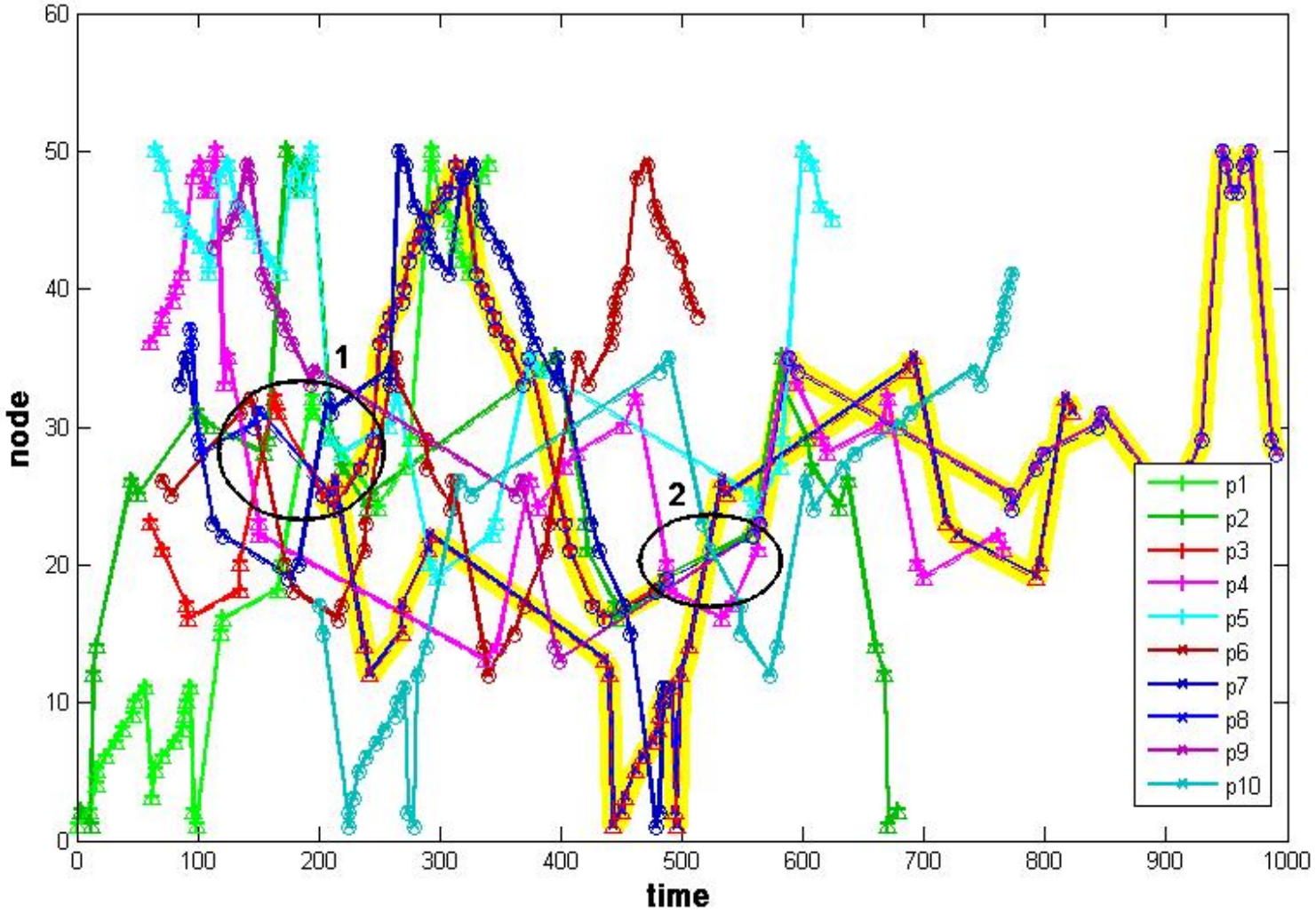


Figure 3.8: Tracking result using Objective 1 only (i.e., there is no adaptation) in feature graph. The paths of different people are shown using different colors, solid lines representing the ground truth and tracking results are showed by dotted lines. The wrong links are circled, while the wrong paths due to these links are highlighted. Details of one of the wrong links shown in the bigger circle are given in Fig. 3.4. The PSF values and their adaptation are also shown. The portions inside the two circles are zoomed in Fig. 3.10. (The figure is best viewed on a color monitor.)

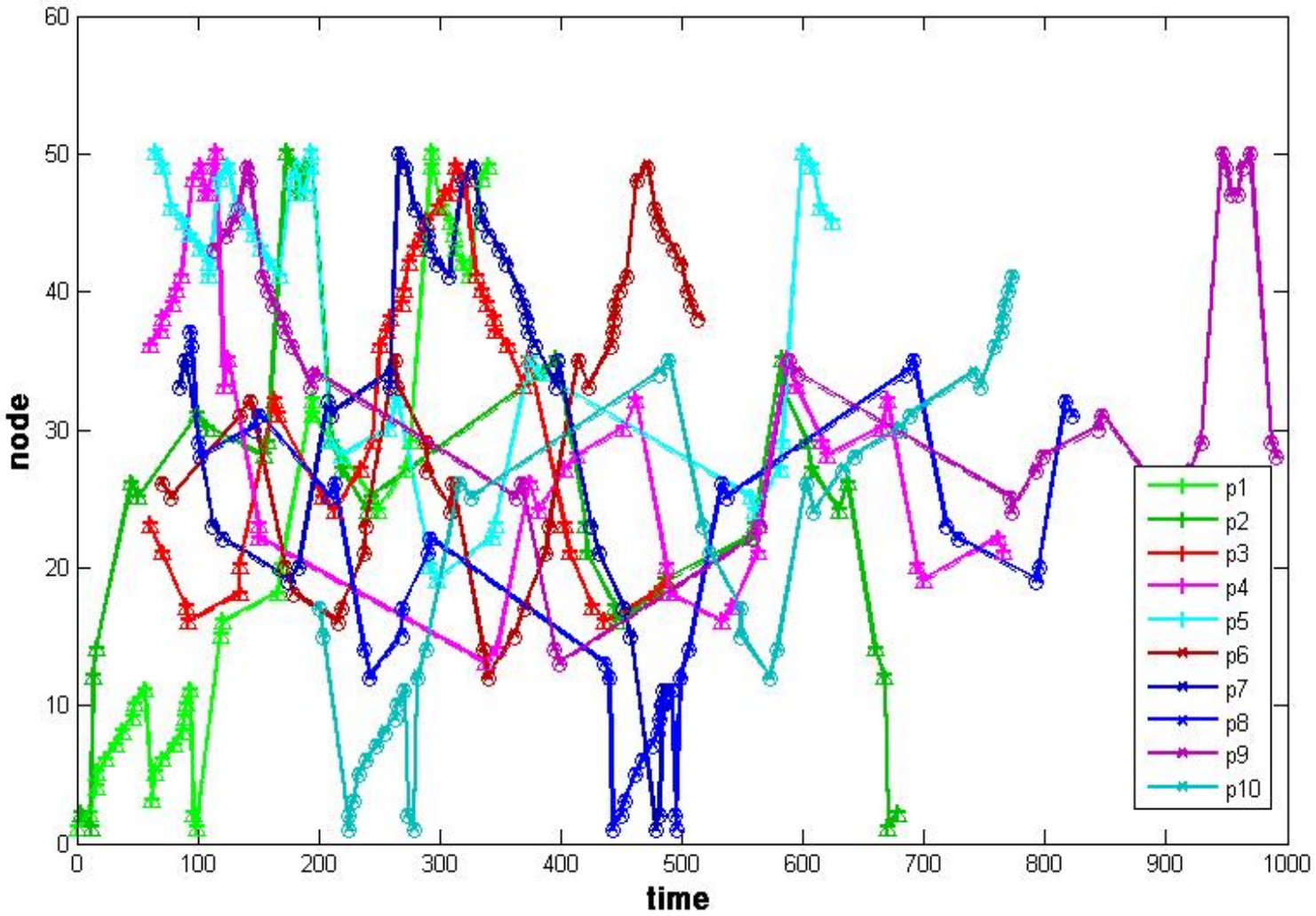


Figure 3.9: Tracking result using our stochastic adaptive algorithm using the same representation as Fig. 3.8. Compare the color of the paths along the highlighted portions of Fig. 3.8. The colors of the dotted and solid lines match indicating correct path estimation. (The figure is best viewed on a color monitor.)

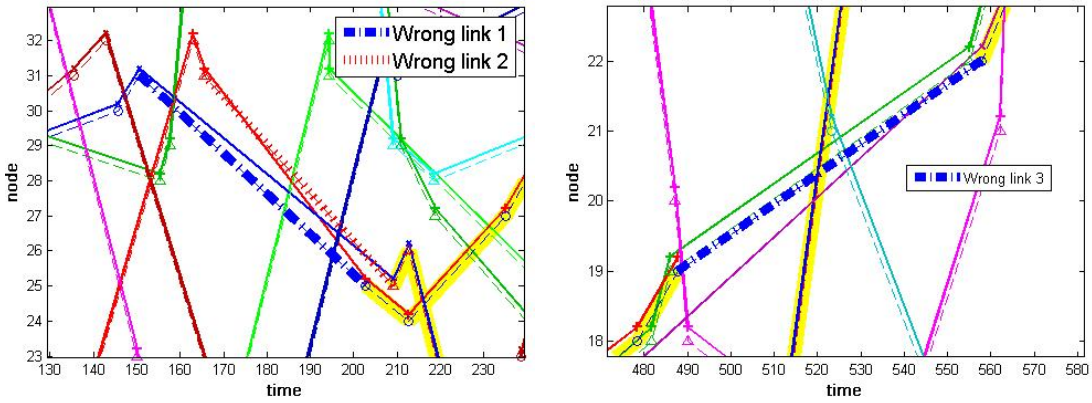


Figure 3.10: The portions inside the two circles in Fig. 3.8 are zoomed-in. The incorrect edges are marked.

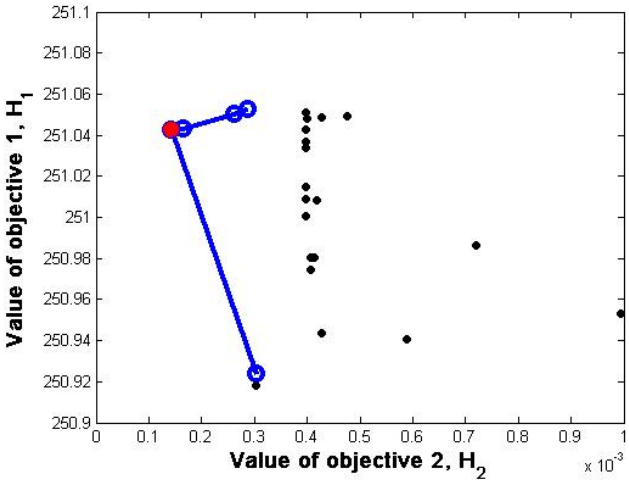


Figure 3.11: A plot of the solution space with some possible values of H_1 and H_2 obtained through a random simulation. The path taken by the optimization strategy is shown in bold, with the final solution marked with a filled circle, which overlaps with the ground truth.

the paths. There are three wrong links (circled in Fig. 3.8 – two in the bigger circle, and one in the smaller circle) due to mistakes in feature similarity estimation. The paths that are estimated wrongly due to these wrong links are highlighted. In Fig. 3.9, we show the correct result by adapting the graph edge weights using the proposed stochastic adaptive strategy. Details of the adaptation of the weights in the wrong link shown in the bigger circle are given in Fig. 3.4. By comparing the colors along the paths highlighted in Fig.

3.8, we can see that these mistakes have been corrected. Both these figures are best viewed on a computer monitor.

We show the zoomed-in portions of the tracks inside the two circles of Fig. 3.8 in Fig. 3.10. In the bigger circle, the two tracks are cross-connected. In the smaller circle, the track of Person 3 is connected to that of Person 9. We plot different points in the solution space of H_1 and H_2 for this dataset and mark the path taken by the local optimization of Section 3.5 in Fig. 3.11. This figure should be compared with the conceptual plot of the multi-objective optimization space in Fig. 3.5. The points in the solution space were obtained through a random simulation.

3.6.2 Experiments on Data Set 2

The network consists of 7 cameras and 26 entry/exist nodes. The cameras are installed in both indoor and outdoor environments which consist of large illumination and appearance changes. We consider 9 people moving across the network for about 7 minutes. The total video was about 200 MB. Examples of some of the images of the people and entry/exit nodes are shown in Fig. 3.13. Note the significant changes of appearance. Some of the appearance features are shown in Fig. 3.12.

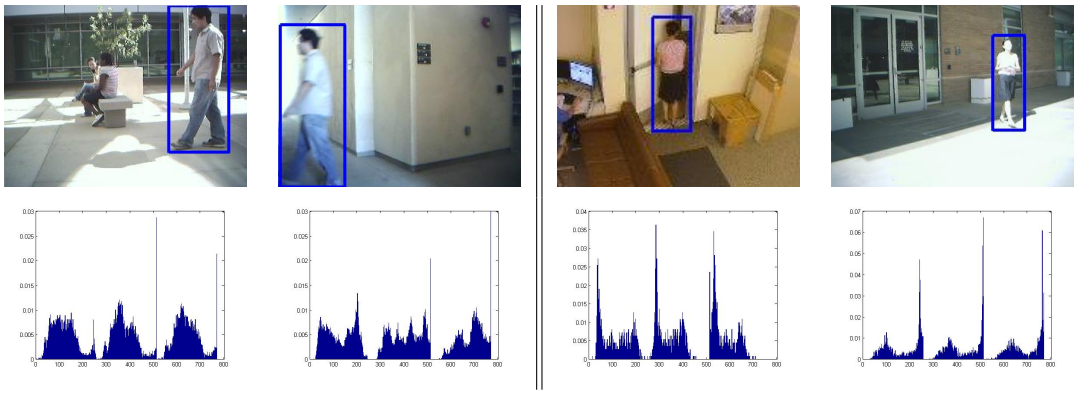


Figure 3.12: Example the features used in the similarity estimation. Images are shown on the top, and the normalized color histograms are shown on the bottom.

Fig. 3.14 shows the tracking result using optimal path searching using Objective 1 only (no adaptation), similar to the method in [39]. 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

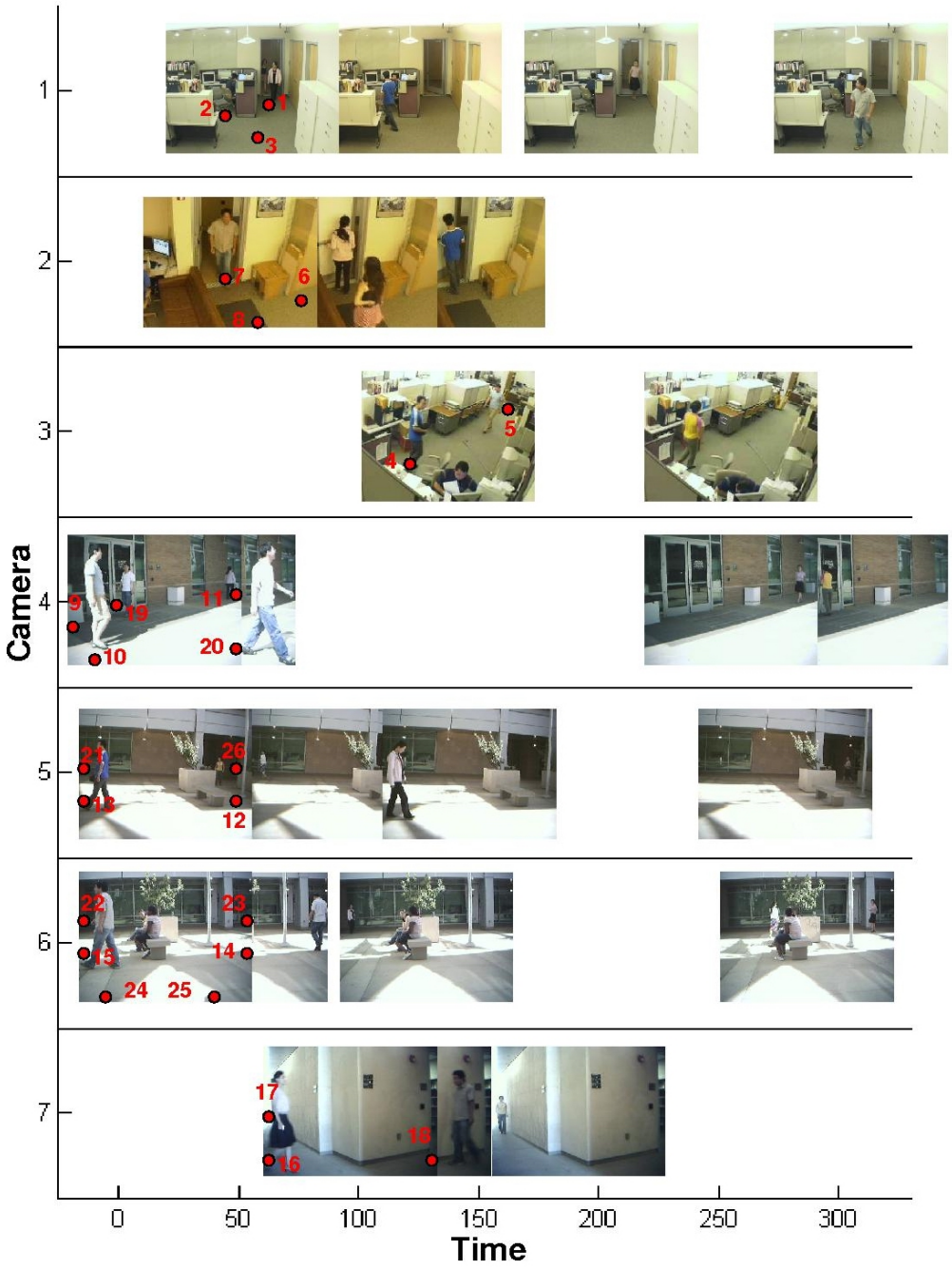


Figure 3.13: Example of some of the images of the people in the network. The horizontal axis is the time when these images were observed, while the vertical axis is the index of the cameras. Some of the entry/exit nodes are marked on the images.

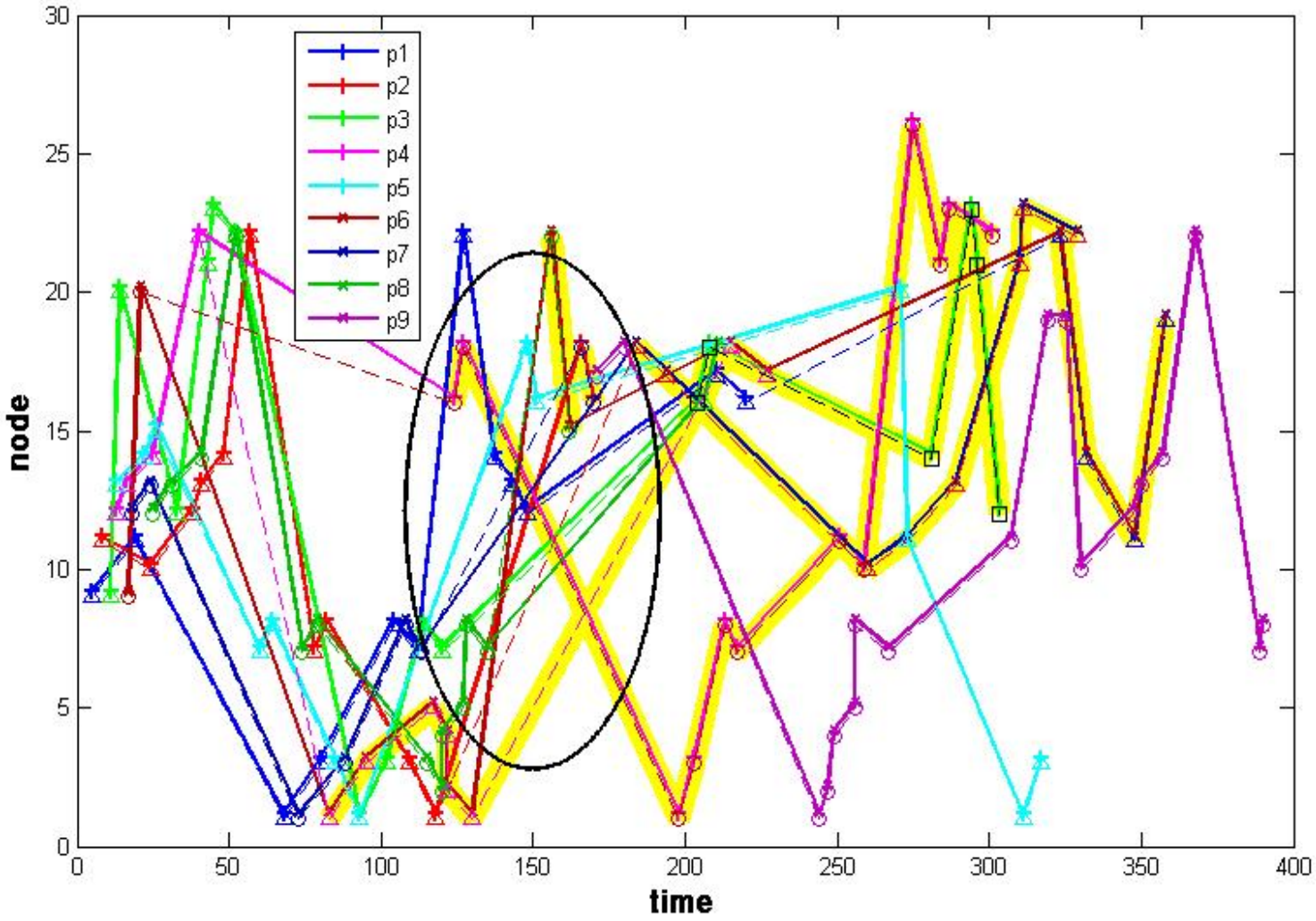


Figure 3.14: Tracking result using Objective 1 only (i.e., there is no adaptation) in feature graph. Horizontal axis is time, while vertical axis is the entry/exit node where a feature is observed. The paths of different people are shown using different colors, solid lines representing the ground truth and tracking results are showed by dotted lines. The wrong paths due to these links are highlighted. Details of the wrong link shown in the bigger circle are given in Fig. 3.16. (The figure is best viewed on a color monitor.)

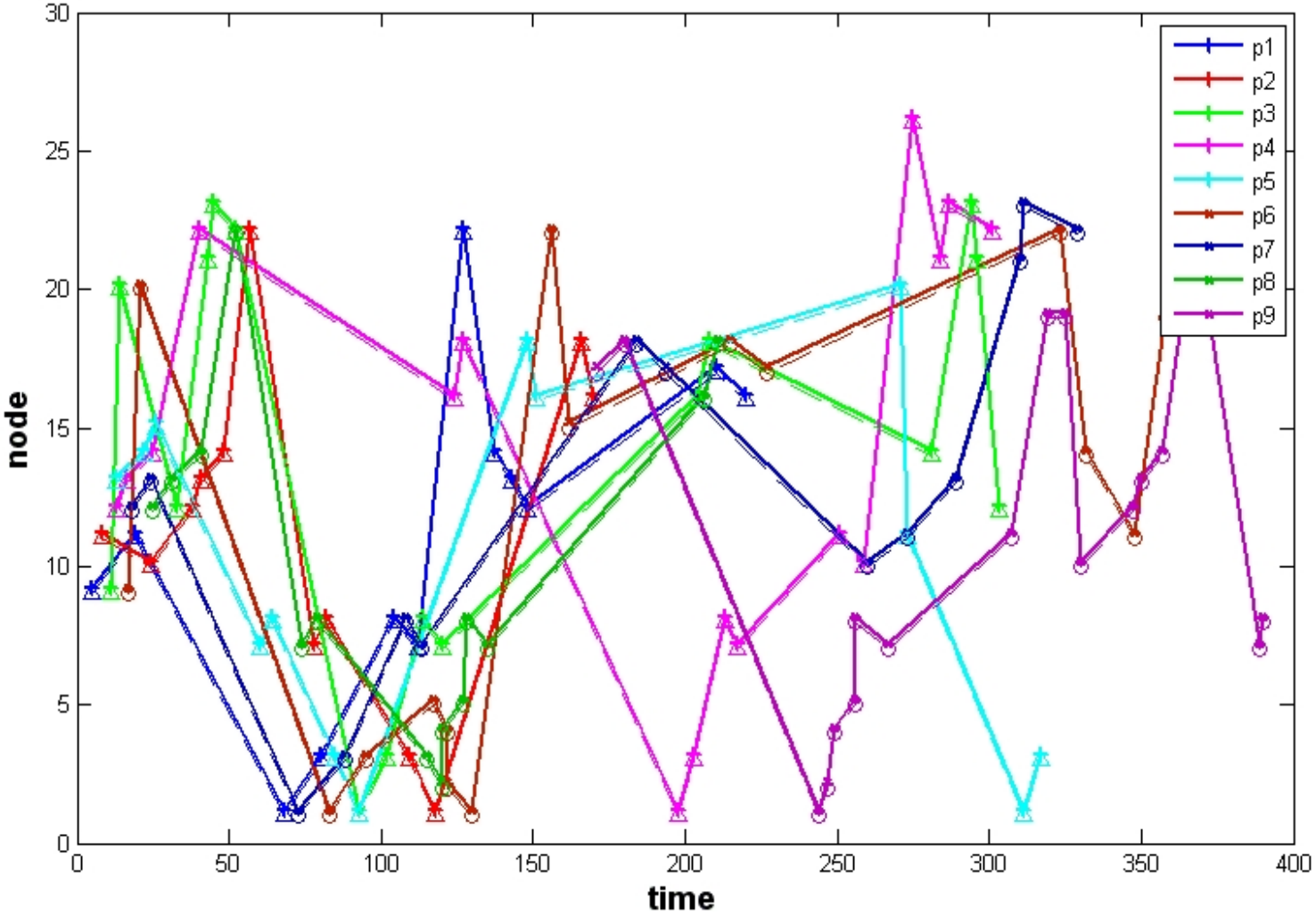


Figure 3.15: Tracking result using our stochastic adaptive algorithm using the same representation as Fig. 3.8. Compare the color of the paths along the highlighted portions of Fig. 3.8. The colors of the dotted and solid lines match indicating correct path estimation. (The figure is best viewed on a color monitor.)

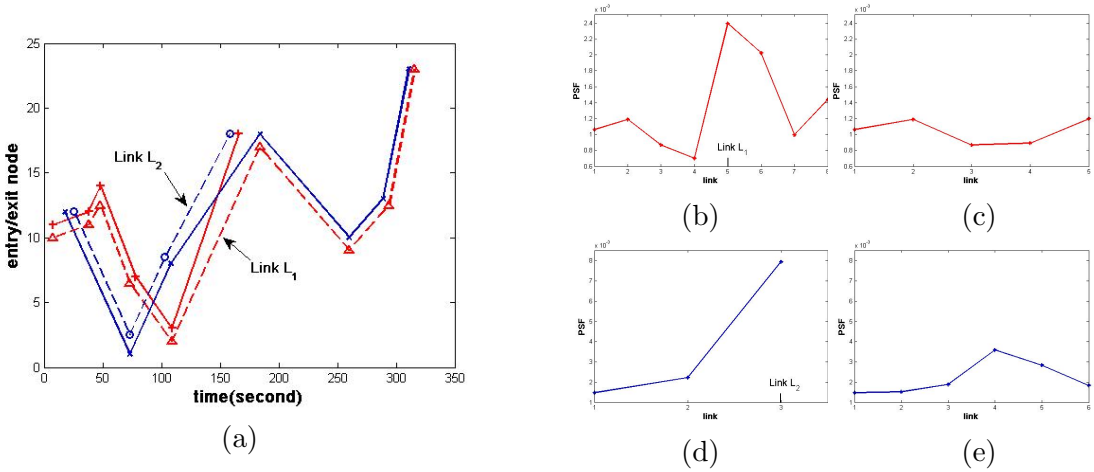


Figure 3.16: (a): Paths of two people (in blue and red) obtained from the camera network: solid lines are the ground truth and dotted lines are the tracking results using Objective 1 only (no adaptation). (b)-(c): PSF values along the incorrect (estimated) and correct (ground truth) path of person 1 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 the wrong path is higher than the variance along correct path.

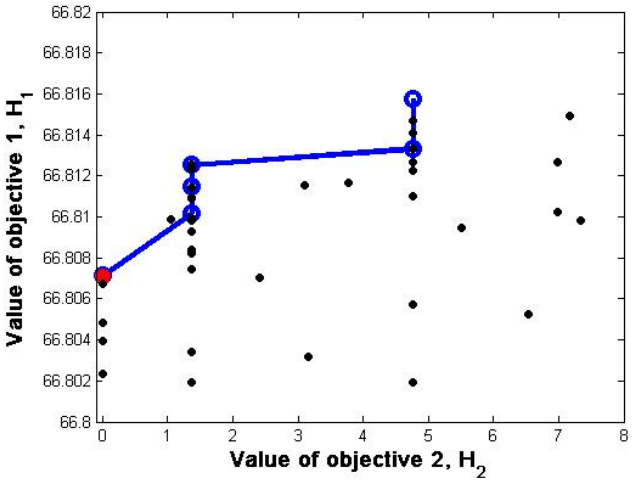


Figure 3.17: A plot of the solution space with some possible values of H_1 and H_2 obtained through a random simulation. The path taken by the optimization strategy is shown in bold, with the final solution marked with a filled circle.

results. Thus, for correct results the colors should match across all the paths. There are six wrong links in this experiment due to mistakes in feature similarity estimation. Details of one of the wrong links shown in the circle are given in Fig. 3.16, along with the adaptation

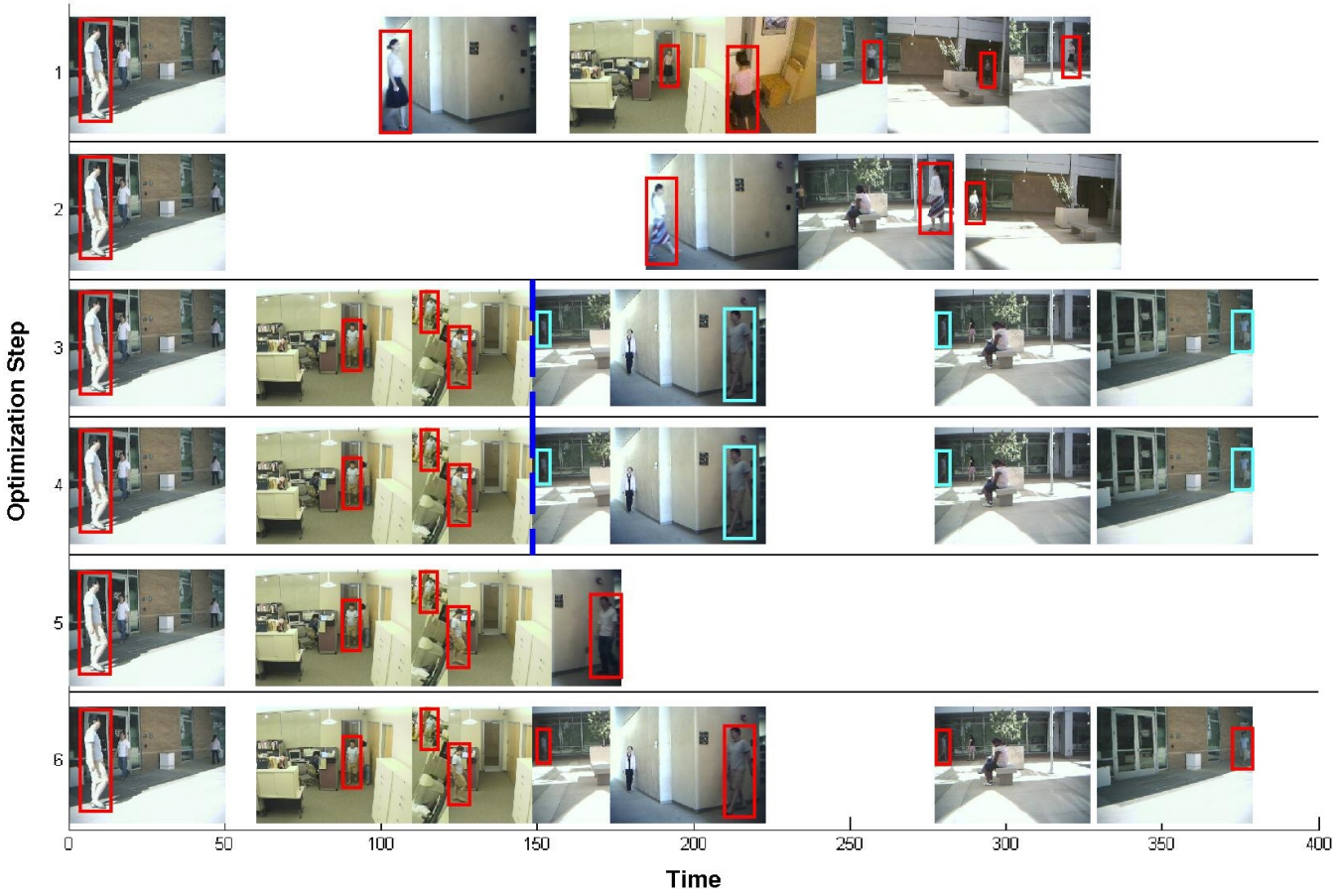


Figure 3.18: Example of the adaptation of the tracks for one person in Data Set 2. Each row shows the track of one person, marked with a box, as inferred by the multi-objective optimization in one iteration step. A correct result should have the same person in the entire row. We see that in optimization Step 1, i.e., the result from maximizing Objective 1, there are a number of wrong correspondences. After the final step of the optimization, all the correspondences for this person are correct, thus establishing a correct track. In Steps 3 and 4, the algorithm identified the same person as two different people and broke up the track into two tracks, which is also an incorrect result. We put a blue dividing line to clarify that two tracks were identified. Also, note that this is the result for only one person. Thus, even though optimization steps 3 and 4 give the same result for this person, there were differences for other individuals.

of the PSF values. The paths that are estimated wrongly due to these wrong links are highlighted. Compared with previous example, there are more wrong links, which is to be expected because of the poorer quality of the data. In Fig. 3.15, we show our result by adapting the graph edge weights using the proposed multi-objective optimization strategy. By comparing the colors along the paths highlighted in Fig. 3.14, we can see that these mistakes have been corrected. Both these figures are best viewed on a computer monitor.

Details of the solution space and the path followed by the local optimization of Section 3.5 are shown in Fig. 3.17. We show possible values of the cost functions H_1 and H_2 through a random simulation. The blue solid line shows our iteration steps for finding the final solution. In this case, we can see that the we choose the optimal path (marked by a star) as the one which maximizes H_1, while simultaneously minimizing H_2. Examples of the images for the track of Person 6, which is initially wrongly estimated, are shown in Fig. 3.18. We start with multiple wrong correspondences (Iteration 1), to the track being broken into two (Iterations 3/4), to the correct track in Iteration 6.

We also show some results of tracking multi-activities with video captured by two non-overlapping cameras by combining the framework described in Chapter 2. The setup of the cameras is shown in Fig. 3.19. The sequences consist the activities of three people. The tracking results are shown in the left of Fig. 3.20. In the right half of Fig. 3.20, the model switching result between two cameras is clearly shown. By analyzing the results, we see that we can maintain correspondences between the two cameras. Also, we can recognize that Person 2, who was walking with a briefcase in Camera 1, is without it when she reappears in Camera 2. Similarly, Person 3 picks up a briefcase in transiting from Camera 2 to Camera 1. However, Person 1 continues walking with the backpack.

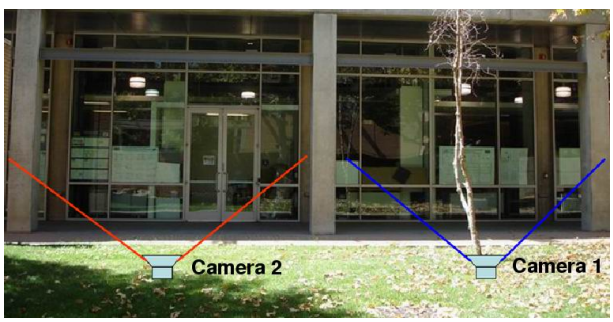


Figure 3.19: The setup of the two cameras.

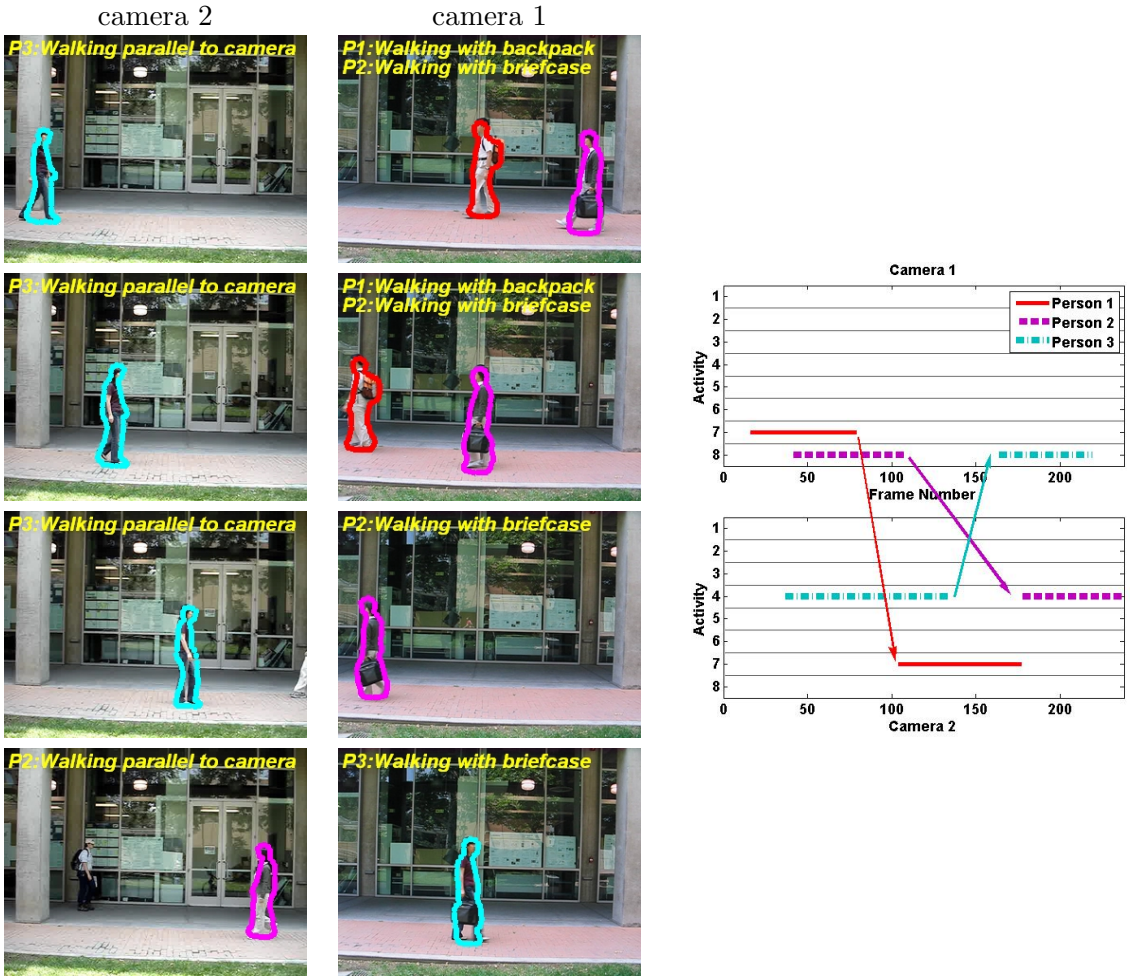


Figure 3.20: Tracking and change detection results for two non-overlapping cameras. Person 1 walked from the view of camera 1 to camera 2 with a backpack. Person 2 walked with a briefcase in the view of camera 1 first and gave the briefcase to Person 3 in the non-overlapping region, then walked into the view of camera 2. Person 3 walked from camera 2 to camera 1. Left: Tracking results with the activity models used to track shown on the top. Right: Model switching results. The labels denote: 1: Standing, 2: Sitting, 3: Walking toward camera, 4: Walking parallel to camera, 5: Walking with small package, 6: Walking with box, 7: Walking with backpack, 8: Walking with briefcase. (The figure is best viewed on a monitor.)

3.6.3 Discussion of Experimental Results

• *Distributed Processing*: In this framework, the tracking at each camera can be done locally and only the integration over space and time needs to be done centrally. Thus, we need to transmit only the tracked feature vectors to a central processor. The worst case scenario is when we need to transmit the feature vectors for every frame, i.e., a new person appears/disappears in every frame (a highly unrealistic situation). We take the video captured by one of the cameras in Data set 2 as an example to show the savings in data transmission. The video is about 6.5 minutes. If we need to transmit features for every frame, the amount of data is 9.2MB, compared with 17.2MB for MPEG2 compressed video if the entire video was transmitted without any local processing. However, since we need to transmit feature vectors only when there are objects appearing/disappearing, the actual data amount we transmitted in this case is 132KB. This provides a saving of 99% over transmitting MPEG compressed video.

• *Computational time*: The computational time depends on the size of the feature graph (number of feature nodes, see Fig. 3.2) and the iteration steps needed to reach a local minimum. Our Matlab code runs on a computer with 1.8GHz CPU. The computational time for data set 1 is about 550 seconds for analyzing 20 minutes of data and that for data set 2 is about 350 seconds for analyzing about 7 minutes of video.

• *Advantage of Heuristic Optimization Scheme*: The plots in Fig. 3.11 and 3.17 show the solution space obtained by randomly choosing different paths in the feature graph. A direct optimization of the multi-objective function would have required searching over such different paths to obtain the final solution. Our strategy constrains the optimization space close to the correct solution and allows us to search over a small set of points in this solution space.

3.7 Conclusions

In this chapter, we proposed a novel multi-objective optimization framework for tracking multiple people in a large network of non-overlapping video cameras. We considered the uncertainty of feature similarity estimation 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 using discriminant analysis to correct for wrong correspondences.

We showed that the above two conditions can be addressed by treating the issue of tracking in a camera network as a multi-objective optimization problem which could be solved using a stochastic adaptive strategy. We demonstrated the effectiveness of our system by showing results on a real life camera networks.

Chapter 4

Distributed Control and Tracking In A Camera Network

4.1 Introduction

Networks of video cameras are being installed in many applications, *e.g.*, video surveillance, national and homeland security, assisted living facilities, etc. It is natural to expect that these camera networks would be used to track targets at multiple resolutions, *e.g.*, multiple people, a single person, a face. For efficiency and maximum resource utilization, it is desirable to *actively* control the cameras so as to track the targets based on the requirements of the scene being analyzed. It would be prohibitively expensive to have a static setup that would cater to all possible situations. Currently, similar applications try to cover the entire area or the most important parts of it with a set of passive cameras but have difficulty in acquiring high resolution shots selectively.

It is also desirable that the tracking and control mechanism be distributed due to constraints of bandwidth, secure transmission facilities, and difficulty in analyzing a huge amount of data centrally. In such situations, the cameras would have to act as *autonomous agents* and decisions would have to be taken in a distributed manner. However, to be able to track all the targets in an area under surveillance accurately, the cameras should be working cooperatively with each other. This is because each camera's parameter settings entail certain constraints on other cameras. Also, if a target is observed by multiple cameras, there should be a consensus on the state (*e.g.*, position) of the target even if each camera

is an autonomous agent.

The cooperative and decentralized nature of this problem leads us to explore a game theoretic solution. Specifically, we employ a framework in which the optimization of local sensor utility functions leads to an optimal value for a global utility. This achieves the goal of keeping track of all targets at an acceptable resolution and some at high resolution. The optimal camera parameters in the sensor network are determined dynamically according to these utility functions and negotiation mechanisms based on the estimation of the states, i.e. position and velocity, of the targets. To employ suitable negotiation mechanisms between the different sensors is of great importance since the cameras have to take strategic decisions according to the perceived or anticipated actions of the other cameras. This entire framework leads to a completely decentralized approach.

4.2 Technical Rationale

4.2.1 Problem Statement

The overall goal of our work in this chapter is to develop a distributed multi-target tracking and camera network control framework to observe and keep track of all targets at the desired resolutions in an active camera network. We consider a network of synchronized calibrated cameras with pan, tilt, and zoom capabilities. Each camera has an embedded processing unit for local processing of the sensed video data. Since the cameras are calibrated, they can determine through their homographies the position of a target on a ground plane.

Some of the cameras in the network may have overlapping fields of view and a target may therefore be viewed by several cameras simultaneously. Due to inaccuracies in calibration and single-view target tracking methods, the different cameras viewing the same target will not have exactly the same measurement of the target's state on the ground plane. Consequently, it is necessary for the cameras to collaborate to reach a consensus on the actual state of the target. In our framework, this consensus must be reached only through point-to-point communications between neighboring cameras without the use of any central processing unit.

As the targets move from one camera's FOV into another, and as the cameras change their parameters, the distributed tracking system must also be able to keep track

of the targets in a seamless way. Also, a camera may have to change its parameters in coordination with other cameras so as to keep the targets imaged at the given desired resolutions. All this requires the tracking algorithm to be robust to dynamically changing camera network topologies.

4.2.2 Necessity of collaboration in a camera network

We start by motivating the necessity of an intelligent camera network and a cooperative strategy. For purposes of explanation, we will refer to this problem as Cooperative Target Acquisition (CTA). Two questions we need to address are the following - (i) why do we need active cameras (as opposed to having a network of cameras with a fixed set of parameters) and (ii) why does the control strategy need to be cooperative?

The main reason for having a dynamically configurable network is that it would be prohibitively expensive to have a static setup that would cater to all possible situations. For example, suppose we needed to focus on one person (possibly non-cooperative) or specific features (e.g., face) of the person as he walks around an airport terminal and obtain a high resolution image of him while keeping track of other activities also going on in the terminal. To achieve this, we will either need to dynamically change the parameters of the cameras where this person is visible or have a setup whereby it would be possible to capture high resolution imagery irrespective of where the person is in the terminal. The second option would be very expensive and a huge waste of resources, both technical and economical. Therefore we need a way to control the cameras based on the sensed data. Currently, similar applications try to cover the entire area or the most important parts of it with a set of passive cameras, and have difficulty in acquiring high resolution shots selectively.

The control strategy must necessarily be cooperative because each camera's parameter settings entail certain constraints on other cameras. For example, if a camera zooms in to focus on the face of one particular person, thus narrowing its field of view (FOV), it *risks* losing much of the person and the surroundings. Another camera can compensate for this by adjusting its parameters to track the person with a lower zoom setting. This requires analysis of the video data in a *network-centric manner*, leading to a cost-effective method to obtain high resolution images for features at dynamically changing locations.

Figure 4.1 shows a simple example to illustrate these points. While Camera 1 keeps the entire scene in focus, it cannot get high resolution features on any of the people.

This is done by Cameras 2, 3 and 4, but they have narrow FOVs. In fact, we can see that Camera 2 can no longer keep track of the person in the white shirt with its current settings, but can keep track of the other person. Camera 3, therefore, takes responsibility for tracking the person in the white shirt. With the information already available, Camera 4 is able to obtain a high resolution facial image. Since events will unfold dynamically over a large area, these assignments must be made dynamically. For example, if the person in the green shirt does not need to be tracked, Camera 2 can be released for some other task. However, this requires controlling the cameras such that they work in a collaborative manner. It would be very difficult, if not impossible, for a human operator to analyze multiple video streams and change the camera parameters simultaneously. Hence, the necessity for automated solutions.

4.2.3 Necessity of a decentralized strategy

As the problem complexity increases, it may be difficult to analyze all the data in a centralized manner and come up with an effective CTA strategy. There may not be enough bandwidth and transmission power available to send all the data to a central station. It is also not information rate efficient to transmit an entirely unprocessed video stream. Furthermore, security of the transmission and interception by a hostile opponent may be a factor in some applications. Finally, even if it is possible to securely transmit all the data to a central unit, it may not be the best strategy as it is often intractable to come up with a centralized optimal policy given the complex nature of these systems and the environments where they are deployed. In this framework, each camera must take its own decisions based on analysis of its own sensed data and negotiation mechanisms with other sensors.

4.2.4 Overview of solution strategy

We propose a special application of the Kalman-Consensus Filter presented in [66] to solve the problem of finding a consensus on the state (position and velocity) of multiple targets in a dynamic camera network with possibly overlapping fields of view. In our proposed distributed tracking algorithm, the cameras will only communicate with their neighbors and their specific communication links will vary in time as the targets move through the area under surveillance. Details of the tracking approach are provided in Section

Figure 4.1: An example of a camera network where different cameras work collaboratively to fulfill different tasks. While Camera 1 keeps the entire scene in focus, it cannot get high resolution features on any of the people. This is done by Cameras 2, 3 and 4, but they have narrow FOVs. In fact, we can see that Camera 2 can no longer keep track of the person in the white shirt with its current settings, but can keep track of the other person. Camera 3, therefore, takes responsibility for tracking the person in the white shirt. With the information already available, Camera 4 is able to obtain a high resolution facial image.

4.5.

The cooperative and distributed nature of this problem also leads us to explore a game theoretic solution for the camera control problem as detailed in Section 4.4. This is achieved by the optimization of local sensor utility functions leading to an optimal value for a global utility, i.e. keep view all targets at an acceptable resolution and some at high resolution.

Figure 4.2 shows an overview of our distributed multi-resolution tracking system in a network of self-configuring cameras. Each of the three neighboring smart cameras in this figure has its own embedded tracking module, control module and Kalman-Consensus filter. The tracking module receives the video data from the camera and performs the tracking of the targets in its FOV. Since the camera is calibrated, the tracking module can determine

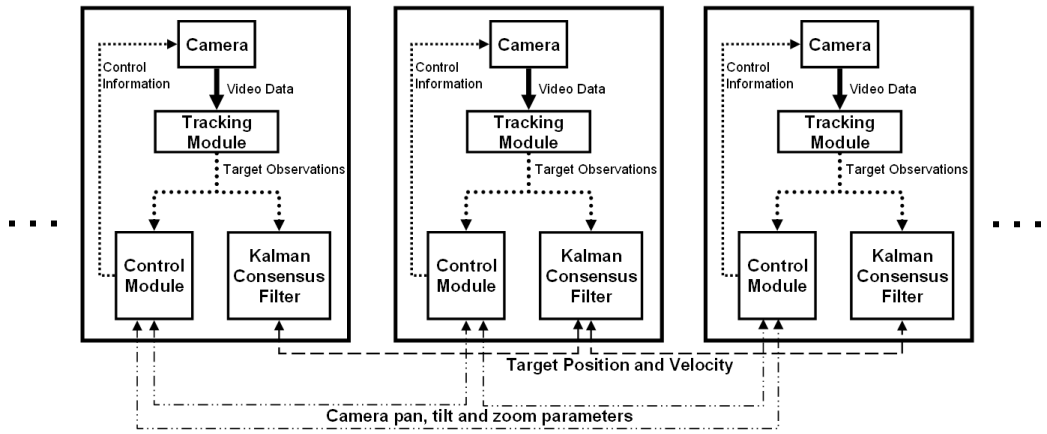


Figure 4.2: A diagrammatic representation of the proposed distributed tracking and control system with three neighboring smart cameras.

the (noisy) position of each target on the ground plane. The Kalman-Consensus filter then uses this target position information together with position and velocity information from other neighboring cameras to come to a consensus with these cameras about the actual state of the target.

The control module on the other hand, changes the parameters of the camera as necessary to track the different targets at the desired resolutions while also keeping the entire area in view. The parameter change is decided based on the parameters of neighboring cameras following the game-theoretic framework presented in Section 4.4. The parameters of the neighboring cameras are obtained through a negotiation process. The Kalman-Consensus filter and control module run independently and asynchronously in each smart camera.

As can be noticed, each camera exchanges only information regarding its own PTZ parameters and states of targets that are in its FOV. There is no video information being sent to any other camera or to a central processing station. It should also be pointed out that although the control module communicates with all of its neighbors, the Kalman-Consensus filter in our framework communicates only with a subset of the neighboring cameras. This is advantageous because the state information exchange for the consensus tracking requires a higher data rate than the control negotiation information. By reducing the number of neighboring cameras that need to communicate with each other during the

consensus tracking, we reduce the overall communication bandwidth required for the system. As can be seen clearly, this entire framework leads to a completely distributed approach for multi-target tracking and camera self-configuration.

4.2.5 Relation to previous work

Some recent work has dealt with networks of vision sensors, namely computing the statistical dependence between cameras, computing the camera network topology, tracking over the network, and camera handoff [45, 50, 57, **?**, 83]. There has also been recent work on tracking people in a multi-camera setup with overlapping fields of view [15, 47]. However, these methods are not totally distributed and do not deal with the the problem of tracking *and* control in an active camera network.

In [59], a distributed target tracking approach using a cluster-based Kalman filter was proposed. Here, a camera is selected as a cluster head which aggregates all the measurements of a target to estimate its position using a Kalman filter and sends that estimate to a central base station. Our proposed tracking system differs from this method in that each camera in a neighborhood has a consensus-based estimate of the target's state and thus there is no need for additional computation and communication to select a cluster head. Furthermore, we consider a dynamic camera network in which, as the cameras change their parameters, the targets are being kept track of seamlessly. As will be described in Section 4.5, we apply in a special way the distributed Kalman-Consensus filter [66] which has been shown to be more effective than other distributed Kalman filter schemes. Consensus schemes have been gaining popularity in computer vision applications involving multiple cameras [86].

A related work that deals with tracking targets in a camera network with PTZ cameras is [73]. Here, the authors proposed a mixture between a distributed and a centralized scheme using both static and PTZ cameras in a virtual camera network environment. Our approach to camera control and tracking, however, is completely distributed using consensus algorithms and a game-theoretic framework [2]. Preliminary work on this game theoretic approach for camera control was presented in [79]. However, our goal in that paper was merely to cover an entire area, and there was no attempt to address the tracking problem, which is critical for a distributed approach to work.

A broad framework for the vehicle-target assignment problem using game theory was presented in [2]; however, it does not include a risk vs. reward trade-off (as the dynamics of the targets is not considered) and does not consider the constraints imposed by video cameras. Unlike the above methods, we develop target acquisition strategies in a camera network that rely on cooperative decision making in dynamic environments.

4.3 A Review of Distributed Estimation and Cooperative Control in A Sensor Network

We briefly review some basic concepts related to distributed consensus and cooperative control in a sensor network that are directly relevant to our work.

4.3.1 Distributed control through cooperation

The theme of *cooperative control* has received significant attention in recent years, especially the design of autonomous vehicles with intelligent and coordinated action capabilities to achieve an overall objective. In [2], an autonomous vehicle-target assignment problem in which a group of vehicles are expected to assign themselves to a set of targets to optimize a global utility function is considered. However, rather than optimizing the global utility function directly, the emphasis was on designing vehicles that are individually capable of making coordinated decisions to optimize their own local utilities, which then indirectly translated into the optimization of a global utility function. This approach enabled the vehicles to operate in environments with limited information, communication, and computation, and still be able to optimize a global utility autonomously.

This problem was formulated as a multi-player game where each vehicle was interested in optimizing its own utility. The notion of pure Nash equilibrium can be used to represent the target assignments that are agreeable to the rational vehicles, i.e. the assignments at which there is no incentive for any vehicle to unilaterally deviate. As has been shown in [99], the combination of using the Wonderful Life Utility (WLU) as the vehicle utility and Spatial Adaptive Play (SAP) as the negotiation mechanism leads to an optimal assignment of targets within the set of pure Nash equilibria. The precise definitions of the utility functions and negotiation mechanisms for our application will be provided in Section

4.4.

4.3.2 Distributed state estimation

In the multi-agent systems literature, *consensus* means to reach an agreement regarding a certain quantity of interest that depends on the state of all sensors in a network. There is no central unit that has access to all the data from the sensors. Consequently, a *consensus algorithm* is an interaction rule that specifies information exchange between a sensor and its neighbors so that all the nodes reach a consensus. The interaction topology of a network of sensors is represented using a graph $G = (V, E)$ with the set of nodes $V = \{1, 2, ..., n\}$ and edges $E \subseteq V \times V$. Each sensor node $i = 1, ..., n$ is associated with a state $x_i \in \mathbb{R}$. Reaching a consensus means asymptotically converging to a one-dimensional *agreement space* characterized by the equation $x_1 = x_2 = ... = x_N$.

There have been recent attempts to achieve dynamic state estimation in a consensus-like manner. In contrast to a central Kalman filter where state information coming from several sensors is fused in a central station, Distributed Kalman Filters (DKF) compute a consensus-based estimate on the state of interest with only point-to-point communication between the sensors [66]. A distributed Kalman filtering (DKF) strategy that obtains consensus on state estimates was presented in [66].The overall performance of this so-called Kalman-Consensus filter has been shown to be superior to other distributed approaches. It is on this DKF strategy that we base our distributed tracking algorithm on. The mathematical details are presented in Section 4.5.2. A thorough review of consensus in networked multi-agent systems can be found in [65].

4.4 Cooperative Control Using Game Theory

Our goal is to develop a decentralized strategy for coordinated control that relies on local decision-making at the camera nodes, while being aligned with the suitable global criterion of tracking multiple targets at multiple resolutions. For this purpose, we propose to use novel *game theoretic* ideas that rely on *multi-player learning and negotiation mechanisms* [22]. The result is a decision making process that aims to optimize a certain global criterion based on individual decisions by each component (sensor) and the decisions of other

interconnected components. In the next section, we provide an intuitive reasoning to justify the presented solution strategy. Then we describe the criteria that each camera should use to evaluate its own sensed data and the design of the negotiation mechanisms. Finally, we describe the game-theoretic approach for distributed control of the sensor network.

4.4.1 Motivation for game theoretic formulation

We aim to design a *decentralized* CTA strategy for a network of imaging sensors whereby the sensors will optimally assign themselves to a set of targets. This is done using a game theoretic formulation whereby each sensor is a rational decision maker, optimizing its own utility function which indirectly translates to the optimization of a global utility function.

In the context of the motivating example we presented in Section 4.2.2, we provide an intuitive rationale why game theory is a feasible solution strategy. The targets $\{\mathcal{T}\}$ are the opponents. A target loses and is eliminated from the game if it is captured in an image of the desired resolution (specified by application requirements). Our team of cameras scores each time it obtains a desired resolution image of a target; however, there is risk associated with each high resolution image attempt, as explained in detail in Section 4.2.2 (acquisition of each target at high resolution risks losing the global picture of the relative positions of all the targets and subsequent loss of track of a target if it leaves the field of view (FOV) of its imaging device.)

When the number of targets is larger than the number of cameras, the necessity of cooperation between the cameras is obvious. Even when the number of cameras is larger than the number of targets, the benefit of cooperation and information sharing is clear. A camera tracking T_i at relatively low zoom allows other cameras to track other targets at higher zoom levels, provided the state of the camera tracking T_i could be communicated to them. In addition, cameras at intermediate zoom levels can allow tracking of multiple targets at a sufficient level of accuracy to enable other cameras to attempt higher risk, but higher reward, images at high zoom levels. This shows that a decentralized strategy for camera control is feasible through local decision making and negotiations with neighboring cameras.

The objective of the team is to optimize the global utility function by having each

camera optimize its own local utility function, again subject to risk, but now also subject to predictions of the actions of the other cameras. The first step is to find suitable local utility functions such that the objectives of each camera are localized to that camera, yet aligned with a global utility function. The second step is to propose an appropriate negotiation mechanism between cameras to ensure convergence of the distributed solution toward the global solution. The actual computation of these functions depends upon the analysis of the sensed video.

4.4.2 Precise problem statement and notation

The overall goal of the imaging network is to track all the objects of interest in the area where they are deployed and acquire high resolution imagery of some of these objects (specified by a user or another application). The global utility function should provide a measure of this objective.

Let us consider N_t targets in the entire area of deployment and N_c sensors that need to be assigned to these targets. In the camera network setup, each target will be represented by a location vector and a resolution parameter, e.g., in Figure 4.1, one target could be the face of the person in the white shirt with a high resolution setting, while other targets could be all moving objects in the entire observed area with a low resolution parameter setting. Let the cameras be denoted as $\mathcal{C} = \{C_1, ..., C_{N_c}\}$ and the targets as $\mathcal{T} = \{T_1, ..., T_{N_t}\}$. Camera $C_i \in \mathcal{C}$ will select its own set of targets $a_i (= \{T_i\}) \in \mathcal{A}_i$, where $\mathcal{A}_i$ is the set of targets that can be assigned to C_i, by optimizing its own utility function $U_{C_i}(a_i)$. This is known as target assignment in the game theory literature. Our problem is to design these utility functions and appropriate negotiation procedures that lead to a mutually agreeable assignment of targets resulting in meeting the global criterion.

4.4.3 Game theory fundamentals

A well-known concept in game theory is the notion of Nash equilibrium. In the context of our image network problem, it will be defined as a choice of targets $a^* = (a_1^*, ..., a_{N_c}^*)$ such that no sensor could improve its utility further by deviating from a^*. Obviously, this is a function of time since the targets are dynamic and the sensors could also be mobile or capable of panning, tilting and zooming. For our problem, a Nash equilibrium will be

reached at a particular instant of time when all the cameras are tracking all the targets in the deployment region at an acceptable resolution and there is no advantage for a particular camera to choose some other target to track.

Mathematically, if a_{-i} denotes the collection of targets for all cameras *except* camera C_i, then a^* is a *pure Nash equilibrium* if

$$U_{C_i}(a_i^*, a_{-i}^*) = \max_{a_i \in \mathcal{A}_i} U_{C_i}(a_i, a_{-i}^*), \forall C_i \in \mathcal{C}. \tag{4.1}$$

4.4.4 Choice of utility functions

Target Utility.

If the number of cameras viewing T_i at an acceptable resolution r_0 is $n_i^{r_0}$, then the utility of covering T_i using a particular assignment profile a is

$$U_{T_i}(a) = \begin{cases} 1 & \text{if } n_i^{r_0} > 0 \\ 0 & \text{if } n_i^{r_0} = 0 \end{cases} \tag{4.2}$$

Global Utility.

From the target utility function, we can now define the global utility function as the sum of the utilities generated by tracking all the targets, i.e.,

$$U_g(a) = \sum_{T_i} U_{T_i}(a). \tag{4.3}$$

Camera Utility.

We now come to the all important question of defining camera utility in a suitable manner so that it is aligned to the global utility function. A target utility, U_{T_i}, represents the overall value of tracking a target T_i at an acceptable resolution. The camera utility, U_{C_i}, represents a particular sensor's share of this value. We will say that *the camera utility is aligned with the global utility when the sensor can take an action improving its own utility if and only if it also improves the global utility* [92]. This will require prediction of the actions of team-mates, which will be achieved through the negotiation mechanisms described in

the next section. Under this definition, we can use a number of utility functions that have been proposed in the game theory literature [60, 92]. In our application, we propose to use what is known as Wonderful Life Utility (WLU). In WLU, the utility of a sensor tracking a particular target is the marginal contribution to the global utility as a result of this action, i.e., the sensor utility is the change in the global utility as a result of that sensor tracking that particular target as opposed to not tracking it. The exact expression is

$$U_{C_i}(a_i, a_{-i}) = U_g(a_i, a_{-i}) - U_g(a_{-i}). \tag{4.4}$$

This definition of camera utility works well if the teammate predictions are accurate and stable. As shown in [60], this sensor utility leads to a potential game with the global utility function as the potential function, and hence they are aligned with the global utility. This ensures that the resulting set of targets that are chosen will be included within the set of pure Nash equilibria. In particular, the Wonderful Life Utility has been widely used in economics [58].

4.4.5 Negotiation mechanisms

Tracking objects of interest in a dynamic setting requires negotiation mechanisms between the different sensors, allowing them to come up with the strategic decisions described above. Each sensor negotiates with other sensors to (i) accurately estimate the state of the targets, (ii) accurately predict their team-mates' parameters, and (iii) decide its own action. Note that this entails each camera to consider risk, along with the rewards measured by the utility functions described above. This makes each sensor truly autonomous thus providing robustness in uncertain and adversarial environments, where there could be a lack of adequate communication resources, and incomplete knowledge of the sensed data and environmental conditions of the other sensors. Hence the cooperation between the sensors is limited to exchanging information about the states of other sensors and targets, not the actual video data.

The overall idea is to use *learning algorithms for multi-player games* [60]. A particularly appealing strategy for this problem is Spatial Adaptive Play (SAP) [99]. This is because it can be implemented with a low computational burden on each camera and leads to an optimal assignment of targets with arbitrarily high probabilities for the WLU

described above. In a particular step of the SAP negotiation strategy, a camera is chosen randomly according to a uniform distribution. This camera can update the targets it is tracking so as to maximize its own utility by taking into account the proposed targets of all the other cameras in the previous step.

Let us consider a concrete example to illustrate this. Suppose that the chosen camera, say C_i with zoom z_i, detects a new object within its FOV that it may need to track at a high resolution (specified by a user, for example.) Now, C_i can choose to track this object or, if the object is being tracked by some other camera, it can leave it to that camera. This is where the negotiation mechanism comes in. If C_i has access to the assignment of the other cameras at a previous step, it can determine whether that camera will be able to maintain tracking at the desired resolution or the object will be out of that camera's FOV. Thus, C_i can reach an informed decision based on its own utility function and the negotiation mechanism that allows it to access the previous decisions of the other cameras. It also requires predicting the position of the target at a future time instant. By proposing this action, it can monitor the responses of the other cameras and assess the risk/reward tradeoff.

Application of SAP negotiation mechanism

In our method for camera control in a sensor network, we adopt the Spatial Adaptive Play (SAP) strategy. Let us consider a camera C_i which is viewing the area under surveillance at an acceptable resolution. At any step of SAP negotiations, C_i is randomly chosen from the pool of cameras in the network according to a uniform distribution [1], and only this camera is given the chance to update its proposed parameter settings. At any negotiation step, C_i proposes a parameter setting to maximize its own utility based on other cameras' parameters at the previous step. After C_i updates its settings, it broadcasts its parameters (i.e. pan, tilt and zoom) to the entire network. Based on calibration data, the other cameras can then calculate the area covered by C_i and use that information to update its parameters after being chosen at any negotiation step. After the negotiation converges, the entire area of interest is viewed at an acceptable resolution. An implicit assumption is that the amount of time it takes for the assignments to be finalized is less that the time

[1] The issue of choosing which camera to update in a decentralized manner will be dealt with in future work. Some suggestions for implementing this can be found in [2].

for the targets to move from one camera to another. This will be true if each camera has sufficient processing power and if cameras are assigned to targets only when the targets will be in a cameras' FOV for a reasonable amount of time.

Algorithm 1 Maximize utility and negotiate

Input: Camera C_i, Video from C_i
C_i calculates parameters needed to maximize its utility at r_0 based on area covered by other cameras obtained from negotiation mechanism
if C_i needs to change its parameters to maximize its utility **then**
  C_i changes its parameters
  C_i broadcasts its parameters (pan/tilt/zoom) to all the cameras in the network
end if

Algorithm 2 Initialize cameras to cover entire area at acceptable resolution

Initial State: Cameras are not covering entire area at r_0
while entire area is not being covered at r_0 **do**
  Choose a camera C_i randomly according to a uniform distribution
  Call *Algorithm1*(C_i)
end while
End State: Cameras are covering entire area at r_0. Each target is represented by (T_k, r_0).

Algorithm 3 Track all targets at an acceptable resolution

Initial State: Cameras are covering entire area at r_0
for $t = t_0, t_1, t_2, ...$ **do**
  All C_is track all (T_k, r_0)s in their FOVs
  All C_is predict $\hat{p}_k(t + \Delta)$ for all (T_k, r_0)s in their FOVs
  if a C_i determines that a $(T_{k_{out}}, r_0)$ will move out of its FOV **then**
    C_i broadcasts $\hat{t}_{k_{out}}$ and $\hat{p}_{k_{out}}$ to all the cameras in the network
  end if
  if a C_i determines that it can track $(T_{k_{out}}, r_0)$ based on $\hat{t}_{k_{out}}$ and $\hat{p}_{k_{out}}$ **then**
    C_i takes over tracking of $(T_{k_{out}}, r_0)$
  end if
end for
End State: All cameras are tracking all targets at an acceptable resolution

Let us now consider the case where a specific application requires the tracking of a target at a high resolution. Initially, if a camera C_j determines that it can view the target at the highest resolution based on the parameters of the other cameras in the network, it starts tracking the target. Once C_j takes on the task of tracking the target at a high resolution, it will predict the state (i.e. position and velocity) of the target at each time step $t + \Delta$, where Δ is the prediction time. If C_j predicts that it will not be able to view the target after a time t_{out}, it will asynchronously broadcast t_{out} and the target's predicted position at t_{out} to the entire network indicating that it needs to handoff the tracking to

Table 4.1: Variables used in algorithms

t	discrete time step
$\{C_l\}_{l=1}^{N_c}$	Set of all cameras
$\{T_k\}_{k=1}^{N_t}$	Set of all targets
(T_k, r_0)	Target being tracked at acceptable resolution r_0
(T_k, r_h)	Target being tracked at specified high resolution r_h
C_i	Camera tracking (T_k, r_0)
C_h	Camera tracking (T_k, r_h)
Δ	Prediction time step
$\hat{p}_k(t+\Delta)$	Predicted position of T_k at time $t+\Delta$
$\hat{t}_{k_{out}}$	Predicted time in which T_k is expected to leave C_l's FOV
$\hat{p}_{k_{out}}$	Predicted position of T_k at $\hat{t}_{k_{out}}$
$T_{k_{out}}$	Target about to leave C_l's FOV

Algorithm 4 Track some targets at high resolution, others at acceptable resolution

Initial State: All cameras are tracking all targets at r_0
$handoffHighResTargetFlag = 0$
for $t = t_0, t_1, t_2, ...$ **do**
  if a (T_k, r_h) has been selected (manually or automatically) **then**
    $newHighResTargetFlag = 1$
  end if
  Choose a camera C_i randomly according to a uniform distribution
  if $newHighResTargetFlag == 1$ **AND** C_i can track (T_k, r_h) based on the range of its FOV **then**
    C_i starts tracking (T_k, r_h) and becomes C_h
    $newHighResTargetFlag == 0$
  else if $handoffHighResTargetFlag == 1$ **AND** C_i can track $(T_{k_{out}}, r_h)$ based on $\hat{t}_{k_{out}}$, $\hat{p}_{k_{out}}$ being within the range of its FOV **then**
    C_i takes over tracking of $(T_{k_{out}}, r_h)$ and becomes C_h
    Previous C_h becomes C_i
    $handoffHighResTargetFlag == 0$
  else
    Call *Algorithm1*(C_i)
  end if
  C_h predicts $\hat{p}_k(t+\Delta)$ for (T_k, r_h)
  if C_h determines that (T_k, r_h) will move out of its FOV thus becoming $(T_{k_{out}}, r_h)$ **then**
    Broadcast $handoffHighResTargetFlag = 1$, $\hat{t}_{k_{out}}$ and $\hat{p}_{k_{out}}$ of (T_k, r_h) to all the cameras in the network
  end if
end for

another camera. When a camera C_i $(i \neq j)$ is now chosen to update its parameters at any negotiation step, it will first determine if it can adjust its parameter to view the target at the desired high resolution based on the predicted target parameters broadcast by C_j. If it can do so, it will set its parameters accordingly and take over the tracking of the target. However, if C_i determines it is not able to view the target at a high resolution, it will simply

continue with the negotiation cycle maximizing its own utility as presented above. Once C_j is not able any more to track the target, it returns to the negotiation cycle to maximize its own utility.

This entire negotiation approach based on game theory can be seen in more detail in Algorithms 1-4 (variables are described in Table 4.1). In these algorithms, $t = \{t_0, t_1, t_2, ...\}$ need not be contiguous time instants and will depend upon the time it takes for each target assignment to be complete. We will assume that the computation time to come up with a set of camera parameters, given a target configuration, is much smaller than the time it takes for the target configuration to change. Since all possible targets will always be tracked at a low resolution, this implies that we are concerned only with the time interval for which a high resolution target is chosen. Hence, it is a very reasonable assumption.

4.4.6 Theoretical Analysis of Proposed Framework

It has been shown that for the decentralized control strategy it is possible to achieve convergence to a Nash equilibrium when the targets are static []. However, for the case of dynamic targets and utility functions that depend upon video analysis algorithms, we are not aware of any results that prove convergence. In this regard, not only will the proposed algorithms be unique, but we expect to make fundamental theoretical results in this area. Below we outline some preliminary steps in this regard that build directly upon existing results in game theory. The dynamics of the targets and the specific choice of utility functions, tied to the video analysis algorithms, is not considered. However, it provides a preliminary framework to enable the development more sophisticated results this project will lead to.

Existence of Pure Nash Equilibrium

Definition 1. A game is called a **potential game** if the incentive of all players to change the strategy can be expressed in a global potential function. These games are studied in [60]. The concept of potentiality helps us to specify an important set of games because of their desirable convergence properties.

Definition 2. A game is an **exact potential game** if there exists potential function ϕ:

$\mathcal{A} \to \mathbb{R}$ such that for every camera, $C_i \in \mathcal{C}$, for every $a_i \in \mathcal{A}_{-i}$ and for every $a_i' \in \mathcal{A}_{-i}$,

$$U_{C_i}(a_i, a_{-i}) - U_{C_i}(a_i', a_{-i}) = \phi(a_i, a_{-i}) - \phi(a_i', a_{-i})$$

Definition 3. A game is an **ordinal potential game** if there exists potential function ϕ: $\mathcal{A} \to \mathbb{R}$ such that for every camera, $C_i \in \mathcal{C}$, for every $a_i \in \mathcal{A}_{-i}$ and for every $a_i' \in \mathcal{A}_{-i}$,

$$U_{C_i}(a_i, a_{-i}) - U_{C_i}(a_i', a_{-i}) > 0 \Leftrightarrow \phi(a_i, a_{-i}) - \phi(a_i', a_{-i}) > 0$$

Theorem 4.4.1. *If game $\Gamma(\mathcal{A}, U_c)$ has a continuous ordinal potential function, ϕ, with a compact strategy space $\mathcal{A}$,*
(i) the equilibrium set of $\Gamma(\mathcal{A}, U_c)$ coincides with the equilibrium set of $\Gamma(\mathcal{A}, \phi)$,
(ii) Γ has at least one Nash Equilibrium in pure strategies.

Proof. (i) follows directly from the definition of an ordinal potential.
(ii) $\hat{a}$ is a Nash Equilibrium for game $\Gamma(\mathcal{A}, \phi)$ if $\hat{a} = \arg\max_{a \in \mathcal{A}} \phi(a)$, and it follows from (i) that $\hat{a}$ is a Nash Equilibrium for game $\Gamma(\mathcal{A}, U_c)$. □

Application to camera networks

As we mentioned earlier, camera utility should be defined in a suitable manner so that it is aligned to the global utility function. For every camera, $C_i \in \mathcal{C}$, when we change its set of targets from a_i to a_i', if

$$U_{C_i}(a_i, a_{-i}) - U_{C_i}(a_i', a_{-i}) > 0 \Leftrightarrow U_g(a_i, a_{-i}) - U_g(a_i', a_{-i}) > 0$$

the camera utility U_{C_i} is said to be aligned with the global utility U_g. By comparing with the notion of ordinal potential game in Definition 2, it is obvious that the camera utilities, $U_{C_i}(a), i = 1, ..., n_c$, are specified to make them "aligned" with the global utility, $U_g(a)$, and the game $\Gamma(\mathcal{A}, U_c)$ is an ordinal potential game with ordinal potential function U_g.

Let a_{opt} denote an optimal set profile, i.e.,

$$a_{opt} \in \arg\max_{a \in \mathcal{A}} U_g(a).$$

Then, according to the proof of Theorem 4.4.1, a_{opt} is a (possibly nonunique) pure Nash

equilibrium of the game.

Convergence of the camera network to Nash Equilibrium

It has been proved in [2] that the profiles generated by SAP converge to the set of Nash equilibria in potential game with arbitrarily high probability. Based on this, we can now say the following.

Theorem 4.4.2. *An ordinal potential game consists of camera utilities $U_{C_i}(a), C_i \in \mathcal{C}$ with the global utility $U_g(a)$ being a potential function. Then the target profiles $a(k)$ generated by SAP satisfy*

$$\lim_{\tau \downarrow 0} \lim_{k \to \infty} \mathbf{Prob}\{a(k) \textit{is an optimal profile}\} = 1. \tag{4.5}$$

4.5 Distributed Multi-Target Tracking Through Consensus

4.5.1 Dynamic Network Topology

As mentioned earlier, we propose a special application of the Kalman-Consensus Filter presented in [66] to solve the problem of finding a consensus on the state of multiple targets in a dynamic camera network with possibly overlapping fields of view. However, unlike the sensor nodes in [66], the nodes in our camera network are directional sensors and a new method for establishing the network topology between the camera nodes must therefore be developed. Furthermore, since the camera network is composed of cameras that change their parameters and therefore their fields of view as needed, the network topology will be dynamically changing also.

Let $\mathcal{C}$ be the set of all cameras in the network. We can then define the subset of all cameras viewing target T_l as $\mathcal{C}_l^v \subset \mathcal{C}$ and the rest of the cameras as $\mathcal{C}_l^p \subset \mathcal{C}$. Each camera C_i will also have its set of *neighboring cameras* $\mathcal{C}_i^n \subset \mathcal{C}$. Since all the cameras can change their PTZ parameters and have therefore several possible fields of view, we define the set $\mathcal{C}_i^n$ as all the cameras with which C_i can potentially have an overlapping field of view. By definition, it becomes clear then that for each $C_i \in \mathcal{C}_l^v$, it is true that $\mathcal{C}_l^v \subset \{\mathcal{C}_i^n \cup C_i\}$, i.e. all of the cameras viewing a specific target are also neighbors. Note that the set of neighbors need not be geographical neighbors.

In our proposed distributed tracking system, only cameras that have an observation of a target T_l will communicate with their neighbors. This feature limits the amount of data exchanged for the consensus tracking. By reducing the number of neighboring cameras that need to communicate with each other during the consensus tracking, we also reduce the power requirements of each camera node and the overall communication bandwidth required for the system.

This way of defining the network connections between the camera nodes brings about a dynamic graph $G_l(k) = (V_l(k), E_l(k))$ representing the network topology of the system with respect to target T_l at time instant k. When there are multiple targets in the area under surveillance, there will be a distinct network topology for each target. Obviously, as T_l moves through the area under surveillance, $G_l(k)$ will also change since other cameras will be viewing T_l and only these cameras will communicate with their neighbors thus creating a dynamic network topology. The network topologies of all the targets will also change when one or more cameras change their parameters. As will be explained in the following section, this definition of network topology for distributed tracking does not affect in any way the performance of the Kalman-Consensus filter presented next.

4.5.2 Kalman-Consensus tracking

To model the motion of a target T_l on the ground plane as observed by camera C_i, we consider a linear dynamical system with process and sensing models

$$\mathbf{x}^l(k+1) = \mathbf{A}^l(k)\mathbf{x}^l(k) + \mathbf{B}^l(k)\mathbf{w}^l(k); \quad \mathbf{x}^l(0) \tag{4.6}$$

$$\mathbf{z}_i^l(k) = \mathbf{H}_i^l(k)\mathbf{x}^l(k) + \mathbf{v}_i^l(k) \tag{4.7}$$

where $\mathbf{w}^l(k)$ and $\mathbf{v}_i^l(k)$ are zero mean white Gaussian noise ($\mathbf{w}^l(k) \sim \mathcal{N}(0, \mathbf{Q}^l), \mathbf{v}_i(k) \sim \mathcal{N}(0, \mathbf{R}_i^l)$) and $\mathbf{x}^l(0)$ is the initial state of the target. We define the state of the target at time step k as $\mathbf{x}^l(k) = (x^l(k), y^l(k), \dot{x}^l(k), \dot{y}^l(k))^T$ where $(x^l(k), y^l(k))$ and $(\dot{x}^l(k), \dot{y}^l(k))$ are the position and velocity of target T_l in the x and y directions respectively. $\mathbf{x}_i^l$ is the state of T_l based on the observations in C_i only. The noisy measurement $\mathbf{z}_i^l(k)$ at camera C_i is the sensed target position $(x_i^l(k), y_i^l(k))$ on the ground plane based on the pre-computed homography.

Our special implementation of the Kalman-Consensus distributed tracking algo-

Algorithm 5 Distributed Kalman-Consensus tracking algorithm performed by every C_i at discrete time step k. The state of T_l is represented by $\mathbf{x}_i^l$ with error covariance matrix $\mathbf{P}_i^l$ (see Sec. 4.5.2).

Input: $\bar{\mathbf{x}}_i^l$ and $\mathbf{P}_i^l$ from time step $k-1$
for each T_l that is being viewed by $\{\mathcal{C}_i^n \cup C_i\}$ **do**
  if $C_i \in \mathcal{C}_l^v$ (i.e. C_i is viewing T_l) **then**
    Obtain ground plane measurement $\mathbf{z}_i^l$ with covariance $\mathbf{R}_i^l$
    Compute information vector and matrix

$$\mathbf{u}_i^l = \mathbf{H}_i^{l\,T}\mathbf{R}_i^{l\,-1}\mathbf{z}_i^l$$
$$\mathbf{U}_i^l = \mathbf{H}_i^{l\,T}\mathbf{R}_i^{l\,-1}\mathbf{H}_i^l$$

    Send message $\mathbf{m}_i^l = (\mathbf{u}_i^l, \mathbf{U}_i^l, \bar{\mathbf{x}}_i^l)$ to neighboring cameras $\mathcal{C}_i^n$
  end if
  Receive messages $\mathbf{m}_j = (\mathbf{u}_j^l, \mathbf{U}_j^l, \bar{\mathbf{x}}_j^l)$ from all cameras $C_j \in \mathcal{C}_l^v$
  Fuse information matrices and vectors

$$\mathbf{y}_i^l = \sum_{j \in \mathcal{C}_l^v} \mathbf{u}_j^l, \quad \mathbf{S}_i^l = \sum_{j \in \mathcal{C}_l^v} \mathbf{U}_j^l$$

  Compute the Kalman-Consensus state estimate

$$\mathbf{M}_i^l = ((\mathbf{P}_i^l)^{-1} + \mathbf{S}_i^l)^{-1}$$
$$\hat{\mathbf{x}}_i^l = \bar{\mathbf{x}}_i^l + \mathbf{M}_i^l(\mathbf{y}_i^l - \mathbf{S}_i^l\bar{\mathbf{x}}_i^l) + \gamma\mathbf{M}_i^l \sum_{j \in \mathcal{C}_l^v} (\bar{\mathbf{x}}_j^l - \bar{\mathbf{x}}_i^l)$$
$$\gamma = 1/(||\mathbf{M}_i^l|| + 1), ||\mathbf{X}|| = (tr(\mathbf{X}^T\mathbf{X}))^{\frac{1}{2}}$$

  Update the state and error covariance matrix for time step k

$$\mathbf{P}_i^l \leftarrow \mathbf{A}^l\mathbf{M}_i^l\mathbf{A}^{l\,T} + \mathbf{B}^l\mathbf{Q}^l\mathbf{B}^{l\,T}$$
$$\bar{\mathbf{x}}_i^l \leftarrow \mathbf{A}^l\hat{\mathbf{x}}_i^l$$

end for

rithm is presented in Algorithm 5. This algorithm is performed distributedly in each camera node C_i. At each time step k and for each target T_l, we assume to be given the estimated target state $\bar{\mathbf{x}}_i^l$ and the error covariance matrix $\mathbf{P}_i^l$ from time step $(k-1)$. Each C_i also knows its set of neighbors $\mathcal{C}_i^n$ from the information used by the camera's control module. At time step $k=0$, the Kalman-Consensus filter is initialized with $\mathbf{P}_i^l = \mathbf{P}_0$ and $\bar{\mathbf{x}}_i^l = \mathbf{x}_i^l(0)$ = average of $\mathbf{z}^l(k)$'s of neighbors viewing T_l.

The following gives a verbal description of Algorithm 5 performed at each C_i distributedly for each T_l that is viewed by $\{\mathcal{C}_i^n \cup C_i\}$. If C_i is viewing a target T_l, it determines T_l's ground plane position $\mathbf{z}_i^l$ and computes the corresponding information vector and matrix with the given measurement covariance matrix $\mathbf{R}_i^l$ and output matrix $\mathbf{H}_i^l$. C_i then sends a message $\mathbf{m}_i^l$ to its neighbors which includes the computed information vector and matrix,

and its estimated target state $\bar{\mathbf{x}}_i^l$ at previous time step $(k-1)$. C_i then receives similar messages $\mathbf{m}_j$ only from the cameras in its neighborhood that are also viewing T_l. The information matrices and vectors received from these messages, and its own information matrix and vector if C_i is viewing T_l, are then fused and the Kalman-Consensus state estimate is computed in a way similar to the method proposed in [66]. Finally, the ground plane state $\bar{\mathbf{x}}_i^l$ and error covariance matrix $\mathbf{P}_i^l$ are updated according to the assumed linear dynamical system.

4.5.3 Handoff, Network Reconfiguration and Fault Tolerance

Through this algorithm, each C_i has a consensus-based ground plane state estimate of each target that is being viewed by its neighboring cameras, even if C_i has never seen some of the targets. Since we are assuming that the network of cameras as a whole is always covering the entire area under surveillance through the game theoretic control framework presented in Section 4.4, the target will always be seen by at least one camera. Also, by our definition of neighboring cameras, a target T_l will always move from one camera C_i's FOV to the FOV of a neighboring camera $C_j \in \mathcal{C}_i^n$. Therefore, C_j can take over the tracking of T_l and find the target correspondence in a seamless way since it had knowledge of T_l's ground plane position through the consensus-tracking before it even entered its FOV. Additional target features could be used to find the target correspondences in a cluttered scene. Furthermore, even as a camera changes its parameters, it can also take over the tracking of the targets in its new FOV immediately since it also knew the position of the targets in its neighborhood beforehand through the consensus-tracking algorithm.

Another advantage of the fact that cameras have knowledge of all the targets in their neighborhood is that in the event of a sudden failure of camera node C_i, the targets that were viewed by C_i are not suddenly lost by the camera network. The neighboring cameras can adjust their parameters to cover the area that was left uncovered by C_i's failure and continue with the consensus-tracking algorithm without any interruption.

We have also considered the fact that a camera may take a short amount of time to change its parameters to a new position. If no camera is viewing the target for the short amount of time it takes for the cameras to come to a new set of parameters to cover the entire area, the target state estimate just follows the assumed linear state equations.

This does not translate to a significant decrease in tracking performance as seen in our experiments. In the next section, we explain how the camera parameters are changed in order to keep the different targets imaged at the specified resolution.

4.6 Experimental Results

4.6.1 Distributed Camera Control

To evaluate the proposed game theoretic method for camera control, we tested our approach on both simulated and real-life network of cameras.

Camera Control on Simulated Data

The area under surveillance is set up as a rectangular region of 20 by 30 meters. Since we cannot know the time and location in which the targets will enter the area, we treat the entire area under surveillance as targets. In our specific implementation presented here, we divide the area into grids in a way similar to [18] to make the problem more tractable. Therefore, we treat each grid of the area as a virtual target. In this simulation, the grids are of size 1x1 meter. The sensor network consists of 14 PTZ cameras with a resolution of 320x240 pixels spread out over the perimeter of the area as seen in Fig. 4.3(a). Initially, the camera parameters (pan/tilt/zoom) are assigned arbitrarily so that the area under surveillance is not entirely covered by the cameras' FOVs.

The first goal is to determine an initial set of camera parameters to cover the entire area at an acceptable resolution. To achieve this, we use the game theoretic approach described above (Algorithm 2). Since each grid in the area is treated as a virtual target, the cameras can update at each negotiation step which of these virtual targets they are tracking in order to maximize their own utility according to the parameter settings of the other cameras at the previous time step. The zoom factor to view the targets at the desired resolution is determined by the the number of pixels a target (assumed here to have a height of 170 cm) occupies on the image plane. At each negotiation step, after a camera updates its pan/tilt/zoom parameters, these parameters are transmitted to the entire network. The other cameras can therefore calculate the new area that this camera is now covering. This process is repeated at each negotiation step in which a camera is chosen randomly to update

its parameters. In this simulated setting, the negotiation converges to a completely covered area at an acceptable resolution after 14 negotiation steps as seen in Fig. 4.3(a) (area in gray is covered, area in white is not covered).

After the entire region is covered at an acceptable resolution, it follows that every moving target in that area is also being viewed at an acceptable resolution given the constraint of covering the whole area. The cameras would be using Algorithm 3. However, a human operator could chose to track a feature (e.g. face) of a specific moving target at a higher resolution for a specific application (e.g. face recognition). This chosen target is the marked target in Fig. 4.3(b). Then, the camera that can view the specified feature of the target at a high resolution takes the task of tracking the target changing its parameters by using Algorithm 4. The results of the tracking can be seen in the dark FOV in Fig. 4.3(c). Since the parameter change of that chosen camera, C_h, left some parts of the area uncovered, the other cameras have to change their parameters during the negotiation in order to once again cover the whole area at an acceptable resolution. That parameter change of the cameras is evident in Fig. 4.3(d). As presented before, C_h will now predict the position of the target at time t. If C_j determines that it will not be able to track the high-resolution target after a time instance t_{out}, it will transmit t_{out} to the camera network as well as the predicted position of the target at t_{out} so that it can handoff the tracking to another camera. As can be seen in Fig. 4.3(d), the high-resolution target is about to exit the camera's field of view and the camera broadcasts the handoff parameters to the network. In Fig. 4.3(e) we see that another camera that was participating in the negotiation decided that it could view the target at a high resolution at t_{out} and therefore took over the tracking of the target. Again, this camera is now in charge of tracking the high-resolution target predicting its position. The other cameras, one more time, adjust their parameters to cover the entire area at an acceptable resolution. This process is repeated until the target exits the area of interest as seen in Fig. 4.3(f). Once the target has left the area under surveillance completely, all the cameras continue with the negotiation maximizing their utilities to keep the entire area covered at an acceptable resolution.

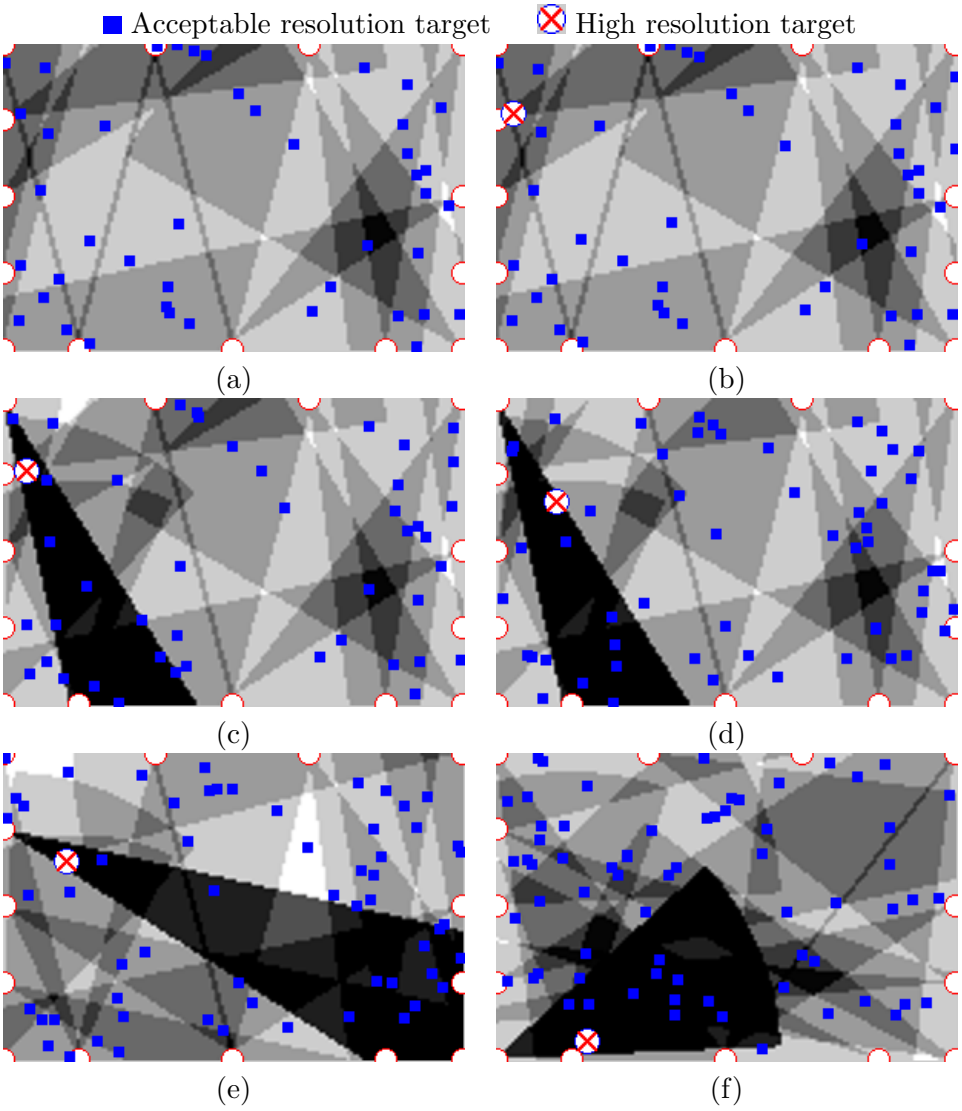


Figure 4.3: Simulated area under surveillance with several targets visible at an acceptable resolution and one high-resolution target. (a) shows the area of interest being completely covered by the cameras at an acceptable resolution r_0 (gray shaded areas are covered, white areas are not) and therefore all the targets in the area are being viewed and tracked at r_0. In (b), a target (T_k, r_h) has been selected for tracking at a high resolution r_h. C_h, the camera with the dark FOV in (c), decided that it could track (T_k, r_h) and adjusted its FOV accordingly. Some of the other cameras readjusted their parameters to maximize their utilities and the area is again entirely covered. (T_k, r_h) is about to exit C_h's FOV in (d). Therefore, C_h broadcasts the estimated exit time and position of (T_k, r_h) to all the cameras to hand off the target. (e) shows how another camera decided that it could view (T_k, r_h) at the estimated position and time. Note how in (e) there is a small part of the area uncovered which is quickly covered again through the cameras' negotiation mechanisms. The entire process above is repeated until (T_k, r_h) exits the area as in (f).

Real-life Data

We also showed experiments with a real-life network of cameras. The area under surveillance is a outdoor region of approximately 15 by 20 meters. The sensor network consists of 6 PTZ cameras with a resolution of 320x240 pixels. Same as experiments on simulated data, we divide the area into grids of size 1x1 meter and treat each grid of the area as a virtual target. Initially, the camera parameters (pan/tilt/zoom) are assigned arbitrarily (see left panel of Figure 4.4) so that the area under surveillance is not entirely covered by the cameras' FOVs.

We use the game theoretic approach described in Algorithm 2 to determine an initial set of camera parameters to cover the entire area at an acceptable resolution. Figure 4.4 shows the initialization results. The results after initialization are shown on the right. It is clear that the entire area has been covered after initialization.

Figure 4.5 shows the results of camera control on a specific task. When one person (bounded with red box)is selected to be viewed at a higher resolution, one camera zooms in, and other cameras automatically adjust their settings to keep the entire area covered. When the highlighted person moves out of the view of his binding camera, a hand off is automatically achieved in our game theory framework.

Figure 4.4: Initialization results using proposed game theory based approach. The original settings are shown on the left, the results after initialization are shown on the right. It is clear that the entire area has been covered after initialization.

4.6.2 Distributed Tracking Using Kalman-Consensus Filter

We tested our approach for tracking and camera self-configuration in a real camera network composed of 10 PTZ cameras looking over an outdoor area of approximately 10000 sq. feet. We divided the area into contiguous blocks and each of the centers of those blocks

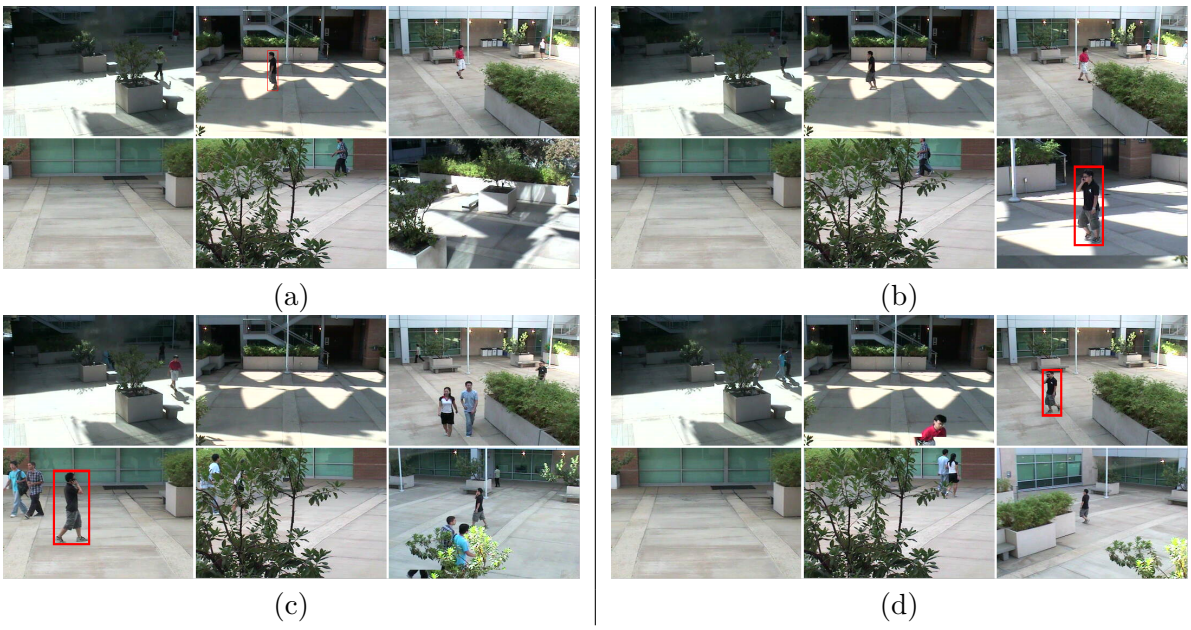
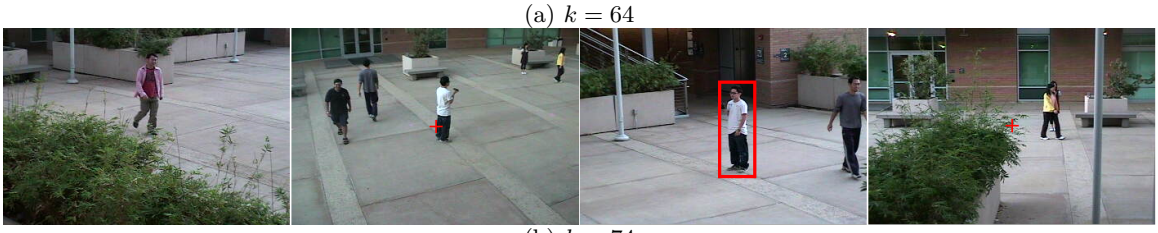


Figure 4.5: Results of camera control on a specific task. When one person (bounded with red box)is selected to be viewed at a higher resolution, one camera zooms in, and other cameras automatically adjust their setting to keep the entire area covered. When the highlighted person moves out of the view of his binding camera, a hand off is automatically achieved in our game theory framework. Notice the changes in the camera settings over time. The video is available at http://www.ee.ucr.edu/ amitrc/CameraNetworks.htm .

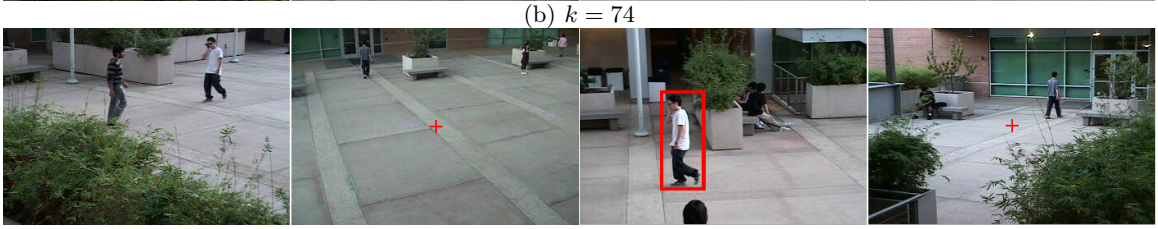
is considered a virtual target of height 1.70m. Therefore, if the camera network as a whole is covering all of the virtual targets in each block at an acceptable resolution $r_0 = 40$ pixels in the vertical direction, all of the actual targets in the area under surveillance are also being viewed at an acceptable resolution.

In the area under surveillance, there were 8 targets in total that were to be tracked using our distributed Kalman-Consensus filtering approach. Figure 4.6 shows the tracking and control results as viewed by each camera at 4 time instants. Due to space constraints, we show only 4 $(C_1, ..., C_4)$ of the 10 cameras in the camera network. At an initial state, all of the cameras have random PTZ parameters and are not covering the entire area under surveillance. After the cameras start running their control modules, they converge to the final configuration seen partly in Figure 4.6(a) covering the entire area. Once the targets start coming into the area, the single-view tracking module in each camera determines the ground plane position of each target in its FOV and sends that information to the

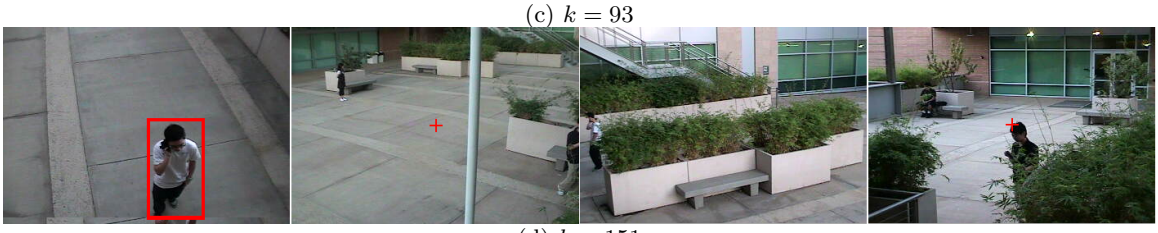
(a) $k = 64$



(b) $k = 74$



(c) $k = 93$



(d) $k = 151$

Figure 4.6: Each column shows one of 4 of the 10 cameras at four time instants denoted by k. A target marked with a box is always tracked at a high resolution. Note that the camera parameters are changing to achieve this while covering the entire area at an acceptable resolution. The other targets are tracked using the Kalman-Consensus filtering approach, but are not marked for clarity. The video is available at http://www.ee.ucr.edu/ amitrc/CameraNetworks.htm .

Kalman-Consensus filter which processes it together with the information received from the Kalman-Consensus filters of neighboring cameras as described in Section 4.5. Figure 4.6(b) shows the instant when a camera C_3 zoomed into a target T_l^h that was marked for being tracked at a high resolution. Since C_3 changed its parameters and left some area it was covering uncovered, the other cameras in the network adjust their own parameters to cover that uncovered area following the negotiation mechanisms described above. These changes in parameters can be seen in Figure 4.6(b) and (c) for cameras C_3 and C_4. As shown in Figure 4.6(d), when C_3 is not able to change its parameters to keep tracking T_l^h, C_1 takes over the tracking and C_3 changes its parameters to cover some area that was left uncovered by C_1. It is to note that every time a target goes from one camera's FOV into another one, or when a camera changes its parameters, the network topologies for the targets change also.

Figure 4.7 shows the distributed Kalman-Consensus tracking trajectories for the 8 targets. The observations of the different cameras are shown in a light gray color. As can be seen, even though the homography-based observations are noisy and the network topology is changing constantly, the Kalman-Consensus filter in each camera comes to a smooth consensus about the actual state of the target.

Figure 4.8 shows the distributed tracking results in the y ground plane direction for one of the targets. The dots correspond to the observations from the different cameras viewing the target while the solid line is the consensus-based estimate. As can be expected, the observations are different for each camera due to calibration and single-view tracking inaccuracies. The vertical dashed lines indicate a change in the dynamic network topology with respect to the target, i.e. when the target goes into or out of a camera's FOV. As can be seen clearly, even though different combinations of cameras view the target at different time instants, the Kalman-Consensus filter finds an estimate of the target's position seamlessly at all times.

4.7 Conclusion

We presented in this chapter a robust approach to distributed multi-target tracking in a network of self-configuring cameras. A novel camera control framework based on game theoretical ideas allowed for viewing some targets at a high resolution while keeping the

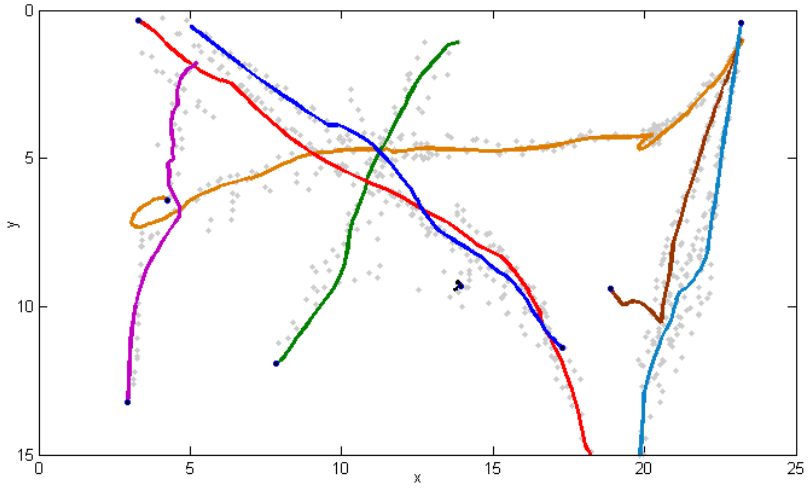


Figure 4.7: Distributed Kalman-Consensus tracking trajectories for 8 targets. Observations from all cameras are shown in a light gray color.

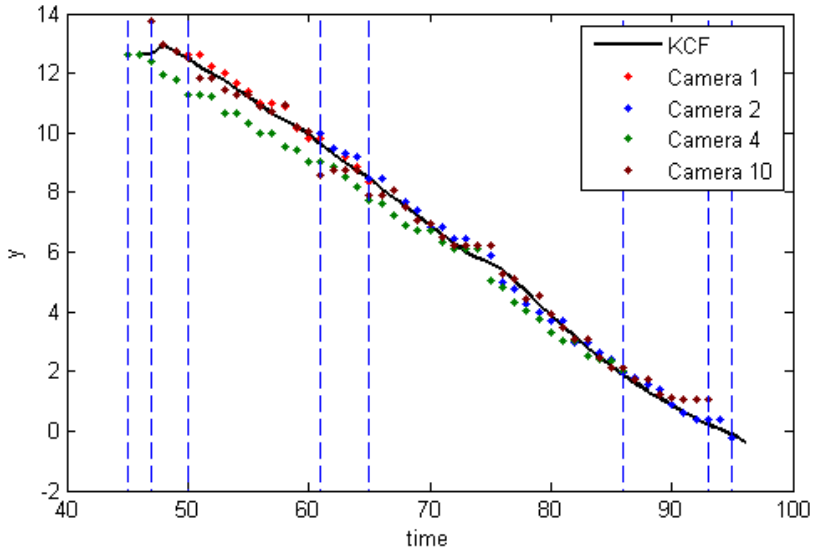


Figure 4.8: Tracking results in the y ground plane direction for one of the targets.

entire area under surveillance covered at an acceptable resolution. This is achieved through the appropriate choice of utility functions and negotiation mechanisms. Simulation and real-life results show that this method can be effectively utilized for decentralized camera network control. A distributed Kalman-Consensus filtering approach was used together with a dynamic network topology for tracking persistently multiple targets across several camera views in an area under surveillance viewed by a dynamic camera network.

Chapter 5

Distributed Communication

5.1 Introduction

Transmission of video data from multiple sensors over a wireless network requires enormous amount of bandwidth, and could easily overwhelm the system. However, by exploiting the redundancy *between* the video data collected by different cameras, in addition to the inherent temporal and spatial redundancy *within* each video sequence, the required bandwidth can be significantly reduced. Well-established video compression standards, such as MPEG1, MPEG2, MPEG4, H261, and H263, all rely on efficient transform coding of motion-compensated frames, exclusively using the discrete cosine transform (DCT). However, they can only be used in a protocol that encodes the data of each sensor independently. Such methods would exploit spatial and temporal redundancy within each video sequence, but would completely ignore the significant redundancy between the sequences.

In this chapter, we develop the framework for a novel multi-terminal video coding (MTVC) algorithm combining distributed source coding (DSC) and computer vision techniques that impose geometrical constraints between the observed views of the two cameras. This lossy compression scheme takes into account the correlation between the video sensor data, and at the same time keeps the communication between the sensors at a minimum. In broad terms, our scheme relies on alignment of the 2-D video sequences using the epipolar geometry [33] relating the cameras, *which could be located arbitrarily in space*, followed by standard elimination of spatial and temporal redundancy, and finally by quantization of the transform coefficients in a distributed fashion [87]. The epipolar geometry refers to the

relationship between the positions of a stereo camera pair, and can be estimated from a pair of stereo images obtained from these cameras. We propose a distributed motion estimation (DME) that is able to obtain correspondences between the cameras, in addition to computing the temporal motion vectors between the frames. The performance of our algorithm depends, most crucially, on the quality of alignment and the coding efficiency of the distributed quantization scheme. The alignment must result in correspondences between pixels that are maximally correlated, and the distributed coding must optimally exploit this correlation.

We begin by reviewing some of the fundamentals of DSC and describe our MTVC algorithm. We then elaborate on the performance measures and analyze the results. We perform a thorough experimental evaluation of MTVC and achieve the following objectives: (i) We provide a benchmark dataset and performance measures which represent an upper bound on the gain obtained through distributed coding of a pair of video sequences, as opposed to coding each of them separately; (ii) We show the performance of some simple schemes and where they stand with respect to this optimal limit; (iii) We identify the major issues that need to be addressed in future work on MTVC in order to achieve the highest compression rates. In particular, we show that MTVC is capable of providing significant benefits in low bit-rate coding.

We would like to emphasize that the novelty of the proposed scheme lies in the ability to integrate distributed source coding concepts with video analysis tools that exploit the inherent geometrical constraints between two views. There is a significant amount of work that is possible in both these areas. It is possible to integrate more advanced techniques for source coding and motion estimation. While that is the focus of much of our current work, we present a basic algorithm that lays the groundwork for integrating these two parts together.

5.2 Previous Work

There has recently been significant effort in application of DSC techniques to video data. However, in what is broadly known as distributed video coding (e.g., [28, 72, 77]), DSC is utilized only for the exploitation of *temporal* correlation in a *single* video stream, leading to a low-complexity encoder and better error resilience. A recent method [90]

attempts to exploit the redundancy between images available at *different* sensor nodes by independently encoding the images in low resolution and decoding using super-resolution techniques. However, this gain is orthogonal to the coding gains promised by DSC, which could be achieved by exploiting the high correlation between the low-resolution images. Therefore higher coding gains promised by multiterminal source coding theory [5, 68, 78] are not reached. Another recent work [25, 26] developed a distributed image coding technique for a multi-camera setting. Under certain assumptions on the location of the cameras (e.g., the cameras are located along a horizontal line, the objects are within a certain known range from the cameras, and the image intensity field is piecewise polynomial), they derived a lower bound on the minimum number of cameras required to perfectly reconstruct a scene. For image-based rendering applications, [106] exhibits a successful algorithm for Wyner-Ziv coding of the light field whereby complexity is shifted from the encoders to the decoder, but geometrical relationships between camera positions are not taken into account. A recent paper considers inter- and intra-sensor correlation for transmission after making some simplifying assumptions about the locations of the sensors and not making explicit use of the geometrical transformations relating the images in various sensors [12]. Another recent work [96] also addresses multiterminal video coding in its most general sense. However, the technique used for finding corresponding points in the two views is computationally involved, and requires, for example, efficient depth estimation. For a setting where feedback from the central decoder to one of the encoders is allowed, [21] provides a mechanism for efficient exploitation of inter-sensor redundancy.

5.3 Transform Coding of Distributed Sources

The fundamental ingredient of DSC, both in lossless and lossy cases, is *binning* [5, 78], i.e., a many-to-one mapping of the actual data taken from the sources to a limited number of values. Through binning, the correlation between the sources can be exploited without any communication between the sensors.

For two maximally correlated pair of blocks from each view, we use the discrete cosine transform followed by distributed scalar quantization of transform coefficient pairs. Coefficient pairs corresponding to each fixed spatial frequency are encoded independently. Our scalar coding method is provably competitive (in the sense of approaching the rate-

distortion bounds) in high bit rates, which is a promising result for the intended (lower bit-rate) applications.

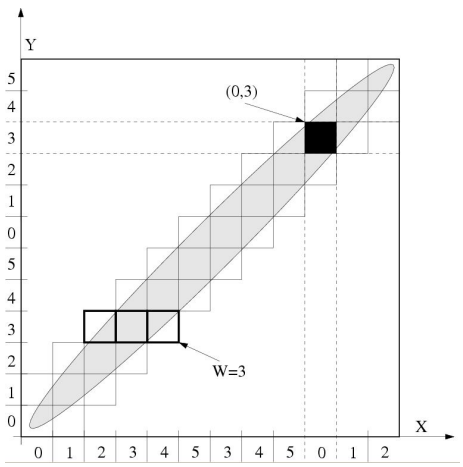


Figure 5.1: Proposed coding scheme with $W = 3$, $N_X = 2$, and $N_Y = 2$. As can be seen, 3 cells indicated in bold, suffice to cover the support of the source, indicated as the shaded area, everywhere.

Let the shaded region shown in Figure 5.1 indicate the *support* of a pair of transform coefficients $\tilde{X}$ and $\tilde{Y}$ we need to quantize. It will suffice to design a coding mechanism which encodes the scalars that are inside the support with a small enough distortion, and simply ignore any pair of values falling outside. The encoding must be performed separately, and therefore the cells used for the covering must consist of Cartesian products of individual intervals. The particular assignment in Figure 5.1 indeed ensures indispensable *unique decodability*, as each pair of codewords pinpoint to a *single* cell that is used in the covering of the support. The example codeword pair shown in the figure, {0,3}, actually corresponds to 4 different cells, but only one of those has a high probability of occurring, and therefore is used for the covering of the support, i.e., as the decoded output. The same statement can be made for all codeword pairs in $\{0,\ldots,5\}\times\{0,\ldots,5\}$.

As in [87], we consider a family of codes parameterized by three integers, W, N_X and N_Y. The dynamic ranges of both $\tilde{X}$ and $\tilde{Y}$ are divided into $W \times N_X \times N_Y$ *intervals*, thereby defining a grid on the two dimensional plane. The achieved fixed-length coding rates for the $\tilde{X}$- and $\tilde{Y}$-encoders are $\lceil \log_2 WN_X \rceil$ and $\lceil \log_2 WN_Y \rceil$, respectively. For jointly Gaussian source pairs, we were able to analyze the performance of our scheme rigorously [87], which proved its competitiveness in two aspects: (i) under the uniform high-resolution quantization regime, by separate encoding of $\tilde{X}$ and $\tilde{Y}$, one can achieve the same total distortion one would achieve even if both $\tilde{X}$ and $\tilde{Y}$ were available at a single sensor node [27, Section 8.3], and (ii) under high-resolution assumption, this simple binning technique can attain total rates as close as 3.05 bits to the asymptotical rate-distortion bound characterized in [68].

5.4 An Overview of the MTVC Algorithm

Though the issues discussed in the previous sections regarding efficient transform and coding of the data can be applied to any sensor network where nodes observe correlated sequences and are required to transmit their findings to a central receiver, our main focus will be on distributed video compression. Multi-terminal video compression stands to gain significantly from DSC due to the inherently high bandwidth of video. Video compression deals with encoding a video sequence after removing the spatial redundancy in each video frame and the temporal redundancy between the frames. Standards such as MPEG1, MPEG2, MPEG4, H261, and H263, outline procedures for achieving this purpose. If we have multiple video sequences from different cameras where there is a significant overlap between the sequences, the above coding standards are inefficient since they do not consider the redundancy in the data at the different sensors. Under this situation, it is necessary to develop distributed video compression schemes that can take advantage of the fact that the data from different sources are correlated. Moreover, this should be done without too much communication between the sensors. Otherwise the savings in bandwidth obtained by considering correlated sources would be offset by the inter-sensor communication.

The theory outlined in the previous section provides an excellent framework to design a distributed lossy video compression scheme. The sensors may be viewing the scene from different viewpoints, but they may have a significant portion of the scene where their field of views (FOVs) overlap. On this overlapping portion, we intend to achieve a very high compression rate using the distributed source coding principles discussed so far. The portion in the frames where there is no correspondence between them will be intra-coded. The geometry between the locations of the sensors will be exploited to understand the correlation between the data at different sensor nodes. We provide below a detailed strategy for developing a distributed video compression strategy for pairs of sensors. Extending it to K sensor nodes will be an issue of future research.

Our scheme relies on obtaining correspondence between the macroblocks (MBs) of the two sensor data at any time instant. Reliable tracking will result in maximally correlated MBs, which, in turn, will lead us to develop an efficient distributed coding scheme. The task of tracking the correspondence of MBs will be achieved by using the motion vectors (MVs) together with the geometrical constraints between the sensors [33]. The MVs can be

computed by any scheme, e.g., as in MPEG. The geometrical relationships are expressed through the epipolar constraint. The epipolar constraint states that given a point in one view (say the left image), its corresponding point in the other view lies on the epipolar line. This reduces the search for correspondences to a 1D problem, provided we can compute the epipolar line. This, in turn, requires information about the camera calibration parameters, i.e. the intrinsic parameters of the camera (we will assume that the focal length is the only intrinsic parameter of interest), as well as the extrinsic parameters (i.e., the position and orientation of the camera reference frame with respect to a fixed reference frame in the world). For this chapter, we assume stationary cameras, which means that the calibration parameters can be estimated from the images obtained from two or more sensors. Though the present scheme is described for a pair of cameras, it can be generalized to larger sets using the multi-camera constraints [33] or dealing with the cameras pairwise.

5.5 MTVC Algorithm

We provide below an overview of our proposed approaches for (i) tracking the correspondence between MBs in the two sequences obtained from cameras A and B (the Distributed Motion Estimation (DME) algorithm), and (ii) compressing the two frames in a distributed fashion using the correspondences.

5.5.1 The Distributed Motion Estimation (DME) Algorithm

Let us label the two video cameras as A and B. We assume, for the purposes of this explanation, that we have calibrated cameras and Camera A knows its position relative to Camera B. We also assume that the two cameras were initially synchronized at time $t = t_1$ so that the corresponding MBs are known at Camera A. The synchronization needs to be done (by Camera A using the initial frames captured by both) at the very start of the transmission, and needs to be refreshed periodically.

• **Problem Definition and Assumptions**: The correspondence between the macroblocks of I_{A1} and I_{B1} is known at Camera A (see Fig. 5.2). The problem is to compute the correspondence between I_{A2} and I_{B2} also at Camera A. The MVs of both pairs of frames $\{I_{A1}, I_{A2}\}$ and $\{I_{B1}, I_{B2}\}$ are computed separately using an MPEG encoding scheme. That is, both frames I_{A2} and I_{B2} are divided into a uniform grid of MBs and MVs are computed

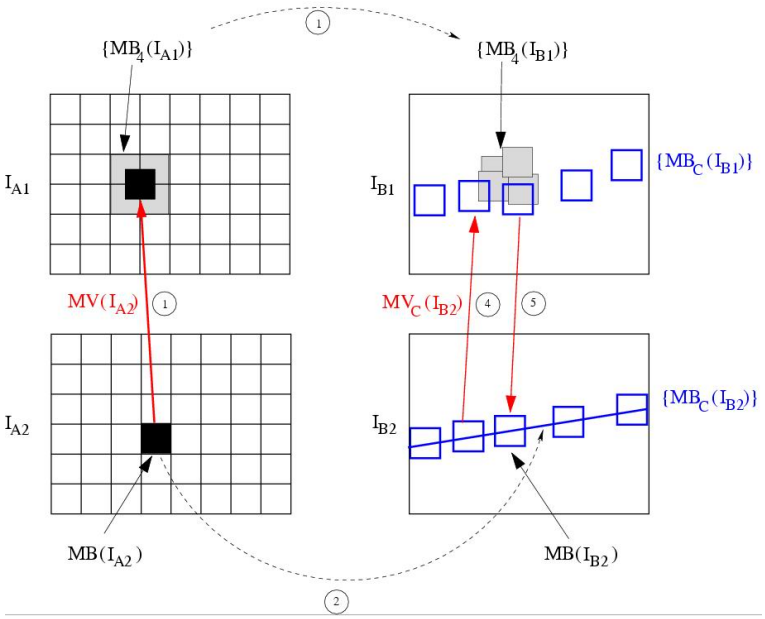


Figure 5.2: Pictorial description of the DME. The numbers in circles indicate the steps of the algorithm.

per macroblock. Let us denote by $\{MV(I_{A2})\}$ the set of MVs from I_{A2} to I_{A1}, and similarly by $\{MV(I_{B2})\}$ those from I_{B2} to I_{B1}. The algorithm is repeated for each macroblock at Camera A.

• **Inter-camera Communication:** Camera B transmits its MVs to Camera A, which uses this information to track which MB in Camera B's view corresponds to the current MB in its own view. Although this information is primarily useful at the decoding stage, it must necessarily be performed at the encoder at Camera A to classify MBs into two classes: overlapping and non-overlapping. If the encoder decides that the current MB is in the overlapping region, it uses DSC. Otherwise, it uses conventional compression.

• **The Premise of the Algorithm:** If a macroblock $MB(I_{A2})$ in I_{A2} is in the overlapping region, then it must have a matching macroblock in I_{B2} centered somewhere on the epipolar line corresponding to the center of $MB(I_{A2})$. In order to find exactly where, we utilize the motion vectors $\{MV(I_{A2})\}$ and $\{MV(I_{B2})\}$ to map both the macroblock $MB(I_{A2})$ and the epipolar line in the previous frame, and use the correspondences already computed between I_{A1} and I_{B1}.

• **The Step-by-Step Algorithm:** A visual description of the steps below is provided in

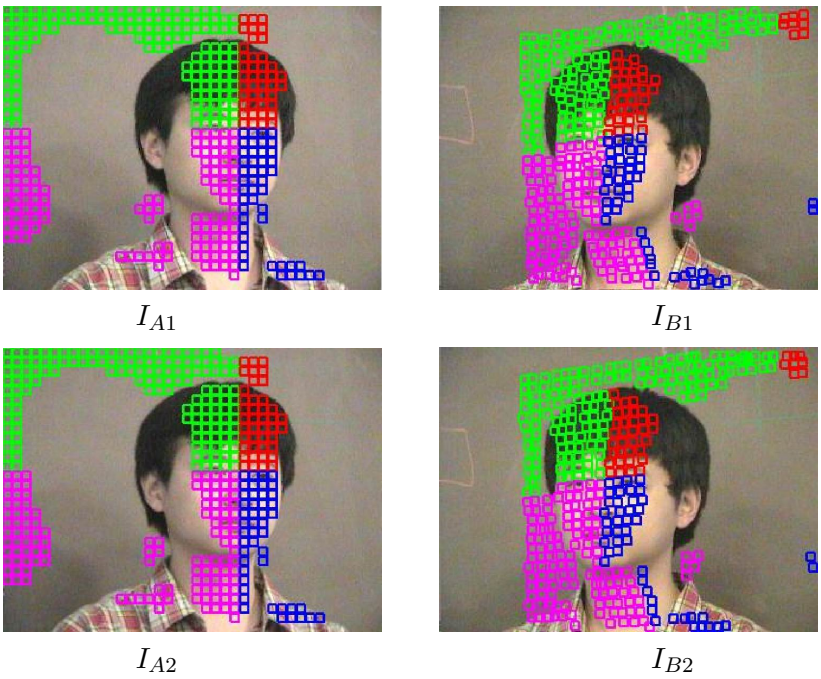


Figure 5.3: MBs on a grid in the left (Camera A) view I_{A1} correspond to indicated blocks in the right (Camera B) view I_{B1} of the first frame. Tracking results are shown on frames I_{A2} and I_{B2}. There is about a 30% overlap between the views, which is not accounted for in traditional single source video encoding strategies, such as MPEG. (The picture is best viewed in color.)

Fig. 5.2.

1. For each macroblock $MB(I_{A2})$, the motion vector $MV(I_{A2})$ leads to four possible MBs in I_{A1}, denoted by $\{MB_4(I_{A1})\}$. Camera A obtains the corresponding four MBs, $\{MB_4(I_{B1})\}$, using the correspondence between I_{A1} and I_{B1}.

2. Using the epipolar constraint, Camera A computes the epipolar line for the center pixel in $MB(I_{A2})$. Denote this line by $EP(I_{B2})$.

3. Sampling the epipolar line (possibly non-uniformly), Camera A creates a sequence of MBs, denoted by $\{MB_C(I_{B2})\}$, the center of each being a sample point on the line. Using $\{MV(I_{B2})\}$ (which it has now access to), it then interpolates to obtain the MVs of this sequence of MBs, denoted by $\{MV_C(I_{B2})\}$.

4. Using $\{MV_C(I_{B2})\}$, Camera A obtains the corresponding sequence of MBs in I_{B1}, denoted by $\{MB_C(I_{B1})\}$.

5. Among the sequence of MBs $\{MB_C(I_{B1})\}$, Camera A chooses the one with the highest

amount of intersection with $\{MB_4(I_{B1})\}$. This MB is then traced back to the frame I_{B2}, and the resultant MB, denoted by $MB(I_{B2})$, is declared to correspond to $MB(I_{A2})$. If there is zero or very small overlap between $\{MB_C(I_{B1})\}$ and $\{MB_4(I_{B1})\}$, it is declared that $MB(I_{A2})$ no longer has a correspondence in I_{B2}.

6. The process is repeated for every MB in I_{A2}. This establishes a correspondence between macroblocks of I_{A2} and I_{B2}. We can now increment the time counter and go back to the Step 1.

Observe that to track correspondence of MBs, the DME algorithm, which runs on Camera A, utilizes *only* the motion information of Camera B, i.e., it need not access the image data of Camera B. Fig. 5.3 shows preliminary results obtained by this algorithm. The top row shows the overlapping MBs (about 30% of the image) in I_{A1} and I_{B1}, while the bottom row shows the tracking result on these MBs using the DME algorithm. The small amount of inter-camera communication can be completely removed if we have a 3D model of the object which can be used to obtain the correspondences between the two views.

5.5.2 Distributed Coding of Corresponding MBs

For both frames IA_2 and IB_2, we will first apply motion compensation to reduce temporal redundancy. However, instead of standard motion compensation $MC(t) = MB(t) - MB(t-1)$, where $MB(t)$ denotes a generic MB and $MB(t-1)$ is the corresponding MB in the previous frame, we propose to adopt a non-unity prediction coefficient $0 < p < 1$ and use $MC(t) = MB(t) - pMB(t-1)$. This non-standard approach is advantageous in two aspects. First, it will provide us with better error resilience, as errors caused by channel erasures will be forgotten quickly (i.e., effectively in a few frames if p is small enough). This, in turn, will also help defer the next I-frame. Second, the corresponding MBs in the two views will enjoy much higher correlation. This can be proved theoretically using a model (such as a Gauss-Markov process [87]). We have also observed this phenomenon in practice.

The residual obtained at Camera B is encoded as in any conventional video coding scheme, e.g., MPEG or H.26x. That is, the residual MBs are transformed using the discrete cosine transform (DCT) and encoded. The non-overlapping portion of Camera A's residual sequence is also encoded using this conventional approach. For the overlapping portion in Camera A's view, however, we propose to use a distributed scalar quantization scheme for

the top transform coefficients, and conventional coding for the rest of the coefficients.

As in any DSC scheme, our algorithm is based on binning of quantization coefficients. We propose a simple algebraic binning scheme based on modulo-W arithmetic, where we term W the "*correlation width*" of the support of the joint distribution governing the transform coefficient pair in corresponding MBs. Fig. 5.1 depicts the proposed scheme with a simple example.

5.6 Performance Evaluation & Analysis

5.6.1 Experiments With Controlled Data

We conducted a detailed evaluation of the performance of this algorithm on a synthesized video sequence with known motion estimates and inter-camera correspondences (see Fig. 5.4) [1]. This allows us to benchmark the upper bound on the savings obtained by our MTVC algorithm under ideal conditions, thus showing clearly the achievable gains. These are the limits practical MTVC algorithms should strive to achieve on this sequence. Then we used simple schemes for estimating the correlation width, W of DSC for corresponding MBs and compared the results against the maximum possible gain. Finally, we computed the gain from DSC, as opposed to separate coding, for the face sequence in Fig. 5.3. From these experiments, the overwhelming conclusion to be drawn is that *MTVC can be very efficient at low bit-rates*, and the challenge for the future is to develop smart strategies for motion estimation and coding. In some ways, this parallels efforts in conventional video compression to develop efficient motion estimation and coding algorithms. We now present the details.

Experiment 1

The goal of the experiment was to quantify the gains obtained from MTVC compared to separate coding, when exact values of motion estimates, inter-camera correspondence and correlation width are available. The reference image (also referred to as zero-degree) was held fixed as Camera A while the Camera B images were obtained at various angles. For each pair, we computed the PSNR vs. bit-rate for separate coding and dis-

[1] This data will be made available to the broad community.

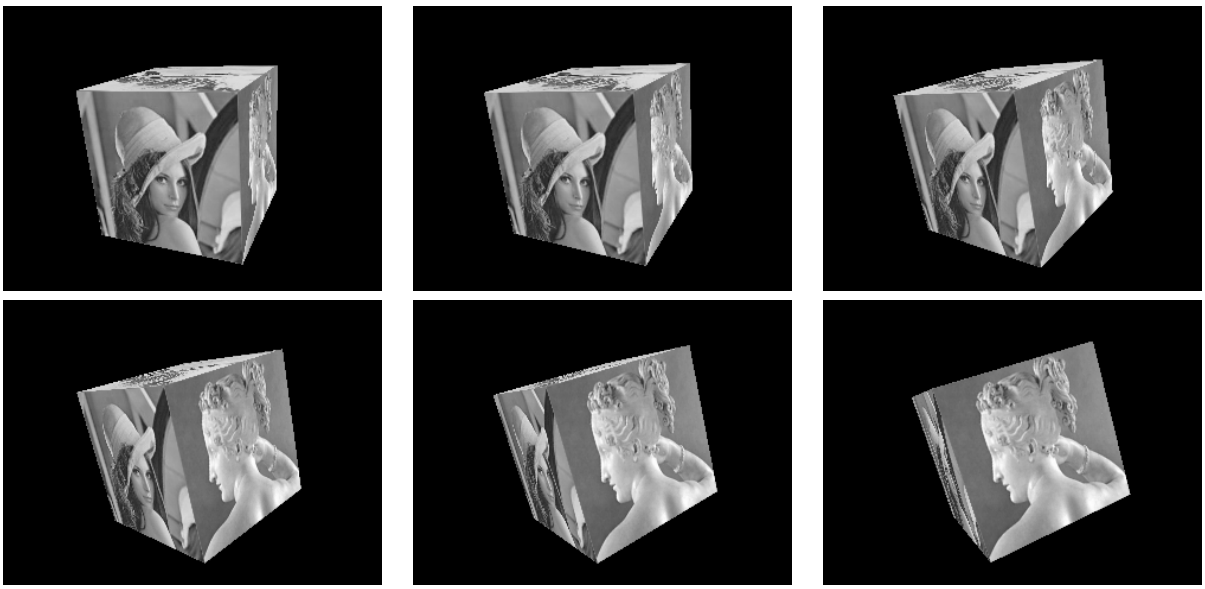


Figure 5.4: Frames of synthesized video sequences were generated with a separation of 5, 15, 25, 35 and 45 degrees from a reference.

tributed coding. The difference was approximately constant for each angle. This is to be expected since for a constant joint distribution, we must increase W by the same factor as we increase the quantization level, N, to keep the unique decodability intact. In other words, W/N must be proportional to the thickness of the equi-probability curve. Since the difference is constant, the gain in bit-rate (for a fixed distortion), expressed as a percentage of the bit-rate for separate coding, is very high for low bit-rates and decreases as the bit-rate increases. For example, in the case that the angle between Camera A and Camera B is 15 degree, the percentage of bit-rate savings decreases from about 63% to 19% as the bit-rate increases from 90 to 620 Kbps (see Fig. 5.5 (b)). This shows the very high potential of MTVC at low bit-rates.

Experiment 2

The goal of this experiment was to test the accuracy of simple schemes for computing the correlation width, W automatically. We again chose pairs as before for separate and distributed coding. The ground truth for motion estimates was used. The "correlation width", W, was chosen manually in the first frame. Thereafter, W for the n^{th} frame was estimated from the corresponding macroblocks in the $(n-1)^{\text{th}}$ frame of the same camera.

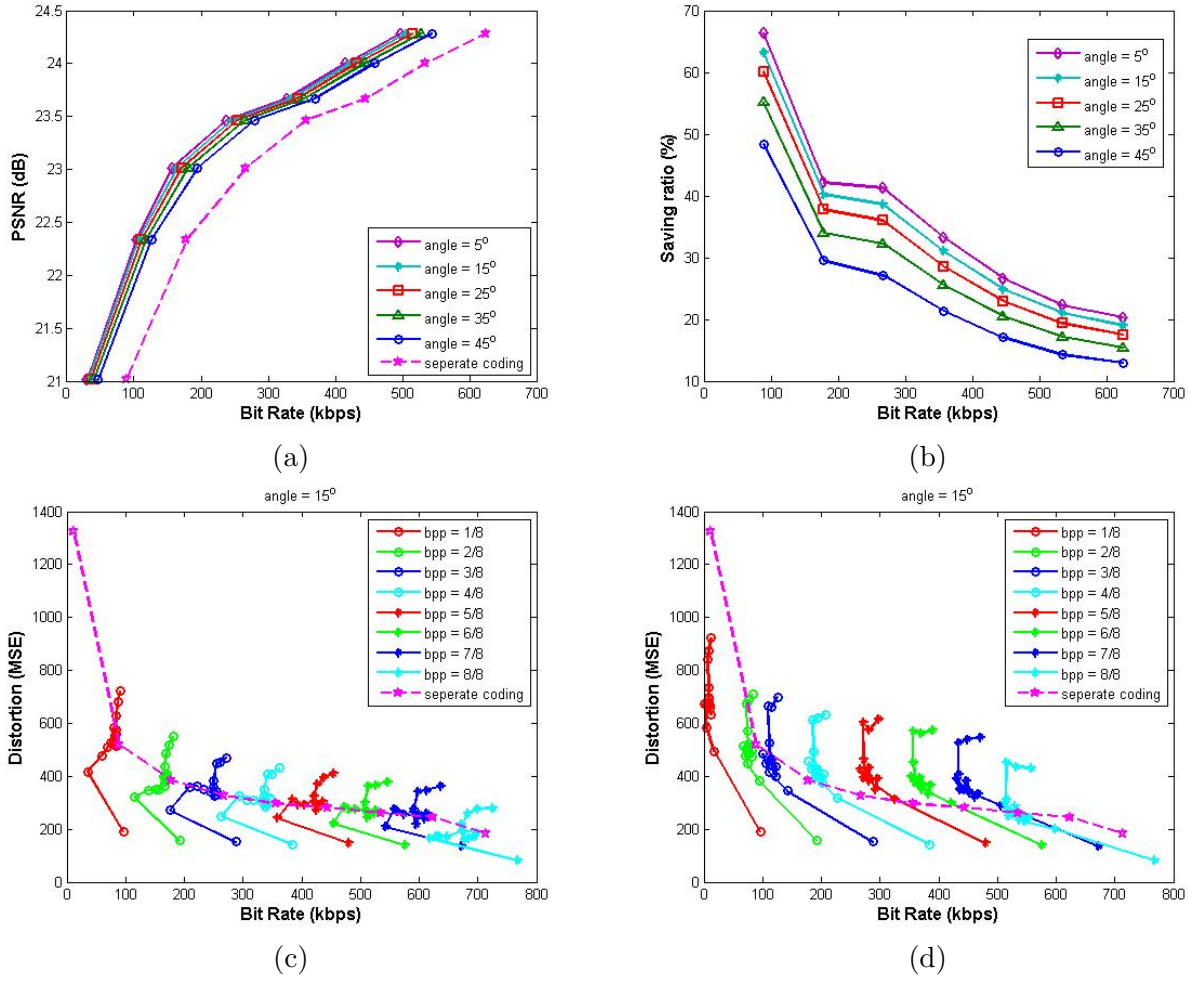


Figure 5.5: (a): Rate distortion curves at various angles with known values of W, compared against separate coding. (b): The gain in bit-rate (for a fixed distortion), expressed as a percentage of the bit-rate for separate coding. (c) (d): The PSNR vs. bit-rate for each frame (shown as a bullet) for the zero-fifteen degree pair. The maximum values of the Ws in the corresponding MBs in the $(n-1)^{\text{th}}$ frame (c) and a fixed W value (d) were used.

The maximum, median and mean values of the Ws in the corresponding macroblocks in the $(n-1)^{\text{th}}$ frame and a fixed value was used. The fixed W value was chosen as the median value of the Ws in the first frame. We plot the PSNR vs. bit-rate for each frame for the zero-fifteen degree pair. The curve for separate coding is also plotted (see Fig. 5.5 (c)(d)). Each distributed coding curve starts at the same bit-rate as the separate coding and then adjusts depending upon the estimated values of W. We find that a number of frames stay below the separate coding curve, but then cross over. This suggests that the main challenge is *efficient coding schemes so that more frames stay below the separate coding curve*, thus yielding a larger gain from MTVC compared to separate coding. We also need methods to reinitialize W automatically, just as motion estimates are reinitialized using an I-frame in conventional video coding.

5.6.2 An Experiment With Face Video Sequences

The goal of this experiment was to plot the maximum achievable gain using MTVC on the face images of Fig. 5.3. We used the DME algorithm to obtain the correspondences. Thereafter, we chose a constant value of W based on an analysis of the corresponding MBs. The curve in Fig. 5.7 plots the savings in bit-rate using MTVC, compared to separate coding of the two channels. This again shows that significant savings are obtained at lower bit-rates. Methods to automatically estimate W should be of such an accuracy that the savings indicated by this curve can be reached on this dataset. Some of the original and decoded face images are shown in Fig. 5.6. These images were obtained with a bit-rate of about 89Kbps and a PSNR of 28dB.

5.7 Conclusions

We have designed a novel distributed lossy video compression scheme that takes into account the correlation between the video sensor data, and at the same time keeps the communication between the sensors at a minimum. Using epipolar geometry relating the video sensors, the correspondence between the macroblocks of multiple views can be tracked at any time instant, with minimal communication between the sensors. After finding corresponding macroblocks, a suitable transform, and a quantization of the transform coefficients in a distributed fashion are applied to eliminate spatial and inter-sensor redundancy. Using

Figure 5.6: Examples of original (top) and decoded (bottom) frames of the face sequence. Frame numbers 2, 7, 12 and 17 of a sequence are shown.

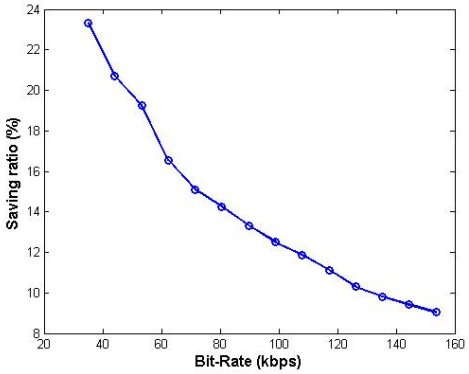


Figure 5.7: The gain in bit-rate (for a fixed distortion), expressed as a percentage of the bit-rate for separate coding for the face data.

the distributed coding we achieve a better compression rate on the overlapping portion of multiple views.

In general, it is a subject of future work to consider how this scheme can give better performance. First, more sophisticated algorithms can be used for DSC. The computational simplicity of our scalar coding scheme can be traded off with more complex mechanisms which use LDPC-based or turbo-coding-based algorithms [1, 3, 7, 23, 24, 53, 76, 102].

The accuracy of the video processing algorithms would also be studied so as to reduce errors due to loss of synchronization, misalignment of macroblocks and segmentaion of the common region between two views. We will also consider extension to moving cameras, which will require recalibration of the sensors. Extension to more than two sensors will be another problem of the future research.

Chapter 6

Conclusion and Future Work

In this thesis, we studied the core issues in network-centric processing, control and communication in a multi-camera network, including tracking people in video through changes of activities, tracking in a non-overlapping camera network, decentralized control and tracking in a camera network, and distributed video compression.

In Chapter 2, we proposed a novel framework for tracking multi-activity sequences without knowing the transition probabilities between the activities. We demonstrated this for applications using only one camera, as well as in a network of non-overlapping cameras. Using a non-linear dynamical feedback system, we showed that our method is able to continuously track sequences of different activities amidst changing background, occlusions and clutter. Measures were designed to detect changes between activities. We demonstrated the effectiveness of our approach by doing a detailed performance analysis and showing experimental results on real life videos of different activities.

A novel multi-objective optimization framework for tracking multiple people in a large network of non-overlapping video cameras was proposed in Chapter 3. We considered the uncertainty of feature similarity estimation 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 using discriminant analysis to correct for wrong correspondences. We showed that the above two conditions can be addressed by treating the issue of tracking in a camera network as a multi-objective optimization problem which could be solved using a stochastic adaptive strategy. We demonstrated the effectiveness of our system by showing

results on a real life camera networks.

Consider the difficulty in analyzing a huge amount of data centrally, we presented a robust approach to distributed multi-target tracking in a network of self-configuring cameras. A novel camera control framework based on game theoretical ideas allowed for viewing some targets at a high resolution while keeping the entire area under surveillance covered at an acceptable resolution. This is achieved through the appropriate choice of utility functions and negotiation mechanisms. Simulation and real-life results show that this method can be effectively utilized for decentralized camera network control. A distributed Kalman-Consensus filtering approach was used together with a dynamic network topology for tracking persistently multiple targets across several camera views in an area under surveillance viewed by a dynamic camera network.

We have designed a novel distributed lossy video compression scheme that takes into account the correlation between the video sensor data, and at the same time keeps the communication between the sensors at a minimum. Using epipolar geometry relating the video sensors, the correspondence between the macroblocks of multiple views can be tracked at any time instant, with minimal communication between the sensors. After finding corresponding macroblocks, a suitable transform, and a quantization of the transform coefficients in a distributed fashion are applied to eliminate spatial and inter-sensor redundancy. Using the distributed coding we achieve a better compression rate on the overlapping portion of multiple views.

A number of promising avenues of research are suggested by the work presented in this thesis, and a few of them are detailed here. Consider the scenario that more than one target is selected to be tracked at a high resolution, it would be interesting to study the relationship between the number of high-resolution targets and the number of cameras needed to cover an entire area under surveillance. Intuitively, as the number of high-resolution targets increases, the number of cameras needed should also increase. But it may not be smart to assign a camera for each individual target, especially when the targets number is large. It would be interesting to find out the relationship.

It is also a subject of future work to consider how the distributed video compression scheme can give better performance. First, more sophisticated algorithms can be used for DSC. The computational simplicity of our scalar coding scheme can be traded off with more

complex mechanisms which use LDPC-based or turbo-coding-based algorithms [1, 3, 7, 23, 24, 53, 76, 102]. The accuracy of the video processing algorithms would also be studied so as to reduce errors due to loss of synchronization, misalignment of macroblocks and segmentaion of the common region between two views. We will also consider extension to moving cameras, which will require recalibration of the sensors. Extension to more than two sensors will be another problem of the future research.

Bibliography

[1] A. Aaron and B. Girod. Compression with side information using turbo codes. In *IEEE Proceedings of the Data Compression Conference*, 2002.

[2] G. Arslan, J. Marden, and J. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *ASME JDSMC*, 129(5), September 2007.

[3] J. Bajcsy and P. Mitran. Coding for the Slepian-Wolf problem with turbo codes. In *Proceedings of GLOBECOM*, 2001.

[4] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[5] T. Berger. Multiterminal source encoding. In G. Longo, editor, *The Information Theory Approach to Communications*, number 229 in CISM Courses and Lectures. Springer, New York, 1978.

[6] O. Boiman and M. Irani. Detecting irregularities in images and in video. In *Proc. of International Conf. on Computer Vision*, 2005.

[7] F. Cabarcas and J. Garcia-Frias. Approaching the Slepian-Wolf boundary using practical channel codes. In *IEEE International Symposium on Information Theory*, 2004.

[8] Y. Censor. Pareto Optimality in Multiobjective Problems. *Appl. Math. Optimiz.*, 4:41–59, 1977.

[9] T.-J. Cham and J. M. Rehg. A multiple hypothesis approach to figure tracking. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[10] T. F. Cootes and C. J. Taylor. Active shape models - "smart snakes". In *Proc. British Machine Vision Conf.*, 1992.

[11] N. O. DaCunha and E. Polak. Constrained Minimization Under Vector-valued Criteria in Finite Dimensional Spaces. *J. Math. Anal. Appl.*, 19:103–124.

[12] J. C. Dagher, M. W. Marcellin, and M. A. Neifeld. A method for coordinating the distributed transmission of imagery. *IEEE Trans. on Image Processing*, 2006.

[13] Doucet,A. and Freitas,N.de and Gordon,N. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[14] I. Dryden and K. Mardia. *Statistical Shape Analysis*. John Wiley and Sons, 1998.

[15] W. Du and J. Piater. Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In *ACCV*, pages I: 365–374, 2007.

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

[17] M. Ehrgott. *Multicriteria Optimization*. Berlin ; New York : Springer, 2000.

[18] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vis. Image Underst.*, 103(3):156–169, 2006.

[19] G. D. Finlayson, B. Schiele, and J. L. Crowley. Comprehensive colour image normalization. In *Proc. of European Conference on Computer Vision*, 1998.

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

[21] M. Flierl and B. Girod. Coding of multi-view image sequences with video sensors. In *Proc. of Intl. Conf. on Image Processing*, 2006.

[22] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. Series on Economic Learning and Social Evolution. MIT Press, Cambridge, MA, 1998.

[23] J. Garcia-Frias and Y. Zhao. Compression of correlated binary sequences using turbo codes. *IEEE Communications Letters*, 5(10):417–419, 2001.

[24] J. Garcia-Frias and W. Zhong. Ldpc codes for compression of multi-terminal sources with hidden markov correlation. *IEEE Communications Letters*, 7(3):115–117, 2003.

[25] N. Gehrig and P. L. Dragotti. DIFFERENT: DIstributed and Fully Flexible image EncodeRs for camEra sensor NeTworks. In *Proc. of Intl. Conf. on Image Processing*, 2005.

[26] N. Gehrig and P. L. Dragotti. Distributed Sampling and Compression of Scenes with Finite Rate of Innovation in Camera Sensor Networks. In *Data Compression Conference*, 2006.

[27] A. Gersho. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, MA, 1993.

[28] B. Girod, A. Margot, S. Rane, and D. Rebollo-Monedero. Distributed video coding. In *Proceedings of the IEEE*, 2005.

[29] W. Grimson, L. Lee, R. Romano, and C. Stauffer. Using Adaptive Tracking to Classify and Monitor Activities in a Site. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 22–31, 1998.

[30] F. Guo and G. Qian. Sample-Efficiency-Optimized Auxiliary Particle Filter. In *Proceedings of IEEE Workshop on Statistical Signal Processing*, 2005.

[31] A. Gupta, A. Mittal, and L. Davis. Cost: An approach for camera selection and multi-object inference ordering in dynamic scenes. In *IEEE ICCV*, 2007.

[32] R. Hamid and J. A. Detection and explanation of anomalous activities: representing activities as bags of event n-grams. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[33] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[34] M. Harville and D. Li. Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages II: 398–405, 2004.

[35] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Trans. on Systems, Man and Cybernetics - Part C: Applications and Reviews*, 34(3):334–352, August 2004.

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

[37] L. Itti and P. Baldi. A principled approach to detecting surprising events in video. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[38] J. Jackson, A. Yezzi, and S. Soatto. Tracking deformable moving objects under severe occlusions. In *IEEE Conf. Decision and Control (CDC)*, 2004.

[39] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking across multiple cameras with disjoint views. In *IEEE Intl. Conf. on Computer Vision*, 2003.

[40] O. Javed, K. Shafique, and M. Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[41] V. Jilkov, X. Li, and D. Angelova. Estimation of Markovian Jump Systems with Unknown Transition Probabilities through Bayesian Sampling. *Lecture Notes in Computer Science*, 2542:307–315, 2003.

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

[43] J. Kang, I. Cohen, and G. Medioni. Continuous tracking within and across camera streams. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[44] V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[45] S. Khan, O. Javed, Z. Rasheed, and M. Shah. Camera Handoff: Tracking in Multiple Uncalibrated Stationary Cameras. In *IEEE Workshop on Human Motion*, 2000.

[46] S. Khan and M. Shah. A multiview approach to tracking people in dense crowded scenes using a planar homography constraint. In *Proc. of International Conf. on Computer Vision*, 2005.

[47] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *ECCV*, 2006.

[48] A. Lanterman, M. Miller, and D. Snyder. The Unification of Detection, Tracking, and Recognition for Millimeter Wave and Infrared Sensors. *SPIE*, 2562:150–161, July 1995.

[49] W. Leoputra, T. Tan, and F. L. Lim. Non-overlapping distributed tracking using particle filter. In *Intl. Conf. on Pattern Recognition*, 2006.

[50] Y. Li and B. Bhanu. Utility-based dynamic camera assignment and hand-off in a video network. *ICDSC*, 2008.

[51] Y. Li and T. Boult. Understanding Images of Graphical User Interfaces: A new approach to activity recognition for visual surveillance. In *ACM UIST*, 2003.

[52] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using relational markov networks. In *Proc. of the International Joint Conference on Artificial Intelligence*, 2005.

[53] A. Liveris, Z. Xiong, and C. Georghiades. Compression of binary sources with side information at the decoder using ldpc codes. *IEEE Communications Letters*, 6(10):440–442, 2002.

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

[55] D. Makris, T. Ellis, and J. Black. The gaps between cameras. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[56] D. Marinakis, G. Dudek, and D. Fleet. Learning sensor network topology through monte carlo expectation maximization. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[57] D. Markis, T. Ellis, and J. Black. Bridging the gap between cameras. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[58] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1996.

[59] H. Medeiros, J. Park, and A. Kak. Distributed object tracking using a cluster-based kalman filter in wireless camera networks. *Selected Topics in Signal Processing, IEEE Journal of*, 2(4):448–463, Aug. 2008.

[60] D. Monderer and L. Shapley. Potential games. In *Games and Economic Behavior*, 1996.

[61] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui. Learning and Detecting Activities From Movement Trajectories Using the Hierarchical Hidden Markov Model. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[62] C. Niu and E. Grimson. Recovering non-overlapping network topology using far-field vehicle tracking. In *Intl. Conf. on Pattern Recognition*, 2006.

[63] B. North, A. Blake, M. Isard, and J. Rittscher. Learning and classification of complex dynamics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(9):1016–1034, 2000.

[64] R. Olfati-Saber. Distributed kalman filter with embedded consensus filters. *44th IEEE Conference on Decision and Control*, 2005.

[65] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007.

[66] R. Olfati-Saber and N. F. Sandell. Distributed tracking in sensor networks with limited sensing range. *Proceedings of the American Control Conference*, June 2008.

[67] N. Oliver, B. Rosario, and A.P.Pentland. A Bayesian computer vision system for modeling human interactions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):831–843, August 2000.

[68] Y. Oohama. Gaussian multiterminal source coding. *IEEE Transactions on Information Theory*, 43(6):1912–1923, 1997.

[69] A. Papoulis. *Probabbility, Random Variables and Stochastic Processes*. McGraw-Hill, Inc., 1991.

[70] S. Park and J. K. Aggarwal. A hierarchical bayesian network for event recognition of human actions and interactions. *Multimedia System*, 10:164–179, 2004.

[71] M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94:590–599, 1999.

[72] R. Puri and K. Ramchandran. Prism: A video coding architecture based on distributed compression principles. *IEEE Transactions on Image Processing*.

[73] F. Qureshi and D. Terzopoulos. Surveillance in virtual reality: System design and multi-camera control. *CVPR*, 2007.

[74] A. Rahimi and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[75] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi. Particle Filtering for Geometric Active Contours with Application to Tracking Moving and Deforming Objects. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[76] D. Schonberg, K. Ramchandran, and S. S. Pradhan. Distributed code constructions for the entire Slepian-Wolf rate region for arbitrarily correlated sources. In *IEEE Proceedings of the Data Compression Conference*, 2004.

[77] A. Sehgal, A. Jagmohan, and N. Ahuja. Wyner-ziv coding of video: An error-resilient compression framework. *IEEE Transactions on Multimedia*, 6(2):249–258, 2004.

[78] D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480, 1973.

[79] B. Song, C. Soto, A. K. Roy-Chowdhury, and J. A. Farrell. Decentralized camera network control using game theory. *Workshop on Smart Camera and Visual Sensor Networks at ICDSC*, 2008.

[80] B. Song, N. Vaswani, and A. Roy-Chowdhury. Closed-loop tracking and change detection in multi-activity sequences. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[81] B. Song, N. Vaswani, and A. K. Roy-Chowdhury. Summarization and Indexing of Human Activity Sequences. In *Proc. of Intl. Conf. on Image Processing*, 2006.

[82] K. Tieu, G. Dalley, and E. Grimson. Inference of non-overlapping camera network topology by measuring statistical dependence. In *IEEE Intl. Conf. on Computer Vision*, 2005.

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

[84] K. Toyama and B. A. Probabilistic Tracking in a Metric Space. In *Proc. of International Conf. on Computer Vision*, 2001.

[85] P. Tresadern and I. Reid. Articulated structure from motion by factorization. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1110–1115, June 2005.

[86] R. Tron, R. Vidal, and A. Terzis. Distributed pose averaging in camera networks via consensus on se(3). *ICDSC 2008*, Sept. 2008.

[87] E. Tuncel. Predictive coding of correlated sources. In *IEEE Information Theory Workshop*, 2004.

[88] N. Vaswani and R. Chellappa. NonStationary Shape Activities. In *IEEE Conf. Decision and Control (CDC)*, 2005.

[89] N. Vaswani, A. Roy-Chowdhury, and R. Chellappa. Shape Activities: A Continuous State HMM for Moving/Deforming Shapes with Application to Abnormal Activity Detection. *IEEE Trans. on Image Processing*, October 2005.

[90] R. Wagner, R. Nowak, and R. Baranuik. Distributed image compression for sensor networks using correspondence analysis and superresolution. In *Proc. of Intl. Conf. on Image Processing*, 2003.

[91] D. Wilson and C. Atkeson. Simultaneous Tracking and Activity Recognition (STAR) Using Many Anonymous, Binary Sensors. In *Proceedings of PERVASIVE*, 2005.

[92] D. Wolpert and K. Tumor. An overview of collective intelligence. In *Handbook of Agent Technology*. AAAI Press/MIT Press, 1999.

[93] X. Xu and B. Li. Adaptive Rao-Blackwellized particle filter and its evaluation for tracking in surveillance. *IEEE Trans. on Image Processing*, 16(3):838–849, 2007.

[94] J. Yan and M. Pollefeys. A factorization-based approach to articulated motion recovery. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 815–821, June 2005.

[95] C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. In *Proc. of International Conf. on Computer Vision*, 2005.

[96] Y. Yang, V. Stankovic, W. Zhao, and Z. Xiong. Multiterminal video coding. In *Information Theory and Applications Workshop*, 2007.

[97] A. J. Yezzi and S. Soatto. Deformation: Deforming motion, shape average and joint registration and approximation of structure in images. *International Journal of Computer Vision*, 53(2):153–167, 2003.

[98] A. Yilmaz and M. Shah. Recognizing Human Actions in Videos Acquired by Uncalibrated Moving Cameras. In *Proc. of International Conf. on Computer Vision*, 2005.

[99] H. Young. *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, 1998.

[100] L. Zedeh. Optimality and Nonscalar-valued Performance Criteria. *IEEE Trans. Automat. contr.*, AC-8, 1963.

[101] T. Zhao and R. Nevatia. Tracking multiple humans in crowded environment. *Computer Vision and Pattern Recognition*, II:406–413, 2004.

[102] Y. Zhao and J. Garcia-Frias. Data compression of correlated nonbinary sources using punctured turbo codes. In *IEEE Proceedings of the Data Compression Conference*, 2002.

[103] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[104] S. Zhou, R. Chellappa, and B. Moghaddam. Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters. *IEEE Trans. on Image Processing*, 13(11):1491–1506, November 2004.

[105] S. K. Zhou, R. Challappa, and B. Moghaddam. Visual tracking and recognition using appearance adaptive models in particle filters. *IEEE Trans. on Image Processing*, 13(11):1491–1506, 2004.

[106] X. Zhu, A. Aaron, and B. Girod. Distributed compression for large camera arrays. In *IEEE Workshop on Statistical Signal Processing*, 2003.

Appendix A

Tracking Error and Bayesian Hypothesis Testing

In this Appendix, we will show, using Bayesian Hypothesis Testing, that the tracking error using an incorrect model is greater than using a correct model under a set of reasonable assumptions.

Consider that in the shape vector of Section 2.2, we treat each landmark separately. Then for each of them we have dynamical system:

$$\begin{aligned} x_{t+1} &= Ax_t + v_t, \\ z_t &= f(x_t) + n_t \end{aligned} \tag{A.1}$$

Assume that:

$$x_t \sim \mathcal{N}(x_0, \sigma_x),\ n_t \sim \mathcal{N}(0, \sigma_n)$$

The output of tracking algorithm is

$$\begin{aligned}
\hat{\theta}_t &= E[f(x)|z_t] \quad \text{where} \quad f(x) = cx + d \\
&= \int_{x_t} f(x)p(x_t|z_t)dx_t \\
&= \frac{1}{p(z_t)} \int_{x_t} f(x)p(z_t|x_t)p(x_t)dx_t \\
&= \frac{1}{p(z_t)} \int_{x_t} f(x) \frac{1}{\sqrt{2\pi\sigma_n^2}} \frac{1}{\sqrt{2\pi\sigma_n^2}} \\
&\cdot \quad \exp\left(-\frac{1}{2}\left(\frac{(z_t - f(x_t))^2}{\sigma_n^2} + \frac{(x_t - x_0)^2}{\sigma_x^2}\right)\right) dx_t
\end{aligned} \tag{A.2}$$

Considering the exponent only

$$\begin{aligned}
&\frac{(z_t - f(x_t))^2}{\sigma_n^2} + \frac{(x_t - x_0)^2}{\sigma_x^2} \\
= \quad &\frac{(z_t - (cx + d))^2}{\sigma_n^2} + \frac{(x_t - x_0)^2}{\sigma_x^2} \\
= \quad &\frac{(x_t - \bar{x}_0)^2}{\sigma_{\bar{x}}^2} + \bar{d}
\end{aligned} \tag{A.3}$$

where

$$\bar{x}_0 = \left(\frac{1}{\sigma_x^2} + \frac{c^2}{\sigma_n^2}\right)^{-1} \left(\frac{c}{\sigma_n^2}(z_t - d) + \frac{x_0}{\sigma_x^2}\right) \tag{A.4}$$

$$\sigma_{\bar{x}}^2 = \frac{1}{\frac{1}{\sigma_x^2} + \frac{c^2}{\sigma_n^2}} \tag{A.5}$$

$$\bar{d} = \frac{(z_t - d)^2}{\sigma_n^2} + \frac{x_0^2}{\sigma_x^2} - \frac{\bar{x}_0^2}{\sigma_x^2}. \tag{A.6}$$

Then

$$\begin{aligned}
\hat{\theta}_t &= \frac{1}{p(z_t)} \int_{x_t} (cx + d) \frac{1}{\sqrt{2\pi\sigma_n^2}} \frac{1}{\sqrt{2\pi\sigma_x^2}} \sqrt{2\pi\sigma_{\bar{x}}^2} \\
&\cdot \quad \frac{1}{\sqrt{2\pi\sigma_{\bar{x}}^2}} \exp\left(-\frac{(x_t - \bar{x}_0)^2}{2\sigma_{\bar{x}}^2}\right) \exp(-\bar{d}/2) dx_t \\
&= \frac{1}{p(z_t)} \frac{1}{\sqrt{2\pi\sigma_n^2}} \frac{1}{\sqrt{2\pi\sigma_x^2}} \sqrt{2\pi\sigma_{\bar{x}}^2} \exp(-\bar{d}/2) \\
&\cdot \quad \int_{x_t} (cx + d) \frac{1}{\sqrt{2\pi\sigma_{\bar{x}}^2}} \exp\left(-\frac{(x_t - \bar{x}_0)^2}{2\sigma_{\bar{x}}^2}\right) dx_t
\end{aligned} \tag{A.7}$$

Since

$$\begin{aligned} p(z_t) &= \int_{x_t} p(z_t|x_t)p(x_t)dx_t \\ &= \int_{x_t} \frac{1}{\sqrt{2\pi\sigma_n^2}} \frac{1}{\sqrt{2\pi\sigma_n^2}} \\ &\cdot \quad \exp\left(-\frac{1}{2}\left(\frac{(z_t - f(x_t))^2}{\sigma_n^2} + \frac{(x_t - x_0)^2}{\sigma_x^2}\right)\right) dx_t \\ &= \frac{1}{\sqrt{2\pi\sigma_n^2}} \frac{1}{\sqrt{2\pi\sigma_x^2}} \sqrt{2\pi\sigma_{\bar{x}}^2} \exp(-\bar{d}/2) \\ &\cdot \quad \int_{x_t} \frac{1}{\sqrt{2\pi\sigma_{\bar{x}}^2}} \exp\left(-\frac{(x_t - \bar{x}_0)^2}{2\sigma_{\bar{x}}^2}\right) dx_t \\ &= \frac{1}{\sqrt{2\pi\sigma_n^2}} \frac{1}{\sqrt{2\pi\sigma_x^2}} \sqrt{2\pi\sigma_{\bar{x}}^2} \exp(-\bar{d}/2) \end{aligned} \tag{A.8}$$

Thus

$$\hat{\theta}_t = c\bar{x}_0 + d \tag{A.9}$$

Now consider 2 alternative models with the same form of the dynamical equations, but

$$\begin{aligned} x_t &\sim \mathcal{N}(x_{01}, \sigma_{x1}) \text{ under Model 1} \\ \text{and } x_t &\sim \mathcal{N}(x_{02}, \sigma_{x2}) \text{ under Model 2} \\ z_t &\sim \mathcal{N}(cx_0 + d, c^2\sigma_x + \sigma_n^2) \end{aligned}$$

$$\begin{aligned} p(z_t|\mathrm{M}_1) &= \frac{1}{\sqrt{2\pi c_1^2\sigma_{x1}^2 + \sigma_n^2}} \exp\left(-\frac{1}{2}\frac{(z_t - (c_1x_{01} + d))^2}{c_1^2\sigma_{x1}^2 + \sigma_n^2}\right) \\ p(z_t|\mathrm{M}_2) &= \frac{1}{\sqrt{2\pi c_2^2\sigma_{x2}^2 + \sigma_n^2}} \exp\left(-\frac{1}{2}\frac{(z_t - (c_2x_{02} + d))^2}{c_2^2\sigma_{x2}^2 + \sigma_n^2}\right) \end{aligned}$$

Using Bayes' Hypothesis Theory, if M_1 is the "correct" model, then

$$\begin{aligned}
& p(z_t|\mathrm{M}_1) > p(z_t|\mathrm{M}_2) \\
\Rightarrow \quad & \frac{1}{\sqrt{2\pi(c_1^2\sigma_{x1}^2+\sigma_n^2)}}\exp\left(-\frac{1}{2}\frac{(z_t-(c_1x_{01}+d_1))^2}{c_1^2\sigma_{x1}^2+\sigma_n^2}\right) \\
> \quad & \frac{1}{\sqrt{2\pi(c_2^2\sigma_{x2}^2+\sigma_n^2)}}\exp\left(-\frac{1}{2}\frac{(z_t-(c_2x_{02}+d_2))^2}{c_2^2\sigma_{x2}^2+\sigma_n^2}\right) \\
\Rightarrow \quad & \frac{(z_t-(c_1x_{01}+d_1))^2}{c_1^2\sigma_{x1}^2+\sigma_n^2}+\log(c_1^2\sigma_{x1}^2+\sigma_n^2) < \frac{(z_t-(c_1x_{02}+d_2))^2}{c_2^2\sigma_{x2}^2+\sigma_n^2}+\log(c_2^2\sigma_{x2}^2+\sigma_n^2)
\end{aligned} \tag{A.10}$$

$$\begin{aligned}
z_t-\hat{\theta}_t &= z_t-(c\bar{x}_0+d) \\
&= z_t-\left(\frac{c^2\sigma_x^2(z_t-d)+cx_0\sigma_n^2}{\sigma_n^2+c^2\sigma_x^2}+d\right) \\
&= \frac{\sigma_n^2}{\sigma_n^2+c^2\sigma_x^2}(z_t-(cx_0+d))
\end{aligned} \tag{A.11}$$

Thus

$$\begin{aligned}
||z_t-\hat{\theta}_{t1}||^2 &= \left(\frac{\sigma_n^2}{\sigma_n^2+c_1^2\sigma_{x1}^2}\right)^2(z_t-(c_1x_{01}+d_1))^2 \\
||z_t-\hat{\theta}_{t2}||^2 &= \left(\frac{\sigma_n^2}{\sigma_n^2+c_2^2\sigma_{x2}^2}\right)^2(z_t-(c_2x_{02}+d_2))^2
\end{aligned} \tag{A.12}$$

If c_1, c_2 and σ_1,σ_2 satisfy

$$c_1^2\sigma_{x1}^2 \geq c_2^2\sigma_{x2}^2 \tag{A.13}$$

then from (A.10) we have

$$\begin{aligned}
& \frac{(z_t-(c_1x_{01}+d_1))^2}{c_1^2\sigma_{x1}^2+\sigma_n^2} < \frac{(z_t-(c_1x_{02}+d_2))^2}{c_2^2\sigma_{x2}^2+\sigma_n^2} \\
\Rightarrow \quad & \frac{(z_t-(c_1x_{01}+d_1))^2}{(c_1^2\sigma_{x1}^2+\sigma_n^2)^2} < \frac{(z_t-(c_1x_{02}+d_2))^2}{(c_2^2\sigma_{x2}^2+\sigma_n^2)^2} \\
\Rightarrow \quad & \left(\frac{\sigma_n^2}{\sigma_n^2+c_1^2\sigma_{x1}^2}\right)^2(z_t-(c_1x_{01}+d_1))^2 < \left(\frac{\sigma_n^2}{\sigma_n^2+c_2^2\sigma_{x2}^2}\right)^2(z_t-(c_2x_{02}+d_2))^2 \\
\Rightarrow \quad & ||z_t-\hat{\theta}_{t1}||^2 < ||z_t-\hat{\theta}_{t2}||^2
\end{aligned} \tag{A.14}$$

Then the overall tracking error

$$\sum_{i=1}^{K} ||z_t^i - \hat{\theta}_{t1}^i||^2 < \sum_{i=1}^{K} ||z_t^i - \hat{\theta}_{t2}^i||^2 \tag{A.15}$$

Thus, we show that the tracking error using an incorrect model is greater.