UNIVERSITY OF CALIFORNIA
RIVERSIDE

Coordinated Sensing in Intelligent Camera Networks

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

Doctor of Philosophy

in

Computer Science

by

Chong Ding

September 2013

Dissertation Committee:

Dr. Amit K. Roy-Chowdhury, Chairperson
Dr. Jay A. Farrell
Dr. Chinya Ravishankar
Dr. Walid A. Najjar
Dr. Victor B. Zordan

The Dissertation of Chong Ding is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

Completing my Ph.D degree has been a long and challenging activity. It has been a great privilege to spend several years in the Bourns College of Engineering. There have

First I would like to thank my advisor Amit K. Roy-Chowdhury, Jay A. Farrell, who has effectively been my co-advisor in all but name. They patiently provided the encouragement and advice I needed to proceed through the doctoral program. I would also like to thank Victor B. Zordan, for his support and guidance through my undergraduate and graduate years, and the rest of my committee, Chinya Ravishankar and Walid A. Najjar for their guidance and advice. I am also grateful to thank Bi Song, who provided significant guidance through the early days of my research and assisted in my first publications.

Other members of the Video Computing Lab also deserve my sincerest thanks for their friendship and assistance in my research. I am thankful for my all of my friends who have supported me through the best and worst moments of my studies: Adam Dou, Nick Tate and Curtis Yu. I should also mention all the members of our local ACM student chapter who were sources of joy and support. I would also like to especially thank Xiaoqing Jin for always being there to listen and provide comfort during all the rough times, and her unwavering belief in me.

The text of this dissertation is partly a reprint of the material that appears in "Collaborative Sensing In A Distributed PTZ Camera Network" (Chong Ding, Bi Song, Akshay A. Morye, Jay A. Farrell and Amit K. Roy-Chowdhury) 2012, "Coordinated Sensing and Tracking for Mobile Camera Platforms" (Chong Ding, Akshay A. Morye, Jay A. Farrell and Amit K. Roy-Chowdhury) 2012, and "Opportunistic Sensing in a Distributed PTZ Camera Network" (Chong Ding, Bi Song, Akshay A. Morye, Jay A.

Farrell and Amit K. Roy-Chowdhury) 2012. I am grateful to my co-author Ackshay A. Morye for our many discussions and assistance in my research.

Finally, I would like to thank all the administrative and technical staff, Amy Ricks, Victor Hill and Danny Haughton, who have helped me though all sorts of administrative and computer system problems.

To my parents, Wan-Xiang Li and Shou-Wei Ding.

ABSTRACT OF THE DISSERTATION

Coordinated Sensing in Intelligent Camera Networks

by

Chong Ding

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2013
Dr. Amit K. Roy-Chowdhury, Chairperson

The cost and size of video sensors has led to camera networks becoming pervasive in our lives. However, the ability to analyze these images efficiently is very much a function of the quality of the acquired images. Human control of pan-tilt-zoom (PTZ) cameras is impractical and unreliable when high quality images are needed of multiple events distributed over a large area. This dissertation considers the problem of automatically controlling the fields of view of individual cameras in a camera network responsible for improving situation awareness (e.g. where and what are the critical targets and events) in a region of interest. This is achieved by understanding the performance of video analysis tasks and designing camera control strategies to improve these tasks through the quality of the source imagery. Optimization strategies, along with a distributed implementation, are proposed, and their theoretical properties analyzed. The proposed methods bring together computer vision and network control ideas.

The approach introduced here consists of a system wide utility measure and a distributed optimization method. The global utility function quantifies the system performance with respect to the goals of the user. The design of several utility functions is presented such as those for improving area coverage and tracking performance. These

utilities can then be combined to create a global utility measure aligned with the desired system behavior. Since exhaustive exploration of the parameter space of large camera networks is intractable, the problem is converted into a cooperative control problem where each camera optimizes local utility functions while negotiating with neighboring cameras. The result is a tractable optimization that increases the global utility until convergence at a local optima.

This approach has been applied to conventional surveillance tasks, such as observing targets in a large area, as well as more complicated tasks involving competing objectives. The performance of the proposed methodologies has also been evaluated on a real life wireless network of pan-tilt-zoom (PTZ) capable cameras.

Contents

List of Figures

Chapter 1

Introduction

Networks of video cameras are being installed for a variety of applications: surveillance and security, environmental monitoring and disaster response. As these sensors continue to progress and become more ubiquitous, smaller, faster, more powerful, their range of utilities will only continue to grow. The increasing capability of such systems leads to more complex applications and places even greater demand on the underlying infrastructure.

The decreasing cost and size of video sensors has led to camera networks becoming increasingly pervasive in our lives. Already, it is common to see video sensors in and around buildings, on robots, on phones, and soon, perhaps even on the things we wear. With many of these types of complex, personalized systems, an incredible amount of data needs to be processed and interpreted.

Existing camera networks often consist of fixed cameras covering large areas. This results in situations where the targets are often not covered at the desired resolutions or viewpoints, thus making analysis of the video difficult, especially when there are specified requirements associated with the targets, such as tracking precision, face

images for identification and so on. Since the total number of cameras is usually dictated by various factors (e.g., cost, ease of deployment) beyond video acquisition fidelity, a possible solution is to integrate the analysis and sensing tasks more closely. This can be achieved by introducing and controlling the parameters of pan-tilt-zoom (PTZ) cameras in the network to dynamically fulfill the analysis requirements. Dynamic target assignment would provide maximal utilization of the network, allowing the cameras to differentially focus on multiple regions of interest leading to effective feature acquisition. Such a setup would provide greater flexibility while requiring less hardware and being less costly.

The focus of most work in video analysis has been on improving the performance of detection, tracking and recognition algorithms in a variety of settings. However, large-scale experimental analysis is consistently demonstrating the limitations in the improvements that can be obtained by focusing solely on the processing side. Analysis of natural video databases that have been studied widely in the computer vision community shows that the performance is strongly dependent on the quality of the acquired video. However, research focused on video acquisition strategies driven by the need to maximize performance goals have been quite limited (a cursory glance through the programs of major vision and image processing conferences or the journal proceedings will show only a handful of methods that consider the acquisition aspects, as opposed to hundreds that focus on video analysis).

Sending the video streams of all cameras to a server or cloud can easily overwhelm the underlying communication infrastructure. A prominent example of this is the ban on video chat on AT & T mobile networks due to fears of congestion. To better scale these camera networks, the visual sensors would require the capability to analyze their own data and perform collaborative decision making in coordination with other

sensors. This would enable the sensors to maneuver themselves optimally, i.e., change the pan, tilt and zoom (PTZ) parameters of the cameras, so as to obtain image sequences that can be analyzed with a high degree of reliability. The optimization would be done during operation of the system, based on the analysis of sensed data from the immediate past, rather than the historic approach of positioning and orienting static visual sensors a priori.

The focus of this research is on dynamically controlling a distributed network of smart mobile camera platforms to complete a variety of important tasks. The first question considered was how to select the fields-of-view (FOV) of all the mobile cameras such that the entire region of interest is sufficiently observed by the camera network. Once the entire region is observable by the camera network, traditional video analysis solutions can be applied to perform detections, tracking, recognition, and other high level computer vision tasks. The second question considered was whether the accuracy of these high level tasks could be improved by dynamically reconfiguring the parameters of the network. Many of the existing video analysis algorithms perform very well with an exception on certain types of data. Could these problems be solved by directing video sensors to only provide useful input data. The third question was then, how to combine these tasks together. When should the cameras gather better data for recognition? When should they improve tracking? What is the best distribution of resources?

A simulation environment and a real world test-bed were used to develop sensing strategies for smart mobile camera networks. The real world test-bed consists of a wireless network of pan-tilt-zoom (PTZ) cameras in an outdoor university setting. Access to the cameras was provided solely through a web based API over the wireless network, meaning that any video or communication with the cameras was affected by the latency and bandwidth of the network. Ideally, the video processing and control would

be done locally on the camera, but due to the constraints of the real world test-bed, this was simulated by streaming the video to other computers for analysis.

1.1 Application Domains

Camera sensor networks have a wide range of applications. A few representative ones are described in the following sections..

1.1.1 Security and Surveillance

Surveillance is one of the primary applications of camera networks, where hundreds or even thousands of cameras monitor large public areas, such as airports, subways, etc. Since cameras usually provide raw video streams, it is difficult to interpret simultaneous video feeds manually. Initial machine or automated analysis of the data is highly desirable, and in some cases necessary, to guide and focus the operators attention. Also, since interesting events are rare the task can get very tedious. Thus it is desirable to utilize intelligent methods for extracting information from image data and come up with a meaningful representation for the user.

1.1.2 Environmental Monitoring

Camera networks can be used to monitor inaccessible areas remotely over a long period of time. Often in this scenario, the cameras are combined with other types of sensors, such that the cameras are triggered only when an event is detected by other sensors used in the network. An example could be monitoring farms to identify the wild animals attacking livestock.

1.1.3 Smart Environments

A smart environment is an area where different kinds of smart devices are continuously working together to aid humans in their daily activities [11]. Multiple sensors serve as "eyes" of the smart environments to capture in real-time the changing characteristics of the user and the environment. Visual sensors are very important components of the sensor network for their capability of capturing information-rich data. Representative examples of smart environments are smart meeting rooms and smart homes.

1.2 Thesis and Contributions

By developing optimal sensing strategies in a network of visual sensors, with a focus on tightly integrating the sensing and processing tasks, targets can be effectively imaged, tracked and identified.

Autonomous robots that can perform desired tasks in security, surveillance, and disaster response operations without continual human guidance are the ultimate goal of this research. Many of the environments in which such tasks occur are either dangerous or mundane to humans. For example, disaster regions can remain very hostile for those involved in search and rescue, while monitoring banks of surveillance cameras is mostly uneventful and quite boring.

However, developing robots capable of carrying out all the necessary tasks (e.g., searching for and tending to victims, neutralizing dangers) without human guidance is still far from realization. This contributions of this dissertation are thus focused on the tangible goal of robots with controllable video sensors, that operate *alongside* humans, to jointly accomplish a given mission.

Consider a scenario where a response team is deployed in a disaster zone. For the team to be effective, situation awareness (eg. where are the victims, adversaries, critical events) is of the utmost importance. In many such scenario's preexisting communication networks tend to be heavily disrupted requiring human and robotic members to rely on less powerful ad-hoc networks.

The question considered is how to select the fields of view (FOVs) of all the cameras such that the entire region of interest is covered. The design of utility functions quantifying the area covered by the camera network is discussed and a possible utility function is defined. A distributed control algorithm is used to optimize the settings of the camera network. Simulation and real life experiments show the benefits of an camera network able to dynamically adjust as the available sensors for performing coverage change over time.

Once the entire region is observed by the camera network detection, tracking, recognition and other high level computer vision algorithms can be performed. The question that immediately follows is whether the images, video and features used in the higher level algorithms can be improved by intelligently selecting the FOVs of the camera network. This dissertation shows how utility functions can be designed towards these system goals and provides a framework for dynamically optimizing the pan tilt zoom settings such that the tracking performance is improved. Experiments performed both in simulation and in a real life camera network show that both the tracking accuracy can be improved and that high quality images of targets can be acquired to use for data association and identification purposes.

The framework described in this dissertation also shows how multiple system goals competing for resources can be defined and optimized in a completely decentralized manner, where multiple smart cameras can decide and change their parameters. Also,

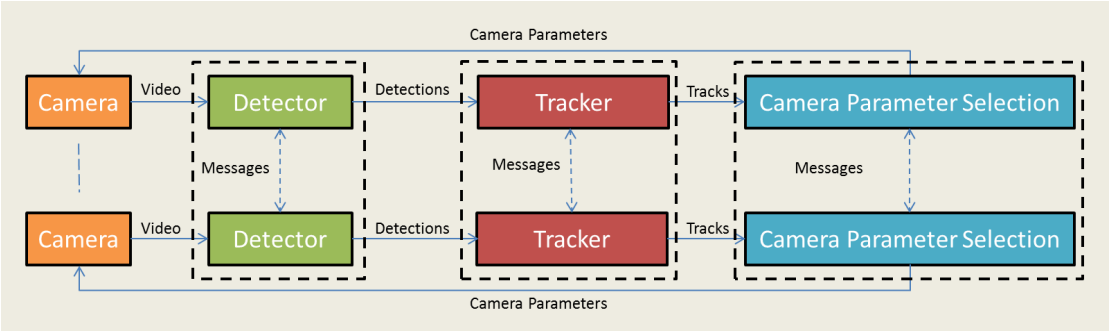


Figure 1.1: Diagram depicting the framework for integrating scene analysis and PTZ control.

the combined optimization of mobile camera platform routes and camera parameters, and the consideration of stochasticity in the target dynamics is presented. Simulation results show the behavior of mobile camera platforms tasked with optimally tracking a stochastic target.

1.3 Framework Overview

Configurations of cameras with large fields-of-view (FOV) can monitor a large area of the environment but may not be able to supply reliable images for recognition tasks. Configurations where the cameras are zoomed in on specific areas of interest can gather useful images for recognition, but result in a very limited view of the scene.

This framework is shown in Fig. (1.1) and can find application in any visual surveillance system containing PTZ cameras. The raw video from each of the cameras is first processed through a detector. Any resulting detections are then associated and used to update the target tracks by the tracker. These tracks are then used to decide on the next set of PTZ settings for each camera. Depending upon the physical setup these modules could be located on a single machine or distributed across many.

In a traditional completely centralized system, the raw video would be sent from each camera to a central server where the detector, tracker and parameter selection modules would all be located. In the fully distributed case, each camera would communicate only necessary information, e.g., the state estimates of the targets. The local objective function of each camera must be aligned with the centralized objective function such that any local decisions, using only locally available information, improve the value of the centralized objective function.

1.4 Organization

The rest of this dissertation is organized as follows. Chap. (2) provides a review of recent literature on problems being actively researched in camera networks. Chap. (3) presents the system models, describing how targets move through time and how measurements are acquired by the camera network. The functions quantifying system performance for different tasks are defined and discussed in Chap. (4). Different algorithms for the optimization of camera parameters are presented in Chap. (5). The experimental results are discussed in Chap. (6) along with the implementation details. Chap. (7) describes the extension to mobile camera networks. Finally, Chap. (8) concludes the dissertation.

Chapter 2

Literature Review

This chapter describes research related to camera networks grouped according to their application domain. Many of the camera networks currently deployed consist of a mixture of static and PTZ cameras. Historically these networks have consisted of mostly static cameras, where the layout is designed to satisfy some predefined objective such as covering an area.

2.1 Camera Network Topology

A fundamental task in wide-area surveillance using camera networks is to monitor the flow of traffic in a region. In environments such as airports this could be the flow of people navigating their way through the airport. For camera networks covering the streets this could involve monitoring the flow of traffic and pedestrians. In order to accomplish this task, an understanding of the patterns of flow between cameras observing the area is necessary.

The constraints present during the installation of the camera network can result in networks where certain portions of the camera network share overlapping fields of

view, while in other regions the fields of view are non-overlapping. If the cameras in the network have non-overlapping field of views, there will be portions of the area that cannot be seen or recorded by the camera network. When targets navigate through such a area, it is critical to understand the relationship between the disconnected fields of view. This can provide information on where a target leaving one field-of-view (FOV) is likely to appear next.

The probability distribution that describes this likelihood is called the transition model and can be represented as a graph, where the cameras are nodes and the edges connect adjacent cameras. Two cameras are adjacent if an object can travel between them without passing through any other cameras. The transition model describes the probability of traffic moving along each path and the time it takes to travel along that path (this can also be represented as a distribution). The camera network topology learning problem is usually referred to as estimating both network topology and transition model. This learning can be done during a separate training phase or continuously during operation of the camera network (e.g., in active and mobile camera networks the network topology may be changing dynamically).

The topology learning problem can be divided into two types: non-overlapping fields of view and overlapping fields of view. A significant portion of the work in this area considers cameras with non-overlapping fields of view. Real life camera networks usually consist of both overlapping and non-overlapping regions. In such cases overlapping fields of view can be considered together as a single large field of view.

Mutual information was used in [82] as a measure of statistical dependence to infer the camera network topology, and a Monte Carlo Markov Chain was using for sampling correspondences between observations. To exploit the abundant visual information provided by the imaging sensors, an appearance-integrated cross-correlation

model was proposed in [54] for topology inference on vehicle tracking data, and person identities were integrated with appearance in [95]. Ground-truth trajectory and object appearance information was used in [32] to learn the topology of a set of cameras as well as the pairwise illumination change between cameras during a training phase. A Bayesian approach was described in [22] for learning higher-order transition models, i.e., where moving objects will probably go after they pass through one or more cameras. In [21], a decentralized approach was proposed for camera network topology discovery based on sequential Bayesian estimation using a modified multinomial distribution. In [46], cross correlation and covariance over thousands of observations of departure and arrival times were utilized to identify adjacent cameras. This method also works for overlapping cameras, as negative transition time indicates that objects enter one camera's view before leaving another camera.

To infer the topology of camera networks with overlapping field of views, a Sequential Probability Ratio Test (SPRT) was utilized to accept or reject the possibility that two cameras observe the same scene based on accumulated sequential detections [44]. A method for inferring which cameras in the network have overlapping fields of view was presented in [10] based on a fixed-length message of automatically detected key points that each camera broadcasts to the rest of the network; the message is called "feature digest" which is a compressed representation of a camera's detected features.

2.2 Wide Area Tracking

Target tracking is one of the most basic tasks required for higher level automated video content analysis. The tracking problem can be defined as estimating trajectories of moving objects over time. Tracking a single target can have many com-

plications, such as occlusion, variation in appearance, and image noise. Once multiple targets are considered the critical issue of data association arises, i.e., the problem of linking together a sequence of observations across image frames based on the fact that they belong to the same object. This is critical in camera networks because it is very likely that many objects will be observed simultaneously by the camera network and they will move between and across the fields of view of multiple cameras. When considering wide area scene analysis, it is also necessary to be able to track over long time intervals. Although a large number of trackers are presently available, their reliability falls off quickly with respect to the the length of the tracks. Stable, long-term tracking is still a very challenging problem in camera networks. For multiple targets, the interaction between the targets may cause errors such as switching between track identifiers, missed detections and false detections. In addition, wide area tracking over a camera network requires solving the problem of handoff between cameras (i.e., which camera is responsible for tracking which target), making it more difficult to construct correspondences across camera views due to significant lighting and view changes.

In order to tracking multiple targets in camera networks, a lot of effort has been devoted to solving the data association problem based on the results of object detection. Multi-Hypothesis Tracking (MHT) in [67] and Joint Probabilistic Data Association Filters (JPDAF) in [4] and [5] are two representative methods. In order to overcome the large computational cost of MHT and JPDAF, various optimization algorithms such as Linear Programming [33], Quadratic Boolean Programming [40], and Hungarian algorithm [61] are used for data association. In [92], data association was achieved through a MCMC sampling-based framework.

Some of the existing methods on tracking in a camera network include [27],[32] and [36]. In [27], a probabilistic approach was presented for finding corresponding

vehicles across cameras on a highway. The appearance of vehicles was modeled by the mean of the color and transition times were modeled as Gaussian distributions. A graph-theoretic framework for addressing the problem of tracking in a network of cameras was proposed in [32]. A Bayesian formulation of the problem of reconstructing the path of objects across multiple non-overlapping cameras was described in [36] using color histograms for object appearance. The authors in [66] 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 [41], a particle filter was used to switch between track prediction for non-overlapping cameras and tracking within a camera. The authors in [35], presented a method for tracking in overlapping stationary and pan-tilt-zoom cameras by maximizing a joint motion and appearance probability model. A multi-objective optimization framework was presented in [70] and [71] for tracking in a camera network.

2.3 Camera Network Layout

The research in this domain focuses on methods to optimally place and orient a network of cameras such that certain task requirements are satisfied. The degree to which a particular camera network placement and configuration satisfies the requirements is determined by factors such as visibility, field-of-view, focus, resolution and pose. The approaches described in this section provide a brief review of past performance measures and optimization methods to find the best solution to the camera layout problem. A comprehensive review of methods to determine camera network layout can be found in [79].

In [90], the authors proposed a generate and test method for determining the position of a camera network for 3D estimation. This method relied on estimating the uncertainty ellipsoid given an arbitrary network of cameras, which could then be used to aid in the placement and orientation of the cameras. An approach for camera network design was proposed in [56] to minimize 3D measurement error. The optimal camera network layout is determined by using a genetic algorithm that considers the error from image based measurements, occlusion and other physical constraints.

The authors in [78] developed a synthesis based method where the placement of the camera network was computed such that certain feature detectability requirements were satisfied. Mittal and Davis [50] use the probability of visibility to determine the location and settings of their camera network. They considered not only static occlusion but also dynamic occlusion due to moving targets. This was further extended in [51] to handle the presence of random occlusions.

Horster and Lienhard [26] used linear programming to determine camera positions in 2D floor plans for maximizing area coverage, minimizing cost given a area coverage constraint, or optimizing camera orientation given fixed positions. Naish et al. [53] compute the initial camera network layout given the expected trajectories of the targets such that the sensing performance is maximized.

Optimal camera placement strategies proposed in [94] and [86] were solved by using a camera placement metric that captures occlusion in 3-D environments, and binary integer programming. The solution to the problem of optimal camera placement given coverage constraints was presented in [19], and can also be used to come up with an initial camera configuration.

2.4 Distributed Processing

Currently most of the data collected by camera networks is analyzed manually, a task that is extremely tedious and limits the potential of the installed network. This has sparked a huge interest in automated analysis in camera networks. This section briefly reviews some research on automated analysis in camera networks, a comprehensive review of which can be found in [68]. Whether the analysis is done manually or automatically, it is important to acquire images that can be analyzed reliably. As the placement of the static cameras is fixed, the performance of the system can not adapt to the sensed data or environmental conditions.

It is desirable that the video analysis tasks be decentralized in many applications. Depending on the application, here may be bandwidth constraints (e.g., mobile networks), security issues, and difficulty in analyzing a huge amount of data centrally. Distributed systems can also be easily installed and allow for operations in remote and hostile environments, possibly alongside humans. They can provide very valuable information to humans (e.g., search and rescue personnel) keeping them out of danger and enabling operations in wider variety of environments than currently possible. Situations such as these require cameras to act as autonomous agents capable of making decisions in a decentralized manner. At the same time, however, the decisions of the cameras need to be coordinated so that there is a consensus on how to complete the task. In a distributed camera network, each camera node would need to processes its own image data locally, extract relevant information and collaborate with the other cameras to reach a shared, global analysis of the scene.

Although there are a number of methods in video analysis that deal with multiple cameras, *distributed* processing in camera networks has received much less attention.

For example, some of the well-known methods for learning a network topology [46, 82], tracking over the network [66, 70], object/behavior detection, matching across cameras, and camera handoff and camera placement [2, 20, 31, 76, 94] do not address the issue of distributed processing. In [47], a cluster-based Kalman filter was proposed for decentralized tracking, where a camera is selected as a cluster head and aggregates information in the cluster. In [64], a mixture between a distributed and a centralized tracking scheme was presented which uses both static and PTZ cameras in a virtual camera network environment.

Recently, some problems in video analysis have been addressed in a distributed manner. A distributed algorithm for the calibration of a camera network was presented in [14]. The problems of object detection, tracking, recognition and pose estimation in distributed camera networks were considered in [69, 89]. Distributed processing has been extensively studied in the multi-agent systems and cooperative control literature . Consensus algorithms [58] are one of the many types of distributed schemes used for collective decision making. Consensus algorithms are run individually by each agent, where through communication its network neighbors, over multiple iterations, ensures that all agents converge to the same decision. The main advantage of consensus algorithms is that this consensus is achieved purely through peer-to-peer communication without the need for a central fusion node, nor the network topology.

A theoretical framework for defining and solving consensus problems for networked dynamic systems was introduced in [59]. Consensus algorithms for reaching an agreement without computing any objective function appeared in the work of [30]. Consensus schemes have also been gaining popularity in computer vision applications involving multiple cameras, such as pose estimation [84] and activity recognition [73]. The Kalman Consensus Filter (KCF) [57] is a popular distributed algorithm used to

compute the average consensus value. This was extended by the authors in [34] to relax some of the constraints and improve the consensus estimates. such as the assumption all targets are observable by each camera.

The authors in [83] extended consensus algorithms to perform linear algebraic operations such as SVD, least squares and PCA in camera networks, while [77] presented an approach to distributed tracking in camera networks. In [73] a method for distributed tracking and activity recognition was proposed, using a consensus based approach. A cooperative estimation mechanism called the networked visual motion observer was used in [25] to solve the estimation of the pose of a moving target in a camera network.

2.5 Camera Selection

As camera networks have historically been composed of mostly static cameras in prearranged locations for surveillance tasks, it is likely that many cameras in the network are either looking at uneventful spaces and may have significant overlap in their field-of-view with other cameras. Camera selection is an area of research that is dedicated to the selection of a subset of cameras for completion of a surveillance task such that scarce resources such as power, bandwidth and storage are conserved. This section provides a brief review of the approaches and performance measures proposed to solve this problem.

The mean squared error of the best 2D position estimate of a single target, in the presence of occlusion, is used by the authors of [18] using a semi-definite programming approach to handle camera selection. [28] considers how to select a "good" subset of sensors such that the error of the position estimate of a target is within an error bound. Soro and Heinzelman [74] compare two camera selection methods for best

imaging performance while minimizing power consumption. In the first method, cameras that minimize the difference in view from the desired synthesized viewpoint are selected. In the second method, the energy limitations of the battery powered cameras are also considered.

The authors in [15] selectively use a subset of cameras depending on the scene. The optimal set of cameras is selected such that the area of interest is sufficiently covered and the frame rate and resolution are of sufficient quality. A review of multiple methods for camera selection and handoff is provided in [43].

2.6 PTZ Camera Control

In many applications, a portion of the installed cameras have pan, tilt and zoom parameters that can be modified dynamically. The increasing quantity controllable cameras and the potential benefit of intelligent sensing makes it necessary to develop strategies for this purpose. The network would then be capable of reconfiguring itself by modifying these parameters. One of the advantages of having a dynamically self-configurable network is that it may be prohibitively expensive or physically challenging to have a static setup able to handle 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 in an area 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 area. 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 issue of actively controlling the camera parameters is closely tied to camera placement since they both affect the quality of images that can be captured. Another relevant problem in camera control is to construct the mapping between the camera parameter space and a reference frame, which can be either an image plane or a ground plane. In [12], a method was proposed to map any pixel location in video image to its corresponding camera PTZ parameter. The authors also addressed a geo-registration technique to map the PTZ camera view-spaces to a wide-area aerial orthophotograph (e.g., Google Map Image).

Early work done in active vision [6] looked at the problem of moving a camera to improve imagery. Performing active vision in a distributed camera network, where the cameras coordinate among themselves, remains relatively unexplored. Much of the research in controllable camera networks in the last decade was focused on centralized solutions to master-slave systems where static cameras directed the PTZ camera. The path planning inspired approach proposed by [65] used static cameras to track all targets in a virtual environment while the PTZ cameras were each assigned to obtain high resolution video from a particular target. This approach showed that given the predicted tracks of all the targets, a set of one-to-one mappings between cameras and targets can be formed to acquire high resolution videos. A method for determining good sensor configurations that would maximize performance measures was introduced in [49]. The configuration framework was based on the presence of random occluding objects and two techniques were proposed to analyze the visibility of the objects.

A more recent approach in [62] and [48] uses the Expectation-Maximization (EM) algorithm to find the optimal configuration of PTZ cameras given a map of activities. The value of each discretized ground coordinate is determined using the map of activities. This approach upon convergence of the EM algorithm, provides the PTZ settings to optimally cover an area given the map of activities. A similar approach in [75] showed how cameras can coordinate between themselves to perform area coverage by playing a potential game.

The game theoretic aproach in [72] and [17] showed how local value functions could be designed to constantly improve the tracking accuracy of all targets observed by a self-configuring camera network. While the approaches in these papers were decentralized and had high accuracy, they were focused largely on designing specific imaging functions on a per-target basis, and not on the overall system optimization required for more complex scenes involving interactions between multiple targets. Therefore, the implementation results were limited to very simple settings. More discussion on how to design games in general for distributed optimization can be found in [42].

Prior work in target tracking with the ability to obtain high resolution shots was shown in [1] using a fixed cost function. This was extended in [16] where the system tracking performance was used as a threshold to enable acquisition of high resolution face images from targets in the area. The main contribution of this paper is to present a much more general approach where interesting events that the targets are involved in are detected, and the system automatically decides when, where and how to image these targets.

2.7 Experiments on Camera Networks

2.7.1 Simulation

Researchers from both computer vision and sensor networks are interested in the sensing and control issues that arise in the visual surveillance of large public spaces. However, deploying a large-scale physical surveillance system may not be easy for those who are interested in experimenting with multi-camera systems. Access to existing private and public camera networks for experimentation is also difficult to obtain. As a means of overcoming this barrier, as well as to avoid privacy laws that restrict the monitoring of people in public spaces, the *Virtual Vision* paradigm was introduced in [80], which facilitates research through the virtual reality simulation of populated urban spaces with camera networks. Within the virtual vision framework, a surveillance system comprising smart cameras that provide perceptive coverage of a virtual reconstruction of New York City's original Pennsylvania Station was developed [81]. This virtual train station is populated by autonomously self-animating virtual pedestrians; virtual passive and active cameras generate multiple synthetic video feeds that emulate those generated by real surveillance cameras monitoring public spaces.

Virtual vision has several advantages: The virtual cameras are very easily relocated and reconfigured, the ground-truth data of the virtual world is readily accessible, experiments are repeatable, and simulated camera networks can be actively controlled thus enabling performance analysis of different camera control algorithms.

2.7.2 Real Life Testbeds

Experimentation in real camera networks can be very challenging for a number of reasons. Engineering issues like data transmission rates or processing capabilities may

limit the kinds of experiments that may be conducted. Moreover, since camera networks span a wide area, it is hard to design controlled experiments that are also practically relevant. The complexity of real-world environments where camera networks are most likely to be deployed, like an airport or a shopping mall, are very hard to replicate in a research setting. Below we list some systems that are currently available and which can be used by researchers in their experimentation. This list is by no means complete; it is intended to provide a sampling of some of the existing camera network experimental testbeds.

In the early 2000s, a virtual world was developed at the University of Toronto to serve as a software laboratory for carrying out camera networks research. This involved a large scale digital reproduction of the Penn train station complete with cameras and autonomous pedestrians [80]. The Sensorium at Boston University was built around the year 2003 and is composed of wireless cameras spread over multiple rooms [55]. All the projects at the Sensorium have the common goal of merging the physical and cyber worlds allowing development of assisted environments for people with disabilities. An Outdoor Video Sensor Network Laboratory, composed of over 30 PTZ cameras, has been developed at U.C. Riverside. This system also has a number of indoor cameras, infra-red and 3D sensors, and ground and aerial unmanned vehicles. A wide area video understanding dataset was collected in this environment and is available to researchers (Videoweb Activities Dataset) [13]. The cameras in this facility are connected over a wireless network allowing for research in resource-constrained and distributed environments. In 2009, researchers at U.C. Santa Barbara installed a large network of static and PTZ cameras in the hallways of all five floors of a campus building and at various other locations on campus. They were interested in many different applications ranging from the understanding the patterns of movements within a building to developing dis-

tributed control algorithms [91]. On the other side of the globe, in the Nullarbor Desert in Western Australia, researchers from the Imperial College of London, Ondrejov Observatory in the Czech Republic, and the Western Australian Museum, set up a network of cameras track and find new meteorites [7]. This shows the potential for application of camera networks in environmental monitoring of remote regions.

Chapter 3

System Modeling and Tracking

This chapter describes the state and dynamical models necessary for tracking and reasoning about the system and targets over time. First, the time propagation models are presented. These show how the system and the targets behave through time, while the measurement model describes the relationship between the camera and detector from Fig. (1.1) to the targets in the scene. This is followed by an introduction to the Kalman filter frameworks that comprise the tracking module used in the experiments.

3.1 Time Propagation Models

The purpose of this section is to define the time propagation and measurement models of our system, which will form the basis for evaluating the performance of the video analysis tasks.

3.1.1 State Variables

At time step k, the system state is described by the state vector $\mathbf{x}(k) \in R^N$. In the case of tracking targets in a camera network, the 3D position of target j in the world

coordinate system is represented by the vector $\mathbf{p}_j^w(k) = [x_j^w, y_j^w, z_j^w]^\top$. If the velocity of the target, $\mathbf{v}_j^w(k) = [v_{x,j}^w, v_{y,j}^w, v_{z,j}^w]^\top$, is also part of the state, then the state vector for target j becomes $\mathbf{x}_j(k) = [\mathbf{p}_j^w(k), \mathbf{v}_j^w(k)]^\top$. Given the state $\mathbf{x}_j(k)$ a motion model can be used to predict how the target moves over a time interval and a measurement model can be used to predict the corresponding measurements that will be acquired by the camera network.

3.1.2 Motion Model

The motion model describes how the target is expected to move over a time interval. A review of target motion models and tracking methods can be found in [9]. Traditional motion models have been defined by a constant physical parameter, such as constant position $\mathbf{p}_j^w(k) = \mathbf{p}_j^w(0)$ and constant velocity $\mathbf{p}_j^w(k) = \mathbf{p}_j^w(0) + \mathbf{v}_j^w(0)$. More complex motion models utilizing scene geometry and learned motion patterns have also been used in recent papers but require more application specific information.

As track j moves, with time step $T = 1s$, throughout the area, its trajectory in discrete time is modeled in state space as,

$$\mathbf{x}_j(k+1) = f(\mathbf{x}_j(k)) + \boldsymbol{\omega}, \tag{3.1}$$

where f is the state transition function, relating the state between time step k and $k+1$, and the random variable $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_{6\times 1}, \mathbf{Q})$ is the process noise. in practice the state transition function is linearized around the point of interest $\mathbf{x}_j$ by computing

$$\boldsymbol{\Phi}_j = \left.\frac{\partial f_j}{\partial \mathbf{x}_j}\right|_{\mathbf{x}_j}, \tag{3.2}$$

resulting in

$$\mathbf{x}_j(k+1) = \mathbf{\Phi}\mathbf{x}_j(k) + \boldsymbol{\omega}, \tag{3.3}$$

as the linear discrete time trajectory model, where $j = 1, \ldots, N_t$ is the target number. Using the constant velocity motion model

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{3.4}$$

is the state transition matrix where $\mathbf{0}$ and $\mathbf{I} \in \Re^{3\times 3}$. Welch and Bishop discuss how to choose the process noise $\mathbf{Q}$ in [87].

3.2 Measurement Model

This section derives the measurement model for track j by camera i. The position of the j-th track in the i-t camera frame is $\mathbf{p}_j^{c_i} = \left[x_j^{c_i}, y_j^{c_i}, z_j^{c_i}\right]^\top$, the rotation from world to the i-th camera frame is denoted by $\mathbf{R}_w^{c_i}$ and the position of the camera in the world frame is $\mathbf{p}_{c_i}^w$.

The position of track j in the i-th camera frame is related to it's position in the world frame by

$$\mathbf{p}_j^w = \mathbf{R}_{c_i}^w \mathbf{p}_j^{c_i} + \mathbf{p}_{c_i}^w \tag{3.5}$$

$$\mathbf{p}_j^{c_i} = \mathbf{R}_w^{c_i}(\mathbf{p}_j^w - \mathbf{p}_{c_i}^w). \tag{3.6}$$

Eqn. (3.6) can be written in homogeneous co-ordinates as

$$\begin{bmatrix} \mathbf{p}_j^{c_i} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^{c_i} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{p}_{c_i}^w \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_j^w \\ 1 \end{bmatrix} \tag{3.7}$$

where $\mathbf{0} \in \Re^{3\times 1}$ and $\mathbf{I} \in \Re^{3\times 3}$ is the identity matrix.

Using the homogeneous co-ordinate representation, the i-th camera's image plane co-ordinates for the j-th track, $\mathbf{p}_j^i = [x_j^i, y_j^i, z_j^{c_i}]^\top$, are

$$\mathbf{p}_j^i = \begin{bmatrix} -\frac{f_{c_i}}{s_x} & 0 & 0 & o_x \\ 0 & -\frac{f_{c_i}}{s_y} & 0 & o_y \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_j^{c_i} \\ 1 \end{bmatrix} \tag{3.8}$$

In Eqn. (3.8), $\mathbf{p}_j^i$ are the homogeneous co-ordinates of the track position on the image plane with $z_j^{c_i}$. The symbols o_x and o_y represent the co-ordinates of the location of the image center in pixel co-ordinates, s_x and s_y represent the effective pixel size in the horizontal and vertical direction, respectively, and f_{c_i} is the focal length of the i-th camera.

The result of performing feature detection on the image from Camera i is a two-dimensional projection of the centroid position of the j-th track on the image plane in pixel coordinates $\mathbf{u}_j^i$ and it's associated error covariance $\mathbf{C}_j^i$. Accounting for noise, the measurement from the i-th camera can be modeled as

$$\mathbf{u}_j^i = h(\mathbf{p}_j^w) + \boldsymbol{\eta}_j^i = \begin{bmatrix} \frac{x^i}{z_j^{c_i}} \\ \frac{y^i}{z_j^{c_i}} \end{bmatrix} + \boldsymbol{\eta}_j^i, \tag{3.9}$$

where we assume that $\boldsymbol{\eta}_j^i \sim \mathcal{N}(\mathbf{0}_{2\times 1}, \mathbf{C}_j^i)$. By defining the matrix of (known) intrinsic

camera parameters $\mathbf{M}_{int}$ and the matrix of extrinsic camera parameters $\mathbf{M}_{ext}$ as

$$\mathbf{M}_{int} = \begin{bmatrix} -\frac{f_{c_i}}{s_x} & 0 & 0 & o_x \\ 0 & -\frac{f_{c_i}}{s_y} & 0 & o_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{M}_{ext} = \begin{bmatrix} \mathbf{R}_w^{c_i} & -\mathbf{R}_w^{c_i}\mathbf{p}_{c_i}^w \\ \mathbf{0}^\top & 1 \end{bmatrix},$$

where $\mathbf{0} \in \Re^{3\times 1}$. We can rewrite Eqn. (3.7) as

$$\mathbf{p}_j^i = \mathbf{M}_{int}\mathbf{M}ext \begin{bmatrix} \mathbf{p}_j^w \\ 1 \end{bmatrix}, \tag{3.10}$$

where the product $\mathbf{M}_{int}\mathbf{M}_{ext}$ gives us the Calibration Matrix.

3.2.1 Linearizing the Measurement Model

Given $\mathbf{p}_{c_i}^w$, $\hat{\mathbf{p}}_j^w$, and $\mathbf{R}_w^{c_i}$, Eqn. (3.10) allows computation of $\hat{\mathbf{p}}_j^i = [\ \hat{x}_j^i, \hat{y}_j^i, \hat{z}_j^{c_i}\ ]^\top$ which enables prediction of the image plane measurement:

$$\hat{\mathbf{u}}_j^i = h_j^i(\hat{\mathbf{p}}_j^w) = \begin{bmatrix} \frac{\hat{x}^i}{\hat{z}_j^{c_i}} \\ \frac{\hat{y}^i}{\hat{z}_j^{c_i}} \end{bmatrix}. \tag{3.11}$$

Subsequent analysis will use the linearized relationship between the measurement error $\mathbf{u}_j^i - \hat{\mathbf{u}}_j^i$ and the position estimation error in global frame $\mathbf{p}_j^w - \hat{\mathbf{p}}_j^w$:

$$\mathbf{u}_j^i \approx \hat{\mathbf{u}}_j^i + \mathbf{h}_j(\mathbf{p}_j^w - \hat{\mathbf{p}}_j^w) \tag{3.12}$$

where $\mathbf{h}_j = \left.\frac{\partial h_j^i}{\partial \mathbf{p}_j^w}\right|_{\hat{\mathbf{p}}_j^w}$. The derivation is as follows:

$$\begin{aligned}
\mathbf{h}|_{\hat{\mathbf{p}}_j^{c_i}} &= \frac{\delta h_j^i}{\delta \mathbf{p}_j^w} \\
&= \begin{bmatrix} \frac{\delta h_j^i}{\delta \hat{\mathbf{p}}_j^i} \end{bmatrix} \begin{bmatrix} \frac{\delta \hat{\mathbf{p}}_j^i}{\delta \mathbf{p}_j^w} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\hat{z}_j^{c_i}} & 0 & -\frac{\hat{x}^i}{(\hat{z}_j^{c_i})^2} \\ 0 & \frac{1}{\hat{z}_j^{c_i}} & -\frac{\hat{y}^i}{(\hat{z}_j^{c_i})^2} \end{bmatrix} \mathbf{M}_{int}\mathbf{M}_{ext} \frac{\delta}{\delta \mathbf{p}_j^w} \begin{bmatrix} \mathbf{p}_j^w \\ 1 \end{bmatrix} \\
&= \mathbf{\Theta}\mathbf{M}_{int}\mathbf{M}_{ext} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\
\mathbf{h}|_{\hat{\mathbf{p}}_j^{c_i}} &= \mathbf{\Theta}\mathbf{\Sigma} \qquad (3.13)
\end{aligned}$$

where,

$$\mathbf{\Theta} = \begin{bmatrix} \frac{1}{\hat{z}_j^{c_i}} & 0 & -\frac{\hat{x}^i}{(\hat{z}_j^{c_i})^2} \\ 0 & \frac{1}{\hat{z}_j^{c_i}} & -\frac{\hat{y}^i}{(\hat{z}_j^{c_i})^2} \end{bmatrix} \qquad (3.14)$$

$$\mathbf{\Sigma} = \mathbf{M}_{int}\mathbf{M}_{ext} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \qquad (3.15)$$

For the purpose of state estimation, we will require the observation matrix $\mathbf{H}_j^i = \frac{\partial \mathbf{u}_j^i}{\partial \mathbf{x}}|_{\hat{\mathbf{u}}_j^i, \hat{\mathbf{x}}}$ which is

$$\mathbf{H}_j^i = [\mathbf{h}, \mathbf{0}] \qquad (3.16)$$

where $\mathbf{0} \in \Re^{2\times 3}$. The linearized measurement model for every measurement at time-step k is thus given by,

$$\mathbf{u}_j^i(k) = \mathbf{H}_j^i(k)\mathbf{x}_j^i(k) + \boldsymbol{\eta}_j^i(k). \tag{3.17}$$

Typically the camera intrinsic matrix $\mathbf{M}_{int}$, extrinsic matrix $\mathbf{M}_{ext}$ and measurement noise $\mathbf{C}_j^i$ are obtained though the calibration process.

3.3 Kalman Filter Tracking

This section presents the equations for the centralized and distributed Kalman filter methods that may be used to perform tracking on a camera network. The standard Kalman filter is introduced first, followed by the distributed Kalman-Consensus filter to show how target state estimates and associated error covariance can be determined in camera networks that have centralized data processing or distributed processing across the cameras.

3.3.1 Centralized Kalman Filter

The Kalman filter as described in [87] is used to estimate the instantaneous state of a linear dynamic system. It is a two step process consisting of a prediction step followed by a measurement update step.

Prediction Step

The state estimate and its error covariance matrix $\mathbf{P}_j$ are propagated between sampling instants according to [29]:

$$\hat{\mathbf{x}}_j(k+1)^- = \boldsymbol{\Phi}\hat{\mathbf{x}}_j(k)^+ \tag{3.18}$$

$$\mathbf{P}_j(k+1)^- = \boldsymbol{\Phi}\mathbf{P}_j(k)^+\boldsymbol{\Phi}^\top + \mathbf{Q}. \tag{3.19}$$

Here $\hat{\mathbf{x}}_j(k+1)^-$ is the a priori state estimate of the target at time step $k+1$ and $\hat{\mathbf{x}}_j(k)^+$ is the a posteriori state estimate at time step k, given measurement $\mathbf{u}_j^i(k)$.

Measurement Correction Step

The state estimate and its error covariance are corrected using the measurement information when a new measurement is available. The Kalman gain, quantifying the change in state, is defined as

$$\mathbf{K}(k) = \mathbf{P}_j(k)^-\mathbf{H}_j(k)^\top \left(\mathbf{H}_j(k)\mathbf{P}_j(k)^-\mathbf{H}_j(k)^\top + \mathbf{C}_j(k)\right)^{-1}, \tag{3.20}$$

and the corrected a posteriori state estimate is calculated by

$$\hat{\mathbf{x}}_j(k)^+ = \hat{\mathbf{x}}_j(k)^- + \mathbf{K}(k)\left(\mathbf{u}_j(k) - \mathbf{H}_j(k)\hat{\mathbf{x}}_j(k)^-\right). \tag{3.21}$$

The a posteriori error covariance is updated though

$$\mathbf{P}_j(k)^+ = \left(\mathbf{I} - \mathbf{K}(k)\mathbf{H}_j(k)\right)\mathbf{P}_j(k)^- \tag{3.22}$$

3.3.2 Kalman-Consensus Filter

The following is a verbal description of the consensus algorithm as shown in Algorithm 1 and proposed in [60]. It is performed at each camera c_i for each target j that is viewed by C_i, the neighboring camera set of c_i and defined as all cameras with which c_i can communicate. If c_i is viewing a target j, it obtains j's position on the ground plane $\mathbf{u}_j^i$, $\mathbf{H}_j^i$ is the identity matrix since the state and measurements are both in the world frame. After that, the corresponding information vector and matrix are computed with the given measurement covariance matrix $\mathbf{C}_j^i$. Camera c_i then sends

Algorithm 1 Distributed Kalman-Consensus tracking algorithm performed by every camera c_i at discrete time step k. The state of target j at camera c_i is represented by $\mathbf{x}_{i,j}$ with error covariance matrix $\mathbf{P}_{i,j}$.

Input: $\hat{\mathbf{x}}_{i,j}$ and $\mathbf{P}_{i,j}$ from time step $k-1$
for each target j that is being viewed by c_i **do**
Obtain measurement $\mathbf{u}_{i,j}$ with covariance $\mathbf{C}_{i,j}$
Compute information vector and matrix

$$\mathbf{z}_{i,j} = {\mathbf{H}_j^i}^\top {\mathbf{C}_j^i}^{-1} \mathbf{u}_j^i \quad \text{and} \quad \mathbf{Z}_{i,j} = {\mathbf{H}_j^i}^\top {\mathbf{C}_j^i}^{-1} \mathbf{H}_j^i$$

Compute the predicted measurement

$$\mathbf{g}_{i,j} = {\mathbf{H}_j^i}^T {\mathbf{C}_j^i}^{-1} \mathbf{H}_j^i \bar{\mathbf{x}}_{i,j}$$

Compute the residue

$$\mathbf{res}_{i,j} = \mathbf{z}_{i,j} - \mathbf{g}_{i,j}$$

Send message $\mathbf{m}_{i,j} = (\mathbf{res}_{i,j}, \mathbf{Z}_{i,j}, \bar{\mathbf{x}}_{i,j})$ to neighboring cameras C_i
Receive messages $\mathbf{m}_{l,j} = (\mathbf{res}_{l,j}, \mathbf{Z}_{l,j}, \bar{\mathbf{x}}_{l,j})$ from all neighboring cameras $c_l \in C_i$
Fuse information

$$\mathbf{y}_{i,j} = \sum_{l \in C_i} \mathbf{res}_{l,j}, \quad \mathbf{S}_{i,j} = \sum_{l \in C_i} \mathbf{Z}_{l,j}$$

Compute the Kalman-Consensus state estimate

$$\mathbf{M}_{i,j} = ((\mathbf{P}_{i,j})^{-1} + \mathbf{S}_{i,j})^{-1}$$

$$\hat{\mathbf{x}}_{i,j}^+ = \hat{\mathbf{x}}_{i,j} + \mathbf{M}_{i,j}\mathbf{y}_{i,j} + \gamma \mathbf{M}_{i,j} \sum_{l \in C_i} (\hat{\mathbf{x}}_{l,j} - \hat{\mathbf{x}}_{i,j})$$

$$\gamma = 1/(||\mathbf{M}_{i,j}|| + 1), ||\mathbf{X}|| = (tr(\mathbf{X}^T\mathbf{X}))^{\frac{1}{2}}$$

Update the state and error covariance matrix for time step k

$$\mathbf{P}_{i,j} \leftarrow \mathbf{\Phi}_j \mathbf{M}_{i,j} {\mathbf{\Phi}_j}^\top + \mathbf{Q}^j$$

$$\hat{\mathbf{x}}_{i,j}^- \leftarrow \mathbf{\Phi}^j \hat{\mathbf{x}}_{i,j}^+$$

end for

a message $\mathbf{m}_{i,j}$ to its neighbors which includes the computed information vector and matrix, and its estimated target state $\hat{\mathbf{x}}_{i,j}$ at previous time step $(k-1)$. Camera c_i then receives similar messages $\mathbf{m}_{l,j}$ only from the cameras in its neighborhood that are also viewing target j. The information matrices and vectors received from these messages, and its own information matrix and vector, if c_i is viewing target j, are then fused and the Kalman-Consensus state estimate is computed. Finally, the ground plane state $\hat{\mathbf{x}}_{i,j}$ and error covariance matrix $\mathbf{P}_{i,j}$ are updated according to the assumed linear dynamical system.

3.3.3 Centralized Extended Kalman Filter

The Extended Kalman filter as described in [87] differs from the Kalman filter in that the state transition and observational models need not be linear functions of the state, but may be differentiable functions. This means that $\mathbf{\Phi}$ and H are the Jacobian matrices of the state transition and measurement functions, linearized around the current estimate. In this work the state transition functions is assumed to be linear, while the observational model is assumed to be non-linear. Therefore, only the observational model is linearized around the current state estimate. It remains a two step process consisting of a prediction step followed by a measurement update step.

Prediction Step

The state estimate and its error covariance matrix $\mathbf{P}_j$ are propagated between sampling instants according to [29]:

$$\hat{\mathbf{x}}_j(k+1)^- = \mathbf{\Phi}\hat{\mathbf{x}}_j(k)^+ \tag{3.23}$$

$$\mathbf{P}_j(k+1)^- = \mathbf{\Phi}\mathbf{P}_j(k)^+\mathbf{\Phi}^\top + \mathbf{Q}. \tag{3.24}$$

Here $\hat{\mathbf{x}}_j(k+1)^-$ is the a priori state estimate of the target at time step $k+1$ and $\hat{\mathbf{x}}_j(k)^+$ is the a posteriori state estimate at time step k, given measurement $\mathbf{u}_j^i(k)$.

Measurement Correction Step

The state estimate and its error covariance are corrected using the measurement information when a new measurement is available. The Kalman gain, quantifying the change in state, is defined as

$$\mathbf{K}(k) = \mathbf{P}_j(k)^-\mathbf{H}_j(k)^\top \left(\mathbf{H}_j(k)\mathbf{P}_j(k)^-\mathbf{H}_j(k)^\top + \mathbf{C}_j(k)\right)^{-1}, \tag{3.25}$$

and the corrected a posteriori state estimate is calculated by

$$\hat{\mathbf{x}}_j(k)^+ = \hat{\mathbf{x}}_j(k)^- + \mathbf{K}(k)\left(\mathbf{u}_j(k) - h\left(\hat{\mathbf{x}}_j(k)^-\right)\right). \tag{3.26}$$

The a posteriori error covariance is updated though

$$\mathbf{P}_j(k)^+ = \left(\mathbf{I} - \mathbf{K}(k)\mathbf{H}_j(k)\right)\mathbf{P}_j(k)^- \tag{3.27}$$

3.3.4 Extended Kalman-Consensus Filter

The Extended Kalman-Consensus filter allows us to track targets on the ground plane using multiple measurements in the image plane taken from various cameras. This allows each camera c_i to have at any time step k, a consensus state estimate $\hat{\mathbf{x}}_{i,j}$ and estimate error covariance $\mathbf{P}_{i,j}$ for each target j. Due to the nonlinear nature of the observation model, the linear Kalman-Consensus filter proposed in [60] cannot be applied as is. An extension to deal with the non-linearity of the observation model is required. Taking into account the nonlinear nature of our dynamical model, an Extended Kalman-

Algorithm 2 Distributed Extended Kalman-Consensus tracking algorithm performed by every camera c_i at discrete time step k. The state of target j at camera c_i is represented by $\mathbf{x}_{i,j}$ with error covariance matrix $\mathbf{P}_{i,j}$.

Input: $\hat{\mathbf{x}}_{i,j}$ and $\mathbf{P}_{i,j}$ from time step $k-1$
for each target j that is being viewed by c_i **do**
Obtain measurement $\mathbf{u}_{i,j}$ with covariance $\mathbf{C}_{i,j}$
Calculate Jacobian matrix $\mathbf{H}_j^i$ with respect to the observation model
Compute information vector and matrix

$$\mathbf{z}_{i,j} = {\mathbf{H}_j^i}^\top {\mathbf{C}_j^i}^{-1} \mathbf{u}_j^i \quad \text{and} \quad \mathbf{Z}_{i,j} = {\mathbf{H}_j^i}^\top {\mathbf{C}_j^i}^{-1} \mathbf{H}_j^i$$

Compute the predicted measurement

$$\mathbf{g}_{i,j} = {\mathbf{H}_j^i}^T {\mathbf{C}_j^i}^{-1} h_i(\bar{\mathbf{x}}_{i,j})$$

Compute the residue

$$\mathbf{res}_{i,j} = \mathbf{z}_{i,j} - \mathbf{g}_{i,j}$$

Send message $\mathbf{m}_{i,j} = (\mathbf{res}_{i,j}, \mathbf{Z}_{i,j}, \bar{\mathbf{x}}_{i,j})$ to neighboring cameras C_i
Receive messages $\mathbf{m}_{l,j} = (\mathbf{res}_{l,j}, \mathbf{Z}_{l,j}, \bar{\mathbf{x}}_{l,j})$ from all neighboring cameras $c_l \in C_i$
Fuse information

$$\mathbf{y}_{i,j} = \sum_{l \in C_i} \mathbf{res}_{l,j}, \quad \mathbf{S}_{i,j} = \sum_{l \in C_i} \mathbf{Z}_{l,j}$$

Compute the Extended Kalman-Consensus state estimate

$$\mathbf{M}_{i,j} = \left((\mathbf{P}_{i,j})^{-1} + \mathbf{S}_{i,j}\right)^{-1}$$

$$\hat{\mathbf{x}}_{i,j}^+ = \hat{\mathbf{x}}_{i,j} + \mathbf{M}_{i,j}\mathbf{y}_{i,j} + \gamma \mathbf{M}_{i,j} \sum_{l \in C_i} (\hat{\mathbf{x}}_{l,j} - \hat{\mathbf{x}}_{i,j})$$

$$\gamma = 1/(||\mathbf{M}_{i,j}|| + 1), ||\mathbf{X}|| = (tr(\mathbf{X}^T\mathbf{X}))^{\frac{1}{2}}$$

Update the state and error covariance matrix for time step k

$$\mathbf{P}_{i,j} \leftarrow \mathbf{\Phi}_j \mathbf{M}_{i,j} {\mathbf{\Phi}_j}^\top + \mathbf{Q}^j$$

$$\hat{\mathbf{x}}_{i,j}^- \leftarrow \mathbf{\Phi}^j \hat{\mathbf{x}}_{i,j}^+$$

end for

Consensus distributed tracking algorithm on the basis of the Kalman-Consensus filter detailed in [60] is shown in Algorithm 2.

The following is a verbal description of the consensus algorithm as shown in Algorithm 2. It is performed at each camera c_i for each target j that is viewed by C_i, the neighboring camera set of c_i and defined as all cameras with which c_i can communicate. If c_i is viewing a target j, it obtains j's position on its image plane $\mathbf{u}_j^i$, and calculates the Jacobian matrix $\mathbf{H}_j^i$ from its observation model and consensus state estimate $\hat{\mathbf{x}}_j$.

After that, the corresponding information vector and matrix are computed with the given measurement covariance matrix $\mathbf{C}_j^i$. Camera c_i then sends a message $\mathbf{m}_{i,j}$ to its neighbors which includes the computed information vector and matrix, and its estimated target state $\hat{\mathbf{x}}_{i,j}$ at previous time step $(k-1)$. Camera c_i then receives similar messages $\mathbf{m}_{l,j}$ only from the cameras in its neighborhood that are also viewing target j. The information matrices and vectors received from these messages, and its own information matrix and vector, if c_i is viewing target j, are then fused and the Extended Kalman-Consensus state estimate is computed. Finally, the ground plane state $\hat{\mathbf{x}}_{i,j}$ and error covariance matrix $\mathbf{P}_{i,j}$ are updated according to the assumed linear dynamical system.

Chapter 4

Design of Camera Network Objective Functions

Consider a network of $N_{cameras}$ PTZ smart cameras each of which contains a detector, a distributed Kalman-consensus tracker and some high level processing tasks (as needed by the application). This chapter studies several different possible use cases of a camera network and how to design utility functions to quantify the system performance.

4.1 Measuring System Performance

In this section we present possible designs of the value functions for several different objectives of our system with respect to the pan-tilt-zoom settings $\mathbf{a} \in A$ of the entire camera network. To ease computation we quantize the area of interest into a set of blocks $B = \{b_1, b_2, ..., b_{N_{blocks}}\}$ as shown in Fig. (4.1).

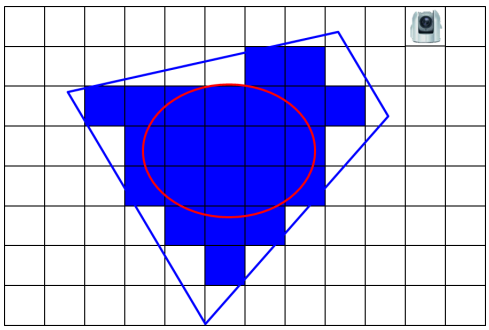


Figure 4.1: Example of discretization of the field of view (in blue) of a camera and the probable positions of the tracked target (in red). Shaded blue regions are the blocks satisfying the area coverage resolution requirement in eqn. (4.2). Blocks within the red ellipse are blocks in the set B_j.

4.1.1 Area Coverage

The requirements of a surveillance application for detecting all targets within a region of interest are naturally expressed in state space. In order to detect all targets present in an area, the entire area must be viewed by cameras in the network. This region of interest can take many different shapes and each camera may only be capable of viewing a small subset of the entire region. By converting the region of interest into a set of blocks we can then define the value to be a function of the blocks covered by the camera network. Depending on the detector and desired detection performance, certain resolution and overlap requirements may be present. This means that a good measure of the area coverage should also contain variables for tuning the resolution requirement and degree of overlap in the camera configurations. Consider the value function

$$U_{area}(\mathbf{a}) = \sum_{l=1}^{N_{blocks}} \left(1 - \prod_{i=1}^{N_{cameras}} (1 - \beta_{i,l})\right), \tag{4.1}$$

where $\beta_{i,l}$ is defined as

$$\beta_{i,l} = \begin{cases} 1 - e^{-\lambda_1 \frac{r_{i,l}}{r_{max,l}}} & \text{if } r_{max,l} > r_{i,l} > r_{min,l} \\ 0 & \text{otherwise} \end{cases}, \tag{4.2}$$

$r_{min,l}$ is the minimum acceptable resolution in terms of pixel height at which each block should be viewed, $r_{max,l}$ is the height in pixels of c_i's image plane, and $r_{i,l}$ is the resolution at which block l is being viewed by c_i.

The conditions under which this value function increases are when area covered with an acceptable resolution increases or the area can be viewed at a higher resolution. The term λ_1 can be adjusted according to how well the detection and tracking performs with respect to the height in pixels of the tracked objects on the image plane. This definition of the value function for area coverage provides the user to specify the range of resolutions to satisfy the performance requirements of the detector.

4.1.2 Tracking

A very popular topic in camera networks and computer vision in general is tracking. The objective of a camera network based tracking application could involve maximizing the tracking accuracy of all targets the system is responsible for. The user of the system may have desired tracking accuracy for different targets or regions of interest. For the system to be able to optimize along these goals, it must have a method for determining the effect of the camera network settings **a** on the tracker performance.

The purpose of the tracking utility is to quantify how well camera c_i believes the system will track target j given some proposed settings for all cameras in the network. Whether we are considering a centralized tracker or a completely distributed system, either of the Kalman filter or the Extended Kalman-consensus filter is capable

of providing a state estimate $\hat{\mathbf{x}}_j$ and error covariance $\mathbf{P}_j$ for each target. The state of each target is represented in the world frame and contains both the position and velocity and is represented as $\bar{\mathbf{x}}_j = [\mathbf{p}_j^w, \mathbf{v}_j^w]$. The error covariance matrix $\mathbf{P}_j$ can be represented in block form as

$$\mathbf{P}_j = \begin{bmatrix} \mathbf{P_{pp}} & \mathbf{P_{pv}} \\ \mathbf{P_{vp}} & \mathbf{P_{vv}} \end{bmatrix}, \tag{4.3}$$

where $\mathbf{P_{pp}}$ represents the error covariance matrix of the position of target j. Assuming calibration we can compute the linearized transformation matrix $\mathbf{H}_j^i$ [85] between the world and image for each camera. As $\mathbf{H}_j^i$ is a function of the PTZ of the camera and the target we can evaluate the expected error covariance of the target given the proposed PTZ settings of all cameras as

$$\mathbf{P}_j^+ = \left((\mathbf{P}_j^-)^{-1} + \sum_{i=1}^{N_{cameras}} \mathbf{H}_j^{i^\top} (\mathbf{C}_j^i)^{-1} \mathbf{H}_j^i \right)^{-1}, \tag{4.4}$$

where $N_{cameras}$ is the number of cameras viewing target j. The corresponding posterior information matrix is denoted as $\mathbf{J}^{j+} = \left(\mathbf{P}^{j+}\right)^{-1}$. We now define the tracking utility as the average trace of the information of all targets,

$$U_{track}(\mathbf{a}) = \frac{1}{N_T} \sum_{j=1}^{N_T} trace(\mathbf{J}^{j+}). \tag{4.5}$$

Since Eqn. (4.4) is exactly the covariance update equation of the Kalman filter in information form, the tracking utility defined here is the average trace of the information of all tracked targets. Maximizing the the tracking utility U_{track} over the parameters of the camera network will result in the setting $\mathbf{a}$ that provides the largest improvement to the tracking information of the tracker.

4.1.3 Bayesian Value

Since utilities such as the tracking utility U_{track} are a function of the target states, they are actually dependent on the random variables $\mathbf{p}_j^w(k+1)$ (i.e. the predicted position of the target at time step k+1) for $j = 1, \ldots, N_{targets}$. Therefore, the optimization will be based on the expected value of the utility function over the distribution of the uncertainty in the estimated position of the target. Hence, we define a Bayesian value function $V(U(\mathbf{a}))$ as:

$$V(U(\mathbf{a})) = E\left\langle U(\mathbf{a}; \mathbf{p}_j^w,\ j = 1, \ldots, N_T)\right\rangle \tag{4.6}$$

$$= \int U(\mathbf{a}) p_{\mathbf{p}}(\zeta)\, d\zeta. \tag{4.7}$$

The dummy variable ζ is used for integration over the ground plane and $p_{\mathbf{p}}$ is the Normal distribution $\mathcal{N}(\hat{\mathbf{p}}_j^w, \mathbf{P}_{\mathbf{pp}}^-)$ of the predicted position of target j in the global frame at the next imaging instant, and $\mathbf{P}_{\mathbf{pp}}^-$ represents the position covariance matrix.

The integral represents the area spanned by the FOV of the camera. Inside the integral, the target dependent utility $U(\mathbf{a})$ is multiplied by the probability distribution function $p_{\mathbf{p}}$, where the maximum value for $p_{\mathbf{p}}$ occurs at the estimated target position $\hat{\mathbf{p}}_j^w$. Thus, integrating over the FOV makes the camera network select a settings profile $\mathbf{a}$ such that most of the ellipsoid formed by the position covariance matrix $\mathbf{P}_{\mathbf{pp}}^-$ around the position estimate $\hat{\mathbf{p}}_j^w$, is in view, thus reducing the risk of not imaging the target.

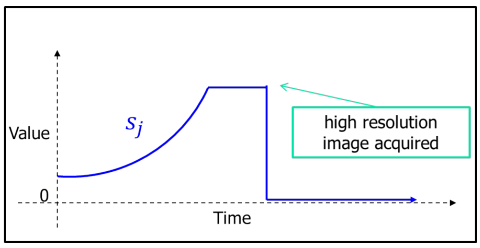


Figure 4.2: Example of how the scaling parameter s_j changes over the lifetime of a track. When a track is initialized s_j begins increasing over time until a high resolution image is obtained. Once this occurs the value is set to 0 for the remaining lifetime of the track. It is possible for a track to die before s_j becomes high enough to trigger a high resolution acquisition.

4.1.4 High Resolution Imaging

For some other surveillance tasks in camera networks it is important to be able to acquire high resolution images of specific targets for identification and recognition purposes. The user would need to be able specify a minimum and maximum resolution. Depending on the application certain targets may have different priorities that the user would also need to set.

An imaging value function would then reward camera parameter settings across the network that achieve high resolution imaging of the tracked objects. This function needs to take into account the expected position $\hat{\mathbf{p}}_j^w(k)^+$ and associated error covariance $\hat{\mathbf{P}}_j^w(k)^+$ of the tracks. The set of blocks $B_j \subseteq B$ are the blocks that are within three standard deviations of $\hat{\mathbf{p}}_j^w(k)^+$ in the area of interest as shown in fig. (4.1). B_j can also be thought of as the set of probable positions that track j is located.

We define the high resolution imaging value function as,

$$U_{image}(\mathbf{a}) = \sum_{j=1}^{N_{tracks}} s_j \left(1 - \prod_{i=1}^{N_{cameras}} (1 - \zeta_{i,j})\right), \tag{4.8}$$

where $\zeta_{i,j}$ is defined as

$$\zeta_{i,j} = \begin{cases} 1 - e^{-\lambda_2 \frac{r_{i,j}}{r_{max,j}}} & \text{if } r_{max,j} > r_{i,j} > r_{min,j} \\ 0 & \text{otherwise} \end{cases}, \tag{4.9}$$

and $r_{min,j}$ is the minimum resolution in terms of pixel height at the set of blocks B_j should be viewed at and $r_{max,j}$ is the maximum acceptable resolution. Here $r_{i,j}$ is the weighted resolution at which all B_j can be viewed by camera c_i,

$$r_{i,j} = \begin{cases} \sum_{b_l \in B_j} (p_{j,l} r_{i,l}), & \text{if } \forall b_l \in B_j, r_{i,l} > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{4.10}$$

where $p_{j,l}$ is the probability target j is at block l which is computed using $\hat{\mathbf{p}}_j^w(k)^+$ and $\hat{\mathbf{P}}_j^w(k)^+$. The scaling parameter s_j is used to weight the value of getting high resolution images of target j. By dynamically changing this parameter over time we introduce opportunistic acquisition of high resolution images into our system. We populate this variable using a function monotonically increasing in time, initialized to a low value when a new track j is added to the system. Once a high resolution image of the track has been acquired $s_j = 0$ for the remaining lifetime of the track. The behavior of this variable over the lifetime of a track can be seen in Fig. (4.2).

4.1.5 Specified Pose Imaging

In certain applications it is important that targets be imaged at specific poses. An example would be in the case of facial recognition an image of the target's face would be needed. Depending on the application there may be a range of satisfactory poses that would be set by the user.

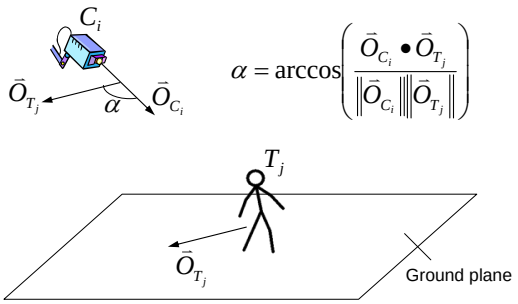


Figure 4.3: Camera viewing angle and target pose.

The pose value function $U_{pose}(\mathbf{a})$ would quantify how well the camera settings satisfy the pose requirements of the targets. This function would need to take into account the direction of the target as well as the direction of the camera. Let target T_j maneuver in the area with a direction vector $\vec{O}_{T_j}$. Defining a vector $\vec{O}_{c_i}$ from camera c_i's position $\mathbf{p}_i^w$ to T_j's estimated position $\hat{\mathbf{p}}_j^w$, the view angle $\theta_{i,j}$ formed by T_j and c_i can be computed as,

$$\theta_{i,j} = \arccos\left(\frac{\vec{O}_{T_j} \cdot \vec{O}_{c_i}}{\|\vec{O}_{T_j}\|\|\vec{O}_{c_i}\|}\right). \tag{4.11}$$

An illustration of the view angle factor is shown in Fig. 4.3. The pose value function can then be defined as,

$$U_{pose}(\mathbf{a}) = \sum_{j=1}^{N_{tracks}} \left(1 - \prod_{i=1}^{N_{cameras}} (1 - \upsilon_{i,j})\right), \tag{4.12}$$

where $\upsilon_{i,j}$ is defined as

$$\upsilon_{i,j} = \begin{cases} 1 - e^{-\lambda_3 \frac{\theta_{i,j}}{\theta_{max,j}}} & \text{if } \theta_{max,j} > \theta_{i,j} > \theta_{min,j} \\ 0 & \text{otherwise} \end{cases}, \tag{4.13}$$

and $\theta_{min,j}$ is the minimum acceptable angle between the target trajectory and the camera orientation and $\theta_{max,j}$ is the maximum acceptable angle. Assuming that the target is

facing in the direction of motion this utility value will provide a measure of how well the images aquired by the camera network satisfy the pose requirements for each target. If there are other methods that can estimate the pose of the target, that vector can be used in the place of the target direction to compute the relative angle and the resulting pose utility.

4.2 Global Utility Function

Depending on the complexity of the task assigned to the camera network by a user. One or more of these utilities may need to be considered at the same time. The following section provides some example scenarios and a possible definition of the global utility U_{global}.

4.2.1 Example 1 - Cover entire area with at least some minimum resolution while maintaining high resolution imagery of certain targets

This example is quite common in surveillance applications where a large network of cameras is used to cover a large area, and a human operator or automated mechanism maintains important targets or events in high resolution. To cover the entire area, the area can be divided into blocks in a way similar to [19] to make the problem more tractable. Then we can use the area coverage utility U_{area} to evaluate the parameter settings, where the user specifies the resolution requirements. For the user specified targets that need to be viewed constantly with higher resolution we also add in the imaging utility U_{image}. The resolution requirement of U_{image} is set to the desired values and the scaling parameter s_j for the user specified targets are set to a constant

high value. The global utility describing the system goals can then be defined as

$$U_{global}(\mathbf{a}) = U_{area}(\mathbf{a}) + U_{image}(\mathbf{a}). \tag{4.14}$$

By defining the global utility in such a way, there is a trade off between optimizing the area coverage and the high resolution imaging of the targets. This desired balance between the two goals can be located by changing the value of the scaling parameter s_j for each target j.

4.2.2 Example 2 - Cover the entrances to an area while optimizing tracking performance of the people in the area

The purpose of this system is to minimize the tracking error of all targets within the region under surveillance and obtain the best possible shots (in terms of tracking accuracy) for each target. The choice of camera parameters determines the expected assignment of targets to cameras, as well as the expected tracking accuracy and risk. Given that the targets are moving, when the entire area is not covered, there is a trade off between tracking gain and coverage risk. Each camera may view only a portion of the area under surveillance. The importance of every location within the observed space may not be equal and must also be considered. Monitoring entrances and exits to the area under consideration will allow us to identify, and subsequently track all targets, whereas monitoring empty space is of little use.

To achieve the system goal, the camera control is aware of the state of the Kalman-Consensus filter and actively seeks to provide it with the most informative measurements for state estimation. The imaging utility U_{image} can be applied to persistently cover the area entrances at a high resolution. The targets in this context would

be the entrances, the resolution requirements would defined based on the detection and recognition requirements of the user and the scaling factor s_j would be set to a high constant value. This means that the camera network will gain value from maintaining settings that can observe the entrances. The tracking utility U_{track} can be used to evaluate the tracking performance of the camera network. The targets in this case would be the people walking around the area. Since the area is not completely covered the tracking utility must also take into account the uncertainty in the location of the people in the area. These two utilities can then be combined together to form the global utility,

$$U_{global}(\mathbf{a}) = U_{image}(\mathbf{a}) + V(U_{track}(\mathbf{a})). \tag{4.15}$$

Chapter 5

Optimization Strategies for Camera Parameter Selection

This chapter describes how the pan-tilt-zoom (PTZ) paramenters of the camera network can be determined at each time step. Given a global utility function representative of the system goals, the objective is to find the set of PTZ settings for all cameras in the network that maximizes the utility value with respect to the target and state estimates at any time instant. Let $C = \{c_1, c_2, ..., c_{N_{cameras}}\}$ be the set of cameras, A_i denote the set of PTZ settings for camera c_i and $A = \{A_1 \times A_2 \times ... \times A_{N_{cameras}}\}$ be the set of all possible settings of the camera network.

5.1 Communication & Vision Graphs

In a distributed camera network, the camera communications can be modeled as a communication graph. The set of blocks representing the entire area is B and the set of blocks that can be covered by camera c_i, given all valid PTZ settings, is $B_i \subseteq B$.

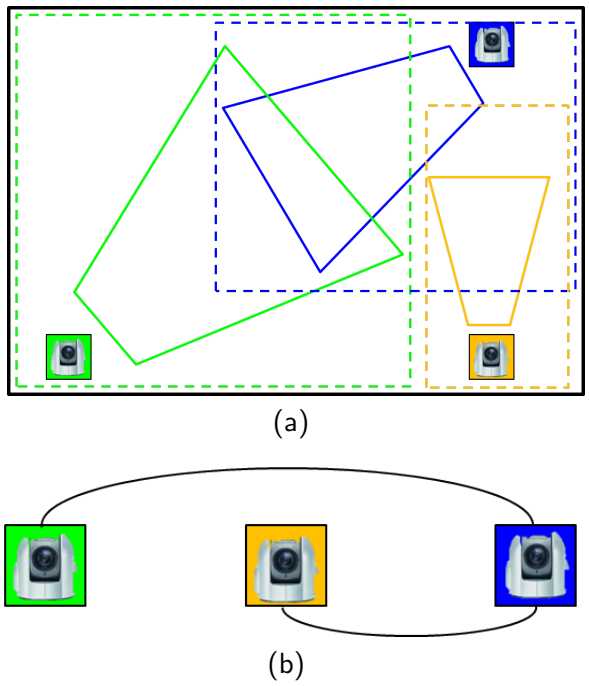


Figure 5.1: (a) Shows a particular set of FOV's for 3 cameras as solid trapezoids. The dotted rectangles show the possible area that can be covered given all the settings available to each camera. (b) Shows the corresponding vision graph, connecting cameras can have overlap in their fields of view.

Definition 1 *The **communication graph** is a graph in which only cameras that can directly send and receive information from each other are connected by an edge.*

Definition 2 *The **vision graph** is a graph where an edge from camera i' to camera i'' means that the set of blocks $B'_i \cap B''_i \neq \emptyset$.*

These graphs are important to our problem as the communication graph determines *who knows what and when*, while the vision graphs determines which cameras share operation regions. The example in Fig. (5.1(a)) shows 3 cameras each covering a portion of the area. From the image we can see that the green camera and the blue camera have overlaping operation regions, as do the blue and orange cameras. Thus, the vision graph for this example is represented by Fig. (5.1(b)).

5.2 Collaborative Sensing

5.2.1 Camera Utility

The global utilities must now be converted into local utilities in order for them to be solved in a distributed fashion. Convergence proofs in game theory [52] require that the local utilities are aligned with the global utility, i.e a change in the local utility affects the global utility similarly. We achieve this by making the utility of our camera equivalent to its contribution to the global utility, i.e.

$$U_{c_i}(\mathbf{a}) = U_{global}(\mathbf{a}) - U_{global}(\mathbf{a}_{-i}), \tag{5.1}$$

where the set $\mathbf{a}_{-i}$ is the set of settings profiles excluding the profile for camera i. This is known as the Wonderful Life Utility (WLU) [88], and as shown in [52] and applied in [3], leads to a potential game using the global utility as the potential function. This allows us to maximize the global utility through the maximization of the utility of each camera.

5.2.2 Potential Games

Below we summarize some known results in game theory based distributed optimization as they pertain to the camera network problem [52].

Definition 3 *A game is an* ***exact potential game*** *if there exists potential function* ϕ*:* $\mathcal{A} \rightarrow \mathbf{R}$ *such that for every camera,* $c_i \in \mathcal{C}$*, for every* $\mathbf{a}_{-i} \in \mathcal{A}_{-i}$ *and for every* $a_i, a_i' \in \mathcal{A}_i$*,*

$$U_{c_i}(a_i, \mathbf{a}_{-i}) - U_{c_i}(a_i', \mathbf{a}_{-i}) = \phi(a_i, \mathbf{a}_{-i}) - \phi(a_i', \mathbf{a}_{-i}),$$

where $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_{N_c}$ *is the strategy space, and* $\mathcal{A}_{-i} = \mathcal{A}_1 \times ... \times \mathcal{A}_{i-1} \times \mathcal{A}_{i+1} \times ... \times \mathcal{A}_{N_c}$*.*

Definition 4 *A game is an **ordinal potential game** if there exists ordinal potential function ϕ: $\mathcal{A} \rightarrow \mathbf{R}$ such that for every camera, $c_i \in \mathcal{C}$, for every $\mathbf{a}_{-i} \in \mathcal{A}_{-i}$ and for every $a_i, a_i' \in \mathcal{A}_i$,*

$$U_{c_i}(a_i, \mathbf{a}_{-i}) - U_{c_i}(a_i', \mathbf{a}_{-i}) > 0 \Leftrightarrow \phi(a_i, \mathbf{a}_{-i}) - \phi(a_i', \mathbf{a}_{-i}) > 0$$

Claim 5 *If a game involving the set of cameras $\mathcal{C}$ has a continuous ordinal potential function, ϕ, then the game has at least one Nash Equilibrium.*

Proof. Let $\mathbf{a} \in \mathcal{A}$ be the set of assigned camera parameters.

Let $\phi : \mathcal{A} \rightarrow \mathbf{R}$ be an ordinal potential function. Consider the case where a single camera changes parameters from a_i to $b_i (a_i, b_i \in \mathcal{A}_i)$.

Let ΔU_{c_i} be the change in utility caused by the change in parameters:

$\Delta U_{c_i} = U_{c_i}(b_i, \mathbf{a}_{-i}) - U_{c_i}(a_i, \mathbf{a}_{-i}) > 0$

Let $\Delta\phi$ be the change in potential caused by the change in parameters:

$\Delta\phi = \phi(b_i, \mathbf{a}_{-i}) - \phi(a_i, \mathbf{a}_{-i}) > 0$

Thus we can conclude that for a single camera's parameters change we get

$\Delta\phi > 0 \Leftarrow \Delta U_{c_i} > 0$

This means we can start from an arbitrary $\mathbf{a}$, and at each step one camera increases it's utility. Likewise, at each step ϕ is increased by $\Delta\phi > 0$. Since ϕ can accept only a finite number of values, it will eventually reach a local maxima. At this point no camera can achieve improvement, and we reach a Nash Equilibrium. ■

5.2.3 Negotiation Mechanisms

The dynamic nature of the targets in the region being observed requires that our cameras communicate with each other in order to decide the set of parameters that

will result in the optimal global utility. Each camera negotiates with its neighbouring cameras to accurately predict the actions of the other cameras before deciding its own action. The overall idea of the proposed negotiation strategy is to use *learning algorithms for multi-player games* [52]. A particularly appealing strategy for this problem is Spatial Adaptive Play (SAP) [**?**]. 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 the WLU described above. Iteration stops if a Nash equilibrium is attained or if the available operation time expires.

Application of SAP Negotiation Mechanism: At any step of SAP negotiations, a camera c_i is randomly chosen from the pool of cameras in the network according to a uniform distribution over the cameras, and only this camera is given the chance to update its proposed parameter settings. At negotiation step k, c_i proposes a parameter setting according to the following probability distribution based on other cameras' parameters at the previous step:

$$p_i(k) = \sigma\left(\frac{1}{\tau}\begin{bmatrix} U_{c_i}(A_i^1, \mathbf{a}_{-i}(k-1)) \\ \vdots \\ U_{c_i}(A_i^{|\mathcal{A}_i|}, \mathbf{a}_{-i}(k-1)) \end{bmatrix}\right) \tag{5.2}$$

for some $\tau > 0$, where $a(k-1)$ denotes the profile of proposed parameter settings at step $k-1$, $\mathcal{A}_i = \{A_i^1, ..., A_i^{|\mathcal{A}_i|}\}$ is the enumeration of all possible parameter settings of camera c_i, and $|\mathcal{A}_i|$ is the number of elements in $\mathcal{A}_i$. $\sigma(.)$ is the logit or soft-max function, and its mth element is defined as

$$(\sigma(x))_m = \frac{e^{x_m}}{e^{x_1} + \cdots + e^{x_n}}.$$

The constant τ determines how likely c_i is to select a parameter setting. If $\tau \to \infty$, c_i will select any setting $a_i \in \mathcal{A}_i$ with equal probability. As $\tau \to 0$, c_i will select a setting from its best response set

$$\{a_i \in \mathcal{A}_i : U_{c_i}(a_i, \mathbf{a}_{-i}(t-1)) = \max_{a_i' \in \mathcal{A}_i} U_{c_i}(a_i', \mathbf{a}_{-i}(t-1))\}$$

with arbitrarily high probability. In our implementation, we let $\tau \to 0$. After c_i updates its settings, it broadcasts its parameters (i.e. pan, tilt and zoom) to all neighboring cameras. The other cameras can then use that information to update their parameters after being chosen at any negotiation step until a consensus is reached. This occurs when the cameras have reached an Nash equilibrium, i.e., when no camera can increase the global utility by changing its parameters.

The utility functions combined with the negotiation strategy guarantees that at each time step the set of parameters chosen for the network of cameras is an optimal solution for the system's goals as defined by the global utility. In practice, we will not need to do this at every time step. The optimization can be done whenever the utility falls below a certain threshold.

5.3 Distributed Solution

As the size of the camera network grows, centralized approaches can become very hard to scale. In this framework each camera is assumed to be an independent entity with its own detector, tracker and control module. The cameras are assumed to be connected to a subset of the rest of the cameras in the network and only have access to information provided by these neighbors. A major benefit is that new cameras with varying capabilities can be easily added or removed from the network.

5.3.1 Distributed Parameter Selection

In order to gain the benefits of a fully decentralized camera network this algorithm can be converted into a distributed solution that runs on each camera. This section will show how to define and optimize local utilities at each camera such that the global utility of the camera network converges towards a local maxima.

The distributed case that is very similar to the centralized case is when there is an edge between all cameras in the communication graph. In this case all cameras have access to the necessary information (the settings of the camera network and the state and covariance of the targets) to compute the global utility of the system and optimize the PTZ settings of the network. There are many possible methods that result in target state estimate convergence throughout the entire network [34]. Given a consensus state estimate for all the targets in the network. The challenge is to reach convergence on $a_i, i \in \mathbf{a}$. This can be achieved by executing the following algorithm at each camera i.

Step 1: Compute the best PTZ setting $a_i$for camera i and send the proposed setting and U_{global} to all other cameras in the network.

Step 2: Receive the proposed settings $\mathbf{a}_{-i}$ and U_{global} the other cameras in the network.

Step 3: If camera i has the best proposed setting, execute the new setting. Update the $\mathbf{a}$ using the best proposed setting.

This can be shown to be a potential game [52] guaranteeing that the cameras will converge to a Nash equilibrium (N.E.). While this is a distributed solution there are a few undesirable characteristics such as allowing only one camera can move at each time step and requiring the settings from all the cameras and state and error covariance from all the targets (as was the case in the solution presented in [17]). As the size of the

camera network grows this can quickly become a problem. A more reasonable design of a distributed system would only require cameras that have the potential to share overlapping fields-of-view to be considered in the control algorithm.

Consider the case when the vision graph, given the PTZ constraints of the camera network, is a subgraph of the communication graph. In such a network, cameras can directly communicate with cameras whose operational regions overlap. The set of blocks representing the entire area is B and the set of blocks that can be covered by camera c_i, given a valid PTZ setting, is $B_i \subseteq B$. If the set of blocks $B_i' \cap B_i'' \neq \emptyset$, then c_i' and c_i'' are connected. Using the example in Fig. (5.1(a)) this would mean that the communication graph would have at least the connectivity shown in Fig. (5.1(b)). If the communication graph has less connectivity then it is possible that multiple cameras choose to complete the same task since they cannot know the settings of the other camera within one communication step.

Using the current algorithm, all cameras require the state of all other cameras which could take multiple hops across cameras to arrive. However, since the only cameras that can affect the value of the local camera's PTZ settings are those that are directly connected to it, new local utilities need to be defined.

In the following sections, let $C_i \subseteq C$ be the set of cameras containing c_i and its neighbors, $A^{c_i} = \times_{i \in C_i} A_i$ be the set of possible states of the cameras in C_i.

5.3.2 Local Area Coverage

Instead of considering all cameras in the network, the local area coverage utility is rewriten to only consider the pan tilt and zoom settings $\mathbf{a}^{c_i} \in A^{c_i}$ as,

$$u_{area}(\mathbf{a}^{c_i}) = \sum_{l \in B_l} \left(1 - \prod_{i \in C_i} (1 - \beta_{i,l}) \right). \tag{5.3}$$

This means that only cameras who are neighbors of c_i can affect this utility value. By defining the local utility in such a way, the change in value of the local utility, denoted by Δu_{area}, and global utility ΔU_{area}, by a change in the PTZ parameters of camera in c_i from a_i to b_i (where $a_i, b_i \in A_i$) are related by:

$$\begin{aligned}
\Delta u_{area} &= u_{area}(b_i, \mathbf{a}^{c_i}_{-i}) - u_{area}(a_i, \mathbf{a}^{c_i}_{-i}) \\
&= \sum_{l \in B_i} \left(1 - \prod_{i \in C_i} (1 - \beta_{i,l}) \right) \Bigg|_{b_i} - \\
&\quad \sum_{l \in B_i} \left(1 - \prod_{i \in C_i} (1 - \beta_{i,l}) \right) \Bigg|_{a_i} \\
&= U_{area}(b_i, \mathbf{a}_{-i}) - U_{area}(a_i, \mathbf{a}_{-i}) \\
&= \Delta U_{area}.
\end{aligned} \tag{5.4}$$

This is true because B_i contains all blocks that can be affected by a change in a_i in U_{area} and all the cameras that can affect the value of blocks in B_i are in the set $C_i \subseteq C$. This means that any increase in the local area coverage utility value u_{area} due to a change in a_i results in an equivalent increase in U_{area}.

5.3.3 Local High Resolution Imaging

The local value function for high resolution imaging of the tracked objects can then be written as,

$$u_{image}(\mathbf{a}^{c_i}) = \sum_{j \in T^i} s_j \left(1 - \prod_{i \in C_i} (1 - \zeta_{i,j}) \right), \tag{5.5}$$

where $\forall j \in T^i, B_j \cap B_i \neq \emptyset$ defines the set of tracks $T^i \subseteq T$. This definition has the following relationship between the local and global utilities,

$$\begin{aligned} \Delta u_{image} &= u_{image}(b_i, \mathbf{a}^{c_i}_{-i}) - u_{image}(a_i, \mathbf{a}^{c_i}_{-i}) \\ &= \sum_{j \in T^{c_i}} \left(1 - \prod_{i \in C_i} (1 - \beta_{i,l}) \right) \Bigg|_{b_i} - \\ &\quad \sum_{j \in T^{c_i}} \left(1 - \prod_{i \in C_i} (1 - \beta_{i,l}) \right) \Bigg|_{a_i} \\ &= U_{image}(b_i, \mathbf{a}_{-i}) - U_{image}(a_i, \mathbf{a}_{-i}) \\ &= \Delta U_{image}. \end{aligned} \tag{5.6}$$

This is true because only targets who have a probability of being in B_i can be affected by a change in a_i, and the only cameras that can affect blocks in B_i are the neighboring cameras C_i. This means that any increase in the local imaging utility value u_{image} due to a change in a_i results in a equivalent increase in U_{image}.

5.3.4 Local Utility Function

Given the new local value functions for area coverage and high resolution imaging defined in eqns. (5.3) and (5.5). The utility function representing the local value of

the system can be represented as

$$u_{local}(\mathbf{a}^{c_i}) = u_{area}(\mathbf{a}^{c_i}) + u_{image}(\mathbf{a}^{c_i}), \tag{5.7}$$

and the change in $\Delta u_{local}^{c_i}$, global utility ΔU_{global} and the PTZ parameters of camera in c_i from a_i to b_i (where $a_i, b_i \in A_i$) are related by:

$$\begin{aligned} \Delta u_{local}^{c_i} &= u_{local}(b_i, \mathbf{a}_{-i}^{c_i}) - u_{local}(a_i, \mathbf{a}_{-i}^{c_i}) \\ &= \Delta U_{global}. \end{aligned} \tag{5.8}$$

Since a change in the local utility $\Delta u_{local}^{c_i}$ is equivalent to the change in the global utility ΔU_{global} for any a_i. We can increase the global utility by increasing any camera's local utility.

5.3.5 Distributed Optimization Algorithm

Note that the above equations are defined such that a positive change in the local utility $\Delta u_{local}^{c_i}$ corresponds to a positive change in the global utility ΔU_{global}. This is necessary to ensure that local decisions are coordinated with the global objective of the system. Given the local utility defined in eqn. (5.7) and its alignment to the global utility of the camera network shown in eqn. (5.8), we modify the algorithm run at each camera i so that more than one camera can change PTZ parameters at each time step.

Step 1: c_i computes the best PTZ setting,

$$\max_{a_i \in A_i} (\Delta u_{local}^{c_i}), \tag{5.9}$$

for camera i and sends the proposed setting a_i and utility $\max_{a_i \in A_i}(\Delta u_{local}^{c_i})$ to all neighboring cameras.

Step 2: Receive the proposed setting a_i' and utility $\max_{a_i' \in A_i'}(\Delta u_{local}^{c_i'})$ from each other camera in the network.

Step 3: If c_i can provide the greatest improvement, i.e.,

$$\max_{c_i' \in C_i}(\max_{a_i' \in A_i'}(\Delta u_{local}^{c_i'})) = \max_{a_i \in A_i}(\Delta u_{local}^{c_i}), \tag{5.10}$$

then execute the proposed settings. In the case that multiple cameras can provide the same improvement any tie breaker such that $g(c_i, c_i') = g(c_i', c_i)$ can be used.

Step 4: Send current settings of c_i to all neighboring cameras.

Step 5: Receive current settings from all neighboring cameras and update $\mathbf{a}^{c_i}$.

5.3.6 Deterministic Equilibrium

The convergence characteristics of the above algorithm are now analyzed and show that the cameras arrive at a Nash equilibrium (N.E.) with one or more cameras deciding to change parameters simultaneously.

Definition 3. A path in the parameter space S of the camera network is a sequence of camera network parameters $(\mathbf{a}^1, \mathbf{a}^2, ..., \mathbf{a}^{N_S})$ such that each two consecutive camera network states differs in at least one camera state.

Definition 4. An improvement path is a path in which $U_{global}(\mathbf{a}^k) < U_{global}(\mathbf{a}^{k+1})$, where $\mathbf{a}^k$ and $\mathbf{a}^{k+1}$ differ in at least one camera state, for all $k = 1, 2, ...$

Definition 5. The camera network is at a Nash equilibrium if $\forall a_{-i} \in A_{-i}$ and $\forall a_i, b_i \in A_i$, $\Delta U_{global} = U_{global}(b_i, \mathbf{a}_{-i}) - U_{global}(a_i, \mathbf{a}_{-i}) \leq 0$ and $\Delta u_{local}^{c_i} = u_{local}(b_i, \mathbf{a}_{-i}^{c_i}) - u_{local}(a_i, \mathbf{a}_{-i}^{c_i}) \leq 0$.

Claim 6 *Given a fixed environment and an arbitrary set of initial camera settings, the network of cameras will improve upon the global utility U_{global} until it reaches a Nash equilibrium.*

Proof. As there are a finite number of camera network configurations, it is easy to see that every improvement path is finite and arrives at a Nash equilibrium. If the camera network is not at a Nash equilibrium then there is a set of cameras such that $\max_{a_i \in A_i}(\Delta u_{local}^{c_i}) > 0$. After applying the condition in Eqn. (5.10) the set of cameras that decide to move is $C_{move} \subseteq C$ resulting in the camera network state $\mathbf{a}_{move}$. Also, $\forall c_i', c_i'' \in C_{move}$, $c_i' \notin C_i''$ and $c_i'' \notin C_i'$. This means that the contribution of c_i' to the global utlity is independent of the contribution of c_i''. The resulting change in the global value can then be written as

$$\Delta U_{global}|_{\mathbf{a}_{move}} = \sum_{i \in C_{move}} (\Delta u_{local}^{c_i}|_{\mathbf{a}_{move}}). \tag{5.11}$$

Since $\Delta u_{local}^{c_i}|_{\mathbf{a}_{move}} > 0, \forall i \in C_{move}$, then $\Delta U_{global}|_{\mathbf{a}_{move}} > 0$. Thus by Def. (4) and (5) the camera network is either at a N.E. or moves to increase the global utility along an improvement path. ■

Chapter 6

Evaluation and System Implementation

This chapter contains the extensive experiments and analysis that have been performed to evaluate the camera control algorithm on various objective functions. The first section describes the setup and results on a simulated network of cameras and targets. Next, results are shown on a real-life camera network with known target tracks (i.e., detection and tracking are not performed live, and actors follow predefined paths). Finally a complete implementation is presented along with the results and analysis.

6.1 Simulation

This section contains the extensive experiments and analysis that have been performed to evaluate the camera control algorithm. The approach was tested on a simulated network of cameras. The goal of the simulation was to test the algorithm under a variety of conditions that are not possible with real data due to the physical limitations of the experimental setup. Also, the performance can be compared with

the ground truth. This simulation setup was very close to the real-life experimental framework .

6.1.1 Area Coverage

The first goal is to determine a set of camera parameters to cover the entire area at an acceptable resolution. To achieve this, the collaborative sensing approach described above (Sec. 5.2) is used. The area under surveillance is set up as a rectangular region of 20 by 30 meters. In the implementation presented here, the area is divided into grids in a way similar to [19] to make the problem more tractable. Thereafter, each grid of the area is treated as a virtual target (since the problem is covering the entire area). In this simulation, the grids are of size 1×1 meter. The sensor network consists of 8 PTZ cameras with a resolution of 320×240 pixels spread out around the perimeter of the area as seen in Fig. 6.1(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. Each grid in the area is treated as a virtual target. At each negotiation step the cameras can update their parameter settings giving the settings of the other cameras at the previous time step, to affect which of these virtual targets they are observing. At each negotiation step, after a camera updates its pan-tilt-zoom parameters, these parameters are transmitted to the other cameras in the 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 in turn to update its parameters. In this simulated setting, the negotiation converges to a completely covered area at an acceptable resolution after typically 17 negotiation steps. The initialization result is shown in Fig. 6.1(a). Areas in different shades of gray are covered. Any white area is not covered. The shading gets darker for a block as more cameras cover it.

Successful convergence is shown in Fig. 6.1(a), where the cameras have settled on their parameters and no uncovered region (white areas) are present.

6.1.2 Area Coverage with a High Resolution Target

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. However, a human operator could chose to observe a feature (e.g. face) of a specific moving target at a higher resolution for a specific application (e.g. face recognition). 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 on the ground plane) occupies on the image plane. This chosen target is the marked target in Fig. 6.1(b). Then, the camera that can view the specified feature of the target at a high resolution takes the task of observing the target changing its parameters. The results of the change can be seen in the dark FOV in Fig. 6.1(c).

Since the parameter change of that chosen camera, c_i, 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. 6.1(d). c_i will now predict the position of the target at time t. If c_i determines that it will not be able to observe 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 observing to another camera. As can be seen in Fig. 6.1(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. 6.1(e) we see that another camera that was participating in

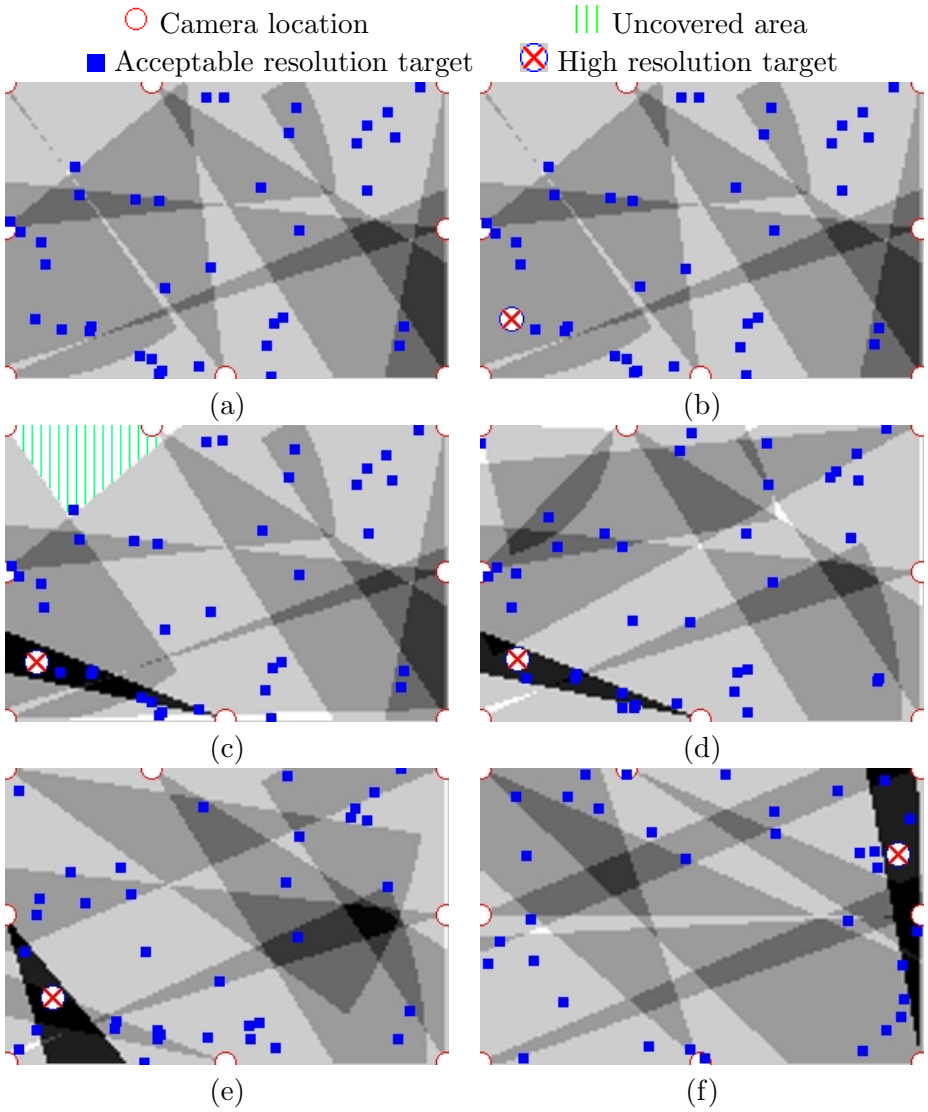


Figure 6.1: Simulated area under surveillance with several targets visible. White indicates uncovered area. Shades of grey indicate coverage. (a) Result after initial convergence shows the area of interest being completely covered by the cameras at an acceptable resolution r_v, therefore all the targets in the area are being viewed at least r_v. (b) shows the initial condition when a target (T_j^h, r_h) has been selected for observation at a high resolution r_h. In (c), c_i, the camera with the dark FOV decided that it could observe (T_j^h, r_h) and adjusted its FOV accordingly. Note now there is a part of the area uncovered due to c_i 's change of parameters. As seen in (d), some of other cameras readjusted their parameters through the negotiation mechanisms to maximize their utilities covering again the entire area under surveillance. At a later time, in (e), when c_i is not able to keep (T_j^h, r_h) in its FOV anymore, based on knowledge of T_j^h the other cameras adjust their parameters to maintain both area coverage and (T_j^h, r_h). The entire process above is repeated until (T_j^h, r_h) exits the area as in (f).

the negotiation decided that it could view the target at a high resolution at t_{out} and therefore took over the observation of the target. Again, this camera is now in charge of observing the high-resolution target and 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. 6.1(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.

Fig. 6.2 shows typical plots of utilities vs. time for this simulation. Utilities for real targets and virtual targets are shown in (a) and (b). (c) shows the global utility. As can be seen, when a camera starts observing a target at a high resolution, some area is left uncovered and the utility for virtual targets decreases. Note that the real targets are randomly simulated, actual change of utilities depends on the real scenario, and should not be the same for different scenarios.

6.1.2.1 Performance Analysis

Through another simulation, the performance of the proposed system is analyzed in the cases of area coverage and target tracking (which are two common modes of operation). The goal of the simulation here is to show the effect of the number of targets on the performance of the proposed method. The simulation was modeled very similarly to the real world setup, allowing experiments that would otherwise be prohibitive due to the setup and maintenance costs involved with active camera network experiments. The simulated targets were evenly distributed around the available area.

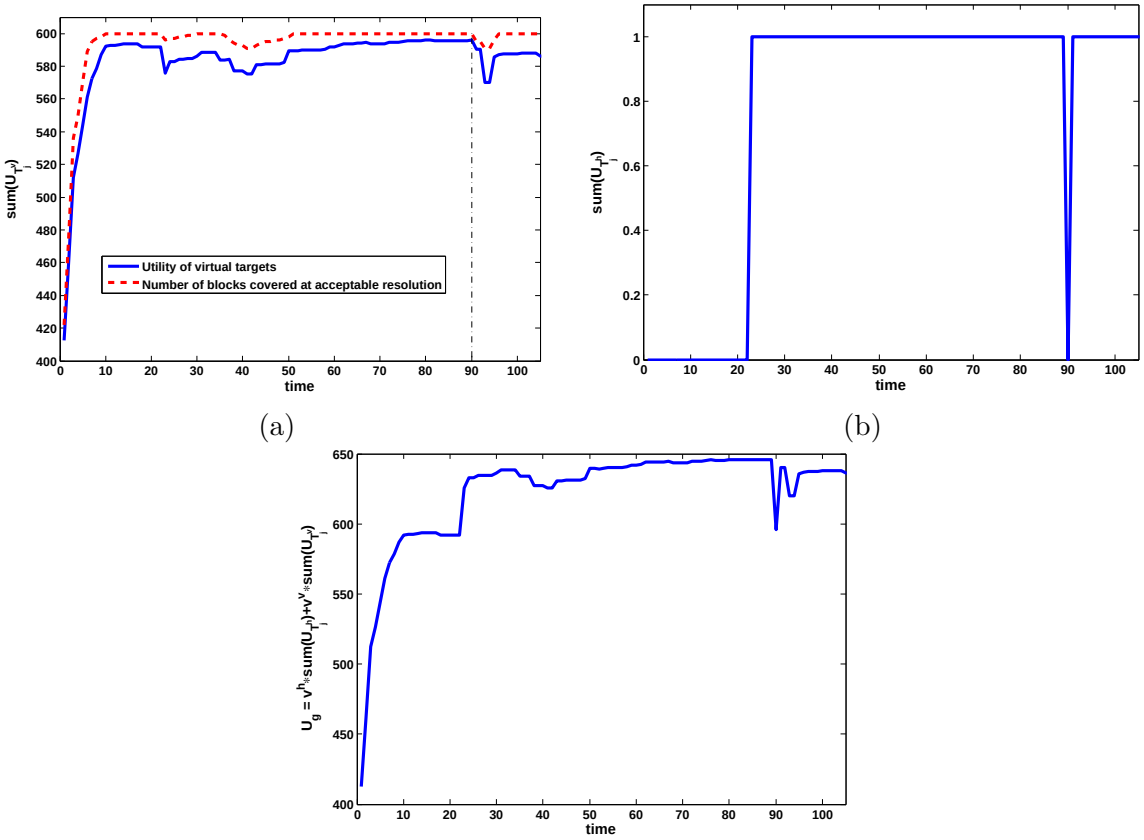
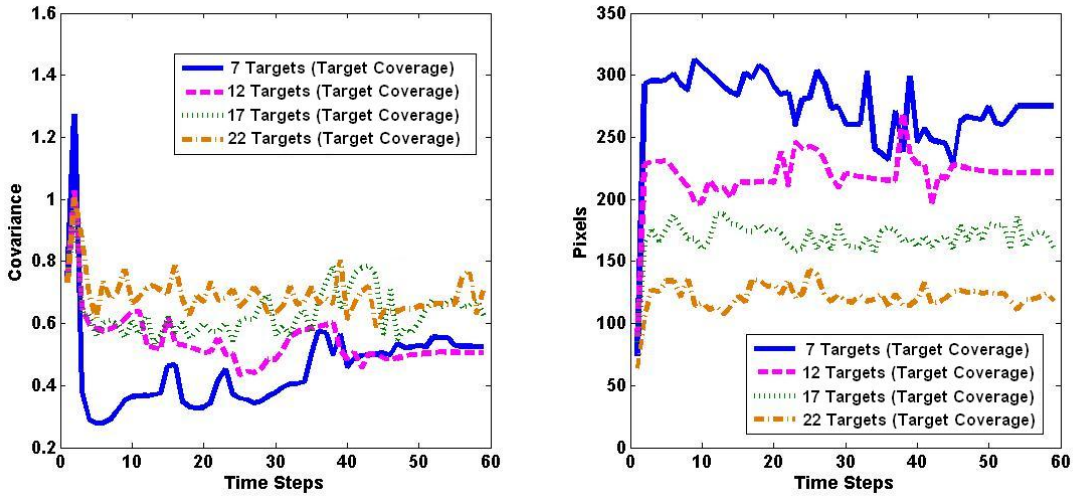
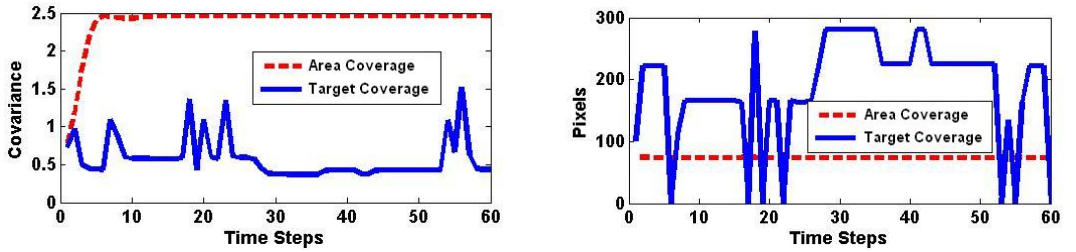


Figure 6.2: Typical plots of utilities for this simulation. (a) shows the coverage of the entire area. The solid line represents the summation of utilities for the virtual targets, i.e., $\sum_{j=1}^{N_g} U_{T_j^v}$ and the dashed line represents the number of area blocks being covered by the camera network at an acceptable resolution. The maximum number of block coverage is 600, i.e. the entire area is being covered. The summation of utilities for the real targets being viewed at higher resolution is shown in (b). The utility of real target is 0 for a very short time period around 90th sec, because the cameras are in the process of target handoff; hence, no camera views the target at high resolution in this period, but the target is still viewed at an acceptable resolution (can be inferred from (a) as entire area is being covered in that period). At all other times, the high resolution is maintained on the real target.In (c), global utility is plotted, here the importance values of real targets and virtual targets are set to be 50 and 1 respectively.



(a) (b)

Figure 6.3: The effects of increasing the number of targets on tracking covariance and image resolution of the targets. (a) shows that as the number of targets in the scene simultaneously increases, the error covariance also increases. This is expected as the cameras must now image a larger region. (b) shows that the increase in targets results in the average resolution of all the targets being imaged decreases. Since more targets require more of the area to be observed the cameras must zoom out. As the number of targets increases to fill the entire area, the average tracking accuracy and resolution will decrease to the area coverage case.



(a) (b)

Figure 6.4: The tracking covariance and resolution of a single target $T_j \in N_t$, for $|N_t| = 22$. (a) shows that as the target moves throughout the area, there are time instants where no measurement is acquired. This is reflected by the spikes in the error covariance when doing target coverage. However it can also be seen that the network will reacquire the target and the overall error covariance is significantly lower than when only optimizing for area coverage. (b) shows the resolution of the target during the scene. Notice that at some time instants the target is unobserved resulting in a resolution of 0 pixels. This does not mean the target is lost by the tracker, but that a measurement is not acquired by a camera in the network.

Figs. 6.3(a) and (b) show that as the number of targets increases, the average tracking covariance does not change significantly, in contrast to the change in average resolution. Note that the resolutions are still at a much better value comparing with the system for Area Coverage (not shown in Fig. 6.3 for clarity of details). An example of one of the targets is shown in Fig. 6.4. The reason for the small effect on tracking covariance increases is due partly to the low weight to the risk in our experiments. This allows the system to choose to leave some targets unobserved at any given time step in favor of improving poorly tracked or poorly resolved shots of targets. We can see in Fig.s 6.4(a) and (b) that at some time instants the target is unobserved. At these instants, the number of pixels imaged is zero and the tracking covariance increases. It is also clear that as the tracking covariance increases, the greater the incentive for the system to image the target at a setting that minimizes the increasing error. Thus, the system will zoom out in order to get an image of the target. This is seen in the plots in Fig. 6.4 where the target is reacquired.

6.1.3 Facial Imaging and Tracking Accuracy

This section shows the results in simulation to discuss the performance of the system tasked with optimizing the tracking accuracy of targets in an area while opportunistically acquiring high resolution face images. The system is setup such that the network optimizes tracking accuracy until a tracking threshold is satisfied before considering taking high resolution face images of targets in the area.

At initialization, all the cameras adjust themselves to cover the entire region under surveillance to acquire state estimates of targets already in the region. For the simulation and corresponding real life experiment, value for obtaining a high-resolution

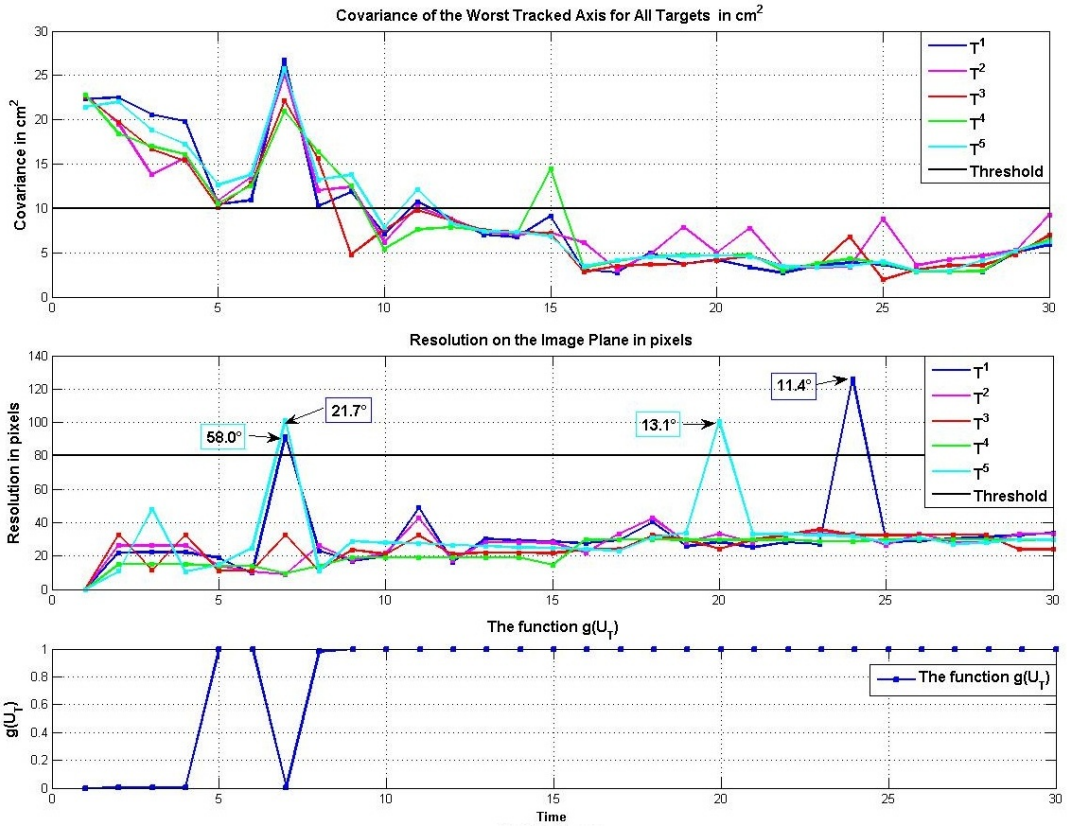


Figure 6.5: Plots for the tracking covariance, image resolution, and the function $g(U_T)$, for $N_T = 5$ targets. As the tracking covariance for all targets approaches the min. covariance threshold of $10cm^2$,, a non-zero value for imaging utility weight $g(U_T)$ is obtained. Subsequently, at time-step t_7, the resolution, pose and/or distance requirement is met, and high-resolution images of targets T_1 and T_5 are captured at 58.0° and 21.7° from the desired angle. This causes degradation in tracking performance and the tracking covariance increases. After tracking specifications are met again, more high resolution images are obtained at time-steps t_{20} and t_{24}, for targets T_1 and T_5, at angular distance of 11.4° and 13.1°. The second set of high-resolution images for the same targets are obtained, due to an improvement over the previous viewing angle.

image would only be added if the tracking performance of the network on all targets met a pre-defined threshold. Also, maximizing the Bayesian Value accounts for the risk of failing to image the target. The area under surveillance was set up as a rectangular region of 20 by 30 meters. For the purpose of simulation, a sensor network of $N_C = 4$ calibrated cameras is set up, located on the four corners of the rectangular region, and the simulation is run for $T = 30$ *secs.* All cameras were assumed to have a resolution of 320×240 pixels.

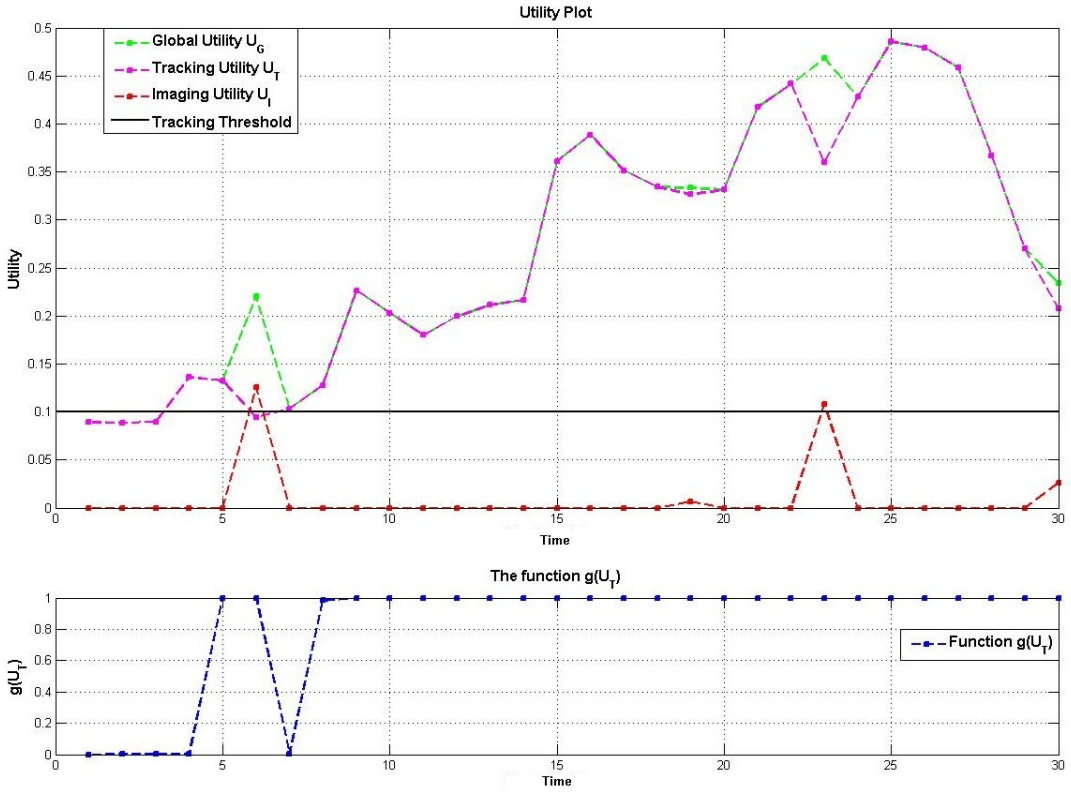


Figure 6.6: Plots of utilities for our simulation. The global utility $U_G(\mathbf{a})$ represents the summation of tracking and imaging utilities. The tracking utility $U_T(\mathbf{a})$ is a measure of the tracking performance of the least accurately tracked target. As $U_T(\mathbf{a})$ satisfies tracking threshold $\bar{P}$, a non-zero value for the function $g(U_T)$ is obtained. If pose, resolution and distance requirements for imaging the target are satisfied, then a spike for the imaging utility $U_I(\mathbf{a})$ can be seen. At time-step t_4, the tracking threshold is satisfied, and a non-zero value for $g(U_T)$ is obtained. At time-step t_6 the pose, resolution and/or distance requirement for a target is satisfied and thus a high value for $U_I(\mathbf{a})$ is seen. This results in capture of a high-resolution image of a target, but also leads to degradation in tracking performance, which can be seen in reduction in $U_T(\mathbf{a})$ at t_6. Another high-resolution image is obtained at t_{23}, leading to degradation in tracking. But, in spite of degradation in $U_T(\mathbf{a})$, it stays above $\bar{P}$, thus enabling $g(U_T)$ to have a non-zero value.

The tracking utility U_T is different from the one described earlier in that rather than minimize the average trace of the covariance, it minimizes the trace of the covariance of the worst tracked target. The weight $g(U_T)$ is a function that scales the value of the face imaging utility depending on how well the tracking threshold is satisfied. The imaging utility U_I considers both the pose and resolution of each target and with the minimum and maximum acceptable pose changing over time as face images closer

to the desired frontal face view are acquired. This was achieved by setting the minimum acceptable pose to be closer to the desired pose than any previously acquired face images.

From Fig.(6.5) it can be seen that the cameras cooperate to reduce the tracking covariance at every time step. Once the tracking accuracy is near the predefined threshold, cameras that have the ability to obtain desirable high resolution images of features may then gain utility for acquiring them. Due to measurement noise, the predicted information and the actual information gained from a measurement may not be equal causing the network to not meet the tracking specification. This causes the weight $g(U_T)$ to drop at time-step t_6 preventing cameras from gaining utility from acquiring high resolution images of features. More high resolution shots are captured at time-steps t_{20} and t_{24}. These images are acquired since the tracking requirements are satisfied and the cameras can obtain high resolution images of the features at better poses.

Fig.(6.6) shows the individual utility functions across simulation time. When the tracking specification is sufficiently satisfied, a sufficiently high value for the weighting function $g(U_T)$ is obtained, which enables the imaging utility $U_I(\mathbf{a})$ to be added to the global utility $U_G(\mathbf{a})$. Thus, in times of opportunity, $U_I(\mathbf{a})$ is added to $U_G(\mathbf{a})$, subject to tracking specification being satisfied, maximizing the global utility.

6.2 Real-life Camera Network with Known Tracks

6.2.1 Area Coverage with Multiple High Resolution Targets

The sensor network consists of 9 PTZ cameras with a resolution of 320×240 pixels. This setting is used to show the performance of the approach with an increasing number of zooming cameras.

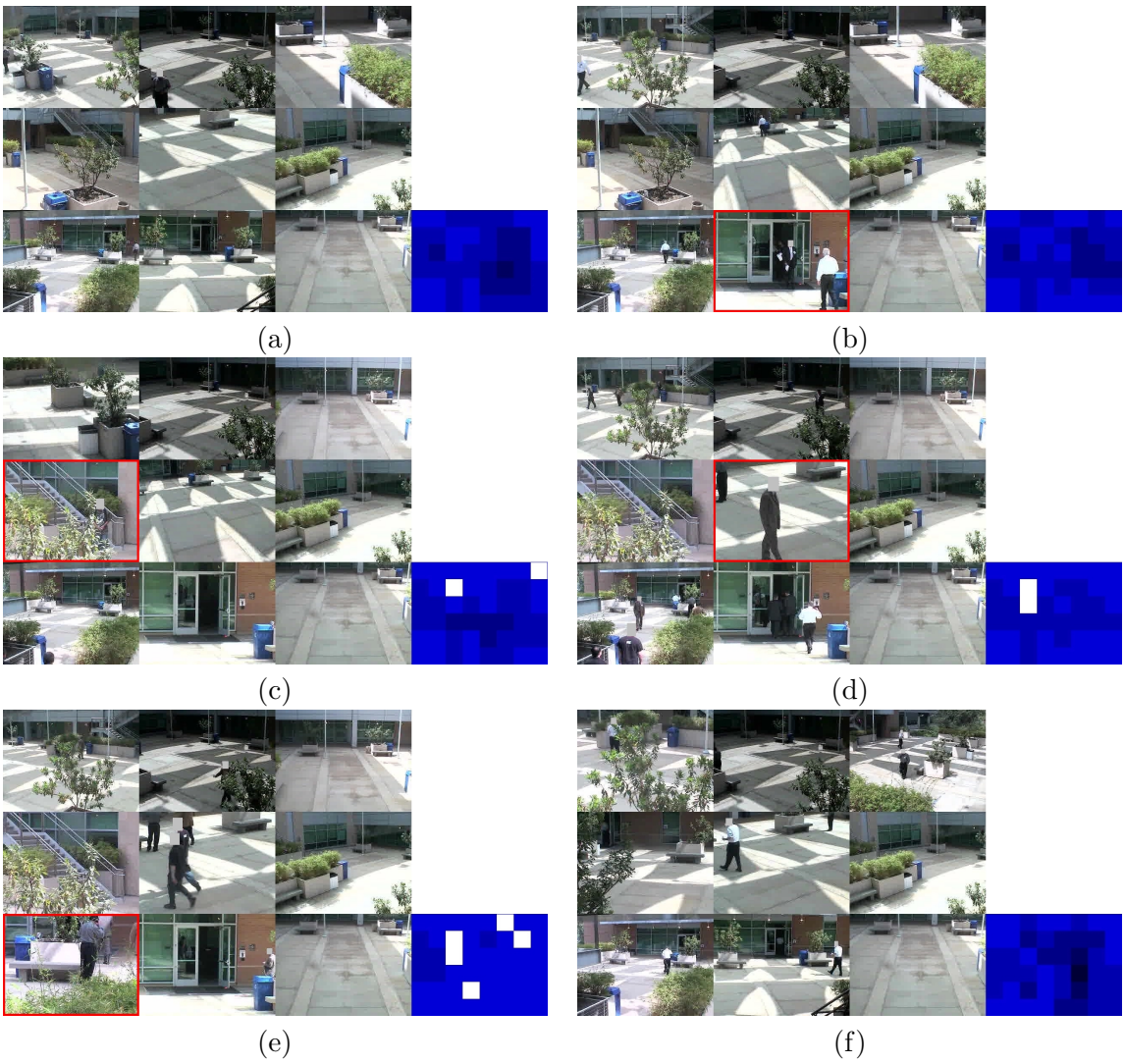


Figure 6.7: Results of game theoretic camera control with the number of cameras that zoom in increasing. The coverage of the entire space is shown on the right-bottom of each sub-figure, where blue areas are covered, white areas are not. The darker the color of a block, the greater the number of cameras that are assigned there. The results with multiple cameras zooming in are showed in (b)-(e). The number of cameras that zoom in is increasing from 1 to 4. From (b) to (e), the view of new zooming in camera is bounded by red lines. It is seen that as the number of zoomed in cameras increases, more areas are left uncovered. (f) shows the re-initialization results when we reset the cameras with none of them zooming in.

Fig. 6.7 demonstrates the ability of the game theoretic approach to self-organize to maintain coverage while allowing some cameras to zoom in for high resolution images. Fig. 6.7 (a) shows the initial convergence result that covers the entire area at an acceptable resolution. The coverage is shown on the bottom-right (blue areas

are covered, white areas are not; the darker the color of a block, the greater the number of cameras that are assigned there). Fig. 6.7 (b) shows that when one camera zooms in (camera c_8, bounded with red lines) the other cameras automatically adjust their settings (camera c_5 zooms out) to keep the area covered. Fig. 6.7 (c)-(e) show the affect of increasing the number of cameras zooming in from 2 to 4. It is seen that as the number of zoomed in cameras increases, more areas are left uncovered. Fig. 6.7 (f) shows the re-initialization result when we reset the cameras with none of them zooming in - the network of cameras can again keep the entire area covered. By comparing 6.7 (f) with (a), it can be noticed that the parameter settings in (f) are different with those in (a), although both of them could satisfy the coverage requirements. This illustrates that the Nash equilibrium is not unique.

6.2.2 Cover Area Entrances while Optimizing Tracking Accuracy

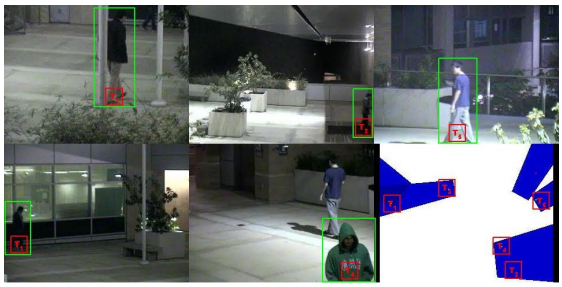
This sections shows the results on a real-life camera network optimizing the tracking accuracy of all targets in an area, and is compared to a camera network optimizing for area coverage.

The camera network for this experiment was composed of 5 PTZ cameras surrounding a 600m^2 courtyard. The area was divided into a number of grids, each of size 1cm^2. Tracked targets were assumed to have a height of 1.80m. Each camera acquires images of resolution 640×480 pixels. Thus r_i^j is the largest number of pixels in the vertical direction occupied by T_j in the image plane of some camera in the network. The cameras were arranged such that 4 of the cameras were located on the same side of the courtyard, with one camera on the opposite side. In the region of interest there were 5 targets in addition to two entrances and exits. Since the entrances and exits

(a) Time Step $k = 0$

(b) Time Step $k = 2$

(c) Time Step $k = 19$

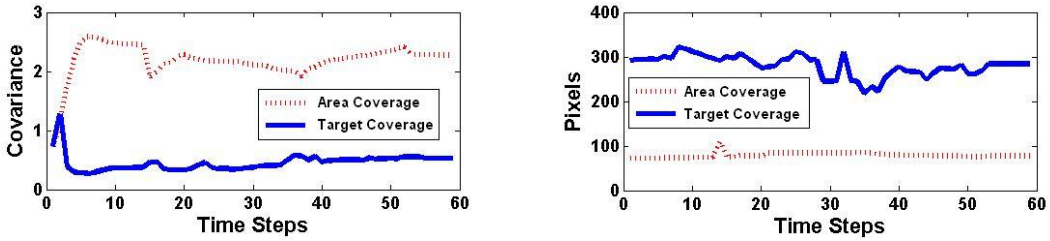


(d) Time Step $k = 36$

Figure 6.8: Dynamic camera control images. Blue regions mark the field of view, lighter regions identify camera overlap. Targets are marked in green with red label. (a) shows the initial setting of the camera network when the entire area is covered. It is plain to see that targets are very small and hard to make out. (b) - (d) show images over the course of the scenario as targets move around the area. There are significantly more pixels on each target than when optimizing only for area coverage.

must be monitored always, they were treated as static virtual targets, leading to a total of 7 targets. Each camera in the setup was an independent entity connected through a wireless network, with the entire system running in real time. For this experiment the targets were assigned specific paths to walk and the detections used were generated from these predefined paths. In an actual application the detections would be generated through use of an actual detector, such as a image based person detector.

At initialization, all of the cameras apply the utility function defined in Sec.4.1.1 for the Area Coverage to cover the entire region under surveillance and to detect targets already in the region. The target detection modules in each camera determine the image plane position of each target in its field of view. This information is then passed along to the Extended Kalman-Consensus filter and is processed along with the information



(a) (b)

Figure 6.9: Comparison of the average tracker covariance and average resolution of all targets being actively tracked by a system for Target Coverage vs. one for Area Coverage. (a) shows the average trace of tracker covariance of targets (i.e., $\mathbf{P}_i^j$) as they move throughout the area. What can be seen is that for the same paths, the system optimizing for tracking performance results in a smaller error covariance of the positions of the targets than the system interested only in area coverage. (b) shows the average resolution of targets over time (i.e., r_i^j). Again notice that the system interested in optimizing tracking performance also results in targets being imaged at a significantly higher resolution than possible in the area coverage case.

from the filters running on neighboring cameras as described in Sec. 3.3.4.

In the Area Coverage problem, the camera networks has to cover the entire area and take shots at lower resolutions, resulting in increased tracking error. The following section will present the results for both the Area Coverage and Target Coverage system (where only targets and entrances are covered), and clearly show the advantages of optimizing the tracking within the control module. The targets followed the same path through the courtyard during the collection of data for both cases. Fig. 6.8 shows the images captured by the actively controlled camera network at different time steps. Fig. 6.8(a) shows the result for the Area Coverage as the initialization. Fig. 6.8(b)-(d) show the results for the Target Coverage. Since targets are free to move about the region under surveillance, the cameras in the network are required to adjust their parameters dynamically to maintain shots of each target that optimize the utility functions presented in Section 4.2.2. To acquire these shots the camera network concedes a large unobserved area. It can be seen in Fig. 6.12 that as time progresses, the average

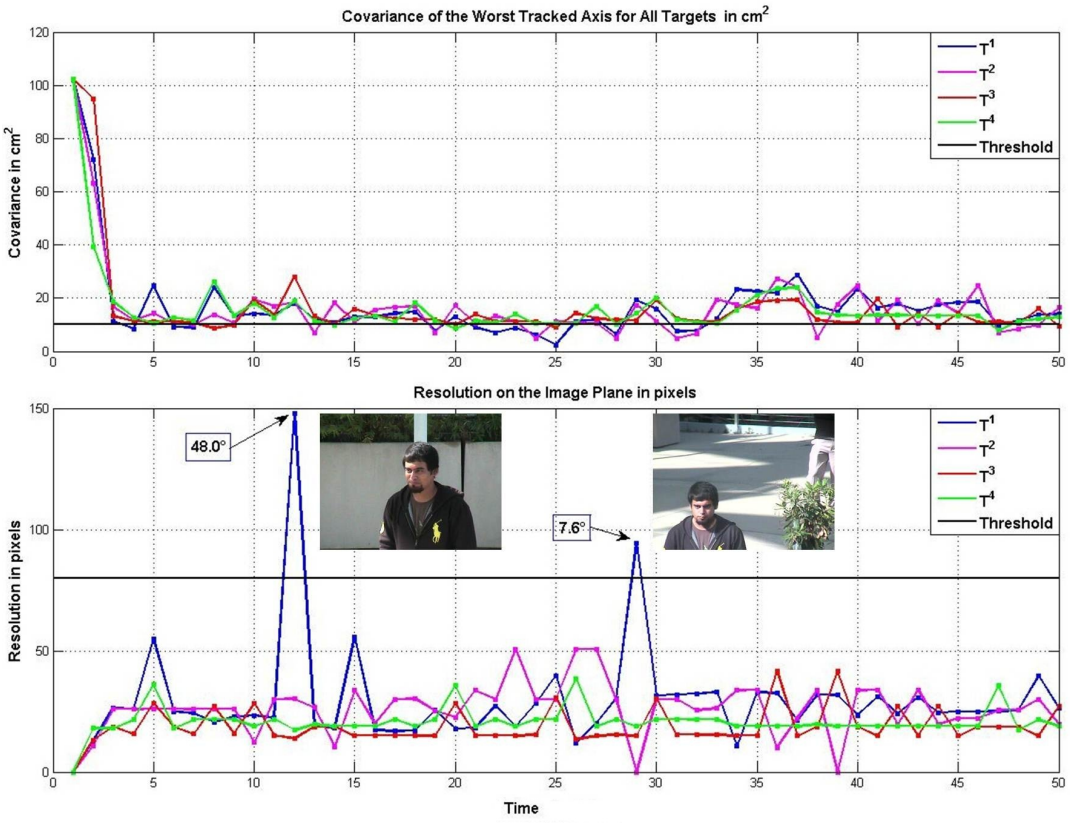


Figure 6.10: Plots for the tracking covariance, image resolution, and the function $g(U_T)$, for $N_T = 4$ targets. As the tracking covariance for all targets approaches the min. covariance threshold of $10cm^2$, a non-zero value for imaging utility weight $g(U_T)$ is obtained. Subsequently, at time-step t_{12}, the resolution, pose and/or distance requirement is met, and a high-resolution image of target T_1 is captured at 48.0° from the desired angle. This causes degradation in tracking performance and the tracking covariance increases. After tracking specifications are met again, another high resolution image is obtained at time-step t_{29} for target T_1, at angular distance of 7.6°. The second high-resolution image for the same target is obtained, due to an improvement over the previous viewing angle.

trace of the covariance and the resolution of all targets settle at a significantly better value (compared to the Area Coverage) when the tracking and control modules are integrated together (as proposed in this paper). This is because at each time step the camera network will choose the set of parameters that optimizes the utility, which is dependent on the error covariance of the Extended Kalman-Consensus filter.

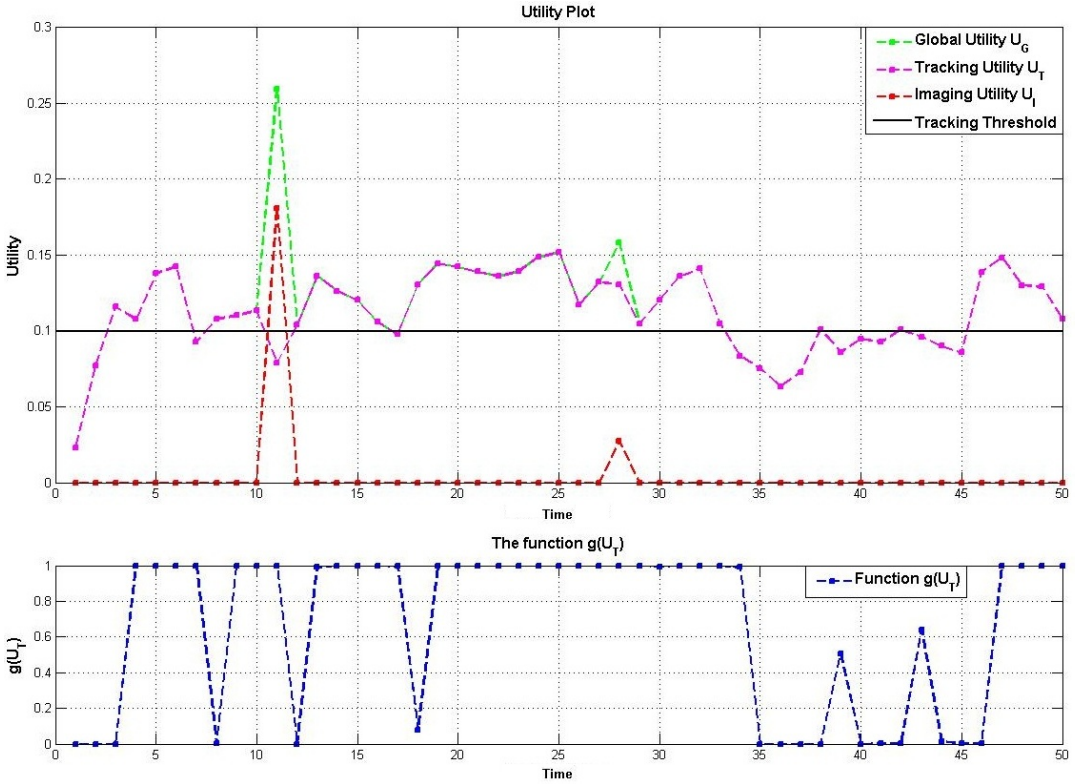
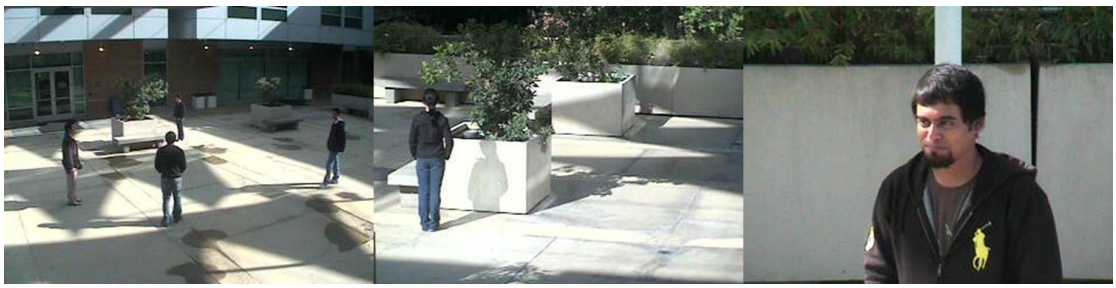


Figure 6.11: Plots of utilities for the real-life experiment performed on a distributed camera network of $N_C = 3$ cameras, to track and image $N_T = 4$ targets. The global utility $U_G(\mathbf{a})$ represents the summation of tracking and imaging utilities. The tracking utility $U_T(\mathbf{a})$ is a measure of the tracking performance of the least accurately tracked target. As $U_T(\mathbf{a})$ satisfies tracking threshold $\bar{P}$, a non-zero value for the function $g(U_T)$ is obtained. If pose, resolution and/or distance requirements for imaging the target are satisfied, then a spike for the imaging utility $U_I(\mathbf{a})$ can be seen. At time-step t_7, the tracking threshold is satisfied, and a non-zero value for $g(U_T)$ is obtained. At time-step t_{11} the pose, resolution and/or distance requirement for a target is satisfied and thus a high value for $U_I(\mathbf{a})$ is seen. This results in capture of a high-resolution image of a target, but also leads to degradation in tracking performance, which can be seen in reduction in $U_T(\mathbf{a})$ at t_{11}. Another high-resolution image is obtained at t_{28}, leading to degradation in tracking. But, in spite of degradation in $U_T(\mathbf{a})$, it stays above $\bar{P}$, thus enabling $g(U_T)$ to have a non-zero value.

6.2.3 Facial Imaging and Tracking Accuracy

The orientation of the target was determined by its estimated velocity vector if the magnitude was above 10 cm/s. Otherwise it is assumed to be unknown and the $U_I^j(\mathbf{a})$ for that target will be 0. The face was assumed to be located in the topmost 40 cm of the target's height and the expected resolution was the height of this region in pixels. This



(a) Images at time-step t_{11}

(b) Images at time-step t_{29}

Figure 6.12: Images captured by $N_C = 3$ cameras at time-steps t_{11} and t_{29} of the experiment. The first high resolution image of a target is acquired at t_{11} by C_3 at angle 48.0° from the desired angle. Another high-resolution image of the target is captured by C_2, at t_{29} at an angle 7.6° from the desired angle.

experiment used a sensor network of three calibrated PTZ cameras and $N_T = 4$ targets, located in and around the area. As H_i^j is dependent on the camera parameters, we calibrated each camera at a particular setting and the rest of the homography matrices were determined by modifying the values of pan tilt and focal length. The results are shown for a period of $T = 50$ seconds. All cameras were set to resolution of 320×240 pixels.

The plots of the utility functions along with the tracking error covariance and the resolution of the faces are shown in Fig.(6.10). By using only $N_C = 3$ cameras to maintain coverage, many situations where the tracking threshold $\bar{P}$ is not met can be clearly seen in the $g(U_T)$ plot in Fig.(6.11). At time-step t_{11} the pose and resolution requirement for a target is satisfied and thus a high value for $U_I(\mathbf{a})$ is seen. This results in capture of a high quality face image at time-step t_{12} of T^1 at 48.0° from the desired

angle. It also leads to reduction in tracking performance due to having one less camera generating measurements, which can be seen as a reduction in $U_T(\mathbf{a})$. Another high-resolution image is obtained at t_{28} for target T^1, at angular distance of 7.6°. The second high-resolution image for the same target is obtained, due to an improvement over the previous viewing angle. This leads also to degradation in tracking. But, in spite of degradation in $U_T(\mathbf{a})$, it stays above $\bar{P}$, thus enabling $g(U_T)$ to have a non-zero value. We can see that in the acquired images the faces of the target are much easier to see than typically available.

6.3 Complete Real-life Implementation

This section shows the results of a system tasked with optimizing the area coverage of a camera network while opportunistically acquiring high resolution images of targets when certain events occur. The events considered in this experiment are when new targets become tracked, targets merge to form groups, and when groups split into smaller groups or individual tracks. The experiments were performed on a wireless PTZ camera network consisting of Axis 215 PTZ-E cameras over a $20m \times 30m$ region. Since the capability to program the cameras directly was not present, the video and commands must be sent across the wireless network through the provided VAPIX API resulting in poorer response time due to latency. The distributed environment was simulated by assigning each camera its own thread. In the physical system a delay ranging from $80ms$ to $200ms$ was present for each message to arrive over TCP. This limits the rate at which the parameters of the camera network can change as there is significant delay in the video response after sending a command.

6.3.1 Detection & Tracking

The detection method used in this experiment was very simple in design. First the area was uniformly divided into a number of overlapping blocks, each $1m \times 1m \times 1.8m$. The frames from each camera are processed using a motion subtraction algorithm to generate a motion image. The blocks in the cameras operational region are then mapped to the image plane using the homography.

This mapping combined with the motion image results in a probability of a person being at every block in the field-of-view of the camera. The probability distribution across the blocks is then segmented to form our detections, represented by a sample mean and sample covariance. Once all the detections have been generated for the current frame, the sample mean and covariance of the detections are used to associate detections to existing tracks.

In this implementation no assumptions are made as to the number of people present in each detection or track. Two people walking close together may result in a single detection with a covariance encompassing both. This means that if two people stay close together for the duration they are in the region they will be tracked as a group rather than as two individual tracks. Since the number of people in each track is not explicitly counted, a few special events can occur.

Split track: When two or more people who are walking together start drifting apart, the detector will return multiple detections that associate to their existing track. The existing track is removed and one new track for each detection is created.

Merge track: When two or more people start walking ever closer to each other, the detector will eventually return a single detection for all of the people. This event occurs when one detection associates to more than one track, this means the

measurement covariance contains the expected position of more than one track. In such cases the existing tracks are removed and a new track is created for the group.

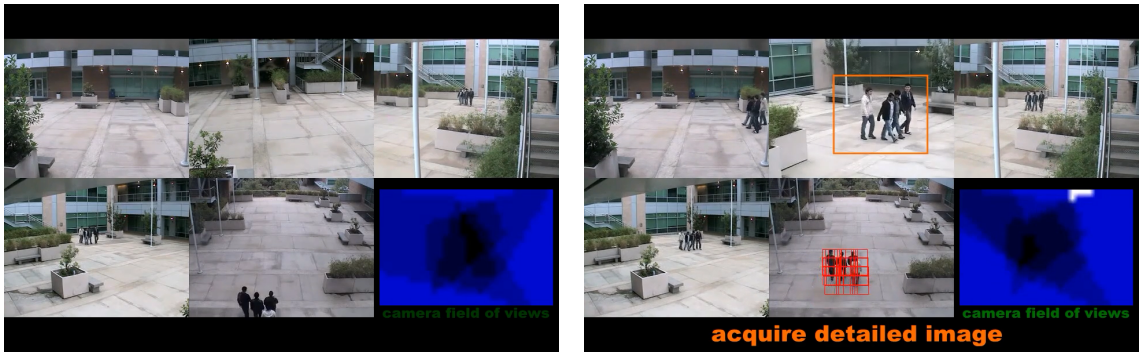
One thing to note here is that as people form groups or split apart there is a short time period where multiple track split and track merge events may occur rapidly until they are close enough to form a group or far enough apart to be tracked separately. The scaling parameter s_j is tuned such that high resolution images are not prioritized until after a track has has existed for a period of time so as to avoid capturing multiple high resolution images during merge and split events.

6.3.2 Parameter Selection

There are many ways to speed up the search for the PTZ setting that maximizes the local utility, the easiest approach is to increase the quantization of the parameter space. For this experiment the available pan settings were quantized into 5 degree increments and bounded based on the position of the camera and the area under surveillance. The tilt settings were quantized into one degree increments and was restricted to a maximum tilt of 30 degrees. Each camera also was restricted to three different zoom settings. Some additional speed up techniques such as a lookup table were also used to improve the computation speed such that each camera can detect, track and decide on parameters within a $33ms$ time window on a $2.66GHz$ Intel Core i560m. However, the communication overhead of the physical setup reduced the rate at which the cameras change their parameters due to a latency of around $80 - 120ms$. Directly implementing the control on the camera hardware itself would eliminate the delay between deciding on the new parameters and the computer vision modules from receiving the video resulting from the parameter change.

6.3.3 High Resolution Imaging and Area Coverage

First Scenario



(a) (b)

Figure 6.13: A group of people enter the area and a high resolution image of the entire group is taken. (a) Shows the initial coverage of the camera network. The blue shaded region shows the area coverage of the current camera settings. Lighter regions indicate less overlap in FOV. (b) Shows the resulting views and area coverage after one camera in the network self selects to acquire a high resolution image of the targets being tracked. Detections are shown as red boxes in the bottom middle camera.

In this short scenario a group of 4 people enters the scene and stays together as they cross the courtyard. The expected behavior of the system is that the cameras will first cover the entire area, then create a single track for the group and acquire of a high resolution image of the group some time before they exit the area.

The first case to test would be if the cameras successfully cover the entire area prior to the entry of any targets . This can be seen in Fig. (6.13(a)) where the decided upon settings cover the whole region. Fig. (6.13(b)) shows the group of four people walking together as they cross the courtyard. As they are walking close together, their detections are difficult to separate and a single track is used to represent the entire group. A short time period after entering the area, a high resolution image of the whole group was taken. The impact on the overall coverage of the area by taking this action was minimal and is represented by the white uncovered region. Once the high resolution

image was acquired the camera returned to the area coverage task resulting in the entire area being covered by the camera network.

Second Scenario

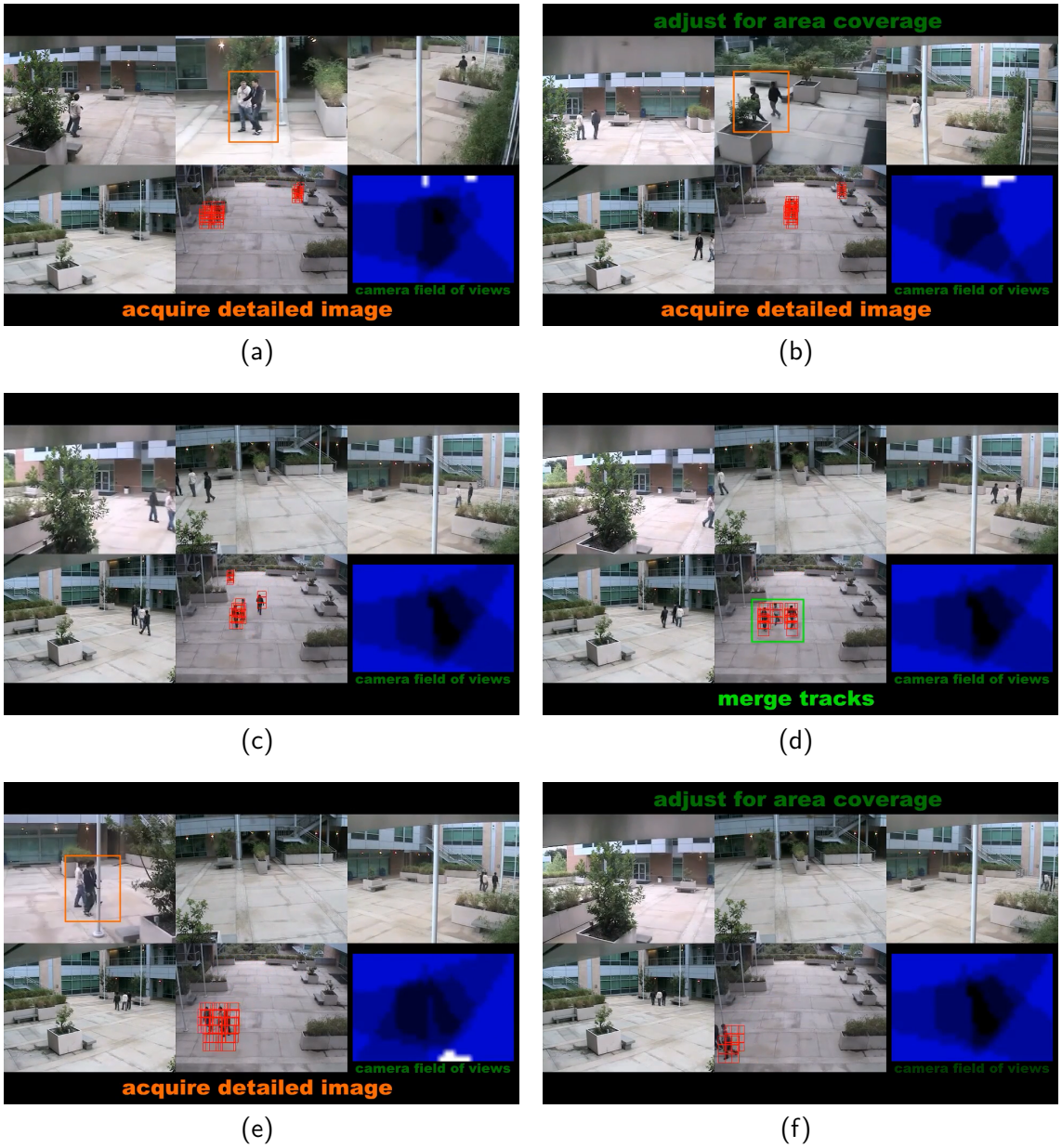


Figure 6.14: Scene involving a 2 pairs of people. The images show the resulting behavior of the camera network as the targets interact.(a) High quality image of the first pair is taken. (b) A high quality image of the second pair is acquired by the camera. (c) Two individuals and a group of two people are distinguishable by the detections, shown in red boxes, in the bottom middle camera. (d) Shows that the detections of the group of two and an individual in the scene can no longer be separated and a new group is formed. (e) A high resolution image is acquired by the upper left camera. (f) The cameras reconfigure the cover the area.

In this scenario two pairs of people enter the scene from opposite sides of the courtyard. One pair splits up shortly upon entering the area, while the other pair stays together. Some time after the first group split, one of its members walks towards and joins together with the other pair forming a group of three as they proceed to exit the area.

The system is expected to form a track and acquire a high resolution image for each of the initial pairs of people this can be seen in Fig. (6.14(a)) and (6.14(a)). After the first group splits, two new tracks are expected to be created and high resolution images of each should be taken. When the track of the second pair and one of the members of the first pair merge, the system is expected to create a new track and capture a corresponding high resolution image.

The rest of the images show the behavior of the camera network in response to the merging of the tracks. In Fig. (6.14(c)) we can see three tracks, two people walking alone and a pair of people walking together. As two of the tracks start getting closer, it becomes more difficult for our basic detector to separate the pair of people from the person walking alone. This can be seen in Fig. (6.14(d)) as the tracks merge to form a group of three people. A high resolution image is taken in Fig. (6.14(e)) a short while after the formation of the new track. After the high resolution image has been acquired the camera returns to the area coverage task and the region is again completely covered as shown in Fig. (6.14(f)).

Third Scenario

The final scenario is the most complicated and shows how the system behaves when it does not have the time to complete all the desired tasks. In this scenario four

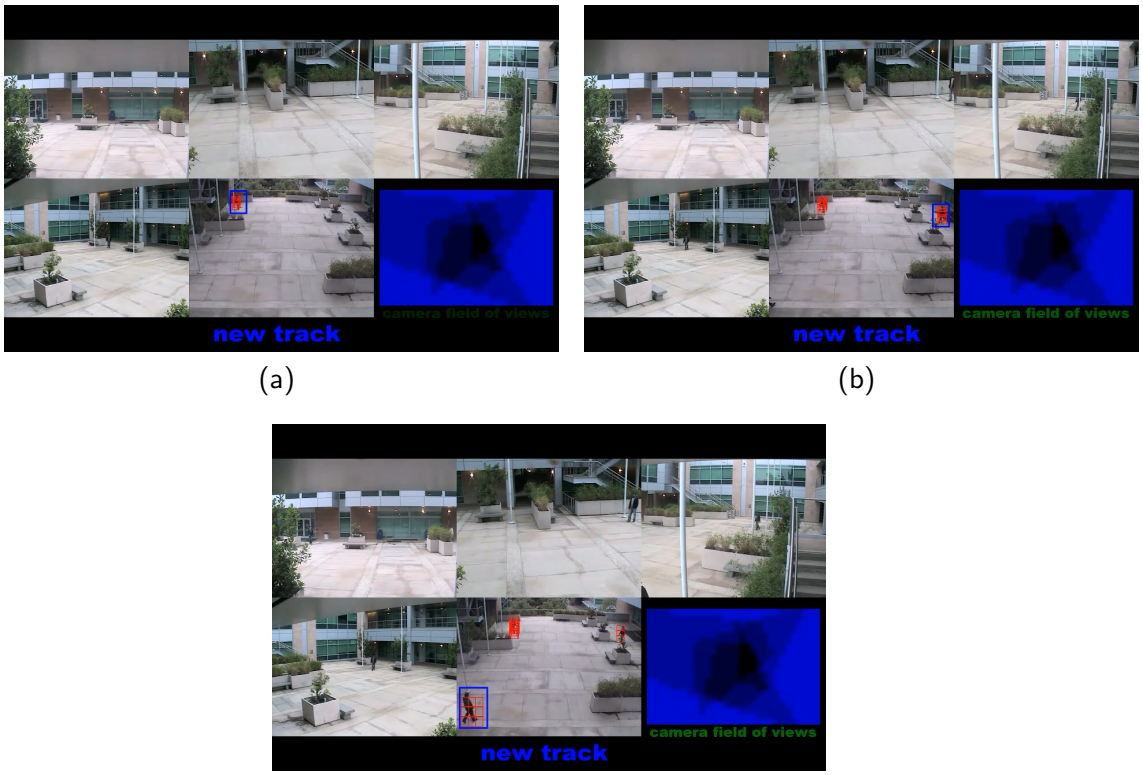


Figure 6.15: Sequence of images showing three targets entering the area under surveillance in rapid succession spawning a new track for each. Detections are shown as red boxes in the bottom middle camera.

people enter the courtyard from each of the four corners within seconds of each other. They each walk towards the opposite end of the courtyard resulting in the four of them meeting in the center of the courtyard before reaching their final destinations at the opposite corner from which they entered.

Ideally the camera network will be able to capture high resolution images from each of the targets before they become difficult to segment by being too close another target. When the targets come together briefly in the center of the courtyard another high resolution image should be acquired as the tracks would have merged together to form a new track. Finally once distinct tracks can again be made for each of the targets, another high resolution image should be taken for each target before they exit the area.

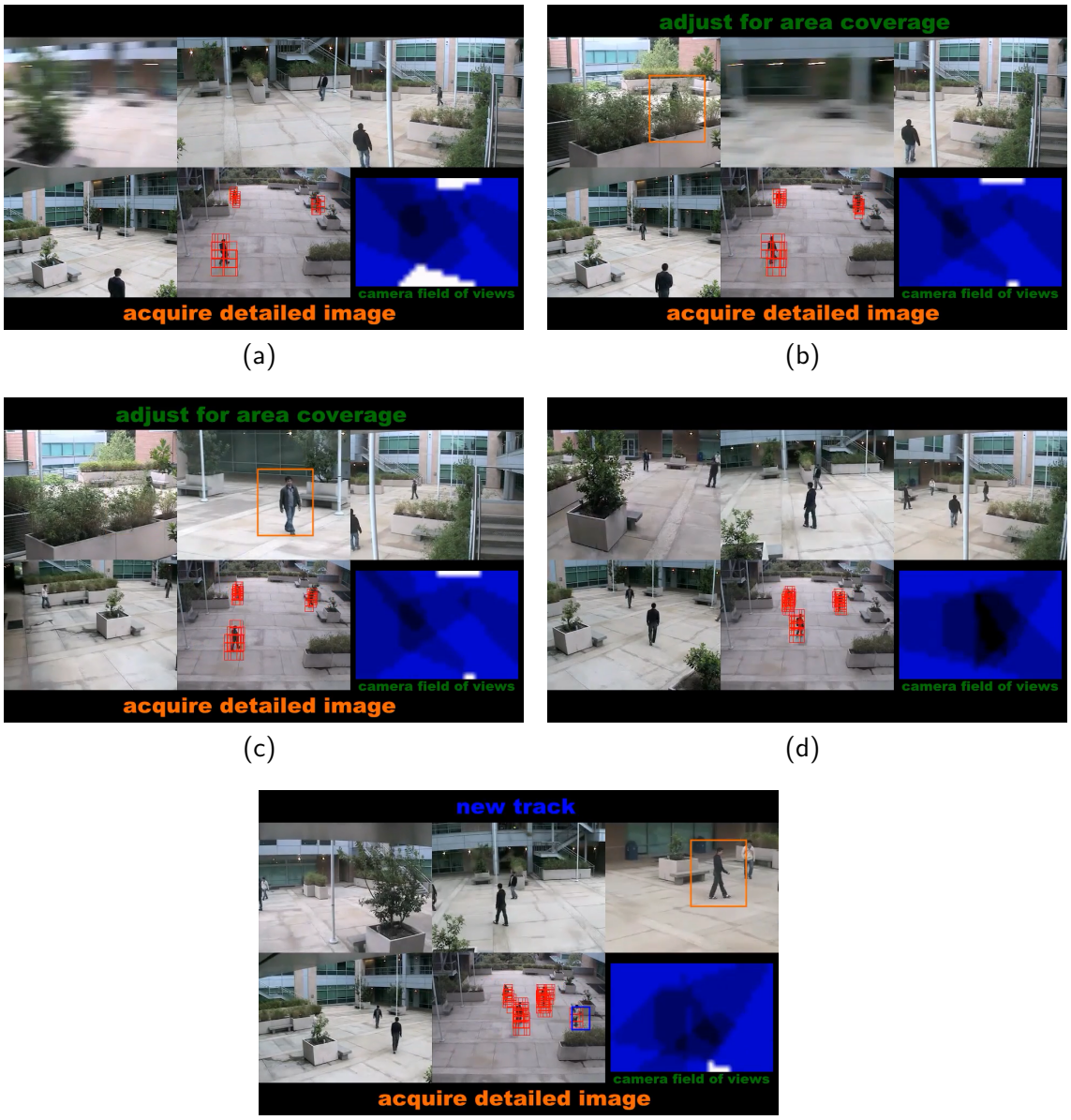


Figure 6.16: The behavior of the cameras when three targets need high resolution images to be taken. (a) Shows the result of the upper left camera moving to capture a high resolution image of a target. (b) The high resolution image is captured while other cameras in the network adjust to recover the unobserved region in white. (c) A high resolution image of another target is acquired while maintaining area coverage. (d) The views after the cameras have completed their high resolution capture of the first two targets and have returned to area coverage. (e) The high resolution image of the third target is acquired by the upper right camera as a new target enters the scene.

From images in Fig. (6.15) we see 3 people entering the scene from different corners of the area within seconds of each other. Since they are far apart, a new track

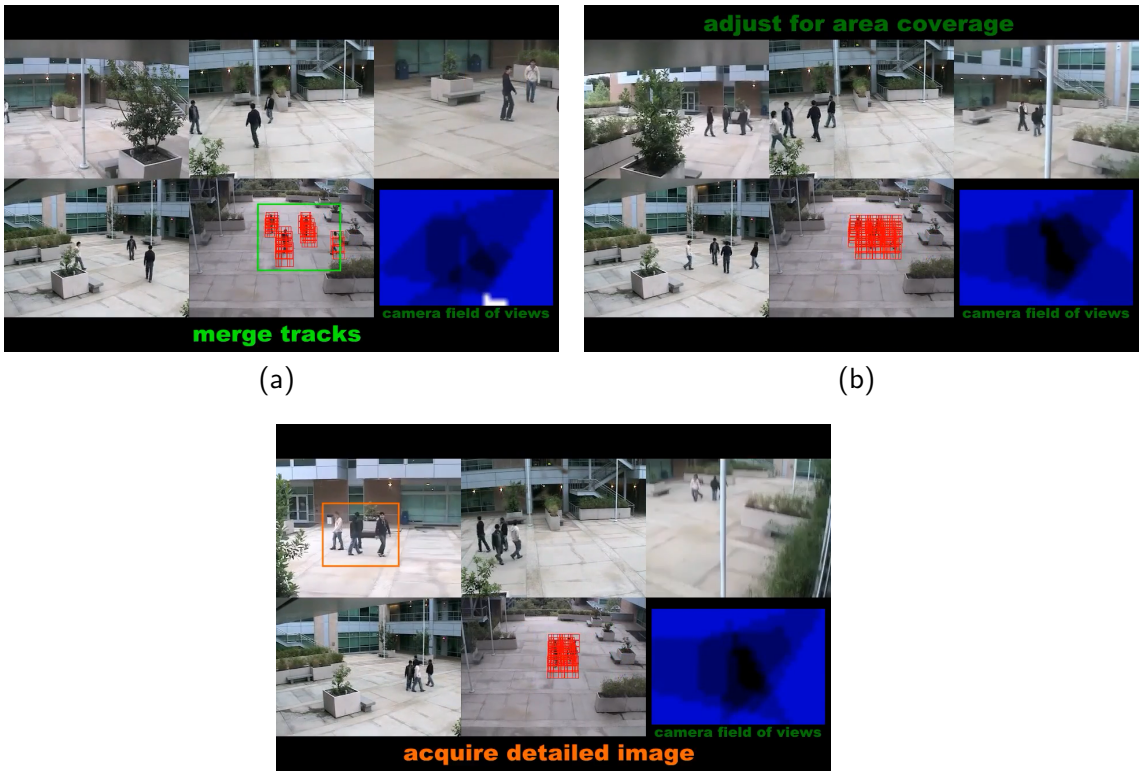


Figure 6.17: Shows the sequence of images taken in response to merging tracks. (a) As the targets move closer together their tracks begin to merge into a single track. (b) The 4 people have grouped together to create a new track. (c) A high resolution image of the entire group is taken by the upper left camera.

is created for each of them. A short while later in Fig. (6.16(a)), the top left camera decides that it should move to get a high resolution image of one of the targets. This leaves an uncovered region, represented in white, that is immediately recovered by the readjustment by some other cameras in the network, seen in Fig. (6.16(b)). While this is happening, the top middle camera in Fig. (6.16(c)) decides to, and acquires a high resolution image of another target. After acquiring these high resolution images the cameras reconfigure in Fig. (6.16(d)) to cover the entire area, before the top right camera in Fig. (6.16(e)) decides to take a high resolution image of the third target as a new target enters the scene. A high resolution image of the fourth target is not taken as he is only briefly in the scene before his track merges with the track of another target.

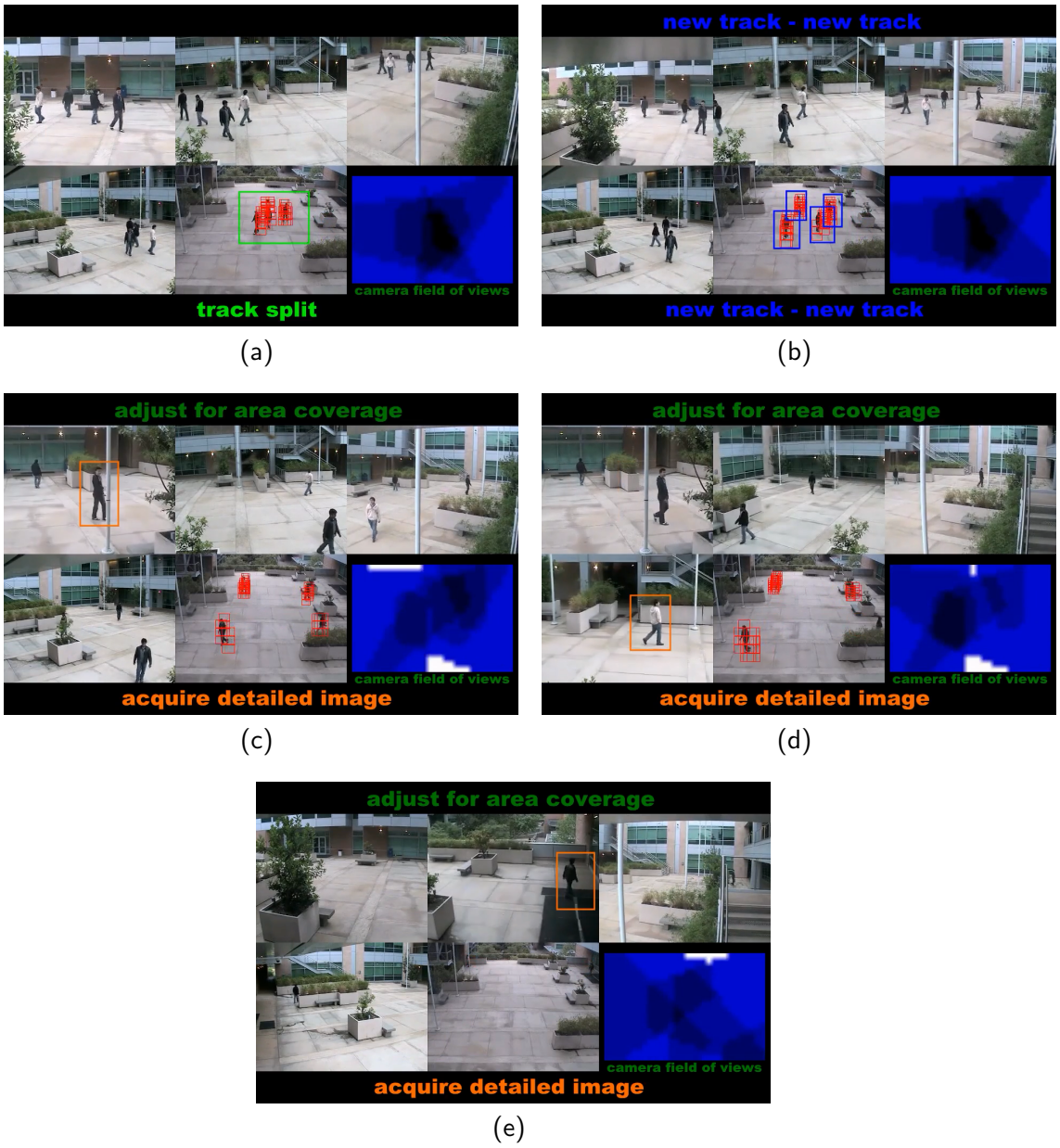


Figure 6.18: Shows the sequence of images taken in response to splitting tracks. (a) The targets are beginning to split apart. (b) A new track is created for each individual as there is enough separation to generate stable tracks. (c), (d) and (e) show high resolution images are acquired for 3 of the individual tracks. The 4th target left the surveillance area before the network could take a high resolution shot.

It is important to note that the third high resolution image is only taken after the other two zoomed in cameras have returned to covering a significant portion of the area. As more cameras zoom in for high resolution images, the amount of area covered by overlapping FOVs decreases. This means that the local value gained by the cameras

currently covering the area increases. Due to the tradeoff between the value gained from area coverage and high resolution imaging, the less cameras responsible for covering the area, the less likely a camera will zoom in for a detailed image.

To take into account occlusions, methods like those presented in [**?**], [86] and [49] can be incorporated with our system to make the decision making more robust. Also if we compare the area coverage in Fig. (6.15) and Fig. (6.17(b)) it can be seen that the area coverage solution is not unique and that the local maxima depends on the initial settings of the cameras.

As the paths of the people in the scene start intersecting shown in Fig. (6.17(a)) it becomes more difficult to segment the detections of the individuals and results in the merging of all four people into a new single track as in Fig. (6.17(b)). Since this is a large group covering a significant portion of the area, very little area is needed to be sacrificed to take a high resolution image of the entire group resulting in the image shown in Fig. (6.17(c)). This is because the camera is required to capture the entire probable region the group may occupy at the next time instant. During the time period where the four tracks are becoming merged into one, many short lived tracks are created as the detector begins to have difficult separating individual tracks. If the targets stay in a position where the detector cannot consistently separate or group them together, a long term track will not exist and no high resolution images will be acquired.

As the scene progresses, the people start to separate as can be seen in Fig. (6.18(a)). A short while after the split, distinct tracks for each person can again be formed in Fig. (6.18(b)). A sequence of high resolution images for each of the targets are then taken as shown in fig. (6.18c-e). What is noticeable in this scene is that a high resolution image of one of the targets was not squired before he exited the area after the group split. The reason for this was because the time span from when the individual

track was created until the time he left the area was too short for any camera to gain utility from capturing a high resolution image.

Chapter 7

Mobile Camera Platforms

In recent years, small unmanned vehicles, both terrestrial and aerial, have found application in tasks such as surveillance, search and rescue, mapping, and real-time monitoring. Determination of the world-frame coordinates of the tracked object is referred to as *geolocation*. The focus of this chapter is on the problem of optimally coordinating the motions of multiple vehicles, each equipped with pan-tilt-zoom (PTZ) cameras, for the purpose of tracking a target, with underlying stochastic dynamics, moving on the ground. It leads to the development of optimal routing and camera PTZ parameter control strategies that maximize the tracking performance (i.e., minimize the geolocation error). The combined optimization of routes and camera parameters, and the consideration of stochasticity in the target dynamics, sets this work apart from the research in both camera networks [72] and dynamic vehicle routing [8], as well as previous work on UAV coordination for optimal tracking [63].

The dynamics of each target is modeled using a stochastic difference equation. An Extended Kalman Filter [29] is used to track the target on the ground, using measurements obtained from video. Geolocation for video cameras is achieved by using the

pixel coordinates, intrinsic and extrinsic camera parameters, and the terrain data to estimate the location and associated error covariance of the target in the world frame. Using dynamic programming, the optimal coordinated control policies are computed, yielding the vehicle trajectories and camera PTZ settings, which minimize the expected fused geolocation error covariance. Results for both ground and air vehicles are shown, the optimal trajectories, PTZ settings and the geolocation error covariance are also reported.

7.1 Related Work

There has been a significant amount of work in coordinated target tracking. For two vehicles, maintaining a 90° angle of separation in relation to the target minimizes the fused geolocation error covariance as the camera-to-target line-of-sight vectors are orthogonal [24]. Much work, such as in [23] and [37], has been dedicated to designing controllers with angular separation as the goal. Methods to achieve diverse viewing angles have also been explored in [39] and [38]. The controller developed in [37] tasks aerial vehicles to orbit the target periodically while maintaining a fixed standoff distance. Each of these studies design control laws that should produce better geolocation estimates without explicitly considering the geolocalization. More recently, [63] and [**?**] have directly optimized the geolocation error covariance using online receding horizon controllers.

The geolocation error is highly sensitive to the relative position of the vehicle to the target, and the zoom capabilities of the video sensor. When the relative horizontal distance between the vehicle and target, with respect to the height, is large, the resulting error covariance is significantly elongated in the viewing direction. As this relative

horizontal distance shrinks, the geolocation error covariance tends towards circularity. Thus, purely from a camera-measurement-based state estimation perspective, the ideal path for the tracking vehicle would be to match the target's motion while directly overhead. However, the different dynamics of the tracking vehicles and the targets may preclude such positions from being maintained or even acquired. Additional vehicles working in concert may be able acquire measurements resulting in a fused geolocation error covariance of a circular nature.

The work here differs significantly from the work above in two important ways. First, the risk of failing to image a target is considered. For non-deterministic targets, there is a very real risk of failing to image the target if a significant portion of its probable locations is not covered by the FOV. Second, PTZ cameras are considered and their settings are *jointly* optimized with the vehicle routes. This leads to a optimal joint control policy for a long horizon using dynamic programming by minimizing the expected fused geolocation error covariance. The benefit of this approach is that long term paths can be evaluated and planned for multiple UAVs tracking a non-deterministic target while minimizing the risk of losing track it.

The rest of the chapter is organized as follows. First, the dynamics of the mobile platform are discussed. The proposed dynamic programming approach, including both the cost function and PTZ computations, are then described. The chapter concludes with a discussion on the simulation and the results.

7.2 Target and Vehicle Models

Consider mobile camera platforms (i.e., vehicles) tasked with tracking moving targets. Each of these vehicles moves at fixed forward speed while maintaining a con-

stant altitude. The target is located on the ground plane and has a non-deterministic trajectory. A PTZ camera is mounted on each of the vehicles and is used to acquire measurements of the target. The main objective is to optimize the motion of the vehicles and PTZ settings of the cameras with respect to the joint estimation error covariance across all vehicles for each target. It is assumed that each vehicle can communicate with other nearby vehicles or a base station to fuse acquired measurements. It is also assumed that the world frame location of each vehicle and the extrinsic camera-to-vehicle parameters are accurately known.

For target tracking in the world frame a dynamic model with a linearized discrete time state propagation as described earlier in Chap. (3). The video sensor mounted on each mobile platform acquires image plane measurements of all targets in its field of view (FOV). The two main coordinate frames that are used in video tracking are the world coordinate frame (also called the topographic frame), where the target and vehicle are located, and the sensor coordinate frame (i.e., pixels in image plane).

7.2.1 Camera Platform Model

A Dubins vehicle is a planar vehicle that has a fixed forward velocity with a bounded turning radius. This provides a simple model for a airborne mobile camera platforms with both a fixed altitude and velocity. The model neglects sideslip. For the i-th vehicle, let $\mathbf{p}_i^w = [x_i^w, y_i^w, z_i^w]^\top$ denote its position in the world frame, let $\mathbf{a}_i = [\rho_i, \tau_i, F_i]$ be the pan, tilt and focal length of the camera, and ψ_i denote its heading. The kinematics of the camera platform with fixed altitude z_i^w, velocity v_i and max

turning rate $w_{max} > 0$, are described by

$$\begin{aligned} \dot{x_i^w}(k) &= v_i cos(\psi_i(k)) \\ \dot{y_i^w}(k) &= v_i sin(\psi_i(k)) \\ \dot{\psi}_i(k) &= u_i(k), |u_i| \leq w_{max}. \end{aligned} \tag{7.1}$$

To discretize the Dubins vehicle dynamics, we apply a zero order hold (ZOH) on the control input at a sampling time of 1 second [63]. The discrete-time equivalent model for non-zero input is

$$\begin{aligned} x_i^{w+} &= \frac{v_i}{u_i}\left[sin(\psi_i + u_i) - sin(\psi_i)\right] + x_i \\ y_i^{w+} &= \frac{v_i}{u_i}\left[cos(\psi_i) - cos(\psi_i + u_i)\right] + y_i \\ {\psi_i}^+ &= u_i + \psi_i, \end{aligned} \tag{7.2}$$

and for zero input is

$$\begin{aligned} x_i^{w+} &= v_i cos(\psi_i) + x_i \\ y_i^{w+} &= v_i sin(\psi_i) + y_i \\ {\psi_i}^+ &= \psi_i. \end{aligned} \tag{7.3}$$

Because Dubin vehicles use only three inputs (left, straight, right), $u_i(k) \in U$ where $U = \{-w_{i,max}, 0, w_{i,max}\}$.

7.3 Optimal Vehicle Routes and Camera Settings

The technique of dynamic programming can be used to solve a wide array of applications described by dynamic equations-of-motion that require optimized paths. Dynamic programming is applied to produce an optimal control policy for a group of mobile cameras tracking a stochastic target. The utility function is presented first, followed by the computation of the PTZ settings. Naive implementation of dynamic programming results in an exponential time algorithm. By considering relative vehicle-to-target coordinates instead of absolute coordinates, we reduce the total amount of computation.

7.3.1 Utility Function

The utility function is designed to optimize estimation performance over a planning horizon. Performance is quantified as a function of the expected fused geolocation information

$$E\left\langle \mathbf{J} \right\rangle = \sum_{i=1^{N_c}} \mathbf{J}_i Pr\{{}^w\mathbf{p} \in FOV_i\} \tag{7.4}$$

$$= \sum_i \mathbf{J}_i \int_{FOV_i} p_{\mathbf{p}}\left(\zeta\right) d\zeta. \tag{7.5}$$

The dummy variable ζ is integrated over the ground plane and $p_{\mathbf{p}}$ is the Normal distribution $\mathcal{N}({}^w\hat{\mathbf{p}}, \mathbf{P}^-_{\mathbf{pp}})$ of the predicted position of the target in the global frame at time t_k.

For optimization, the function $g\left(E\left\langle \mathbf{J} \right\rangle\right) : \mathbf{S}^n_{++} \mapsto \Re^+$ is chosen. Ideally, g is a convex function. For this work, g is chosen to be the trace. The trace is easily computed and linear, but has the deficiency that it can be increased by increasing one diagonal

element while leaving another component near zero, yielding an elongated estimation error ellipsoid.

The value function is

$$V(\mathbf{z}(k)) = trace(E\langle \mathbf{J_{pp}} \rangle), \tag{7.6}$$

where $\mathbf{z}(k)$ is concatenate vector of target camera platform state vectors. The resulting utility function is evaluated over a planning horizon of N_s time steps according to

$$U_0(\mathbf{z}) = \frac{1}{N_s} \sum_{k=0}^{N_s-1} V(\mathbf{z}(k)). \tag{7.7}$$

Backwards induction is used in dynamic programming to find the optimal control policy $\prod_k(\mathbf{z})$, which maps each state $\mathbf{z}(k)$ to an optimal control input for each vehicle at time t_k. Let N_c be the number of camera platforms. The optimal value is determined through the standard technique of value iteration where the optimal utility for Eqn. (7.7) is computed using the recursive function

$$U_k(\mathbf{z}) = \max_{\mathbf{u} \in U^{N_c}} (V(\mathbf{z}(k)) + U_{k+1}(f(\mathbf{z}, \mathbf{u}))), \forall \mathbf{z}, \tag{7.8}$$

from $k = N_s - 1$ to 0 with $V_{N_s}(\mathbf{z}) = 0, \forall \mathbf{z}$. The input $\mathbf{u}$ corresponding to the maximum value in Eqn. (7.8) is stored in $\prod_k(\mathbf{z})$. The state propagation model $f(\mathbf{z}, \mathbf{u})$ returns the state $\mathbf{z}(k+1)$.

The optimal PTZ settings $\mathbf{a}_i(k+1)$ of the video cameras are dependent on the expected state of the target $\hat{\mathbf{x}}(k+1)$ and the corresponding covariance $\mathbf{P}(k+1)$. The risk of failing to acquire an image of the target, is enhanced by ensuring that the expected position of the target and its associated error covariance ellipse are within

Algorithm 3 $U_k(\mathbf{z})$

for each the i-th camera platform **do**
  Compute $\mathbf{a}_i = [\rho_i, \tau_i, F_i]$.
    First $[\rho_i, \tau_i]$ are obtained by centering the view on ${}^w\hat{\mathbf{p}}(k)$.
    Then map $\mathbf{P_{pp}}(k)$ to the image plane and compute F_i of the minimum bounding view.
end for
Propagate ${}^w\hat{\mathbf{x}}(k)$ and $\mathbf{P_{pp}}(k)$ to ${}^w\hat{\mathbf{x}}(k+1)$ and $\mathbf{P_{pp}}(k+1)$
for each $\mathbf{u} \in U^{N_c}$ **do**
  Compute $U_{k+1}(\mathbf{z})$ and save if max
end for
return V($\mathbf{z}$(k)) + max $U_{k+1}(\mathbf{z})$

the FOV of the camera. The amount of information obtained when imaging a target is directly proportional to the zoom of the camera. To reduce computation, instead of searching over the PTZ parameters during the optimization phase, we solve for the minimum bounding FOV of the target's uncertainty ellipse. The steps involved in this process are shown in Algorithm 3.

7.4 Simulation

To demonstrate the effectiveness of this approach a simulation framework consisting of a $500m$ x $500m$ area was prepared. In this area two vehicles were considered with the objective of tracking a non-deterministic target. The target was initialized with a starting velocity of 5m/s along the x-axis and was propagated through time according to Eqn. (3.3). The parameters used for this dynamical model are as follows:

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} 33 & 0 & 50 & 0 \\ 0 & 33 & 0 & 50 \\ 50 & 0 & 100 & 0 \\ 0 & 50 & 0 & 100 \end{bmatrix}.$$

The target track was propagated for 120 seconds. We considered two scenarios: one for land-based camera platforms, and the other for aerial platforms. For the scenario considering land vehicles the altitude of the camera was set to $z_i^w = 2m$ and for the scenario with aerial vehicles, $z_i^w = 100m$. In both cases the vehicles were initialized with a velocity $v_i = 15m/s$ along the x-axis. Planning for N_s seconds ahead is done at every time instant by propagating the estimated target state, and estimated error covariance from $\hat{\mathbf{x}}(k)$ and $\hat{\mathbf{P}}(k)$ through to $\hat{\mathbf{x}}(k + N_s)$ and $\hat{\mathbf{P}}(k + N_s)$. Planning for very long horizons is undesirable since the error in the expected location of a non-deterministic target increases as the target is propagated through time.

7.4.1 Terrestrial Vehicle

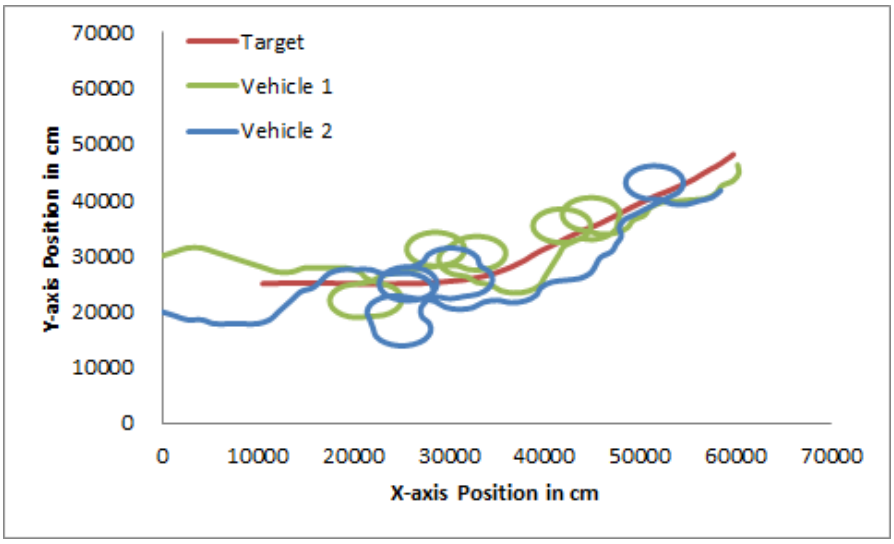
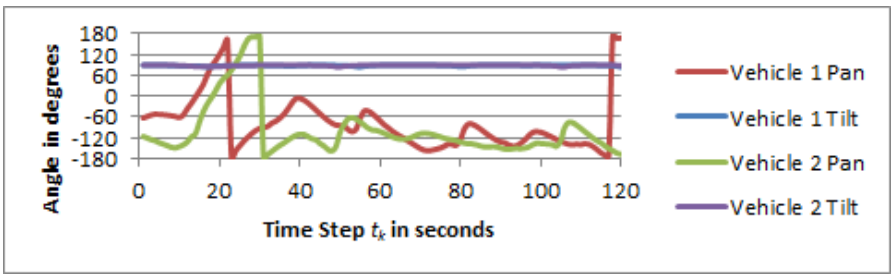
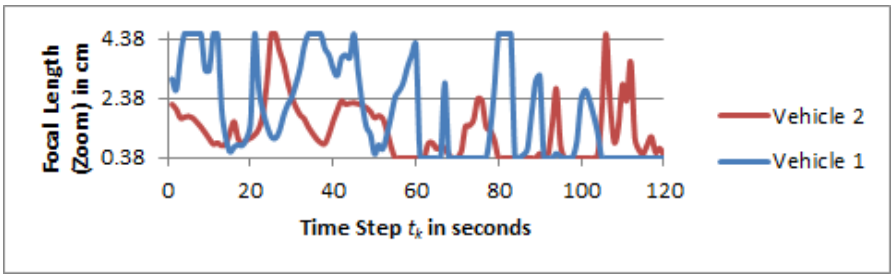


Figure 7.1: Trajectories for two ground vehicles tracking a target with $N_s = 1$ planning steps.

The first scenario considered was target state estimation over a 120 second time window by two land based or low flying camera platforms. Fig. **??** shows the final



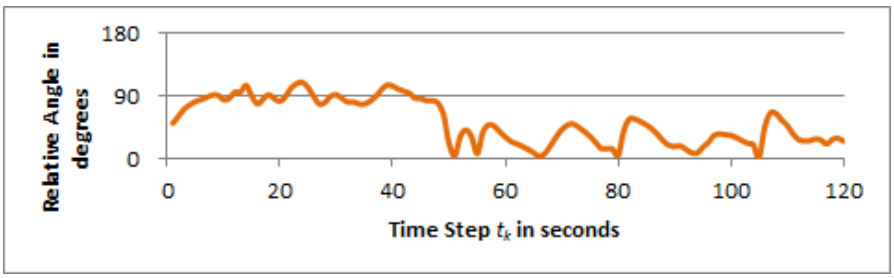
(a)



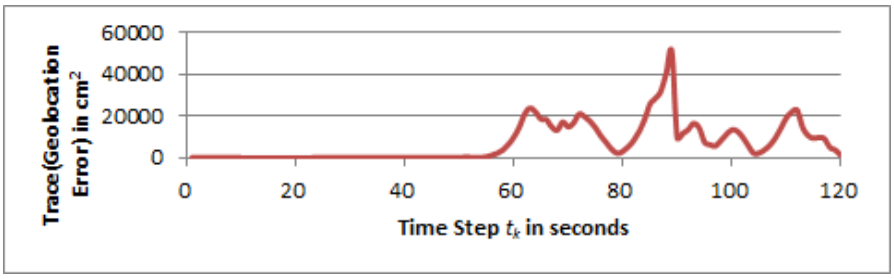
(b)

Figure 7.2: Resulting PTZ plots from two ground vehicles tracking a target with $N_s = 1$ planning steps. Fig. (a) shows the change in pan and tilt while Fig. (b) shows the change in zoom during the scenario. This change depends on the distance between the vehicle and the target as well as the amount of geolocation error in the target position, e.g. a larger error requires the vehicle to observe a larger region to guarantee detection of the target.

paths taken by the two land vehicles. The nonexistent planning horizon means that each vehicle can and will take the best move available to it even if it results in a terrible set of choices at the next time step. This can also be seen at $t = 50$ where the geolocation error covariance is reduced by moving closer to the target at the expense of maintaining orthogonal error covariance ellipses. Because the video sensor is located only 2m above the ground, the resulting geolocation error covariance for a single measurement will be far from circular. In order to maintain an accurate target geolocation estimate, a second measurement is required, preferably with an orthogonal geolocation error covariance. This can be clearly seen in Fig. 7.3(a) and 7.3(b) for $t > 50$, where the covariance is significantly increased when the angle between the two vehicles, relative to the target, strays away from 90°.



(a)



(b)

Figure 7.3: Resulting plots from two ground vehicles tracking a target with $N_s = 1$ planning steps. The explosion in tracking error from lack of planning can be demonstrated in Fig. (b). From Fig. (a) the relative angle between the two vehicles strays from the ideal 90 degrees when there is a large uncertainty in the target position.

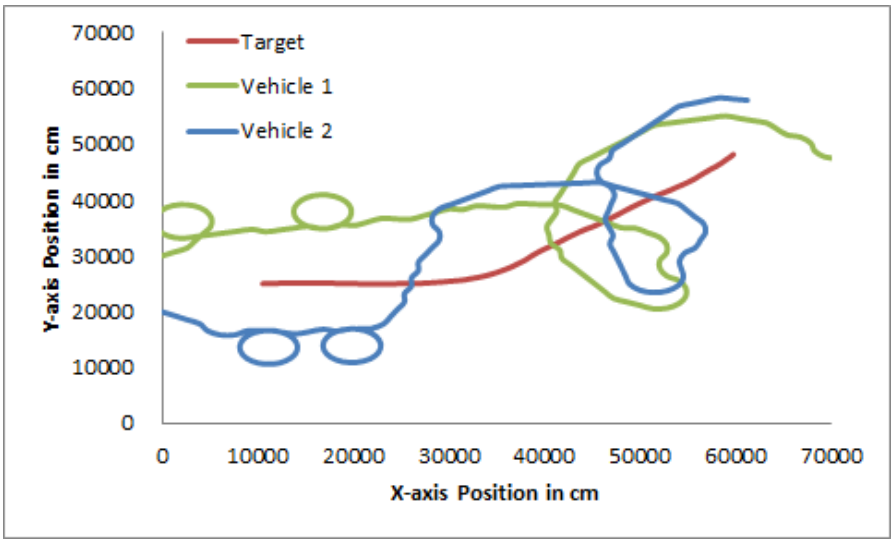
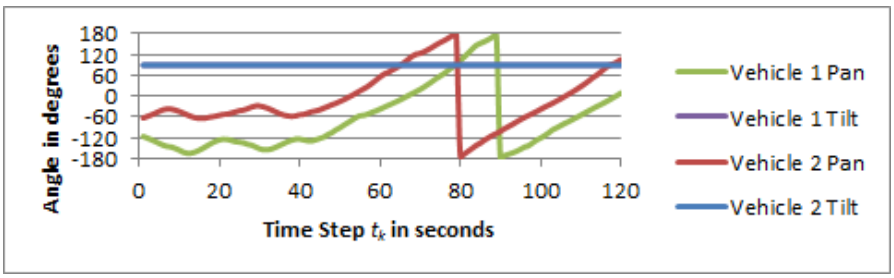
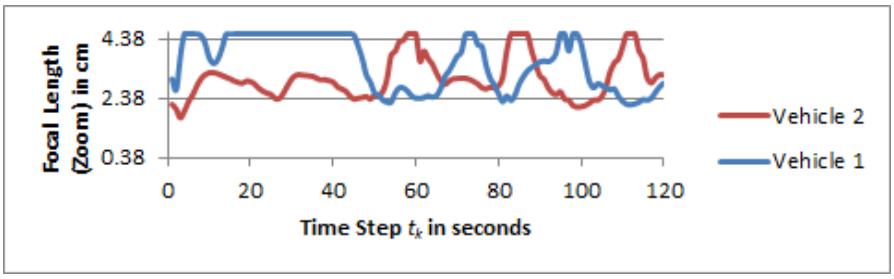


Figure 7.4: Trajectories and results for 2 ground vehicles tracking a target with $N_s = 3$ planning steps.

Preplanning the trajectories for the entire 120 second duration is not only unreasonable for a stochastic target, it is also prohibitive in terms of computation. However, no planning, as we have just shown, can lead to situations causing large



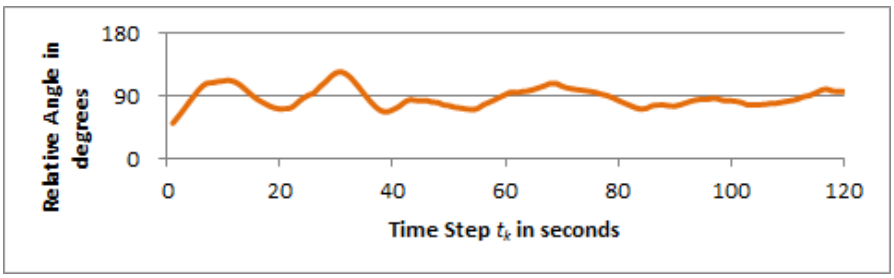
(a)



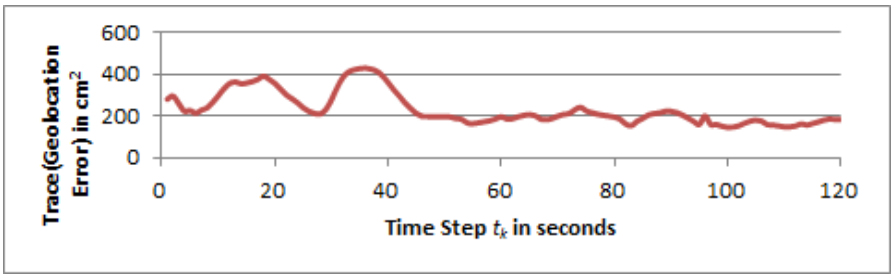
(b)

Figure 7.5: Resulting PTZ plots for 2 ground vehicles tracking a target with $N_s = 3$ planning steps. Fig. (a) shows the change in pan and tilt while Fig. (b) shows the change in zoom during the scenario. Both vehicles maintain a relatively high zoom compared with the results from the $N_s = 1$ planning results. The change in zoom here can mostly be attributed to the distance between vehicle and target since the error covariance is almost constant.

fluctuations in tracking accuracy. In the worst case the target track may be lost as the actual and expected vehicle positions diverge and actual position fails to be in the camera FOVs. The number of steps to plan ahead is dependent on the desired tracking accuracy and the amount of randomness in the vehicle motions, which is quantified by the process noise covariance matrix $\mathbf{Q}$. Fig. 7.4 shows the results of planning $N_s = 3$ steps ahead, for the same target trajectory. The computational cost of such a plan length is minimal and may be easily computed every second. From the trajectories exhibited in Fig. **??**, we can observe that the vehicles maintain a much further distance from the target than in the previous case. The relative angles between the two vehicles, shown in Fig. 7.6(a), also stay close to the 90° separation required for minimal instantaneous geolocation error covariance. The propagation of the expected target position and error covariance for



(a)



(b)

Figure 7.6: Resulting plots from two ground vehicles tracking a target with $N_s = 3$ planning steps. Fig. (b) shows the tracking error of the target. Fig. (a) shows the relative angle between the two vehicles. Notice that the geolocation error decreases as the relative angle between the two vehicles stabilizes around 90 degrees.

the duration of the plan results in paths that ensure optimal measurements of targets that are governed by our model. It is also clear from Fig. 7.6(b) that the geolocation error covariance of the tracker is very smooth and not subject to the large fluctuations of the previous scenario.

7.4.2 Aerial Vehicle

The second scenario considered is the target state estimation over a 120 second time window by two aerial camera platforms. The aerial vehicles are set to fly at a fixed height of 100m. Since the video sensor is located at a height much greater than that of the ground vehicles, the resulting geolocation error covariance will tend more toward

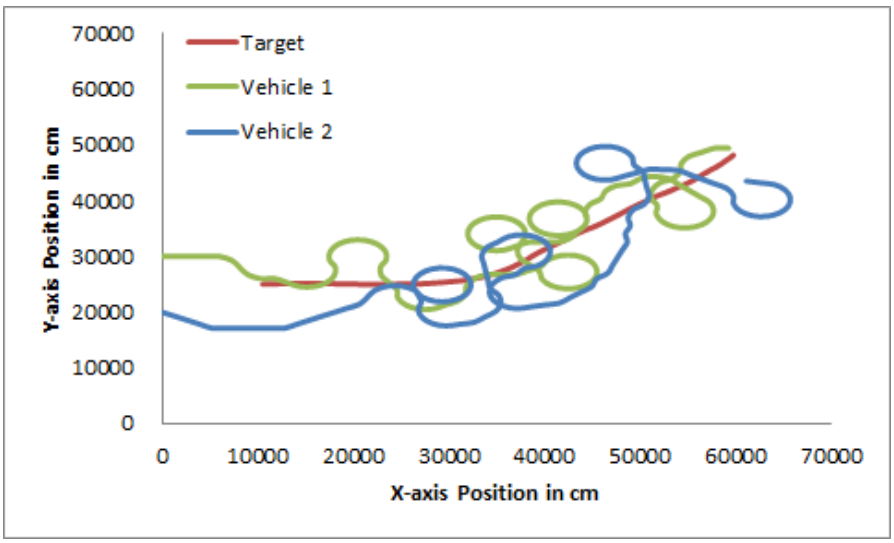
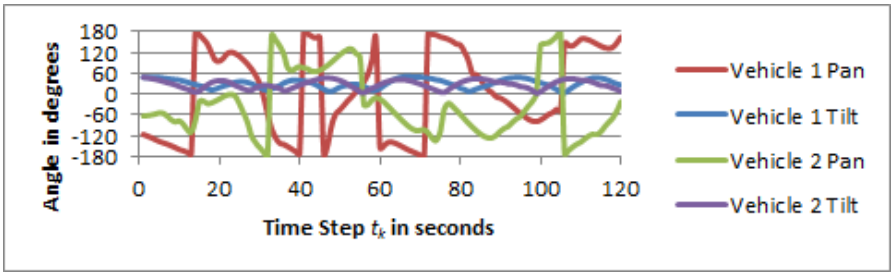
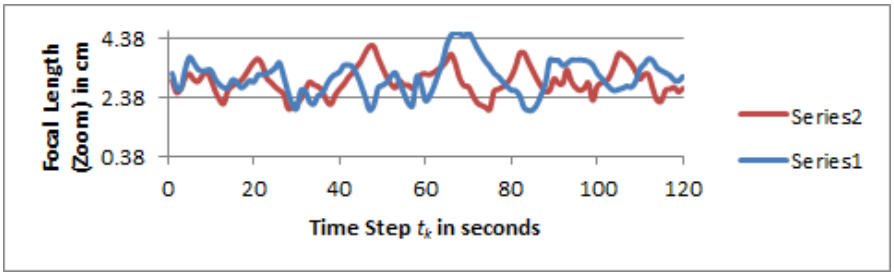


Figure 7.7: Trajectories and results for 2 aerial vehicles tracking a target with $N_s = 1$ planning steps.



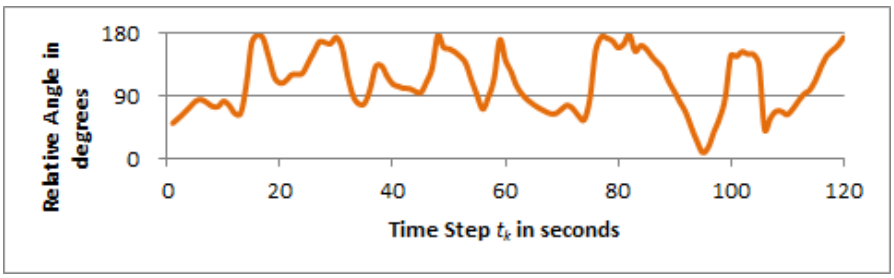
(a)



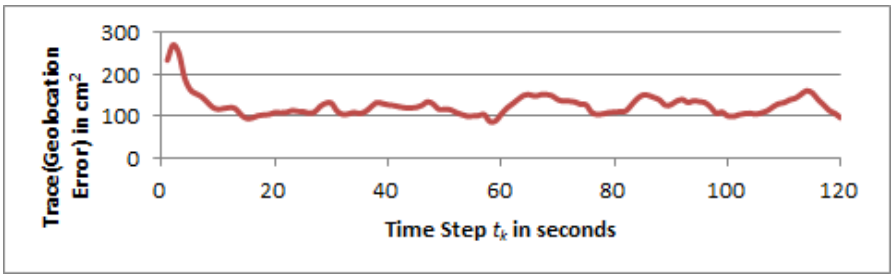
(b)

Figure 7.8: Resulting PTZ plots for 2 aerial vehicles tracking a target with $N_s = 1$ planning steps. Fig. (a) shows the change in pan and tilt while Fig. (b) shows the change in zoom during the scenario. Both vehicles maintain a relatively high zoom. The change in zoom here can mostly be attributed to the distance between vehicle and target since the error covariance is almost constant.

circularity than it did for the terrestrial vehicle. While a second measurement, with an orthogonal geolocation error covariance, is still preferred, it is no longer as necessary



(a)



(b)

Figure 7.9: Resulting plots from two ground vehicles tracking a target with $N_s = 3$ planning steps. Fig. (b) shows the tracking error of the target. Fig. (a) shows the relative angle between the two vehicles. Notice that the geolocation error is mostly stable even as the relative angle between the two vehicles changes dramatically. This is mostly due to the fact that as the vehicle is at a much higher altitude, the measurement error is shaped less like an ellipse and more circular.

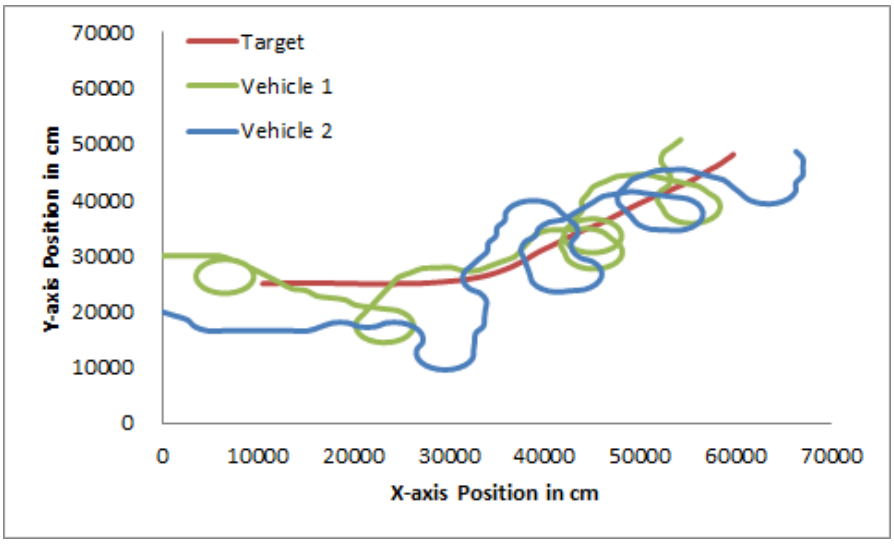
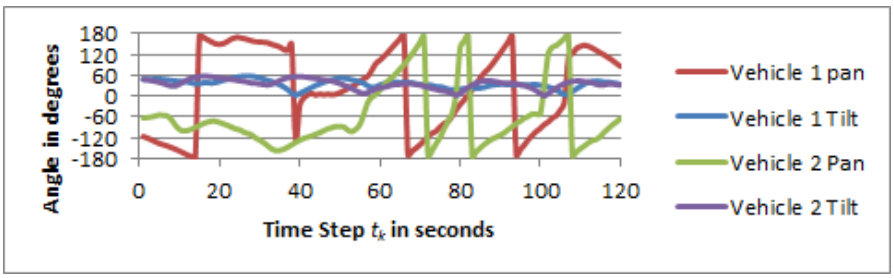
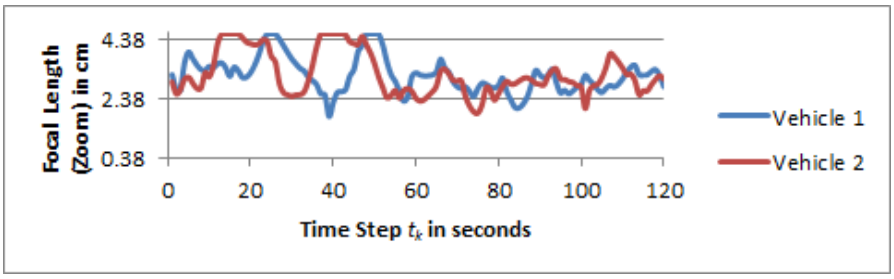


Figure 7.10: Trajectories and results for 2 aerial vehicles tracking a target with $N_s = 3$ planning steps.

to maintaining good target geolocation estimates. This is supported by the results for both the $N_s = 1$ and $N_s = 3$ plans.



(a)

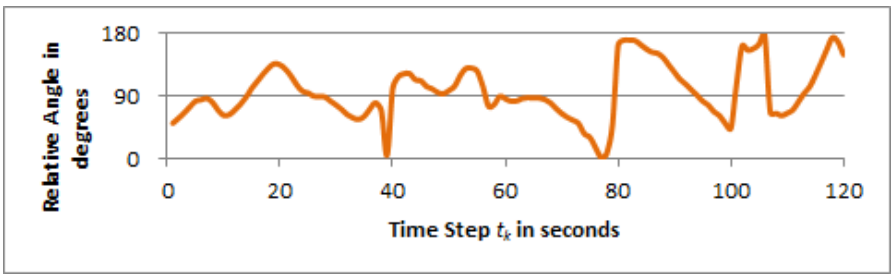


(b)

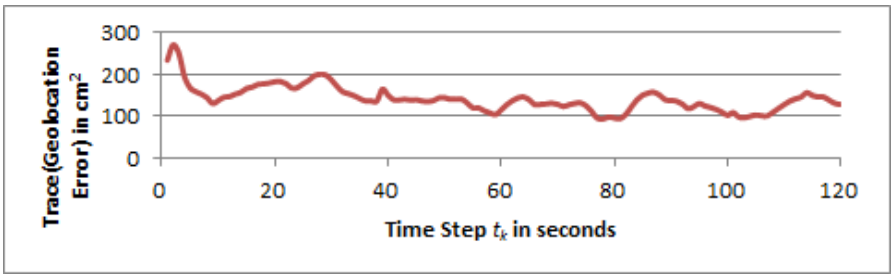
Figure 7.11: Resulting PTZ plots for 2 aerial vehicles tracking a target with $N_s = 3$ planning steps. Fig. (a) shows the change in pan and tilt while Fig. (b) shows the change in zoom during the scenario. Both vehicles maintain a relatively high zoom. The change in zoom here can mostly be attributed to the distance between vehicle and target since the error covariance is almost constant.

The paths traversed by the aerial vehicles for $N_s = 1$ are displayed in Fig. 7.7. It is immediately apparent that the vehicles have paths that follow the target more tightly that the planning scenario for ground vehicles. This is mostly because the the aerial vehicles can get good measurements without needing to maintain a particular angular separation due to the mostly circular measurement error covariances as can be observed in Fig. 7.9(a) and 7.9(b). The zoom value of the vehicles shown in Fig. (7.8(b)) also rarely hits the maximum zoom values since the vehicles are close enough to the target that all the probable locations of the target occupy the image without needing to zoom all the way in.

The paths traversed by the aerial vehicles for $N_s = 3$ displayed in Fig. 7.10 are very similar to the paths of the vehicle planning for $N_s = 1$. Unlike the ground vehicles, planning ahead for multiple time steps does not bring much benefit since it is hard for



(a)



(b)

Figure 7.12: Resulting plots from two ground vehicles tracking a target with $N_s = 3$ planning steps. Fig. (b) shows the tracking error of the target. Fig. (a) shows the relative angle between the two vehicles. Notice that the geolocation error is mostly stable even as the relative angle between the two vehicles changes dramatically. This is mostly due to the fact that as the vehicle is at a much higher altitude, the measurement error is shaped less like an ellipse and more circular. There is a slight increase in the error covariance since the vehicle has to consider the locations of the target for multiple future time instants, thus maintaining a further distance from the target.

the vehicles to get into a situation where the target position cannot be estimated well. Looking at the error covariance and the angular separation between the vehicles shown in Fig. 7.12(a) and 7.12(b), it is clear that the angular separation has little effect on the error covariance. The zoom value of the vehicles shown in Fig. (7.11(b)) are also similar to the $N_s = 1$. Since the target error covariance does not become large the vehicles never zoom all the way out to be able to cover all the probable locations.

Chapter 8

Conclusions

This dissertation discussed an approach to optimize various scene analysis performance criteria through distributed control of a dynamic camera network. An Extended Kalman-Consensus filtering approach was used to track multiple targets in a dynamic network. A camera control framework using a distributed optimization approach allowed selection of parameters to acquire images that best optimized the user specified requirements. The uncertainty in state estimation when deciding upon camera settings, and the effect of different utility function weights was also considered. The effectiveness of the system was also demonstrated by showing results from real life implementations on a camera network in addition to analysis of results from simulations.

A method to prioritize tasks for a distributed camera network to co-operatively track all targets and procure high resolution images, when the opportunity arises, subject to target pose and other criteria was also discussed. Utility functions were designed to evaluate the PTZ settings of the cameras for both tracking and feature imaging. These were used within a Bayesian distributed optimization framework to select optimal PTZ settings for the camera network at every time instant. The results also showed how

utilities could be designed to reduced the resources required to enable high quality feature acquisition. This could enable the camera network to satisfy multiple performance requirements that cannot be achieved using a static camera network.

A key contribution of this dissertation is showing how the parameter selection can be distributed, given knowledge of the vision graph, in a camera network so that multiple cameras can change parameters simultaneously. This allows for easy scalability and increases the number potential applications. Prior approaches allowed for only one camera to move at a time while this approach allows multiple cameras to move simultaneously as the network expands.

Using a wireless testbed of Axis PTZ cameras the behavior of the framework was explored under multiple scenarios in real time. The experiments performed on the real life network showed that it was possible to improve tracking accuracy and feature acquisition through the dynamic control of PTZ camera networks. This work can lay the foundations for autonomous systems with embedded video processors that are capable of opportunistic sensing, efficient navigation and scene analysis.

8.1 Future Work

Future research in the dynamic control of camera networks is in developing autonomous systems consisting of fixed and mobile cameras with embedded intelligence capable of video acquisition and analysis over wide areas. This is a highly interdisciplinary area that can bring together researchers in computer vision, control theory, machine learning and various branches of mathematics and statistics. The vision is to develop an intelligent network of mobile agents, each equipped with visual sensors and the capability of analyzing its own data and taking decisions in coordination with other

agents. This would enable the agents to maneuver themselves optimally so as to obtain images that can be analyzed with a high degree of reliability. This will require that the system has the ability to jointly optimize camera locations and sensing parameters to satisfy a scene understanding objective. Some initial steps taken in this direction were described in this dissertation.

This research direction falls within the broad domain of multi-agent systems, which are systems of interacting intelligent agents where each individual is able to sense only a part of the system and communicate with a time-varying set of neighbors. Thus coordination is central to multi-agent systems. Broadly speaking, coordination refers to an agent being aware of other agents in its environment be it for resource allocation, task allocation, communication, or movement. When the agents have the capability of moving, coordination must be reflected in both task allocation and motion control, and one primary research topic is the Vehicle Routing Problem (VRP). Equipping these vehicles with cameras would enable tasks in which visual analysis is key, e.g., visual tracking and monitoring, object and event recognition. This leads to the Mobile Camera Networks (MCN) problem, which has some fundamental differences with the VRP problem. While video analysis will not be the focus of research in this direction, the algorithms would obviously have to be developed with an awareness of the capabilities on the analysis side. Distributed solutions to many of the basic video analysis tasks will be key to these systems. The vision of such multi-agent systems equipped with cameras will also require development of suitable hardware and software platforms that will be able to satisfy the computation and communication requirements. Networking and communication constraints, as well as power resources, will be critical issues in many application domains.

In summary, camera networks is an upcoming area of research with a highly interdisciplinary flavor and a promise for addressing problems in application domains where there is a critical need for such technology, e.g., disaster response, wide area surveillance, assisted living. While many of the building blocks, especially in relation to video analysis, are in place, the network-level solutions, like distributed estimation, resource constraints or integration of sensing and analysis tasks, are very much in their infancy. The area holds a lot of promise for a number of reasons - basic research problems that span multiple disciplines, interdisciplinary research problems that bring together computer science, electrical engineering and mathematics, and technology development that will transition the research into critically necessary application domains.

Bibliography

[1] A. A. Morye and C. Ding and B. Song and A. K. Roy-Chowdhury and J. A. Farrell. Optimized Imaging and Target Tracking within a Distributed Camera Network. In *American Control Conference*, 2011.

[2] A. Alahi, D. Marimon, M. Bierlaire, and M. Kunt. A master-slave approach for object detection and matching with fixed and mobile cameras. In *Intl. Conf. on Image Processing*, 2008.

[3] G. Arslan, J. Marden, and J. Shamma. Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation. *ASME Journal of Dynamic Systems, Measurement and Control*, 129(5), September 2007.

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

[5] Y. Bar-Shalom, T. Fortmann, and M. Scheffe. Joint Probabilistic Data Assocation for Multiple Targets in Clutter. *Information Sciences and Systmes*, 1980.

[6] A. Blake and A. Yuille, editors. *Active Vision*. MIT Press, 1992.

[7] Philip A. Bland, Pavel Spurn, Martin C. Towner, Alex W. R. Bevan, Andrew T. Singleton, William F. Bottke, Richard C. Greenwood, Steven R. Chesley, Lukas Shrben, Jiri Borovika, Zdenek Ceplecha, Terence P. McClafferty, David Vaughan, Gretchen K. Benedix, Geoff Deacon, Kieren T. Howard, Ian A. Franchi, and Robert M. Hough. An anomalous basaltic meteorite from the innermost main belt. *Science*, 325(5947):1525–1527, 2009.

[8] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011.

[9] Chaw-Bing Chang and J. Tabaczynski. Application of state estimation to target tracking. *Automatic Control, IEEE Transactions on*, 29(2):98–109, 1984.

[10] Z. Cheng, D. Devarajan, and R. Radke. Determining vision graphs for distributed camera networks using feature digests. *EURASIP Journal on Advances in Signal Processing: Special Issue on Visual Sensor Networks*, 2007.

[11] D. Cook and S. Das. *Smart Environments: Technology, Protocols and Applications*. Wiley-Interscience, 2005.

[12] J. W. Davis. Camera Control and Geo-Registration for Video. In *Distributed Video Sensor Networks*. Springer, 2011.

[13] G. Denina, B. Bhanu, H. Nguyen, C. Ding, A. Kamal, C. Ravishankar, A. Roy-Chowdhury, A. Ivers, , and B. Varda. VideoWeb Dataset for Multi-camera Activities and Non-verbal Communication. In *Distributed Video Sensor Networks*. Springer, 2011.

[14] D. Devarajan, Z. Cheng, and R.J. Radke. Calibrating Distributed Camera Networks. *Proceedings of the IEEE Special Issure on Distributed Smart Cameras*, 96(10):1625–1639, October 2008.

[15] Dieber, B. and Micheloni, C. and Rinner, B. Resource-Aware Coverage and Task Assignment in Visual Sensor Networks. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(10):1424–1437, 2011.

[16] Chong Ding, A. A. Morye, J. A. Farrell, and A. K. Roy-Chowdhury. Opportunistic Sensing in a Distributed PTZ Camera Network. In *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, pages 1–6, 2012.

[17] Chong Ding, Bi Song, A. A. Morye, J. A. Farrell, and A. K. Roy-Chowdhury. Collaborative Sensing in a Distributed PTZ Camera Network. *Image Processing, IEEE Transactions on*, 21(7):3282 –3295, july 2012.

[18] Ali O. Ercan, Danny B. Yang, Abbas El Gamal, and Leonidas J. Guibas. Optimal placement and selection of camera network nodes for target localization. In *Proceedings of the Second IEEE international conference on Distributed Computing in Sensor Systems*, DCOSS'06, pages 389–404, Berlin, Heidelberg, 2006. Springer-Verlag.

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

[20] E. Ermis, V. Saligrama, P. Jodoin, and J. Konrad. Abnormal behavior detection and behavior matching for networked cameras. In *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, 2008.

[21] R. Farrell and L. S. Davis. Decentralized discovery of camera network topology. In *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, 2008.

[22] R. Farrell, D. Doermann, and L. S. Davis. Learning higher-order transition models in medium-scale camera networks. In *IEEE Workshop on Omnidirectional Vision*, 2007.

[23] E. W. Frew. Sensitivity of cooperative geolocalization to orbit coordination. In *AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, Aug. 2007.

[24] G. Gu, R.R. Chandler, C.J. Schumacher, A. Sparks, and M. Pachter. Optimum cooperative UAV sensing based on Cramer-Rao bound. In *American Control Conference*, June 2005.

[25] T. Hatanaka and M. Fujita. Cooperative estimation of averaged 3-d moving target poses via networked visual motion observer. *Automatic Control, IEEE Transactions on*, 58(3):623–638, 2013.

[26] E. Hörster and R. Lienhart. On the optimal placement of multiple visual sensors. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, VSSN '06, pages 111–120, New York, NY, USA, 2006. ACM.

[27] T. Huang and S. Russel. Object Identification In A Bayesian Context. In *Proceeding of IJCAI*, 1997.

[28] V. Isler and R. Bajcsy. The sensor selection problem for bounded uncertainty sensing models. *Automation Science and Engineering, IEEE Transactions on*, 3(4):372–381, 2006.

[29] J.A. Farrell. Aided Navigation: GPS with High Rate Sensors. pages 72–75, 2008.

[30] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Automatic Control*, 48(6):988–1001, Jun 2003.

[31] O. Javed, S. Khan, Z. Rasheed, and M. Shah. Camera Handoff: Tracking in Multiple Uncalibrated Stationary Cameras. In *IEEE Workshop on Human Motion*, pages 113–118, Los Alamitos, CA, USA, Dec. 2000.

[32] Omar Javed, Zeeshan Rasheed, Khurram Shafique, and Mubarak Shah. Tracking across multiple cameras with disjoint views. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 952–, Washington, DC, USA, 2003. IEEE Computer Society.

[33] H. Jiang, S. Fels, and J. Little. A Linear Programming Approach for Multiple Object Tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[34] A. T. Kamal, C. Ding, B. Song, J. A. Farrell, and A.K. Roy-Chowdhury. A generalized kalman consensus filter for wide-area video networks. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 7863–7869, 2011.

[35] J. Kang, I. Cohen, and G. Medioni. Continuous Tracking Within and Across Camera Streams. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

[36] V. Kettnaker and R. Zabih. Bayesian Multi-Camera Surveillance. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 1999.

[37] D. B. Kingston. *Decentralized control of multiple UAVs for perimeter and target surveillance*. PhD thesis, Brigham Young University, Provo, Utah, Dec 2007.

[38] D.J. Klein and K.A. Morgansen. Controlled collective motion for trajectory tracking. In *American Control Conference*, 2006.

[39] E. Lalish, K.A. Morgansen, and T. Tsukamaki. Oscillatory control for constant-speed unicycle-type vehicles. In *Decision and Control, 2007 46th IEEE Conference on*, pages 5246–5251, 2007.

[40] B. Leibe, K. Schindler, and L. V. Gool. Coupled Detection and Trajectory Estimation for Multi-Object Tracking. In *IEEE Intl. Conf. on Computer Vision*, 2007.

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

[42] N. Li and J. Marden. Designing games for distributed optimization. In *IEEE Conf. on Decision and Control*, Florida, USA, Dec. 2011.

[43] Yiming Li and B. Bhanu. A comparison of techniques for camera selection and handoff in a video network. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–8, 2009.

[44] Z. Mandel, I. Shimshoni, and D. Keren. Multicamera topology recovery from coherent motion. In *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, 2007.

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

[46] D. Markis, T. Ellis, and J Black. Bridging the Gap Between Cameras. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

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

[48] C. Micheloni, B. Rinner, and G. L. Foresti. Video Analysis in Pan-Tilt-Zoom Camera Networks. *IEEE Signal Processing Magazine*, 27(5):78–90, September 2010.

[49] A. Mittal and L.S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *International Journal of Computer Vision*, 76:31–52, 2008.

[50] Anurag Mittal and Larry S. Davis. Visibility analysis and sensor planning in dynamic environments. In *European Conference on Computer Vision*, 2004.

[51] Anurag Mittal and Larry S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *Int. J. Comput. Vision*, 76(1):31–52, January 2008.

[52] D. Monderer and L.S. Shapley. Potential games. *Games and Economic Behavior*, (1), May 1996.

[53] M.D. Naish, E.A. Croft, and B. Benhabib. Simulation-based sensing-system configuration for dynamic dispatching. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 5, pages 2964–2969 vol.5, 2001.

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

[55] Michael J. Ocean, Azer Bestavros, and Assaf J. Kfoury. Snbench: Programming and virtualization framework for distributed multitasking sensor networks, 2006.

[56] Gustavo Olague and Roger Mohr. Optimal Camera Placement for Accurate Reconstruction. *Pattern Recognition*, 35:927–944, April 2002.

[57] R. Olfati-Saber. Kalman-consensus filter : Optimality, stability, and performance. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 7036–7042, 2009.

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

[59] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Automatic Control*, 49(9):1520–1533, Sep 2004.

[60] R. Olfati-Saber and N. F. Sandell. Distributed tracking in sensor networks with limited sensing range. In *Proceedings of the American Control Conference*, pages 3157 – 3162, Seattle, WA, USA, June 2008.

[61] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. Multi-Object Tracking Through Simultaneous Long Occlusions and Split-Merge Conditions. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2006.

[62] C. Piciarelli, C. Micheloni, and G.L. Foresti. PTZ camera network reconfiguration. In *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, pages 1–8, Como, Italy, Aug. 2009.

[63] S. Quintero, F. Papi, D. Klein, L. Chisci, and J. Hespanha. Optimal UAV coordination for target tracking using dynamic programming. In *Proc. of the 49th Conf. on Decision and Contr.*, Dec. 2010.

[64] F.Z. Qureshi and D. Terzopoulos. Surveillance in Virtual Reality: System Design and Multi-Camera Control. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[65] F.Z. Qureshi and D. Terzopoulos. Planning Ahead for PTZ Camera Assignment and Handoff. In *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, pages 1–8, Como, Italy, Aug-Sep 2009.

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

[67] D. Reid. An Algorithm for Tracking Multiple Targets. *IEEE Trans. Automatic Control*, 24(6):843–854, 1979.

[68] Amit K. Roy-Chowdhury and Bi Song. *Camera Networks: The Acquisition and Analysis of Videos over Wide Areas*. Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers, 2012.

[69] A.C. Sankaranarayanan, A. Veeraraghavan, and R. Chellappa. Object Detection, Tracking and Recognition for Multiple Smart Cameras. *Proceedings of the IEEE Special Issure on Distributed Smart Cameras*, 96(10):1606–1624, October 2008.

[70] B. Song and A. Roy-Chowdhury. Stochastic Adaptive Tracking in a Camera Network. In *IEEE Intl. Conf. on Computer Vision*, 2007.

[71] B. Song and A. Roy-Chowdhury. Robust Tracking in A Camera Network: A Multi-Objective Optimization Framework. *IEEE Journal on Selected Topics in Signal Processing: Special Issue on Distributed Processing in Vision Networks*, 2008.

[72] Bi Song, Chong Ding, A.T. Kamal, J.A. Farrell, and A.K. Roy-chowdhury. Distributed camera networks. *Signal Processing Magazine, IEEE*, 28(3):20 –31, may 2011.

[73] Bi Song, A.T. Kamal, C. Soto, Chong Ding, J.A. Farrell, and A.K. Roy-Chowdhury. Tracking and activity recognition through consensus in distributed camera networks. *Image Processing, IEEE Transactions on*, 19(10):2564–2579, 2010.

[74] S. Soro and W. Heinzelman. Camera selection in visual sensor networks. In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 81–86, 2007.

[75] C. Soto, B. Song, and A. K. Roy-Chowdhury. Distributed Multi-Target Tracking In A Self-Configuring Camera Network. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

[76] B. Stancil, C. Zhang, and T. Chen. Active Multicamera Networks: From Rendering to Surveillance. *IEEE Journal on Selected Topics in Signal Processing Special Issue on Distributed Processing in Vision Networks*, August 2008.

[77] M. Taj and A. Cavallaro. Distributed and Decentralized Multicamera Tracking. *IEEE Signal Processing Magazine*, 3:46–58, May 2011.

[78] K. Tarabanis, R.Y. Tsai, and A. Kaul. Computing occlusion-free viewpoints. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(3):279–292, 1996.

[79] K.A. Tarabanis, P.K. Allen, and R.Y. Tsai. A survey of sensor planning in computer vision. *Robotics and Automation, IEEE Transactions on*, 11(1):86–104, 1995.

[80] D. Terzopoulos. Perceptive agents and systems in virtual reality. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, Osaka, Japan, 2003.

[81] D. Terzopoulos and F.Z. Qureshi. Virtual vision: Virtual reality subserving computer vision research for camera sensor networks. In *Distributed Video Sensor Networks*. Springer, 2011.

[82] K. Tieu, G. Dalley, and W.E.L Grimson. Inference of Non-Overlapping Camera Network Topology by Measuring Statistical Dependence. In *IEEE Intl. Conf. on Computer Vision*, 2005.

[83] R. Tron and R. Vidal. Distributed Algorithms for Camera Sensor Networks. *IEEE Signal Processing Magazine*, 3:32–45, May 2011.

[84] R. Tron, R. Vidal, and A. Terzis. Distributed Pose Averaging in Camera Sensor Networks via Consensus on SE(3). In *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, 2008.

[85] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision*. Prentice Hall, 1998.

[86] Zongjie Tu and Prabir Bhattacharya. Game-theoretic surveillance over arbitrary floor plan using a video camera network. *Signal, Image and Video Processing*, pages 1–17, 2013.

[87] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[88] D. Wolpert and K. Tumor. A Survey of Collectives. In K. Tumer and D. Wolpert, editors, *Collectives and the Design of Complex Systems*. Springer-Verlag, New York, NY, 2004.

[89] C. Wu and H. Aghajan. Real-Time Human Pose Estimation: A Case Study in Algorithm Design for Smart Camera Networks. *Proceedings of the IEEE Special Issure on Distributed Smart Cameras*, 96(10):1715–1732, October 2008.

[90] John Jun Wu, Rajeev Sharma, and Thomas S. Huang. Analysis of uncertainty bounds due to quantization for three-dimensional position estimation using multiple cameras. *Optical Engineering*, 37(1):280–292, 1998.

[91] Jiejun Xu, Zefeng Ni, Carter De Leo, Thomas Kuo, and Bangalore Manjunath. Spatial-temporal understanding of urban scenes through large camera network. In *Proceedings of the 1st ACM international workshop on Multimodal pervasive video analysis*, MPVA '10, pages 29–32, New York, NY, USA, 2010. ACM.

[92] Q. Yu, G. Medioni, and I. Cohen. Multiple Target Tracking Using Spatio-Temporal Markov Chain Monte Carlo Data Association. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[93] F. Zhang and N. Leonard. Cooperative filters and control for cooperative exploration. *IEEE Trans. Automatic Control*, 55(3):650–663, March 2010.

[94] Jian Zhao, S.S. Cheung, and Thinh Nguyen. Optimal camera network configurations for visual tagging. *Selected Topics in Signal Processing, IEEE Journal of*, 2(4):464–479, 2008.

[95] X. Zou, B. Bhanu, and A. Roy-Chowdhury. Continuous learning of a multilayered network topology in a video camera network. *EURASIP Journal on Image and Video Processing, Special Issue on Video-Based Modeling, Analysis, and Recognition of Human Motion*, 2009.