On-Line Control of Active Camera Networks

Dumitru Adrian Ilie

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2010

Abstract

Dumitru Adrian Ilie: On-Line Control of Active Camera Networks.
(Under the direction of Greg Welch.)

Large networks of cameras have been increasingly employed to capture dynamic events for tasks such as surveillance and training. When using active (pan-tilt-zoom) cameras to capture events distributed throughout a large area, human control becomes impractical and unreliable. This has led to the development of automated approaches for on-line camera control.

I introduce a new approach that consists of a stochastic performance metric and a constrained optimization method. The metric quantifies the uncertainty in the state of multiple points on each target. It uses state-space methods with stochastic models of the target dynamics and camera measurements. It can account for static and dynamic occlusions, accommodate requirements specific to the algorithm used to process the images, and incorporate other factors that can affect its results. The optimization explores the space of camera configurations over time under constraints associated with the cameras, the predicted target trajectories, and the image processing algorithm. While an exhaustive exploration of this parameter space is intractable, through careful complexity analysis and application domain observations I have identified appropriate alternatives for reducing the space. Specifically, I reduce the spatial dimension of the search by dividing the optimization problem into subproblems, and then optimizing each subproblem independently. I reduce the temporal dimension of the search by using empirically-based heuristics inside each subproblem. The result is a tractable optimization that explores an appropriate subspace of the parameters, while attempting to minimize the risk of excluding the global optimum.

The approach can be applied to conventional surveillance tasks (e.g., tracking or face recognition), as well as tasks employing more complex computer vision methods (e.g., markerless motion capture or 3D reconstruction). I present the results of experimental simulations of two such scenarios, using controlled and natural (unconstrained) target motions, employing simulated and real target tracks, in realistic scenes, and with realistic camera networks.

Acknowledgments

Thanks to:

- My advisor, Prof. Greg Welch, for his steadfast support over the years and for the many discussions and insights without which this research would not have been possible.
- My committee members:
  - Prof. Henry Fuchs and Prof. Anselmo Lastra for their insights, guidance and support, and for providing timely feedback on drafts of this thesis.
  - Dr. Christopher Jaynes for agreeing to serve on my committee, taking the time to fly to UNC for my defense and providing valuable feedback on an earlier draft of this thesis.
  - Prof. Marc Pollefeys for prompting the use of complexity analysis and telling me about the simulator which I used in my experiments.
  - Prof. Sanjoy Baruah for agreeing to serve on my committee, accommodating my defense schedule, and for providing insights into scheduling research and timely feedback on drafts of this thesis.
- My collaborators:
  - Stephen Guy for his invaluable help with the motion planning experiment, Sean Curtis for the subway station model, and Prof. Ming Lin for allowing them to help me.
  - Li Guan for providing sample reconstruction results.

  - Andrei State and Herman Towles for the many discussions and suggestions that helped me solve many problems along the way.
  - Chris Broaddus et al. from Sarnoff Corporation for providing the GPS tracks for the USMC exercise experiment.
  - Dr. Amela Sadagic from the Naval Post-graduate School for her expertise in military training.

- My family: my parents Constantin and Elena for believing in me, my brother George for always being there for me through college and beyond, and last but certainly not least my wife Barbara for her love, patience and support.

- Funding sources:
  - ONR grant N00014-08-C-0349 for "Behavior Analysis and Synthesis for Intelligent Training (BASE-IT)," led by Greg Welch (lead PI) and Henry Fuchs (PI) at UNC, Amela Sadagic (PI) at the Naval Post-graduate School, and Rakesh Kumar (PI) and Hui Cheng (Co-PI) at Sarnoff. Roy Stripling, Ph.D., Program Manager.
  - Cisco Systems grant for "Prototype for Two-station, Four-Person, Proper Eye-Gaze Telepresence System," Greg Welch and Henry Fuchs.
  - National Library of Medicine grant N01-LM-6-3544 for "3D Telepresence for Medical Consultation" led by Greg Welch (lead PI) and Prof. Henry Fuchs (PI) at UNC, Prof. Bruce Cairns, M.D. (Co-PI) at UNC, Prof. Ketan Mayer-Patel (Co-PI) at UNC, and Prof. Diane Sonnenwald (Co-PI) at Goteborg University and the University College of Boras. Craig Locatis, Scalable Information Infrastructure Project Officer.
  - NSF ITR grant 0121657 "Electronic Books for the Tele-Immersion Age: A New Paradigm for Teaching Surgical Procedures," led by Greg Welch (PI) at UNC and

Andy van Dam (PI) at Brown University. Ephraim Glinert and William Bainbridge, Program Managers.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

An old Chinese proverb says that "a picture is worth a thousand words" [Hep10]. Ever since their invention, cameras have proved adept at succinctly telling complex stories with just a single still image. Mainstream adoption has brought along developments such as color and video. Recent years have witnessed a tremendous increase in the deployment of cameras from street corners and building hallways to cellular phones, accompanied by a similar increase in images and videos captured and stored. With this increase came the need to process all the captured information automatically and extract parts that are relevant for a particular purpose. Computer vision has risen to this challenge, automating many tasks in application domains like surveillance, tracking, motion capture, and 3D reconstruction.

Of these application domains, 3D reconstruction and motion capture are more demanding in terms of the number of cameras required, but they are also capable of extracting the most information about the dynamic events being captured. Approaches include multi-view dynamic scene modeling [GFP07, GFP08, Gua10], space carving [KS00], 3D video [MWTN04], image-based visual hulls [MBR$^+$00] and mixed-scale motion recovery [Dav02]. These and other approaches have been demonstrated in environments ranging from a tabletop to an entire room. For example, Figure 1.1 shows such a camera setup and results obtained by Guan et al. in [GFP07, GFP08] for reconstructing events taking place in an area of approximately 10 meters in diameter. [MWTN04] and [Dav02] both show results for single targets in room-sized environments. There is demand to extend these and other approaches to *large environments*, where events can happen at *multiple dynamic locations*, simultaneously.

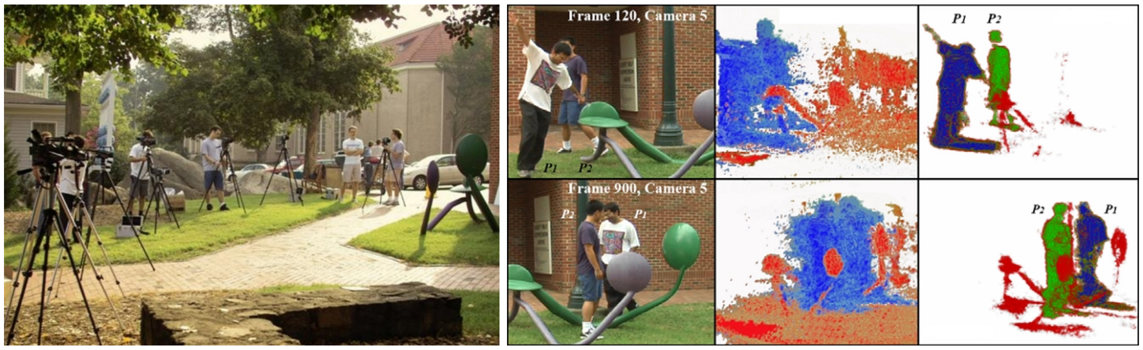


Figure 1.1: 3D reconstruction of a room-sized environment, from [GFP07, GFP08]. (Left) Outdoor data capturing with 9 cameras, from [GFP07]. (Right) Scene modeling results from [GFP08].

An example application domain with multiple capture locations in the same environment is the capture of complex surgeries for surgical training. This is currently accomplished by a single camera operator shooting standard video. The environments consist of large operating rooms, with events taking place around the operating table and the monitoring equipment. As surgical personnel move around, they may occlude the camera's view. To address the demand for giving trainees better understanding, Yang [Yan03] applied his View-Dependent Pixel Coloring method to help with synthesizing new views for surgical trainees, but the covered volume only included the surgery site inside a cubic camera rig 1 meter on a side. Figure 1.2 shows the camera rig and some results from [Yan03]. There is interest in automated capture of events taking place throughout the surgical room, with multiple cameras, and using computer vision approaches to reconstruct [WYB$^+$05].

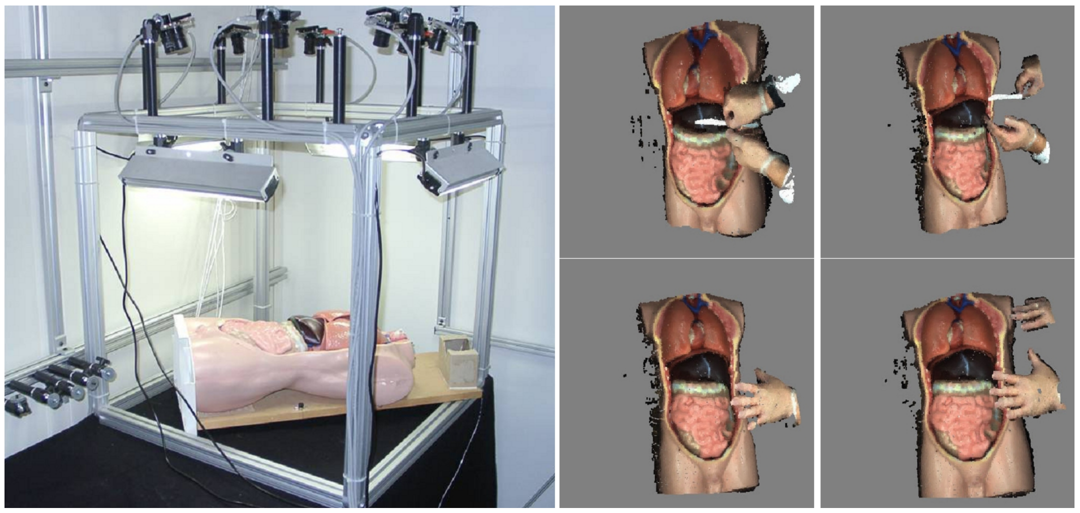


Figure 1.2: Camera rig and view synthesis results, from [Yan03].

An example application domain with large environments is capturing and providing feedback for military training exercises. Military training facilities are large sites with complex infrastructures, and some feature cameras that capture training exercises. Figure 1.3 shows a diagram of the infrastructure at the Kilo 2 MOUT (Military Operations on Urban Terrain) Training Facility at Camp Pendleton, CA. There are 37 *active cameras* (cameras that can change their pan, tilt and zoom settings) and 245 static cameras with various fixed fields of view in an area of approximately 1000×1200 feet. The infrastructure also includes other types of sensors, such as GPS (Global Positioning System) and INS (Inertial Navigation Sensors), which are worn by exercise participants to help with behavior analysis. Cameras and other sensors are placed on poles such as the one shown in Figure 1.4. Captured images are used for providing exercise participants with basic feedback. The active cameras are steered by a human operator manually, using a joystick. The approach described in [BGV+09] is currently being deployed at the facility to allow semi-automatic control by issuing high-level commands such as "follow this target or group of targets." There is interest in fully automating the capture process and using computer vision for automatic analysis and detection of correctable behaviors [SWB+09].

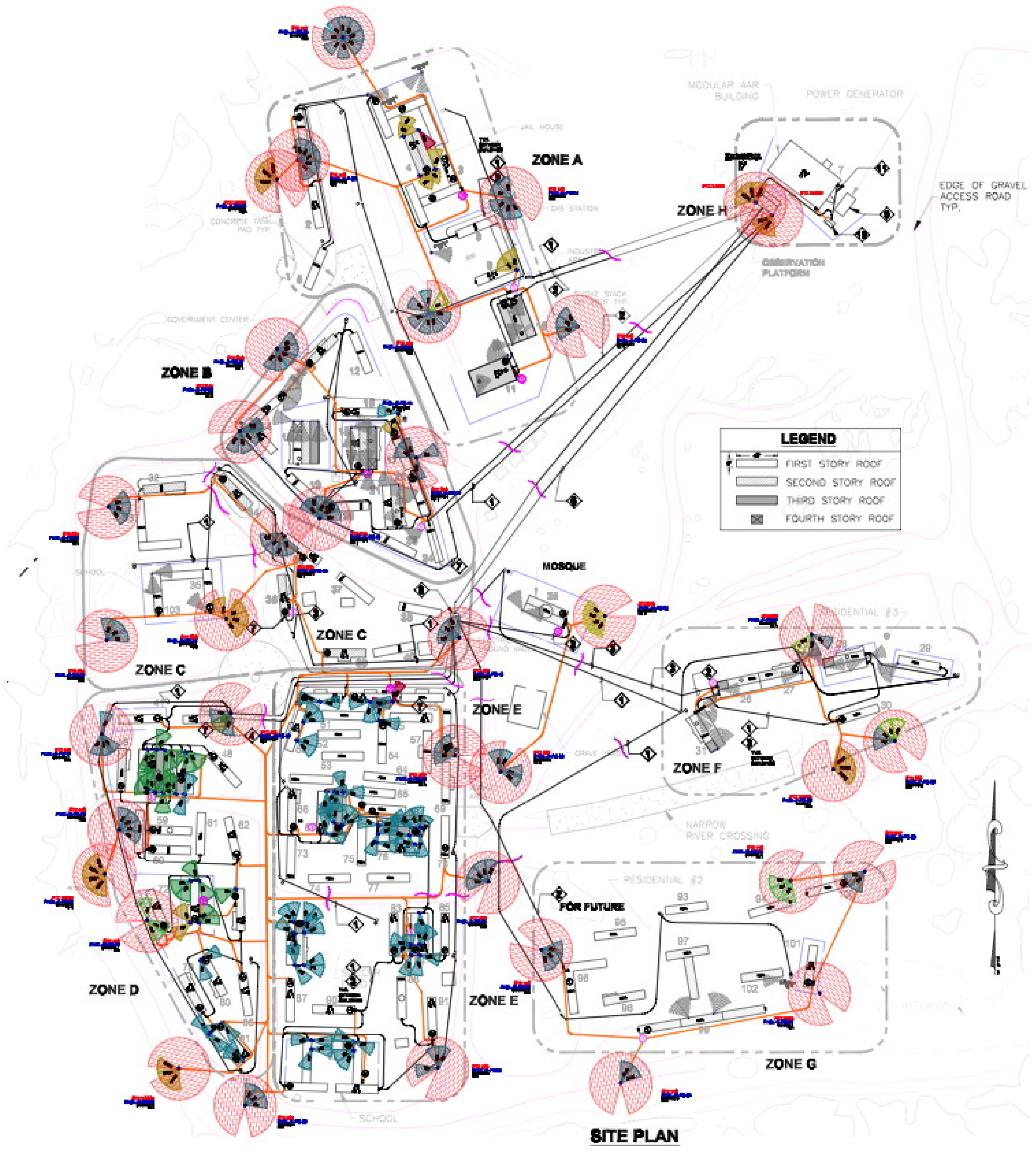


Figure 1.3: Diagram of camera infrastructure at Camp Pendleton, CA.
Active cameras are shown as red circles. Fixed cameras are shown as blue and green wedges.

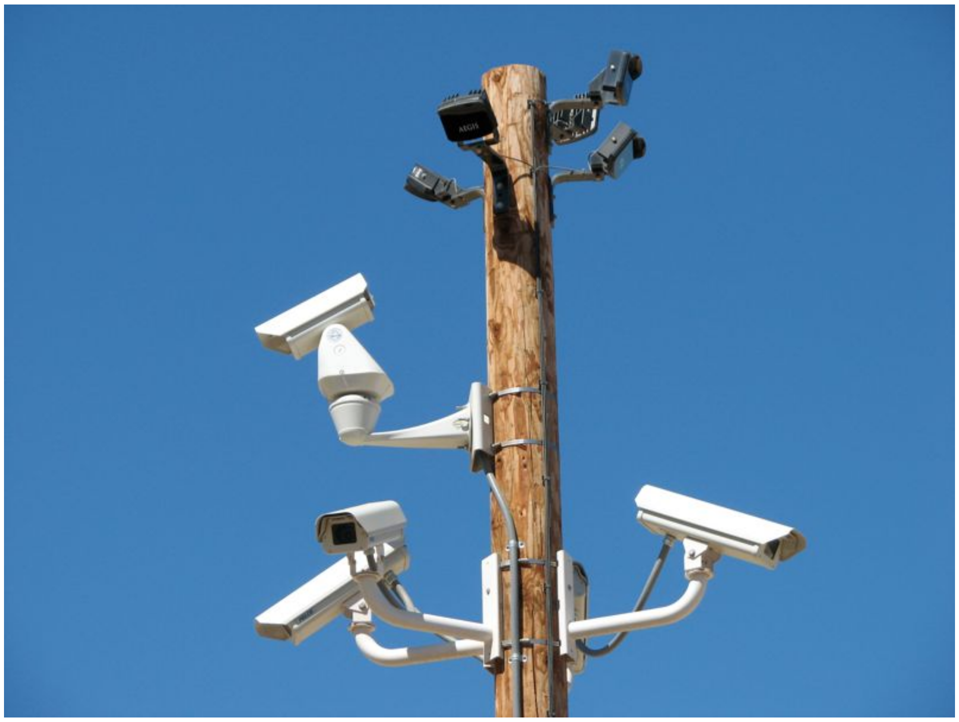


Figure 1.4: Light pole with sensors at Camp Pendleton, CA.
The bottom 3 cameras are fixed. The camera in the middle is an active camera. Other sensors are placed on top. Image by Dr. Amela Sadagic.

In practice, in many such large environments events only take place in a few *regions of interest* (ROIs), separated by regions of space where nothing of interest happens. I call such environments *sporadic*. If the locations of the ROIs are static, acceptable results can be obtained by straightforward replication of a *volumetric cell* for each ROI, each cell containing similar camera setups to the ones used used for smaller environments. However, if the locations of the ROIs are dynamic, coverage needs to be ensured throughout the entire volume. A brute-force extension by dividing the environment into cells and replicating the camera setups inside each cell is impractical: the number of cameras needed would be proportional to the number of cells in the environment. Additionally, camera placement is difficult for volumetric cells adjacent to each other. Other requirements, such as communication bandwidth, data storage, and computing power may increase as well.

One practical solution to this problem is using active cameras to cover sporadic environments. Active cameras have been used in surveillance [CLK+00] and computer vision fields such as motion capture [Dav02] and robotics [DM02]. What makes them so versatile is their capability to *maneuver*: when covering large, sporadic environments, they can change their pan and tilt settings to aim in the direction of dynamic ROIs, and zoom in or out to enclose the ROIs in their field of view. However, this versatility comes at a cost: in order to capture dynamic events, active cameras need to be controlled on-line, in real-time, and camera control decisions need to be made as events are happening. Control decisions need to take into account *factors* such as target dynamics and camera capabilities, as well as *requirements* from the computer vision algorithms the images are captured for, such as preferred camera configurations, capture durations or image resolutions.

The goal of this research is to control a network of active cameras on-line, in real-time, such that they capture multiple events taking place simultaneously in a sporadic environment and produce the best possible images for processing using computer vision algorithms.

1.1 Approach

In this thesis, I investigate the problem of on-line control of active cameras for best possible capture of dynamic events taking place in sporadic environments. I approach camera control as an *optimization problem* over the space of possible *camera configurations* (combinations of camera settings) and over time, under *constraints* derived from knowledge about the cameras, the predicted ROI trajectories and the computer vision algorithm the captured images are intended for. The computer vision algorithm that processes the images can be run in real-time, and provide feedback to the camera control method. An example of such an algorithm that can run in real-time is 3D tracking. Alternatively, constraint derivation can be done a priori, and more time-consuming computer vision algorithms can be run on the captured images long after the events have taken place. For example, 3D reconstruction algorithms can be time-consuming, but rigorously specifying their requirements allows the capture of images that can still guarantee good results even without real-time feedback.

In order to ensure complete coverage of multiple events taking place in a sporadic, large environment, active cameras need to be controlled on-line, in real-time, automatically. Many past surveillance approaches that deal with controlling active cameras are master-slave camera setups, aimed at specific surveillance tasks such as tracking or biometric tasks such as face recognition. These approaches work well in their domains, but are unable to provide the best imagery for complex computer vision tasks such as 3D reconstruction and motion capture, because they are not designed to take into account their specific requirements and the factors that influence their results. I present relevant previous approaches in more detail in Section 2.4.

I introduce a *conceptual framework* that allows casting camera control as a constrained optimization problem and applies both to simple surveillance tasks and to more complex computer vision approaches. Optimization methods rely on objective functions that quantify the "goodness" of a candidate solution in the search space. For camera control, this objective function is a *performance metric* that evaluates dynamic, evolving camera configurations. Application domains such as camera placement and selection make use of such performance metrics custom-tailored to their requirements. These existing metrics are not readily applicable to measuring the performance of active cameras in complex computer vision approaches like 3D reconstruction and motion capture because they fail to consider some of the factors that are important in these cases, such as temporal issues emerging from the dynamic nature of active cameras, and specific requirements of each computer vision algorithm. I present a few previous metrics in more detail in Section 2.4.2.

I describe a stochastic performance metric that provides a general and efficient means of quantifying performance as the expected uncertainty in target state space. The metric uses *state-space models* to describe target dynamics and measurement uncertainties. State-space models allow easy incorporation of requirements from the computer vision algorithms that will be using the images, as well as factors that have been known to affect their results. I describe this performance metric in detail in Chapter 5.

In Chapter 4, I examine two alternative search spaces for the optimization problem: the space of camera settings and the space of camera to ROI assignments. I first show that exhaustive exploration of both search spaces is intractable for non-trivial setups. Next, I use complexity analysis to arrive at *insights* on how to make the optimization tractable by targeting parameter subspaces for reducing complexity, while still seeking the smallest possible uncertainty in the target state space. I use the insights to arrive at a tractable control method to manage an *active camera network* (ACN) on-line, in real-time. The method, described in Chapter 6, combines a global assignment of cameras to ROIs that divides the problem into subproblems with a local optimization inside each subproblem. To validate the camera control method, I implemented a prototype *camera control system* and performed simulation-based experimental evaluation with real and synthetic ROI trajectories.

1.2 Thesis Statement

A stochastic objective function provides a general and efficient means to quantify the expected uncertainty in a 3D or higher-dimensional target state space, while accounting for target dynamics and measurement uncertainties, and supporting the incorporation of factors and constraints specific to the computer vision algorithm. This objective function can then be used to transform the problem of on-line real-time control of active cameras into a constrained optimization problem over the parameter space of camera configurations and over time.

While a complete optimization over this parameter space is intractable for on-line scenarios, careful complexity analysis offers insights into parameter subspaces that can be targeted for complexity reduction. Specifically, an approach that combines global camera-target assignment with local parameter planning facilitates on-line optimization aimed at achieving minimal uncertainty in the target state space over a tractable parameter subspace.

1.3 Innovations

In this section, I briefly present the innovations in my research. I first provide a short history on how I arrived at my dissertation, then list the innovations brought forth in this thesis. I also briefly mention a few other research contributions that are not immediately related to the topic of this thesis.

1.3.1 History of Research Innovations

Problem Domain: My interest in cameras and computer vision started with my work with advisor Prof. Greg Welch and committee member Prof. Henry Fuchs in the "Electronic Books for the Tele-Immersion Age" (eBooks) project, sponsored by the National Library of Medicine, and the "3D Telepresence for Medical Consultation: Extending Medical Expertise Throughout, Between, and Beyond Hospitals" (3DMC) project, sponsored by the National Science Foundation. In the eBooks project, we worked toward creating multimedia "electronic books" that would help surgical trainees better understand complex surgeries. The viewpoint synthesis work of Yang [Yan03] went a long way in advancing one of the ways complex surgeries are taught from a single, constrained view through the lens of a camera maneuvered by a human operator to unconstrained views from arbitrary viewpoints. Soon after, I became aware of surgeries that took place in multiple ROIs, which got me thinking of ways to keep the number of cameras low, while still capturing events as they happened in all ROIs. Active cameras were the obvious solution, and I started out by exploring an intelligence amplification approach to placing active cameras with Andrei State in [SWI06]. This simulation-based approach provides an interactive visual display of camera coverage and effective resolution. Among other applications, it helps users decide where to place active cameras in an environment such that desired parts of the environment could be reached by the camera in a specified time interval.

The "Behavior Analysis and Synthesis for Intelligent Training" (BASE-IT) project, sponsored by the Office of Naval Research, offered me the opportunity to solve a similar problem for a different type of environment: a military training facility where multiple events were happening simultaneously. Fortuitously, training facilities already are equipped with many cameras, some of them active, and there was interest in better using the active cameras to help provide training feedback to exercise participants. While working on this project, research discussions with advisor Prof. Greg Welch and committee members Prof. Henry Fuchs and Prof. Anselmo Lastra have helped steer me in the right direction when carving out the problem to solve and making sure I would arrive at a practical, useful solution. Figure 1.5 shows where my thesis research fits in the larger BASE-IT research project.

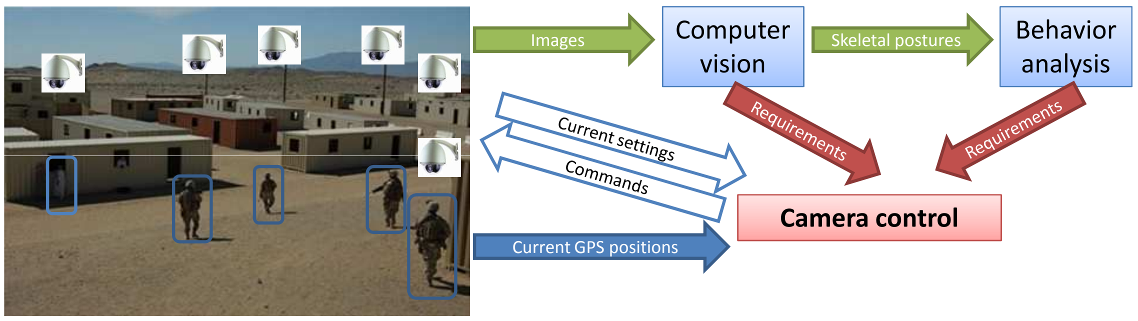


Figure 1.5: Camera control in the BASE-IT project.

Performance Metric: Approaching camera control as an optimization problem emerged from discussions with my advisor, Prof. Greg Welch. Our goal was to introduce a control approach that would provide the best possible images for computer vision algorithms, and we needed a way to characterize images captured by the cameras. Around the same time, Danette Allen, one of Prof. Welch's advisees, was finishing her dissertation [All07], in which she introduced "a stochastic framework for evaluating and comparing the expected performance of sensing systems for interactive computer graphics." One of the systems she analyzed was the camera setup shown in Figure 1.6. In [IWM08], my coauthors and I extended Danette's metric, the steady-state uncertainty in the system's ability to resolve 3D features, to evaluating active camera configurations. Uncertainty is an intuitive metric, as it measures the error in state space, and state-space models allowed the incorporation of many factors known to influence the results of computer vision algorithms. However, it soon became apparent that evaluating uncertainty at steady-state conflicted with the need to aggregate the metric over time. The entropy-based approach of Denzler et al. [DZ01, DBN01, DB01, DB02, DZN02] gave me the insight to evaluate uncertainty at each step and aggregate over time, using predicted ROI trajectories to compute possible occlusions.

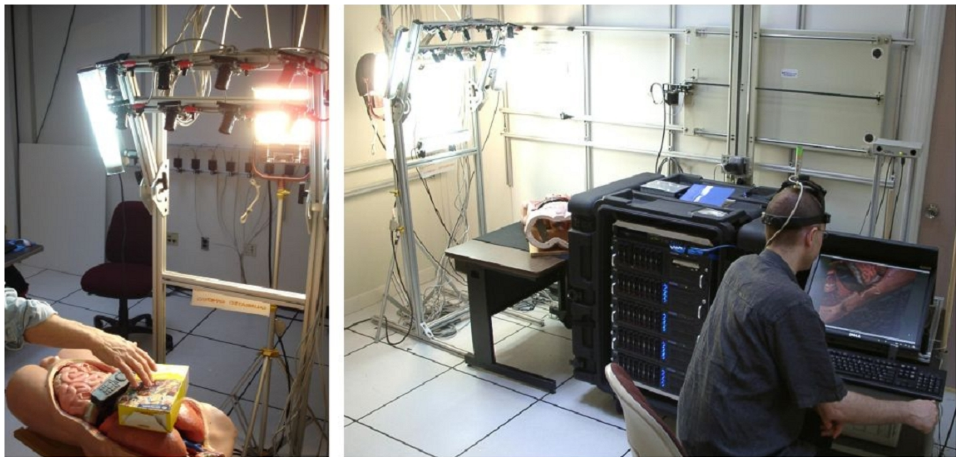


Figure 1.6: Camera setup from the 3DMC project, from [All07].

Empirically-based Heuristics: The next step was to provide an optimization method to explore the space of possible solutions. I first proposed a heuristic-based approach that was rooted in empirical studies of application requirements, typical target motions and camera limitations. These studies benefited from discussions with collaborators in the BASE-IT project both from UNC (Prof. Jan-Michael Frahm, Herman Towles) and from other institutions (Prof. Amela Sadagic, NPS and Chris Broaddus, Sarnoff Corporation). The heuristics were validated experimentally in a simple simulated setup that employed a simulator [TCB07] suggested by committee member Prof. Marc Pollefeys. Discussions with committee member Prof. Sanjoy Baruah made me aware of similar problems in radar dwell scheduling. Studies of related work in surveillance provided me with insights on the types of problems encountered and typical methods employed to solve them. While useful in their specific domains, none of these approaches fit the requirements of camera control for computer vision algorithms such as 3D reconstruction and motion capture, so a new approach was needed.

Complexity Analysis: My heuristics were shown to be effective in a limited number of cases, but lacked a rigorous justification for choosing them. In a committee meeting, Prof. Marc Pollefeys suggested a different approach: "the metric should be king," in the sense that in addition to being used to evaluate configurations, the metric should also govern the choices of rules and heuristics applied to reduce the search space size. This insight has spurred me in a new direction: using complexity analysis to compute the size of search space and identify appropriate alternatives for reducing it. Specifically, complexity analysis allowed me to examine the impact of each decision on the trade-off between the reduction in the search space size and the risk of not finding the optimal solution.

1.3.2 Thesis-Related Innovations

This thesis brings forth the following innovations:

- A general *conceptual framework* for automatic camera control that (Chapter 3):
  - defines control as an *optimization problem* where uncertainty is minimized in the target state space;
  - supports *known uses* for camera control (e.g., motion capture, recognition, surveillance, 3D reconstruction);
- Complexity analysis of two alternative control spaces:
  - camera settings, Section 3.3;
  - camera to ROI assignments, Section 4.2;
- Development of a set of *insights* aimed at *tractable control* (Chapter 4);
- A novel optimization *objective function* (Chapter 5) that:

  - measures and predicts task performance using *stochastic state-space models and methods*;
  - accommodates task *requirements* and known *factors* that affect task performance;

- A specific *method for on-line, real-time automatic camera control* (Chapter 6):
  - a proximity-based clustering algorithm for *breaking down the problem* into subproblems;
  - a greedy camera-subproblem assignment scheme for *reducing the computational burden* of the assignment of cameras to subproblems;
  - a local optimization process for *maintaining the best possible ROI coverage* inside each subproblem;

- Implementation of a prototype *camera control system* and simulation-based *experimental evaluation* with *real and synthetic ROI trajectories* (Chapter 7).

1.3.3 Additional Innovations

During my time at UNC I also worked on several other research projects and had the opportunity to interact and collaborate with a number of researchers and students. While not directly related to the research presented in this thesis, working on these projects has provided me with a solid background and prepared me to tackle the problem of camera control. Working on projector-based and head-mounted displays [ILW$^+$04] with graduate student Kok-Lim Low and Prof. Greg Welch, Prof. Anselmo Lastra and Prof. Henry Fuchs in the "Being There" and "Electronic Books for the Tele-Immersion Age" (eBooks) projects helped me better understand computer graphics and introduced me to tracking in general and the HiBall tracker [WBV$^+$01] in particular. Working with graduate student Ruigang Yang in the eBooks project got me familiarized with cameras and 3D reconstruction [Yan03] requirements. As part of my work in the “3D Telepresence for Medical Consultation: Extending Medical Expertise Throughout, Between, and Beyond Hospitals” (3DMC) project, I looked into improving 3D reconstruction quality by ensuring color consistency across multiple cameras through the use of hardware calibration and software refinement [IW05]. Finally, while working on the 3DMC, “Prototype for Two-station, Four-Person, Proper Eye-Gaze Telepresence System” and “Behavior Analysis and Synthesis for Intelligent Training” (BASE-IT) projects, I developed and maintained an application for photometric and geometric calibration of multiple cameras.

1.4 Dissertation Outline

I begin with a summary of related research in Chapter 2. In Chapter 3, I approach camera control as an optimization problem and analyze its complexity for two alternative search spaces. I use the analysis results in Chapter 4 to suggest and evaluate the impact of a set of guidelines aimed at reducing the search complexity. In Chapter 5 I describe the performance metric used in the optimization process, and show how task requirements and factors that are known to influence task performance can be incorporated into the metric. In Chapter 6, I use the insights gained from the guidelines in Chapter 4 to arrive at a tractable optimization method and present some details about its implementation into a prototype camera control system. I use simulation-based experiments to evaluate the performance of the prototype camera control system in Chapter 7. I conclude with Chapter 8, where I also discuss future plans for this research.

Chapter 2

Related Work

This chapter describes some related research efforts, grouped by their application domain: camera placement, camera selection, view planning, camera management for surveillance, camera control and camera calibration.

2.1 Camera Placement

The degree to which a particular arrangement of cameras, or *camera configuration*, satisfies task requirements can be characterized by what I call a *performance metric*. I call the factors that influence this metric *performance factors*. Example performance factors used in camera placement approaches include extrinsic camera parameters, optical camera parameters (pixel size, aperture, focal length, exposure time, gain, hue, saturation), camera models (such as perspective projection), object models (polygonal description, motion), and task constraints (visibility, field of view, focus, pixel resolution, incidence angle). The approaches listed in this section provide insights into previous performance metrics, the performance factors they take into account, and the optimization methods they employ to find the best solution.

Given knowledge about the cameras, the environment and the tasks to accomplish, *camera placement* aims to determine the camera configurations that best satisfy the task requirements. Camera placement methods include: generate and test [YHS95], synthesis [Cow88, TTK96], simulation [FCOL00, SWI06], expert systems [Mas95] and fuzzy logic [SSSAH04]. A comprehensive review of camera placement methods can be found in [TTA95].

Wu et al. [WSH98] use the the 2D quantization error on the camera image plane to estimate the uncertainty in the 3D position of a point when using multiple cameras. They model the quantization error geometrically, using pyramids, and the uncertainty region as an ellipsoid around the polyhedral intersection of the pyramids. The paper presents a computational technique for determining the uncertainty ellipsoid for an arbitrary number of cameras. Finally, the volume of the ellipsoid is used as a performance metric. Chen [Che02] improves this metric by taking into account probabilistic occlusion, and applies it to optimally place cameras for motion capture. Davis [Dav02] uses the resulting fixed camera arrangement in combination with steering 4 pan-tilt cameras for mixed-scale motion recovery.

Olague and Mohr [OM02] present an approach for camera network design to obtain minimal errors in 3D measurements. Error propagation is analyzed to obtain an optimization criterion. The camera projection model is used to express the relationship between 2D and 3D points, and the error is assumed to come only from image measurements. The covariance matrix of the 3D points is approximated using a Taylor expansion, and the maximum eigenvalue is used as the optimization criterion. Optimization is performed using a genetic algorithm, incorporating geometric and optical constraints such as occlusion.

Allen [All07] introduces steady-state uncertainty as a performance metric for optimizing the design of multi-sensor systems. In previous work [IWM08] my coauthors and I illustrate the integration of several performance factors into this metric and envision applying it to 3D reconstruction using PTZ cameras.

Mittal and Davis [MD04] compute the probability of visibility in the presence of dynamic occluders, under constraints such as field of view, fixed occluders, resolution, and viewing angle. Optimization is performed using cost functions such as the number of cameras, the occlusion probabilities, and the number of targets in a particular region of interest. In [MD08], they introduce a framework for incorporating visibility in the presence of random occlusions into sensor planning. The probability of visibility is computed for all objects from all cameras. A deterministic analysis for the worst case of uncooperative targets is also presented. Field of view, prohibited areas, image resolution, algorithmic (such as stereo matching and background appearance) and viewing angle constraints are incorporated into sensor planning, then integrated with probabilistic visibility into a capture performance metric. The metric is evaluated at each location, for each orientation and each given sensor configuration, and aggregated across space. The aggregated metric value is then optimized using simulated annealing and genetic algorithms.

Horster and Lienhart [HL06] use exact algorithms (linear programming) and constructive heuristics (greedy and dual sampling) to place fixed cameras in 2D floor plans while attaining the following goals: maximum coverage, best coverage under given cost, best orientations given positions, and cheapest configuration for given coverage percentage.

Naish et al. [NCB01b] compute the initial poses of the cameras in an active camera network given the expected object trajectory, such that the system effectiveness is maximized. This is achieved using a constrained, non-linear, direct search method in combination with simulations of the sensing system performance (i.e., dynamic dispatching to adjust the sensor poses in response to the object motion). The method in this work is a complement to their dynamic dispatching methodology presented in [NCB00, NCB01a], which selects and maneuvers subsets of sensors to achieve optimal data acquisition in real-time.

Ram et al. [RRA$^+$06] propose a performance metric based on the probability of accomplishing a given task for placing sensors in a system of cameras and motion sensors. The task is capturing frontal information of a symmetric target moving inside a convex region. Tasks are first decomposed into two subtasks: object localization and image capture. Prior knowledge about the sensors is used to assess the suitability of each sensor for each subtask, forming a performance matrix. Interaction among sensors is decided using the matrix such that assigning sensors to subtasks leads to maximum overall performance. Camera performance is evaluated as the probability of capturing the frontal part of the symmetric object.

Object orientation is modeled across a plane as a uniformly-distributed random variable. Motion sensors are transmitter-receiver pairs, placed on a grid. A trade-off between grid density and camera field of view is presented. A performance metric is computed as the capture probability at each grid point, averaged over the entire grid.

Bodor et al. [BSP05] compute the optimal camera poses for maximum task observability given a distribution of possible target trajectories. They develop a general analytical formulation of the observation problem, in terms of the statistics of the motion in the scene and the total resolution of the observed actions. An optimization approach is used to find the internal and external camera parameters that optimize the observation criteria. The objective function being optimized is directly related to the resolution of the targets in the camera images, and takes into account two factors that influence it: the distance from the camera to each target's trajectory and the angles that lead to forshortening effects.

Fleishman et al. [FCOL00] present a predictive, automatic camera placement for image-based modeling from scenes with known geometry. They employ a visibility algorithm that starts with a large database of potential camera positions to produce a minimal subset of camera positions that covers every visible polygon in a scene.

In [SWI06], my coauthors and I describe an interactive software simulator that assists with the design of multi-camera setups. The simulator assists users in interactively placing and manipulating multiple cameras within a pre-modeled 3D environment. It depicts the exact coverage of each camera (including indications of occluded and overlap regions), the effective spatial resolution on the surfaces, and the dynamic coverage by PTZ cameras of areas reachable within a user-selectable time interval.

In the business domain, camera placement has mostly been employed for placing cameras used in surveillance. Praetorian [L-310] is an example commercial solution that accomplishes this task.

2.2 Camera Selection

Sensor selection is usually performed in order to save power in wireless sensor networks. In the case of cameras, selection can be performed to minimize power consumption, as well as the use of other scarce resources such as bandwith and storage. Given a set of cameras, the goal of *camera selection* is to select the optimal subset that provides the best task performance under constraints given by limited resources. In this context, camera selection can be regarded as an optimization problem just like camera placement, but over a different parameter space. And, just as in the case of camera placement, the selection approaches in this section describe optimization methods as well as performance metrics and factors.

The authors of [EGG06, EYGG06] optimize a metric for camera placement and selection for localization of a single target in the presence of occlusion. Their metric is the minimum mean squared error of the best linear estimate of the object location in 2D. The placement problem is shown in [EYGG06] to be equivalent to a classical inverse kinematics robotics problem, which can be solved efficiently using gradient descent techniques. The selection problem is shown to be a combinatorial optimization problem for which finding the optimal solution is too costly. A semi-definite programming approximation for the problem is shown to achieve close to optimal solutions. In [EGG06], the authors deal with 2D static and dynamic occlusions.

Isler and Bajcsy [IB05] address the problem of selecting sensors to minimize the error in target position estimation. They consider a sensor model where measurements are polygonal, convex subsets of a plane that can be merged by intersecting them. An approximation algorithm that guarantees an estimation error within a factor of 2 of the least possible error is presented, and a constant number of sensors is shown to suffice for a good estimate. An experiment is presented where 19 cameras are used to estimate the position of a target on a known plane. The problem formulation is relaxed to where a set of possible locations of the target is given instead of a single estimate and the sensing model is non-convex and/or non-polygonal.

Denzler et al. [DB01, DBN01] present an information theoretic framework for camera data selection in 3D object tracking and derive a performance metric based on the uncertainty in the state estimation process. In [DZN02], the authors derive a performance metric based on conditional entropy to select the camera parameters that result in sensor data containing the most information for the next state estimation.

Chowdhury and Chellappa [CC04] address the problem of many algorithms selecting and processing more data than necessary in an attempt to overcome unacceptable performance in their results. They introduce an information-theoretic criterion for evaluating the performance of a 3D reconstruction by considering the change in mutual information between a scene and its reconstructions.

Soro and Heinzelman [SH07] propose methods for camera selection for best image performance from a desired viewpoint with minimal power consumption. They synthesize image data from a selection of cameras whose fields of view overlap with the desired field of view. Two camera selection methods are compared. The first method selects cameras that minimize the difference between the images provided by the selected cameras and the image that would be captured by a real camera from the desired viewpoint. The second method considers the energy limitations of the battery powered camera-nodes, as well as their importance in the 3D coverage preservation task.

2.3 View Planning

View planning, also known as *next best view* (NBV), computes the sequence of sensor poses that best satisfy a performance criterion when capturing a scene, while sometimes also minimizing the number of poses and the length of the path between them. It is employed in diverse domains such as object inspection [SS91, TTK96, TG94, CL04], image-based modeling and rendering [FCOL00], object reconstruction [WDH$^+$06, WDAN07], and indoor environment reconstruction [Low06]. Tarabanis et al. [TTA95], Scott et al. [SRR03] and Low [Low06] present comprehensive surveys of view planning methods. While the temporal aspect in view planning only refers to sensor dynamics and assumes static scenes, some of the performance factors employed in the approaches in this section are still relevant to the control of active cameras.

In his Ph.D. dissertation, Low [Low06] presents a new and efficient next-best-view algorithm for 3D reconstruction of indoor environments using active range sensing. He adopts a greedy approach to approximate the solution due to the intractability of the view planning problem and the lack of global geometric information. He formulates a performance metric that includes positioning, sensing and registration constraints, and reconstruction performance requirements. He employs a hierarchical approach that greatly accelerates the evaluation of the metric for a large set of views by exploiting the various spatial coherences in the acquisition constraints and reconstruction performance requirements when evaluating the view metric.

Chen and Li [CL04] solve an instance of the next best view problem in the context of automatic sensor placement for model-based robot vision. They present an approach based on a genetic algorithm that takes into account factors such as visibility, field of view, resolution, depth of field, overlap, occlusion, contrast, and reachability.

In [DDN03], the authors present a way to solve the next best view problem for active object recognition. They formally define the selection of additional camera views as an optimization problem and show how to use reinforcement learning for viewpoint training and selection in continuous state spaces. They also present an approach for the fusion of multiple views based on recursive density propagation. Experimental results show that a minimal number of views is selected, and optimal object recognition is achieved with respect to the classification.

2.4 Camera Scheduling for Surveillance

The most likely domain to benefit from my camera control approach is surveillance. Given a camera network, the goals of *surveillance* include domain-specific tasks such as following targets and obtaining 2D imagery for identification or behavioral analysis. Camera scheduling approaches encountered in surveillance applications can be categorized as *centralized* and *distributed* scheduling approaches. Centralized approaches can be broadly classified as adaptations of scheduling methods from other domains and optimizations. I list a few relevant examples from each category below.

2.4.1 Adapted Methods from Other Domains

One strategy to solve the camera scheduling problem is to apply scheduling policies from other domains.

A common surveillance problem is the acquisition of high resolution images of as many targets as possible before they leave the scene. A possible solution is to translate the problem into a real-time scheduling problem with deadlines and random new target arrivals. The authors of [dBP05, BdBP05] propose limiting the temporal extent of the schedules, due to the stochastic nature of the target arrivals and the requirement that a schedule be computed in real-time. They also propose taking into account the physical limitations of PTZ cameras, specifically the fact that zooming is much slower than panning and tilting. The camera is modeled as an interceptor with limited resources (adjustment speeds), and the target dynamics are assumed known or predictable. The overall stochastic problem is decomposed into smaller deterministic problems for which a sequence of saccades can be computed. The problem of choosing the best subset of targets for a camera to intersect in a given time is an instance of Time Dependent Orienteering (TDO): given a set of moving targets and a deadline, find the subset with the maximum number of targets interceptable before the deadline. TDO is a problem for which no polynomial-time algorithm exists. The optimal camera tour for a set of targets is computed by solving a Kinetic Traveling Salesperson Problem (KTSP): given a set of targets that move slower than the camera and the camera's starting position, compute the shortest time tour that intercepts all targets. KTSP has been shown to be NP-hard. After limiting the schedule duration, KTSP is reformulated as a sequence of TDO problems. Targets are placed in a queue sorted on their predicted residual time to exit the scene, and an instance of TDO is solved by exhaustive search for the first $7-8$ targets in the queue.

Naish et al. [NCB01b, NCB03, BNE$^+$06] propose applying principles from dispatching service vehicles to the problem of optimal sensing. They first propose a method for determining the optimal initial sensor configuration, given information about expected target trajectories [NCB01b]. The proposed method improves surveillance data performance by maneuvering some of the sensors into optimal initial positions, mitigating measurement uncertainty through data fusion, and positioning the remaining sensors to best react to target movements. As a complement of this work, the authors present a dynamic dispatching methodology that selects and maneuvers subsets of available sensors for optimal data acquisition in real-time [NCB03]. The goal is to select the optimal sensor subset for data fusion by maneuvering some sensors in response to target motion while keeping other sensors available for future demands. Demand instants are known a priori, and scheduling is done up to a rolling horizon of demand instants. Sensor fitness is assessed using a visibility measure that is inversely proportional to the measurement uncertainty when unoccluded and zero otherwise. Aggregating the measurements for several sensors is done using the inverse of the geometric mean of the visibility measures for all the sensors involved. A greedy strategy is used to assign the best k sensors for the next demand instant, then to assign remaining sensors to subsequent demand instants until no sensors remain or the rolling horizon is reached. The sensor parameters are adjusted via re-planning when the target state estimate is updated. In [BNE$^+$06] the authors describe an updated implementation using vehicle dispatching principles for tracking and state estimation of a single target with 4 PTZ cameras and a static overview camera.

Costello et al. [CDBF04] present and evaluate the performance of several scheduling policies in a master-slave surveillance configuration (a fixed camera and a PTZ camera). Their goal is to capture data for identifying as many people as possible, and it can be broken into two objectives: capture high resolution images for as many people as possible and view each person for as long as possible. The proposed solution is to observe each target for an empiricallydetermined period of time, then move on to the next target, possibly returning to the first target if it is still in the scene. The camera scheduling problem is considered similar to a packet routing problem: the deadline and amount of time to serve become known once a target enters the scene. However, the deadlines are only estimated, serving a target for a preset time does not guarantee the task is accomplished, and a target can be served multiple times. This can be treated as a multi-class scheduling problem, with class assignments done based mainly on the number of times a target has been observed (other factors can be taken into account). The paper evaluates several greedy scheduling policies. The static priority (always choose from the highest class) policies analyzed were: random, first come, first serve (FCFS+), earliest deadline first (EDF+). Dynamic priority policies include EDF, FCFS and current minloss throughput optimal (CMTO). CMTO assigns a weight to each class, and tries to minimize the loss due to dropped packets. Scheduling was done by looking ahead to a horizon (cut) specified by the earliest time a packet will be dropped based on the packet deadlines. A list was formed with the highest weight packets with deadlines earlier than the cut, and the packet that results in the most weight served by the cut was selected. EDF+ was shown to outperform FCFS+ and CMTO in percentage of targets captured, but was worst in terms of the number of targets captured multiple times.

Lim et al. [LMD05, LMD07] propose solving the camera scheduling problem using dynamic programming and greedy heuristics. The goal of their approach is to capture images that satisfy task-specific requirements such as: visibility, movement direction, camera capabilities, and task-specific minimum resolution and duration. They propose the concept of *task visibility intervals* (TVIs), intervals constructed from predicted target trajectories during which the task requirements are satisfied. TVIs for a single camera are combined into MTVIs (multiple TVIs). Single camera scheduling is solved using dynamic programming (DP). A directed acyclic graph (DAG) is constructed with a common source and a common sink, (M)TVIs as nodes, and edges connecting them if the slack start time of one precedes the other. DP starts from the DAG sink, adjusts the weights of the edges and terminates when all nodes are covered by a path. Multi-camera scheduling is NP-hard, and is solved using a greedy approach, picking the (M)TVI that covers the maximum number of uncovered tasks. A second proposed approach uses branch and bound algorithm that runs DP on a DAG with source-sink subgraphs for each camera, connected by links from the sinks of some subgraphs to the sources of others. The greedy approach is shown to have significantly decreasing performance when the number of cameras increase.

Qureshi and Terzopoulos [QT05a, QT05b, QT07] present a Virtual Vision paradigm for the design and evaluation of surveillance systems. They use a virtual environment simulating a train station, populated with synthetic autonomous pedestrians. The system employs several wide field-of-view calibrated static cameras for tracking and several PTZ cameras for capturing high-resolution images of the pedestrians. The PTZ cameras are not calibrated. A coarse mapping between 3D locations and gaze direction is built by observing a single pedestrian in a preprocessing step. Fixation and zooming are purely 2D and do not rely on 3D calibration. Local Vision Routines (LVRs) are employed for pedestrian recognition, identification, and tracking. The PTZ controller is built as an autonomous agent modeled as a finite state machine, with free, tracking and searching as possible states. When a camera is free, it selects the next sensing request in the task pipeline. The authors note that, while bearing similarities to the packet routing problem as described by Costello in [CDBF04], scheduling cameras has two significant characteristics that set it apart. First, there are multiple "routers" (in this case, PTZ cameras), an aspect the authors claim is better modeled using scheduling policies for assigning jobs to different processors. Second, camera scheduling must deal with additional sources of uncertainty due to the difficulty estimating when a pedestrian might leave the scene and the amount of time for which a PTZ camera should track and follow a pedestrian to record video suitable for the desired task. Third, different cameras are not equally suitable for a particular task, and suitability varies with time. A weighted round-robin scheduling scheme with a FCFS+ priority policy is proposed in [QT05a] for balancing two goals: getting high resolution images and viewing each pedestrian for as long or as many times as possible. Weights are modeled based on the adjustment time and the camera-pedestrian distance. The danger of a majority of the jobs being assigned to the processor with the highest weight is avoided by sorting the PTZ cameras according to their weights with respect to a given pedestrian and assigning the free PTZ camera with the highest weight to that pedestrian. Ties are broken by selecting the pedestrian who entered the scene first. Other possible tie breaking options like EDF+ were not considered because they require an estimate of the exit times of the pedestrians from the scene, which are difficult to predict. The amount of time a PTZ camera spends viewing a pedestrian depends upon the number of pedestrians in the scene, with a minimum set based on the number of frames required to accomplish the surveillance task. Weighted scheduling is shown to outperform non-weighted scheduling.

Taylor et al. [TCB07] present a visual surveillance simulation testbed based on a commercial game engine. The tool simulates static, PTZ and omnidirectional cameras with realistic effects for noise and radial distortion, and offers ground truth for algorithm testing. Several indoor and outdoor virtual environments are used to illustrate the range of possible testing scenarios, and a practical demonstration of using the tool to develop and evaluate surveillance algorithms is given. I use this simulation testbed for my experiments in Section 7.1. In Section 7.1.1, I describe an extension of this tool's functionality to allow for target movement under program control.

2.4.2 Optimization of a Metric

Another approach to camera scheduling is the optimization of a cost or performance metric at each time step. The schedule consists of a sequence of actions that lead to the optimal camera configuration at each subsequent time step. Several metrics have been proposed in contexts such as sensor planning [TTA95, MD08], error analysis [WSH98], and sensor selection [CD08, DZN03].

In [DZN03], Denzler et al. present a performance metric for selecting the optimal focal length in 3D object tracking. The determinant of the a-posteriori state covariance matrix is used as a measure of uncertainty derived from the expected conditional entropy given a particular action. Visibility is taken into account by considering whether observations can be made and using the resulting probabilities as weights. Optimizing this metric over the space of possible camera actions yields the best actions to be taken by each camera. The authors of Deutsch et al. [DZDN04, DND05] improve the process by using sequential Kalman filters to deal with a variable number of cameras and occlusions, predicting several steps into the future and speeding up the computation. Sommerlande and Reid add a Poisson process to model the potential of acquiring new targets by exploring the scene [SR08b], examine the resulting camera behaviors when zooming [SR08a], and evaluate the effect on the performance of random and first-come, first-serve (FCFS) scheduling policies [SR08c].

Krahnstoever et al. [KYL$^+$08] present a system for controlling four PTZ cameras to accomplish a biometric task. Target positions are known from a tracking system with 4 fixed cameras. Scheduling is accomplished by computing plans for all the cameras: lists of targets to cover at each time step. Plans are evaluated using a probabilistic performance objective function to optimize the success probability of the biometric task. The objective function is the probability of success in capturing all targets, which depends on a quantitative measure for the performance of each target capture. The capture performance is evaluated as a function of the incidence angle, target-camera distance, tracking performance (worse near scene boundaries), and PTZ capabilities. A temporal decay factor is introduced to allow repeated capture of a target. Optimization is performed asynchronously, via combinatorial search, up to a time horizon. Plans are constructed by iteratively adding camera-target assignments, defining a directed acyclic weighted graph, with partial plans as nodes and difference in performance as edge weights. Plans that cannot be expanded further are terminal nodes and candidate solutions. A best-first strategy is used to traverse the graph, followed by coordinate ascent optimization through assignment changes. All plans are continuously revised at each time instant. New targets are added upon detection from monitoring a number of given entry zones.

Broaddus et al. [BGV$^+$09] present “ACT vision”, a system consisting of a network of PTZ cameras and GPS sensors covering a single connected area that aims to maintain visibility of designated targets. They use a joint probabilistic data association algorithm to track the targets. Cameras are tasked to follow specific targets based on a cost calculation that optimizes the task to camera assignment and performs hand-offs from camera to camera. They compute a “cost matrix” C that aggregates terms for target visibility, distance to maneuver, persistence in covering a target and switching to an already covered target. Availability is computed as a “forbidden matrix” F. They develop two optimization strategies: one that uses the minimum number of cameras needed, and another that encourages multiple views of a target for 3D reconstruction. The task to camera assignment is performed using an iterative greedy k-best algorithm.

Mittal and Davis [MD04] compute the probability of visibility in the presence of dynamic occluders, and optimize using cost functions such as the number of cameras, occlusion probabilities, or number of targets in a particular region of interest under constraints such as field of view, fixed occluders, resolution, and viewing angle.

Yous et al. [YUK07] propose a camera assignment scheme based on the visibility analysis of a coarse 3D shape produced in a preprocessing step to control multiple Pan/Tilt cameras for 3D video of a moving object. The optimization is then extended into the temporal domain to ensure smooth camera movements. The interesting aspect of this work is that it constructs its 3D results from close-up images of parts of the object being model, instead of trying to fit the entire object within the field of view of each camera.

Scotti et al. [SMR03] use a metric that takes into account the position of the targets and the area of their projection in the image, and optimize it using self-organizing maps. The result is used to tune the parameters of image processing algorithms used in the surveillance task.

2.4.3 Distributed Surveillance

In distributed surveillance the decisions are arrived at through contributions from collaborating or competing autonomous agents. Proponents of distributed approaches argue that overall intelligent behavior can be the result of the interaction between many simple behaviors, rather than the result of some powerful but complicated centralized processing. Examples of the some of the issues and reasoning behind distributed processing as implemented in 3^{rd} generation surveillance systems can be found in [MOFR01, OFR01, RSJ03].

Matsuyama and Ukita [MU02] describe a distributed system for real-time multi-target tracking. The system is organized in three layers (inter-agency, agency and agent). Agents dynamically interchange information with each other. An agent can look for new targets or and can join an agency which is already tracking a target. When multiple targets get too close to be distinguishable from each other, the agencies tracking them are joined until the targets separate.

In [QT05b, QT07] Qureshi and Terzopoulos apply their Virtual Vision paradigm for the design and evaluation of a distributed surveillance system. Local Vision Routines (LVRs) and a state model are still employed, as in the centralized system described in[QT05a]. However, cameras can organize into groups to accomplish tasks using local processing and inter-camera communication with neighbors in wireless range. The node that receives a task request is designated as the supervisor and it broadcasts the request to its neighbors. While camera network topology is assumed as known, no scene geometry knowledge is assumed, only that a target can be identified by different cameras with reasonable accuracy. Each camera computes its own relevance for a task, based on whether it is free or not, how well it can accomplish the task, how close it is to the limits of its capabilities, and reassignment avoidance. The supervisor forms a group and greedily assigns cameras to tasks, giving preference to cameras that are free. Cameras are removed from a group when they cease to be relevant to the group task. Inter-group conflicts are solved at the supervisor of one of the conflicting groups as a constraint satisfaction problem, and each camera is ultimately assigned to a single task. Communication and camera failures are accounted for, but supervisor failure is solved by creating new groups and merging old groups. No performance comparison is attempted between the centralized and distributed scheduling approaches.

2.5 PTZ Camera Control

One of the factors that need to be taken into account in active camera control is the fact that PTZ cameras are mechanical devices, with electrical motors and moving parts. Knowledge about the way cameras react to commands is a fundamental component of designing a camera control system.

Several characteristics of the human oculo-motor system have been suggested to be useful also for controlling cameras. Many active camera systems employ a control scheme based on two different modes, called *smooth pursuit* and *saccade*. In humans, a saccade is used to center the fovea on the target at some point in the future, and guarantee that smooth pursuit can take over. Rivlin and Rotstein [RR00] present a way to systematically evaluate the benefits of implementing such a control scheme based both on image processing constraints and on control considerations. The problems they consider include: sampling measurements of continuous phenomena and producing continuous control signals, large processing time delays, tight mechanical specifications, and interactions between controllers. They show that in the case of non-uniform image resolution, the size of the fovea can be optimized as a trade-off between computation time and tighter tracking specifications. The problem is formulated as a one-variable maximization by using linear optimal control theory.

Qureshi and Terzopoulos [QT05a] describe a different control scheme. Their camera control is modeled as an augmented finite state machine with 4 states: free, tracking, searching and lost. To acquire images of a target, a camera would first choose an appropriate gaze direction at the widest zoom, then fixate and zoom in after the target is positively identified. Traditional proportional derivative controllers are shown to generate unsteady control signals resulting in jittery camera motion, so the authors propose dual-state controllers for fixation and zooming, with act and maintain as the 2 states.

Davis [Dav02] presents a mixed-scale motion recovery approach that uses 4 pan-tilt cameras to recover small scale motion. He uses gradient descent to find nearby local minima and avoid large camera maneuvers, and prediction to alleviate the latency in camera response time.

2.6 Calibrating PTZ Cameras

One of the hurdles that have to be overcome in order to use active cameras for 3D reconstruction is imprecise geometric calibration due to the lack of precision and repeatability in setting the values for the pan, tilt and zoom settings. The approaches presented in this section are efforts directed at solving this problem.

In [SP04], Sinha and Pollefeys discuss the problem of recovering the calibration of an ACN. They present a two-step procedure for estimating the intrinsic parameters of each camera over the entire zoom range. The first step is to use a panorama to accurately determine the intrinsic parameters at the lowest zoom setting. A grid of overlapping images is captured, and pairwise homographies are computed and chained with respect to a reference image. The intrinsic parameters are computed using two separate bundle adjustments: the first to correctly align the images and the second to refine the estimate obtained using a linear algorithm that computes intrinsic parameters of a purely rotating camera. The second step is to determine the intrinsic parameters at increasing zooms by capturing images in a fixed direction while progressively zooming in. Both steps are fully automatic and do not assume any knowledge of the scene structure. The paper concludes with a discussion on how to compute the extrinsic parameters for all the cameras by computing epipolar geometry for sufficient camera pairs and how the approach can be used in the context of active camera networks.

Wu et al. [WZB$^+$06] use two extended Kalman filters to simultaneously estimate zoom and pan-tilt parameters based on read-outs from the cameras. Frame-to-frame homographies are used to improve the Kalman Filter estimates, as read-outs are slow and imprecise. The Kalman Filters use uncorrelated constant velocity models. There is one filter for the zoom setting, and another for the pan and tilt settings combined. The zoom filter is applied first, then the zoom estimate is used in the pan and tilt filter.

Chapter 3

Problem Analysis

In this chapter, I approach the control of active cameras as an optimization problem and present a complexity analysis of two possible search spaces.

When used for active camera control, I consider optimization to be the continual exploration of the space of possible solutions in search for the best solution under constraints ranging from the physical limitations of the cameras to algorithmic requirements. An essential element of optimization methods is the *objective function*. For my camera control approach, this objective function is a *performance metric* that evaluates the performance of dynamic, evolving camera configurations in providing images best suitable for the computer vision algorithm that will process them.

3.1 Conceptual Framework

Approaching camera control as an optimization problem can be formalized by defining the concepts involved. I organize these concepts into the conceptual framework below:

- **Active camera networks (ACNs)**: an ensemble of PTZ cameras that collaborate to accomplish a specific task.
- **Regions of interest (ROIs)**: the static or dynamic 3D volumetric regions where events take place. For example, a static ROI could be an entry point into she scene, and a

dynamic ROI would include a single target and would move with it.

- **Configuration**: the set of camera settings values for all the cameras in the ACN.
- **Time unit**: the unit used to discretize time.
- **Dwell time**: the period of time during which a camera holds its settings constant and captures images. The duration of a dwell can vary from the time it takes to capture a single frame (in surveillance tasks that continuously follow targets) to infinity (in computer vision tasks that work with fixed cameras).
- **Uninterruptible dwell**: the dwell time needed by some computer vision algorithms for tasks such as calibration and model initialization. This dwell is considered uninterruptible because interrupting it would make the aforementioned tasks fail.
- **Interruptible dwell**: the dwell time needed by computer vision algorithms that work better with continuous image sequences. This dwell is considered interruptible because interrupting it does not cause undesirable consequences.
- **Transition time**: the period of time it takes a camera to change its settings. Most PTZ cameras are able to capture during transitions, but not many applications can make use of the resulting images, due to the blurring that usually occurs during transitions.
- **Planning method**: the algorithm responsible for controlling the cameras.
- **Planning cycle**: the period of time needed by the planning method to generate the best possible plan to control the cameras under the task constraints.
- **Planning step**: a transition time followed by a dwell time.
- **Planning horizon**: the period of time over which the planning method looks ahead during each planning cycle.

- **Plan**: a sequence of planning steps up to the planning horizon, including camera parameters at each time unit.

- **Camera contribution**: the information provided by a camera's capture of an event during a time unit.

- **Plan quality**: the aggregated contributions of all cameras over the time horizon for a particular plan.

- **Performance metric**: a numerical measure of a plan's quality.

- **Client**: the computer vision task for which the cameras are capturing images.

Given this framework, the requirements of specific tasks can be specified as in the following examples:

- In a typical **surveillance client application** where the task is to simply capture images of the targets, there would be no uninterruptible dwells, and interruptible dwells will have varying, typically small durations. Most of the time, the cameras would make short transitions following the targets. Longer transitions would be needed when cameras switch targets. Planning would mostly involve deciding whether to perform a short transition while following the same target or a long transition when switching targets. The planning horizon would need to be as long as the longest possible transition followed by the longest possible interruptible dwell. The contribution of a single camera to the capture of each target would typically be sufficient. The overall complexity of the decision process would be minimal, so the planning cycle would be short. The performance metric could range from a simple count of targets captured over time to incorporating other factors such as image resolution.

- In contrast, a **3D reconstruction client application** would have a very different set of requirements. Uninterruptible dwells may be needed for geometric calibration and internal model reinitialization. The planning would need to balance the need for longer interruptible dwells at lower zoom factors with the reduced image resolution that typically results from such dwells. A longer planning horizon would need to be explored, resulting in a more complex planning process and a longer planning cycle. Multiple cameras would typically be needed to contribute to the capture of a target. The performance metric would likewise be more complex, in order to properly accumulate contributions from multiple cameras and to accommodate application-specific requirements.

3.2 Search Space Size

The first step in trying to assess whether an optimization approach can be implemented in real-time is to compute the size of the search space to be explored and the computational complexity of various approaches to exploring that space.

Intuitively, the optimal solution at each time instant would be the *camera configuration* (set of all camera setting values) that results in the optimal objective function value. However, in order to accommodate the constraints stemming from the dynamic nature of active cameras and the events they capture, as well as the special requirements of computer vision algorithms, a simple search for the best-performing camera configuration at each time instant is no longer sufficient. Instead, the optimization method and its associated performance metric must also include time as a dimension of the search space. The performance metric must not only aggregate camera contributions over all cameras and all ROIs, but also over time. The following examples illustrate a few constraints that can only be incorporated into the performance metric and the planning method when time is a dimension of the search space:

- **Cameras cannot change their settings instantaneously** [CDBF04, BNB04], so the setting values of an optimal camera configuration have to be computed not for the current time instant, but for a future time instant that takes into account the time it takes the cameras involved to apply the desired camera settings. This constraint can be taken into account by having configuration changes incur *temporal costs*: during the time when cameras change their settings, they do not contribute to the capture of any ROI, resulting in worse performance when aggregated over time.

- **Computer vision algorithms have special requirements**: many need accurate geometric camera calibration such as the one described in [Zha99], and some rely on internal models [BC08] that need to be re-initialized when camera parameters change. These operations require image sequences captured with constant camera settings. This constraint can be taken into account by having configuration changes also incur *algorithmic costs*: the control method would have cameras fixed during these calibration and re-initialization procedures. Moreover, many computer vision algorithms, such as the optical flow-based algorithm in [BC08], work better with continuous image sequences captured by fixed cameras than with separate images captured with different camera settings. This constraint can be taken into account by having the performance metric accumulate camera contributions over time.

These and other requirements of computer vision algorithms affect the optimization process: instead of finding the optimal configuration at each time instant, it needs to look ahead in time to find the configuration that would become optimal at a future time instant and remain optimal for a specific duration. Consequently, for my approach, the search space is the space of possible camera configurations over time. Constraints are derived from knowledge about the cameras, the predicted ROI trajectories and the computer vision algorithm the captured images are intended for.

Let $nConfigs$ be the total number of possible camera configurations at each evaluation. Let $nSteps$ be the number of steps in the plan, $lStep$ be the average length of a plan step in cycles, and $lCycle$ be the length of a planning cycle in time units. The number of plan cycles $nCycles$ up to the planning horizon is related to the number of steps $nSteps$ as follows:

$$nCycles = nSteps \cdot lStep \tag{3.2.1}$$

The number of time units *nUnits* up to the planning horizon is related to the number of steps *nSteps* as follows:

$$nUnits = nSteps \cdot lStep \cdot lCycle \tag{3.2.2}$$

The planning process can be represented as a perfect *n*-ary tree of height *nSteps*, with $n = nConfigs$ possible alternatives to be evaluated for each plan step. The size of the search space in terms of the number of plans to evaluate, *nPlans*, can be computed as the number of leaf nodes in the tree:

$$nPlans = \prod_{k=1}^{nSteps} nConfigs = nConfigs^{nSteps} \tag{3.2.3}$$

A single performance metric evaluation aggregates the *contributions* of each camera to the capture of each ROI over the *nUnits* time units until the planning horizon. In the computational complexity analysis in this thesis, I consider the *contribution computation* to be the atomic operation to perform when evaluating the metric. All computations are expressed in terms of this atomic operation.

Let *nROIs* be the total number of ROIs. The number of camera contributions evaluated for a single metric computation is:

$$nContribs = nCams \cdot nROIs \cdot nUnits \tag{3.2.4}$$

Let *SC* be the computational complexity of an exhaustive search. Using *nPlans* from Equation 3.2.3 and *nContribs* from Equation 3.2.4, the general formula for *SC* in terms of the number of contribution computations is:

$$SC = nPlans \cdot nContribs = nConfigs^{nSteps} \cdot nCams \cdot nROIs \cdot nUnits \tag{3.2.5}$$

In the Search Complexity Equation 3.2.5, it is the exponential term defined as *nPlans* in the Plans Equation 3.2.3 that grows the fastest. In the following sections, I look at the complexity of exhaustively exploring two different search spaces. In Chapter 4, I show the effect rules and heuristics can have on reducing the search complexity *SC*.

3.3 The Space of Camera Settings

One way to explore the space of possible camera configurations would be to simply try out all the possible combinations for the pan, tilt and zoom setting values. Let P be the number of possible values for the pan setting, T the number of possible values for the tilt setting, and Z the number of possible values for the zoom setting. The number of possible camera settings combinations to explore for each planning cycle is the product $P \cdot T \cdot Z$. The number of possible camera settings combinations for each camera c during a plan step of length $lStep$ planning cycles is:

$$nSettings_c = \prod_{s=1}^{lStep} (P \cdot T \cdot Z) = (P \cdot T \cdot Z)^{lStep} \tag{3.3.1}$$

Note that the camera settings corresponding to the configuration for the current plan step are not necessarily applied during the first planning cycle in the step. Instead, all $P \cdot T \cdot Z$ settings combinations must be explored at each cycle into the future up to $lStep$ cycles, because during each cycle there is an implicit decision whether to leave the camera with its current settings from the previous plan step, or to change them to a new configuration corresponding to the current plan step.

When using *nCams* cameras, assuming they are of the same type, the number of possible camera configurations to explore for each plan step of length $lStep$ cycles is:

$$nConfigs_{settings} = \prod_{c=1}^{nCams} nSettings_c = (P \cdot T \cdot Z)^{lStep \cdot nCams} \tag{3.3.2}$$

Substituting Equation 3.3.2 into the Search Complexity Equation 3.2.5, the total number of contribution computations when exhaustively exploring the entire camera settings space for all cameras all the time is:

$$SC_{settings} = \left((P \cdot T \cdot Z)^{lStep \cdot nCams}\right)^{nSteps} \cdot nCams \cdot nROIs \cdot nUnits \qquad (3.3.3)$$

To illustrate the search complexity of exhaustively exploring this search space with a concrete numerical example, I set the number of possible values for the camera settings to $P = 36$, $T = 9$ and $Z = 5$. I also set the time horizon to $nUnits = 10$, the length of a cycle to $lCycle = 1$, and assume there are only $nSteps = 2$ steps before the planning horizon, resulting in the average length of a plan step $lStep = 5$ cycles. Using Equation 3.3.1, these values result in $nSettings_c = 1620^5$. I set the number of cameras to $nCams \in \{10, 20, 30\}$, and vary the number of ROIs $nROIs$ between 5 and 20. Figure 3.1 shows a plot of the search complexity in terms of computed contributions. Because for some values of $nCams$ and $nROIs$, the search complexity $SC_{settings}$ is too large to be represented in MatLab, I plotted $log_{10}(SC_{settings})$ instead, and used a logarithmic scale to compress the vertical dimension.

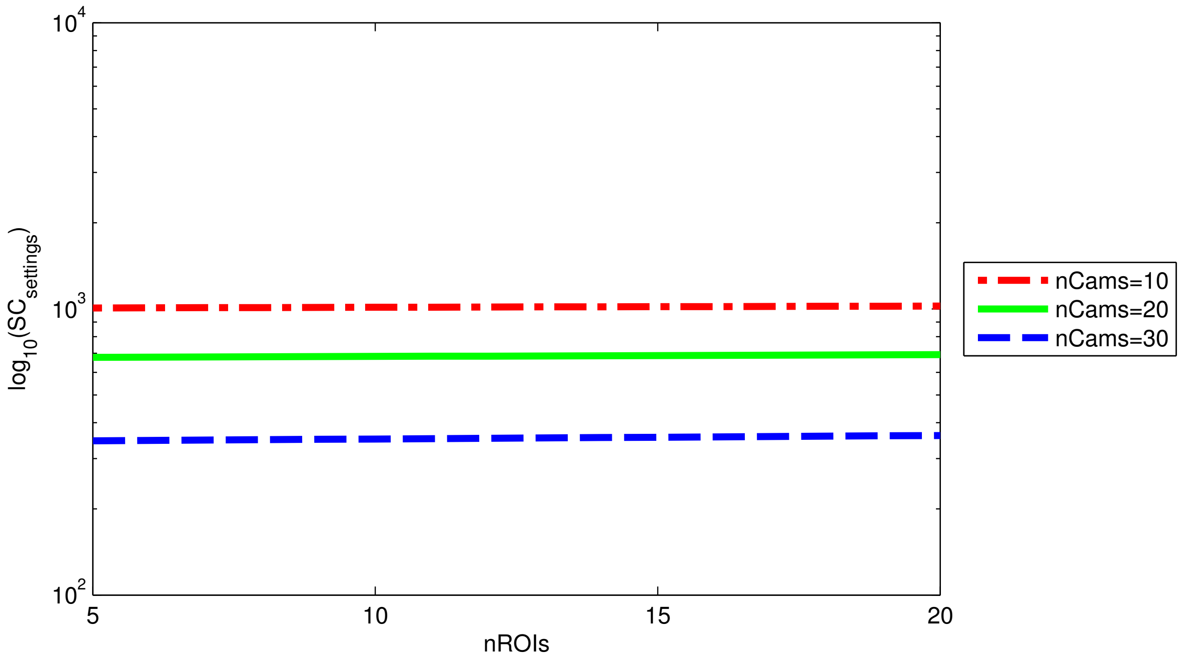


Figure 3.1: Numerical example for exhaustive exploration of the camera settings space. $P = 36$, $T = 9$ and $Z = 5$, $nUnits = 10$, $lCycle = 1$, $nSteps = 2$.

3.4 The Space of Camera to ROI Assignments

Exploring the camera settings space is fruitless when a combination of settings does not lead to the capture of a ROI. If the approximate location of the ROIs is known, this shortcoming can be avoided by reformulating the decision process as follows. Instead of choosing the best camera settings, the decision during each cycle could be which ROI to capture and for how long, in effect *assigning* a camera to a particular ROI for a particular time interval. Given a camera to ROI assignment, the approximate trajectory of the ROI, and a range of possible capture durations, the camera settings corresponding to the best possible contribution of the camera can be computed using geometric reasoning, as I show in Algorithm 6.6 and Figure 6.1.

Under these conditions, the optimization process of finding the best plan is equivalent to assigning cameras to ROIs in combinatorial fashion, computing the camera settings corresponding to all possible capture start cycles and durations, and aggregating the resulting contributions to compute the performance metric value. The number of possible *captures* (start cycles and durations) during a plan step of length $lStep$ is:

$$nCaptures = \sum_{s=1}^{lStep} (lStep - s + 1) = \frac{lStep \cdot (lStep + 1)}{2} \tag{3.4.1}$$

The number of possible assignments to explore during each cycle t for each camera c is:

$$nAssigns_{t,c} = \prod_{r=1}^{nROIs} nCaptures = (nCaptures)^{nROIs} \tag{3.4.2}$$

The number of camera configurations to evaluate during each cycle t for all cameras is:

$$nConfigs_t = \prod_{c=1}^{nCams} nAssigns_{t,c} = \left(nCaptures^{nROIs}\right)^{nCams} \tag{3.4.3}$$

If $nCams < nROIs$, simply assigning cameras to ROIs means that some ROIs are not being covered, which may result in sub-optimal overall performance as measured by the performance metric. This situation is even more likely for 3D reconstruction, because 3D reconstruction algorithms often require more than one camera to cover a ROI in order to obtain results. To alleviate this problem, multiple ROIs can be combined and covered as a single *combined ROI*, at the expense of lower resolution for the projection of each ROI into the camera images. The number of possible *ROI combinations*, computed as the number of non-empty subsets of the set of ROIs, is:

$$nROIcombs = \sum_{k=1}^{nROIs} \binom{nROIs}{k} = 2^{nROIs} - 1 \tag{3.4.4}$$

Replacing *nROIs* with *nROIcombs* in Equation 3.4.3 results in:

$$nConfigs_{assigns} = nCaptures^{nROIcombs \cdot nCams} \tag{3.4.5}$$

Substituting Equation 3.4.5 into the Search Complexity Equation 3.2.5, the total number of contribution computations when exhaustively exploring the camera to ROI assignments space is:

$$SC_{assigns} = \left(nCaptures^{nROIcombs \cdot nCams}\right)^{nSteps} \cdot nCams \cdot nROIs \cdot nUnits \tag{3.4.6}$$

Note that the *nContribs* term defined in Equation 3.2.4 stays the same: even though a camera is specifically assigned to a combined ROI, its contribution is still evaluated for each individual ROI.

To illustrate the search complexity of exhaustively exploring this search space with a concrete numerical example, I set the same conditions as in the example in Section 3.3: $nUnits = 10$, $lCycle = 1$, $nSteps = 2$, $nCams \in \{10, 20, 30\}$, $nROIs = 5 \ldots 20$. The number of captures is $nCaptures = {}^{5\cdot(5+1)}/_{2} = 15$ possible captures during each step. The number of ROI combinations is $nROIcombs = 2^5 - 1 = 31$. Figure 3.2 shows a plot of the logarithm of the search complexity $log_{10}(SC_{assigns})$ in terms of computed contributions, on a logarithmic scale.

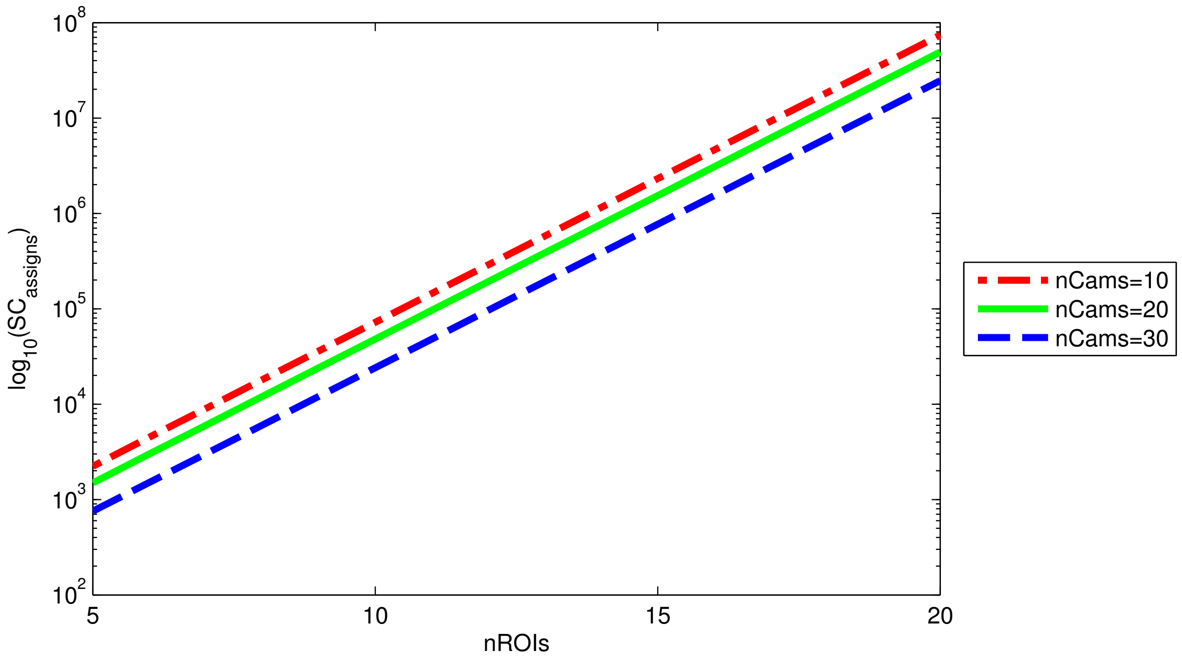


Figure 3.2: Numerical example for exhaustive exploration of the camera to ROI assignments space.
$nUnits = 10$, $lCycle = 1$, $nSteps = 2$.

3.5 Computational Complexity

Equations 3.3.3 and 3.4.6 both have *nCams* and *nSteps* at the exponent, so in terms of the number of cameras and the number of planning steps, exhaustive exploration of both search spaces presented in this chapter takes exponential time because the search space size is exponential. Equation 3.4.6 has $nROIcombs = 2^{nROIs} - 1$ at the exponent, so exhaustive exploration of the camera to ROI assignment space is worse than exponential, since the exponent *nROIcombs* itself is an exponential function in terms of the number of ROIs *nROIs*. Consequently, exhaustive exploration of either search space is intractable for non-trivial problems.

Chapter 4

Making Exploration Tractable

In Chapter 3, I have shown that exhaustively exploring the search spaces of camera settings and camera to ROI assignments is intractable. *Rules* and *heuristics* are often employed to reduce search complexity by eliminating parts of the search space without exploring them. Rules offer provable guarantees that the optimal solution is not missed, while heuristics usually offer much weaker guarantees, if any. This chapter presents a selective exploration rule and several heuristics: lazy planning, myopic planning, grouping ROIs, decomposing the problem into subproblems, and combining the ROIs inside a subproblem.

4.1 Exploring the Space of Camera Settings

In this section, I present several strategies for reducing the search complexity of exploring the space of camera settings.

I illustrate the search complexity of these strategies with concrete numerical examples. For all the examples in this section, I use similar conditions as in the plots shown in Chapter 3. The numbers of possible values for the camera settings are $P = 36$, $T = 9$ and $Z = 5$. The time horizon is $nUnits = 10$, the length of a cycle is $lCycle = 1$, there are only $nSteps = 2$ steps before the planning horizon, and the average length of a plan step is $lStep = 5$ cycles. From Equation 3.3.1, $nSettings_c = 1620^5$. The number of cameras is $nCams \in \{10, 20, 30\}$, and the number of ROIs $nROIs$ varies between 5 and 20. Because for some values of $nCams$ and *nROIs*, the search complexity SC is too large to be represented in MatLab, I plot $log_{10}(SC)$ instead, and use a logarithmic scale to compress the vertical dimension.

4.1.1 Selective Exploration

Domain-specific knowledge can be used to reduce the search complexity by reducing the search space size. For example, instead of searching the entire space of camera settings, the entire pan and tilt range can be covered at the smallest zoom, and further exploration of smaller fields of view by zooming in can be done only where ROIs are in view. Additionally, the number of metric evaluations can be decreased by keeping the overlap between camera views to a minimum. The pan and tilt settings values are in a spherical coordinate system. To achieve minimum overlap between views, the number of pan setting values explored can decrease as the tilt value increases: for a constant field of view, fewer views with minimal overlap are required as the tilt value increases from the sphere's equator toward its poles. The number of camera views with minimum overlap can be approximated using solid angles inside a unit sphere. Let P_{max} and P_{min} be the maximum and minimum pan values. Let T_{max} and T_{min} be the maximum and minimum tilt values. Let Z_{max} and $Z_{min} = 1$ be the maximum and minimum zoom values. Let ϕ and θ be the horizontal and vertical field of view angles at the lowest zoom.

The solid angle corresponding to the part of the sphere covered by the camera over the entire range of pan and tilt settings, at the lowest zoom, can be computed as:

$$A_S = (sin(T_{max} + \theta/2) - sin(T_{min} - \theta/2))(P_{max} - P_{min} + \phi) \tag{4.1.1}$$

The solid angle inside the field of view pyramid with apex angles ϕ and θ can be computed as:

$$A_P = 4\,arcsin(sin(\phi/2)\,sin(\theta/2)) \tag{4.1.2}$$

For each camera c, the number of camera views with minimum overlap at the lowest zoom can be approximated as:

$$nViews_c = \frac{A_S}{A_P} \tag{4.1.3}$$

Let G_P, G_T and G_Z be the granularities of the pan, tilt and zoom settings, or the differences between two successive setting values. Each of the $nViews_c$ views can be explored hierarchically and selectively, starting with the lowest zoom and zooming in only in camera views where there are ROIs. Maintaining minimum overlap between camera views while zooming can be performed in a binary partitioning fashion by doubling the zoom value at every binary tree depth d, corresponding to zoom $\times 2^d$. Each *parent* camera view from depth d gets split into 4 *child* camera views at depth $d+1$, up to a maximum depth $D = log_2(Z_{max})$. In the ideal case, this results in an increased metric value for 1 of the 4 child camera views due to the increased image resolution, and zero for the metric values corresponding to the other 3 child camera views. Only the child camera view with the increased value needs to be explored by becoming a parent for further splitting, while the other 3 child camera views do not need to be explored. When the metric value decreases (zoomed in too far), or more than 1 of the 4 child camera views have non-zero metric values (the ROIs are split between camera views), a local optimization must be performed to compute the optimal configuration and its corresponding metric value inside the PTZ domain enclosed by the parent camera view. The camera field of view at zoom $\times 2^d$ can be computed as $\phi/2^d$. The number of zoom settings between zoom factor $\times 2^d$ and the maximum zoom value Z_{max} can be computed as $(Z_{max}-2^d)/G_Z$. If the local optimization process is a simple exhaustive exploration, the number of camera configurations explored at depth d by each camera c can be approximated as:

$$nLocal_{c,d} = \left(\frac{\phi}{2^d} \cdot \frac{1}{G_P}\right) \cdot \left(\frac{\theta}{2^d} \cdot \frac{1}{G_T}\right) \cdot \left(\frac{Z_{max} - 2^d}{G_Z}\right) \tag{4.1.4}$$

At depth d, let $f_{c,d}$ be the probability that camera c's view gets explored by splitting, and

$g_{c,d} = 1 - f_{c,d}$ the probability that camera c's view gets explored though local optimization. The number of camera configurations that are evaluated at depth $d \in \{0, 1, \ldots, D\}$ for camera c is:

$$nConfigs_{c,d} = nViews_c \cdot \left(4 + nLocal_{c,d} \cdot g_{c,d}\right) \cdot \prod_{i=1}^{d} (f_{c,i-1}) \tag{4.1.5}$$

The total number of possible configurations for camera c for a time step of length $lStep$ time units is:

$$nConfigs_c = \left(\sum_{d=0}^{D} \left(nConfigs_{c,d}\right) \right)^{lStep} \tag{4.1.6}$$

When using *nCams* cameras, assuming they are of the same type, the number of possible camera configurations to explore for each time step is:

$$nConfigs_{selective} = \prod_{c=1}^{nCams} nConfigs_c = (nConfigs_c)^{nCams} \tag{4.1.7}$$

Substituting Equations 4.1.7,3.2.1 into the Search Complexity Equation 3.2.5, the total number of contribution computations when selectively exploring the entire camera settings space is:

$$SC_{selective} = (nConfigs_c)^{nCams \cdot nSteps} \cdot nCams \cdot nROIs \cdot nUnits \tag{4.1.8}$$

Equation 4.1.8 shows that selective exploration helps reduce search complexity by reducing the base of the exponential function, $nConfigs_c$. However, the computational complexity remains exponential in terms of both *nCams* and $nSteps$. A special case occurs if at the lowest zoom some ROIs are split between multiple camera views. This situation can be handled by local exhaustive exploration. The computation in Equation 4.1.8 ignores this special case.

To enable comparison with the exhaustive exploration numerical example in Section 3.3, I set the corresponding values for the settings granularities to $G_P = G_T = 10$, and assume the maximum zoom setting value is $\times 20$, with the corresponding granularity $G_Z = 4$ and the maximum depth $D = 4$. I set the probabilities $f_{c,d} = g_{c,d} = 0.5$ for depths $d = 0, 1, 2, 3$. Finally, $f_{c,4} = 0$ and $g_{c,4} = 1$ to ensure local exhaustive exploration at the highest level of zoom. I also set $P_{min} = T_{min} = 0^\circ$, $P_{max} = 360^\circ$, $T_{max} = 90^\circ$, $\phi = 80^\circ$ and $\theta = 60^\circ$. Using Equation 4.1.3, the number of camera views with minimum overlap at the lowest zoom is $nViews_c \simeq 8$. The number of possible configurations for a single camera c is $nConfigs_c \simeq 1095$. Figure 3.1 shows a plot of the selective search complexity in terms of computed contributions.

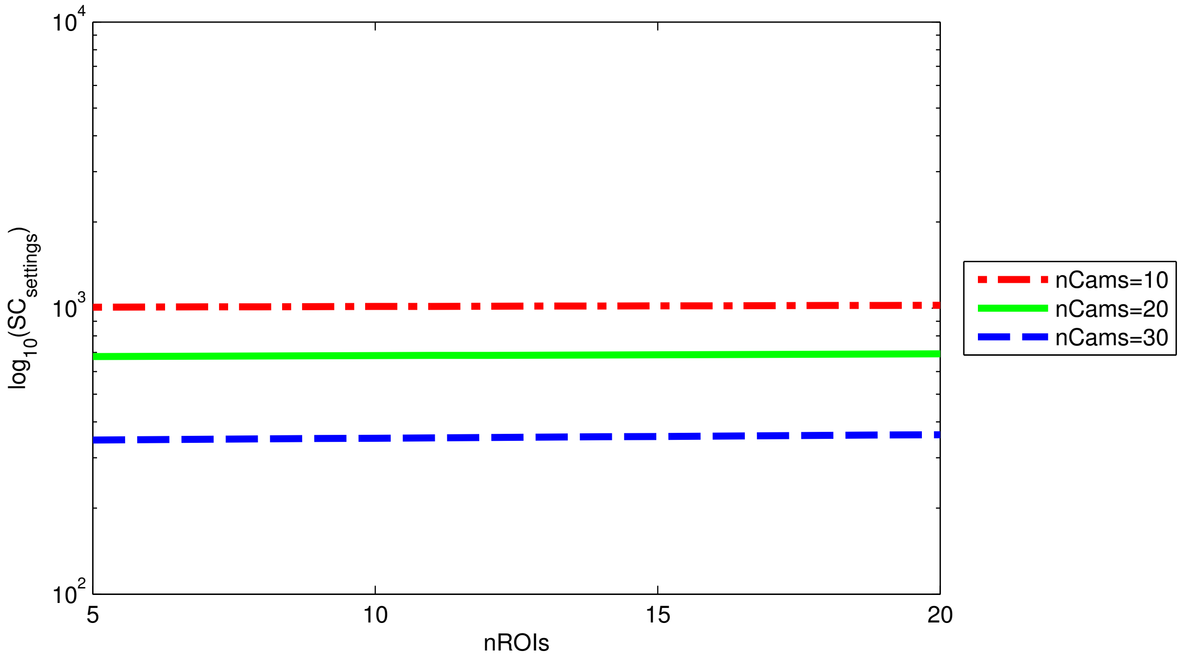


Figure 4.1: Numerical example for selective exploration of the camera settings space. Setting values are the same as in Figure 3.1.

4.1.2 Lazy Planning

During each evaluation cycle, a decision is made whether to keep the current camera settings or to switch to a new configuration. In Section 3.3, the exhaustive exploration process evaluates these decisions for all the cycles up to the planning horizon. One way to reduce search complexity is to have the planning process be *lazy*, and only make this decision for the first cycle in a planning step. Subsequent cycles will simply leave in place the result of the decision made during the first cycle. The reasoning behind this heuristic is that the predicted trajectories available during subsequent cycles are less precise, and the planning process will get a chance to make a decision for each cycle when it becomes the current cycle, and is provided with the most recent trajectory estimates. The number of possible camera settings combinations for each camera c during a plan step becomes (Cf. Equation 3.3.1):

$$nSettings_c = P \cdot T \cdot Z \tag{4.1.9}$$

When using $nCams$ cameras, assuming they are of the same type, the number of possible camera configurations to explore for each plan step becomes (Cf. Equation 3.3.2):

$$nConfigs_{settings:lazy} = \prod_{c=1}^{nCams} nSettings_c = (P \cdot T \cdot Z)^{nCams} \tag{4.1.10}$$

Substituting Equations 4.1.10, 3.2.1 and 3.3.2 into the Search Complexity Equation 3.2.5, the total number of contribution computations becomes (Cf. Equation 3.3.3):

$$SC_{settings:lazy} = \left((P \cdot T \cdot Z)^{nCams}\right)^{nSteps} \cdot nCams \cdot nROIs \cdot nUnits \tag{4.1.11}$$

Equation 4.1.11 shows that lazy exploration helps reduce search complexity by removing $lStep$ from the exponent. However, the computational complexity remains exponential in terms of both $nCams$ and $nSteps$.

Figure 4.2 shows a plot of the lazy search complexity in terms of computed contributions. The settings are the same as in Section 3.3.

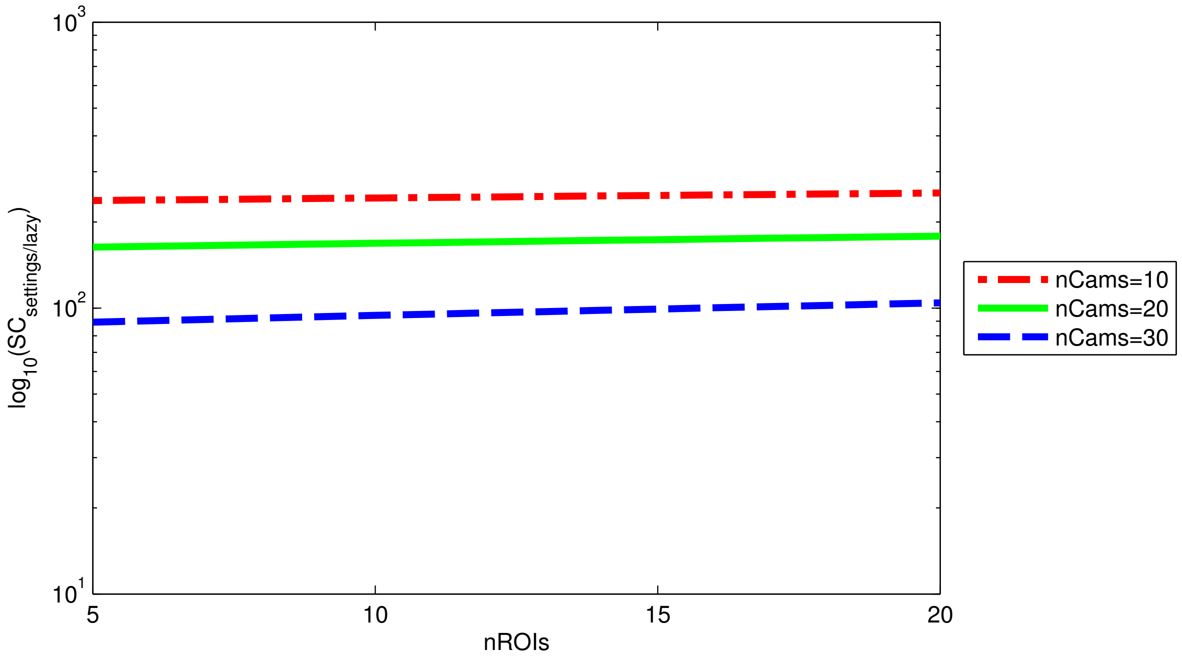


Figure 4.2: Numerical example for lazy exploration of the camera settings space. Setting values are the same as in Figure 3.1.

4.1.3 Myopic Planning

The current configuration of each camera, corresponding to the first step in a plan, is arguably the most important: configurations corresponding to subsequent steps are always revised in subsequent optimization cycles. This enables the planning method to take advantage of the most recent information in predicting the ROI trajectories. Consequently, the optimization at steps greater than 1 can be less precise. The planning process can be *myopic*: exhaustive optimization during the first time step, and heuristic computations for subsequent steps up to the time horizon.

Myopic planning assumes that only the first camera dwell is interruptible. The entire space of camera configurations is only explored for the first planning step, resulting in dwell durations corresponding to each explored configuration. During subsequent steps, pan, tilt and zoom settings are decided heuristically given the ROI trajectories, and result in dwells that are assumed uninterruptible.

Myopic planning transforms the decision tree by replacing all the decisions at depth greater than 1 with heuristics. The transformed decision tree only has branches at the first step. If the number of configurations to evaluate at step 1 is $nConfigs$, the number of configurations to evaluate at steps greater than 1 is also $nConfigs$, as each configuration from step 1 results in a single heuristic configuration at subsequent steps. The number of plans to evaluate becomes:

$$nPlans_{myopic} = \sum_{s=1}^{nSteps} (nConfigs) = nConfigs \cdot nSteps \tag{4.1.12}$$

Substituting Equations 4.1.12, 3.2.1 and 3.3.2 into the Search Complexity Equation 3.2.5, the total number of contribution computations becomes (Cf. Equation 3.3.3):

$$SC_{settings:myopic} = \left((P \cdot T \cdot Z)^{lStep \cdot nCams} \cdot nSteps \right) \cdot nCams \cdot nROIs \cdot nUnits \tag{4.1.13}$$

Equation 4.1.13 shows that lazy exploration helps reduce search complexity by removing $nSteps$ from the exponent and making it a simple term in the product. The computational complexity in terms of $nSteps$ is now polynomial, but remains exponential in terms of $nCams$.

Figure 4.3 shows a plot of the myopic search complexity in terms of computed contributions. The settings are the same as in Section 3.3.

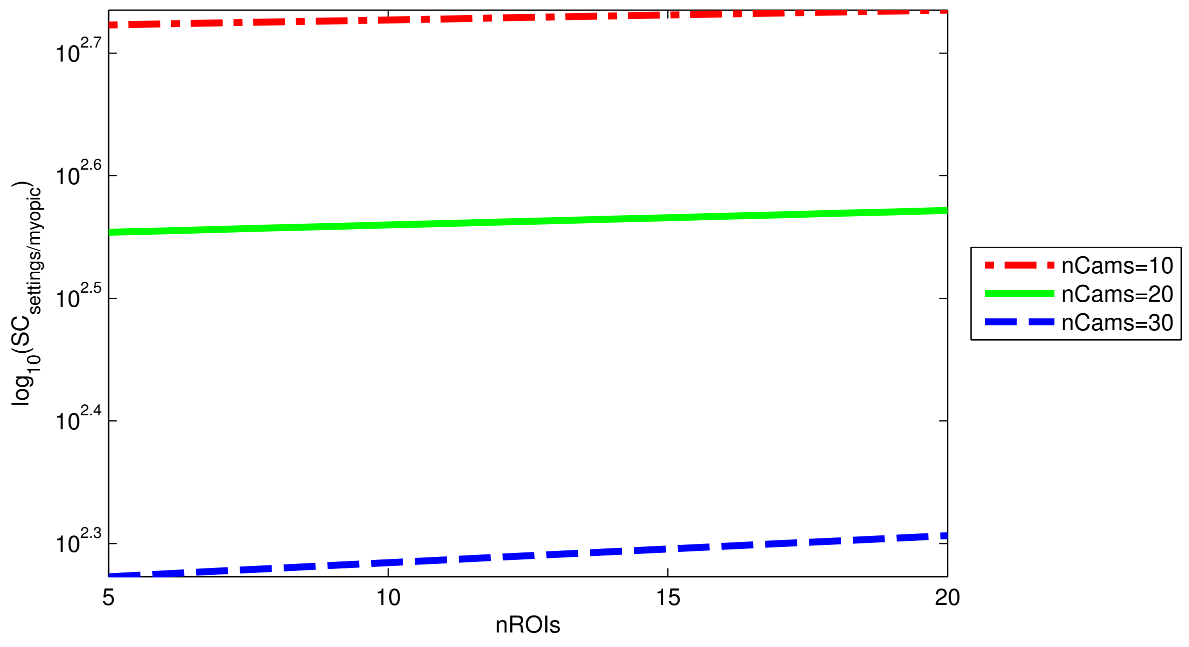


Figure 4.3: Numerical example for myopic exploration of the camera settings space. Setting values are the same as in Figure 3.1.

4.1.4 Independent Planning

In distributed approaches, each camera agent makes its own decisions. If the configuration of each camera is explored independently, the number of configurations for all cameras becomes:

$$nConfigs_{settings:indep.} = \sum_{c=1}^{nCams} nSettings_c = (P \cdot T \cdot Z)^{lStep} \cdot nCams \qquad (4.1.14)$$

Substituting Equation 4.1.14 into the Search Complexity Equation 3.2.5, the total number of contribution computations becomes:

$$SC_{settings:indep.} = \left((P \cdot T \cdot Z)^{lStep} \cdot nCams\right)^{nSteps} \cdot nCams \cdot nROIs \cdot nUnits \qquad (4.1.15)$$

Equation 4.1.15 shows that independent exploration helps reduce search complexity by moving *nCams* from the exponent to the base of the exponential function. The computational complexity in terms of *nCams* is now polynomial, but remains exponential in terms of *nSteps*.

Figure 4.4 shows a plot of the independent search complexity in terms of computed contributions. The settings are the same as in Section 3.3.

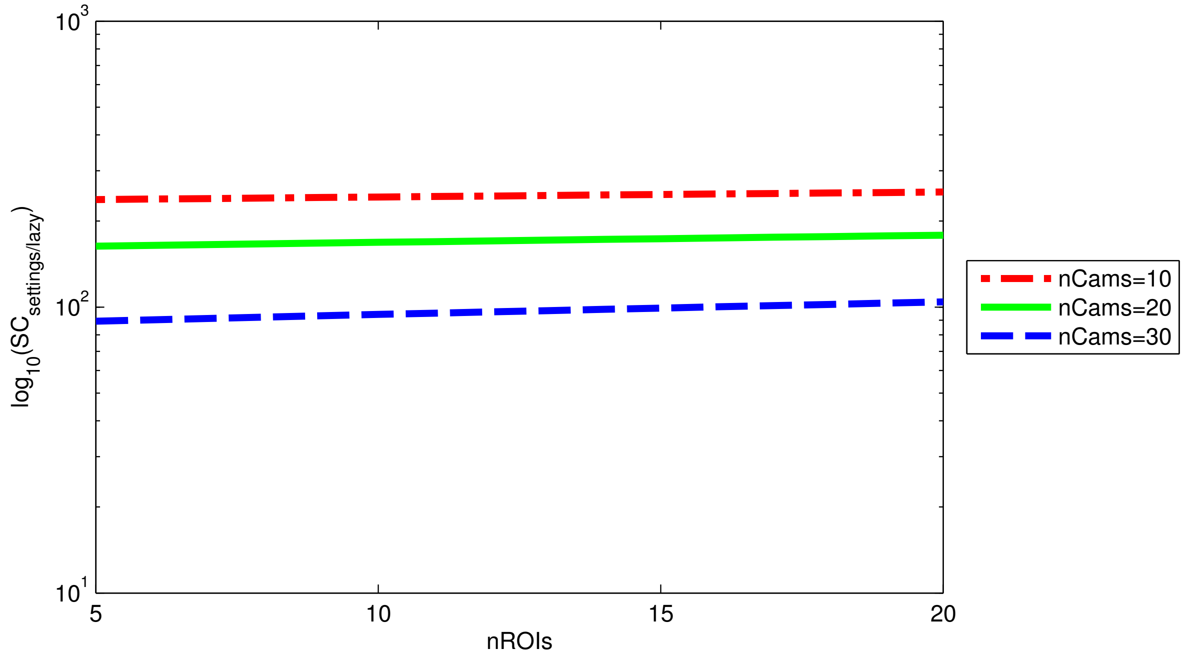


Figure 4.4: Numerical example for independent exploration of the camera settings space. Setting values are the same as in Figure 3.1.

4.2 Exploring the Space of Camera to ROI Assignments

In this section, I present a few strategies that can be employed to reduce the search complexity of exploring the space of camera to ROI assignments.

I illustrate the search complexity of each strategy with concrete numerical examples. For all the examples in this section, I set the time horizon to $nUnits = 10$, the length of a cycle to $lCycle = 1$, and assume there are only $nSteps = 2$ steps before the planning horizon, resulting in the average length of a plan step $lStep = 5$ cycles and $nCaptures = {}^{5\cdot(5+1)}/_{2} = 15$ possible captures during each step. I set the number of cameras to $nCams \in \{10, 20, 30\}$, and vary the number of ROIs $nROIs$ between 5 and 20. The number of ROI combinations is $nROIcombs =$

$2^5 - 1 = 31$.

4.2.1 Lazy Planning

The lazy planning heuristic described in Section 4.1.2 can also be applied to exploring the space of camera to ROI assignments. When only exploring alternatives during the first cycle in a plan step, the number of possible captures (start cycles and durations) during a plan step of length $lStep$ becomes (Cf. Equation 3.4.1):

$$nCaptures = lStep \tag{4.2.1}$$

Substituting Equations 4.2.1 and 3.2.1 into the Search Complexity Equation 3.2.5, the total number of contribution computations becomes (Cf. Equation 3.3.3):

$$SC_{assigns:lazy} = \left(lStep^{nROIcombs \cdot nCams}\right)^{nSteps} \cdot nCams \cdot nROIs \cdot nUnits \tag{4.2.2}$$

Equation 4.2.2 shows that lazy exploration helps reduce search complexity by reducing the base of the exponential from $nCaptures$ to $lStep$. However, the computational complexity remains exponential in terms of both $nCams$ and $nSteps$.

Figure 4.5 shows a plot of the lazy search complexity in terms of computed contributions. The settings are the same as in Section 3.4.

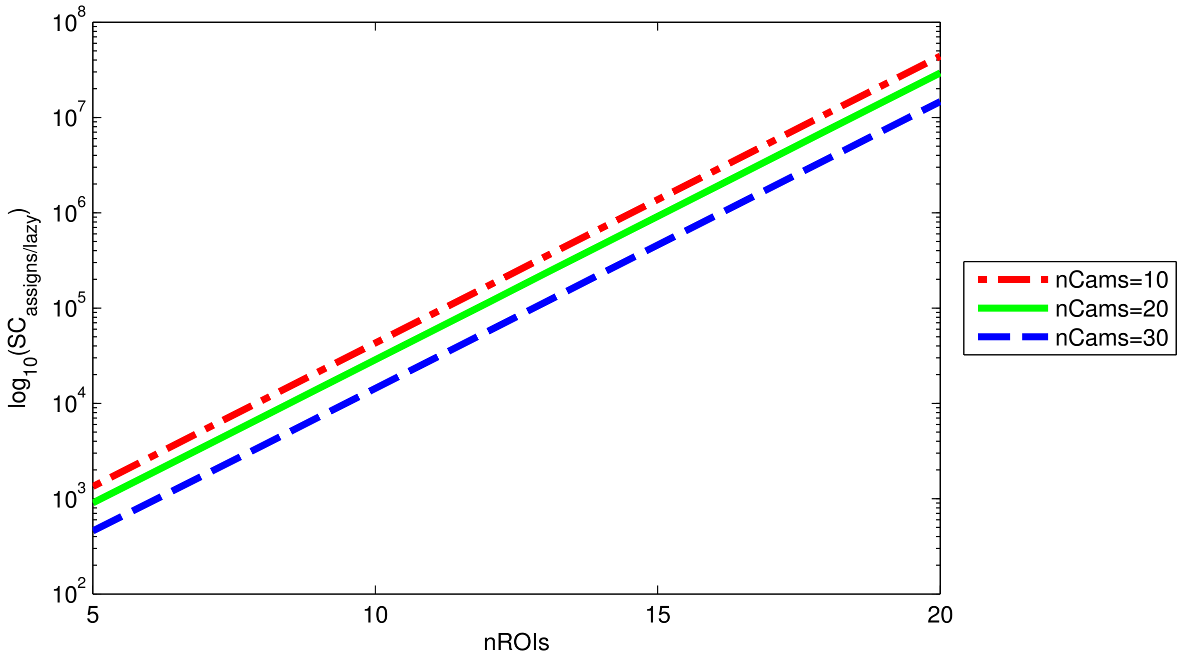


Figure 4.5: Numerical example for lazy exploration of the camera to ROI assignments space. Setting values are the same as in Figure 3.2.

4.2.2 Myopic Planning

The myopic planning heuristic described in Section 4.1.3 can also be applied to exploring the space of camera to ROI assignments. The number of plans to evaluate *nPlans* is the same as in Equation 4.1.12:

$$nPlans_{myopic} = \sum_{s=1}^{nSteps} (nConfigs) = nConfigs \cdot nSteps \tag{4.2.3}$$

Substituting Equations 4.2.3, 3.2.1 and 3.3.2 into the Search Complexity Equation 3.2.5, the total number of contribution computations becomes (Cf. Equation 3.3.3):

$$SC_{assigns:myopic} = \left(nCaptures^{nROIcombs \cdot nCams} \cdot nSteps\right) \cdot nCams \cdot nROIs \cdot nUnits \tag{4.2.4}$$

Just as in Section 4.1.3, Equation 4.2.4 shows that myoic exploration helps reduce search complexity by removing *nSteps* from the exponent and making it a simple term in the product. The computational complexity in terms of *nSteps* is now polynomial, but remains exponential in terms of *nCams*.

Figure 4.6 shows a plot of the myopic search complexity in terms of computed contributions. The settings are the same as in Section 3.4.

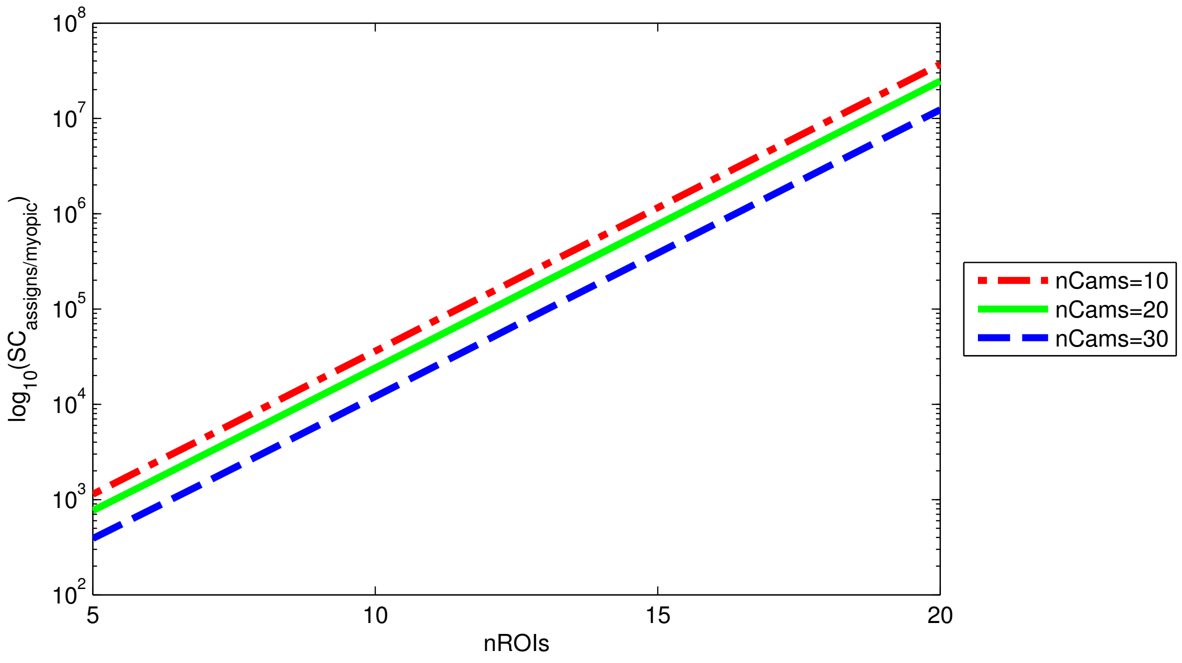


Figure 4.6: Numerical example for myopic exploration of the camera to ROI assignments space.
Setting values are the same as in Figure 3.2.

4.2.3 Independent Planning

Independent planning as described in Section 4.1.4 can also apply in the space of camera to ROI assignments. The number of camera configurations to evaluate during each cycle t for all cameras is:

$$nConfigs_t = \sum_{c=1}^{nCams} nAssigns_{t,c} = \left(nCaptures^{nROIcombs}\right) \cdot nCams \qquad (4.2.5)$$

Substituting Equation 4.2.5 into the Search Complexity Equation 3.2.5, the total number of contribution computations when exhaustively exploring the camera to ROI assignments space is:

$$SC_{assigns:indep.} = \left(nCaptures^{nROIcombs} \cdot nCams\right)^{nSteps} \cdot nCams \cdot nROIs \cdot nUnits \qquad (4.2.6)$$

Just as in Section 4.1.4, Equation 4.2.6 shows that independent exploration helps reduce search complexity by moving *nCams* from the exponent to the base of the exponential function. The computational complexity in terms of *nCams* is now polynomial, but remains exponential in terms of $nSteps$.

Figure 4.7 shows a plot of the independent search complexity in terms of computed contributions. The settings are the same as in Section 3.4.

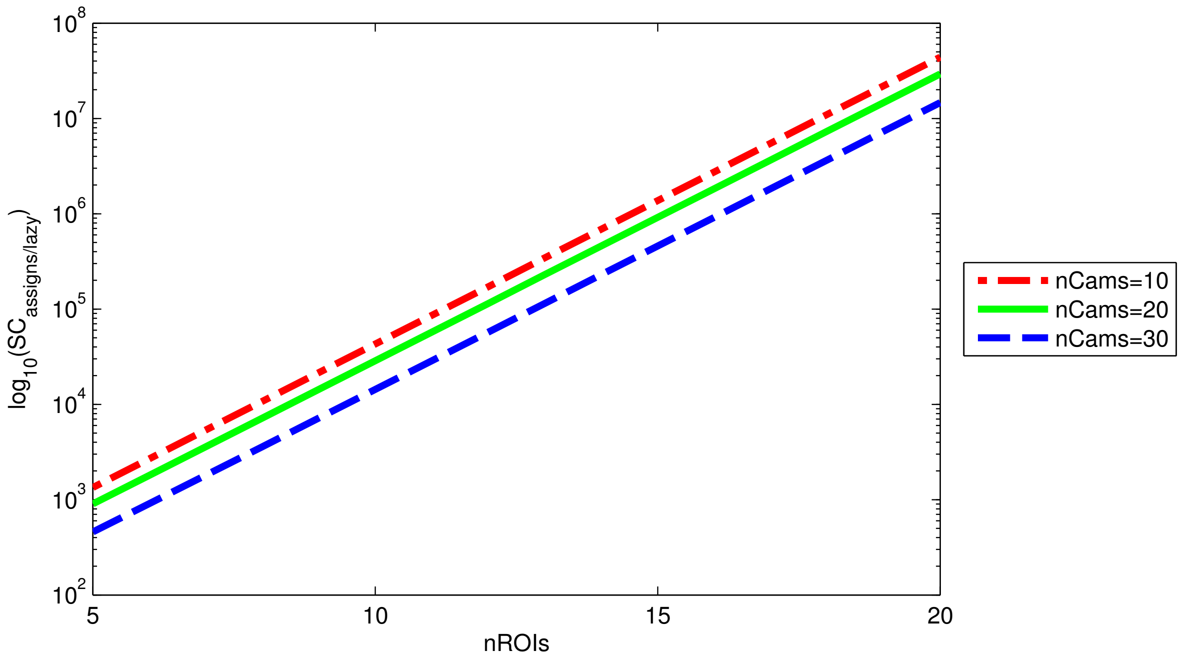


Figure 4.7: Numerical example for independent exploration of the camera to ROI assignments space.
Setting values are the same as in Figure 3.2.

4.2.4 Grouping ROIs

One of the ways to reduce $SC_{assigns}$ in Equation 3.4.6, is to reduce the exponent of $nCaptures$. One of the terms in the exponent is $nROIcombs = 2^{nROIs} - 1$, which reflects the exponential number of possible ways to combine ROIs, defined in Equation 3.4.4. Combining ROIs is necessary to alleviate the problem of assigning a typically insufficient number of cameras to a potentially large number of individual ROIs. However, not all possible ROI combinations are useful: some ROIs are so far apart that trying to cover them simultaneously with a camera would require zooming out either beyond the capabilities of the camera or at the cost of decreased resolution of the ROIs in the camera image. Additionally, some ROI combinations can include others: combining two ROIs will lead to cameras covering any other ROI located between them, so these other ROIs should be included in the initial ROI combination. Instead of generating useless ROI combinations and using the performance metric to determine that they lead to worse performance than others, *proximity* can be used to group ROIs. Each proximity-based ROI group can be treated as an individual ROI, and further combined with other ROI groups or individual ROIs. A group can contain as few as a single ROI, or as many as all the ROIs.

Let $nGroups$ be the number of ROI groups formed by proximity, with $nGroups \ll nROIs$, replacing $nROIs$ in all formulas in Section 4.2.

The number of combined ROIs, becomes (Cf. Equation 3.4.4):

$$nROIcombs_{groups} = \sum_{k=1}^{nGroups} \binom{nGroups}{k} = 2^{nGroups} - 1 \simeq 2^{nGroups} \qquad (4.2.7)$$

Note that $nROIcombs_{groups} \ll nROIcombs$, because $nGroups \ll nROIs$.

Equation 3.4.6 becomes:

$$SC_{groups} = \left(nCaptures^{nROIcombs_{groups} \cdot nCams}\right)^{nSteps} \cdot nCams \cdot nGroups \cdot nUnits \qquad (4.2.8)$$

Comparing the Combined ROI Equation 3.4.6 with the Groups Equation 4.2.8, the benefits of grouping are apparent: the number of contribution computations decreases because the exponent decreases. However, grouping using proximity as a criterion is a greedy heuristic, so it may result in not finding the true global optimal camera configuration. The computational complexity remains exponential in terms of both *nCams* and *nSteps*. The number of groups *nGroups* is the expression of the flexibility of this heuristic: groups can be generated based on other criteria besides proximity. The grouping process can be made part of the optimization process if camera configurations corresponding to the groups generated during grouping are evaluated during the optimization.

For a numerical example, I set the number of ROI groups to $nGroups = \lceil nROIs/5 \rceil$. Figure 4.8 shows a plot of the search complexity in terms of computed contributions. The settings are the same as in Section 3.4.

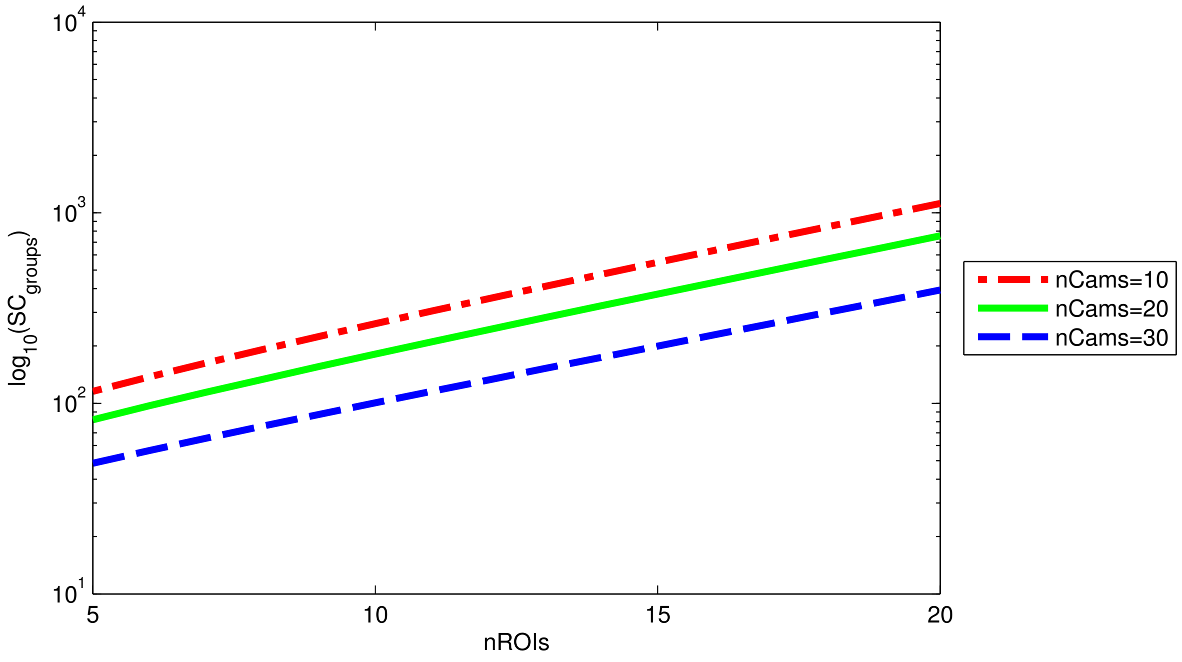


Figure 4.8: Numerical example for the exploration of the camera to grouped ROI assignments space.
Setting values are the same as in Figure 3.2, and $nGroups = \lceil nROIs/5 \rceil$.

4.2.5 Decomposing the Problem into Subproblems

In Section 4.2.4, I showed that using proximity to group ROIs reduces the number of ROI combinations *nROIcombs*. However, all possible combinations of ROI groups are still explored, even if they are far from each other, and impossible to cover even at the widest field of view. Additionally, all possible assignments of cameras to ROI combinations are still explored, even if a camera is very far from the ROI group combination it was assigned to. As I show in Sections 5.4.6 and 5.4.8, such assignments usually result in worse performance as measured by the performance metric. To avoid these under-performing assignments and to speed up computation, a greedy approach based on proximity and other heuristics can be used to assign each ROI group combination to cameras that are close by. Once these greedy assignments are made, the problem remains to decide which of the cameras assigned to a ROI group combination get further assigned to which ROI group inside each group combination. These local assignment *subproblems* are simply smaller scale versions of the global assignment problem.

Let *nSubProbs* be the total number of subproblems that can be formed using proximity. Let each subproblem p contain $nROIs_p$ ROIs and be covered by $nCams_p$ cameras closest to it.

The number of possible combined ROIs, computed as the number of non-empty subsets of the set of ROIs by applying Equation 3.4.4 at the level of each group, is:

$$nROIcombs_p = \sum_{k=1}^{nROIs_p} \binom{nROIs_p}{k} = 2^{nROIs_p} - 1 \tag{4.2.9}$$

Under the simplifying assumption that a camera only contributes to the ROIs inside the subproblem it has been assigned to, *nROIs* can be replaced by $nROIs_p$ in Equation 3.2.4:

$$nContribs_p = nCams_p \cdot nROIs_p \cdot nUnits \tag{4.2.10}$$

The number of contributions to evaluate inside each subproblem can be computed by applying Equation 3.4.5 at the level of each subproblem:

$$nConfigs_p = nCaptures^{nROIcombs_p \cdot nCams_p} \simeq nCaptures^{2^{nROIs_p} \cdot nCams_p} \tag{4.2.11}$$

The search complexity in terms of contribution computations can be computed by applying the Combined ROI Equation 3.4.6 at the level of each subproblem p and summing the results:

$$SC_{subprobs} = \sum_{p=1}^{nSubProbs} \left(\left(nCaptures^{nROIcombs_p \cdot nCams_p} \right)^{nSteps} \cdot nCams_p \cdot nROIs_p \cdot nUnits \right) \tag{4.2.12}$$

The dominant term in the Subproblems Equation 4.2.12 is the additive term corresponding to the subproblem with the largest exponent $nROIcombs_p \cdot nCams_p$. This term is minimized when all subproblems are of the same size: $nROIs_p = nROIs/nSubProbs$ and $nCams_p = nCams/nSubProbs$. Under the assumption that optimizing the performance metric in each group leads to the overall optimum, the optimization process can be performed independently at the level of each group.

Comparing the Combined ROI Equation 3.4.6 with the Subproblems Equation 4.2.12, the benefits of decomposing the problem into subproblems are apparent: the number of contribution computations decreases, and the optimization process can be run independently for each group, in parallel. However, decomposing the problem into subproblems can lead to a suboptimal solution, because assignment decisions are not made through optimization. The merits of decomposing the problem into subproblems should be judged on whether the risk of not finding the global optimum is offset by the benefit of being able to explore a smaller search space in less time.

The following figures show plots of the search complexity in terms of computed contributions.

For a first example, corresponding to the situation where subproblems are assigned to a constant number of available processors, I chose a constant number of subproblems $nSubProbs = 8$. Figure 4.9 shows a plot of the number of computed contributions $SC_{subprobs}$, given by the

Subproblems Equation 4.2.12, for 8 subproblems. I set $nROIs_p = \lceil {}^{nROIs}/_8 \rceil$ and $nCams_p = \lceil {}^{nCams}/_8 \rceil$. Note the staircase look for $nROIs \in \{8, 16\}$, due to the use of the *ceil* function to determine $nCams_p$ and $nROIs_p$.

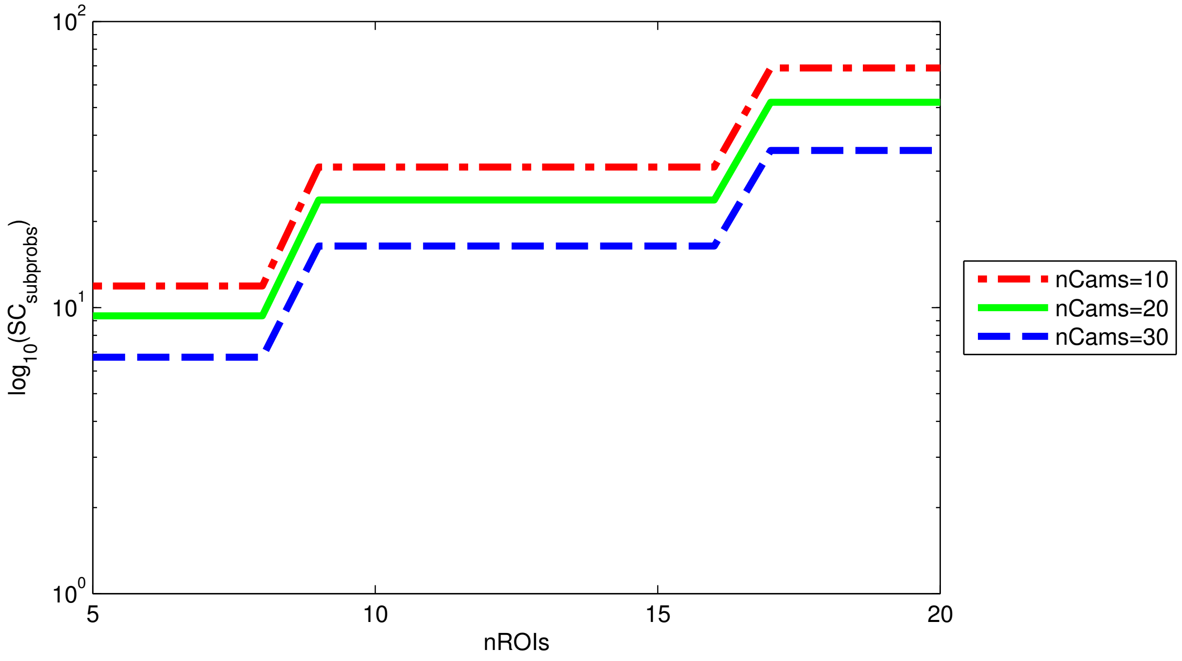


Figure 4.9: Numerical example for optimization using 8 subproblems. Setting values are the same as in Figure 3.2, and $nSubProbs = 8$ subproblems.

As a second example, corresponding to the situation where subproblems are run as threads, I chose to make the number of subproblems *nSubProbs* as large as possible, to spread the computation in as many threads as possible. If *MC* is the minimum number of cameras required for 3D reconstruction, the largest number of subproblems that can be formed while ensuring each subproblem has at least *MC* cameras assigned to it is $nSubProbs = \lceil {}^{nCams}/_{MC} \rceil$. The corresponding values for $nROIs_p$ and $nCams_p$ are $nROIs_p = \lceil {}^{nROIs}/_{nSubProbs} \rceil$ and $nCams_p = MC$. Figure 4.10 shows a plot of the logarithm of the number of computed contributions $SC_{subprobs}$, given by the Groups Equation 4.2.12, for a varying number of subproblems. I chose the minimum number of cameras required for 3D reconstruction $MC = 2$.

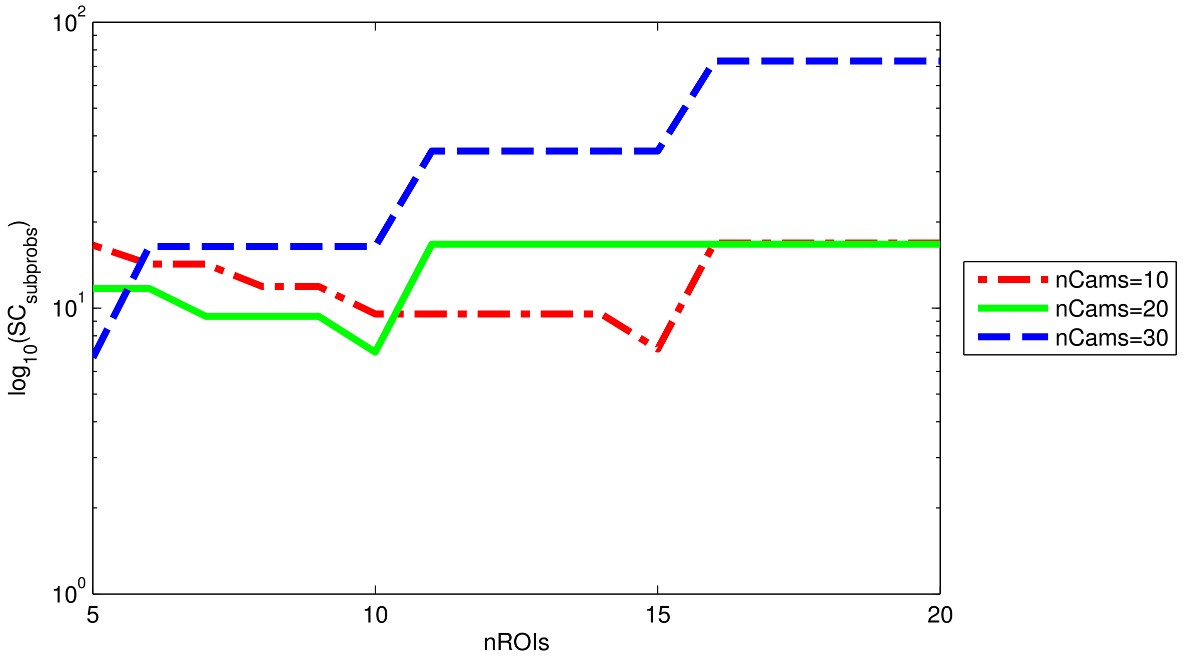


Figure 4.10: Numerical example for optimization using a variable number of subproblems. Setting values are the same as in Figure 3.2, and $nSubProbs = \lceil {}^{nCams}/_{2} \rceil$ subproblems.

As the number of cameras *nCams* increases, so does the number of subproblems *nSubProbs*. The number of ROIs in each subproblem $nROIs_p$ decreases, resulting in a smaller overall search complexity. The *ceil* function affects the search complexity in a more complex way, as *nSubProbs* is no longer constant.

Comparing Figure 4.9 with Figure 4.10, in the case of a fixed number of subproblems the search complexity increases with the number of cameras, while in the case of a varying number of subproblems increasing the number of cameras allows creating more subproblems and subdividing the work among them, which results in lower search complexity.

4.2.6 Grouping All ROIs Inside a Subproblem

Taking the proximity grouping heuristic further, ROIs grouped by proximity into a subproblem are close to each other and should be covered together. At the level of each subproblem, it may not be necessary to explore all possible combinations of ROIs $nROIcombs_p$, computed by applying the Combined ROI Equation 4.2.9 at the level of each group as $2^{nROIs_p} - 1$. Instead, all the ROIs can be grouped into a single ROI group, so $nROIs_p = 1$ and $nROIcombs_p = 1$.

Equation 4.2.12 becomes:

$$SC_{subprobs:grouped} = \sum_{p=1}^{nSubProbs} \left(\left(nCaptures^{nCams_p} \right)^{nSteps} \cdot nCams_p \cdot nUnits \right) \qquad (4.2.13)$$

This heuristic reduces the amount of work to be done inside each subproblem by completely eliminating the assignment of cameras to ROIs, since all cameras are assigned to a single ROI group. The only work remaining is the actual planning: deciding the number of planning steps and the duration of each step. The computational complexity remains exponential in terms of the number of steps $nSteps$.

I illustrate the search complexity of decomposing the problem into subproblems and grouping the ROIs inside a subproblem under the same conditions as in Section 4.2.5.

Figure 4.11 shows a plot of the number of computed contributions $SC_{subprobs:grouped}$ for the example corresponding to the situation where subproblems are assigned to a constant number of available processors. Cf. Figure 4.9. Figure 4.12 shows a plot of the logarithm of the number of computed contributions $SC_{subprobs:grouped}$ for the example where subproblems are run as threads. Cf. Figure 4.10.

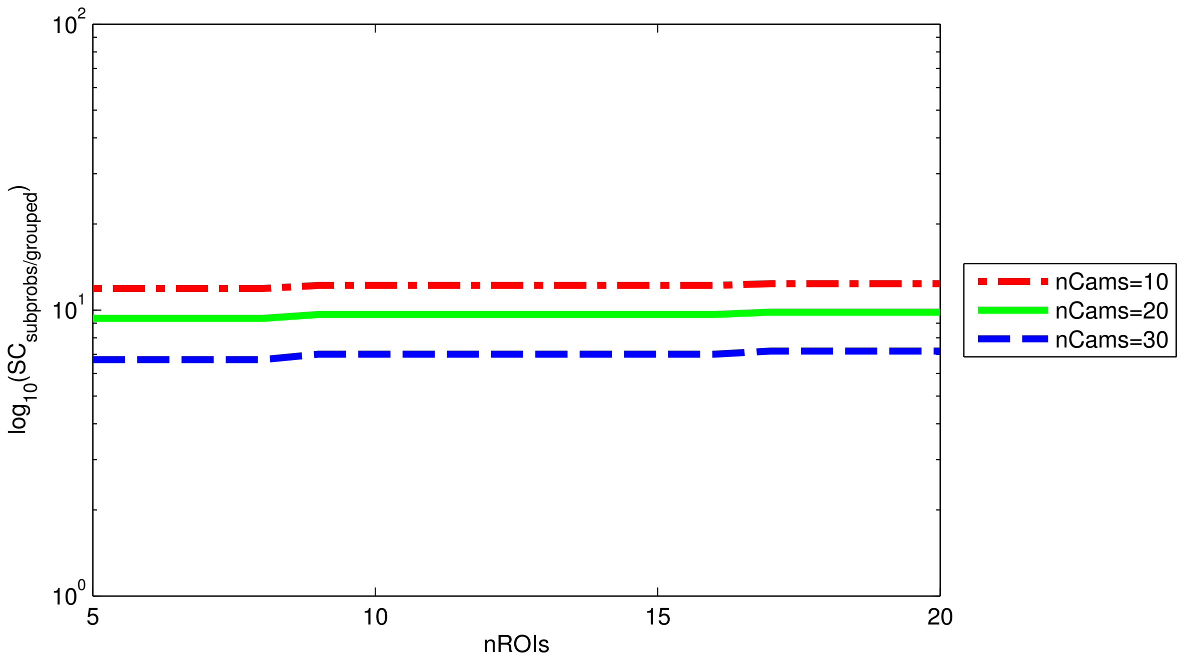


Figure 4.11: Numerical example for optimization using 8 subproblems with grouped ROIs. Setting values are the same as in Figure 3.2, and $nSubProbs = 8$ subproblems.

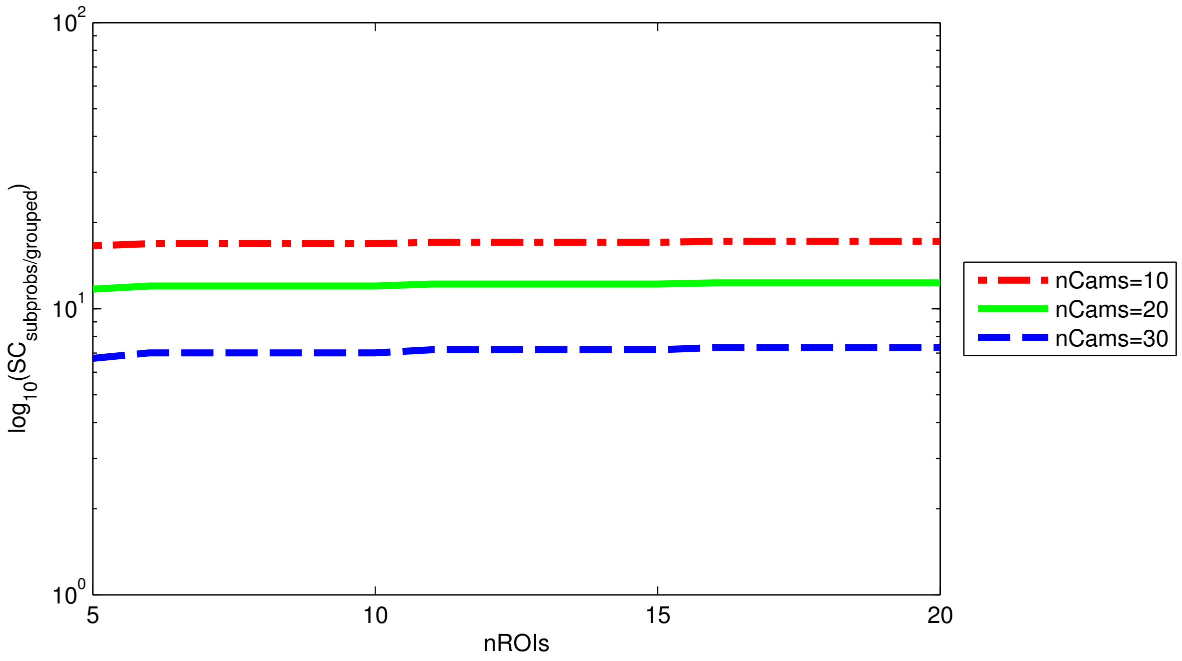


Figure 4.12: Numerical example for optimization using a variable number of subproblems with grouped ROIs.
Setting values are the same as in Figure 3.2, and $nSubProbs = \lceil nCams/2 \rceil$ subproblems.

Because $nROIs_p$ and $nROIcombs_p$ are constant, the complex effect of the *ceil* function in Section 4.2.5 no longer appears when comparing Figures 4.11 and 4.12: the search complexity increases with the number of cameras for both a constant and a varying number of subproblems.

4.3 Tunable Parameters

In addition to the strategies presented above, the search complexity is affected by other factors for which analysis is not straightforward. Nevertheless, their effects are easy to predict. The list below gives a few examples:

- **Planning horizon**: Theoretically, in order to allow comparison between all possible schedules, the planning horizon should be as long as the longest capture possible at the lowest zoom of any camera capturing any ROI. In practice, this can result in very long horizons that can only be explored in real-time using myopic planning, as the planning horizon length is $nUnits = nSteps \cdot lStep \cdot lCycle$, and only for myopic planning the search complexity SC does not have any of these terms in the exponent. Consequently, the planning horizon can be set to the minimum between the longest capture duration and the longest horizon explorable in real-time.

- **Visibility threshold**: Occlusions can be used to reduce the number of possible plans to evaluate. Since cameras do not capture during occlusions and transitions, in order to maximize capture time one strategy could be to plan transitions during times when a camera's view would have been occluded anyway. One important parameter that affects this ability is the visibility threshold. A high visibility threshold will detect more occlusions, and result in fewer possible plans to explore. A lower visibility threshold will increase the number of plans to explore and it may also make plans and captures less fragmented.

- **Planning frequency**: One of the advantages of breaking down the problem into subproblems is the decoupling of the global process of assigning cameras to subproblems and the local process of optimizing the plan for each assigned camera. These processes can run at different frequencies. Running them every cycle ensures that plans adapt quickly to changes in the ROIs, but quick adaptation may also mean the captures may be very fragmented. Running them less often would result in less fragmented plans, but will incur the risk of losing track of some ROIs.

- **Trajectory prediction model parameters**: The accuracy of the predicted ROI trajectories also affects the planning process. Erroneous or uncertain trajectories result in the need to alter the plans, so proper tuning of prediction models is needed to minimize the occurrence of such circumstances.

- **Proximity grouping algorithm**: The way in which ROIs are grouped affects the planning method's ability to plan captures. Also, ROI groups need to be stable over time, to minimize the need for assignment changes that follow group membership changes.

4.4 Comparison

In the previous sections I presented rules and heuristics, and showed their impact on the search complexity with numerical examples. Table 4.1 shows a summary of the rules and heuristics and their impact on a specific numerical example chosen to facilitate comparisons. For the numerical example, I set the time horizon to $nUnits = 10$, the length of a cycle to $lCycle = 1$, and assume there are only $nSteps = 2$ steps before the planning horizon, resulting in the average length of a plan step $lStep = 5$. I also set the number of cameras to $nCams = 10$ and the number of ROIs to $nROIs = 10$. For the space of camera settings, I set the number of possible values for the camera settings to $P = 36$, $T = 9$ and $Z = 5$. For grouping, I set $nGroups = 2$. For decomposing into subproblems, I set $nSubProbs = 5$.

Search Space	Rule / Heuristic	Equation No.	Numerical Example	Comp. in *nCams*	Comp. in *nROIs*	Comp. in *nSteps*
camera settings	exhaustive exploration	3.3.3	$SC_{settings} \simeq 9 \cdot 10^{323}$	exp.	poly.	exp.
	selective exploration	4.1.8	$SC_{selective} \simeq 6 \cdot 10^{63}$	exp.	poly.	exp.
	lazy planning	4.1.11	$SC_{settings:lazy} \simeq 1.6 \cdot 10^{67}$	exp.	poly.	exp.
	myopic planning	4.1.13	$SC_{settings:myopic} \simeq 6 \cdot 10^{163}$	exp.	poly.	poly.
	independent planning	4.1.15	$SC_{settings:indep.} \simeq 1.2 \cdot 10^{45}$	poly.	poly.	exp.
camera to ROI assignments	exhaustive exploration	3.4.6	$SC_{assigns} \simeq 1.5 \cdot 10^{732}$	exp.	>exp.	exp.
	lazy planning	4.2.2	$SC_{assigns:lazy} \simeq 2.3 \cdot 10^{436}$	exp.	>exp.	exp.
	myopic planning	4.2.4	$SC_{assigns:myopic} \simeq 7.8 \cdot 10^{367}$	exp.	>exp.	poly.
	independent planning	4.2.6	$SC_{assigns:indep.} \simeq 8.3 \cdot 10^{137}$	poly.	>exp.	exp.
	grouping ROIs	4.2.8	$SC_{groups} \simeq 3.7 \cdot 10^{73}$	exp.	unknown	exp.
	decomposing into subproblems	4.2.12	$SC_{subprobs} \simeq 2.6 \cdot 10^{16}$	exp.	unknown	exp.
	grouping ROIs inside subproblems	4.2.13	$SC_{subprobs:grouped} \simeq 1 \cdot 10^{7}$	exp.	const.	exp.

Table 4.1: Summary of rules and heuristics, with a numerical example and computational complexities.

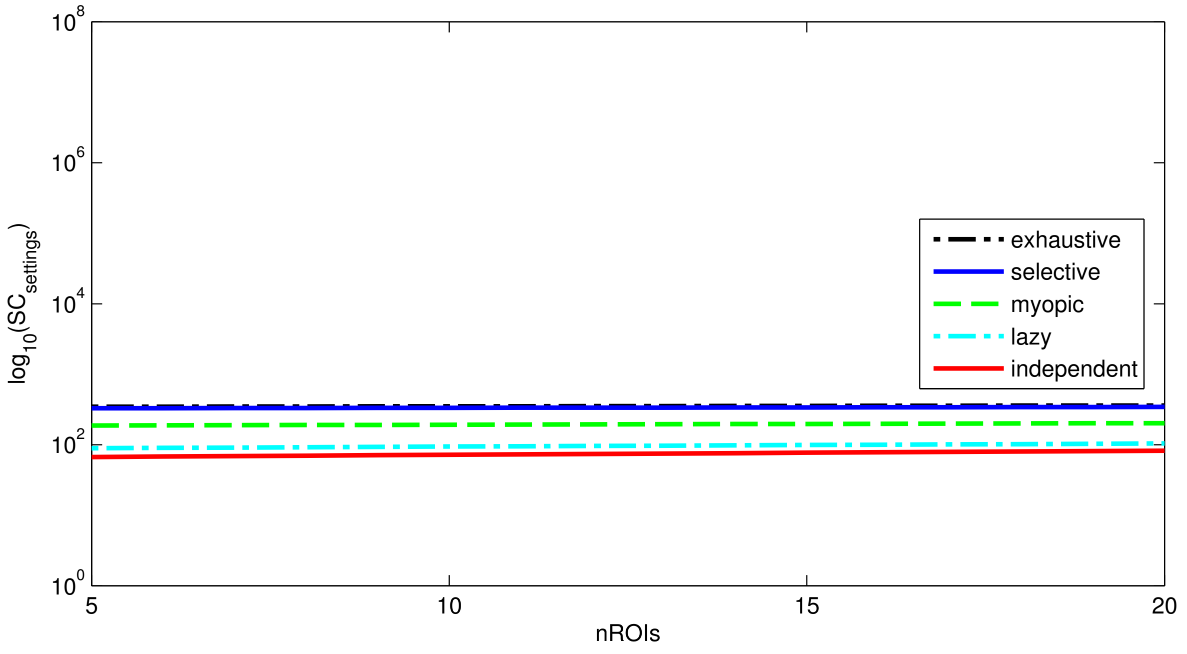


Figure 4.13: Numerical example for comparison between strategies to reduce the search complexity of the camera settings space for $nCams = 10$.

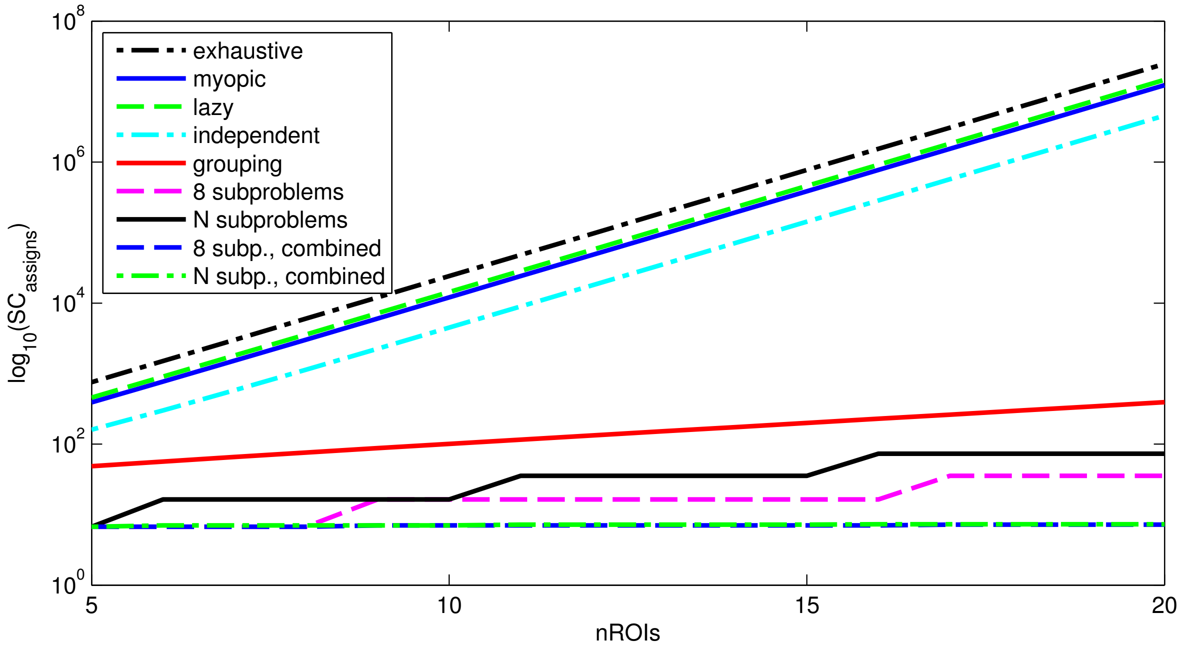


Figure 4.14: Numerical example for comparison between strategies to reduce the search complexity of the camera to ROI assignments space for $nCams = 10$.

Figures 4.13 and 4.14 show comparisons of the effects of all the rules and heuristics in this chapter for $nCams = 10$ on the same vertical scale. The space of camera settings is smaller than the space of camera to ROI assignments, but the effect of rules and heuristics is more pronounced when applied to the latter.

4.5 Complexity in Wall-Clock Time

The values in Table 4.1 and all the numerical examples in this chapter are in terms of the number of atomic contribution computations. To give a sense of how that translates in wall-clock time, I measured the time it took to compute a contribution over 900 trials on a $1.8GHz$ single-core computer under normal load, and averaged them to arrive at a value of approximately 0.2 milliseconds (200 microseconds). That means that the smallest value in Table 4.1, $SC_{subprobs:gtouped} \simeq 1 \cdot 10^7$ contribution computations translates into 2000 seconds. This is how long it would take the $1.8GHz$ single-core computer under normal load to evaluate all the plans in to the space of assignments of 10 cameras to 5 subproblems with grouped ROIs over a planning horizon of 10 time units. It is likely that faster computers and code optimizations will bring this value down significantly, but unlikely that it would run in real-time. This shows that while individually each strategy in this chapter reduces the search complexity, none of them are sufficient by themselves to bring it close to real-time. In Chapter 6, I present a camera control method that explores the camera to ROI assignments space using a combination of the strategies described in this chapter.

Chapter 5

Performance Metric

There are many possible metrics for estimating the performance of an ACN in a specific task. Depending on the application, one may consider image resolution, tracking accuracy, coverage duration, or number of ROIs captured. For many computer vision applications task performance depends on the ACN's ability to resolve features associated with the ROIs. This chapter details a metric that characterizes this ability, inspired by the pioneering work of Denzler et al. [DZ01, DBN01, DB01, DB02, DZN02] as well as the performance metric introduced by Allen in [All07] and further developed with coauthors including myself in [AW05, WAIB07, IWM08]. Additionally, application-specific factors and spatio-temporal trade-offs are considered.

5.1 Background

My metric uses state-space models to describe target dynamics and measurement uncertainties. In this section, I provide a brief introduction to state-space models and the Kalman filter framework which allows evaluation of the metric over time.

5.1.1 State-Space Models

State-space models [KSH00] are used in applications such as Kalman filter-based tracking to mathematically describe the measurement system and the expected target motion. In [WB01], they are characterized as "essentially a notational convenience for estimation and control problems, developed to make tractable what would otherwise be a notationally-intractable analysis." In state-space models, variables (states, inputs and outputs) are represented using vectors, and equations are represented as matrices.

5.1.1.1 State Variables

The *internal state variables* are defined as the smallest possible subset of system variables that can represent the entire system state at a given time. Formally, at time step t, the system state is described by the *state vector* $\bar{x}_t \in \mathbb{R}^n$. For example in the case of tracking, the user's 3D position is represented by the vector $\bar{x}_t = \begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T}$. If orientation is also part of the state, the vector becomes $\bar{x}_t = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^\mathsf{T}$, where ϕ, θ and ψ are roll, pitch and yaw Euler angles (rotation around the x-, y- and z-axis respectively). Given a point in the state space, a mathematical *motion model* can be used to predict how the target will move over a given time interval. Similarly, a *measurement model* can be used to predict what will be measured by each sensor, such as 3D GPS coordinates or 2D camera image coordinates.

5.1.1.2 Motion Model

A motion model (also called *process model*) describes the expected target motion. Traditionally, such models have been described in terms of a physical parameter that stays constant over time, resulting in models for constant position (*CP*), constant velocity (*CV*) and constant acceleration (*CA*) [CT84]. These traditional stochastic models are shown in Figure 5.1.

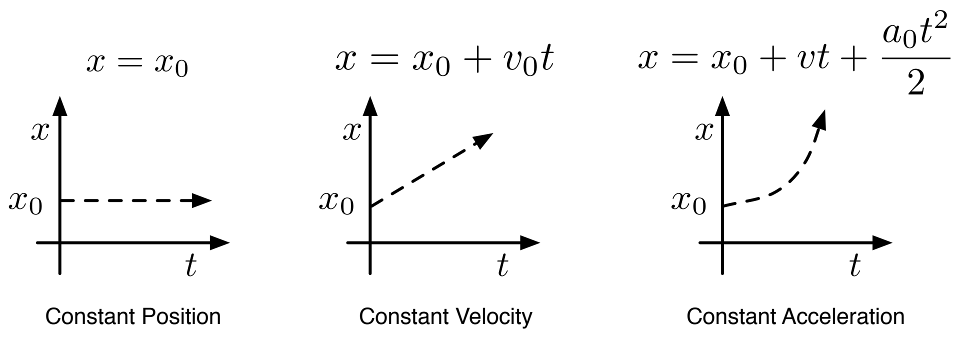


Figure 5.1: Traditional motion models, from [WAIB07].

In the process of stochastic estimation, integrated normally-distributed random noise is used to replace the constant component of each model. For example, the *CV* model in Figure 5.1 becomes $x = x_0 + vt$, with velocity $v = \int a$, and acceleration a is a normally-distributed variable $a \sim \mathcal{N}(0,q)$. Incorporating this random component into each model results in the models known as Position (*P*), Position-Velocity (*PV*) and Position-Velocity-Acceleration (*PVA*), respectively [WAIB07]. Figure 5.2 uses integrals to illustrate the relation between position x with its temporal derivatives and the "driving" noise source $\mathcal{N}(0,q)$ for the *P*, *PV* and *PVA* models.

$$\mathrm{N}(0,q) \rightarrow \int \xrightarrow{x} \quad \mathrm{N}(0,q) \rightarrow \int \xrightarrow{\dot{x}} \int \xrightarrow{x} \quad \mathrm{N}(0,q) \rightarrow \int \xrightarrow{\ddot{x}} \int \xrightarrow{\dot{x}} \int \xrightarrow{x}$$

P Motion Model PV Motion Model PVA Motion Model

Figure 5.2: Stochastic motion models, from [WAIB07].

Choosing the right model for the expected motion plays a crucial role in obtaining good state estimates. The *P* model is most appropriate for situations where there is little to no motion. The *PV* model is used when the motion is fairly constant. The *PVA* model is used for situations in which there are sudden, rapid changes in speed and direction. In Section 3.3 of her thesis [All07], Allen presents a detailed discussion of these models and gives some examples where they are applied.

For a particular state vector $\bar{x}$, the change in state over time can be modeled using deterministic and random components as follows:

$$\bar{x}_{t+1} = f(\bar{x}_t) + \bar{w} \tag{5.1.1}$$

The *state transition function* f is the deterministic component that relates the state at time step t to the state at time step $t+1$. The random variable $\bar{w} \sim \mathcal{N}(0, Q)$ is called *process noise*. In practice, f is linearized about the point of interest $\bar{x}$ in the state space by computing the corresponding Jacobian matrix A:

$$A = \frac{\partial}{\partial \bar{x}} f(\bar{x}) \Big|_{\bar{x}} \tag{5.1.2}$$

This results in the following discrete-time linear equation:

$$\bar{x}_{t+1} = A\bar{x}_t + \bar{w} \tag{5.1.3}$$

While such linearizations can lead to sub-optimal results, they provide a computationally efficient means for state estimation (see [All07], Section 6.1.1).

The continuous-time equivalent of Equation 5.1.3 is the following:

$$\frac{d\bar{x}}{dt} = A_c\bar{x} + q_c \tag{5.1.4}$$

Here A_c is an $n \times n$ continuous-time state transition matrix, and $\bar{q}_c = [0, ..., 0, \mathcal{N}(0, q)]^\mathsf{T}$ is an $n \times 1$ continuous-time process noise vector with corresponding $n \times n$ noise covariance matrix $Q_c = E\left\{q_c q_c^\mathsf{T}\right\}$, where $E\{\}$ indicates expected value [WAIB07]. Table 5.1 shows the continuous-time parameters (the state x, the transition matrix A_c and the process noise covariance matrix Q_c) for the P, PV and PVA models.

Model	$\bar{x}$	A_c	Q_c
P	$[x]$	$\begin{bmatrix} 0 \end{bmatrix}$	$\begin{bmatrix} q \end{bmatrix}$
PV	$\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}$
PVA	$\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}$

Table 5.1: Parameters for the *P*, *PV* and *PVA* models, from [WAIB07].

The discrete-time covariance matrix Q corresponding to the random variable $\bar{w}$ in Equation 5.1.3 can be computed by integrating the continuous-time process in Equation 5.1.4. The solution to this integration is given in [GA93] as:

$$Q = \int_0^{\delta t} e^{A_c t} Q_c e^{A_c^{\mathsf{T}} t} dt \tag{5.1.5}$$

Using the corresponding parameters A_c and Q_c from Table 5.1, matrix Q can be computed for the *P*, *PV* and *PVA* models [WAIB07]:

$$Q_P = [q\,\delta t] \tag{5.1.6}$$

$$Q_{PV} = \begin{bmatrix} q\frac{\delta t^3}{3} & q\frac{\delta t^2}{2} \\ q\frac{\delta t^2}{2} & q\,\delta t \end{bmatrix} \tag{5.1.7}$$

$$Q_{PVA} = \begin{bmatrix} q\frac{\delta t^5}{20} & q\frac{\delta t^4}{8} & q\frac{\delta t^3}{6} \\ q\frac{\delta t^4}{8} & q\frac{\delta t^3}{3} & q\frac{\delta t^2}{2} \\ q\frac{\delta t^3}{6} & q\frac{\delta t^2}{2} & q\delta t \end{bmatrix} \tag{5.1.8}$$

Welch and Bishop discuss the process of choosing q in [WB01].

5.1.1.3 Measurement Model

Similarly to the process model, the measurements obtained from a sensor can be modeled using a deterministic and a random component. The observation at time step t is the *measurement vector* $z_t \in \mathbb{R}^m$. It is related to the state via the following equation:

$$\bar{z}_t = h(\bar{x}_t, \bar{a}_t) + \bar{v} \tag{5.1.9}$$

The non-linear *measurement function* h is the deterministic component that relates the state $\bar{x}_t$ to the measurement $\bar{z}_t$. The vector parameter $\bar{a}_t$ is the *action* taken at time step t, which comprises all parameters that affect the observation process. The action is considered performed before the measurement is taken. The random variable $\bar{v} \sim \mathcal{N}(0, R)$ represents the *measurement noise*. Just as with the state transition function f, the measurement function h is linearized about the point of interest $\bar{x}$ in the state space by computing the corresponding Jacobian matrix H:

$$H = \frac{\partial}{\partial \bar{x}} h(\bar{x}) \Big|_{\bar{x}} \tag{5.1.10}$$

The measurement model becomes:

$$\bar{z}_t = H\bar{x}_t + \bar{v} \tag{5.1.11}$$

In practice, Jacobian matrix H and measurement noise covariance matrix R are determined through sensor calibration.

For example, in the case of a GPS sensor, the measurement function h could transform a 3D point (latitude, longitude, altitude) into a local 3D coordinate system (x, y, z) used for tracking or 3D reconstruction. This transformation is likely a 3×3 linear transform H that can be used directly in Equation 5.1.11.

As a second example, in the case of cameras, the measurement function h is embodied by the camera's *projection matrix* $Proj$, which projects a homogeneous 3D point $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^{\mathsf{T}}$ to a homogeneous 2D image pixel $\begin{bmatrix} u' & v' & 1 \end{bmatrix}^{\mathsf{T}}$ as follows:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = Proj \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.1.12}$$

$$u' = {}^{u}\!/_{w} \tag{5.1.13}$$

$$v' = {}^{v}\!/_{w} \tag{5.1.14}$$

The projection matrix $Proj$ is typically determined through a geometric calibration process like the one in [Zha99]. One common way to define matrix $Proj$ is as follows:

$$Proj = K\,[Rot|Tr] \tag{5.1.15}$$

The 3×3 rotation matrix Rot and the 3×1 translation vector Tr represent the camera *extrinsic parameters*, and specify the transform between the world coordinate system and the camera's coordinate system. The 3×4 matrix $[Rot|Tr]$ is the concatenation of matrix Rot and vector Tr. The *intrinsic parameters* are represented by matrix K, a 3×3 matrix of the form:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1.16}$$

f_x and f_y are the camera focal lengths, measured in pixels, in the x and y directions. s is the skew, typically zero for cameras with square pixels. c_x and c_y are the coordinates of the image center in pixels.

Since the camera measurement process embodied by the computation of u' and v' is not linear, the following Jacobian is used in Equation 5.1.11:

$$H = \begin{bmatrix} \frac{\partial u'}{\partial x} & \frac{\partial u'}{\partial y} & \frac{\partial u'}{\partial z} \\ \frac{\partial v'}{\partial x} & \frac{\partial v'}{\partial y} & \frac{\partial v'}{\partial z} \end{bmatrix} \tag{5.1.17}$$

State-space models allow taking into account measurements from multiple, heterogeneous sensors. Allen describes such a system in her thesis ([All07], Section 5.2.3). In Chapter 7, I present experimental evaluations using a hybrid system that takes measurements from multiple cameras and GPS sensors.

5.1.2 The Kalman Filter

The Kalman Filter [WB01] is a stochastic estimator for the instantaneous state of a dynamic system that has been used both for tracking and for motion modeling [KYS01]. It can also be used as a tool for performance analysis [GA93] when actual measurements (real or simulated) are available. This section provides a brief introduction.

5.1.2.1 Equations

The Kalman filter consists of a set of mathematical equations that implement a predictor-corrector type estimator. These equations can be described using matrices A, H, Q and R defined in the state-space models in Section 5.1.1, and the initial state covariance P_0. The equations for the predict and correct steps are as follows:

Time Update (Predict Step)

- Project state ahead:

$$\hat{x}_t^- = f\left(\hat{x}_{t-1}, t-1\right) \tag{5.1.18}$$

$\hat{x}_t^- \in \mathbb{R}^n$is the *a priori* state estimate at time step t. $\hat{x}_{t-1} \in \mathbb{R}^n$is the *a posteriori* state estimate at time step $t-1$, given measurement z_{t-1}^-. f is the state transition function.

- Project error covariance ahead:

$$P_t^- = A_t P_{t-1} A_t^\mathsf{T} + Q \tag{5.1.19}$$

A_t is the Jacobian matrix of partial derivatives of f with respect to x at time step t.

Measurement Update (Correct Step)

The measurement update can only be performed if a measurement is available.

- Compute Kalman gain:

$$K_t = P_t^- H_t^\mathsf{T} \left(H_t P_t^- H_t^\mathsf{T} + R\right)^{-1} \tag{5.1.20}$$

H_t is the Jacobian matrix of partial derivatives of h with respect to x at time step t. Since h is a function of the selected action $\bar{a}_t$, both H_t and K_t are functions of $\bar{a}_t$.

- Update state estimate with measurement $\bar{z}_t$:

$$\hat{x}_t^+ = \hat{x}_t^- + K_t\left(\bar{z}_t - h\left(\hat{x}_t^-, a_t\right)\right) \tag{5.1.21}$$

The expression $K_t\left(\bar{z}_t - h\left(\hat{x}_t^-, \bar{a}_t\right)\right)$ is called *innovation*, and quantifies the change in state over a single time step.

- Update error covariance:

$$P_t^+ = \left(I - K_t H_t\right) P_t^- \tag{5.1.22}$$

Note that the a posteriori state covariance P_t^+ only depends on the selected action $\bar{a}_t$ and not on the measurement $\bar{z}_t$. This allows evaluation of P_t^+ over time in absence of measurements.

5.1.2.2 Sequential Evaluation

The sequential Kalman filter is a sequential evaluation method for the Kalman filter. The time update (predict) phase is identical to the one in the standard Kalman filter. The sequential evaluation takes place in the measurement update (correct) phase. Each sensor $s = 1 \dots c$ is given its own *subfilter*. The estimate $\hat{x}_t^-$, P_t^- from the predict phase becomes the a priori state estimate for the first subfilter:

$$\hat{x}_t^{-(1)} = \hat{x}_t^-, \qquad P_t^{-(1)} = P_t^- \tag{5.1.23}$$

Each sensor s incorporates its measurement $\bar{z}_t{}^{(s)}$, as in Equation 5.1.21:

$$\hat{x}_t^{+(s)} = \hat{x}_t^{-(s)} + K_t^{(s)} \left(\bar{z}_t{}^{(s)} - h^{(s)} \left(\hat{x}_t^{-(s)}, \bar{a}_t{}^{(s)} \right) \right) \tag{5.1.24}$$

The error covariance is also updated with the contribution $C_t^{(s)}$ of each sensor s at time step t [DND05], as in Equation 5.1.22:

$$C_t^{(s)} = I - K_t^{(s)} H_t^{(s)} \tag{5.1.25}$$

$$P_t^{+(s)} = C_t^{(s)} P_t^{-(s)} \tag{5.1.26}$$

The output of each subfilter becomes the input of the next subfilter:

$$\hat{x}_t^{-(s+1)} = \hat{x}_t^{+(s)}, \qquad P_t^{-(s+1)} = P_t^{+(s)} \tag{5.1.27}$$

Equations 5.1.26 and 5.1.27 can be written for the entire set of subfilters as:

$$\hat{x}_t^{+(c)} = \hat{x}_t^{-(1)} + \sum_{s=1}^{c} K_t^{(s)} \left(\bar{z}_t{}^{(s)} - h^{(s)} \left(\hat{x}_t^{-(s)}, \bar{a}_t{}^{(s)} \right) \right) \tag{5.1.28}$$

Similarly, Equation 5.1.26 can be written for the entire set of subfilters as:

$$P_t^{+(c)} = \prod_{s=1}^{c} C_t^{-(s)} P_t^{-(1)} \tag{5.1.29}$$

The sequential Kalman filter allows simply skipping a sensor if it does not generate a measurement during a particular time step. However, the contribution $C_t^{(s)}$of each sensor s at time step t depends on the a priori covariance of subfilter s, so the final a posteriori state $\hat{x}_t^+ = \hat{x}_t^{+(c)}$ and covariance $P_t^+ = P_t^{+(c)}$ depend on the order in which the subfilters are evaluated. In practice, the effects of ordering are usually ignored.

5.2 Estimating and Predicting Performance

I define the performance of a camera configuration as its ability to resolve features in the working volume, and measure it using the uncertainty in the state estimation process. Uncertainty in the state $\bar{x}$ can be measured using the error covariance P_t^+ computed in the Kalman filter Equation 5.1.22. Allen and Welch [AW05] note that error covariance estimation conceptually reverses the signal direction one normally thinks about in tracking. Tracking can be thought of as propagating a signal from a 3D point through a sensor and into an estimation algorithm. Error covariance estimation instead propagates estimated measurement noise signals from the sensors through the measurement models and into the working volume, where they are combined with expectations of target motion to produce an estimate of the error at that point.

In Chapter 3, I discussed the need to incorporate time as a dimension of the search space due to the dynamic nature of the events being captured and the characteristics of the active cameras used to capture images. I concluded that spatial aggregation of metric values over the environment for the current camera configuration is not sufficient, and future camera configurations need to be evaluated as well. This results in the performance metric evaluating a *plan* (a sequence of camera configurations). The difficulty is that at each time instant a camera's measurement can be successful or unsuccessful. Denzler et al. [DZN02] introduced a way to deal with visibility at each step. Deutsch et al. [DZDN04] extended this approach to multiple steps into the future using a visibility tree, and then sped up the evaluation by linearizing the tree and extended the approach to multiple cameras using a sequential Kalman filter in [DWN06]. I employ a similar approach, but use a norm of the error covariance P_t^+ instead of entropy as my performance metric, and use a different method to aggregate it over space and time. As the metric measures the uncertainty in the state, when comparing camera configurations smaller values are better and larger values are worse.

In [IWM08], I introduced the concept of *surrogate models* to allow evaluation of the metric in state-space only where needed: at a set of 3D points associated with each ROI. The metric values are aggregated over the state elements in the surrogate model of each ROI, over each agency and over the entire environment. At all aggregation levels, weights can be used to give more importance to a particular element.

To evaluate future camera configurations, I repeatedly evaluate the Kalman filter equations, stepping forward in time, while using process models to predict ROI trajectories and updating the measurement models with the appropriate camera parameters. When looking into the future, no actual measurements $\bar{z}_t$ are available at time t, but estimated measurements $\hat{z}_t = h\left(\hat{x}_t^-, a_t\right)$ can be used instead. Substituting $\hat{z}_t$ for $\bar{z}_t$ results in zero innovation. Equation 5.1.21 becomes simply:

$$\hat{x}_t^+ = \hat{x}_t^- \tag{5.2.1}$$

At each time step, a camera measurement can be successful or not, depending on a variety of factors such as visibility, surface orientation, etc. If the measurement is assumed successful, the a posteriori state error covariance P_t^+ is computed as in Equation 5.1.22. If the measurement is assumed unsuccessful, the measurement update step cannot be performed, and $P_t^+ = P_t^-$. The two outcomes can be characterized by two distributions with the same mean $\hat{x}_t^+$ and covariances P_t^+ and P_t^-. Given the probability that a measurement is successful ms, these distributions can be considered as components of a Gaussian mixture $\mathscr{M}$ [DWN06]:

$$\mathscr{M} = ms \cdot \mathscr{N}\left(\hat{x}_t^+, P_t^+\right) + (1-ms) \cdot \mathscr{N}\left(\hat{x}_t^+, P_t^-\right) \tag{5.2.2}$$

The covariance of the Gaussian mixture $\mathscr{M}$ is:

$$P_t^{+'} = ms \cdot P_t^+ + (1-ms) \cdot P_t^- = (I - ms \cdot K_t H_t)\, P_t^- \tag{5.2.3}$$

Since the two distributions have the same mean $\hat{x}_t^+$, $\mathscr{M}$ is unimodal and can be approximated by a new Gaussian distribution $\mathscr{M}'\left(\hat{x}_t^+, P_t^{+'}\right)$, as shown in [DWN06]. It follows that to incorporate the outcome of an observation, one simply has to compute the success probability and replace the computation of the a posteriori error covariance in Equation 5.1.22 with the one in Equation 5.2.3.

To aggregate over time, Deutsch et al. [DZDN04, DWN06] propose simply using the metric value at the horizon. In practice, I observed that the value at the horizon is very sensitive to the camera configurations and ROI positions during the last few time steps before the horizon. For example, if a camera view is occluded during the last time step, the metric value would increase to reflect the absence of the measurement. Depending on the circumstances, this metric increase could end up penalizing plans that would have performed well in previous intervals. To fully characterize the evolution of the metric value over time, my approach is to aggregate all the values up to the horizon instead.

In summary, the performance metric is computed by repeatedly stepping through the Kalman filter equations and changing relevant state-space model parameters at each time step. The state is initialized using the current Kalman filter state estimate. The Kalman correct step is evaluated sequentially, for all sensors involved (e.g., cameras and GPS), as described in Section 5.1.2.2. To convert the error covariance into a single number, the function $KalmanMetric()$ called at each time step returns the square root of the maximum value on the diagonal of the portion of the error covariance matrix P_t^+ corresponding to the position part of the state. Other functions such as the determinant or the trace can be used instead

Algorithm 5.1 Evaluating plans using the performance metric.

```
function $Metric(R,P,H)$
Input: set of ROIs $R$, plan $P$, planning horizon $H$
Output: metric value $m$
$m = 0$
for all ROIs $r \in R$
    $x_0^- = GetCurrentState(r)$
    $P_0^- = GetCurrentCovariance(r)$
    $m_r = 0$
    for $t = 1 \dots H$
        $(x_t^-, P_t^-) = KalmanPredict(x_{t-1}^+, P_{t-1}^+)$
        $(x_t^+, P_t^+) = KalmanCorrect(x_t^-, P_t^-)$
        $m_r = m_r + KalmanMetric(P_t^+)$
    $m = m + m_r \cdot r_{importance}$
return $m$
```

of the diagonal maximum. One advantage of using the square root of the covariance in the metric function is that the measurement unit for the metric is the same as the measurement unit of the state space. For example, if the state consists of 3D point positions measured in meters, the metric value will also be in meters. This makes it more intuitive for a system user to specify application requirements such as the desired maximum error in a particular area where important events take place.

Aggregation over space and time can be performed using weighted sums, with weights being used to give more importance at various levels, such as to an element of a ROI's surrogate model, to a ROI, to an agency, or to a time instant. Algorithm 5.1 presents the process in detail, and illustrates aggregation using importance weights for each ROI.

The metric computation described here works in tandem with a Kalman filter that is used to estimate the ROI trajectories. At each time instant, the filter incorporates the latest measurements from cameras and other sensors, and saves the *current estimate* (x_0^-, P_0^-). This estimate is the starting point for all metric evaluations, which then step through the Kalman filter equations up to the planning horizon using simulated measurements to evaluate all candidate plans.

5.3 Incorporating Task Requirements Into the Metric

Many task requirements become easy and intuitive to incorporate in the performance metric due to the metric being in state-space. In this section, I give a few examples of how common task requirements can be incorporated, grouped by the application domain.

5.3.1 Tracking

The requirements of a tracking application are naturally expressed in state-space. For example, the user of a tracking system may specify the desired tracking accuracy for a particular region where events might be more important or more likely to happen, or for a particular ROI enclosing a person deemed more important. This importance appears as a weight in the aggregation part of the metric function, and affects the optimization process.

The state can be modeled as a single 3D point if the user is only interested in the target's position, or multiple points placed for example at the skeleton joints if the user desires motion capture. Models can be augmented with local surface orientation if the user wants to take into account self-occlusion. The state can also be reduced to 2D if the tracking system employs a ground plane assumption, in which case a single camera measurement is sufficient to determine the target location at any time.

As mentioned in Section 5.1.1, state models may also be augmented with hidden variables such as target speed and acceleration, if appropriate, depending on the expected characteristics of the target motion.

5.3.2 Surveillance

Surveillance tasks have very diverse goals. Here I give two examples of surveillance tasks and how they can be accommodated by my approach.

- **Following a person or object throughout the environment**: Enclose the person or object in a ROI and set its importance higher than the importances of other ROIs. The camera control method will automatically follow the more important ROI and exhibit complex behaviors such as coordination and hand-offs between cameras.

- **Capturing images of people's faces**: Augment the state model with points on the person's face and the person's movement direction, and use the approach in Section 5.4.3 to incorporate the incidence angle into the aggregation function. To ensure the capture of as many faces as possible, have a single successful capture drastically decrease the uncertainty. This will result in cameras capturing other faces that are comparably more uncertain. As time passes, repeated captures of the same person's face if still present in the environment can be ensured by having uncertainty increase slowly over time in the process model.

5.3.3 3D Reconstruction

When the client is a 3D reconstruction method, surrogate models can be adjusted to better fit the requirements of the particular 3D reconstruction approach.

For example, models for stereo reconstruction could include local surface orientation, which combined with an appropriate aggregation function would give more weight to samples better suited for stereo matching. This change also results in the metric giving preference to camera setups that have been shown to work better in stereo reconstruction: cameras placed relatively close to each other and aimed in a similar direction.

Models for volumetric reconstruction approaches could consist of a medial axis-like representation: a set of 3D centroids and rays to the target surface. The metric to be optimized could then be an aggregation of the uncertainties in the length of the rays together with the ray lengths themselves, since volumetric approaches typically seek a 3D model of minimum volume. This change also results in the metric giving preference to camera setups that have been shown to work better in volumetric reconstruction: cameras placed uniformly around a target, in an outside-looking-in arrangement.

5.3.4 New ROIs

There are multiple ways of incorporating new persons or objects entering the environment.

One approach is suggested by previous work in surveillance such as [KYL$^+$08], and applies to enclosed environments with a limited number of entry points. In this case, previous approaches have scheduled cameras to periodically survey the entry points for any potential new targets. A similar approach can be straightforwardly applied to my method by having static ROIs enclose the entry points, and automatically creating new dynamic ROIs enclosing any new targets when they are detected entering the scene. Given appropriate surrogate models for each ROI, the camera control method would automatically plan available cameras to both survey the static ROIs enclosing the entry points and to track the newly created dynamic ROIs.

Another approach was introduced by Sommerlande and Reid in [SR08a]. They model the probability of new targets entering the scene using a background Poisson process, and integrate the noise injected by the process into the performance metric from [DWN06]. Since my performance metric is similar to the one in [DWN06], this approach is easily applicable to it as well.

A final approach is to have the state of new ROIs bootstrapped by input from an external system, for example GPS or a tracking system that uses a few static cameras with wide fields of view to cover the entire scene. This is another area where state-space models demonstrate their usefulness: besides helping to acquire new targets, measurements from these external devices can be incorporated into the sequential evaluation process described in Section 5.1.2.2, and thus contribute to reducing the uncertainty in the state.

5.4 Incorporating Performance Factors into the Metric

Researchers have attempted to express the intricacies of camera performance factors such as placement, resolution, field of view, focus, etc. into metrics that could measure and predict camera performance in computer vision algorithms. Table 5.2 lists the most common factors encountered in the various related domains.

Factor	Previous work
Image resolution	[All07, BSP05, Che02, CL04, Cow88, FCOL00, IWM08, Mas95, MD04, OM02, SSSAH04, SWI06, TTK96, WSH98, YUK07]
Focus, Depth of field	[CL04, Cow88, IWM08, Mas95, MD04, OM02, SSSAH04, TTK96, WSH98, YUK07]
Field of view	[All07, BSP05, CL04, Cow88, DB02, IWM08, Mas95, MD04, OM02, SSSAH04, SWI06, TTK96, YUK07]
Occlusion	[All07, Che02, CL04, Cow88, EGG06, FCOL00, IWM08, Mas95, MD04, OM02, SSSAH04, SWI06, TTK96, YUK07]
Object distance	[All07, BSP05, IWM08, Mas95, WSH98]
Incidence angle	[BSP05, IWM08, Mas95, MD04, OM02, SSSAH04, YUK07]

Table 5.2: Performance factors in previous approaches.

When using fixed cameras, one would typically choose a metric that properly characterizes as many factors as possible. When using PTZ cameras, one also has to weigh the benefits of incorporating additional performance factors against the requirement that the metric computation is incorporated into a system capable of controlling cameras in real-time. In this section, I show how many factors that have been known to influence the performance of computer vision tasks can be incorporated into my performance metric, and how they affect the metric values.

5.4.1 Experiment Setup

To illustrate the effects of these factors, I chose a simple simulated setup, with 3 cameras positioned 6 meters above ground, as shown in a top-down view in Figure 5.3, and a single ROI. The target inside the ROI is a person walking from coordinates $(-20,0,0)$ to coordinates $(20,0,0)$ over 21 seconds with a constant speed of 1 meter per second. The surrogate model of the target consists of 4 equidistant points on the vertical axis, each within a cubic region 0.5 meters on the side, over a total distance of 2 meters representing the height of a person. Additionally, I assume GPS measurements are available and evaluate their impact. Slight variations of this experiment setup are used in some of the experiments in this section.

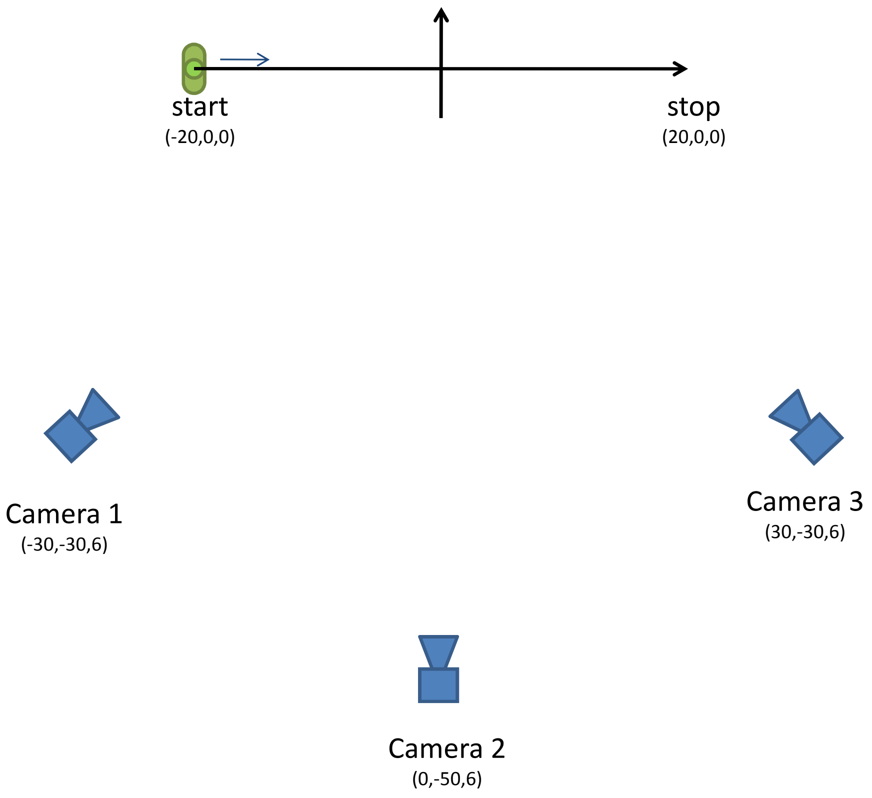


Figure 5.3: Overview of the setup for performance metric experiments.

5.4.2 Occlusion

Occlusions result in cameras being unable to provide measurements, or providing measurements that are erroneous and should be ignored. They are taken into account in computing the metric by using the occlusion probability o in the Kalman filter measurement update step as shown in Equation 5.2.3. The probability of making a successful measurement is $ms = 1 - o$.

To verify the impact of occlusion on the metric values, I set up the following experiment. Camera 1 and Camera 3 are fixed, their field of view set to enclose the entire target trajectory. An occluder is placed between Camera 2 and the target such that the target is occluded from Camera 2's view for an interval of 3 seconds, starting 8 seconds into the experiment. Figure 5.4 shows a top-down view of the experiment setup.

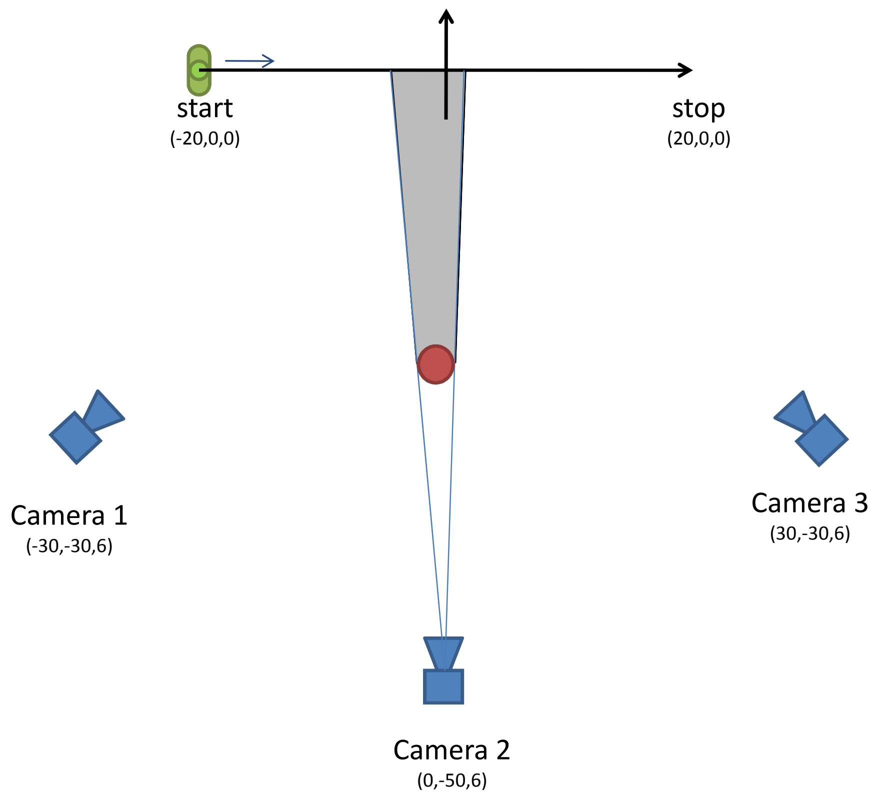


Figure 5.4: Overview of the setup for the occlusion experiment.

Figure 5.5 shows the metric values over time. The GPS is considered unavailable. The metric values for a situation when the occlusion did not occur are shown for comparison. Mechanical noise is set to 1° in both pan and tilt. The evaluation horizon for the metric is 5 seconds.

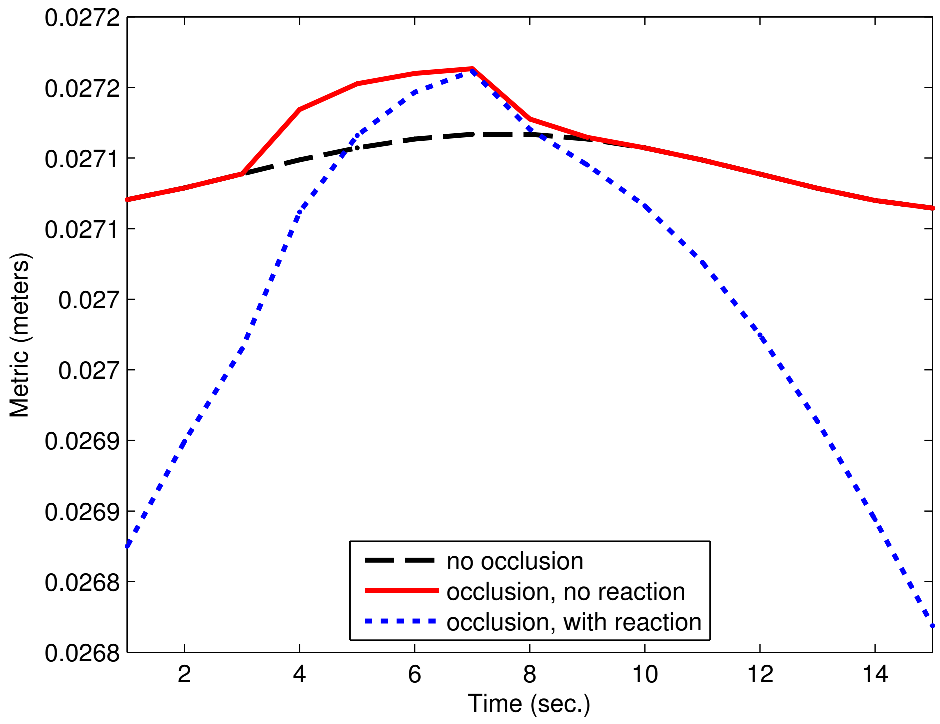


Figure 5.5: Effect of occlusion on metric values.

In a first run, Camera 2 is left fixed, such that its field of view encloses the entire trajectory. For the first 3 seconds, the metric values are identical to the ones shown for comparison without occlusion. As the metric aggregates values over 5 seconds, the effect of the occlusion starts to affect its value after $8-5=3$ seconds. The metric values are greater for the times when the occlusion occurs during the evaluation horizon.

In a second run, I illustrate the effect of being able to predict the occlusion and react accordingly: instead of being zoomed out to capture the entire target trajectory, Camera 2 is first zoomed in to capture for the first 8 seconds, then is transitioned during the occlusion, and finally is zoomed in to capture for the last 9 seconds. As expected, the higher zoom results in smaller metric values than if the camera had not reacted to the occlusion.

5.4.3 Incidence Angle to ROI Surface

If the ROI model includes normals at each point, the incidence angle α between the ray from the camera and the ROI surface at that point can be taken into account in a similar fashion to the way occlusion is handled. The probability of making a successful measurement *ms* can also include some function of the incidence angle α, such as $cos(\alpha)$ or $\sqrt{cos(\alpha)}$ as a multiplicative term. To show this effect in action, I use the same experimental setup, but only turn on Camera 3. Figure 5.6 shows a top-down view of the experiment setup.

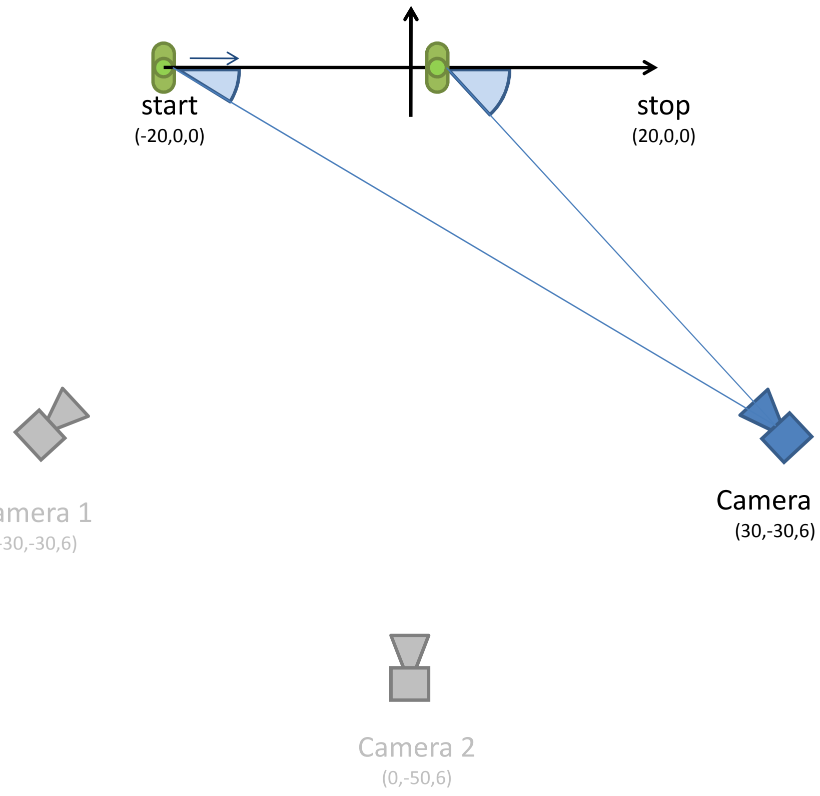


Figure 5.6: Overview of the setup for the incidence angle experiment.

Figure 5.7 shows the metric values over time. The time horizon for the metric was set to 5 seconds, and I used $\sqrt{cos(\alpha)}$ as the multiplicative term. The person model is augmented with a surface normal in the positive *x* direction for the top point representing their head. As the person approaches Camera 3, the angle between the normal and the ray from the camera to their location increases. The visibility of the person's face decreases, which results in an increase of the metric values.

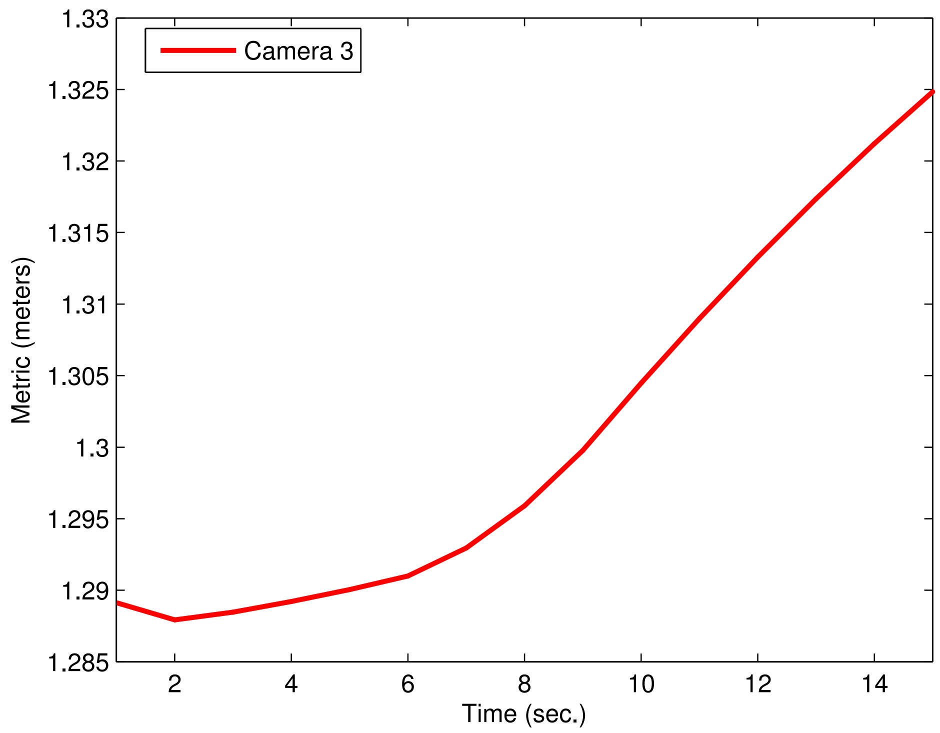


Figure 5.7: Effect of incidence angle on metric values.

5.4.4 Image Resolution

Image resolution is taken into account in the measurement model, as it is one of the factors in the computation of each camera's projection matrix. A lower image resolution results in larger uncertainties in the ROI 3D space, reflected in higher metric values. Figure 5.8 shows the metric values over time when using "full-resolution" (640×480) and "half-resolution" (320×240) images. The time horizon for the metric was set to 5 seconds.

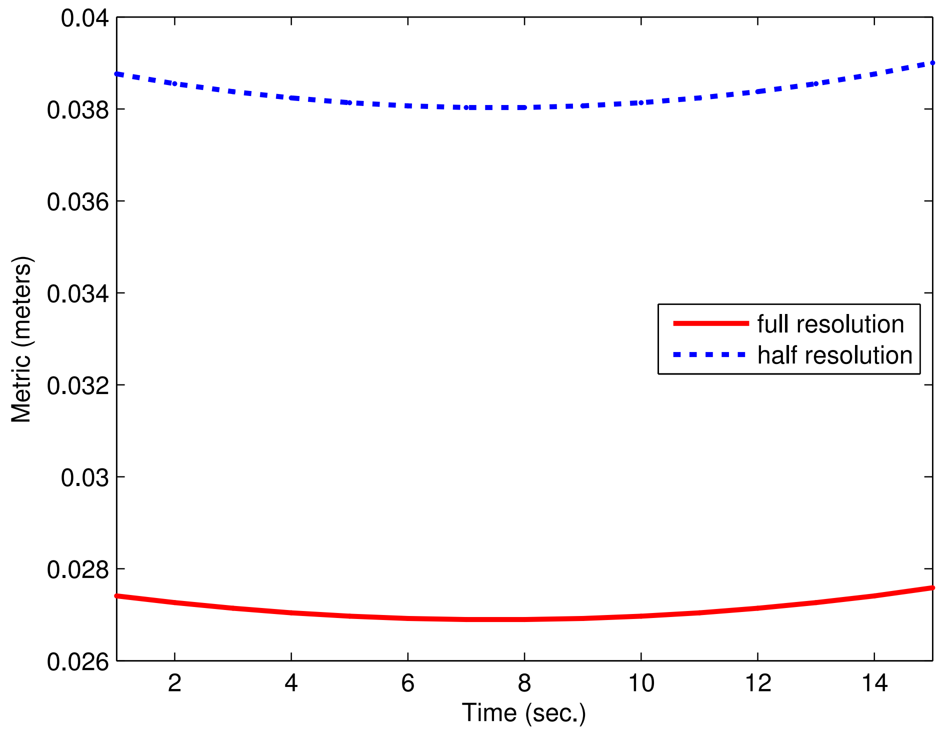


Figure 5.8: Effect of camera image resolution on metric values.

5.4.5 Number of Cameras

The number of cameras capturing a particular ROI affects the metric values as well: the more cameras, the smaller the uncertainty. To verify this, I use the same experiment setup as before, and turn on cameras one by one. Figure 5.9 shows the metric values over time for $0, 1, 2$ and 3 cameras. The time horizon for the metric is 5 seconds. In the single camera case, only Camera 1 was active, and the uncertainty increases over time because the ROI is moving away from it. Adding Camera 2 helps decrease the uncertainty. Adding Cameras 3 helps further, but there are diminishing returns in terms of reducing the uncertainty.

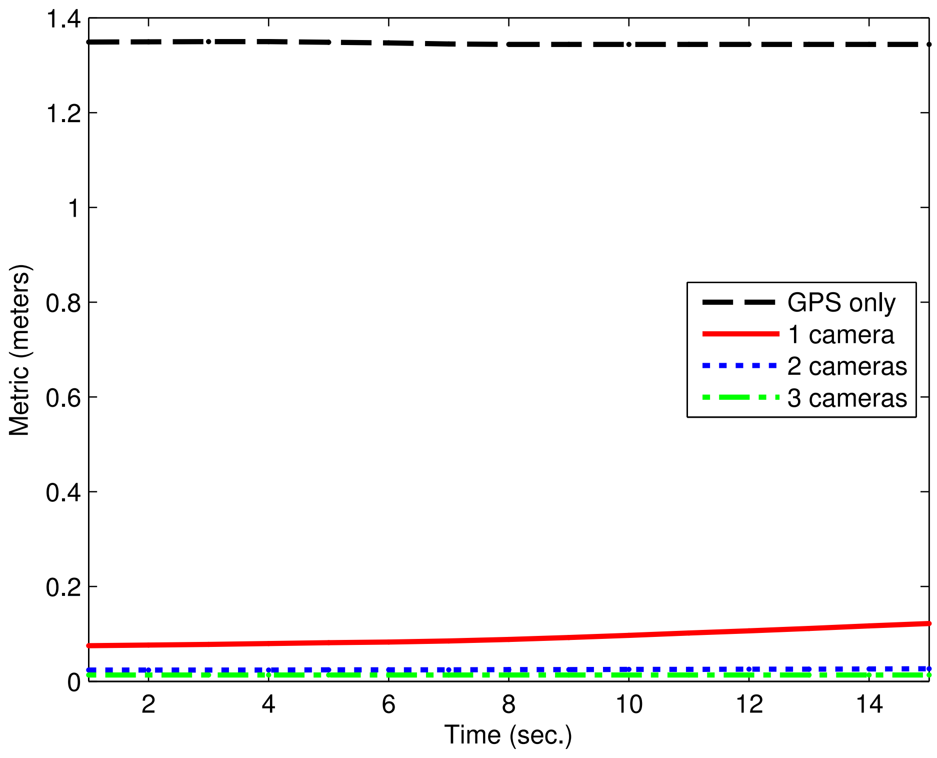


Figure 5.9: Effect of the number of cameras on metric values.

5.4.6 Camera Field of View and Mechanical Noise

To maximize the size of the ROI in the camera images, cameras should be zoomed in as much as possible. However, at high zooms mechanical vibrations tend to affect PTZ cameras, and following the ROIs may become harder because camera movements may lack the necessary precision. Also, fast-moving ROIs may prematurely exit the field of view.

The field of view of the camera appears in the measurement model as shown in Section 5.1.1.3, contributing to the computation of each camera's projection matrix. Intuitively, when the field of view increases, the 3D volume inside the solid angle corresponding to a particular pixel in the image increases as well, and there are less image pixels covering a particular 3D volume.

Mechanical vibrations can be modeled as angular noise, measured in degrees for both the pan and tilt motors as AN_{pan} and AN_{tilt} respectively. The corresponding localization error, measured in pixels, can be computed using the value of the current field of view. Pixel values PN_{pan} and PN_{tilt} are computed as the sides of the rectangle enclosed in the solid angle α, where α is the solid angle measuring AN_{pan} and AN_{tilt} degrees on the horizontal and vertical respectively, and centered on the ray from the camera center of projection to the 3D point being imaged.

The noise values PN_{pan} and PN_{tilt} are multiplied with a scalar PN representing the noise in the measurement process itself (measured in pixels) and added to the diagonal of the measurement noise matrix R_t in the measurement model described in Section 5.1.1.3:

$$\begin{aligned} R_t(1,1) &= R_t(1,1) + PN_{pan} \cdot PN \\ R_t(2,2) &= R_t(2,2) + PN_{tilt} \cdot PN \end{aligned} \tag{5.4.1}$$

To look at the effect of camera field of view and mechanical noise on the metric values, I use the experiment setup in Figure 5.3, with all 3 cameras turned on. The cameras' pan and tilt settings are fixed, and set to look towards the origin, where the ROI is predicted to be after 10 seconds, midway through the experiment. The zoom setting is varied to change the field of view of the cameras from 0 to 90°. Figure 5.10 shows the metric values evaluated over the entire duration of the experiment (20 seconds), plotted against the varying field of view of the cameras.

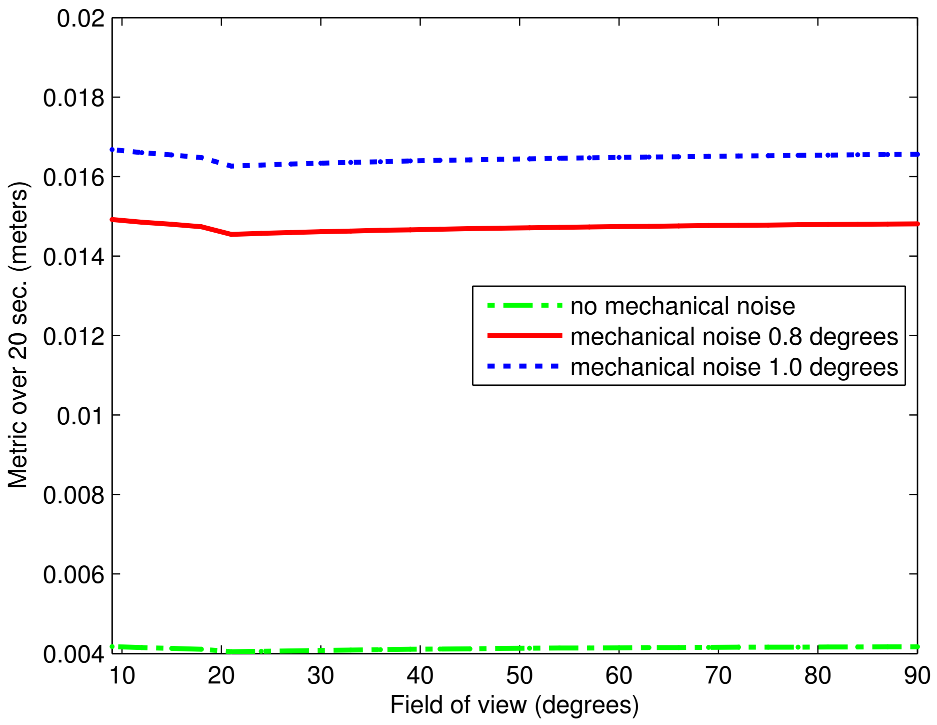


Figure 5.10: Effect of camera field of view and mechanical noise on metric values.

When the field of view is small, the cameras cannot see the entire trajectory, and the metric values are higher. As cameras zoom out, more and more of the trajectory becomes visible, and the metric values decrease. At a field of view of around 22°, the entire trajectory becomes visible. Larger fields of view mean that at that time the target's image occupies less pixels, and the metric values increase to reflect the system's diminishing ability to resolve 3D features.

I ran the experiment 3 times, with different values for the mechanical noise due to the motors of the PTZ unit. As expected, uncertainty increased with the amount of mechanical noise.

5.4.7 Angle Between Cameras as Seen from the ROI

To show the impact of the angle between cameras as seen from the ROI, I reproduce the conditions of an experiment described by Chen in [Che02]. Camera 1 is placed at various positions on a circle of 50 meters radius around the origin, 7 meters above the ground plane.

Camera 2 is fixed. Camera 3 is disabled. Figure 5.11 shows a top-down view of the experiment setup.

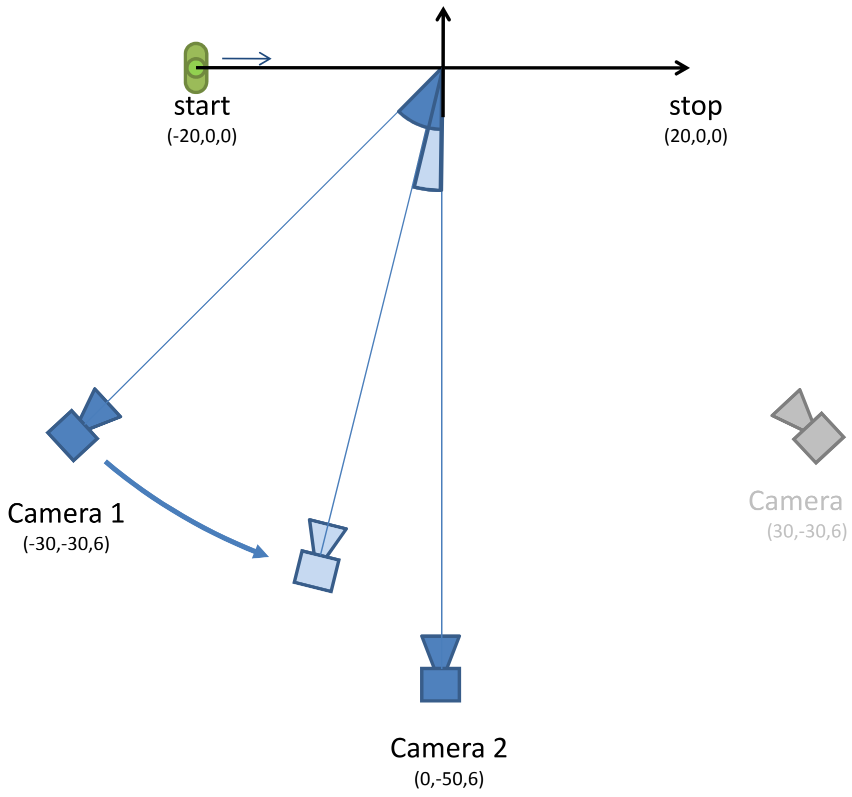


Figure 5.11: Overview of the setup for the angle between cameras experiment.

Figure 5.12 (Cf. 4.13 (middle) from [Che02]) shows the metric values evaluated for a duration of 20 seconds plotted against the angle between the two cameras as seen from the point of view of the target.

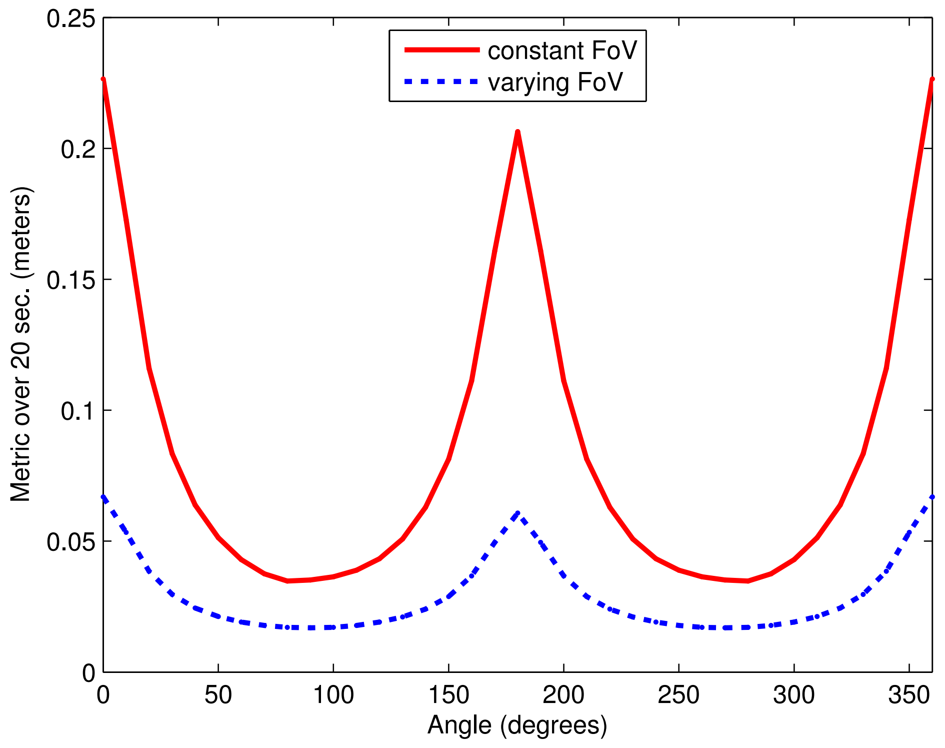


Figure 5.12: Effect of the angle between two covering cameras as seen from the target on metric values.

Just as in the experiment by Chen [Che02], the smallest metric values are around 90° and 270°. The highest metric values are at 0° and 180°. The metric value is smaller at 180° (cameras looking towards each other) than at 0° (cameras on top of each other). In a first run, the field of view of both cameras is held fixed at its maximum value, 90°. A second run shows the effect of zooming in the fixed camera as much as possible, and varying the field of view of the moving camera to tightly enclose the target trajectory for the entire duration of the experiment. As expected and shown in Section (5.4.6), zooming in reduces the metric value.

5.4.8 Camera-ROI Distance

The effect of the distance between a camera and the ROI it is capturing can be incorporated into the measurement noise model of each camera as an additive term on the diagonal of the measurement noise covariance matrix R_t described in Section 5.1.1.3. The exact relationship can be determined through measuring experiments, as shown by Allen in [All07]. I quantify the effect of distance on the metric values with an experiment in which I use the linear relationship derived by Allen: $CCDnoise = 0.00004d + 0.002$. In this experiment, Camera 1 and Camera 3 are fixed, while Camera 2 is translated between 20 and 80 meters away from the origin, between coordinates $(0, -20, 6)$ and $(0, -80, 6)$. Figure 5.13 shows a top-down view of the experiment setup.

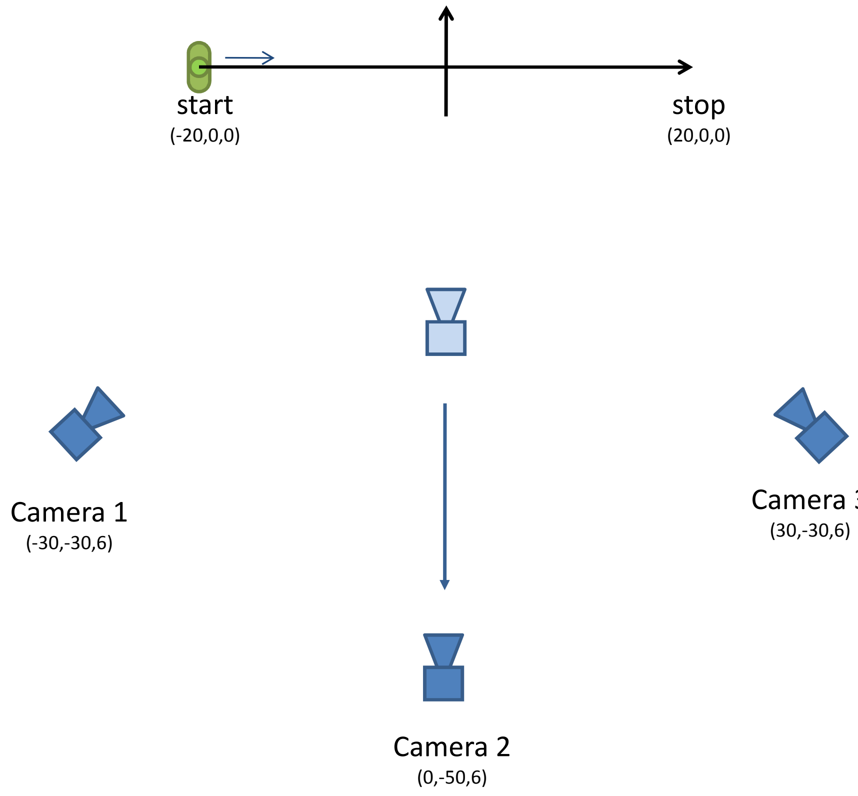


Figure 5.13: Overview of the setup for the camera-ROI distance experiment.

Figure 5.14 shows the metric values evaluated over 20 seconds plotted against the distance of Camera 2 to the origin. Mechanical noise is set to 1° in both pan and tilt.

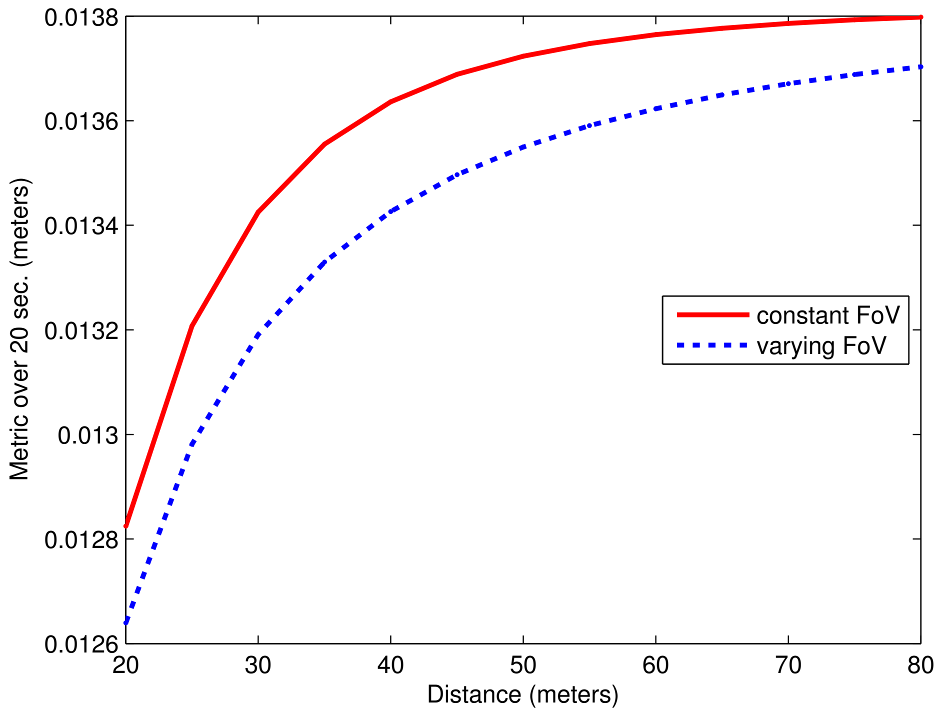


Figure 5.14: Effect of camera-target distance on metric values.

In a first run, the field of view of Camera 2 is held fixed at the smallest zoom, with a field of view of 90°. The entire ROI trajectory is visible. The metric values increase with distance.

In a second run, the field of view of Camera 2 is varied so that the entire target trajectory is always visible and enclosed as tightly as possible. The metric values still increase in spite of increased resolution compared to the first run, mainly because the mechanical noise measured in pixels increases when the field of view decreases, as previously shown in Section 5.4.6.

5.4.9 Frequency of Measurements From Other Types of Sensors

As described in Section 5.1.1.3, the state-space model allows my approach to take into account measurements from sensors other than cameras. For example, a GPS sensor could provide measurements as well. These measurements are incorporated in the correct step of the sequential Kalman filter.

The frequency with which the GPS measurement is incorporated affects the metric values.

The experiment in Figure 5.15 shows the metric value over time when the GPS is unavailable, available every second, every 2 seconds and every 4 seconds. The evaluation horizon for the metric is 5 seconds.

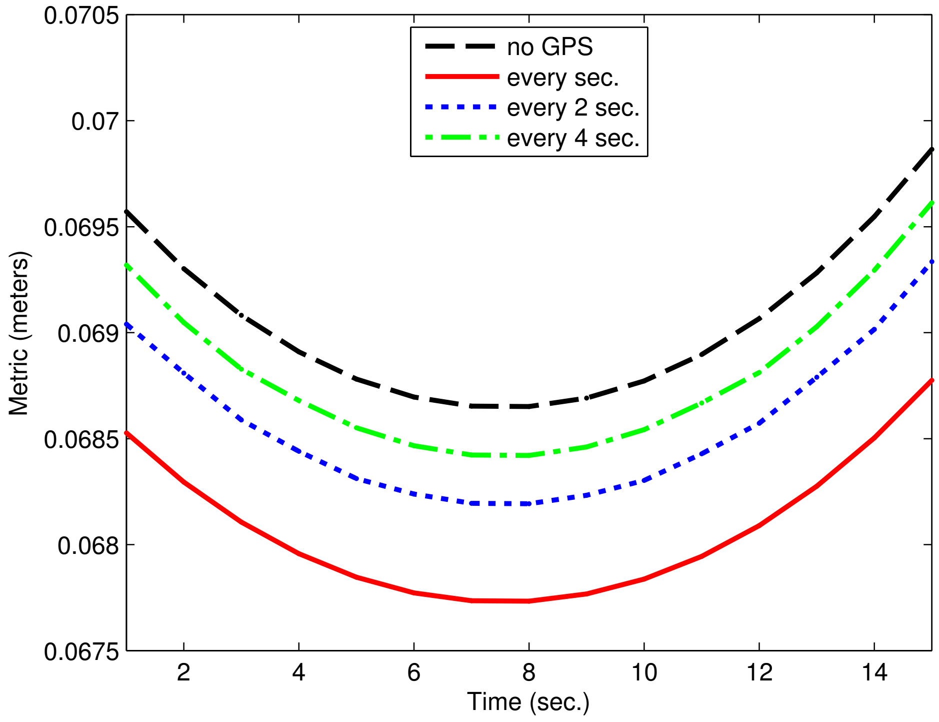


Figure 5.15: Effect of GPS frequency on metric values.

The metric values are smaller when incorporating any GPS measurements than when the GPS is unavailable, and decrease with increasing frequency of GPS measurements.

5.4.10 Measurement Noise from other Types of Sensors

The amount of noise in the GPS signal can influence the metric value. The noise value is placed on the diagonal of the measurement noise matrix R_t corresponding to the GPS in the measurement model described in Section 5.1.1.3.

This experiment shows the effect of the amount of noise present in the GPS signal, which is assumed to be available during all time intervals. Figure (5.16) shows the metric value over time. The evaluation horizon for the metric is 5 seconds.

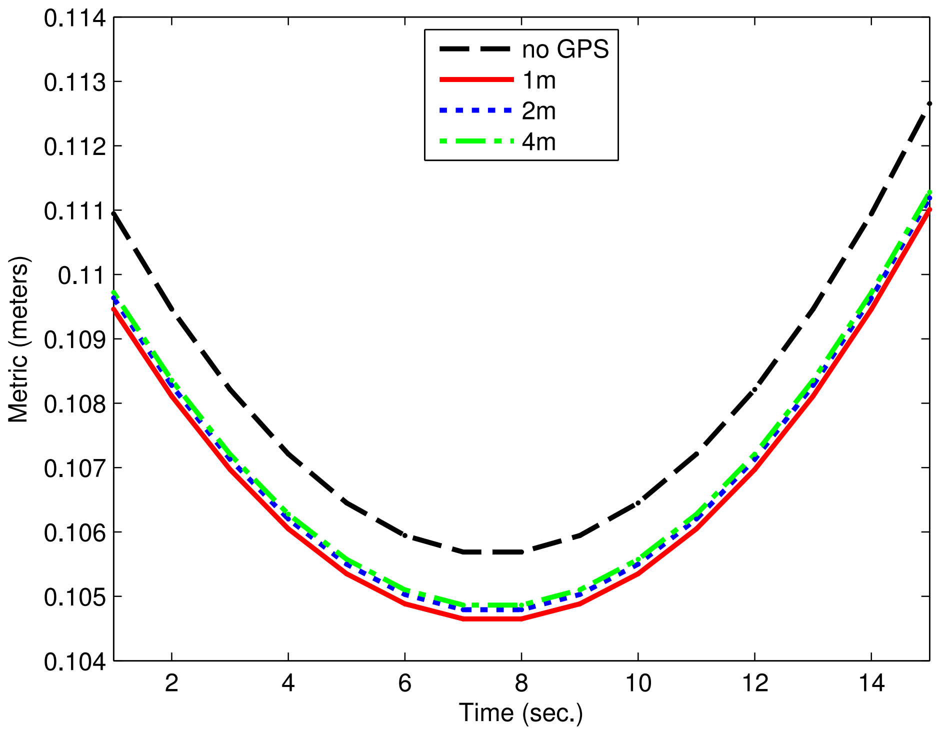


Figure 5.16: Effect of GPS noise on metric values.

As expected, higher amounts of noise in the GPS signal result in higher metric values, but still smaller than if the GPS was not available.

5.4.11 Camera Focus

Points associated with the ROIs can be out of focus due to factors such as lens aberrations and light diffraction. In [IWM08], I showed that focus can be incorporated as a source of noise in the measurement model of each camera k. I equated the amount of noise injected to the diameter of the circle of confusion cc_k, computed using the thin lens equation [Ray02] as:

$$cc_k = \frac{a_k}{ps_k} \frac{f_k}{d_k - f_k} \frac{|D - d_k|}{D}, \tag{5.4.2}$$

where a_k is the camera aperture diameter, f_k is the focal length of the lens, d_k is the focus distance and D is the distance from the camera to the point of the surrogate model where the uncertainty is evaluated. The term ps_k is the pixel size in real-world units, and is used as a scaling factor because cc_k is measured in pixels. The result is added to the diagonal of the measurement noise covariance matrix R_t described in Section 5.1.1.3.

5.4.12 Dynamic Occlusion

Dynamic occlusions are occlusions of a ROI by another part of itself or another ROI. They can be integrated into the metric computations the same way as static occlusions: by using the occlusion probability (see Section 5.4.2). The details in this section refer to surrogate models containing 3D points and associated position uncertainties, but the method can be generalized for other surrogate models.

The dynamic occlusion probability of a particular ROI r at a particular time instant t can be computed by casting rays through a number of pixels in each camera toward 3D points m in the surrogate model of the ROI, and then computing and aggregating the dynamic occlusion probabilities for all rays. I use a weighted average as the aggregation function, where the weights are the probabilities that the rays intersect the points in the surrogate model of the ROI. The computation is performend at discrete moments in time, at the end of each planning cycle.

I compute the probability that a ray from a camera's center of projection through a pixel in its image intersects a model point using the 3D uncertainty in the model point's position. To simplify the computation, I replace the ellipsoid representing the uncertainty in the position of the model point with a sphere of the same volume. This approximation is equivalent to replacing the position distribution with an isotropic 3D Gaussian distribution. I compute the value of this isotropic 3D Gaussian function at the 3D point on the ray closest to the center of the sphere. I consider this value as the peak of a 1D Gaussian function g_m that represents the probability that the model point is on the ray. I approximate the variance of g_m (its width) using the distance between the two points where the ray intersects the sphere. Finally, the intersection probability is computed as the integral of g_m over a distance interval $[Dmin, Dmax] \subset [0, \infty]$.

I use the same 1D Gaussian function to compute the occlusion probability o for model points m associated with ROI r by integrating over distance along each ray using the following formula:

$$o = \int_{Dmin}^{Dmax} \left(\sum_{r_i} Prob\left(Dist\left(c, m_i\right) < x\right) \right) \cdot \left(Prob\left(Dist\left(c, m\right) > x\right)\right) dx \tag{5.4.3}$$

The term $\sum_{r_i} Prob\left(Dist\left(c, m_i\right) < x\right)$ is the probability that any model points m_i of any ROI r_i is closer to camera c than the distance x. The term $Prob\left(Dist\left(c, m\right) > x\right)$ is the probability that the model point m of ROI r currently being evaluated is farther away from camera c than the distance x. Basically, at each distance x, the probability of occlusion for model point m of ROI r by model points m_i of all other ROIs r_i is the probability that any of the points m_i are closer to the camera than x and that point m is farther away from the camera than x. These probabilities are computed as follows:

$$\begin{aligned} Prob\left(Dist\left(c, m_i\right) < x\right) &= \textstyle\int_{Dmin}^{x} g_{m_i}\left(y\right) dy \\ Prob\left(Dist\left(c, m\right) > x\right) &= \textstyle\int_{x}^{Dmax} g_{m}\left(y\right) dy \end{aligned} \tag{5.4.4}$$

Algorithm 5.2 presents the overall process of computing dynamic occlusions. It calls several functions, listed and explained below.

Function $EvaluationPixels\left(c, m, t\right)$ generates a set of pixels in camera c's image that are likely to result in rays that would intersect ROI model point m at time t. Function $MakeRay\left(c, p\right)$ takes in a camera c and a pixel p, and returns the 3D ray from the camera center that passes through the given pixel in the camera image.

Function $GaussianEvaluator\left(m, ray\right)$, takes in a model point m and a ray. It constructs an *evaluator* object that stores three distances along the ray: the distance to the point where the evaluation should start, the distance to the point with the peak value, and the distance to the point where the evaluation should end. The start and end evaluation distances correspond to the two points where the ray intersects the uncertainty sphere around the surrogate model point. The evaluator objects have member functions $LessThan\left(x\right)$ and $MoreThan\left(x\right)$ to com-

Algorithm 5.2 Computing dynamic occlusion probabilities.

```
function $ComputeDynamicOcclusions(c, r, R, t)$
Input: camera $c$, current ROI $r$, set of ROIs $R$, time $t$
Output: occlusion probability $r.occluded$
$po = 0$, $wpo = 0$
for all model points $m$ of ROI $r$
    if $\sim IsVisible(c, m, t)$ continue
    $P = EvaluationPixels(c, m, t)$
    for each pixel $p \in P$
        $ray = MakeRay(c, p)$
        $e_m = GaussianEvaluator(m, ray)$
        $E = \emptyset$
        for all ROIs $r' \in R \setminus \{r\}$
            for all model points $m'$ of ROI $r'$
                if $\sim IsPointVisible(c, m', t)$ continue
                if $\sim IsPointClose(c, m', t, ray)$ continue
                $E = E \cup \{GaussianEvaluator(m', ray)\}$
        $pr = e_m.MoreThan(Dmin)$
        $wpo = wpo + pr$
        $po = po + pr \cdot ComputeOcclusion(e_m, E)$
$r.occluded = {}^{po}/_{wpo}$
```

pute the probabilities in Equations 5.4.4 given a distance x and use the trapezoidal rule for evaluating the integrals.

Function $ComputeOcclusion(e_m, E)$ computes an approximation of the integral in Equation 5.4.3 using the trapezoidal rule. Its inputs are the the evaluator object e_m for the currently evaluated model point m of ROI r and the set of evaluator objects E for all points m_i of ROIs r_i. The distances stored in all evaluator objects are sorted in increasing order, and the integral o is computed by calling the member functions $LessThan()$ and $MoreThan()$ of the evaluator objects with each of the sorted distances as the parameter.

Dynamic occlusion is very expensive to compute, because for each planning cycle it needs to look at all possible pairs of ROIs as seen from all cameras, and then loop through all the model points associated with each ROI and through all the pixels associated with each model point. This is why early exit conditions such as the ones in functions $IsPointVisible()$ and

IsPointClose() are necessary. *EvaluationPixels*() also provides opportunities for early exits by returning an empty set of pixels when the ROI is far from the camera, or the image of the model point occupies very few pixels. Some flexibility in the running time can also come from adjusting the number of sample pixels chosen to cast rays through for each model point. Additionally, the computation can be made to exit early and the ROI r being evaluated can be declared occluded when the occlusion probability accumulated for some of its model points m becomes greater than a given threshold. Furthermore, computing dynamic occlusions might be unnecessary: in the simulated experiments in Chapter 7, the effect of dynamic occlusions was negligible, mostly because their durations were usually shorter than the duration of a planning cycle, making them unlikely to occur during the discrete evaluation times at the end of each planning cycle.

5.5 Effects of Planning on Metric Behavior

In this section I show two examples of the impact of planning on metric behavior. The experiments use the same simulated setup as the ones in Section 5.4. These types of experiments help provide insights into how to further reduce the computational complexity by reducing the number of *captures* (start times and dwell lengths) proposed for evaluation when generating candidate plans.

5.5.1 Duration of First Capture

Under the assumption that cameras do not capture during transitions, making a transition has a similar effect to occlusion: there is no measurement from the camera to incorporate into the metric and reduce its value. In this experiment, Camera 1 and Camera 3 are fixed, while Camera 2 is set to cover the target trajectory in 2 captures, separated by a transition that takes 2 seconds. Figure 5.17 shows the metric values evaluated over 20 seconds, plotted against the duration of Camera 2's first capture, or the time when Camera 2's transition is scheduled to start.

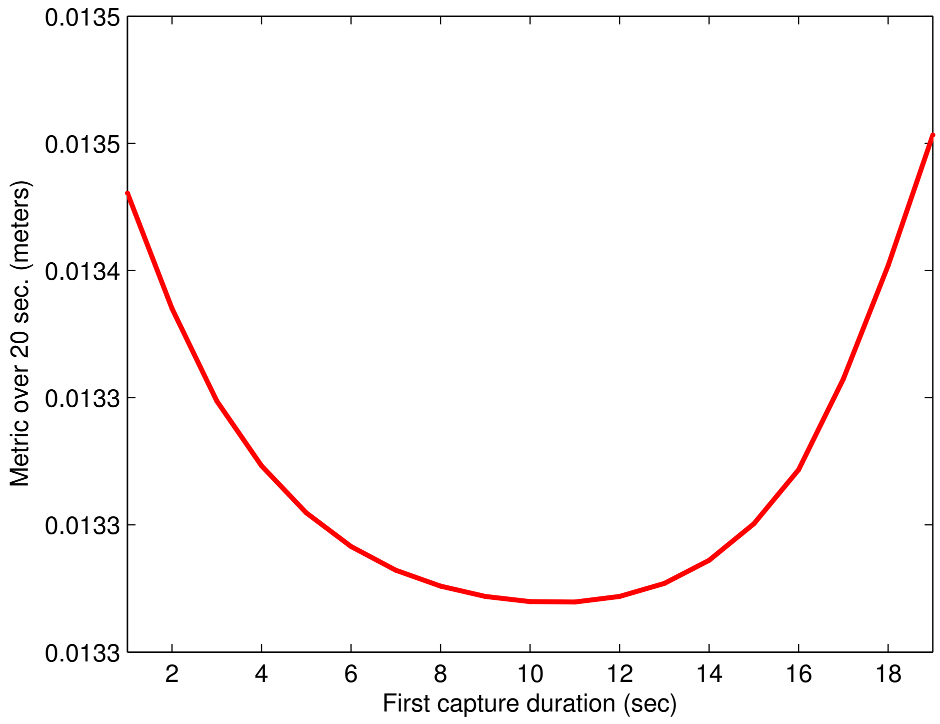


Figure 5.17: Effect of first capture duration on metric values.

Initially, the metric values decrease as the initial high uncertainty is reduced by capturing longer and longer. The second capture starts with an already low metric value and simply incorporates more measurements to maintain it low. As Camera 2's first capture duration increases, its field of view during that capture has to increase as well, leading to higher metric values. The second capture becomes shorter and shorter, and while having a decreasing field of view helps, it does not reduce the metric values enough. The shape of the curve reveals that if there are 2 captures before the horizon, the best break-down is to have their durations be balanced.

5.5.2 Coordinating Camera Transitions

Depending on how many cameras are available, the decision when to plan a transition can affect the metric values in different ways. The experiment shown in Figure 5.18 illustrates the impact of coordinating camera transitions. Camera 1 and Camera 3 are scheduled to cover the ROI trajectory in two captures, separated by a transition that takes 2 seconds. The surface plots show the metric values evaluated over 20 seconds, plotted against the time when the camera transitions are scheduled to start.

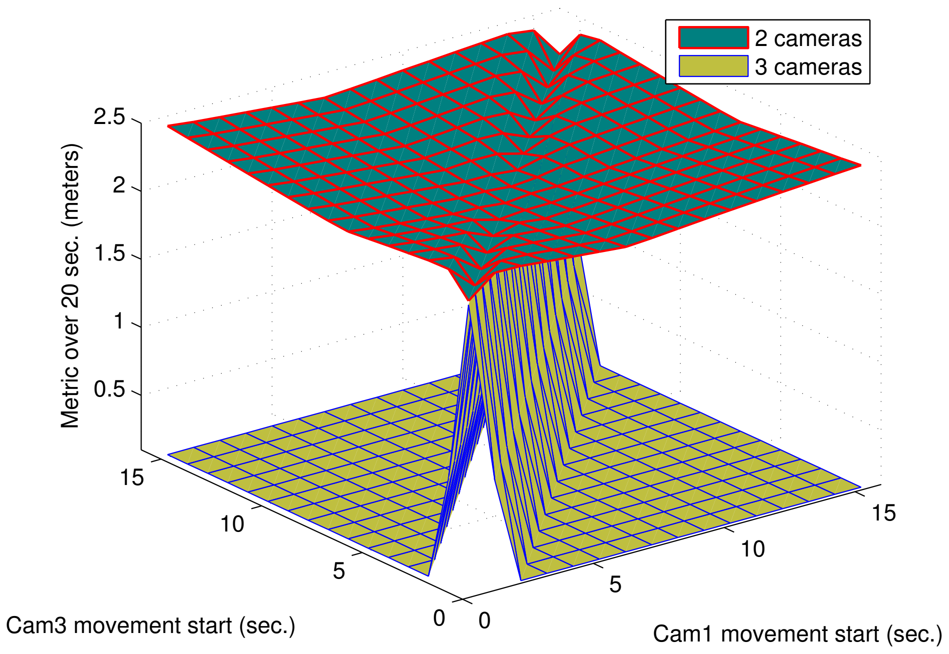


Figure 5.18: Effect of coordinating camera transitions on metric values.

In a first run, Camera 2 is disabled. The metric values are shown in the higher surface plot in Figure 5.18. The shape of the surface reveals that balanced capture intervals are favored. The lowest metric values are achieved when the two cameras transition simultaneously, appearing as a dip on the diagonal of the surface plot. This is expected: if the transitions do not overlap, during the transition of either camera, the other camera is left to capture the ROI by itself. The experiment in Section 5.4.5 has shown that a single camera capturing by itself can only play a small part in reducing the metric value. When the transitions overlap, there is a short period with no measurements, but the increase in uncertainty over this period is amortized over time by having both cameras capture the rest of the time.

In a second run, Camera 2 is enabled and zoomed out to capture for the entire duration of the experiment. The metric values are shown in the lower surface plot in Figure 5.18. The shape of the surface reveals that balanced capture intervals are still favored (there is still a slight slope toward the diagonal), but having Cameras 1 and 3 transition simultaneously actually leads to higher metric values that appear as a crest on the diagonal of the surface plot. The reason for this reversal is the fact that if Camera 1 and Camera 3 were transitioned simultaneously, they would leave Camera 2 capturing the target by itself, while if they transitioned at different times, there would always be at least two cameras (either Camera 1 and Camera 2 or Camera 3 and Camera 2) capturing the ROI.

5.6 Comparison with Related Work

Early camera performance metrics dealt with image resolution and how 2D image errors propagate into 3D [WSH98, OM02]. When trying to account for other factors that influence camera performance, previous research efforts ran into difficulties aggregating heterogeneous factors, such as image resolution and occlusion, into a single number suitable for use as a performance metric. Elements of probability theory have helped bring some heterogeneous factors together [CD00, Che02, CD08], but there were still performance factors that could not be integrated this way. Other approaches used empirically-based functions to aggregate heterogeneous factors [KYL+08].

I started investigating performance metrics in [IWM08] by extending the work by Allen [All07] to incorporate many of the performance factors encountered when using active cameras. Steady-state uncertainty easily accommodated many performance factors, but could not account for differences when planning over different planning horizons. The information theory-based approach pioneered by Denzler et al. evolved from static systems and using mutual information in [DB01] to using conditional entropy in [DZN03], and to extensions to multiple cameras and multiple steps into the future in [DWN06]. Their ideas presented a solution to the shortcomings of steady-state uncertainty, and my work on incorporating other performance factors was applicable to their performance metric. Finally, to avoid the problems encountered in practice when evaluating the metric only at the horizon, I aggregate the metric values over time instead.

Chapter 6

Camera Control Method

In Chapter 3, I have shown that exhaustive exploration of both the camera settings and camera to ROI assignments spaces is intractable. In Chapter 4, I looked at several strategies to make the exploration tractable, and have shown that the number of available strategies is larger for the camera to ROI assignments space. In this chapter I describe an implementation of a camera control method that explores the camera to ROI assignments space, and makes use of elements from the strategies in Section 4.2 to arrive at a system that can control cameras in real-time on current hardware. This implementation is but one point in the space of possible trade-offs between performing a more comprehensive search and achieving real-time performance.

The most effective strategy in Section 4.2 was decomposing the problem into subproblems and grouping the ROIs inside each subproblem. My camera control method implements this strategy, resulting in two components: *global assignment* and *local planning*. The global assignment component groups ROIs into *agencies* based on proximity and assigns cameras to each agency. The local planning component is run at the level of each agency, and is responsible for finding the best plans for all the cameras assigned to that agency. Another advantage of decomposing the problem into subproblems is the opportunity to run the two components of my approach at different frequencies: for example, the global assignment component can be run once every *N* cycles, while the local planning component can be run once per cycle at the level of each agency. For simplicity, and without loss of generality, in this chapter I discretize time in units of length equal to the longest possible duration of a planning cycle, which I set to $lCycle = 1$ second.

6.1 Notation

Before presenting the algorithms that comprise my method, I introduce a few notation elements. I use standard sets notation: $\{\ldots\}$ is a set, $\emptyset$ is the empty set, $\subset$ represents a subset, $\in$ represents membership, $\equiv$ represents equality, $\neq$ represents inequality, $\setminus$ represents set difference, $\cup$ represents set union, and $\cap$ represents set intersection. Sets are denoted by capital letters like A, C, PS, R, R' and set members by small letters like a, c, p, r. A set member can be selected by its number. For example, $C[n]$ is the n-th camera in set C.

Additionally, I use several variables, including the following:

- a plan $P_{\ldots}$ is denoted as a set of configurations over time, to which standard set operations can be applied,
- $P_{\ldots}[start \ldots stop]$ is plan $P_{\ldots}$ between times $start$ and $stop$,
- P_a is the plan for all cameras in agency a, and $P_{a,c}$ is the plan for camera c in agency a,
- $a.CurrentPlan$ is the current plan for agency a,
- $r.trajectory[t]$ represents the surrogate model of ROI r at time t, from which the top and $bottom$ points can be selected,
- R_a is the set of ROIs in agency a,
- C_a is the set of cameras assigned to agency a,
- $C_{avail.}$ is the set of available cameras,
- $c.useful$ is a Boolean variable that specifies whether a camera c has been designated as useful or not,

- $c.TransitionDuration$ and $c.UninterruptibleDwellDuration$ are the transition and uninterruptible dwell durations for camera c,
- $c.AspectRatio$ is the aspect ratio of camera c,
- *c.choices* is the set of capture choices (start time and duration) for camera c.

6.2 Global Assignment

The global assignment component accomplishes two tasks: grouping ROIs into agencies and assigning cameras to agencies. I describe both tasks in detail in the following subsections.

6.2.1 Forming Proximity-Based Agencies

In order to reduce its complexity, I break down the optimization problem into subproblems, each assigned to a separate agency. Each agency is concerned with the locally-optimal capture of a number of ROIs and is assigned a number of cameras. The experiment in Section 5.4.8, which showed that the metric values increase with distance, confirms the intuition that proximity is a good criterion for breaking down the problem into subproblems. I cluster together ROIs that are close to each other and predicted to be heading in the same direction. The method I employ uses predicted ROI trajectories to cluster the ROIs into a minimum number of non-overlapping clusters of diameter at most D, and builds agencies to solve the local optimization problem for the ROIs inside each cluster. Standard clustering algorithms such as k-means ([TSK05], Chapter 8) are not immediately applicable, because the ROI trajectories are dynamic. Additionally, the membership of all ROI clusters needs to exhibit hysteresis to help keep the assignments stable. When an agency's ROI membership changes, all cameras assigned to it need to reevaluate their current plans. Moreover, available cameras currently assigned to other agencies might contribute more to the modified agency, so their possible contributions need to be evaluated as well. If these evaluations result in changes in plans or assignments, cameras need to transition, and end up spending less time capturing, which usually results in higher metric values. To avoid these problems, I use a *minimal change* heuristic, consisting of 3 steps:

1. assign all unassigned ROIs to the agencies closest to them; form new agencies for isolated ROIs;

2. if any agency has become too large: iteratively remove ROIs from it until it becomes small enough; assign removed ROIs to the agencies closest to them as in step 1; remove any empty agencies;

3. if any two agencies are close enough to each other, merge them into a single agency.

Algorithm 6.1 describes the process in more detail.

Procedure $AddToCloseAgencies(r, D, H, A)$ tries to add ROI r to the agency in A that is closest to it. If no agency is close enough, a new agency is created. Algorithm 6.2 presents the procedure.

6.2.2 Greedy Assignment of Cameras to Agencies

In Sections 4.2.5 and 4.2.6, assigning ROIs to subproblems is done heuristically to avoid evaluating under-performing assignments. A good heuristic must be employed in order to minimize the risk of not finding the global optimum. I use a greedy heuristic to assign cameras to each agency, based on their potential contribution. The heuristic iteratively finds the camera-agency assignment that best improves the metric value for the agency. Improvement is measured using the ratio m_{after}/m_{before}, and the best improvement *bi* corresponds to the smallest ratio. The initial value for *bi* is $1+\varepsilon$, where ε is a very small number. If a camera does not improve the metric for any agency, it is not used for the duration of the current cycle. The final greedy plan is compared with a plan obtained by simply prolonging the current plan up to the planning horizon, and the greedy assignment is only applied if the greedy plan is better.

Algorithm 6.1 Minimal change ROI clustering.

```
procedure $ClusterROIs(D,H,R,A)$
Input: cluster diameter $D$, horizon $H$, ROI set $R$, agencies set $A$
$PredictROITrajectories(T,H)$
for all unassigned ROIs $r \in R$
    $AddToClosestAgency(r,D,H,A)$
for all agencies $a \in A$
    repeat
        find ROIs $\{r_1,r_2\} \subset R_a$ most distant over $H$
        if $Distance(r_1,r_2,H) > D$
            $R' = R_a \setminus \{r_1,r_2\}$
            if $Distance(r_1,R',H) > Distance(r_2,R',H)$
                $r = r_1$
            else
                $r = r_2$
            $R_a = R_a \setminus \{r\}$
            $AddToClosestAgency(r,D,H,A)$
            if $R_a \equiv \emptyset$
                $A = A \setminus \{a\}$
    until $Distance(r_1,r_2,H) \leq D$
repeat
    find agencies $a_1, a_2$ closest to each other
    if $Distance(a_1,a_2,H) \leq D$
        $a = Merge(a_1,a_2)$
        $A = A \setminus \{a_1,a_2\} \cup \{a\}$
until $Distance(a_1,a_2,H) > D$
```

Algorithm 6.2 Adding a ROI to the agency closest to it.

```
procedure $AddToClosestAgency(r,D,H,A)$
Input: ROI $r$, cluster diameter $D$, horizon $H$, agencies set $A$
$d_{min} = \infty$
for all agencies $a \in A$
    $d = Distance(r,a,H)$
    if $d < D$
        $R_a = R_a \cup \{r\}$
        $d_{min} = d$
        $a_{min} = a$
if $d_{min} \equiv \infty$
    $a = CreateAgency()$
    $A = A \cup \{a\}$
else
    $a = a_{min}$
$R_a = R_a \cup \{r\}$
```

However, previously unassigned cameras are assigned to the agency they most contributed to. Proximity is evaluated in function *IsClose* () to only try assigning cameras to nearby agencies.

Algorithm 6.3 describes the greedy assignment process. Only *available* cameras are taken into account at every planning cycle. A camera is considered available if its state for the next planning cycle is: an interruptible dwell, the start of a transition or occluded. A number of functions called by the algorithm are described in the following pages. The number of assignments evaluated is $nAgencies \cdot nCams_{avail.}(nCams_{avail}+1)/2$.

If the number of assignments evaluated by Algorithm 6.3 is too large for real-time evaluation, a slightly different heuristic can be employed to reduce it to $nAgencies \cdot nCams_{avail.}$ by only looking at the best improvement for each camera. While faster, the results of this alternative assignment process depend on the order in which cameras are assigned. Algorithm 6.4 describes this alternative assignment process.

Plans corresponding to each camera-agency assignment are generated heuristically by assuming the worst-case scenario: the camera is repeatedly set to transition, then capture for as

Algorithm 6.3 Greedy assignment of cameras to agencies.

```
procedure $GreedyAssignment\,(A, C, H)$
Input: set of agencies $A$, set of cameras $C$, horizon $H$
$C_{avail.} = FindAvailableAgents\,(C)$
$P_{current} = \emptyset$
for all agencies $a \in A$
    for all cameras $c \in C_a$
        $P_a = BuildPlan\,(c, a, 1, H, true)$
        $P_{current} = P_{current} \cup P_a$
$P_{greedy} = P_{current}$
while $C_{avail.} \neq \emptyset$
    $bi = 1 + \varepsilon$
    $m_{before} = Metric\left(A, P_{greedy}, H\right)$
    for all cameras $c \in C_{avail.}$
        $c.useful = false$
        for all agencies $a \in A$
            if $\sim IsClose\,(c, a, H)$ continue
            $P_a = BuildPlan\,(c, a, 1, H, c.agency \equiv a)$
            $AddCameraToAgency\,(c, a, P_a)$
            $m_{after} = Metric\left(A, P_{greedy} \cup \{P_a\}, H\right)$
            $RemoveCameraFromAgency\,(c, a)$
            $i_{c,a} = {}^{m_{after}}\!/\!{}_{m_{before}}$
            if $i_{c,a} \leq bi$
                $bi = i_{c,a}$
                $c_{best} = c$
                $a_{best} = a$
                $P_{best} = P_a$
                $c.useful = true$
    if $bi \leq 1$
        $AddCameraToAgency\,(c_{best}, a_{best}, P_{best})$
        $P_{greedy} = P_{greedy} \cup P_{best}$
        $C_{avail.} = C_{avail.} \setminus \{c_{best}\}$
    for all cameras $c \in C_{avail.}$
        if $\sim c.useful$
            $C_{avail} = C_{avail.} \setminus \{c\}$
```

Algorithm 6.4 A faster greedy assignment of cameras to agencies.

```
procedure $GreedyAssignment2(A,C,H)$
Input: set of agencies $A$, set of cameras $C$, horizon $H$
$C_{avail.} = FindAvailableAgents(C)$
$P_{current} = \emptyset$
for all agencies $a \in A$
    for all cameras $c \in C_a$
        $P_a = BuildPlan(c,a,1,H,true)$
        $P_{current} = P_{current} \cup P_a$
$P_{greedy} = P_{current}$
for all cameras $c \in C_{avail.}$
    $bi = 1 + \varepsilon$
    $m_{before} = Metric(A, P_{greedy}, H)$
    $c.useful = false$
    for all agencies $a \in A$
        if $\sim IsClose(c,a,H)$ continue
        $P_a = BuildPlan(c,a,1,H,c.agency \equiv a)$
        $AddCameraToAgency(c,a,P_a)$
        $m_{after} = Metric(A, P_{greedy} \cup \{P_a\}, H)$
        $RemoveCameraFromAgency(c,a)$
        $i_{c,a} = m_{after}/m_{before}$
        if $i_{c,a} \leq bi$
            $bi = i_{c,a}$
            $a_{best} = a$
            $P_{best} = P_a$
            $c.useful = true$
    if $c.useful$
        $AddCameraToAgency(c, a_{best}, P_{best})$
        $P_{greedy} = P_{greedy} \cup P_{best}$
    $C_{avail.} = C_{avail.} \setminus \{c\}$
```

Algorithm 6.5 Generating heuristic plans.

```
function $BuildPlan(c, a, start, stop, prolong)$
Input: camera $c$, agency $a$, planning $start$ and $stop$ times, bool $prolong$
Output: heuristic plan $P$
$t = start$
$P_a = a.CurrentPlan$
if $prolong$
    $P_a = ProlongCurrentCapture(P_a)$
    $t = EndOfFirstCaptureTime(P_a)$
$P_a = P_a[1 \ldots t]$
while $t < stop$
    $t_{FoV} = WidestFoVCaptureTime(c, a, t, stop)$
    $t_{occl} = NextOcclusionStart(c, a, t, stop)$
    $e = min(t_{FoV}, t_{occl})$
    $P_a[t \ldots e] = PlanTransitionAndCapture(c, a, t, e)$
    if $t_{FoV} < t_{occl}$
        $t = e$
    else
        $t = NextOcclusionEnd(c, a, t, stop) - c.TransitionDuration$
return $P_a$
```

long as possible, with a field of view as wide as possible. While other scenarios, in which the camera captures for shorter intervals, may result in better outcomes due to tighter fields of view (see Section 5.4.6), the goal of this heuristic is only to help assess the potential contribution a camera can have to the capture of the ROIs in the agency it is being assigned to. If a camera is assigned to an agency it is already a member of, the heuristic offers the option of prolonging the current capture until the visibility of ROIs in the agency drops below a given threshold. The heuristic also takes into account occlusions, and plans transitions during occlusions to minimize the time when the camera is not capturing (see Sections 5.4.2 and 5.5.2). This heuristic can also be used as a contingency in local planning if the number of cameras in the agency is too large. Algorithm 6.5 presents the process in detail.

Once the start cycle and capture duration are decided for a planning step in a camera's plan, I use geometric reasoning to compute the corresponding camera settings. To speed up computation, I use a 2.5D approximation. The *PlanTransitionAndCapture*() function, illustrated in Figure 6.1 and described in Algorithm 6.6, builds a point set *PS* from the points at the bottom and the projections of the points at the top of the surrogate model of each ROI onto the *XY* plane as seen from the camera, and computes the camera parameters from 2D angles and distances in the plane. I have found that this approximation provides satisfactory results in practice.

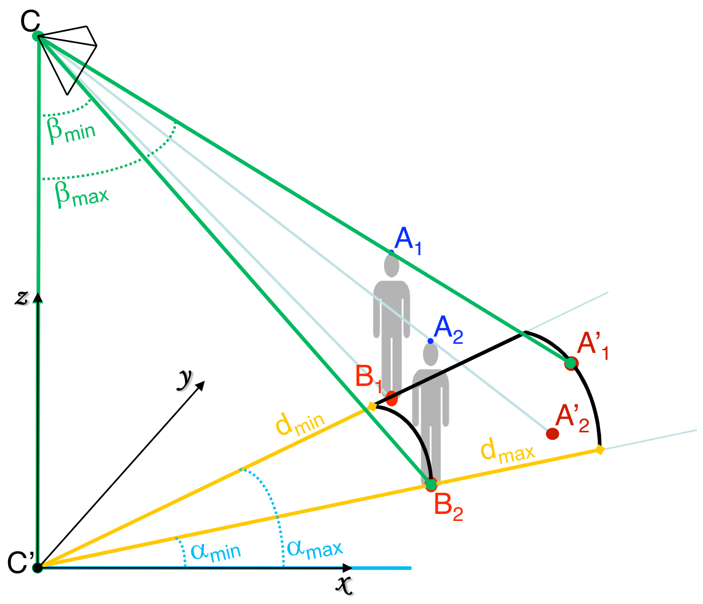


Figure 6.1: Computing optimal parameters for a camera.

It is worth noting that two small changes in this global assignment approach make it general enough to apply to camera selection as well. A scenario likely to be encountered in practice is that the camera infrastructure installed at a site might have both F fixed cameras and A active cameras, as well as a number D of devices to record the video streams coming from the cameras. If there are not enough devices to record all the streams $(D < A + F)$, a modified Algorithm 6.3 can be used to decide which streams to record. The changes would simply be to return after assigning D cameras and to not call the *PlanTransitionAndCapture*() function for fixed cameras.

Algorithm 6.6 Computing camera settings corresponding to a plan step.

```
function $PlanTransitionAndCapture(c, a, start, stop)$
Input: camera $c$, agency $a$, planning $start$ and $stop$ times
Output: PTZ settings for plan step $P[start \ldots stop]$
$PS = \emptyset$
$t = start$
while $t \leq stop$
    for all ROIs $r \in R_a$
        $p_1 = ProjectOntoPlane(r.trajectory[t].top)$
        $p_2 = r.trajectory[t].bottom$
        $PS = PS \cup \{p_1, p_2\}$
    $t = t + 1$
compute angles $\alpha_{min}$ and $\alpha_{max}$
compute distances $d_{min}$ and $d_{max}$
compute angles $\beta_{min}$ and $\beta_{max}$
$Hfov = \alpha_{max} - \alpha_{min}$
$Vfov = \beta_{max} - \beta_{min}$
if $Hfov/Vfov > c.AspectRatio$
    $Hfov = c.AspectRatio * Vfov$
$Pan = (\alpha_{max}+\alpha_{min})/2$
$Tilt = (\beta_{max}+\beta_{min})/2$
$Zoom = Hfov/FoV_{1\times}$
```

6.3 Local Planning

The local planning component applies the heuristic in Section 4.2.6: all ROIs inside each agency are grouped together into a single ROI group. All cameras assigned to each agency capture all the ROIs in the ROI group, and no further camera-ROI assignment decisions are made at this level. The planning decisions made at the local level are when and for how long each camera should capture, and when it should transition. The number of possible plans is reduced by using the myopic and lazy planning strategies in Sections 4.2.2 and 4.2.1: only the first planning step is explored in detail (subsequent planning steps are generated heuristically). Additionally, the decision whether to keep the current configuration or to transition toward a new one is made only during the first cycle in the first planning step. Static occlusions do not depend on the camera parameters, so they are precomputed for each camera in the variable *c.visib.* and used to further reduce the number of possible plans in the $ComputeStaticVisibilities()$ function. Other restrictions, such as how many planning steps of minimum length can fit before the planning horizon, can be taken into account as well in the $FindChoices()$ function. All remaining possible plans are explored exhaustively using backtracking. Algorithm 6.7 presents the local planning procedure, Algorithm 6.8 details the backtracking algorithm, and Algorithm 6.9 shows a simple example of how the set of choices can be computed.

6.4 Comparison with Related Work

Many previous camera control methods were targeted at specific tasks and used methods adapted from domains such as real-time scheduling. Other methods try to solve the problem by optimizing a performance metric. The camera control method presented in this chapter belongs to the latter category. It improves on previous methods by specifically taking into account complex performance factors relevant to high-level computer vision tasks such as 3D reconstruction, while still attaining real-time performance. The list below enumerates a few

Algorithm 6.7 Generating local plans.

```
function $LocalPlanning(C_a, a, H)$
Input: camera set $C_a$, agency $a$, planning horizon $H$
Output: local plan $P_a$
$C_{avail.} = FindAvailableAgents(C_a)$
for all cameras $c \in C_{avail.}$
    $c.visib. = ComputeStaticVisibilities(c, a, 1, H)$
    $P_{a,c}[1 \ldots H] = BuildPlan(c, a, 1, H, true)$
    $c.choices = FindChoices(c, a, 1, H)$
$m_{best} = Metric(\{a\}, P_a, H)$
$P_{best} = P_a$
$BackTrack(a, C_{avail.}, 0, 0, P_{best}, H)$
return $P_{best}$
```

Algorithm 6.8 Exhaustive exploration using backtracking.

```
function $BackTrack(a, C, cn, n, P_{best}, H)$
Input: agency $a$, camera set $C$, current camera number $cn$,
current choice number $n$, current best plan $P_{best}$, horizon $H$
Output: current local best plan $P_{best}$
$cn = cn + 1$
if $cn < size(C)$
    $c = C[cn]$
    for all choices $i \in c.choices$
        $P_{best} = BackTrack(a, C, cn, i, P_{best}, H)$
else
    $P_a[1, c.choices(n)] =$
    $PlanTransitionAndCapture(c, a, 1, c.choices(n))$
    $P_a[c.choices(n+1), H] = BuildPlan(c, a, c.choices(n) + 1, H, true)$
    $m = Metric(\{a\}, P_a)$
    if $m < m_{best}$
        $m_{best} = m$
        $P_{best} = P_a$
return $P_{best}$
```

Algorithm 6.9 Finding possible plan choices.

```
function $FindChoices(c, a, start, stop)$
Input: camera $c$, agency $a$, planning $start$ and $stop$ times
Output: set of choices $S$
$t = start + c.TransitionDuration + c.UninterruptibleDwellDuration$
$t_{FoV} = WidestFoVCaptureTime(c, a, t, stop)$
$S = \emptyset$
while $t \le t_{FoV}$
    if $\sim IsOccluded(c, t)$
        $S = S \cup \{t\}$
return $S$
```

of what I consider to be desired characteristics of a camera control method:

- *performance metric*: whether the approach evaluates the performance of a camera configuration;
- *deals with occlusion*: whether the approach deals with static and dynamic occlusions;
- *minimize dead time*: whether the approach attempts to minimize the time spent by cameras transitioning instead of capturing;
- *temporal trade-off*: whether the approach considers and compares present and future configurations;
- *trajectory prediction*: whether the approach reacts to changes in the ROI trajectories;
- *camera setting change time*: whether the approach takes into account the time it takes for camera settings to change;
- *new ROIs*: whether the approach is able to deal with new ROIs;
- *N ROIs per camera*: whether the approach can assign one camera to view multiple ROIs;

- *N cameras per ROI*: whether the approach can assign multiple cameras to view a single ROI.

Table 6.1 summarizes the extent to which some of the related camera control approaches mentioned in Chapter 2 have the desired characteristics listed above. The control method described in this chapter has all the desired characteristics.

	Approaches				
Charac-teristic	[NCB01b, NCB03, BNE+06]	[dBP05, BdBP05]	[LMD05, LMD07]	[QT05a, QT07]	[CDBF04]
perf. metric	yes	no	yes	yes	no
occlusions	no	no	yes	no	no
minimize dead time	yes	yes	yes	yes	yes
temporal trade-off	yes	no	yes	no	no
traj. prediction	yes	no	yes	no	yes
settings change time	yes	yes	yes	yes	yes
new ROIs	no	static	static	no	yes
N ROIs per camera	no	no	no	no	yes
N cameras per ROI	yes	no	yes	yes	no

	Approaches				
Charac-teristic	[KYL+08]	[DND05, SR08a]	[BGV+09]	[MU02]	this research
perf. metric	yes	yes	yes	limited	yes
occlusions	no	yes	static	no	yes
minimize dead time	yes	no	no	no	yes
temporal trade-off	yes	yes	no	no	yes
traj. prediction	yes	yes	yes	yes	yes
settings change time	yes	no	yes	no	yes
new ROIs	static	no	yes	yes	yes
N ROIs per camera	no	no	yes	yes	yes
N cameras per ROI	yes	yes	yes	yes	yes

Table 6.1: Comparison with related work.

Chapter 7

Experimental Results

In this chapter, I present experimental results using the prototype implementation described in Chapter 6. Each experiment showcases different aspects of my performance metric and control method in action.

7.1 USMC Training Exercise

7.1.1 Experiment Setup

The first set of experimental results uses filtered data from a training exercise performed by members of the United States Marines Corp (USMC) captured on-site at the Sarnoff Corporation in Princeton, NJ. It illustrates how my prototype implementation can control 6 PTZ cameras to observe 6 ROIs, each ROI enclosing a single exercise participant. I used the game engine-based simulator from [TCB07] to run multiple simulations using the same input data. The simulator provides means of controlling a number of virtual cameras in a simulated environment and retrieving images from them.

Figure 7.1 shows an overview of the exercise site, modeled in the game level editor Hammer [Val06].

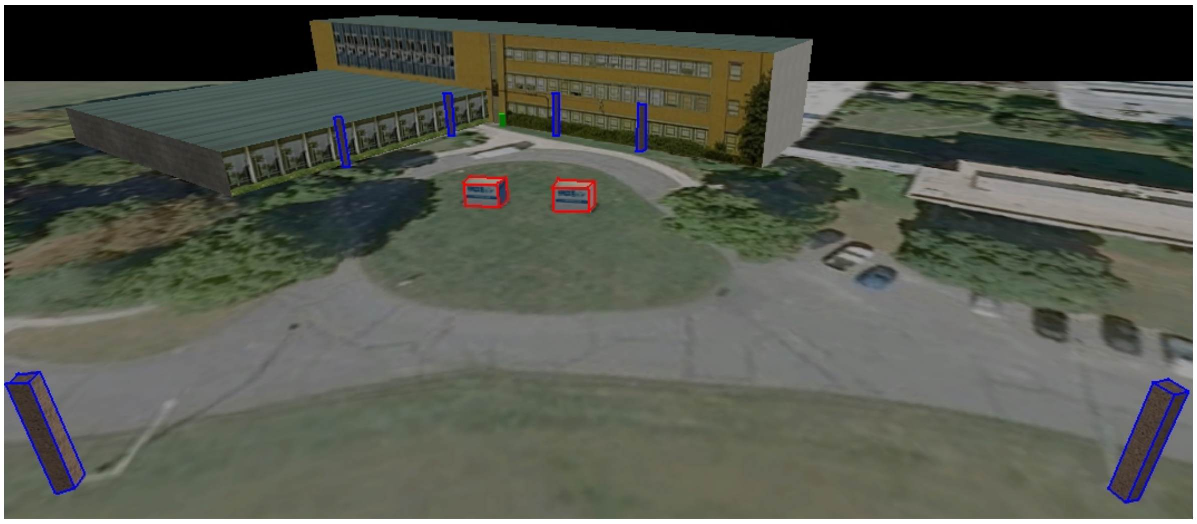


Figure 7.1: Overview of the USMC training exercise site as modeled in the game level editor Hammer [Val06].
The pillars outlined in blue represent the places where the virtual cameras are mounted: 4 cameras along the building walls and 2 cameras on light poles in the parking lot. There are also 2 shipping containers, outlined in red, which serve as props during the exercise.

Trajectories were first captured as GPS measurements over time. The approach described in [BGV$^+$09] was used to capture images from 2 PTZ cameras and refine the trajectory estimates. I filtered the trajectory data to smooth out noise, sampled it at every second and used the samples as input for the prototype control method implementation. During an experiment run, I use the 3D points on the trajectories to generate simulated 2D measurements in each virtual camera and 3D (longitude, latitude, altitude) simulated GPS measurements. I add precomputed noise to each measurement before incorporating it into the performance during the sequential Kalman filter evaluation process described in Section 5.1.2.2.

The surrogate model for each exercise participant consistted of 2 cubic regions, 1 meter on the side, stacked on the vertical axis. The coordinates of the center of each region were included in the state, and a PV state model was used in the Kalman filter-based performance metric evaluation process. To compute the visibility of each cubic region, I shot a ray through 7 points associated with the region, and sampled the precomputed visibility map at the intersection point between the ray and the plane of the visibility map. The 7 sample points were the region center and the 6 points at ± 0.5 meters in the x, y and z axis directions. The point visibilities were aggregated as follows: the center point had a weight of $1/2$, and the 6 exterior points each had a weight of $1/12$.

The simulator does not provide means of programmatically controlling the virtual characters, so I used a plugin architecture from [Cas07] to extend its functionality. I wrote a custom plugin script to act as a server to which client applications can connect and send commands. I implemented commands to retrieve a list of available participants, move a participant to a new position, change a participant's orientation and retrieve the current position and orientation of a participant.

For each simulated camera image, the simulator also provides ground truth (silhouettes, bounding boxes, total and visible pixel counts) for the virtual characters in the image. This feature is aimed at testing image processing algorithms, but I used it to compute the visibility of a virtual character placed in a particular position as seen from each of the 6 cameras used in my experiments. I sequentially placed a virtual character at positions on a 2D regular grid spanning the area, aimed all cameras at it, and queried the simulator for the ground truth pixel counts of the virtual character as seen by each camera. Using this process, I precomputed the visibilities over the area where the exercise took place and stored them as a per-camera *visibility maps*. The visibility at a point (x,y) on the 2D grid for camera c was computed as:

$$V_c^{(x,y)} = \frac{PC_c^{visible}}{PC_c^{total}} \tag{7.1.1}$$

where PC_c^{total} and $PC_c^{visible}$ are the total and visible pixel counts in camera c's image. Figure 7.2 shows a top-down view of the camera positions and the visibility maps for all 6 cameras, overlaid on top of a satellite image.

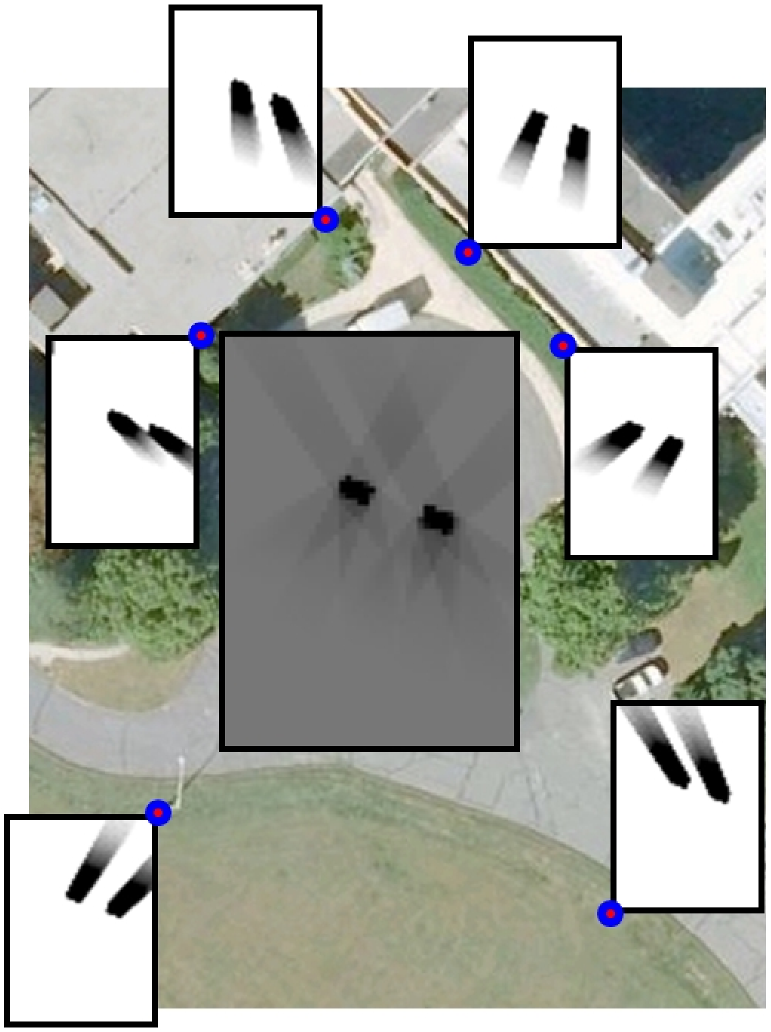


Figure 7.2: Top-down view of positions and visibility maps for the 6 cameras used in the USMC exercise experiments.
Cameras are shown as blue circles, each accompanied by its precomputed visibility map. Brighter values denote higher visibility. The aggregated visibility is also shown as a grayscale image overlaid on a top-down satellite view of the site and aligned to match the area where the exercise took place. The 2 shipping containers appear as dark spots of zero visibility in the aggregated visibility image, and cast “shadows” in each camera’s visibility map.

The exercise scenario was for a squad of 4 Marines to patrol, cross a danger zone between 2 buildings, perform 2 cordon searches on a civilian, neutralize a sniper threat, and move out to secure an area. Figure 7.3 shows top-down views of the participants tracks at important times during the scenario.

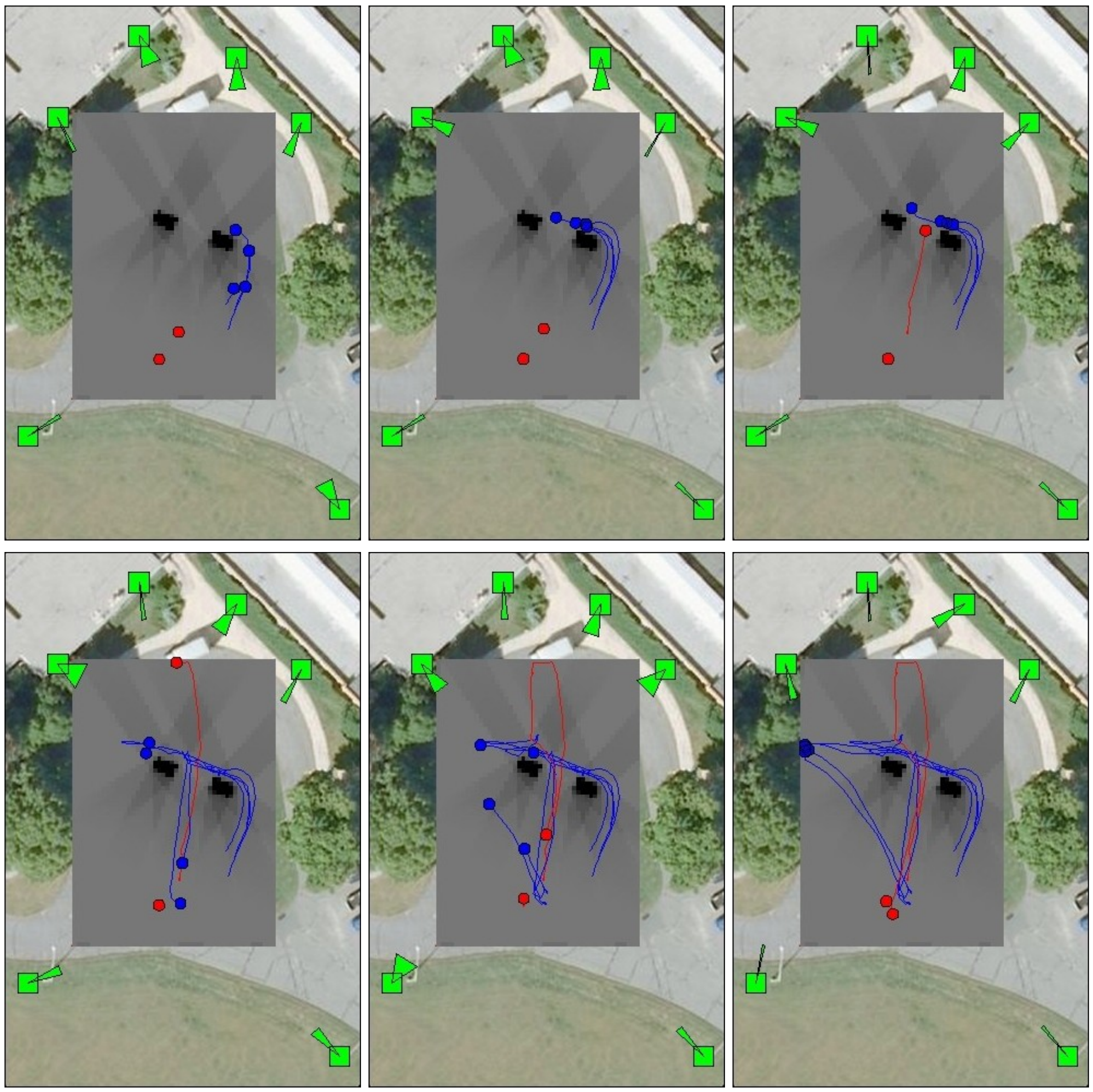


Figure 7.3: Top-down views of participant tracks in USMC training exercise scenario. Left to right, top to bottom: (1) start, $Time = 9$. (2) cross danger zone, $Time = 24$. (3) cordon search, $Time = 47$. (4) neutralize threat, $Time = 90$. (5) move out, $Time = 148$. (6) end, $Time = 170$. Marine tracks are shown in blue, civilian and sniper tracks are shown in red. Camera positions and orientations are shown in green. To provide context, in each image all elements are overlaid on top of a satellite view and an aggregated visibility map.

7.1.2 Results

Using this simulated setup, I was able to run several experiments and test various combinations of parameters for my approach on the same input ROI trajectories. I set the transition duration to 2 seconds and the minimum capture duration (uninterruptible dwell) to 4 seconds for all cameras. The experiment duration was 170 seconds.

For experiments 1...4, I show the following:

- a plot of the metric values over time (the metric quantifies uncertainty in the state, so lower values are better);
- a timeline of the camera states and agency memberships (each camera has two color-coded time strips: the top strip for agency membership, shown as shades of gray, and the bottom strip for camera states, shown as colors);
- a stacked bar plot of the camera states, to better show when cameras are capturing and how many cameras are capturing at a particular time.

Camera states are one of the following:

- *capturing* (green) - during camera captures;
- *occluded* (red) - when the aggregate ROI visibility is below the visibility threshold;
- *moving* (blue) - when the camera is transitioning between captures;
- *moving while occluded* (cyan) - when the camera is transitioning during a time interval when it would have been occluded;
- *unused* (yellow) - when the global assignment heuristic fails to find an agency that the camera can contribute to.

For experiments 5...8, I show a mesh plot of metric values over time under varying circumstances. For experiment 9 I show a combined plot of the metric values for 4 different situations.

7.1.2.1 Experiment 1: Default Settings

Under the default settings, I set the clustering diameter $D \simeq 22$ meters, the length of a planning cycle to $lCycle = 1$ second, and assume that the GPS signal is available every second. The visibility threshold is set to $T_v = 0.75$. Figure 7.4 shows the results.

The metric values are higher and noisy when the participants move: at the beginning (when all cameras are zoomed out cover all participants, $Time = 5 \ldots 15$), then during the neutralization of the threat (when participants are most spread out and move the fastest, $Time = 70 \ldots 105$), and finally when the Marines move out (when Marines are moving fast again, $Time = 140 \ldots 160$). Conversely, the metric values are lower and flat when the ROIs are stationary, during the 2 cordon searches starting at $Time = 50$ and $Time = 130$. Camera 4 is unused for 1 second, when the greedy assignment method described in Section 6.2 and Algorithm 6.3 fail to find an agency it would improve upon.

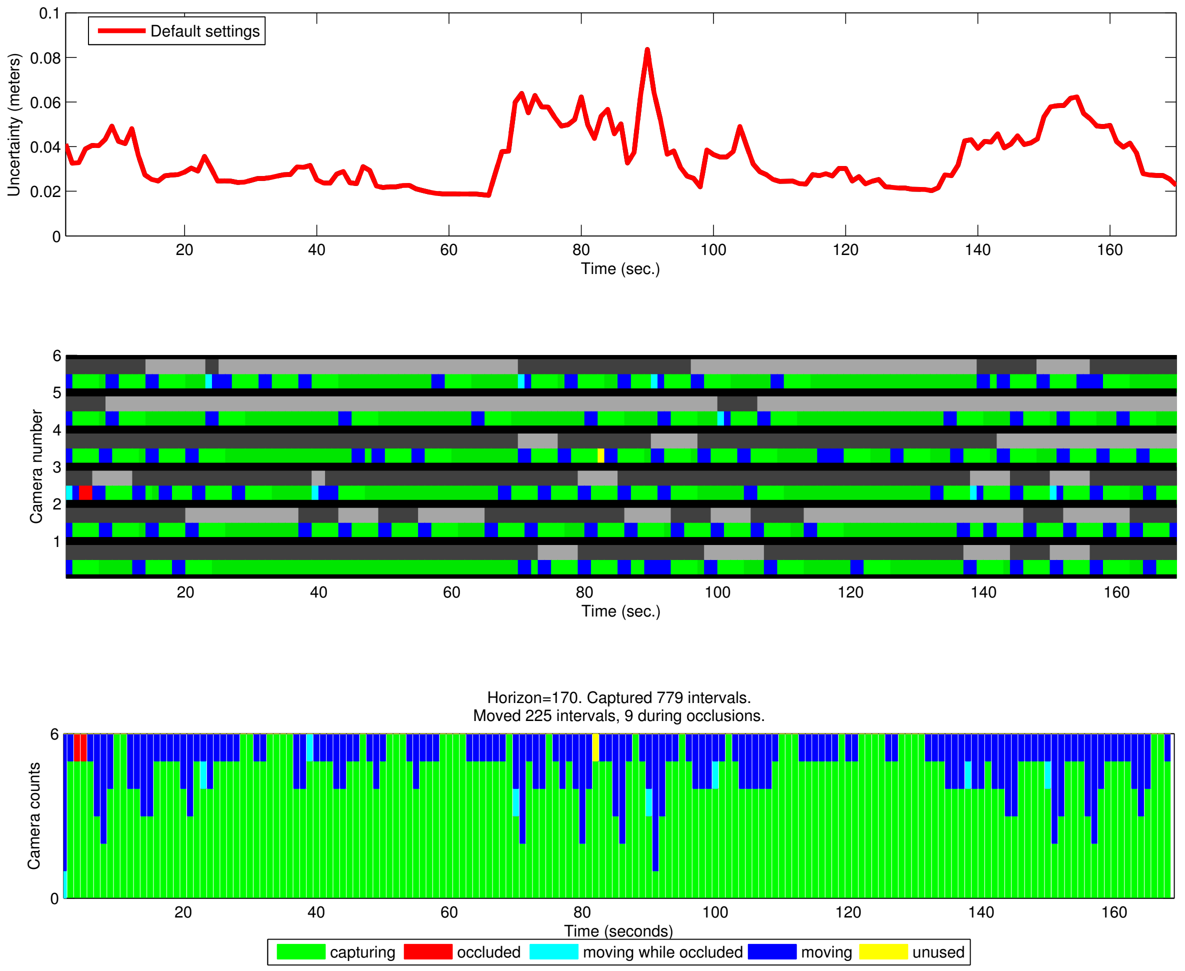


Figure 7.4: Metric, camera states and agency memberships over time for the default experiment settings.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.

The following figures show how my camera control method automatically achieves desirable complex behaviors: reacting to predicted occlusions, adapting to changes in predicted ROI motion and coordinating transitions. Participants are grouped into agencies, shown as circles with varying shades of gray. Cameras assigned to a particular agency have their base represented as a square of the same shade of gray. Camera orientation is shown as a triangle color-coded by the camera state: green when capturing, blue when in transition. When capturing, camera fields of view are shown as white lines.

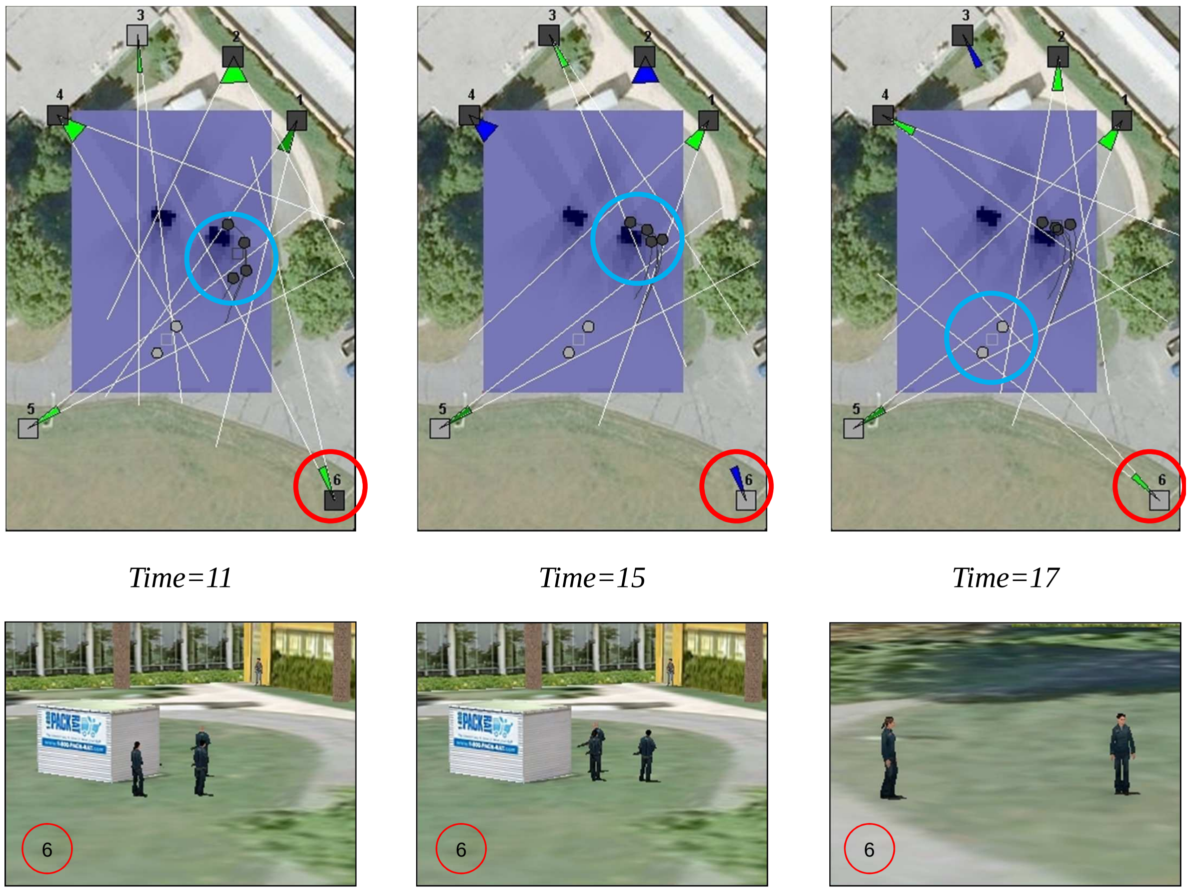


Figure 7.5: Reacting to predicted occlusions.
Top row: top-down views of cameras and exercise participants, overlaid on top of a satellite image and aggregated visibility map.
Bottom row: simulated images for camera 6.
The 4 Marines are grouped into an agency, and the civilian and the sniper into a second agency. At $Time = 11$, Camera 6 is capturing the 4 Marines, which are approaching the first building. At $Time = 15$, 2 of the Marines are already behind the first building, and the other 2 Marines are predicted to follow them, so Camera 6 is set to transition to where it can provide better coverage. At $Time = 17$, the transition has ended, and Camera 6 is assigned to a different agency and capturing the civilian and the sniper.

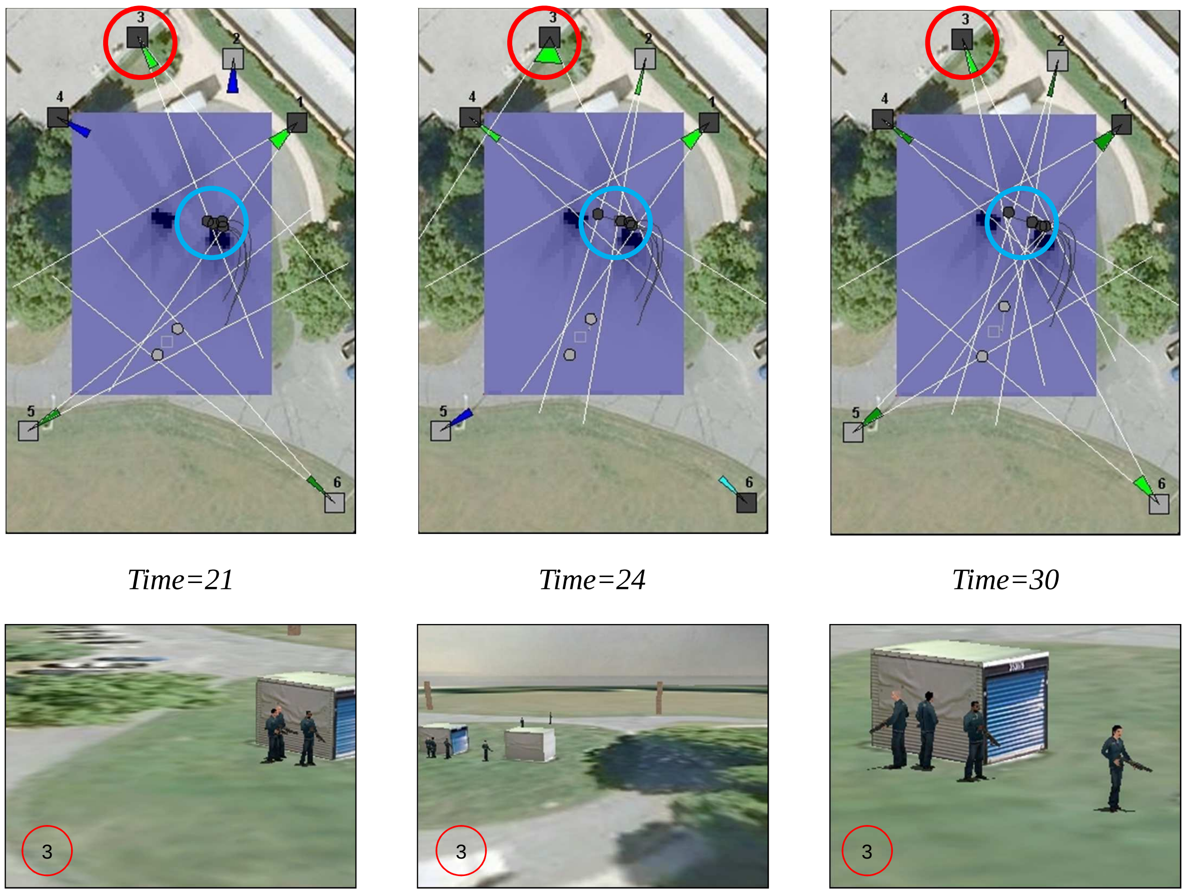
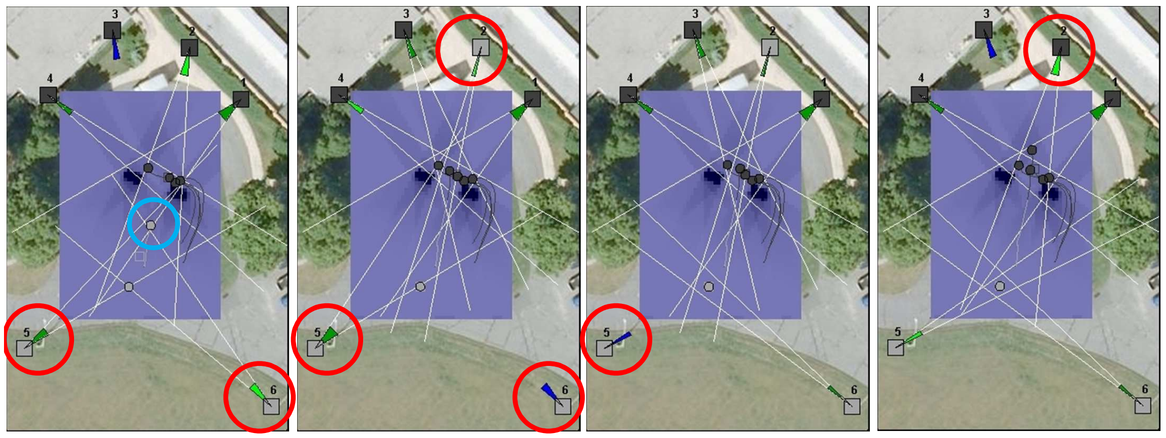


Figure 7.6: Adapting to changes in predicted ROI motion.
Top row: top-down views of cameras and exercise participants, overlaid on top of a satellite image and aggregated visibility map.
Bottom row: simulated images for Camera 3.
The 4 Marines are grouped into an agency, and the civilian and the sniper into a second agency. At $Time = 21$, the Marines are about to exit the field of view of Camera 3, and the first Marine is moving fast to cross the danger zone between the 2 buildings. To avoid losing track of the fast-moving Marine, at $Time = 24$, Camera 3 is set to zoom out in order to cover the predicted trajectory of the Marine. After crossing the danger zone, the Marine stops to cover the other 3 Marines behind him. Since the new predicted trajectory of the Marine is shorter, there is no need for Camera 3 to be zoomed out. At $Time = 30$, Camera 3 is zoomed back in to cover the Marines at higher resolution.



Time=42 *Time*=58 *Time*=65 *Time*=68

Figure 7.7: Coordinating transitions.
Top-down views of cameras and exercise participants, overlaid on top of a satellite image and aggregated visibility map.
At $Time = 42$, Cameras 5 and 6 cover the civilian and the sniper. As the civilian is heading toward the Marines for the cordon search, he exits the field of view of the cameras, but will soon be covered by the cameras covering the Marines. Cameras 5 and 6 can now zoom in to better cover the sniper. However, while either camera transitioned, it would leave the other camera covering the sniper by itself. Meanwhile, the Marines are covered by 4 cameras, one of which can help cover the sniper. At $Time = 58$, Camera 2 has been reassigned from covering the Marines to covering the sniper, and Camera 6 can begin to zoom in. At $Time = 65$, Camera 5 is zoomed in as well. Finally, at $Time = 68$, once both Cameras 5 and 6 are zoomed in and covering the sniper, Camera 2 is reassigned back to covering the Marines and the civilian.

The structure of the performance metric for this experiment, in terms of components such as the surrogate model used and the aggregation function is suitable for 3D reconstruction approaches such as the one by Li Guan [Gua10]. Figure 7.8 shows reconstruction results obtained by Li Guan using simulated images from 4 cameras captured at $Time = 56$, during the cordon search.

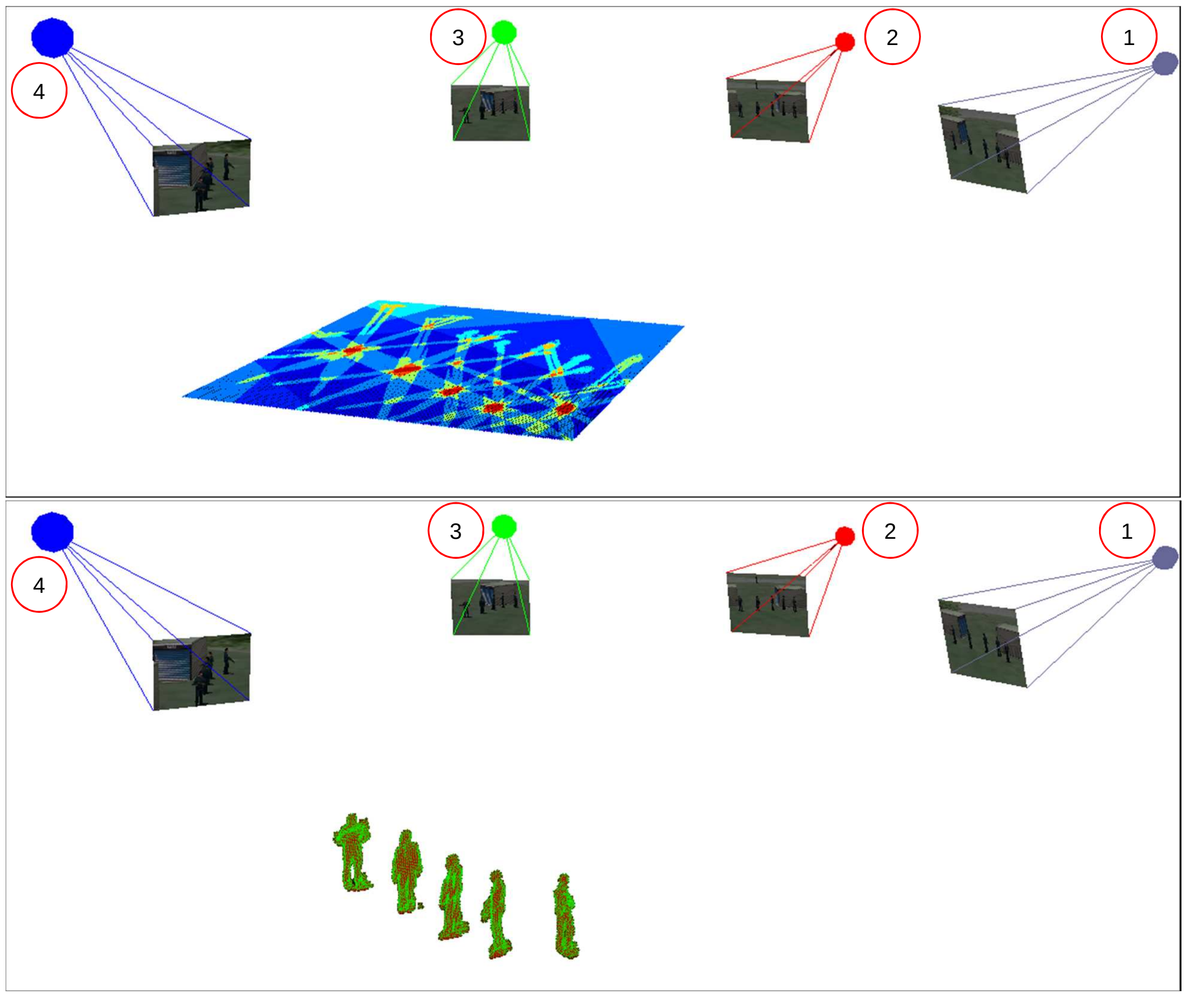
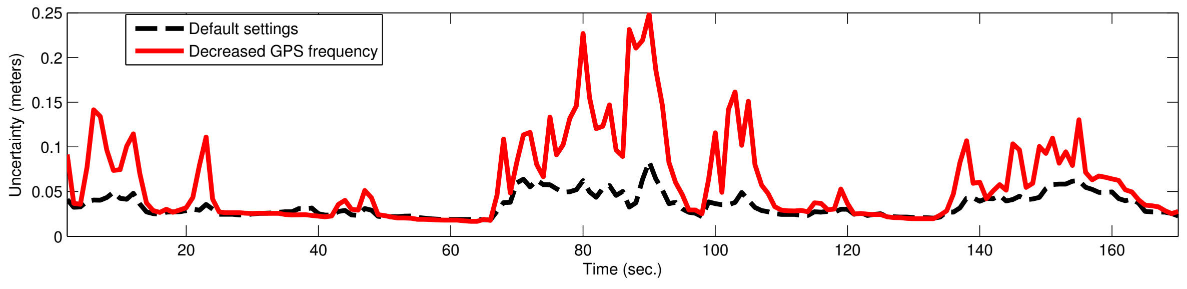
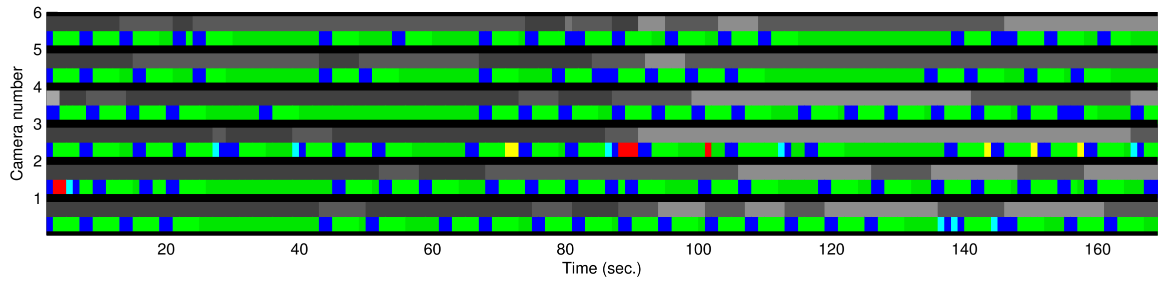


Figure 7.8: 3D reconstruction results using method from [Gua10].
Top: rendering of a single slice of the reconstructed volume showing occupancy probabilities.
Bottom: rendering of 3D reconstruction results obtained by thresholding.
Also shown are the poses and images of the cameras involved.

7.1.2.2 Experiment 2: Decreasing GPS Frequency

To test what would happen if the GPS signal was not available every second, I set its interval to 5 seconds. I simulated this situation by having the sequential Kalman filter evaluation only incorporate GPS measurements every 5^{th} cycle. Figure 7.9 shows the results.





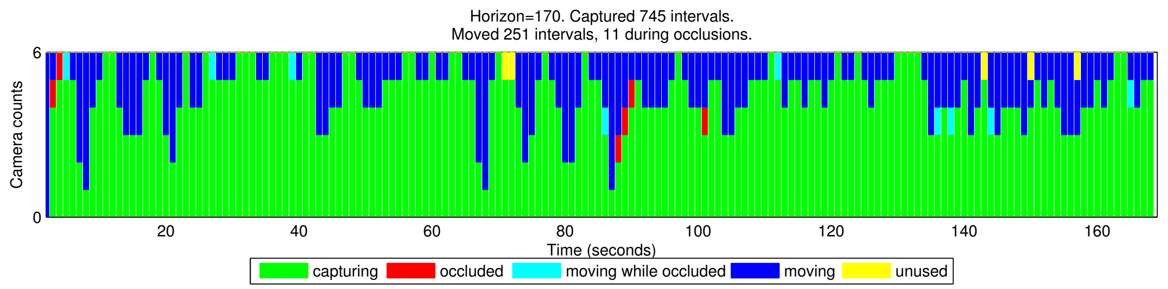


Figure 7.9: Metric, camera states and agency memberships over time for decreasing the GPS frequency.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.

Compared with the default settings, the metric values are higher and noisier. Also, due to the increased uncertainty, the clustering algorithm creates more agencies over time. Camera 3 ends up being unused for 5 seconds, when the greedy assignment method described in Section 6.2 and Algorithm 6.3 fail to find an agency it would improve upon.

7.1.2.3 Experiment 3: Eliminating the GPS

I also tested what would happen if the GPS signal was unavailable. I simulated this situation by not having the sequential Kalman filter evaluation incorporate any GPS measurements. The input tracks are still used to generate 2D measurements for each camera. Figure 7.10 shows the results. Note how at times 21, 40, 68 and 69 all cameras are transitioning.

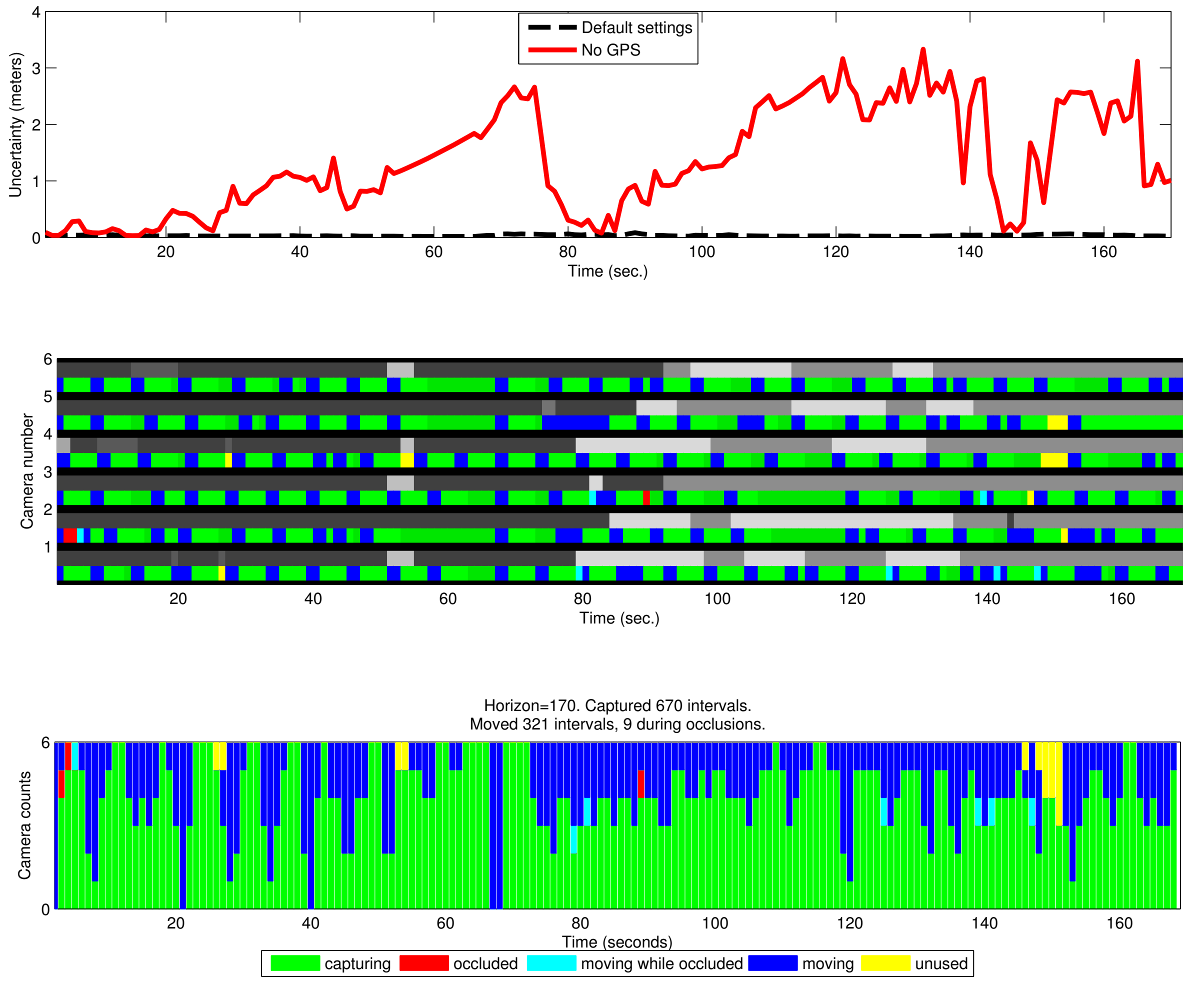


Figure 7.10: Metric, camera states and agency memberships over time for eliminating the GPS.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.

Compared with the default and decreased GPS frequency settings, the metric values are much higher. Due to the increased uncertainty in the ROI positions, the cameras are zoomed out more, and spend more time transitioning between captures. The clustering algorithm creates even more agencies over time (note the increased number of shades of gray on the timeline plot).

7.1.2.4 Experiment 4: Global Assignment Only

The local planning component can be time-consuming if due to an unbalanced assignment an agency ends up with too many cameras. I tested performance degradation when only the global assignments and their heuristic plans are computed at every planning cycle. In this experiment, I set the clustering diameter to $D \simeq 18$ meters. Figure 7.11 shows the results.

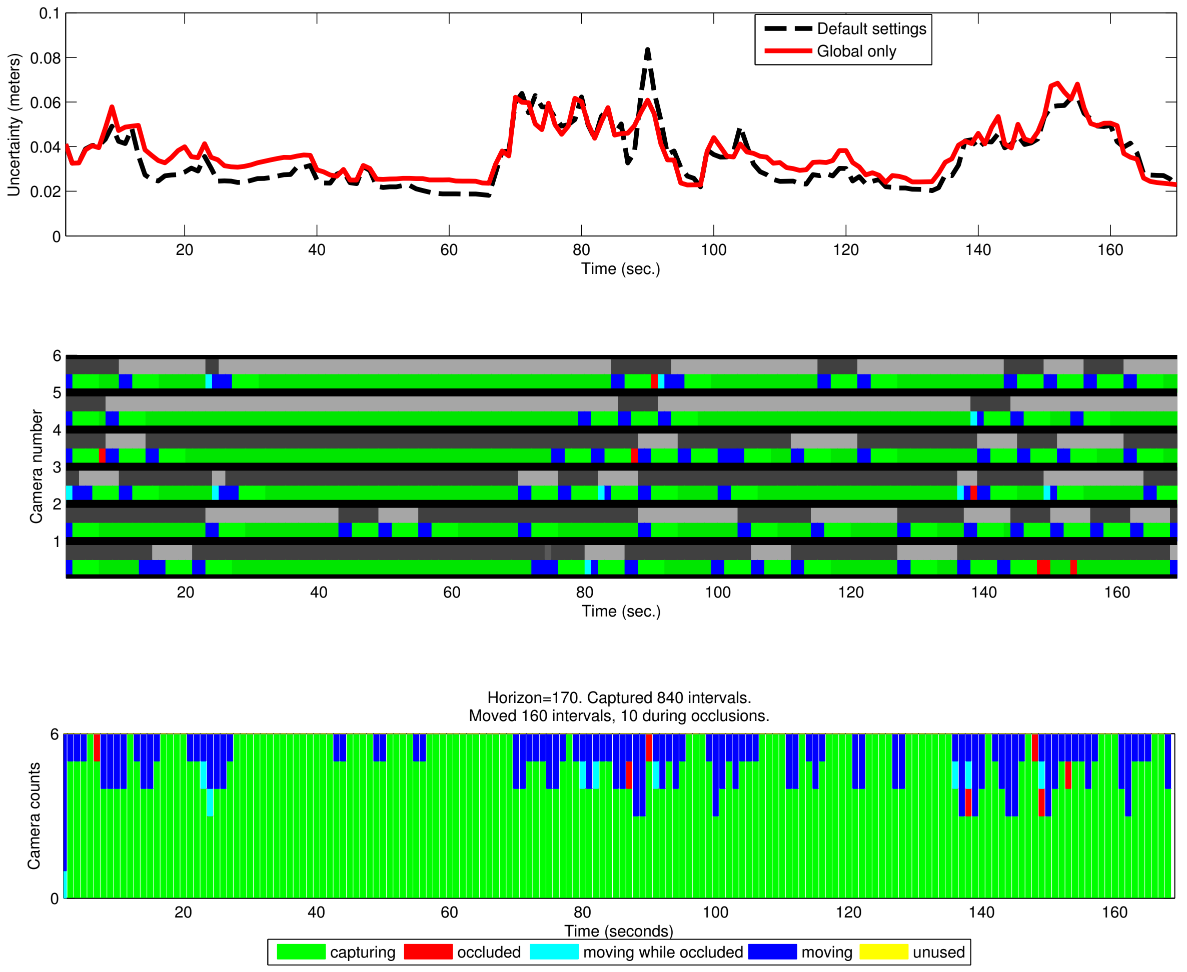


Figure 7.11: Metric, camera states and agency memberships over time for global assignment only.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.

The metric values are comparable with the values for the default settings, with little performance degradation. The global assignment process uses the *BuildPlan*() heuristic function described in Algorithm 6.5, which prolongs the current capture by leaving camera settings unchanged when the ROIs are still visible. This results in two behaviors with effects that cancel each other out: long, uninterrupted captures when the ROIs are stationary, and lower resolution due to cameras being more zoomed out.

7.1.2.5 Experiment 5: Changing the Visibility Threshold

One of the tunable parameters for my method, mentioned in Section 4.3, is the visibility threshold. In this experiment, I tested varying the visibility threshold between 0.1 and 1.0, while keeping all other settings constant as in the default settings experiment in Section 7.1.2.1. Figure 7.12 shows the resulting metric values over time for all visibility threshold values.

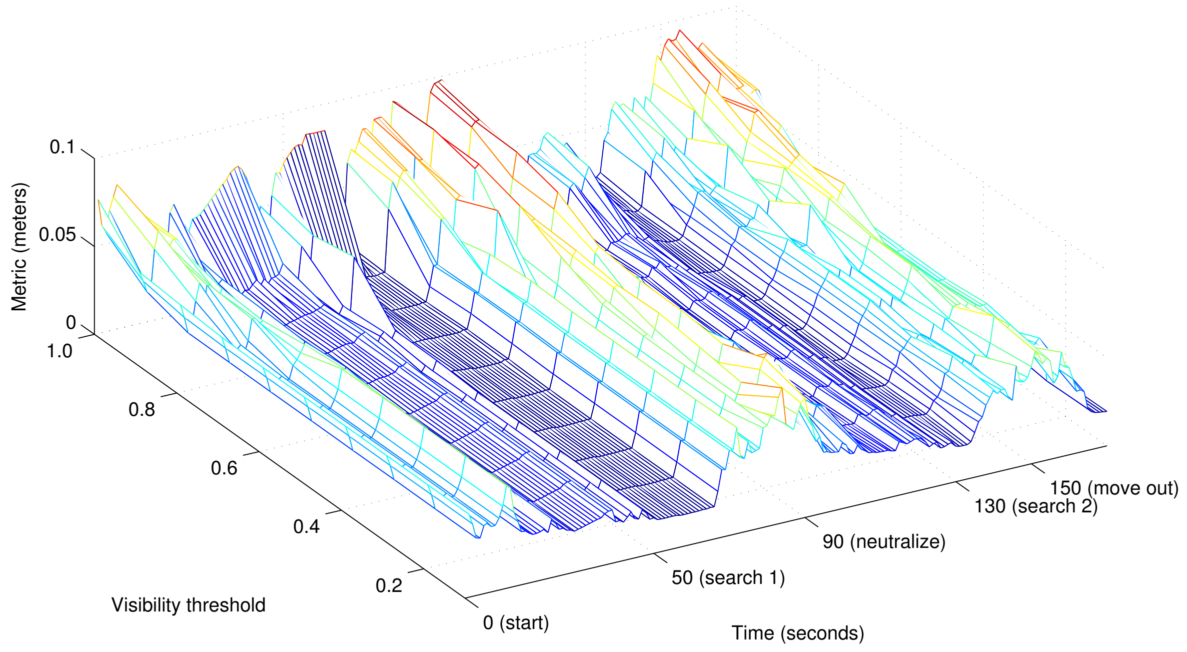


Figure 7.12: Metric over time when varying the visibility threshold.
The time axis shows important moments when the metric values are either consistently high or consistently low.

The general shape of the metric plot is similar to the curve in the default settings experiment in Section 7.1.2.1. As the visibility threshold increases, more occlusions are detected and taken into account by the control method in the *FindChoices*() function described in Algorithm 6.9. This also results in captures that are shorter and more fragmented, and an increase in the metric values overall.

7.1.2.6 Experiment 6: Changing the Planning Frequency

Another tunable parameter mentioned in Section 4.3 is the planning frequency. In this experiment, I tested varying the planning interval between 1 and 10 cycles, while keeping all other settings constant as in the default settings experiment in Section 7.1.2.1. When no planning was performed, I simply prolonged the current captures. Figure 7.13 shows the resulting metric values over time for all planning frequency values.

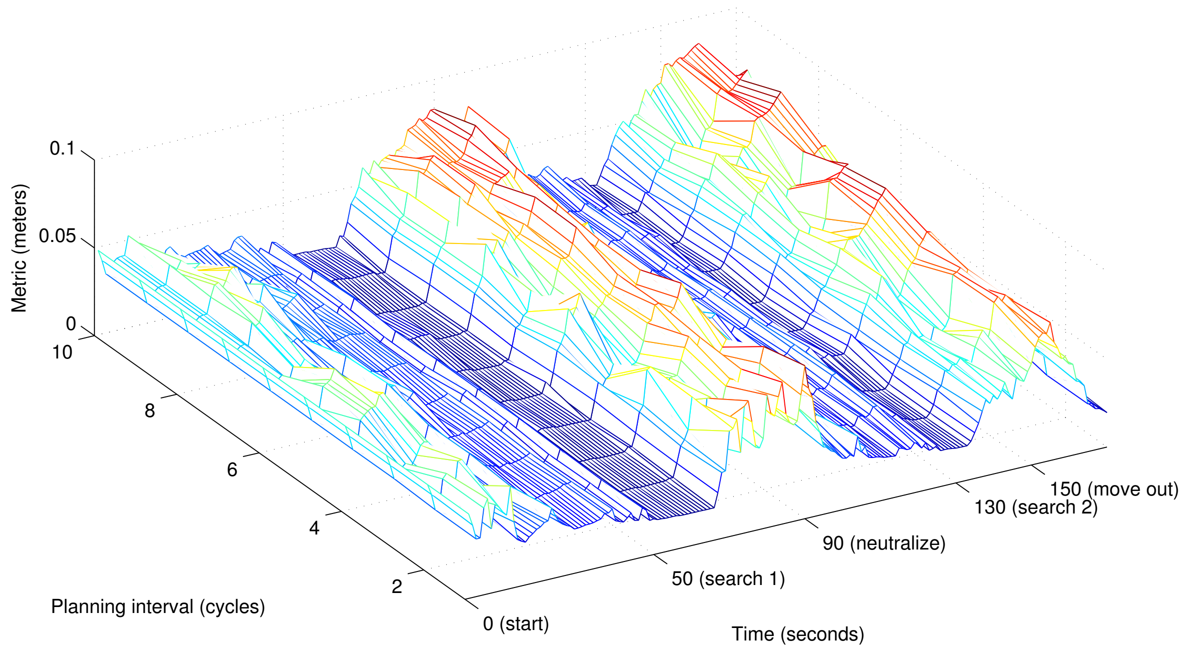


Figure 7.13: Metric over time when varying the planning frequency.
The time axis shows important moments when the metric values are either consistently high or consistently low.

Performing planning less often means less disruptions in camera membership for each agency, but also an increased number of times when wrong trajectory predictions lead to the ROIs not being captured. The general shape of the metric plot remains similar to the curve in the default settings experiment in Section 7.1.2.1. As expected, the metric values increase with increasing intervals between planning runs, because there are less and less opportunities for the control method to update the plans for each camera and correct for wrong trajectory predictions.

7.1.2.7 Experiment 7: Changing the Planning Horizon

Another tunable parameter mentioned in Section 4.3 is the planning horizon. In this experiment, I tested varying the planning horizon between 10 and 28 cycles, while keeping all other settings constant as in the default settings experiment in Section 7.1.2.1. Figure 7.14 shows the resulting metric values over time for all planning horizon values.

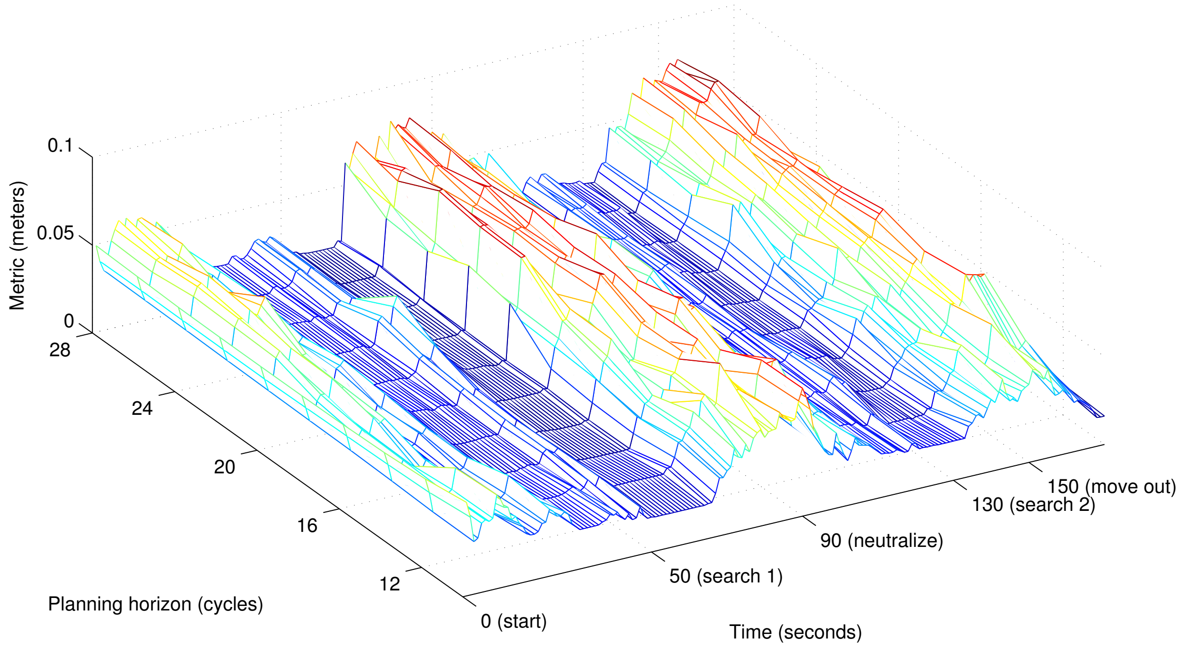


Figure 7.14: Metric over time when varying the planning horizon.
The time axis shows important moments when the metric values are either consistently high or consistently low.

The general shape of the metric plot stays similar to the curve in the default settings experiment in Section 7.1.2.1. As the planning horizon increases, the heuristic function *FindChoices*() described in Algorithm 6.9 results in cameras that are zoomed out to cover longer time intervals, less pixels occupied by the ROIs in the camera images, and increasing metric values overall.

7.1.2.8 Experiment 8: Changing the Clustering Diameter

Another factor that affects the performance of the camera control method is the clustering diameter used in Algorithm 6.1. In this experiment, I tested varying the clustering diameter between 2.5 and 25 meters, while keeping all other settings constant as in the default settings experiment in Section 7.1.2.1. Figure 7.15 shows the resulting metric values over time for all clustering diameter values.

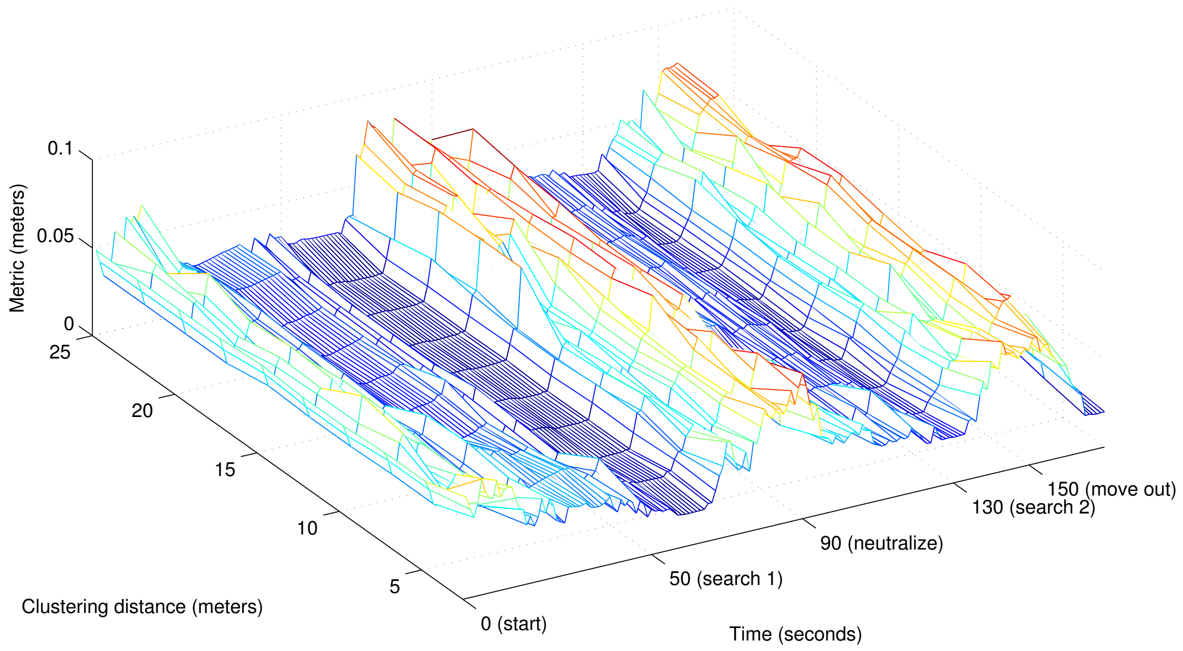


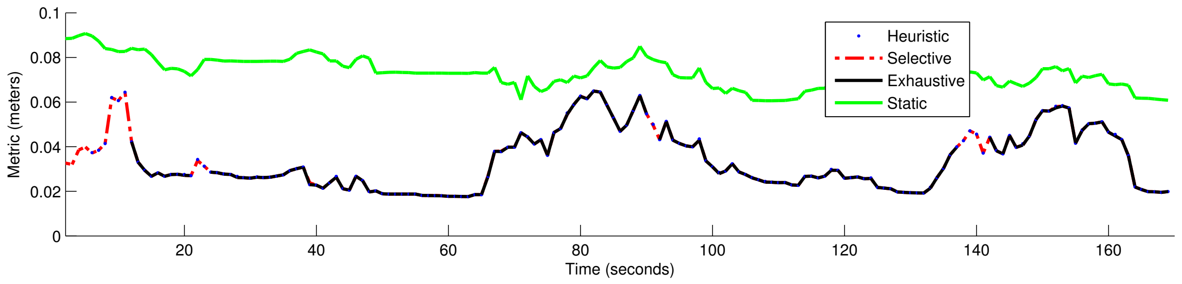
Figure 7.15: Metric over time when varying the clustering distance.
The time axis shows important moments when the metric values are either consistently high or consistently low.

The general shape of the metric plot is still similar to the curve in the default settings experiment in Section 7.1.2.1. Similarly to Experiment 7 in Section 7.1.2.7, as the clustering diameter increases, the heuristic function $FindChoices()$ described in Algorithm 6.9 results in cameras that are zoomed out to cover larger clusters, less pixels occupied by the ROIs in the camera images, and increasing metric values overall.

7.1.2.9 Experiment 9: Changing the Local Planning Exploration Method

One of the most important requirements for an active camera control method is that it runs in real-time. For agencies with many cameras, local planning can become very time-consuming. In this experiment, I tested three different ways of coming up with choices for each camera, while keeping all other settings constant as in the default settings experiment in Section 7.1.2.1. The first exploration method is *heuristic exploration*, which simply fills each camera's plan using the $BuildPlan()$ function described in Algorithm 6.5. The second exploration method is *selective exploration* which, when the number of cameras in an agency is below a threshold, explores all choices given by the $FindChoices()$ function described in Algorithm 6.9, and uses the heuristic plan otherwise. This method is used by default, since it allows exhaustive exploration whenever possible while still guaranteeing real-time performance. The third exploration method is *exhaustive exploration,* which explores all possible choices. Since this method takes a long time to run, I only ran it when all agencies had at most 5 cameras, or 151 out of the total 170 cycles. Finally, for comparison, I show the *static* case, where all cameras are fixed, aimed at the center of the environment, and zoomed out to $\times 1$, or a field of view of 60°. Throughout the experiment, I set the planning horizon to 15 seconds and evaluated the metric over the entire horizon.

Figure 7.16 shows the resulting metric values over time for the 3 exploration methods and the static case. Figure 7.16 (bottom) shows the difference between the selective and heuristic methods and the exhaustive method.



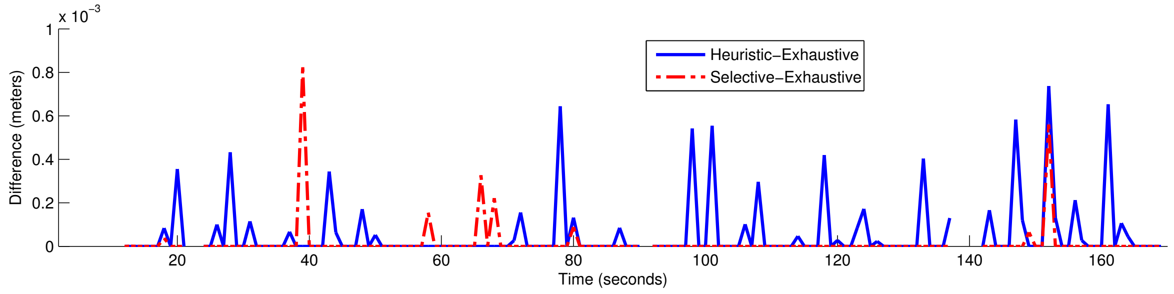


Figure 7.16: Metric over time when changing the local planning exploration method.
Top: Metric values over time for various exploration methods.
Bottom: Differences in metric values over time.

As expected, the metric values in the case of static cameras are significantly higher, but part of the price paid for the lower resolution is offset thanks to the fact that there are no transitions. The 3 curves corresponding to the exploration methods are similar in shape to the curve in Experiment 1 in Section 7.1.2.1. They are also very close to each other, to the point of being indistinguishable at the scale in Figure 7.16 (top). For comparison, the heuristic method arrived at the same result as the exhaustive method for 109 out of 151 cycles, or 72% of the time, and the selective method for 143 cycles, or 94% of the time. For the remaining times when the results were different, the average difference in the metric values was 0.01m for the heuristic method and 0.0087m for the selective method.

7.2 Simulated Subway Station

7.2.1 Experiment Setup

For a second set of simulated experiments, I used synthetic tracks generated by the crowd simulation method from [GLM10]. The crowd simulation method outputs 2D tracks of virtual people walking from given started positions to specified goals, while avoiding obstacles such as walls or other people. It can also produce 2D and 3D renderings of crowd simulations.

In my experiments, the simulated environment is a subway station. The experiment scenario is for 25 people, each enclosed in a separate ROI, to first walk towards a billboard where train schedules are displayed, linger for a specified amount of time, then walk towards the exit of their choice. Figure 7.17 shows a rendering of the station.

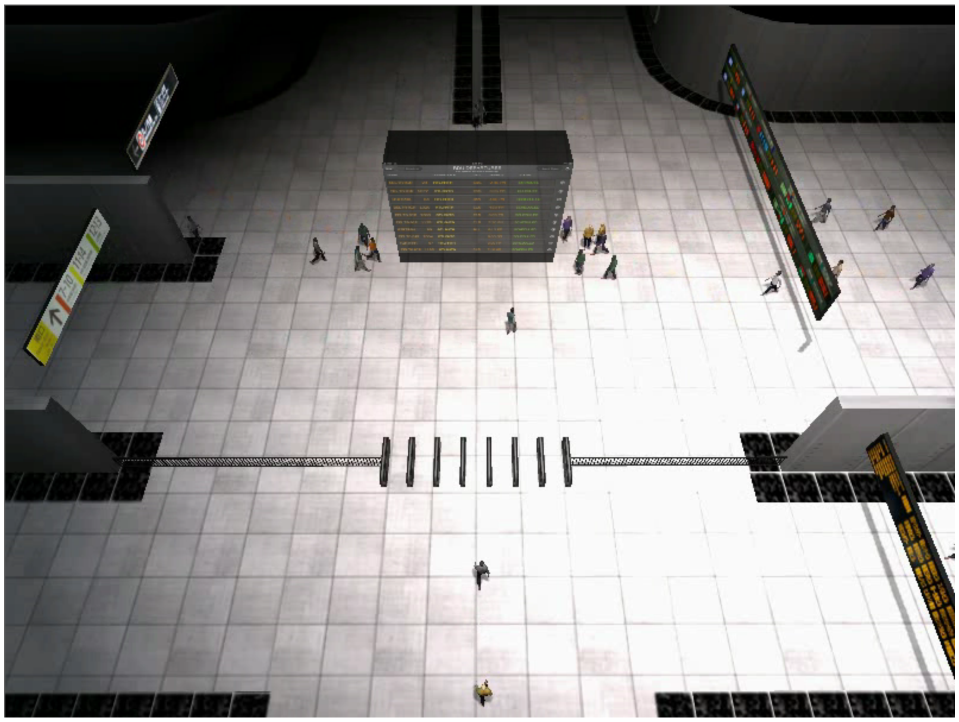


Figure 7.17: 3D rendering of virtual subway station using the simulator from [GLM10].

I placed 10 cameras along the walls. Figure 7.18 shows a top-down view of the station, with the camera positions highlighted.

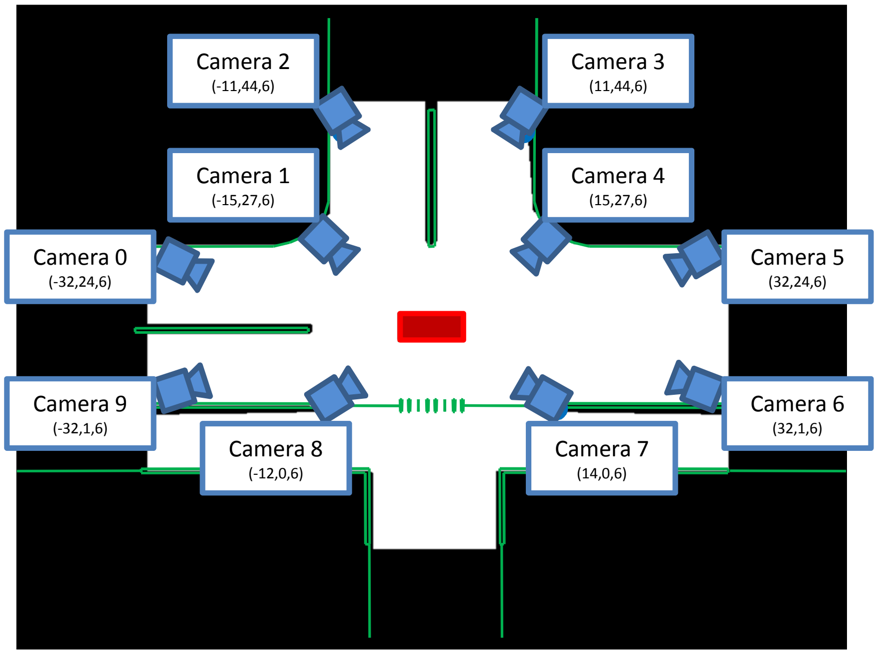


Figure 7.18: Overview of subway station with camera positions.
The area where the people are being captured is shown in white. The cameras are shown in blue, the walls are shown in green, and the billboard is shown in red.

For this set of experiments, computing the visibility maps using an approach like the one in the USMC training exercise experiments would have required implementing new functionality in the crowd simulation rendering application. Visibility would have taken too long to evaluate for all the cameras over the entire environment. Instead, I manually drew the visibility maps for each camera given its position and the positions of static occluders in the environment (the station walls and the billboard). Figure 7.19 shows the resulting visibility maps next to each camera.

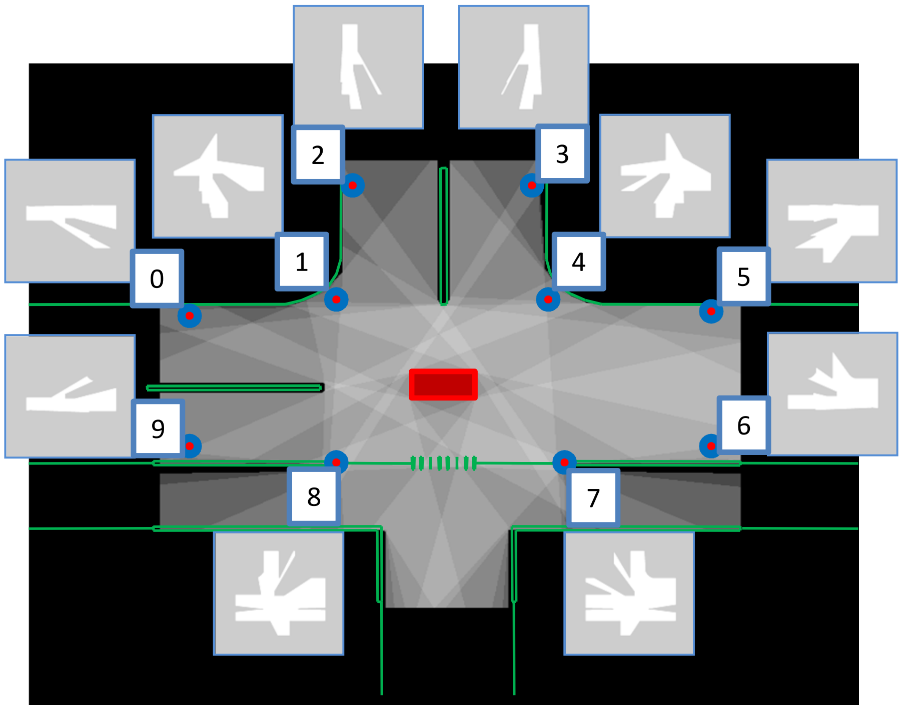


Figure 7.19: Overview of subway station with camera visibility maps.
Cameras are shown as blue circles, walls are shown in green, and the billboard is shown in red. The aggregated visibility map is also overlaid on top of the station diagram.

The surrogate model for each person consisted of 4 cubic regions 0.5 meters on the side, stacked on the vertical axis. The coordinates of the center of each region were included in the state, and a *PV* state model was used in the Kalman filter-based performance metric evaluation process.

As the number of cameras and the number of ROIs are much higher than in the previous set of experiments, I used the faster greedy assignment method in Algorithm 6.4 and a modified version of the heuristic function *FindChoices*() described in Algorithm 6.9 to achieve real-time performance. The changes to Algorithm 6.9 were aimed at reducing the number of choices generated to a single capture for the entire horizon and a small number of captures of minimum duration (uninterruptible dwell).

Similarly to the set of experiments in Section 7.1, I set the transition duration to 2 seconds and the minimum capture duration (uninterruptible dwell) to 4 seconds. The experiment duration was 308 seconds. While a true GPS signal would not be available inside a subway station, the experiments in this section still rely on an external tracker for initialization and periodic GPS-style 3D measurements.

7.2.2 Results

Just as in Section 7.1.2, for the experiments in this section, I show a plot of the metric values over time; a timeline of the camera states and agency memberships; and a stacked bar plot of the camera states.

7.2.2.1 Experiment 1: Default Settings

Under the default settings, I set the clustering diameter $D \simeq 13.5$ meters, assumed that the external tracking signal was available every second, and that the length of a planning cycle was $lCycle = 1$ second. The planning horizon was set to $H = 15$ seconds. The visibility threshold was set to $T_v = 0.5$. Figure 7.20 shows the results.

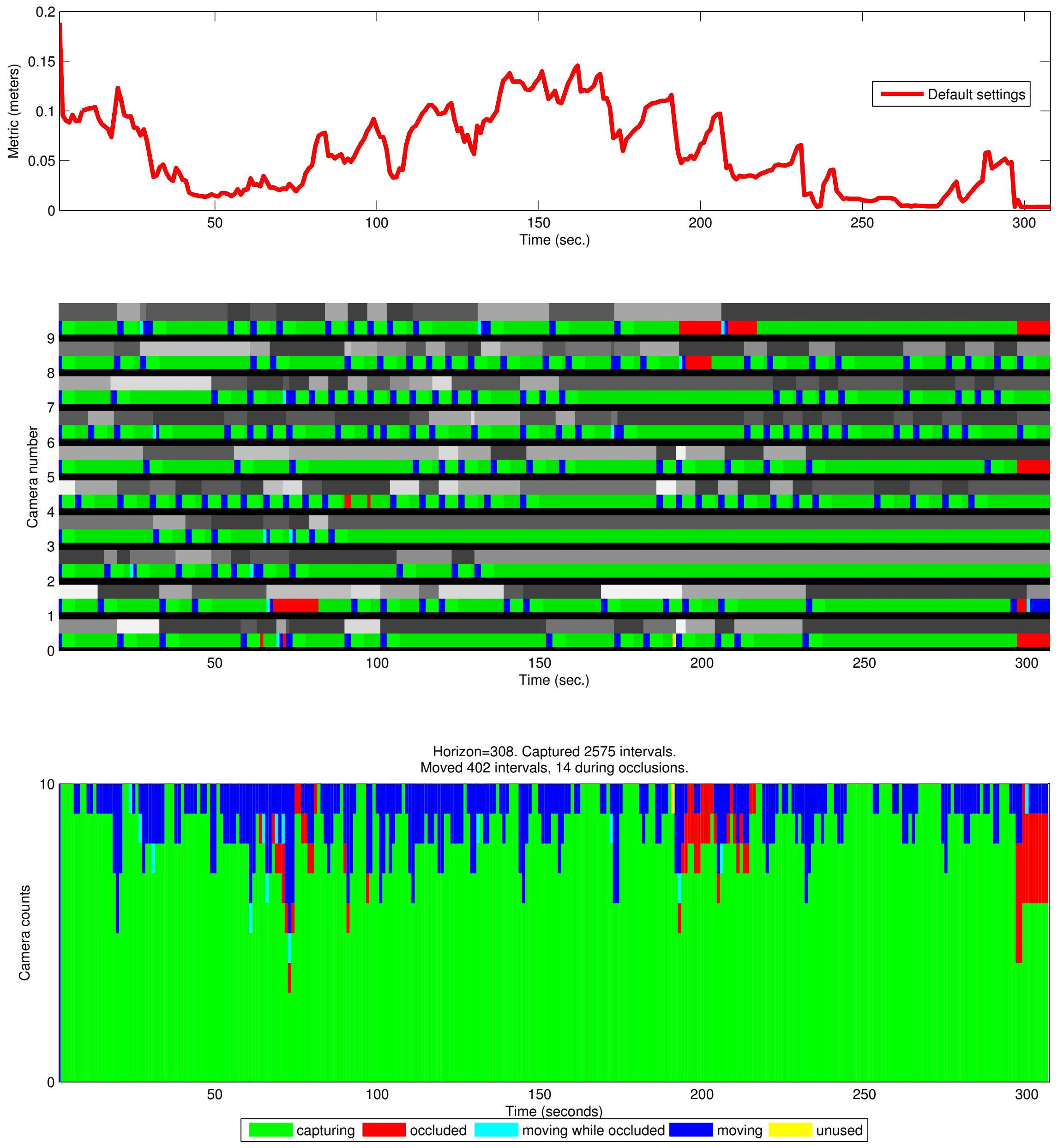


Figure 7.20: Metric, camera states and agency memberships over time for the default settings.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.
Compare with the similar plots in the USMC training exercise Experiments $1-4$, Sections 7.1.2.1-7.1.2.4, Figures 7.4-7.11.

7.2.2.2 Experiment 2: Using Incidence Angle to Capture Frontal Images Only

One task often required of surveillance approaches is capturing frontal images of people. To enable my approach to accomplish this task, I added a surface normal to the top points in the ROI models, pointing in their heading direction. I also lowered the importance of the bottom 3 points in the model of each ROI. These changes resulted in aiming cameras at a person when the person is facing the camera. This new behavior takes precedence over the default behavior of capturing a person regardless of whether they are walking toward or away from the camera. As there were times when none of the 25 persons were walking towards a particular camera, this behavior led to a significant increase in the metric values when compared to the default settings from Section 7.2.2.1. Figure 7.21 shows the results.

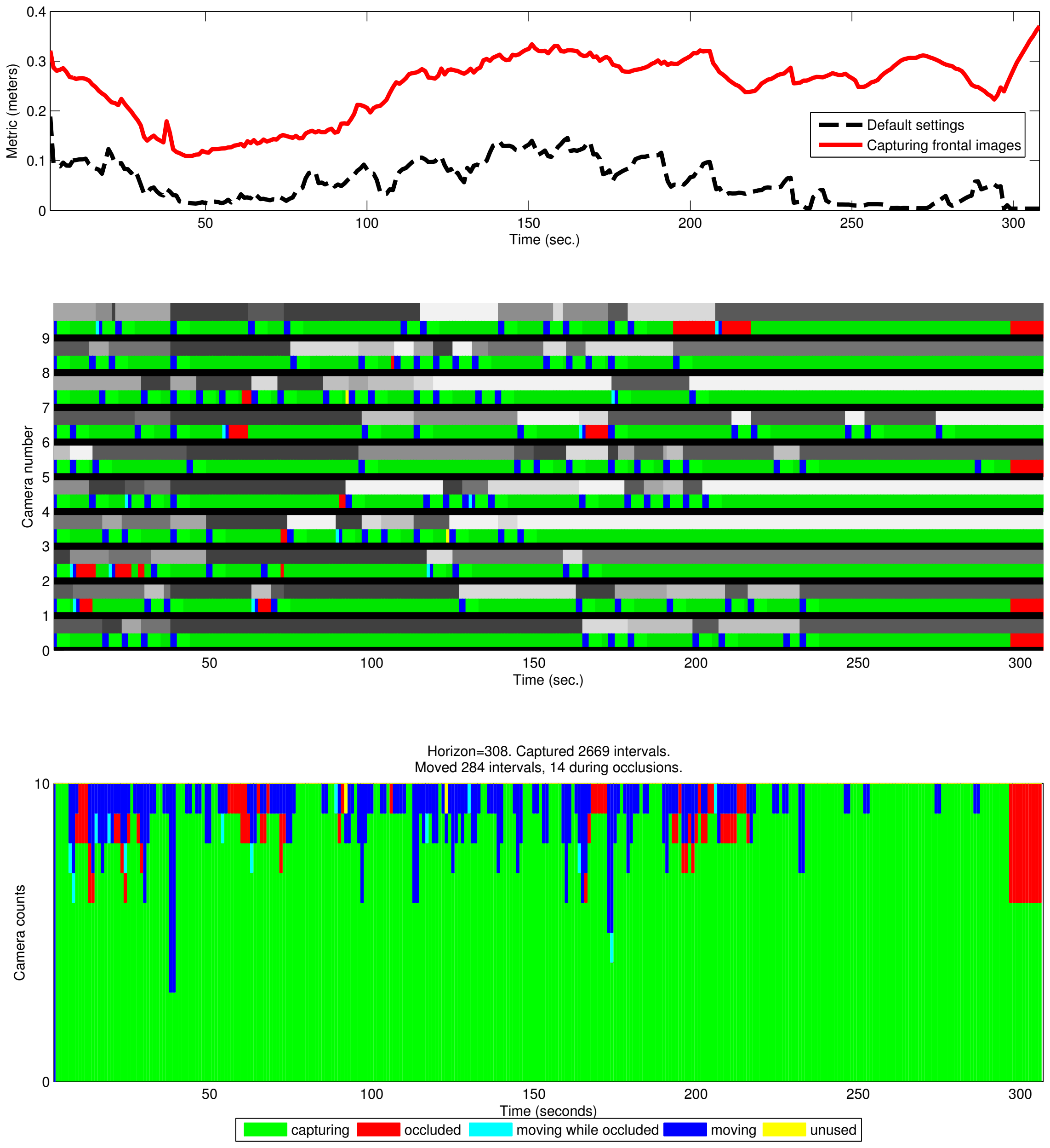


Figure 7.21: Metric, camera states and agency memberships over time when capturing frontal images.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.

7.2.2.3 Experiment 3: Decreasing the Planning Horizon

A smaller scale version of Experiment 7 in Section 7.1.2.7, this experiment explores the effect of reducing the planning horizon to $H = 10$ seconds. The shorter horizon left the modified

FindChoices() heuristic from Algorithm 6.9 with a single choice, that of capturing for the entire interval up to the horizon or up to the first long occlusion, and no attempts were made to plan multiple, shorter captures with smaller fields of view. This resulted in less transitions (358 vs. 402 in Experiment 1), but cameras were more zoomed out, and the metric values ended up being comparable to the default condition. Figure 7.22 shows the results.

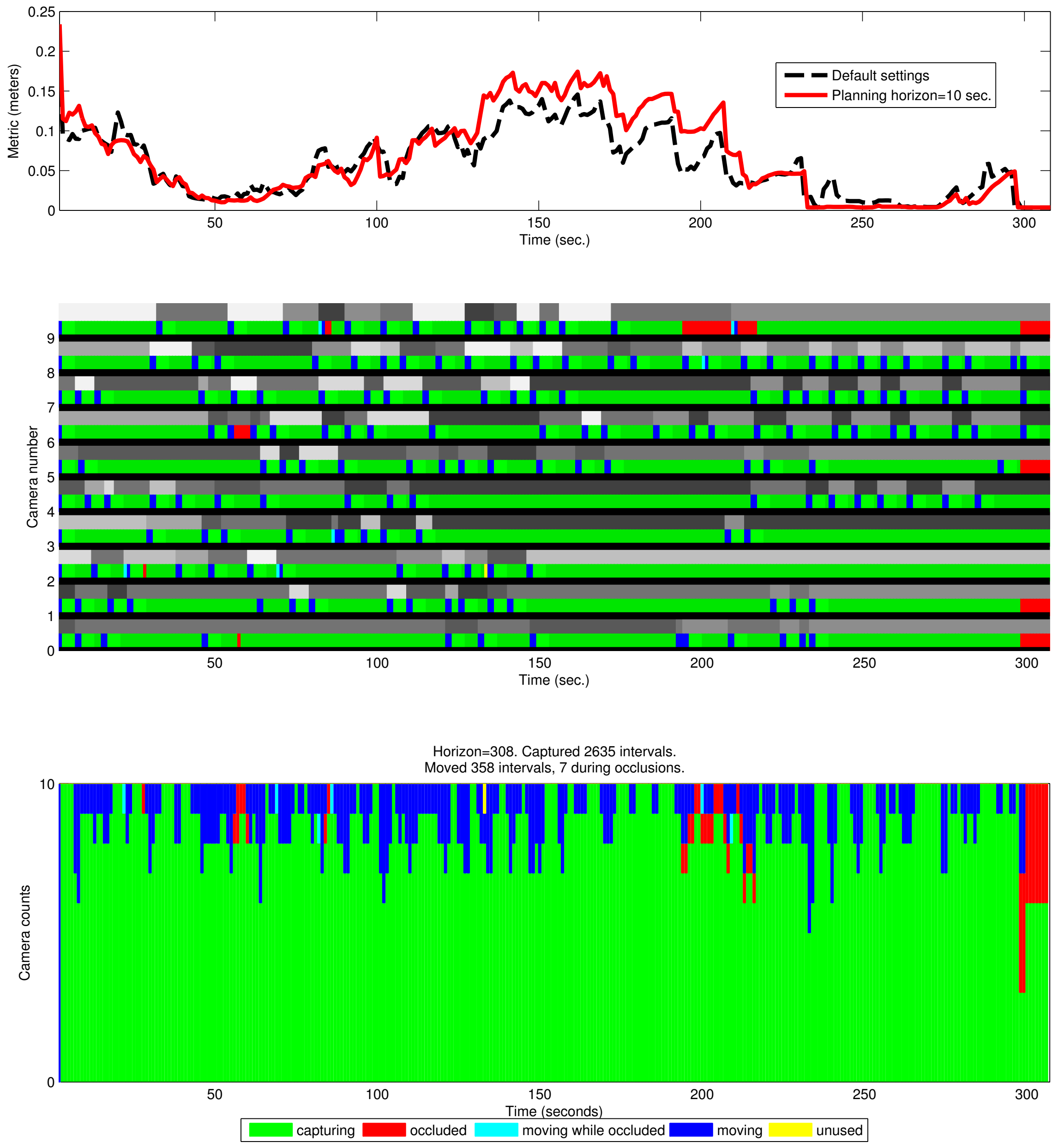


Figure 7.22: Metric, camera states and agency memberships over time when reducing the planning horizon to $H = 10$ seconds.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.

7.2.2.4 Experiment 4: Increasing the Visibility Threshold

A smaller scale version of Experiment 5 in Section 7.1.2.5, this experiment explores the effect of increasing the visibility threshold to $T_v = 0.75$, or decreasing the occlusion threshold to

$T_o = 0.25$. One would expect that the cameras would capture for longer periods with a higher visibility threshold. However, this higher threshold also makes planning more sensitive to errors in trajectory prediction, and zooming out for longer captures also leads to higher metric values. As a result, the metric values remain comparable to the default condition. Figure 7.23 shows the results.

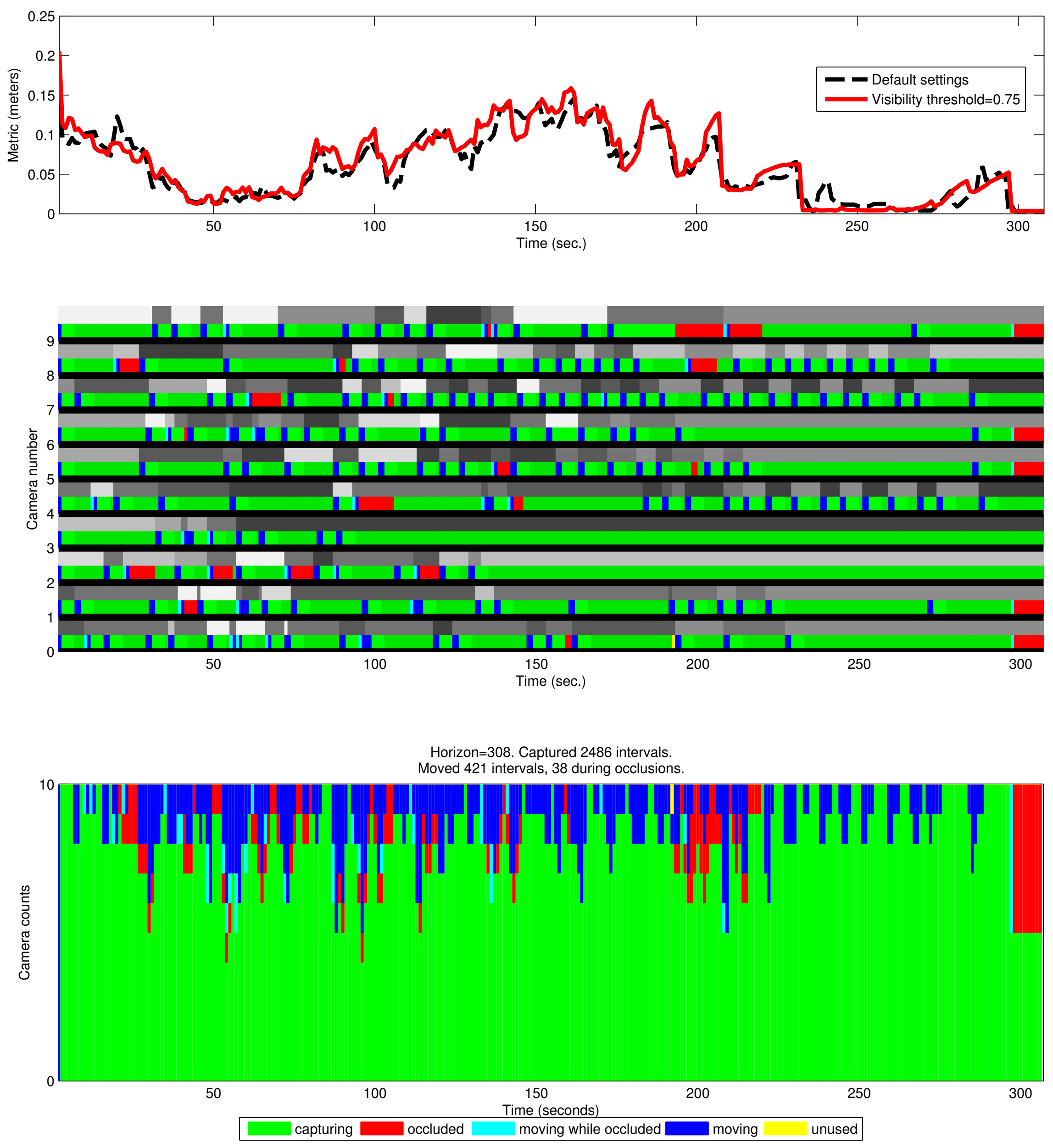


Figure 7.23: Metric, camera states and agency memberships over time for increasing the visibility threshold.
Top: plot of metric values over time (lower values are better).
Middle: timeline plot of agency memberships (shades of gray) and camera states (colors).
Bottom: stacked bar plot of camera states.

Chapter 8

Summary, Conclusions and Future Work

8.1 Summary

In this thesis, I introduced a novel camera control approach consisting of two main components: a stochastic performance metric and a constrained optimization method.

The performance metric I presented in Chapter 5 uses state-space stochastic models of target dynamics and camera measurements to quantify the uncertainty in the state of the targets. State-space models enable the performance metric to account for well-known quality factors such as static and dynamic occlusions, accommodate requirements specific to the algorithm used to process the images, and incorporate other factors that can affect its results.

In Chapter 3, I looked at two alternative search spaces, and showed that an exhaustive exploration of either parameter space is intractable for non-trivial problems. In Chapter 4, I investigated the impact of a few alternative strategies for reducing the size of both search spaces. I showed that for non-trivial situations none of the reduction strategies would by itself guarantee that an optimization-based camera control method would be able to explore the reduced search space in real-time. However, since the strategies reduce search space sizes along different dimensions, they can be used simultaneously to combine their search space size reduction effects.

The optimization method I presented in Chapter 6 is just one of many possible alternatives of combining multiple reduction strategies. It searches the space of camera configurations over time for solutions that satisfy constraints derived from the camera capabilities, the predicted target trajectories, and the computer vision algorithm used to process the images. The spatial dimension of the search is reduced by dividing the optimization problem into subproblems, and then optimizing each subproblem independently. The temporal dimension of the search is reduced by using empirically-based heuristics inside each subproblem. The resulting method is a tractable optimization that explores an appropriate subspace of the parameters, while attempting to minimize the risk of excluding the global optimum.

I also explored how my approach can be adapted to domains ranging from conventional surveillance tasks such as tracking and face recognition to tasks employing more complex computer vision methods such as motion capture and 3D reconstruction. Adaptations involved straightforward changes in components of my method such as the surrogate models representing the state, the performance metric aggregation function, and the target clustering algorithm.

Finally, in Chapter 7, I presented experimental evaluations using two scenarios: a USMC training exercise from which real GPS target tracks were captured, and a simulated crowd in a virtual subway station. In both situations, I used realistic camera placements likely to be encountered in an exercise range or a real subway station: cameras were placed on light poles and along walls. In the USMC scenario, I showed that simple heuristics based on practical observations perform very well when compared with exhaustive exploration.

8.2 Conclusions

Automated on-line camera control approaches have emerged as a fertile research domain in attempts to replace unreliable human camera control in demanding applications such as surveillance and training. The research in this thesis has led to the following conclusions:

- Measuring and predicting the performance of an active camera network using a stochastic performance metric allows casting camera control as an optimization problem. In Section 5.2, I introduced a performance metric based on stochastic state-space models.

- Camera performance factors and application requirements can be easily incorporated into the performance metric. I showed how to incorporate performance factors in Section 5.4 and application requirements in Section 5.3.
- Exhaustive exploration of the resulting parameter spaces is intractable for non-trivial problems. In Section 3.5, I concluded that exhaustive exploration is exponential in terms of the number of cameras and the number of planning steps.
- The size of the search space can be reduced through empirically-based strategies. In Chapter 4, I presented rules and heuristics for reducing the size of the search space.
- No such strategy can, by itself, guarantee real-time performance. In Sections 4.4 and 4.5, I showed that even for a small problem none of the strategies can come close to running in real-time.
- The strategies in Chapter 4 affect different parts of the search space, so they can be used simultaneously to combine their effects. In Chapter 6, I described one combination of strategies that can achieve real-time performance.
- In local planning, the performance of heuristic exploration can be very close to the performance of exhaustive exploration. In Section 7.1.2.9, I showed that for the USMC exercise experiments the differences are minimal.

8.3 Future Work

The results shown in this research are based on simulations. The next step is to apply my approach to setups with real cameras. The prototype system I used in the experiments in Chapter

7 includes a module for interfacing with cameras. When running the USMC exercise experiments, it communicated via network with the camera simulator in [TCB07], controlling the simulator's virtual cameras and retrieving the simulated images. The simulator has a similar interface to the one encountered in real PTZ cameras: control is performed via specially-formed URLs, and images are streamed separately, on demand. In order to work with real cameras, the network communication library needs to be replaced. However, my planning method, including the camera interface module, will likely remain unchanged.

Working with real cameras is a task planned for the next year of the BASE-IT project. The first phase will be a lab-based setup, with cameras mounted on the ceiling of the new lab at UNC. Long-term plans include transitioning to the existing infrastructure at a training facility like the one diagrammed in Figure 1.3. This infrastructure also includes many fixed cameras and other sensors like GPS and INS, which need to be integrated through the use of additional communication modules.

A situation likely to be encountered in practice is the availability of many more cameras than recording devices. In this case, camera selection is needed to decide which camera streams to capture. At the end of Section 6.2, I showed how camera selection can be performed as a straightforward extension of my approach. If tracking is available from an external system like GPS, the camera poses and capabilities are enough for my method to select fixed or active cameras and plan captures and transitions for the active cameras it selected. A more challenging situation is when target tracks come from a video-based tracking system that uses the same cameras that are also used for capturing images for computer vision tasks such as 3D reconstruction. In this situation, my method needs to be augmented to balance possibly conflicting requirements from multiple clients (e.g., tracking and computer vision). This is a major change with significant consequences on the performance of the method and the plans it computes.

The performance of a real camera system would likely benefit from the use of "smart" cameras that can perform low-level processing tasks such as target detection. In Section 5.3.4, I listed several ways of dealing with new targets entering the scene: fixed entry points, probabilistic background modeling, and external input. The current implementation relies on input from an external tracking system. The other two alternatives take advantage of information that may be available about the environment and the expected target behavior. Fixed entry points can be monitored continuously with fixed cameras or sporadically with active cameras, using my planning method and fixed ROIs. Probabilistic background modeling can only be used as part of the planning method for active cameras. Information about the environment can also be used to correct erroneous trajectory predictions. For example, if a Kalman filter-based trajectory prediction has a target passing through a wall, the prediction could be corrected to have the target stop in front of the wall. The effects of these changes to my method are yet to be studied.

My current implementation uses a proximity heuristic for grouping ROIs into agencies, and a greedy heuristic for assigning camera agents to agencies. It would be interesting to explore other clustering algorithms and evaluate the performance of my method when using them. One interesting direction to explore in clustering would be to allow overlaps, i.e. to allow an ROI to be a member of multiple agencies. Additionally, the current greedy assignment scheme exclusively assigns an agent to an agency. There may be situations when, with a small change in its settings, a camera could cover the ROIs in another agency nearby. In such situations, it may be beneficial to explore sharing an agent among multiple agencies. Sharing agents would require a protocol for agencies to cooperate when computing the camera parameters for agents that are shared with other agencies.

The main strategy my approach employs to reduce the search space size is decomposing the problem into subproblems and solving the subproblems at the level of each agency. Another possible research direction is decentralizing my approach further by having cameras estimate their own relevance for each client task. In the current implementation, the global assignment process is centralized. Distributing the assignment process by making it collaborative is likely to reduce computation loads in large camera networks, possibly at the expense of increased bandwidth required for collaboration, as well as possible additional system latency and weaker optimal performance guarantees.

Bibliography

[All07] Bonnie Danette Allen. *Hardware Design Optimization for Human Motion Tracking Systems*. PhD thesis, University of North Carolina at Chapel Hill, December 2007. xiii, 10, 11, 16, 71, 73, 74, 78, 88, 100, 110

[AW05] Bonnie Danette Allen and Greg Welch. A general method for comparing the expected performance of tracking and motion capture systems. In *12th ACM Symposium on Virtual Reality Software and Technology*, pages 201–210, New York, NY, November 2005. ACM Press, Addison-Wesley. 71, 81

[BC08] Luca Ballan and Guido Maria Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT*, Atlanta, GA, USA, June 2008. 37

[BdBP05] Andrew D. Bagdanov, Alberto del Bimbo, and Federico Pernici. Acquisition of high-resolution images through online saccade sequence planning. In *3rd ACM international workshop on Video surveillance & sensor networks*, pages 121–130, 2005. 22, 127

[BGV$^+$09] Christopher Broaddus, Thomas Germano, Nicholas Vandervalk, Ajay Divakaran, Shunguang Wu, and Harpreet Sawhney. Act-vision: active collaborative tracking for multiple ptz cameras. In *Proceedings of SPIE: Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, volume 7345, Orlando, FL, USA, April 2009. 3, 28, 127, 129

[BNB04] Ardevan Bakhtari, Michael D. Naish, and Beno Benhabib. Active vision for the autonomous surveillance of dynamic, multi-object environments. In *ASME International Mechanical Engineering Congress and Exposition*, Anaheim, CA, USA, November 11-15 2004. 36

[BNE$^+$06] Ardevan Bakhtari, Michael D. Naish, Maryam Eskandari, Elizabeth A. Croft, and Beno Benhabib. Active-vision-based multisensor surveillance-an implementation. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 36(5):668–680, September 2006. 23, 127

[BSP05] R. Bodor, P. Schrater, and N. Papanikolopoulos. Multi-camera positioning to optimize task observability. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 552–557, September 2005. static cameras. 18, 88

[Cas07] Mattie Casper. EventScripts Python. http://python.eventscripts.com, 2007. 130

[CC04] Amit K. Roy Chowdhury and Rama Chellappa. An information theoretic criterion for evaluating the quality of 3d reconstructions from video. *IEEE Trans. on Image Processing*, 13(7):960–973, July 2004. 20

[CD00] Xing Chen and James Davis. Camera placement considering occlusion for robust motion capture. Technical Report CS-TR-2000-07, Stanford University, December 2000. 110

[CD08] Xing Chen and James Davis. An occlusion metric for selecting robust camera configurations. *Machine Vision and Applications Journal*, 19(4):217–222, July 2008. 26, 110

[CDBF04] Cash J. Costello, Christopher P. Diehl, Amit Banerjee, and Hesky Fisher. Scheduling an active camera to observe people. In *ACM 2nd international workshop on Video surveillance & sensor networks*, pages 39–45, 2004. 23, 25, 36, 127

[Che02] Xing Chen. *Designing Multi-Camera Tracking Systems for Scalability and Efficient Resource Allocation*. PhD thesis, Stanford University, 2002. 16, 88, 97, 98, 99, 110

[CL04] S. Y. Chen and Y. F. Li. Automatic sensor placement for model-based robot vision. *IEEE Transactions on Systems, Man and Cybernetics*, 34(1):393–408, February 2004. 20, 21, 88

[CLK$^+$00] Robert T. Collins, Alan J. Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt1, and Lambert Wixson. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, The Robotics Institute, Carnegie Mellon University, Pittsburgh PA, May 2000. 5

[Cow88] Cregg. K. Cowan. Model-based synthesis of sensor location. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 900–905, April 1988. 15, 88

[CT84] C. B. Chang and J. A. Tabaczyinski. Application of state estimation to target tracking. *IEEE Transactions on Automatic Control*, ac-29(2):98–109, February 1984. 72

[Dav02] James Davis. *Mixed Scale Motion Recovery*. PhD thesis, Stanford University, 2002. 1, 6, 16, 31

[DB01] J. Denzler and C. Brown. An information theoretic approach to optimal sensor data selection for state estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):145–157, 2001. 11, 19, 71, 110

[DB02] J. Denzler and C. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):145–157, 2002. 11, 71, 88

[DBN01] Joachim Denzler, C. M. Brown, and H. Niemann. Optimal camera parameter selection for state estimation with applications in object recognition. *Lecture Notes in Computer Science*, 2191:305–, 2001. 11, 19, 71

[dBP05] Alberto del Bimbo and Federico Pernici. Towards on-line saccade planning for high-resolution image sensing. *Pattern Recognition Letters*, 27(15):1826–1834, November 2005. 22, 127

[DDN03] Frank Deinzer, Joachim Denzler, and Heinrich Niemann. Viewpoint selection - planning optimal sequences of views for object recognition. In N. Petkov and M. A. Westenberg, editors, *Computer Analysis of Images and Patterns*, number 2756 in Lecture Notes in Computer Science, pages 65–73, Heidelberg, 2003. Springer. 21

[DM02] Andrew J. Davison and David W. Murray. Simultaneous localisation and map-building using active vision. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002. 6

[DND05] Benjamin Deutsch, Heinrich Niemann, and Joachim Denzler. Multi-step active object tracking with entropy based optimal actions using the sequential kalman filter. In *IEEE International Conference on Image Processing*, volume 3, pages 105–108, 2005. 27, 80, 127

[DWN06] Benjamin Deutsch, Stefan Wenhardt, and Heinrich Niemann. Multi-step multi-camera view planning for real-time visual object tracking. *Lecture Notes in Computer Science*, 4174:536–545, 2006. 82, 83, 87, 110

[DZ01] Joachim Denzler and Matthias Zobel. On optimal observation models for kalman filter based tracking approaches. Technical report, Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, 2001. 11, 71

[DZDN04] Benjamin Deutsch, Matthias Zobel, Joachim Denzler, and Heinrich Niemann. Multi-step entropy based sensor control for visual object tracking. *Lecture Notes in Computer Science*, 3175:359–366, 2004. 27, 82, 83

[DZN02] Joachim Denzler, Matthias Zobel, and Heinrich Niemann. On optimal camera parameter selection in kalman filter based object tracking. In *24th DAGM Symposium on Pattern Recognition*, pages 17–25, 2002. 11, 20, 71, 81

[DZN03] Joachim Denzler, Matthias Zobel, and Heinrich Niemann. Information theoretic focal length selection for real-time active 3-d object tracking. In *International Conference on Computer Vision*, volume 1, pages 400–407, October 2003. 26, 110

[EGG06] A. O. Ercan, A. El Gamal, and L. J. Guibas. Camera network node selection for target localization in the presence of occlusions. In *Distributed Smart Cameras*, October 2006. 19, 88

[EYGG06] 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 *IEEE International Conference on Distributed Computing in Sensor Systems*, June 2006. 19

[FCOL00] Shachar Fleishman, Daniel Cohen-Or, and Dani Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19(2):101–110, 2000. 16, 18, 20, 88

[GA93] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering Theory and Practice*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ USA, 1993. 75, 78

[GFP07] Li Guan, Jean-Sebastien Franco, and Marc Pollefeys. 3d occlusion inference from silhouette cues. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, June 2007. xiii, 1, 2

[GFP08] Li Guan, Jean-Sebastien Franco, and Marc Pollefeys. Multi-object shape estimation and tracking from silhouette cues. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, June 2008. xiii, 1, 2

[GLM10] Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. Modeling collision avoidance behavior for virtual humans. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010. xv, 150

[Gua10] Li Guan. *Multi-view Dynamic Scene Modeling*. PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, April 2010. xv, 1, 138, 139

[Hep10] Daryl H. Hepting. The history of a picture's worth. http://www2.cs.uregina.ca/ hepting/research/web/words/history.html, 2010. 1

[HL06] E. Horster and R. Lienhart. On the optimal placement of multiple visual sensors. In *4th ACM international workshop on Video surveillance and sensor networks*, pages 111–120, Santa Barbara, California, USA, 2006. 17

[IB05] Volkan Isler and Ruzena Bajcsy. The sensor selection problem for bounded uncertainty sensing models. In *IPSN*, pages 151–158, 2005. 19

[ILW$^+$04] Adrian Ilie, Kok-Lim Low, Greg Welch, Anselmo Lastra, Henry Fuchs, and Bruce Cairns. Combining head-mounted and projector-based displays for surgical training. *Presence: Teleoperators and Virtual Environments*, 13(2):128–145, 2004. 13

[IW05] Adrian Ilie and Greg Welch. Ensuring color consistency across multiple cameras. In *Tenth IEEE International Conference on Computer Vision*, volume 2, pages 1268–1275, October 2005. 14

[IWM08] Adrian Ilie, Greg Welch, and Marc Macenko. A stochastic quality metric for optimal control of active camera network configurations for 3D computer vision tasks. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, Marseille, France, October 2008. 10, 16, 71, 82, 88, 103, 110

[KS00] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, July 2000. 1

[KSH00] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ, 2000. 72

[KYL$^+$08] Nils Krahnstoever, Ting Yu, Ser-Nam Lim, Kedar Patwardhan, and Peter Tu. Collaborative real-time control of active cameras in large scale surveillance systems. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, Marseille, France, October 2008. 27, 87, 110, 127

[KYS01] N. Krahnstoever, M. Yeasin, and R. Sharma. Towards a unified framework for tracking and analysis of human motion. In *Proc. IEEE Workshop on Detection and Recognition of Events in Video*, Vancouver, Canada, July 2001. 78

[L-310] L-3 Communications. Praetorian. http://www.l3praetorian.com/, 2010. 18

[LMD05] Ser-Nam Lim, Anurag Mittal, and Larry Davis. Constructing task visibility intervals for a surveillance system. In *3rd ACM international workshop on Video surveillance & sensor networks*, pages 141–148, 2005. 24, 127

[LMD07] Ser-Nam Lim, Anurag Mittal, and Larry S. Davis. Task scheduling in large camera networks. In *Asian Conference on Computer Vision*, 2007. 24, 127

[Low06] Kok-Lim Low. *View Planning for Range Acquisition of Indoor Environments*. PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2006. 20, 21

[Mas95] Scott Mason. Expert system-based design of close-range photogrammetric networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50(5):13–24, October 1995. 16, 88

[MBR$^+$00] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *ACM Siggraph*, pages 369–374, 2000. 1

[MD04] A. Mittal and L. Davis. Visibility analysis and sensor planning in dynamic environments. In *European Conference on Computer Vision*, 2004. 16, 28, 88

[MD08] Anurag Mittal and Larry S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *International Journal of Computer Vision*, 76(1):31–52, January 2008. 16, 26

[MOFR01] L. Marcenaro, F. Oberti, G. L. Foresti, and C.S. Regazzoni. Distributed architectures and logical-task decomposition in multimedia surveillance systems. In *IEEE*, volume 89, pages 1419–1440, Rosemont, IL, USA, October 2001. 29

[MU02] Takashi Matsuyama and Norimichi Ukita. Real-time multitarget tracking by a cooperative distributed vision system. In *Proceedings of the IEEE*, volume 90, pages 1137–1150, July 2002. 29, 127

[MWTN04] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. In *Computer Vision and Image Understanding*, volume 96, pages 393–434, New York, NY, USA, July 2004. Elsevier Science Inc. 1

[NCB00] Michael D. Naish, Elizabeth A. Croft, and Beno Benhabib. Dynamic dispatching of coordinated sensors. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 3318–3323, Nashville, TN, USA, October 8-11 2000. 17

[NCB01a] Michael D. Naish, Elizabeth A. Croft, and Beno Benhabib. Dispatching of coordinated proximity sensors for object surveillance. In *ASME International Mechanical Engineering Congress & Exposition, Symposium on Intelligent Sensors*, New York, NY, USA, November 2001. 17

[NCB01b] Michael D. Naish, Elizabeth A. Croft, and Beno Benhabib. Simulation-based sensing-system configuration for dynamic dispatching. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 2964–2969, Tucson, AZ, USA, October 2001. 17, 23, 127

[NCB03] Michael D. Naish, Elizabeth A. Croft, and Beno Benhabib. Coordinated dispatching of proximity sensors for the surveillance of manoeuvring targets. *Robotics and Computer-Integrated Manufacturing*, 19(3):283–299, June 2003. 23, 127

[OFR01] Franco Oberti, G. Ferrari, and Carlo S. Regazzoni. Allocation strategies for distributed video surveillance networks. In *International Conference on Image Processing*, volume 2, pages 415–418, October 2001. 29

[OM02] Gustavo Olague and Roger Mohr. Optimal camera placement for accurate reconstruction. *Pattern Recognition*, 35(4):927–944, April 2002. 16, 88, 110

[QT05a] Faisal Z. Qureshi and Demetri Terzopoulos. Surveillance camera scheduling: a virtual vision approach. In *3rd ACM international workshop on Video surveillance & sensor networks*, pages 131–140, Hilton, Singapore, 2005. ACM Press New York, NY, USA. 25, 29, 30, 127

[QT05b] Faisal Z. Qureshi and Demetri Terzopoulos. Towards intelligent camera networks: a virtual vision approach. In *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 177–184, Beijing, China, October 2005. 25, 29

[QT07] Faisal Qureshi and Demetri Terzopoulos. Surveillance in virtual reality: System design and multi-camera control. In *Proceedings of Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 25, 29, 127

[Ray02] Sidney F. Ray. *Applied Photographic Optics*. Oxford: Focal Press, third edition, 2002. 103

[RR00] Ehud Rivlin and Hector Rotstein. Control of a camera for active vision: Foveal vision, smooth tracking and saccade. *International Journal of Computer Vision*, 39(2):81–96, September 2000. 30

[RRA$^+$06] G.S.V.S. Siva Ram, K.R. Ramakrishnan, P.K. Atrey, V.K.Singh, and M.S.Kankanhalli. A design methodology for selection and placement of sensors in multimedia surveillance systems. In *4th ACM international workshop on Video surveillance & sensor networks*, 2006. 17

[RSJ03] P. Remagnino, A.I. Shihab, and G.A. Jones. Distributed intelligence for multi-camera visual surveillance. *Pattern Recognition*, 37(4):675–689, April 2003. 29

[SH07] Stanislava Soro and Wendi B. Heinzelman. Camera selection in visual sensor networks. In *International Conference on Advanced Video and Signal based Surveillance*, September 2007. 20

[SMR03] G. Scotti, L. Marcenaro, and C.S. Regazzoni. Som based algorithm for video surveillance system parameter optimal selection. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 370–375, July 2003. 28

[SP04] Sudipta N. Sinha and Marc Pollefeys. Towards calibrating a pan-tilt-zoom camera network. In Peter Sturm, Tomas Svoboda, and Seth Teller, editors, *5th Workshop on Omnidirectional Vision*, Prague, 2004. 31

[SR08a] Eric Sommerlade and Ian Reid. Cooperative surveillance of multiple targets using mutual information. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008. 27, 87, 127

[SR08b] Eric Sommerlade and Ian Reid. Information theoretic active scene exploration. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, May 2008. 27

[SR08c] Eric Sommerlade and Ian Reid. Information-theoretic decision making for exploration of dynamic scenes. In *Proceedings of the 5th International Workshop on Attention in Cognitive Systems (WAPCV)*, May 2008. 27

[SRR03] William R. Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys*, 35(1):64–96, March 2003. 20

[SS91] Shigeyuki Sakane and T. Sato. Automatic planning of light source and camera placement for an active photometric stereo system. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1080–1087. IEEE Robotics and Automation Society, April 1991. 20

[SSSAH04] M. Saadat-Seresht, F. Samdzadegan, A. Azizi, and M. Hahn. Camera placement for network design in vision metrology based on fuzzy inference system. In *XXth ISPRS Congress*, July 2004. 16, 88

[SWB$^+$09] Amela Sadagic, Greg Welch, Chumki Basu, Chris Darken, Rakesh Kumar, Henry Fuchs, Hui Cheng, Jan-Michael Frahm, Mathias Kolsch, Neil Rowe, Herman Towles, J. Wachs, and Anselmo Lastra. New generation of instrumented ranges: Enabling automated performance analysis. In *2009 Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC-2009)*, Orlando, FL, 2009. 3

[SWI06] Andrei State, Greg Welch, and Adrian Ilie. An interactive camera placement and visibility simulator for image-based vr applications. In *Engineering Reality of Virtual Reality 2006 (3D Imaging, Interaction, and Measurement; IS&T/SPIE 18th Annual Symposium on Electronic Imaging Science and Technology)*, San Jose, CA, January 2006. 9, 16, 18, 88

[TCB07] Geoffrey R. Taylor, Andrew J. Chosak, and Paul C. Brewer. OVVV: Using virtual worlds to design and evaluate surveillance systems. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 11, 26, 128, 164

[TG94] G. H. Tarbox and S. N. Gottschlich. Ivis: an integrated volumetric inspection system. In *Second CAD-Based Vision Workshop*, pages 220–227, Champion, PA, USA, February 1994. 20

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005. 114

[TTA95] Konstantinos A. Tarabanis, Roger Y. Tsai, and Peter K. Allen. A survev of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995. 16, 20, 26

[TTK96] Konstantinos A. Tarabanis, Roger Y. Tsai, and A. Kaul. Computing occlusion-free viewpoints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):279–292, March 1996. 15, 20, 88

[Val06] Valve Corporation. Valve Hammer Editor. http://developer.valvesoftware.com/wiki/Category:Hammer, 2006. Build 4037. xv, 128, 129

[WAIB07] Greg Welch, B. Danette Allen, Adrian Ilie, and Gary Bishop. Measurement sample time optimization for human motion tracking/capture systems. In Gabriel Zachmann, editor, *Trends and Issues in Tracking for Virtual Environments, Workshop at the IEEE Virtual Reality 2007 Conference*, Charlotte, NC, March 2007. xii, xiv, 71, 73, 74, 75

[WB01] Greg Welch and Gary Bishop. An introduction to the Kalman filter: SIGGRAPH 2001 course 8. In *Computer Graphics*, Annual Conference on Computer Graphics & Interactive Techniques. ACM Press, Addison-Wesley Publishing Company, Los Angeles, CA, USA, SIGGRAPH 2001 course pack edition, August 12-17 2001. 72, 76, 78

[WBV$^+$01] Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, and D'nardo Colucci. High-performance wide-area optical tracking: The hiball tracking system. *Presence: Teleoperators and Virtual Environments*, 10(1):1–21, 2001. 13

[WDAN07] Stefan Wenhardt, Benjamin Deutsch, Elli Angelopoulou, and Heinrich Niemann. Active visual object reconstruction using d-, e-, and t-optimal next best views. In *IEEE Conference on Computer Vision and Pattern Recognitio*, pages 1–7, June 2007. 20

[WDH$^+$06] Stefan Wenhardt, Benjamin Deutsch, Joachim Hornegger, Heinrich Niemann, and Joachim Denzler. An information theoretic approach for next best view planning in 3-d reconstruction. In *18th International Conference on Pattern Recognition*, volume 1, pages 103–106, Hong Kong, 2006. 20

[WSH98] 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*, 37:280–292, January 1998. 16, 26, 88, 110

[WYB$^+$05] Greg Welch, Ruigang Yang, Sascha Becker, Adrian Ilie, Dan Russo, Jesse Funaro, Andrei State, Kok-Lim Low, Anselmo Lastra, Herman Towles, M.D. Bruce Cairns, Henry Fuchs, and Andy van Dam. Immersive electronic books for surgical training. *IEEE Multimedia*, 12(3):22–35, July-September 2005. 2

[WZB$^+$06] Shunguang Wu, Tao Zhao, Christopher Broaddus, Changjiang Yang, and Manoj Aggarwal. Robust pan, tilt and zoom estimation for ptz camerab y using meta data and/or frame-to-frame correspondences. In *9th International Conference on Control, Automation, Robotics and Vision*, 2006. 31

[Yan03] Ruigang Yang. *View-dependent pixel coloring: a physically-based approach for two-dimensional view synthesis*. PhD thesis, The University of North Carolina at Chapel Hill, 2003. xiii, 2, 3, 9, 13

[YHS95] Seungku Yi, Robert M. Haralick, and Linda G. Shapiro. Optimal sensor and light source positioning for machine vision. *Computer Vision and Image Understanding*, 61(1):122–137, January 1995. 15

[YUK07] Sofiane Yous, Norimichi Ukita, and Masatsugu Kidode. An assignment scheme to control multiple pan/tilt cameras for 3d video. *Journal of Multimedia*, 2(1):10–19, February 2007. 28, 88

[Zha99] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision*, 1999. 37, 77