Intelligent Perception in Virtual Sensor Networks and Space Robotics

by

Faisal Zubair Qureshi

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Intelligent Perception in

Virtual Sensor Networks and Space Robotics

Faisal Zubair Qureshi

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2007

Intelligent perception is a fundamental requirement of systems that exhibit sophisticated autonomous operation in complex dynamic worlds. It combines low-level, bottom-up, data-driven vision with high-level, top-down, knowledge-based processes. This thesis develops two embodied, task-oriented vision systems that exhibit autonomous, intelligent, goal-driven behavior through intelligent perception.

In Part I of the thesis, we develop a prototype surveillance system featuring a visual sensor network comprising wide field-of-view (FOV) passive cameras and pan/tilt/zoom active cameras. Novel multicamera control strategies enable the camera nodes to collaborate both in tracking pedestrians of interest that move across the FOVs of different cameras and in acquiring close-up videos of pedestrians as they travel across extended areas. Impediments to deploying and experimenting with appropriately extensive camera networks in large, busy public spaces would make our research more or less infeasible in the real world. However, a unique centerpiece of our approach is the virtual vision paradigm, in which we employ a visually and behaviorally realistic simulator in the design and evaluation of our surveillance systems. In particular, we employ a virtual train station populated by autonomous, lifelike virtual pedestrians, wherein easily reconfigurable virtual cameras generate synthetic video feeds that emulate those acquired by real surveillance cameras monitoring public spaces.

In Part II of the thesis, we develop a cognitively-controlled vision system that combines

low-level object recognition and tracking with high-level symbolic reasoning to tackle difficult space robotics problems, specifically satellite rendezvous and docking. There is significant interest in performing these operations autonomously, and our work is a step in this direction. Reasoning and related elements, among them intention, context, and memory, contribute to improve performance. We demonstrate the vision system controlling a robotic arm that autonomously captures a free-flying satellite. To date, this is the only satellite-capturing system that relies exclusively on vision to estimate the pose of the satellite and can deal with an uncooperative satellite.

Dedication

I dedicate this thesis to my parents, Zubair and Masuma, and to my wife, Lisa.

Acknowledgements

First and foremost, I thank Professor Demetri Terzopoulos, my advisor, without whose guidance, support, and encouragement this thesis would never have been completed. I am fortunate to have worked with him. It has been a pleasure all along. I dare say that I have had the best supervisor ever and I couldn't have asked for more.

I would like to thank Professor Sven Dickinson, who provided me with encouragement, collaboration, and sane advice during our many meetings. I am grateful to him for his guidance, and for serving on my thesis committee. Next, I would like to thank professors Allan Jepson, and Hector Levesque for serving on my proposal and dissertation committees. My special thanks to Professor Christopher Brown of the University of Rochester for serving as the external examiner of my thesis.

I owe my gratitude to Wei Shao and Mauricio Plaza whose work on the autonomous pedestrians and virtual Penn station model at New York University enabled us to develop the virtual vision paradigm for camera network research. Their enabling work was supported in part by a grant from the Defense Advanced Research Projects Agency (DARPA) of the US Department of Defense. I am thankful to Dr. Tom Strat, formerly of DARPA, for his generous support and encouragement.

For the second part of the thesis, I acknowledge the valuable technical contributions of Piotr Jasiobedzki, Ross Gillett, Hong Ng, Shawn Greene, Josh Richmond, Michael Greenspan, Michael Liu, and Amy Chan. This work was funded by MDA Space Missions, Ltd. (formerly MD Robotics, Ltd.) and Precarn Associates. My thanks to Dr. Loris Gregoris for motivation, encouragement, and support.

I owe my thanks to my colleagues and lab mates, among them Kiam Choo, Diego Macrini, Sam Hasinoff, Ady Ecker, Stratis Ioannidis, Qinxin Yu, and Chakra Chennubhotla. Thank you everybody at the DGP Lab and the Computer Vision Lab for the discussions, laughter, food and drink.

I am grateful to my parents, Zubair Masood and Masuma Zubair, and my siblings, Ayesha

and Usman, for their love, support, and encouragement during my long years in graduate school.

Finally, I am indebted to my wife, Lisa Tam, whose love and support saw me through to the completion of this thesis. Thank you, Lisa.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis is concerned with intelligent perception. Intelligent perception goes beyond bottom-up, data-driven, low-level vision to also include top-down, knowledge-driven, high-level visual processes. Intelligent perception is needed if we are to realize fully functional, truly intelligent systems, such as intelligent multi-camera surveillance systems and smart robotic agents capable of performing useful work in complex environments. The operation of these systems, along with their performance, need to be understood within the context of their environments as well as the goals of the system. With the objective of developing systems with intelligent perception, the focus of this thesis is to develop theory, computational tools, and algorithms for integrated low-level and high-level aspects of image understanding. In particular, we are interested in embodied, task-oriented vision systems that combine low-level vision with high-level symbolic reasoning. The latter encodes knowledge about the world and uses this knowledge to guide the vision system in a deliberative, task-directed manner.

Within the above intelligent perception framework, this thesis develops two systems capable of supporting sophisticated, autonomous behavior. One system is in the domain of sensor networks and visual surveillance, while the other is in the domain of space robotics. Both systems draw heavily upon the intelligent agent metaphor; i.e., their overall architecture is the result of local interactions between low-level perception routines mediated by high-level,

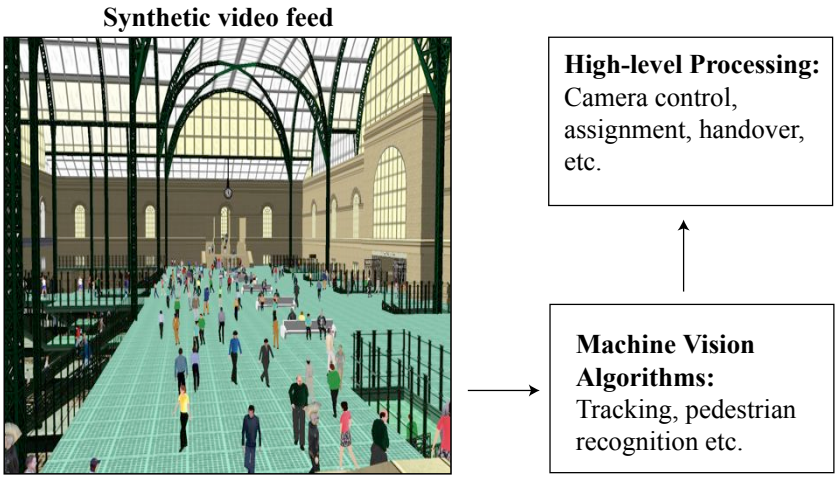


Figure 1.1: The virtual vision paradigm.

deliberative processes. The next two sections motivate and preview each of the two major components of this thesis in more detail.

1.1 Intelligent Perception for Visual Sensor Networks

Recent advances in camera and video technologies have made it possible to network numerous video cameras together in order to provide visual coverage of large public spaces such as airports and train stations. As the size of the camera network grows and the level of activity in the public space increases, it becomes infeasible for human operators to monitor the multiple video streams and identify all events of possible interest, or even to control individual cameras in performing advanced surveillance tasks, such as zooming in on a moving subject of interest to acquire one or more facial snapshots. Consequently, a timely challenge for computer vision researchers is to design camera sensor networks capable of performing visual surveillance tasks automatically, or at least with minimal human intervention.

In the first part of this thesis, we develop *Virtual Vision*, a paradigm that prescribes visually and behaviorally realistic virtual environments for the design of intelligent surveillance systems and the meaningful experimentation with such systems (Figure 1.1). Virtual vision investigates the possibility of developing and evaluating camera network control algorithms, such as camera

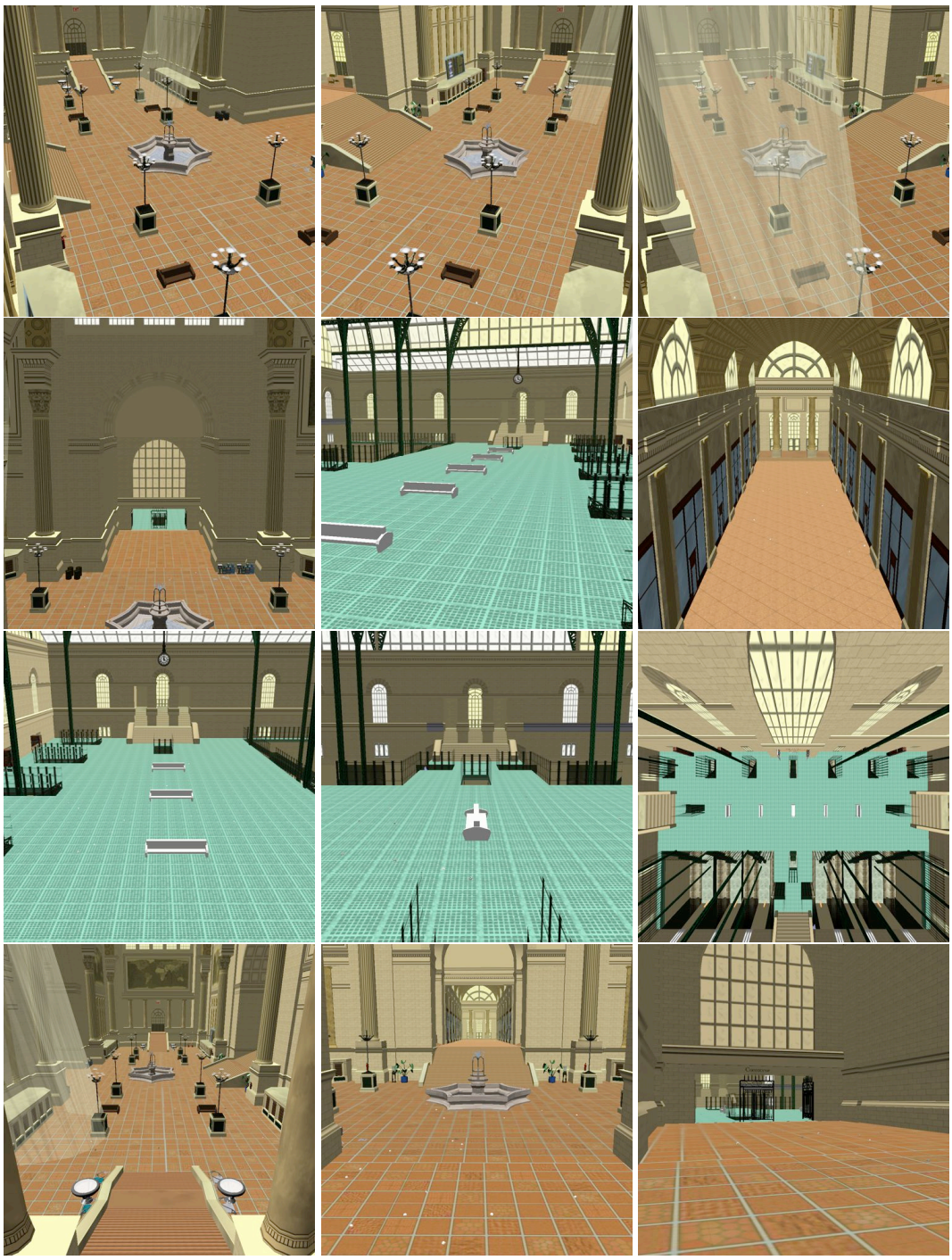
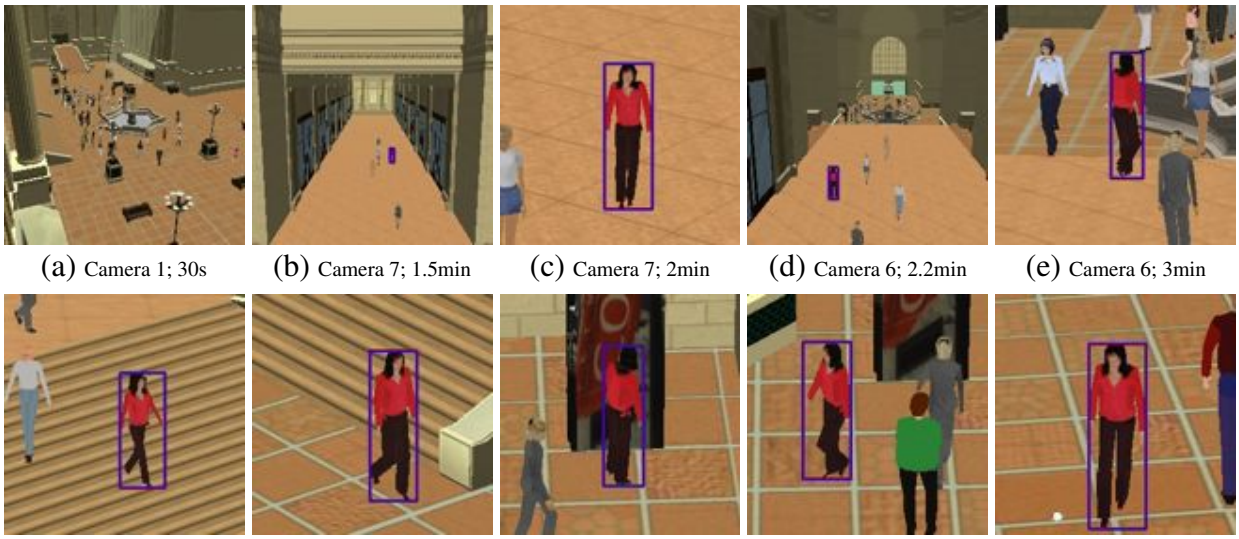


Figure 1.2: Synthetic video feeds from multiple virtual surveillance cameras situated in the (empty) Penn Station environment.

assignment and hand-off, by deploying virtual networks in simulated environments. It allows us to study high-level control problems that frequently arise in networks comprising smart cameras under realistic conditions. We believe that virtual vision is a powerful visual sensor network research paradigm and that the theory and algorithms developed within virtual vision will work without significant modification in the real world.

Skeptics might argue that virtual vision relies on simulated data, which can lead to inaccurate results. In particular, they may worry that virtual video lacks the subtleties of real video and that meaningful evaluation of a machine vision system is impossible in the absence of real video. But our high-level camera control routines do not directly process any raw video. They are dependent on the lower-level recognition and tracking routines, which mimic the performance (including failure modes) of a state-of-the-art pedestrian localization and tracking system and generate realistic input data for the high-level routines. We believe that our simulator enables us to develop and test camera network control algorithms under realistic assumptions derived from physical camera networks, and that these algorithms should readily port to the real world.

An important issue in smart camera networks is how to compare camera network algorithms. Two possible approaches are: 1) time-shared physical camera networks and 2) realistic simulation environments. Gathering benchmark data from time-shared physical networks comprising passive, fixed-zoom cameras involves simple video capture. On the other hand, gathering benchmark data for networks comprising active pan/tilt/zoom cameras requires scene reenactment for every run, which is clearly infeasible in most cases. Costello et al. [2004], who compared various schemes for scheduling an active camera to observe pedestrians present in the scene, ran into this issue and resort to Monte Carlo simulation to evaluate camera scheduling approaches. They conclude that evaluating scheduling policies on an physical testbed comprising a single active camera is extremely complicated. Our virtual vision approach provides a viable alternative that, among other benefits relative to a physical camera network, offers convenient and unlimited repeatability.



(a) Camera 1; 30s (b) Camera 7; 1.5min (c) Camera 7; 2min (d) Camera 6; 2.2min (e) Camera 6; 3min

(f) Camera 2; 4.3min (g) Camera 3; 5min (h) Camera 3; 6min (i) Camera 3; 13min (j) Camera 10; 13.4min

Figure 1.3: A pedestrian is tracked by cameras as she makes her way through the train station.

Within the virtual vision paradigm, we develop a prototype surveillance system featuring a visual sensor network comprising wide field-of-view (FOV) passive cameras and pan/tilt/zoom (PTZ) active cameras (Figure 1.2). Novel multi-camera control strategies enable the camera nodes to collaborate both in tracking pedestrians of interest that move across the FOVs of different cameras and in capturing close-up videos of pedestrians as they travel through designated areas (Figure 1.3). The sensor network supports task-dependent node selection and aggregation through local decision-making and inter-node communication. We treat node selection as a constraint satisfaction problem. Lacking a central controller, our solution is scalable and robust against node failures.

1.2 Intelligent Perception for Space Robotics

Satellite servicing is the task of maintaining and repairing a satellite in orbit. It extends the operational life of the satellite, mitigates technical risks, reduces on-orbit losses, and helps manage orbital debris. Hence, it is of interest to multiple stakeholders, including satellite operators, manufacturers, and insurance companies [Middleton et al. 1984; Davinic et al. 1988].

Figure 1.4: Images acquired during satellite capture. The left and center images were captured using the shuttle bay cameras. The right image was captured by the end-effector camera. The center image shows the arm in hovering position prior to the final capture phase. The shuttle crew use these images during satellite rendezvous and capture to locate the satellite at a distance of approximately 100m, to approach it, and to capture it with the Canadarm—the shuttle's manipulator.

Although replacing a satellite is more cost-effective in some cases, on-orbit servicing is critical for more expensive satellite systems, such as space-based laser and global positioning system constellations, or for one-of-a-kind systems like the Hubble telescope, which costs $2.5 billion. As early as the 1980s, the US National Aeronautics and Space Administration (NASA) realized the importance of on-orbit servicing for protecting their assets in space [Middleton et al. 1984].

Our space robotics work reported in this thesis was done in collaboration with MD Robotics, Ltd. (currently MDA Space Missions), a Canadian company that has supported human space flight since the early 1980s through advanced robotic systems, such as the Space Shuttle's Canadarm and the Mobile Servicing System for the International Space Station. The company, which undertakes extensive R&D projects in-house and through collaborations with universities and research institutions, regards autonomy as a necessary capability for future space robotics missions. The reported work was done as part of the ROSA (Remote Operation with Supervised Autonomy) project [Gillett et al. 2001], which arose from this long-term vision. ROSA's goal is to advance the state of the art (operator commands and discrete-event scripted control) by making possible a remote system that can perform decisions in real time within a dynamic environment using high-level artificial intelligence techniques combined with robotic behavioral control and machine vision.

Figure 1.5: The servicer robot captures the satellite using vision in harsh lighting conditions approximating those in orbit.

In the second part of this thesis, we develop an intelligent perception system whose purpose is to solve difficult space robotics problems; in particular satellite rendezvous and docking (AR&D) (Figure 1.4). Our system combines low-level object recognition and tracking with high-level symbolic reasoning, in a novel architecture that we call the *Cognitive Controller* or CoCo. The reasoning module, which encodes a model of the environment, performs deliberation to 1) guide the vision system in a task-directed manner, 2) activate vision modules depending on the progress of the task, 3) validate the performance of the vision system, and 4) suggest corrections to the vision system when the latter is performing poorly. Reasoning and related elements, among them intention, context, and memory, contribute to improve the performance (i.e., robustness, reliability, and usability). We demonstrate the prototype vision system in an MDRobotics Ltd. lab environment emulating relevant conditions in space that can autonomously capture a satellite (Figure 1.5).

1.3 Thesis Contributions

Although intelligent perception is the theme that underlies our research, the work reported in this thesis straddles multiple disciplines, not just machine vision, but also classical artificial intelligence, sensor networks, and computer graphics. Our work also contributes to the fields of agent architectures, space robotics, and multi-camera surveillance systems (sometimes referred to as visual sensor networks). The research presented in this thesis has appeared in part in the following publications: [Qureshi et al. 2004b] [Qureshi et al. 2004a] [Qureshi et al. 2005a]

[Qureshi et al. 2005b] [Qureshi and Terzopoulos 2005a] [Qureshi and Terzopoulos 2005b] [Qureshi and Terzopoulos 2006a] [Qureshi and Terzopoulos 2006b]. Our specific contributions are as follows:

- Combining computer graphics and computer vision, we develop and demonstrate the advantages of the virtual vision paradigm in designing, experimenting with, and evaluating a prototype large-scale surveillance system [Qureshi and Terzopoulos 2005b; Qureshi and Terzopoulos 2005a]. Other researchers should be able to use our virtual vision simulator and build upon our results, or do the work necessary to implement our new algorithms in physical camera networks.

- In the context of sensor networks and visual surveillance, we propose a novel camera network control strategy that does not require camera calibration, a detailed world model, or a central controller [Qureshi and Terzopoulos 2005b]. The overall behavior of the network is the consequence of the local processing at each node (camera) and inter-node communication. The network is robust to node and communication link failures; moreover, it is scalable due to the lack of a central controller. Visual surveillance tasks are performed by groups of one or more camera nodes. These groups, which are created on the fly, define the information sharing parameters and the extent of collaboration between nodes. A group keeps evolving—i.e., old nodes leave the group and new nodes join it—during the lifetime of the surveillance task. One node in each group acts as the group supervisor and is responsible for group level decision making.

- We present a new constraint satisfaction problem formulation for resolving group-group interactions among multiple cameras.

- We propose a sensor management scheme that appears well suited to the challenges of designing camera networks for surveillance applications that are potentially capable of fully automatic operation.

- Furthermore, our effort has resulted in 1) new image-driven pan/tilt and zoom controllers for active PTZ cameras, 2) an automatic scheme for learning the mapping between 3D points and the internal pan/tilt settings of a PTZ camera, and 3) an online surveillance camera scheduling scheme [Qureshi and Terzopoulos 2005a].

- In the context of space robotics, we develop an intelligent vision system capable of capturing a non-cooperative, free-flying satellite without human assistance [Qureshi et al. 2004b; Qureshi et al. 2005b]. The system is novel and unique inasmuch as it is the only AR&D system that uses vision as its primary sensory modality and can deal with an uncooperative target satellite. Other AR&D systems either deal with target satellites that communicate with the servicer craft about their heading and pose, or use other sensing aids, such as radars and geostationary position satellite systems. Our autonomous satellite rendezvous and docking demonstration is a proof of concept. The fact that Boeing won the contract for building such a prototype satellite rendezvous and docking system is in part due to our work. We understand that interested space agencies and contractors, notably including NASA and Boeing, were planning to develop a prototype system for in-orbit testing. NASA launched the Orbital Express in September 2006.

- An important technical contribution of our work is CoCo, which is a new approach to high-level control architectures for vision-based autonomous robots [Qureshi et al. 2004a]. CoCo's reactive module is an ethologically-inspired behavior-based system, whereas its deliberative module is based on cognitive robotics. CoCo's deliberative module can support multiple specialized planning modules, which makes it truly taskable. CoCo also features a powerful and non-intrusive scheme for combining deliberation and reactivity, which heeds advice from the deliberative module only when it is safe to do so. Here, the deliberative module advises the reactive module through a set of motivational variables. In addition, the reactive module presents the deliberative module with a tractable, appropriately-abstracted interpretation of the real world. The reactive mod-

ule constructs and maintains the abstracted world state in real-time using contextual and temporal information.

1.4 Thesis Overview

The remainder of the thesis is presented in two major parts. Chapters 2–6 comprise Part I on sensor networks, while Chapters 7–9 comprise Part II on space robotics.

Chapter 2 introduces and motivates our work on visual sensor networks, surveillance, and virtual vision. Chapter 3 presents relevant literature on camera networks. We develop visual analysis routines used by our virtual camera networks in Chapter 4. We demonstrate the virtual vision paradigm by developing two camera networks capable of strategic visual surveillance tasks with minimal reliance on a human operator in Chapters 5 and 6. Specifically, in Chapter 5 we present a camera scheduling strategy that enables active PTZ cameras to capture high resolution video of the pedestrians present in the designated area. In Chapter 6 we present a fully distributed camera network capable of observing a pedestrian as he meanders through the fields of view of multiple active PTZ cameras.

Chapter 7 introduces and motivates our work on satellite autonomous rendezvous and docking. Chapter 8 briefly reviews the relevant background literature. Chapter 9 presents the theoretical and practical aspects of our AR&D system implementation.

Part III concludes this thesis. Chapter 10 summerizes our work and suggests possible directions for future research.

Finally, we present appendices in Part IV. Chapter A gives the details of $\alpha\beta$ tracker for satellite pose validation. Chapter B explains vision modules handover for our autonomous satellite rendezvous and controller. Chapter C describes fuzzy logic based sensor fusion for our space robotics application and Chapter D provides some background on representing rotations using quaternions.

Part I

Intelligent Perception for Visual Sensor Networks

Chapter 2

Introduction and Motivation

As we mentioned in Chapter 1, a timely challenge for computer vision researchers is to design camera sensor networks capable of performing visual surveillance tasks automatically, or at least with minimal human intervention. We regard the design of an autonomous visual sensor network as a problem in resource allocation and scheduling, where the sensors are treated as resources required to complete the desired sensing tasks. Imagine a situation where the camera network is asked to capture high-resolution videos of every pedestrian that passes through a region of interest.[1] Passive cameras alone cannot satisfy this requirement and active PTZ cameras must be recruited to capture high-quality videos of pedestrians. Often there will be more pedestrians in the scene than the number of available cameras, so the PTZ cameras must intelligently allocate their time among the different pedestrians. A resource management strategy can enable the cameras to decide autonomously how best to allocate their time to observing the various pedestrians in the scene. The dynamic nature of the observation task further complicates the decision making process; e.g., the amount of time a subject spends in the designated area can vary dramatically between different pedestrians, an attempted video recording by a PTZ camera might fail due to occlusion, etc.

[1] The captured video can subsequently be used for further biometric analysis, e.g., by a facial, gesture, or gait recognition routine.

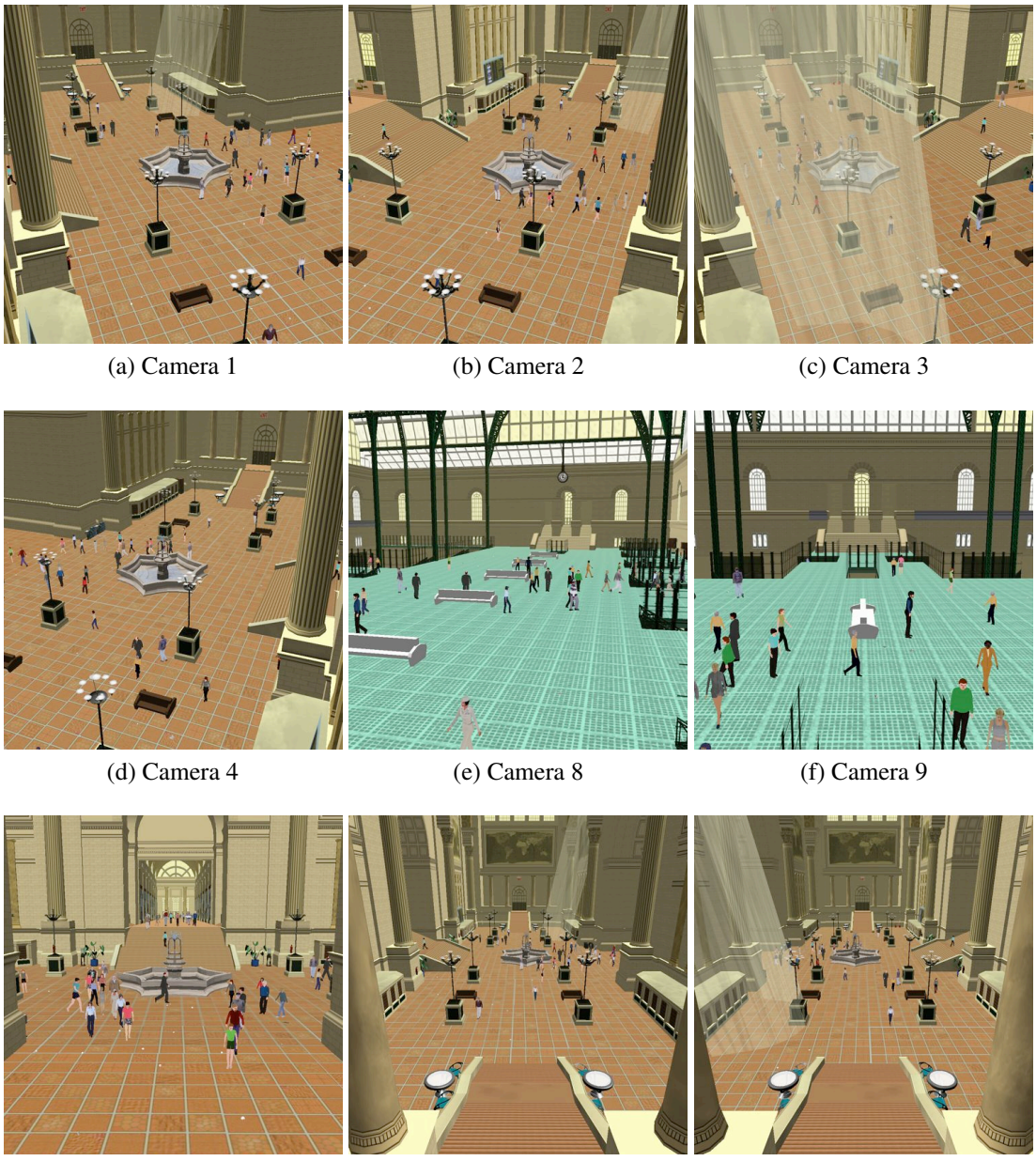
(a) Waiting Room (b) Concourses and Platforms

(c) Arcade (d) Concourses and Platforms

Figure 2.1: A large-scale virtual train station populated by self-animating virtual humans (Courtesy Shao and Terzopoulos).

2.1 The Virtual Vision Paradigm

Deploying a large-scale surveillance system is a major undertaking whose cost can easily be prohibitive for most computer vision researchers interested in designing and experimenting with multi-camera systems. Moreover, privacy laws impede the monitoring of people in public spaces for experimental purposes. To overcome these obstacles, Terzopoulos [2003] proposed a *Virtual Vision* approach to designing surveillance systems using a virtual train station environment populated by fully autonomous, lifelike virtual pedestrians that perform various activities



(a) Camera 1 (b) Camera 2 (c) Camera 3

(d) Camera 4 (e) Camera 8 (f) Camera 9

(g) Camera 10 (h) Camera 15 (i) Camera 16

Figure 2.2: Synthetic video feeds from multiple virtual surveillance cameras situated in the Penn Station environment. Camera numbers represents the cameras shown in Figure 2.3.

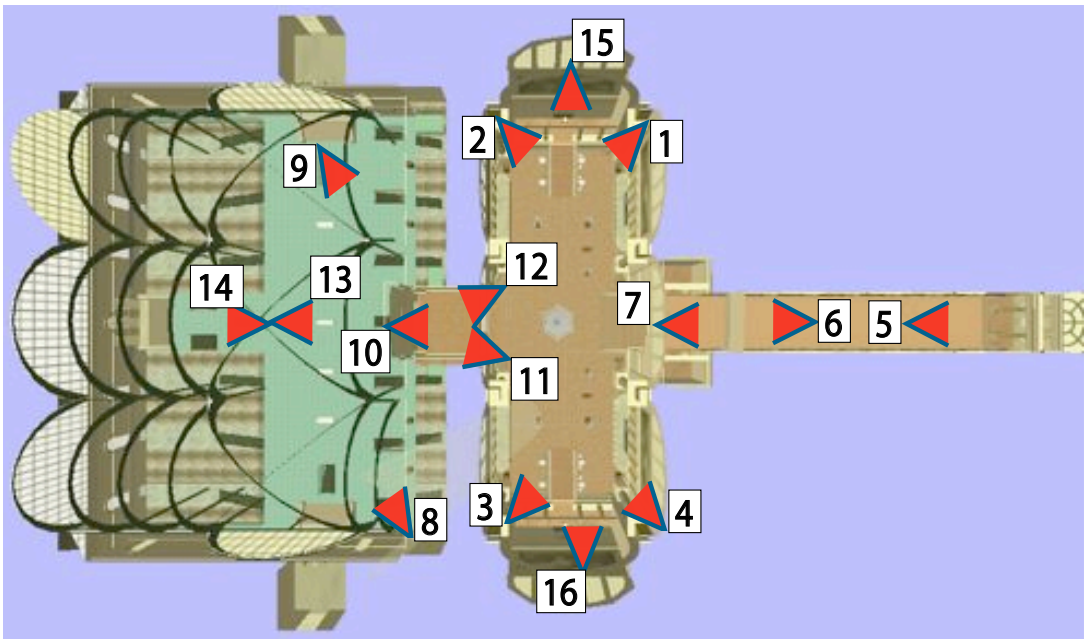


Figure 2.3: Plan view of the virtual Penn Station environment with the roof not rendered, revealing the concourses and train tracks (left), the main waiting room (center), and the long shopping arcade (right). (The yellow rectangles indicate station pedestrian portals.) An example visual sensor network is shown comprising 16 simulated active (pan-tilt-zoom) video surveillance cameras. Figure 2.2 shows a sampling of images captured by cameras 1, 2, 3, 4, 8, 9, 10, 15, and 16.

(Figure 2.1) [Shao and Terzopoulos 2005a]. Cost considerations and legal impediments aside, the use of realistic virtual environments also offers significantly greater flexibility during the design and evaluation cycle, thus enabling many more iterations of the scientific method.

Pursuing the virtual vision philosophy, we demonstrate a surveillance system comprising static and active simulated video cameras that provide perceptive coverage of a large virtual public space; in our case, a train station (Figure 2.3), a reconstruction of the original Pennsylvania Station in New York City, which was demolished in 1963. The virtual cameras situated throughout the expansive chambers of the station generate multiple synthetic video feeds (Figure 2.2) that emulate those generated by real surveillance cameras monitoring public spaces. The station is populated by autonomously self-animating virtual pedestrians (Figure 2.1). The advanced pedestrian animation system combines behavioral, perceptual, and cognitive human simulation algorithms [Shao and Terzopoulos 2005a]. The simulator can efficiently synthe-

size well over 1000 self-animating pedestrians performing a rich variety of activities in the large-scale indoor urban environment. Like real humans, the synthetic pedestrians are fully autonomous. They perceive the virtual environment around them, analyze environmental situations, make decisions and behave naturally within the train station. They can enter the station, avoiding collisions when proceeding though portals and congested areas, queue in lines as necessary, purchase train tickets at the ticket booths in the main waiting room, sit on benches when they are tired, purchase food/drinks from vending machines when they are hungry/thirsty, etc., and eventually proceed to the concourse area and down to the train tracks. A graphics pipeline renders the busy urban scene with considerable geometric and photometric detail, as shown in Figure 2.1.

Our unique combination of computer vision and advanced graphics technologies offers several advantages:

1. Our simulator runs on (high-end) commodity PCs, obviating the need to grapple with special-purpose hardware and software.

2. The virtual cameras are very easily relocated and reconfigured in the virtual environment.

3. The virtual world provides readily accessible ground-truth data for the purposes of surveillance algorithm/system validation.

4. Simulation time can be prolonged relative to real, "wall-clock time"; i.e., arbitrary amounts of computation can be performed per simulation time unit, thereby enabling one to evaluate the competence of collections of sophisticated visual surveillance algorithms that cannot currently be expected to run in real time.

5. Experiments are perfectly repeatable in the virtual world, so we can easily modify algorithms and parameters and immediately determine their effect.

2.2 The Surveillance System

Within the virtual vision paradigm, we develop and evaluate a visual sensor network consisting of fixed, wide field-of-view (FOV) passive cameras and PTZ active cameras. We develop novel multi-camera control strategies that enable the simulated camera nodes to collaborate both in tracking pedestrians of interest that move across the FOVs of different cameras and in capturing close-up videos of pedestrians as they travel through designated areas. The network supports task-dependent node selection and aggregation through local decision-making and inter-node communication. Treating node selection as a constraint satisfaction problem, we propose a solution that is scalable and robust against node failures, since it lacks a central controller.

For the task of capturing high-quality videos of pedestrians as they move through a designated area, we assume that the wide-FOV stationary cameras are calibrated,[2] which enables the network to estimate the 3D locations of pedestrians through triangulation. However, we do not require the PTZ cameras to be calibrated. Rather, during a learning phase, the PTZ cameras learn a coarse mapping between the 3D locations and the gaze-direction by observing a single pedestrian in the scene. A precise mapping is unnecessary since we model each PTZ camera as an autonomous agent that can invoke a search behavior to find the pedestrian using only coarse hints about the pedestrian's 3D position. The network uses a weighted round-robin strategy to assign PTZ cameras to the various pedestrians. Each pedestrian creates a new sensing request in the task queue. Initially, each sensing request is assigned the same priority; however, the decision making process uses domain-specific heuristics, such as the distance of the pedestrian from a camera or the heading of the pedestrian, to evaluate continuously the priorities of the sensing requests. The PTZ cameras handle each task in priority sequence. A warning is issued when a sensing request cannot be met.

[2]This assumption is justifiable given the success of numerous automatic static camera calibration schemes [Pedersini et al. 1999; Gandhi and Trivedi 2004].

Chapter 3

Related Work

Our virtual vision approach towards designing smart camera networks is made possible by recent developments in advanced modeling of virtual environments populated with autonomous, artificial humans, advances in computer vision towards detecting, identifying, and tracking targets, and advances made by the sensor networks community. In this chapter, we review approaches related to our virtual vision paradigm, and we provide background on areas of multi-camera systems and sensor networks. Both offer unique challenges and opportunities for developing and studying distributed problem solving; thus, we also discuss distributed problem solving, albeit briefly. We conclude the chapter with a comparative summary highlighting the novel aspects of our work.

3.1 Artificial Worlds for Computer Vision

Ten years ago, Terzopoulos and Rabie introduced a purely software-based approach to designing active vision systems, called *animat vision* [Terzopoulos and Rabie 1997]. The animat vision approach prescribes replacing real, hardware cameras/robots, which are typically used by computer vision researchers, with artificial animals, or animats [Wilson 1990], situated in physics-based virtual worlds to study and develop active vision systems. They demonstrate the animat vision approach by implementing biomemetic active vision systems for artificial fishes

and virtual humans [Rabie and Terzopoulos 2000]. These active vision systems comprise algorithms that integrate motion, stereo, and color analysis to support robust color object tracking, vision-guided navigation, visual perception, and obstacle recognition and avoidance abilities. Together, these algorithms enable the artificial animal to sense, understand, and interact with its dynamic virtual environment. The animat vision approach appears particularly useful for modeling and ultimately reverse-engineering the powerful vision systems found in higher-level animals. Furthermore, it obviates the need for grappling with real hardware—cameras, robots, and other paraphernalia—at least during the initial stages of research and development, thereby yielding in huge savings in terms of both the effort involved in maintaining the supporting hardware and the cost involved in acquiring it. The algorithms developed within the animat vision approach were later adapted for a mobile vehicle tracking and traffic control system [Rabie et al. 2002], which is a testimony to the usefulness of animate vision approach for designing and evaluating complex computer vision systems.

As mentioned earlier, Terzopoulos [2003] proposed the *Virtual Vision* paradigm for video surveillance systems research. He envisions designing and evaluating video surveillance systems using *Reality Emulators*. These are virtual environments of considerable complexity, inhabited by autonomous, lifelike agents. The work presented in the next few chapters realizes his vision. We have developed our virtual camera networks within a reality emulator developed by Shao and Terzopoulos [2005b; 2005a]—a virtual train station populated with lifelike, self-animating pedestrians.

In concordance with the virtual vision paradigm, Santuari et al. [Santuari et al. 2003; Bertamini et al. 2003] advocate the development and evaluation of pedestrian segmentation and tracking algorithms using synthetic video generated within a virtual museum simulator containing scripted characters. Synthetic video is generated via a sophisticated 3D rendering scheme, which supports global illumination, pedestrians' shadows, and visual artifacts like depth of field, motion blur, and interlacing. Presently they have used their virtual museum environment to develop static background modeling, pedestrian segmentation, and pedestrian

tracking algorithms. They focus on low-level computer vision, whereas our work goes far beyond this and focuses on high-level computer vision issues, especially multi-camera control in large-scale camera networks.

3.2 Multi-Camera Systems

Previous work on multi-camera systems has dealt with issues related to low and medium-level computer vision, namely identification, recognition, and tracking of moving objects [Collins et al. 2002; Comaniciu et al. 2002; Trivedi et al. 2000; Stillman et al. 1998; Javed et al. 2003]. The emphasis has been on tracking and on model transference from one camera to another, which is required for object identification across multiple cameras [Khan and Shah 2003]. Multiple cameras have also been employed either to increase the reliability of the tracking algorithm [Kang et al. 2003] (by overcoming the effects of occlusion or by using 3D information for tracking) or to track an object as it meanders through the fields of view (FOVs) of different cameras. In most cases, object tracking is accomplished by combining some sort of background subtraction strategy and an object appearance/motion model [Siebel 2003]. Numerous researchers have proposed camera network calibration to achieve robust object identification and classification from multiple viewpoints, and automatic camera network calibration strategies have been proposed for both stationary and actively controlled camera nodes [Pedersini et al. 1999; Gandhi and Trivedi 2004; Devarajan et al. 2006]. References [Ihler et al. 2004; Marinakis et al. 2005; Mallett 2006] present schemes to learn sensor (camera) network topologies.

Little attention has been paid, however, to the problem of controlling or scheduling active cameras when there are more objects to be monitored in the scene than there are active cameras. Some researchers employ a stationary wide-FOV camera to control an active tilt-zoom camera [Collins et al. 2001; Zhou et al. 2003; Costello et al. 2004; Hampapur et al. 2003]. Generally speaking, the cameras are assumed to be calibrated and the total coverage of the cameras is

restricted to the FOV of the stationary camera. We discuss some of the more relevant systems in the following section.

3.2.1 Video Surveillance

Video surveillance is among the most recognized applications of multi-camera systems and in the last five years a large fraction of multi-camera systems have been developed for the purposes of video surveillance. We restrict our discussion to video surveillance systems that combine passive and active cameras in order to provide multi-resolution visual coverage of the designated area, as these are more relevant to the work presented in this thesis. We point the reader to [Hu et al. 2004] for a recent survey of the many image analysis techniques being explored in the context of the video surveillance.

Video surveillance is becoming ever more pervasive in today's society [Marcenaro et al. 2001]. Presently, it is used for transportation monitoring [Pavlidis et al. 2001], urban security [Kettnaker and Zabih 1991], tourism [Sacchi et al. 1999], and battlefield awareness [Oppelt 1995].[1] As the size of the surveillance system grows, however, the enormous flux of video data renders the central monitoring paradigm—a hallmark of the so called first [Collins and Williams 1961] and second [Sacchi et al. 1999] generation surveillance systems—infeasible. First and second generation surveillance systems rely solely on the human operator to identify events of interest. Most surveillance systems in use today are said to be third generation systems. These employ visual analysis to assist the operator in monitoring events of interest. Still they adhere to the central processing/monitoring paradigm and skirt networking and intelligent camera control issues. Here, the monitoring center is a potential bottleneck, which can adversely affect the overall scalability of third generation surveillance systems. Researchers acknowledge the limitations of third generation surveillance systems and stress the need for more flexible architectures that support distributed computing and employ intelligent and mi-

[1] Interestingly, video surveillance has been used by military and security agencies for the last four decades, since the invention of closed-circuit television cameras, commonly known as CCTVs.

gratory agents as part of the solution [Marcenaro et al. 2001]. There is currently considerable interest within the research community, and also in the industry, to develop next generation video surveillance systems.

Enhancing situational awareness is the foremost purpose of a surveillance system, smart or otherwise. Situational awareness, argues Hampapur et al. [2005], requires information to be collected and analyzed at various spatio-temporal scales. Low-resolution video, for example, is sufficient for detecting and tracking a person, yet identifying a person typically requires a closeup facial snapshot. It is infeasible to provide high-resolution video coverage of a large space, such as an airport or train station, using static cameras alone. A possible solution is to design surveillance systems comprising both static and active PTZ cameras. Static cameras collect low-spatial, high-temporal resolution data, whereas PTZ cameras collect high-spatial and low-temporal resolution data. Static and PTZ cameras thus complement each other and collaborate to provide multi-scale visual coverage of the designated area.

The Visual Surveillance and Monitoring (VSAM) project looked at several fundamental issues in detection, tracking, classification, and *in-situ* calibration in multi-camera systems [Collins et al. 2000]. Several enabling technologies, such as robust target tracking, localization, event detection, and recognition [Haritaoglu et al. 1998; Stauffer and Grimson 2000], were developed within the umbrella of VSAM. Relevant to the work presented here are the two camera coordination strategies investigated for VSAM—camera handover and camera slaving. During camera handover, the 3D location of the target is used to select the most suitable camera to attend to the target. Here, the camera selection is based on the proximity of the camera to the target. The 3D location of the target is estimated either through triangulation or by intersecting the backprojected viewing rays with the terrain. In camera slaving, a calibrated wide field-of-view stationary camera drives an active pan/tilt camera to view the target. The viewing direction of the active camera is computed by establishing the correspondences between the 3D locations and the pan/tilt settings of the active camera. When the 3D location of a pedestrian is not available, the viewing direction for the active camera (to bring the pedestrian within the

field of view) is computed using a learned mapping between the pixel locations in the static camera image and the internal pan/tilt settings of the active camera [Collins et al. 2001].

Karuppiah et al. present a multi-camera system consisting of panoramic and PTZ cameras [Karuppiah et al. 2001; Zhu et al. 2000]. A noteworthy feature of this system is the lack of a central processing station. Panoramic cameras together form a virtual stereo sensor capable of estimating the 3D location of a target either through triangulation or via monocular cues. The active cameras are calibrated and rely on the target's 3D position while servoing to keep it in view. The system is organized into three layers. Sensors (panoramic and PTZ cameras) constitute the lowest layer. The second layer comprises the software entities, called Resource Managers, that establish the communication between the various sensors. User Agent entities, which make up the third layer, enable the resource managers to communicate with the user.

A similar approach involving calibrated static and pan/tilt cameras is presented in [Micheloni et al. 2005]. Data from multiple static cameras is fused to estimate the 3D location of the pedestrian. An active camera uses calibration information to bring the target into the center of its view. After initial repositioning, the active camera autonomously tracks the target [Foresti and Micheloni 2003], thereby avoiding the communication overhead associated with master-slave configurations. The active camera periodically sends its pan/tilt settings to the static cameras. The static cameras can use this information to decide whether or not the active camera is tracking the correct target. If it is not, the active camera is repositioned. Foresti and Micheloni [2003] propose a robust real-time tracking system for outdoor image sequences using an active pan/tilt camera. The system assumes a moving object (the target) against a mostly static background. The motion induced during the pan/tilt operation is modeled via translation, which is estimated by tracking features that belong to the background. The Lucas-Kanade feature tracker is employed for this purpose. The current frame and the motion-compensated background image are processed to locate mobile objects. This scheme does not handle the zoom operation, as the camera motion induced between two frames during a zoom operation violates the assumption that the two successive frames are related through a translation.

Remaginino et al. [2001] describe a two-camera system that employs image analysis—specifically, object tracking, stereo-localization, and activity inference—to assist the operators in the monitoring task.

Collins et al. [2002] demonstrate a surveillance system where multiple calibrated active PTZ cameras track a single person in 3D. Here, the PTZ cameras actively adjust their PTZ settings to track the target. The 3D position of the target is estimated through triangulation when the target is visible in multiple cameras, and the 3D position is used to select appropriate pan/tilt and focus settings to maintain the target within the field-of-view. This allows a PTZ camera to follow the estimated position of the target even during occlusions. PTZ cameras use appearance-based pedestrian signatures (mean-shift algorithm [Comaniciu et al. 2000]) to track a pedestrian. Cameras continuously servo to maintain the center of the tracked person within the image. The zooming operation requires detailed zoom-parameter calibration. Fixation parameters from multiple cameras are combined at a central location to estimate the 3D position of the target through triangulation.

Zhou et al. [2003] track a single person using an active camera. When multiple people are present in the scene, the person who is closest to the last tracked person is chosen.

Camera Scheduling

Ser-Nam et al. present a scheme for scheduling available cameras in a task-dependent fashion [Lim et al. 2003]. Here, the tasks are defined within a world model that consists of the ground plane, traffic pathways, and detailed building layouts. The scheduling problem is cast as a temporal logic problem that requires access to a central database consisting of current camera schedules, viewing parameters, and pedestrian trajectories. Unlike ours, this scheme requires a detailed scene model, which is cumbersome to acquire.

The work of Hampapur et al. [2003] deals with the issues of deciding how cameras should be assigned to various people present in the scene. They use PTZ cameras to capture close-up facial images of the persons present in the scene for recognition purposes. Unlike the work

presented here, however, they only provide general guidelines for assigning cameras to persons: 1) location specific, 2) orientation specific, 3) round-robin, and 4) activity-based assignment. They do not propose a scheduling scheme. Their system consists of multiple PTZ cameras driven by a 3D wide-baseline stereo tracking system (master-slave configuration).

Costello et al. [2004] evaluate various strategies for scheduling a single active camera to acquire biometric imagery of the people present in a scene. They evaluate three static priority policies—Random, First Come, First Serve (FCFS+), and Earliest Deadline First (EDF+)—and one non-static priority policy—Current Minloss Throughput Optimal (CMTO) [Givan et al. 2002]—on a Monte Carlo simulation that models pedestrian arrival as a Poisson process and pedestrian locomotion via a random waypoint model. They also demonstrate their approach on a cooperative active camera system comprising a passive and an active PTZ camera in a master-slave configuration. Here, a static camera estimates the 3D positions of the pedestrians (and vehicles) and drives the PTZ camera. Correspondences between the two cameras are established manually. They assume near perfect tracking during evaluations. They conclude that active camera scheduling is an online scheduling problem and that given the uncertainty inherent in the sensing process, a greedy approach is more suited to the task of scheduling active cameras. They show that EDF+ and CMTO perform poorly when the predictions about the pedestrians departure times are inaccurate. Furthermore, they stress that predicting pedestrian departure times is nontrivial. FCFS+ exhibits comparable performance to that of EDF+ when the pedestrians spend roughly the same amount of time in the designated area.[2] Unlike the work of Costello et al., our work can handle more than one active PTZ camera. Interestingly, Costello et al. argue that Monte Carlo simulation is necessary to evaluate camera scheduling approaches, as evaluating scheduling policies on an actual setup is extremely complicated. Our Virtual Vision approach provides a viable alternative to using an actual physical setup, at least during the development and evaluation phase.

Recently Costello and Wang [2005] propose a distributed scheduling algorithm for task-

[2]This assumption is true for a wide range of scenarios.

ing multiple active cameras to observe pedestrians. The proposed algorithm is scalable as it restricts communication between neighboring cameras; however, it operates under restrictive assumptions: 1) observation duration is constant for each person (i.e., no preemption), 2) the path that the person takes through the designated area is known *a priori* or can be predicted with reasonable accuracy (i.e., offline, clairvoyant), 3) there is no overlap between the regions of coverage of active cameras (to avoid potential camera coordination issues), 4) a person passes through the region of coverage of each camera only once, and 5) calibrated cameras and known network topology. The path taken by a person determines the time he spends in the regions of coverage of each active camera, and in turn, his entry and departure times with respect to each active camera. Active cameras employ the Earliest Deadline First (EDF) scheduling scheme to observe persons present in their regions of coverage. The list of observed persons is passed onto the next neighbor. The EDF scheduling algorithm is myopic and does not consider how its decisions will affect other active cameras in the future, which can adversely affect the performance of the network. Costello and Wang propose a load balancing algorithm to mitigate this effect. Each camera is aware of its neighbors' loads (the number of unobserved persons) and attempts to observe persons that appear to be moving towards the regions of coverage of those neighbors with the highest loads.

We refer the reader to [Stewart and Khosla 1992], which discuss real-time scheduling algorithms for sensor-based control systems. The problem of online scheduling has been studied extensively in the context of scheduling jobs on a multi-tasking computer [Bar-Noy et al. 2002; Sgall 1998] as well as for packet routing in networks [Ling and Shroff 1996; Givan et al. 2002].

3.3 Sensor Networks

Networked systems of small, untethered, battery-powered smart sensors is a new paradigm of computing with the potential to revolutionize the way people interact with their environments by linking together a range of devices/sensors that allow information to be collected, shared,

stored, and processed in new ways [Kahn et al. 1999; Estrin et al. 2001]. Typical applications of sensor networks include environmental monitoring, surveillance, inventory tracking, and smart spaces, among others [Sinopoli et al. 2003]. Camera networks are high-performance multimedia sensor networks, ergo sensor networks are seen as a key enabling technology for designing next generation video surveillance systems.

Unlike the data communication networks in use today, the goal of a sensor network is not just data communication, but event detection and estimation. Acceptable performance for event detection and estimation is typically achieved through the fusion of information from multiple nodes. Energy conservation is a huge concern for sensor networks, as it directly impacts the lifespan of the network. Self-organization is a crucial ability for sensor networks, as manual configuration of sensor networks is infeasible given the large number of nodes (10,000 or even 100,000 nodes). These constraints pose challenging engineering, networking, and distributed control problems, and currently, there is a considerable interest within the research community in developing communication and control strategies for node selection, localization, and aggregation, data routing, bandwidth optimization, and energy conservation in sensor networks [Akyildiz et al. 2002; Xing et al. 2005].

Node communications have large power requirements; therefore, sensor network control strategies attempt to minimize inter-node communication [Chang and Tassiulas 2000; Zhao et al. 2002]. In the Berkeley motes [Levis et al. 2004], for example, the ratio of energy consumption for communication and computation is in the range of 1000–10000. Researchers have proposed task-dependent node aggregation [Zhao et al. 2002], data routing algorithms [Intanagonwiwat et al. 2003], and multi-tier node organization [Kulkarni et al. 2005b] to limit the communication to relevant nodes.

The problem of forming sensor groups based on task requirements and resource availability has received much attention within the sensor networks community [Zhao et al. 2003]. Collaborative tracking, which subsumes the above issue, is considered an essential capability in many sensor networks. Zhao et al. [2002] introduces an information-driven approach to collab-

orative tracking, which attempts to minimize the energy expenditure at each node by reducing inter-node communication. A node selects the next node by utilizing the information gain vs. energy expenditure trade-off estimates for its neighbor nodes. Here, the network nodes are either acoustic or seismic sensors. Huang et al. [2002] start with an initial sparse tree topology of connected nodes and construct a more fully connected network over time. Directed diffusion is another scheme for automatically establishing communication paths in ad hoc networks [Intanagonwiwat et al. 2003].

The Berkeley motes is a recent example of smart sensor-based networked systems [Hill et al. 2000; Levis et al. 2004]. This work has led to a succession of more capable devices and new distributed control algorithms tailored towards sensor networks. Reference [Sinopoli et al. 2003] presents an overview of the research activities dealing with distributed control within sensor networks.

He et al. [2004] network 70 magnetometers (MICA2 motes) to construct an energy-aware surveillance system capable of detecting and tracking targets. Their fully automatic sensor network configures itself in multiple phases. The first phase, *initialization*, requires system-wide message broadcasts for time synchronization, network backbone creation, and system-wide reconfiguration. The nodes discover their immediate network structure during the second phase. A subset of the nodes is selected as sentry (leader) nodes in the third phase. Each sensing region requires at least one sentry. It is important to note here that sentry selection is not task dependent, rather its selection reflects local topological/geographical conditions. Next, in Phase 4, each node reports its status to the base station. Non-sentry nodes alternate between sleep and wake states for power conservation. Each mote that is awake sends its location to the base station upon detecting an event for tracking purposes.

3.4 Smart Camera Networks

Smart cameras are an essential component of the next generation camera networks. These are self-contained vision systems, complete with image sensors, power circuitry, communication interface, and on-board processing capabilities. Smart cameras promise to reduce the communication overhead by distributing the processing across across multiple nodes, effectively turning the network itself into one giant computer. We refer the reader to Section 2.2 of [Mallett 2006], which discusses currently available smart cameras.

The centralized processing paradigm is clearly infeasible for large networks. Distributed processing strategies that restrict communication to the relevant nodes provide a possible solution. Task dependent node aggregation, argues Mallett [2006], is an essential capability to keep the communication overhead within acceptable limits in large networks. She proposes a node aggregation protocol to create a bottom-up organization among nodes based on their locality. She demonstrates her approach by partitioning a camera network into the groups of cameras with overlapping fields of view.

IrisNet is a sensor network architecture tailored towards high-capability multi-media sensors connected via high-capacity communication channels [Gibbons et al. 2003; Campbell et al. 2005]. IrisNet takes a *centralized* view of the network and models it as a distributed database. The distributed database infrastructure provided by IrisNet supports load balancing and data replication. The data is organized into geographical hierarchies using application-specific XML schemas. Data organization allows efficient access to sensor readings. Campbell et al. [2005] develop a parking space finder application within IrisNet, which is intended to direct a driver to the nearest available parking spot. Calibrated passive video cameras identify vacant parking spots via image analysis (background analysis). A coastal imaging application has also been developed using IrisNet. Cameras deployed along the Oregon coastline monitor near-shore phenomena, such as riptides and sandbar formations. Data collected by multiple cameras is combined to provide a composite overhead view of the coastline. To support these applications Campbell et al. have developed low-level image analysis routines for passive camera calibration and image stitching.

A recent sensor-network-inspired multi-camera system is SensEye [Kulkarni et al. 2005b; Kulkarni et al. 2005a]. It is an energy-aware camera network comprising three classes of smart camera nodes: 1) Mote nodes equipped with low-fidelity Cyclops or CMUcam camera sensors[3], 2) Stargate nodes equipped with web-cams[4], and 3) high-resolution PTZ cameras connected to PCs. SensEye demonstrates the benefits of a multi-tiered network—each tier defines a set of sensing capabilities and corresponds to a single class of sensors—over single-tiered networks in terms of low-latencies and energy efficiency. Energy efficiency is achieved by adhering to the following three principles: waking up nodes on demand, favouring the least powerful tier with sufficient resources, and exploiting camera redundancy to localize the events of interest. SensEye assumes calibrated sensor nodes and a known network topology, which limits its capabilities, yet perhaps the biggest weakness of the proposed architecture is that inter-tier and intra-tier interactions results in a much more complex distributed control problem. The authors have demonstrated SensEye on four tasks, object detection (background subtraction), localisation (triangulation), recognition (face recognition plus color signatures), and tracking (a combination of detection, localisation, and recognition). The main operation is as follows: Tier 2 and 3 cameras are kept deactivated to conserve energy and Tier 1 cameras are cycled between the on and off state. When a Tier 1 camera detects an object, Tier 2 and 3 cameras are activated for further processing.

3.5 Self-Organization and Distributed Problem Solving

Grouping computer processes to design fault-tolerant, real-time applications, such as air traffic controllers and stock market visualization tools, is well-established in computer science [Birman et al. 2000]. Process grouping protocols operate under the assumption that processes can reliably communicate with each other. It is unclear how process grouping strategies developed

[3]Crossbow wireless sensor platform: `http://www.xbow.com/Products/Wireless_Sensor_Networks.htm` (Last accessed on 25 January 2007)

[4]Stargate Platform: `http://www.xbow.com/Products/xscale.htm` (Last accessed on 25 January 2007)

for real-time control can be adapted for sensor networks where lost messages are the norm, not the exception. Moreover, without significant communication overhead, it is difficult to ensure that messages arrive at the destination in order. These issues assume even greater importance for sensor networks that are spread over a large geographical area.

Self-organizing protocols are also found in peer-to-peer file sharing applications, such as Napster and Gnutella. In early instances of peer-to-peer applications, a node would send requests to every other node within a specified radius (number of links), which can potentially lead to a large communication overhead. Napster resolved this issue by requiring nodes to register to a central server to find other relevant nodes in the vicinity. Here, the central server acts as a bottleneck [Biddle et al. 2003]. In some purely peer-to-peer applications, each node randomly establishes connections with other nodes [Markatos 2002]. Another solution is to organize nodes hierarchically. A few nodes, called super peers, act as aggregation points for less capable nodes and behave as routing intermediaries [Yang and Garcia-Molina 2003].

Distributed problem solving is closely related to the resource allocation problem in multi-agent systems, which is an active area of research in computer science and economics [Chevaleyre et al. 2006]. Distributed vehicle monitoring as an example application of distributed situation assessment and more generally distributed resource allocation has been studied in the multi-agent systems community since its infancy [Smith 1980; Lesser and Erman 1980; Conway et al. 1983]. Multi-agent resource allocation problems can be studied along various dimensions, namely agent preferences and externalities, allocation strategies (centralized vs. distributed), and objectives (feasible, optimal, or social welfare). For the moment, we ignore agent preferences and objectives, which are typically implicitly defined within sensor networks. Resource allocation strategies studied in multi-agent systems, however, are relevant to sensor networks. Generally speaking, resource allocation strategies for multi-agent systems could be either centralized [Cramton et al. 2006] or distributed [Smith 1980]. We are interested in distributed resource allocation schemes.

3.5.1 Negotiation as a Means to Distributed Problem Solving

The Contract-Net scheme, proposed by Smith [1980], is perhaps the most widely used approach for distributed resource allocation in multi-agent systems. Initially proposed as a general distributed problem solving paradigm, it was quickly adapted by the multi-agent systems and sensor networks community. Interestingly, in their 1983 article, Davis and Smith [1983] explained the Contract-Net protocol using a distributed sensing scenario. Contract-Net is a bilateral trading model consisting of four interactions:

Announcement: An agent (node), referred to as the manager, advertises the resource/task to a number of other nodes.

Bidding: The interested nodes, referred to as the contractors/bidders, send their bids to the manager node.

Assignment: The manager elects the best bid and assigns the resource accordingly.

Confirmation: The selected bidders confirm their intention to take the resource/perform the task.

Any node can assume the role of a manager and commence the resource allocation/problem solving phase by following the Contract-Net protocol.

Many extensions to the Contract-Net protocol have been proposed; reference [Chevaleyre et al. 2006] presents a brief overview of some of the more notable extensions. It is known that when multiple managers negotiate simultaneously with many contractors, the Contract-Net protocol leads to unsatisfactory results [Aknine et al. 2004]. The Concurrent Contract-Net protocol attempts to resolve this issue by 1) allowing temporary bidding and assignments and 2) allowing a bidder node to go back on its commitment. It is worthwhile to keep in mind that the Concurrent Contract-Net protocol obviously has higher communication requirements than that of its classical counterpart.

In distributed problem solving, argue Davis and Smith [1983], no single agent (node) has a complete view of all the activities in the system. Consequently, coherent and organized overall behavior is difficult to guarantee, as an appropriate local organization might not be globally optimal. Distributive systems that attempt to maintain correctness in all aspects of the computation require the local knowledge residing at each node to be correct and consistent. Such distributive systems can be viewed as a centralized system distributed over a network. Lesser and Corkill [Lesser 1991] call such systems, *completely accurate, nearly autonomous* (CA/NA).[5] Such systems are not suitable for sensor networks where algorithms and control structures cannot be replicated and partitioned. Besides, CA/NA approaches come with a significant communication overhead to ensure that the local view at each node is consistent.

Lesser and Corkill [1981] propose an alternative to CA/NA, calling it *functionally accurate, cooperative* (FA/C) distributed systems. Here, local decisions made at each node may lead to unnecessary, redundant, or incorrect processing; however, the system still produces acceptable results. It is expected that the amount of additional communication resulting from incorrect local decisions is less than that required to ensure that the nodes are in consistent states across the network. This notion stems from the idea of *satisficing* problem solving, which explains how complex organizations are able to operate under significant uncertainty coupled with bounded rationality [March and Simon 1958].

Resolving group-group interactions requires sensor assignment to various tasks, which shares many features with Multi-Robot Task Allocation (MRTA) problems studied in multi-agent systems community [Gerkey and Matari 2004]. Specifically, according to the taxonomy provided in [Gerkey and Matari 2004], our sensor assignment formulation belongs to the single-task robots (ST), multi-robot tasks (MR), instantaneous assignment (IA) category. ST-MR-IA are significantly more difficult than single robot task MTRA problems. Task-based robot grouping arises naturally in ST-MR-IA problems, which are sometimes referred to as *coalition formation*. ST-MR-IA is extensively studied and can be reduced to a Set Partition-

[5] IrisNet is an example of CA/NA distributed system.

ing Problem (SPP), which is strongly NP-hard [Garey and Johnson 1978]. However, many heuristics-based set partitioning algorithms exist that have shown to produce good results on large SPPs [Atamturk et al. 1995]. Fortunately, the sizes of MRTA problems, and by extension SPPs, encountered in our camera sensor network setting are small due to the spatial/locality constraints inherent to camera sensors.

We model sensor assignments as a CSP, which we solve using "centralized" backtracking. We have intentionally avoided distributed constraint optimization techniques, such as [Modi et al. 2006] and [Yokoo 2001], due to their explosive communication requirements even for small CSPs. Additionally, it is not obvious how these handle node and communication failures. We consider our strategy to lie somewhere between purely distributed and fully centralized schemes for sensor assignments: Sensor assignment is distributed at the level of the network; whereas, it is centralized at the level of a group.

3.5.2 Active Camera Scheduling

The work presented here differs from existing camera scheduling systems in several important ways: First, it can handle multiple PTZ cameras. Second, the proposed scheduling strategy supports preemption and a multi-class pedestrian model. Third, the PTZ cameras are modeled as autonomous agents that are not driven by the passive cameras. Master-slave configurations, ubiquitous in surveillance systems comprising passive and active PTZ cameras, are inherently limited, not to mention fragile. They do not exploit the full potential of such camera systems. Fourth, the PTZ cameras can automatically learn the mapping between the 3D locations in the world and their internal pan/tilt settings by fixating on a pedestrian during an initial learning phase (Section 4.5). When available, the mapping is used to suggest an initial "look" direction to a PTZ camera; otherwise, the PTZ camera performs an exploratory sweep to find the desired pedestrian.

3.5.3 Smart Camera Network

Our camera network model presented in Chapter 6 has many novel aspects. It does not require camera calibration, a detailed world model, or a central controller. The overall behavior of the network is the result of local computation at each node and bi-lateral interactions between neighboring nodes. Inter-node communications is patterned after the Contract-Net model; however, we augment the communication protocol to address the unique challenges posed by sensor networks, namely unreliable communication and node failures. The network is robust to node and communication link failures; moreover, it is scalable due to the lack of a central controller. Visual surveillance tasks are performed by groups of one or more camera nodes. These groups, which are created on the fly, define the information sharing parameters and the extent of collaboration between nodes. A group keeps evolving—i.e., old nodes leave the group and new nodes join it—during the lifetime of the surveillance task. One node in each group acts as the group supervisor and it is responsible for group level decision making. We also present a novel constraint satisfaction problem formulation for resolving group-group interactions.

As mentioned earlier, Zhao et al. [2002] propose an energy-aware approach to collaborative tracking, where each node selects the next node by utilizing the information gain vs. energy expenditure tradeoff estimates for its neighbor nodes. Within the context of camera networks, it is often difficult for a camera node to estimate the expected information gain by assigning another camera to the task without explicit geometric and camera-calibration knowledge—such knowledge is tedious to obtain and maintain during the lifetime of the camera network. The camera networks presented here, therefore, do without such knowledge, and a node needs to communicate with the nearby nodes before selecting new nodes.

Mallett [2006] also proposes task-specific camera grouping to aid distributed problem solving. However, our work differs from theirs in the following ways: 1) group formation is through mutual selection, 2) leader nodes, one for each group, manage group memberships and group-group interactions, 3) each node can only belong to a single group at any given time, and 4)

group destruction does not require dedicated group destruction nodes.

Our work is in some sense orthogonal to the SensEye camera sensor network [Kulkarni et al. 2005b]. SensEye demonstrates that a multi-tier camera network can result in reduced energy consumption while providing surveillance capability comparable to that of a single-tier network. SensEye, however, does not address the camera grouping and calibration-less active camera control issues.

Chapter 4

The Virtual Vision System

The performance of the camera network is ultimately tied to the capabilities of the low-level machine vision routines responsible for gathering the sensory data. Consequently, when working with camera networks, it is important to make accurate assumptions about the capabilities and performance of the low-level visual sensing processes. We ensure that our assumptions about the low-level visual sensing are correct by implementing a pedestrian tracking system that operates solely upon the synthetic video captured through the virtual cameras. In this chapter, we describe the visual processing routines that support the low-level sensing requirements of the camera networks presented in the next two chapters.

Each camera has a suite of visual analysis routines for pedestrian recognition and tracking, which we dub "Local Vision Routines" (LVRs). LVRs are computer vision algorithms that operate upon the video generated by virtual cameras to identify, locate, and track the pedestrians present in the scene. LVRs faithfully mimic the performance of a state-of-the-art pedestrian recognition and tracking system and exhibit errors usually associated with a pedestrian tracking system operating upon real footage, including the loss of track due to occlusions, sudden change in lighting, and bad segmentation (Figure 4.1). Tracking sometimes locks onto the wrong pedestrian, especially if the scene contains multiple pedestrians with similar visual appearance, i.e., wearing similar clothes. Additionally, the virtual world affords us the benefit of

(a) (b) (c)

Figure 4.1: Tracking Pedestrians 1 and 3. Pedestrian 3 is tracked successfully; however, (a) track is lost of Pedestrian 1 who blends in with the background. (b) The tracking routine loses Pedestrian 3 when she is occluded by Pedestrian 2, but it regains track of Pedestrian 3 when Pedestrian 2 moves out of the way (c).

fine tuning the performance of the recognition and tracking module by taking into consideration the ground truth data readily available from the virtual world. Our prototype surveillance systems, which we describe in Chapters 5 and 6, is designed to operate robustly in the presence of occasional low-level failures.

The remainder of the chapter is organized as follows: We begin by describing synthetic video capture in the next section. Section 4.2 describes pedestrian segmentation and tracking. We present a strategy for estimating the 3D location of the pedestrians using calibrated, static cameras in Section 4.3. We describe the active PTZ camera controller in the following section. Section 4.5 introduces a novel strategy for learning the mapping between the 3D locations and the internal pan-tilt settings of an active PTZ camera. We summarize the chapter in Section 4.6.

4.1 Synthetic Video Feed

Virtual cameras use the OpenGL library and the standard graphics pipeline [Foley et al. 1990] to render the synthetic video feed. Our imaging model emulates imperfect camera color response, compression artifacts, detector and data drop-out noise, and interlaced scanning ef-

fects; however, it does not yet account for such imaging artifacts as depth-of-field, imaging vignetting, and chromatic aberration. Likewise, the rendering engine does not support pedestrian shadows and specular highlights. More sophisticated rendering schemes would address these limitations. Noise is introduced during a post-rendering phase. The amount of noise introduced into the process determines the quality of the input to the visual analysis routines and affects the performance of the pedestrian segmentation and tracking module.

4.1.1 Camera Color Response

We model the variation in color response across cameras by manipulating the Hue, Saturation, and Value channels of the rendered image. Here, Hue refers to the color (e.g., red, blue, or yellow), Saturation refers to the vibrancy of that color (the amount of grayness present in the color), and Value refers to the color brightness. Similarly, we can adjust the tints, tones, and shades of an image by adding the desired amounts of blacks, whites, and grays, respectively [Birren 1976]. Our visual analysis routines rely on color-based appearance models to track pedestrians, ergo camera handovers are sensitive to the variations in the color response of different cameras.[1]

4.1.2 Compression Artifacts

Bandwidth is at a premium in sensor networks, in general, and in camera networks, in particular. In many instances, images captured by camera nodes are transmitted to a central location for analysis, storage, and monitoring purposes. Routinely camera nodes exchange information among themselves during camera handover, camera coordination, and multi-camera sensing operations. The typical data flowing in a camera network is the image/video data, which places much higher demands on a network infrastructure than, say, alpha-numeric or voice data. Consequently, in order to keep the bandwidth requirements within acceptable limits, camera nodes

[1] One solution might be to color-calibrate the camera network during an initial learning phase [Porikli 2003].

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

Figure 4.2: Image compression artifacts. (a) Uncompressed image. The box marks the region enlarged in (b). (c) Compressed image (JPEG). The box marks the region enlarged in (d). The 1000×1000 image shown in (a) is roughly 240Kb; after compression it is reduced to approximately 24Kb (c).

(a) (b)

Figure 4.3: Simulating noise in synthetic video: (a) detector noise and (b) data drop-out noise.

compress the captured images and video before sending them off to other camera nodes or to the monitoring station. JPEG[2] [Wallace 1991] is a commonly used standard for image compression and many hardware network cameras available in the market today employ JPEG compression to optimize bandwidth resources. JPEG is a lossy compression standard that achieves compression by discarding perceptually insignificant information, but introduces undesirable artifacts—blockiness, color distortion, Gibbs effects, and blurring—during the process. These artifacts are more noticeable at higher compression levels.

Compression artifacts together with the low-resolution of the captured images/video pose a challenge for visual analysis routines. Compression artifacts, therefore, are relevant to camera network research. We introduce compression effects into the synthetic video by passing it through a JPEG compressor/decompressor stage before passing it onto the pedestrian recognition and tracking module. Figure 4.2 shows compressed and uncompressed versions of a 1000×1000 image. Notice the compression artifacts around the region boundaries in Figure 4.2(d). The compressed version is roughly 10 times smaller than the uncompressed version.

4.1.3 Detector Noise

We model the ubiquitous detector noise as a data-independent, additive process where the noise has a zero-mean Gaussian distribution. Each pixel in the noisy image (Figure 4.3(a)) is the sum of the true pixel value and a random value drawn from a zero-mean Gaussian distribution. The standard deviation of the Gaussian distribution controls the amount of noise introduced into the image.

4.1.4 Data Drop-Out Noise

Data drop-out noise is caused by errors during the data transmission within the imaging device. The corrupted pixels are either set to the maximum value (snow) or have their bits flipped.

[2]JPEG stands for Joint Photographic Experts Group (`http://www.jpeg.org`—Last accessed on 25 January 2007).

(a) (b)

Figure 4.4: Video interlacing effects: (a) shows a deinterlaced video frame computed by weaving two fields and (b) shows a close-up view of a pedestrian in (a).

Sometimes pixels are alternatively set to the maximum value or zero (salt and pepper noise). The amount of noise determines the percentage of the corrupted pixels and the unaffected pixels remain unchanged (Figure 4.3b).

4.1.5 Interlaced Video

Interlaced scanning was the *de facto* standard for video display until the 1970s and it is still used by all analogue TV broadcast systems, namely PAL, SECAM, and NTSC.[3] It was developed to reduce video flicker in Cathode Ray Tube (CRT) video terminals by improving temporal resolution at the expense of spatial resolution. Interlaced scanning records a frame in two phases: every odd row is scanned during the first phase and the remaining, even rows are scanned during the next phase. The final image consists of both, odd (Field 1) and even (Field 2) rows. Interlaced scanning consumes less bandwidth than the alternative—progressive scan-

[3]NTSC stands for National Television Standards Committee, PAL stands for Phase Alternating Line, and SECAM stands for Sequential Color Memory.

ning, which records the entire image in one step; therefore, a large number of CCTV cameras available today use interlacing to minimize bandwidth requirements.

Interlaced video (a sequence of fields) is converted into non-interlaced video (a sequence of frames) through a process called *deinterlacing*. Moving objects might appear at different locations in even and odd fields due to the time elapsed between their respective captures, ergo the corresponding pixels in the two fields do not always line up leading to jagged edges or a saw-tooth appearance in the deinterlaced frame.

We model interlaced video by rendering twice as many frames as desired and keeping only even and odd rows of alternating frames. Figure 4.4 shows a 640×480 deinterlaced frame. The frame was generated by weaving two fields that were rendered $1/60$ seconds apart. Pedestrians that are moving across the image plane appear blurred around the edges. The amount of blur is proportional to the speed of the pedestrian. Interlacing affects also appear during panning and zooming operations in active PTZ cameras. Deinterlacing artifacts can be lessened by employing more sophisticated algorithms [De Haan and Bellers 1998], yet they cannot be entirely removed.

4.2 Visual Analysis for Pedestrian Tracking

We employ appearance-based models to track pedestrians. Pedestrians are segmented to construct unique and robust color-based signatures, which are then matched across the subsequent frames. In the case of passive cameras, pedestrian segmentation is carried out automatically through background subtraction, whereas for active cameras, where a background model is unavailable, the user provides the initial segmentation, say by initializing a bounding rectangle around the pedestrian of interest. Our background model represents each pixel by its color distribution in HSV space. We store color distributions as histograms.[4] Passive cameras learn their background models by observing the scene without any assistance from the operator and

[4]The histogram is not the best non-parametric density estimate [Scott 1992]. However, it suffices for our purpose.

without using any 3D information.

Segmenting and tracking multiple humans is a challenging problem, one that has drawn considerable attention from the research community over the last two decades. Object tracking and segmentation strategies developed so far rely on one or more of the following visual cues, motion, shape, and appearance (texture/color).[5] Color-based appearance signatures, in particular, have found widespread use in object tracking and segmentation applications [Terzopoulos and Rabie 1997; Fieguth and Terzopoulos 1997; Wren et al. 1997; Comaniciu et al. 2000; Sigal et al. 2004]. More recently, many pedestrian surveillance systems have employed color-based signatures for tracking purposes [Javed et al. 2003; Siebel 2003; Seitner and Hanbury 2006]. Swain and Ballard [1991] showed that object color distribution is a powerful cue for object recognition even in the absence of geometric information. Color-based signatures are efficient, robust to partial occlusions, and invariant to in-image rotations and scale variations. Unfortunately, color-based signatures are sensitive to illumination changes; however, this shortcoming can be mitigated by operating in HSV space instead of RGB space.[6]

4.2.1 Pedestrian Segmentation

Many pedestrian tracking applications exploit the fact that pedestrian segmentation can enormously simplify the subsequent recognition and tracking tasks [Haritaoglu et al. 1998; Oliver et al. 2000; Orwell et al. 2000; Seitner and Hanbury 2006]. Furthermore, pedestrian segmentation can aid in constructing signatures, which can later be used for pedestrian recognition and tracking. Background subtraction is a widely used approach for detecting moving objects against *mostly* static backgrounds, as is the case for our passive cameras. The intuition is that moving objects can be detected by comparing the current frame to a reference frame, which is often called the "background model" [Piccardi 2004].

[5] [Gavrila 1999] provides a slightly outdated, though still relevant, survey of pedestrian tracking algorithms and applications.

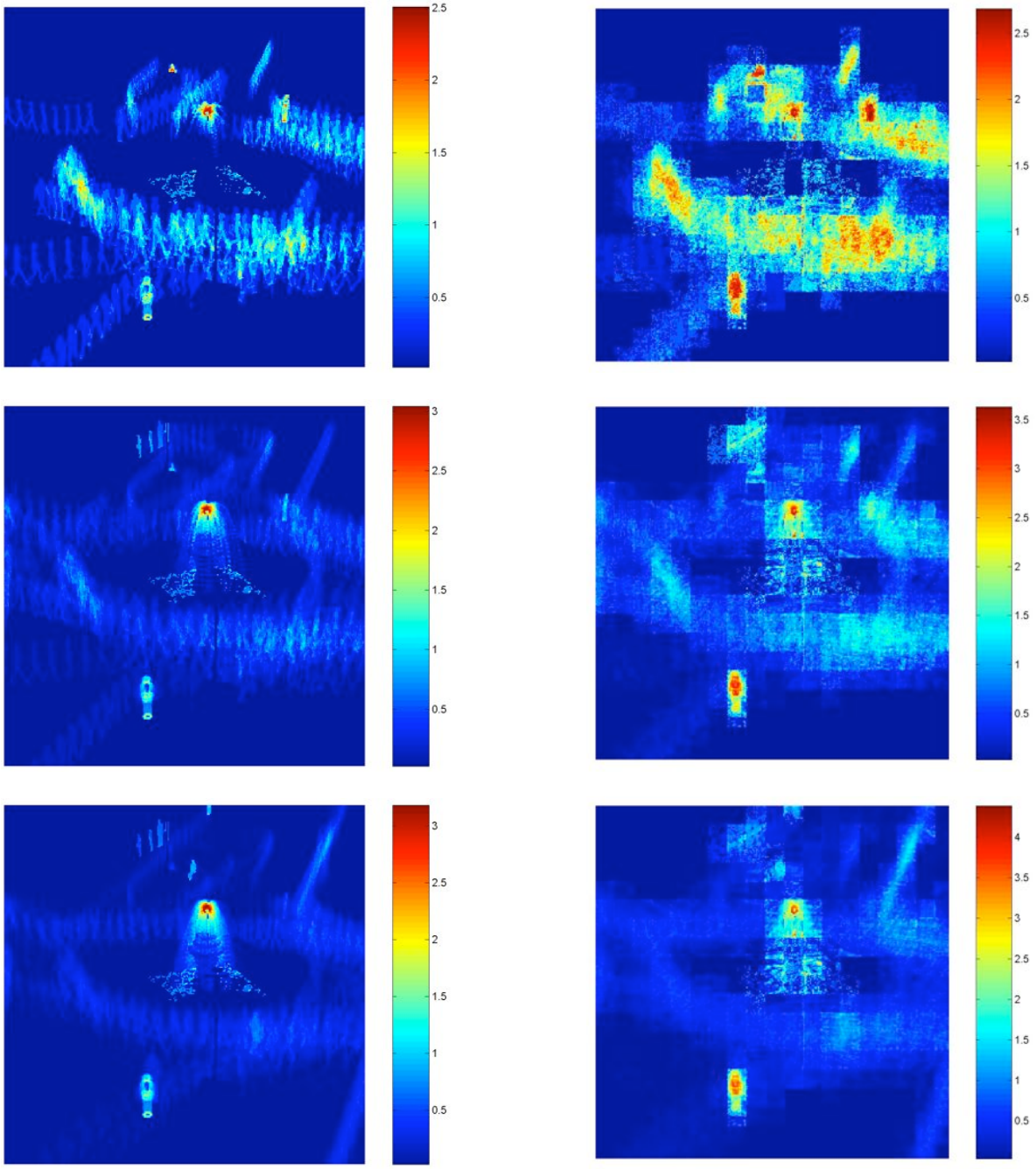
[6] We point the reader to [Lee et al. 2001], which constructs color-based object signatures that are robust to irregular illumination changes.

Many techniques for background subtraction have been explored, all of which estimate the background model from a sequence of frames. These can be broadly divided into two classes: 1) those that assume that the color probability density function (PDF) for a pixel is unimodal and 2) those that allow the pixel color PDF to be multi-modal. Frame differencing, running averages [Toyoma et al. 1999], temporal median filtering [Lo and Velastin 2001; Cucchiara et al. 2003], and running Gaussian averaging [Koller et al. 1994; Wren et al. 1997] schemes belong to the first class. These perform well in situations where background objects are either static or change relatively slowly, so as to allow the background model to adapt to the current value of the background object. A typical scene where the unimodal assumption fails, for example, would be an indoor scene illuminated by a flickering bulb. Methods that allow multi-modal PDFs for each pixel—such as mixture of Gaussians [Stauffer and Grimson 1999], kernel density estimation [Elgammal et al. 2000], and Eigenbackgrounds [Comaniciu 2003]—can handle a wider variety of scenes and, in general, outperform methods belonging to the first class.

Background Modeling for Pedestrian Segmentation

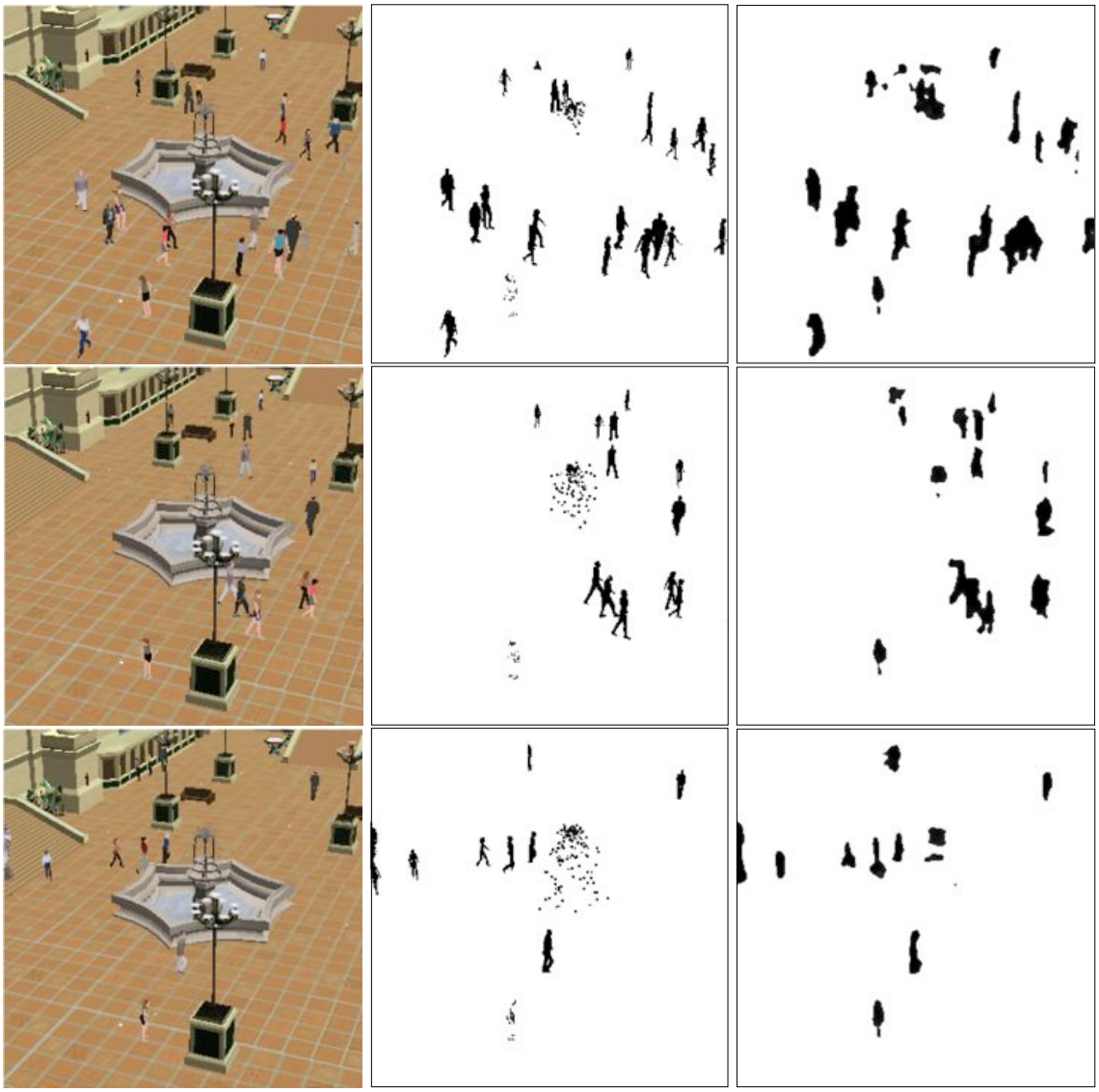
We approximate the PDF of each pixel as a color histogram in HSV color space. HSV space separates the chromaticity (H and S) from the effects of intensity (V); therefore, by sampling the intensity axis V more coarsely than the H and S axes, we can improve the robustness of the background model to lighting changes in the acquired scene. The main advantages of our approach are its simplicity and ease of implementation. Moreover, its performance is adequate for our purposes.

The color histogram (i.e., the PDF) for each pixel is estimated by sampling the pixel values over a sequence of frames. The intuition is that the stationary (background) objects are responsible for the dominant color(s) of the pixel; whereas, moving (foreground) objects are transitory and make smaller contributions to the histogram associated with the pixel. Consequently, the dominant color(s) of the pixel gain more strength in the associated color histogram



Without compression With compression

Figure 4.5: Background modeling. The entropy of each pixel after observing 15 (Row 1), 50 (Row 2), and 150 (Row 3) frames (the scene is similar to that shown in Fig. 4.6). The left column corresponds to the background model that is learned by observing uncompressed frames. In the right column, note the blocking artifacts when the background model is learned by observing compressed frames. Regions of high entropy (Red) indicate higher statistical randomness for the corresponding pixels, which in turn leads to poor foreground detection performance.



Source frames Without compression With compression

Figure 4.6: Pedestrian detection through background subtraction. Rows 1, 2, and 3 represent frame # 10, 20, and 40, respectively. Uncompressed source frames are shown in the first column. Column 2 shows segmented foreground objects for uncompressed frames using a background model that is learned by observing roughly 200 uncompressed frames. Column 3 shows the segmented foreground objects (after convolution with a Gaussian kernel to suppress noise) for compressed source frames. The background model used in the last column is learned by observing approximately 200 compressed frames. In almost all cases, the water spray is incorrectly labeled as a foreground object (false positive).

as more frames become available. Our observations confirm this intuition.

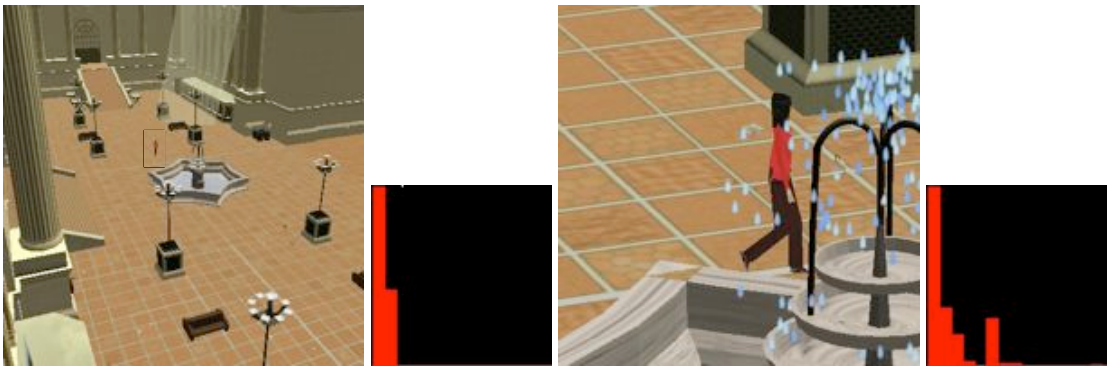
The entropy value of a pixel is a good indicator of the discriminative power of the learned model (Figure 4.5). A high entropy value for a pixel suggests that the associated PDF is "spread out"; i.e., according to the learned model, the pixel in question takes on many values with, say, *almost* equal probability. Such a model does a poor job of classifying a pixel as either a foreground or background pixel.[7]

Background subtraction is carried out by comparing the current value of a pixel against the color histogram associated with that pixel (Figure 4.6). The probability that a pixel belongs to the background is computed by backprojecting the associated normalized color histogram. If the result of the histogram backprojection is above a certain threshold, the pixel belongs to the background; otherwise, the pixel is labeled as a foreground pixel.

Background Model Maintenance

The background model must be able to adapt to a changing background. Given a new sample, there are three mechanisms to update the background model: 1) *blind update*, 2) *selective update*, and 3) *blind+selective update*. In the first case, each new pixel is added to the background model. Each target becomes part of the background, inevitably leading to false negatives. We can counter this effect by increasing the time window over which the samples are taken. Increasing the time window leads to more false positives as the adaptation rate of the model is slow and rare events are not well modeled. In the second case, only those new samples are added to the current background model that are classified as background pixels (through the current background model). Consequently, incorrect detection decisions will result in incorrect detection later, leading to a deadlock situation [Karmann and von Brandt 1990]. The third scheme for updating the background model maintains two models: the short-term model generally uses selective update and only consists of the N most recent samples taken over a

[7]Entropy values associated with a pixel can be used to select an appropriate threshold value for deciding whether or not the pixel belongs to the background [Otsu 1979; Pun 1980].



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

Figure 4.7: Color (Hue) image pixel histograms of a tracked object change drastically with zooming operations. To address this issue, a list of histograms is maintained for different zoom settings. (c) The histogram of the (boxed) subject (a) when the FOV of the camera is set to 45 degrees. (d) The histogram of the same subject when the camera has zoomed in on the subject.

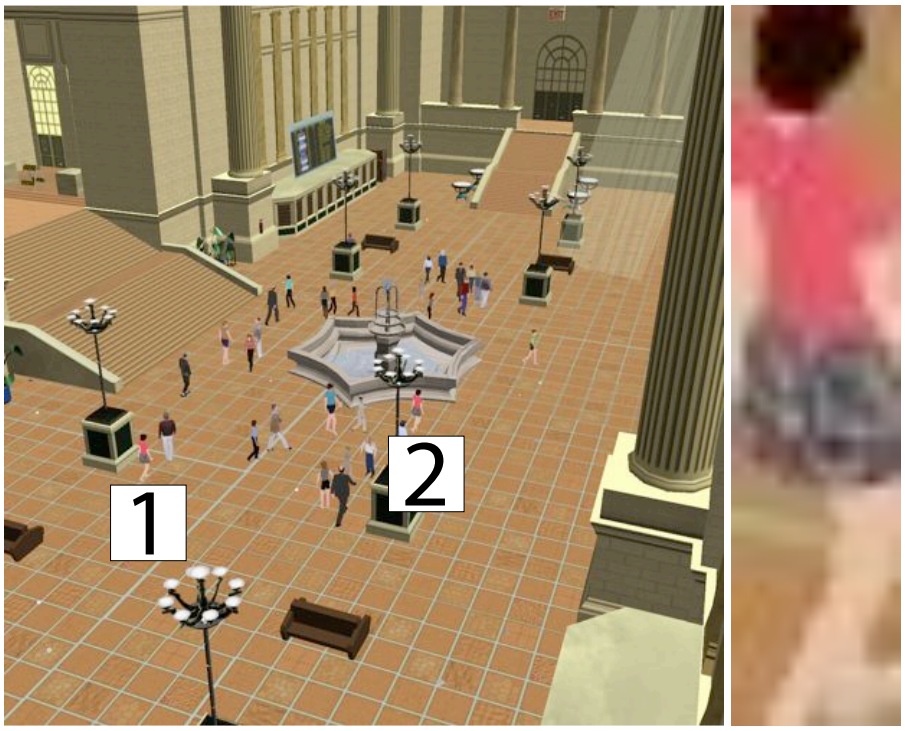
small time window, and the long-term model is updated via blind update and consists of N samples taken over a much longer duration. The intersection of the two detection results will eliminate the false positives from both short-term and long-term models. Unfortunately, it also suppresses those true positives in the first model result that are false negatives in the second model.

Visual surveillance applications are much more sensitive to false negatives than to false positives. For this reason, and for the sake of simplicity, we have selected the blind update rule for background model maintenance.

4.2.2 Pedestrian Tracking

We use appearance-based pedestrian signatures that encode pedestrian color distributions as 3D histograms in HSV colorspace (Figure 4.8). We have empirically selected 32 bins along the first two (Hue-Saturation) dimensions and 16 bins along the last (Value) dimension. These pedestrian signatures are matched across successive frames to track pedestrians.

For PTZ cameras, zooming can drastically change the appearance of a pedestrian (Figure 4.7), thereby confounding conventional appearance-based schemes, such as color his-



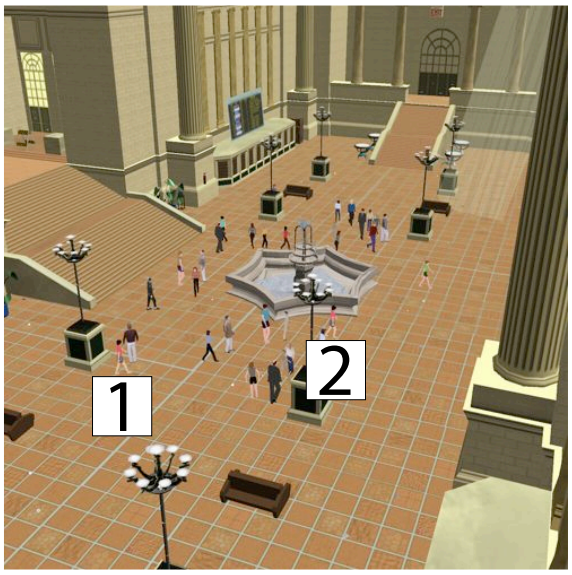
(a) Cam. 2, Frame 1 (b)

Figure 4.8: The user selects the pedestrian indicated by label 1 in image (a) to construct the pedestrian signature. (b) The *user selection*. Notice that the scene also contains another pedestrian (indicated by the label 2) that has a similar color distribution as the selected pedestrian. Pedestrian 1's signature is constructed without first segmenting out the pedestrian through background segmentation, so the constructed signature contains contributions from the background.

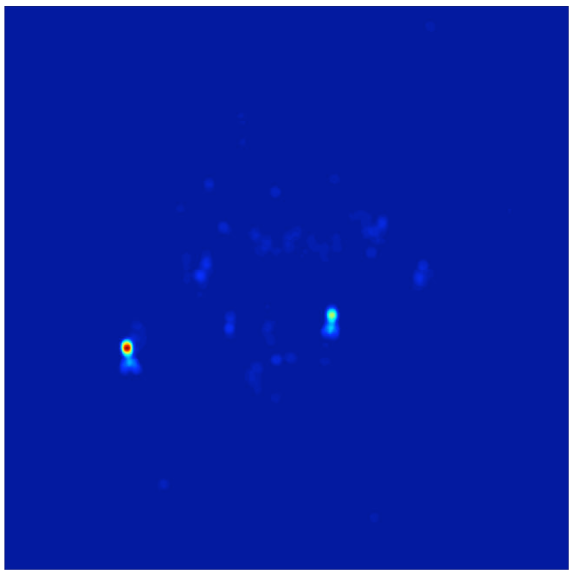
togram signatures. We tackle this problem by maintaining HSV color histograms for several camera zoom settings for each pedestrian. Thus, a distinctive characteristic of our pedestrian tracking routine is its ability to operate over a range of camera zoom settings.

Pedestrian Tracking without Segmentation

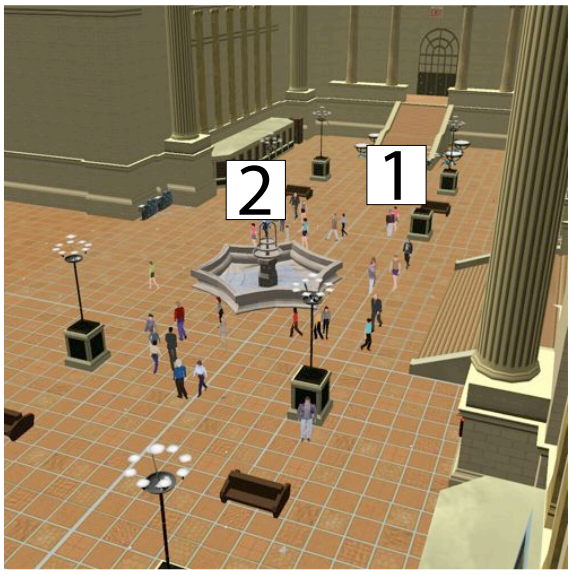
Pedestrian segmentation is difficult for active PTZ cameras due to the lack of a background model. In the absence of pedestrian segmentation, we match pedestrian signatures across frames through *color indexing*. Color indexing, proposed by Swain and Ballard [1991], efficiently identifies objects present in an image using object color distributions in the presence of occlusions and scale and viewpoint changes. It was later adapted by Terzopoulos and Rabie



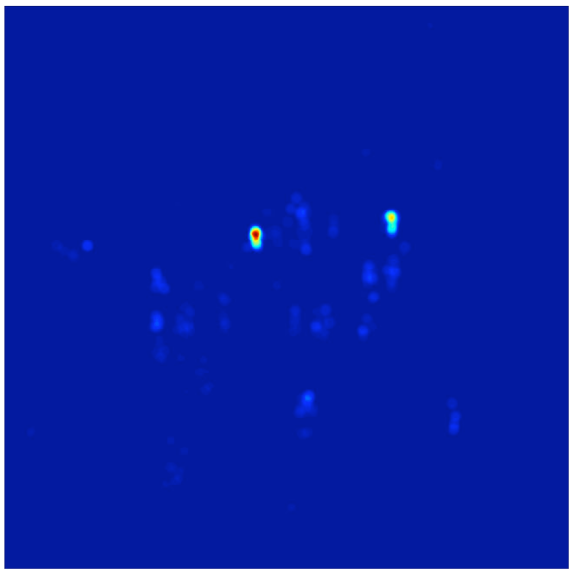
(a) Cam. 2, Frame 2



(b) Correct pedestrian identification



(c) Cam. 4, Frame 2



(d) Incorrect pedestrian identification

Figure 4.9: Pedestrian signature in Figure 4.8 is matched in two different snapshots of the scene captured synchronously for pedestrian tracking purpose. The image (a) is captured by the same camera used to construct the pedestrian signature in Figure 4.8; where as, the image in (c) is captured by a camera located on the opposite side of the Waiting Room. Histogram intersection/backprojection based pedestrian tracking successfuly finds Pedestrian 1 in image (a) inspite of the presence of Pedestrian 2 with similar appearance (b). Pedestrian tracking, however, fails to identify the "correct" pedestrian (i.e., Pedestrian 1) in image (c) and instead locks on Pedestrian 2 (d). Pedestrian tracking does not require the pedestrians segmentation. In (b) and (d) Red indicates a higher match value.

$$I = \boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}$$

$$I * \boxed{1} = \boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}$$

$$I * \boxed{1}\boxed{1}\boxed{1} = \boxed{0}\boxed{1}\boxed{2}\boxed{3}\boxed{2}\boxed{1}\boxed{0}$$

$$I * \boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1} = \boxed{1}\boxed{2}\boxed{3}\boxed{3}\boxed{3}\boxed{2}\boxed{1}$$

Figure 4.10: Multi-scale target localization in histogram backprojected images: Convolving an idealized 7-pixel 1D backprojected image I with 1-tap, 3-tap, and 5-tap summing kernels. Image I is extended with 0 borders for convolution purposes.

[1997] for active vision in artificial animals. In color indexing, targets with similar color distributions are detected and localized through histogram backprojection, which finds the target in an image by emphasizing colors in the image that belong to the target being observed.

For target histogram $T(n)$ and image histogram $I(n)$, where n is the number of bins, define the ratio histogram $R(n)$ as

$$R^i = \frac{T^i}{I^i}, \tag{4.1}$$

$i = 1, \cdots, n$ and T^i, I^i, and R^i are the number of samples in the i^{th} bin of histograms $T(n)$, $I(n)$, and $R(n)$, respectively. $R(n)$ is backprojected into the image, which involves replacing the image pixel values by the values of $R(n)$ that they index. Mathematically,

$$B_{xy} = R^{\text{map}(\mathbf{c}_{xy})}, \tag{4.2}$$

where $\mathbf{c}_{xy}$ is the value of the pixel at location (x, y), the function map$(\mathbf{c})$ maps a 3 dimensional color value (HSV) to the appropriate histogram bin, and B_{xy} is the value of the backprojected image at location (x, y). The backprojected image is then convolved with the a circular disk of area equal to the expected area of the target in the image,

$$B^r = D^r * B, \tag{4.3}$$

where D^r is the disk of radius r. The peak in the convolved image gives the expected (x, y) location of the target in the image. We point the reader to [Swain and Ballard 1991] for a thorough description of this process.

The last step of the color indexing procedure assumes that the area of the target in the image is known *a priori*. Active PTZ cameras violate the above assumption, as the area covered by the target in the image can vary drastically depending upon the current zoom settings of the camera. We propose a novel scheme to localize targets in a histogram backprojected image when the size of the targets in the image is not known beforehand. Our scheme is based on the observation that when the size of the target is equal to the size of the localization kernel (i.e., the disk D^r), the filter response forms a peak at the "true" location of the target. On the other hand, the filter response forms a plateau centered at the "true" location of the target in the image for kernel sizes that are either too large or too small when compared to the size of the target in the image. Figure 4.10 illustrates this phenomenon.

Target Localization in Backprojected Images:

(Step 1) Compute $B^l = B * K^l$, where B is the backprojected image, K^l is the kernel of size l, and $l = 1, \cdots, n$.

(Step 2) Find $(x_*, y_*) = \arg\max_{(x,y)} \sum_{l=1}^{n} B^l$

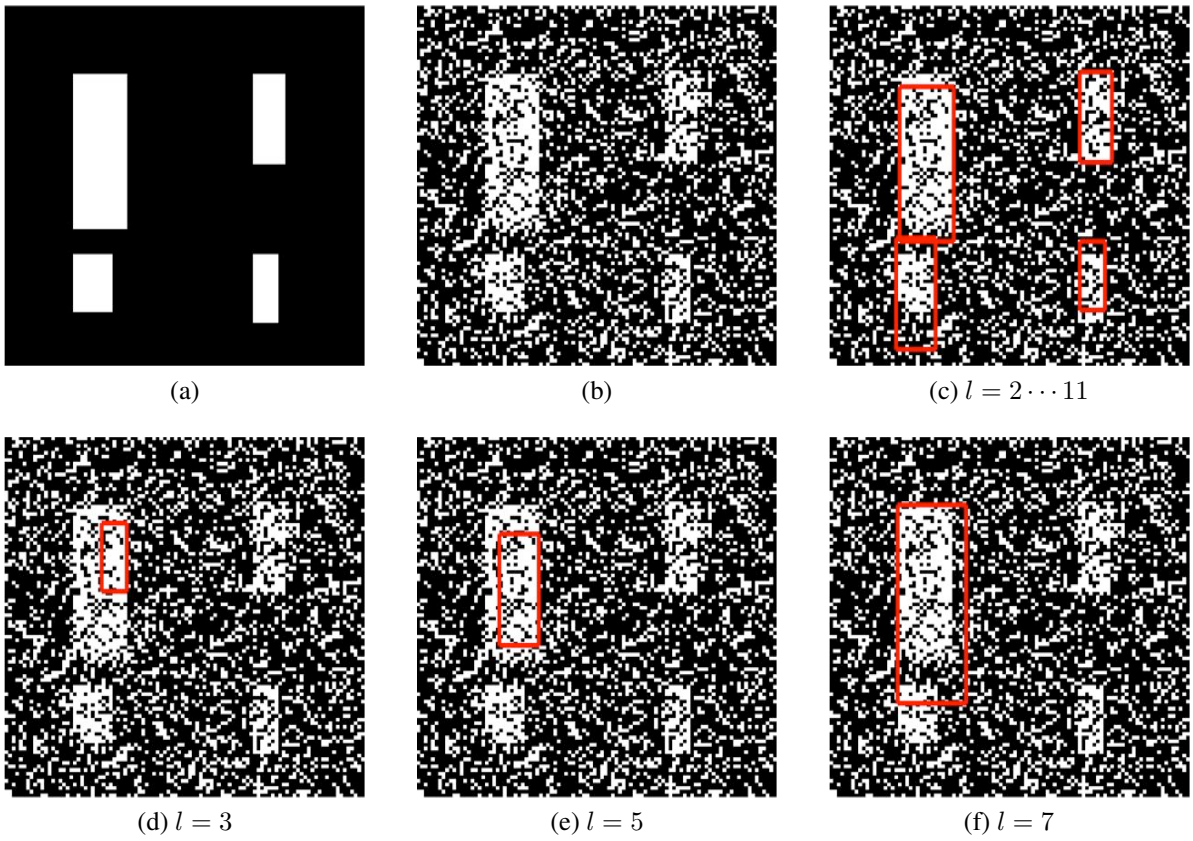
(Step 3) Find $(x_*^l, y_*^l) = \arg\max_{(x,y)\in\mathcal{K}} B^l$. $\mathcal{K}$ is the domain of K^l centered at (x_*, y_*).

(Step 4) Find $l^* = \arg\max_{l} \dfrac{\sum_{(x,y)\in\mathcal{K}}((B^l_{x_*^l y_*^l})^2 - (B^l_{xy})^2)}{|\mathcal{K}|}$

(Step 5) Construct color histogram $H(n)$ of the region of size l centered at (x_*, y_*) using the original image. If $\frac{\sum_i \min(T^i, H^i)}{\sum_i T^i} > \rho$, output region of size l at location (x_*, y_*); otherwise *quit*. ρ is a user-specified threshold.

(Step 6) Remove the region of size l centered at (x_*, y_*) from B^l and repeat steps $1, \cdots, 5$.

Multi-zoom Color Signatures: For *multi-zoom* color signatures $T = \{T_\theta | \theta = 1, \cdots, N_\theta\}$, where θ represents the field-of-view setting corresponding to the stored histogram T_θ and N_θ



(a) (b) (c) $l = 2 \cdots 11$

(d) $l = 3$ (e) $l = 5$ (f) $l = 7$

Figure 4.11: Target localization in backprojected images. We demonstrate our multi-scale localization procedure on synthetic data. We employ rectangular summing kernels $(6l + 1 \times 2l + 1)$. (a) An idealized 2D backprojected image that contains four targets with different sizes. (b) *Salt and Pepper* noise is added to the image to make the problem more challenging. (c) The proposed multi-scale localization procedure successfully identified all four regions. (d)-(f) show localization results using the default procedure for kernel sizes 3, 5, and 7, respectively.

is the total number of histograms stored across multiple zoom settings, we take a union of backprojection results for every stored histogram T_θ.

Pedestrian Tracking with Segmentation

When pedestrian segmentation is available, we can perform signature matching by means of histogram intersection. Unlike the color indexing scheme that operates upon the entire image, this scheme focuses on foreground regions as determined through background subtraction, which results in computational savings when the number of detected foreground objects (pedestrians) is low. We begin by constructing color histograms $P_r(n)$ for each foreground

region r. Next, we compute match score for each region r with respect to the target using the target histogram $T(n)$ through histogram intersection,

$$T \cap P_r = \frac{\sum_i \min\left(T^i, P_r^i\right)}{\sum_i T^i}. \tag{4.4}$$

$T \cap P_r$ takes on values between 0 and 1, where the larger the value of the histogram intersection, the more similar the histograms are deemed to be, The region r^* with the highest match score is selected as the location of the target in the current frame:

$$r^* = \arg\max_r T \cap P_r, \tag{4.5}$$

r indices over the number of foreground regions detected in the frame.

Multi-zoom Color Signatures: The match score of a histogram P_r against a stored *multi-zoom* color signature T is then

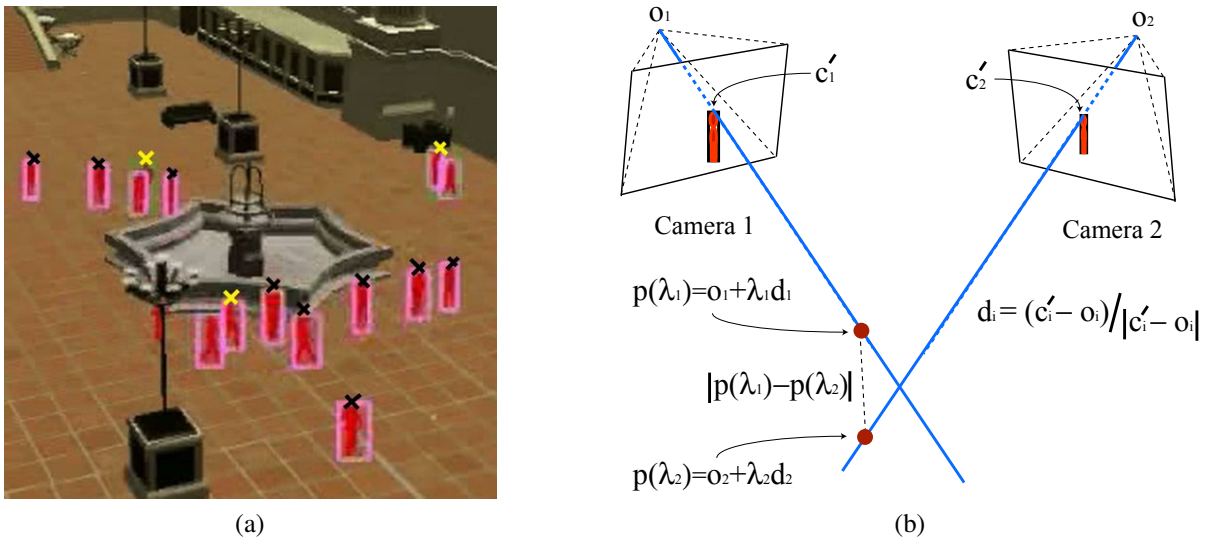
$$d(T, P_r) = \frac{1}{n_\theta} \sum_\theta \left(T_\theta \cap P_r\right), \tag{4.6}$$

where θ represents the field-of-view setting corresponding to the stored histogram T_θ, and n_θ is the total number of histograms stored across multiple zoom settings.

4.3 Pedestrian Localization in 3D

When a pedestrian is visible in two or more passive calibrated cameras, the 3D position of the pedestrian can be computed through triangulation. To estimate the 3D location of a pedestrian, we begin by identifying the set of cameras that are successfully tracking the pedestrian. The 3D positions of pedestrians that are tracked in only one camera cannot be estimated. For every camera correctly tracking the pedestrian, we construct a line passing through the center of projection of the camera and the top extremal point of the regions (black crosses in Figure 4.12(a)) occupied by the pedestrian. Under the assumption that the cameras are synchronized, the intersection of these lines yields an estimate of the 3D position of the pedestrian.[8]

[8]Camera synchronization is a challenging problem for real cameras, especially when the cameras are distributed over a large region. However, synchronizing the virtual cameras is straightforward: We pause the simula-



(a) (b)

Figure 4.12: (a) Pedestrian segmentation identifies regions occupied by one or more pedestrians. Black crosses indicate the top extremal point of the regions occupied by single pedestrians. Yellow crosses indicate the top extremal points for regions where the pedestrian segmentation failed to separate multiple pedestrians. (b) Lines passing through the center of projection and the top extremal points of the pedestrian regions for the same pedestrian in images captured by two cameras. Lines in 3D rarely intersect, so we solve the intersection problem within a minimization framework.

The above computation is carried out at each frame. The 3D trajectory of a pedestrian thus obtained is usually noisy due to poor segmentation and tracking. The 3D trajectory is smoothed on the fly using an *exponentially weighted moving averages* scheme

$$\mathbf{p}_{\mathrm{f}} = \gamma \mathbf{p}_{\mathrm{old}} + (1 - \gamma)\mathbf{p}_{\mathrm{c}}, \tag{4.7}$$

where $\mathbf{p}_{\mathrm{f}}$, $\mathbf{p}_{\mathrm{o}}$, and $\mathbf{p}_{\mathrm{c}}$ is the final, old, and current estimates of the 3D position of the pedestrian, respectively, and $0 \leq \gamma \leq 1$. The parameter γ is set to 1 when $\mathbf{p}_{\mathrm{c}} - \mathbf{p}_{\mathrm{o}} > \eta$ and the time difference $t_{\mathrm{c}} - t_{\mathrm{o}} < t_{\eta}$, capturing the intuition that the pedestrian cannot cover more than η units of distance in less than t_{η} units of time. Here, t_{c} is the current time and t_{o} is the timestamp of the last estimated position. The parameter γ is set to 0 when $t_{\mathrm{c}} - t_{\mathrm{o}} > t_{\mathrm{g}}$, a time threshold for which the last estimate is considered no longer relevant. Old values are updated when $\gamma \neq 0$ for subsequent computations: $\mathbf{p}_{\mathrm{o}} = \mathbf{p}_{\mathrm{f}}$ and $t_{\mathrm{c}} = t_{\mathrm{o}}$. Values for parameters, γ, η, t_{η}, and t_{g} are

tion clock and render the scene using every virtual camera. It is also straightforward, however, to make the virtual cameras drift out of sync.

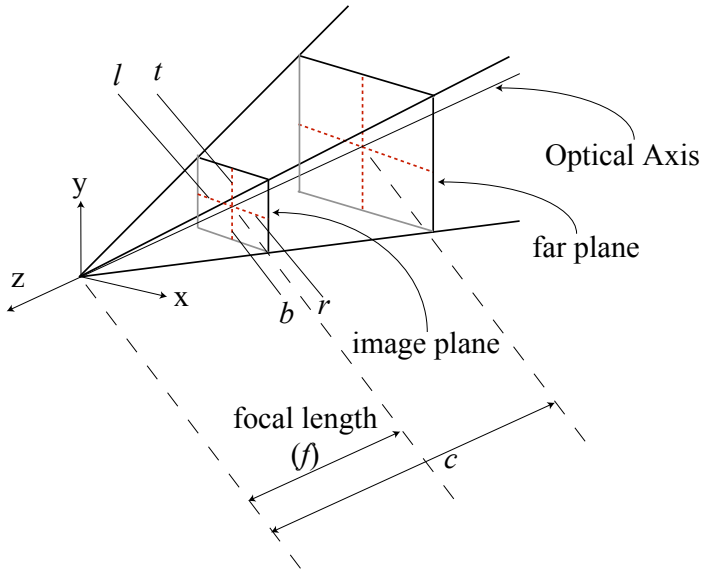


Figure 4.13: Anatomy of a virtual camera: l, r, t, and b are left, right, top, and bottom extents of the image plane, respectively, f is the distance between the center of projection and the image plane, also called the *focal length*, and c is the far plane bounding the view frustum. Any point behind the image plane or outside the view frustum is not rendered in the image.

selected empirically to be 0.6, 10, 330 ms, and 3000 ms, respectively. Next, we describe the triangulation procedure.

4.3.1 The Triangulation Procedure

Our current virtual cameras are ideal pinhole cameras with no radial/lens distortion, so the relationship between a 3D point $\mathbf{m} = (x, y, z)^{\mathrm{T}}$ and its image projection $\widetilde{\mathbf{m}} = (u, v)^{\mathrm{T}}$ is given by

$$\widetilde{\mathbf{m}}_{\mathrm{H}} = \frac{1}{s}\mathbf{A}\mathbf{M}_c\mathbf{m}_{\mathrm{H}}, \tag{4.8}$$

where s is an arbitrary scale factor and the subscript H denotes augmented vectors: $\mathbf{m}_{\mathrm{H}} = (x, y, z, 1)^{\mathrm{T}}$ and $\widetilde{\mathbf{m}}_{\mathrm{H}} = (u, v, \hat{z}, w)^{\mathrm{T}}$, where $\hat{z}, w \in \Re$. The 4×4 matrix $\mathbf{M}_c$ encodes the extrinsic parameters of the camera; i.e., the translation and rotation that relates the world coordinate system the to camera coordinate system, while the 4×4 matrix $\mathbf{A}$ is the *perspective projection matrix* that encodes the intrinsic parameters of the camera. For real cameras, $\mathbf{A}$ and $\mathbf{M_c}$ can be computed through a camera calibration process [Forsyth and Ponce 2002]. For virtual cameras,

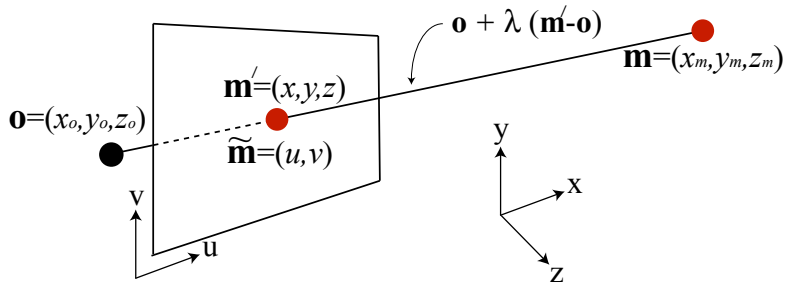


Figure 4.14: $\mathbf{m}$ lies on the line passing through the center of projection and $\mathbf{m}'$. $\widetilde{\mathbf{m}}$ is the projection of $\mathbf{m}$ on the image plane. $\mathbf{m}'$ is $\widehat{\mathbf{m}}$ expressed in the world coordinates.

on the other hand, the $\mathbf{A}$ and $\mathbf{M_c}$ are readily available through the rendering equation

$$\widetilde{\mathbf{m}}_{\mathrm{H}} = \frac{1}{s} \underbrace{\mathbf{M}_v \mathbf{M}_o \mathbf{M}_p}_{\mathbf{A}} \mathbf{M}_c \mathbf{m}_{\mathrm{H}}, \tag{4.9}$$

where

$$\mathbf{M}_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{f+c}{f} & -c \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix}, \tag{4.10}$$

$$\mathbf{M}_o = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{f-c} & -\frac{f+c}{f-c} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.11}$$

$$\mathbf{M}_v = \begin{bmatrix} \frac{u_x}{2} & 0 & 0 & -\frac{u_x-1}{2} \\ 0 & \frac{u_y}{2} & 0 & -\frac{u_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.12}$$

Parameters n, f, l, r, t, and b are defined in Figure 4.13. The quantities u_x and u_y represent the size of the image in pixels.

For a point $\mathbf{m}$ in world coordinates and its projection $\widetilde{\mathbf{m}}$ in the image, we can compute a point $\mathbf{m}'$ in world coordinates such that $\mathbf{m}$ approximately lies on the line passing through the

center of projection of the camera and $\mathbf{m}'$ (Figure 4.14). $\mathbf{m}'$ can be computed by re-arranging (4.9) as follows:

$$\mathbf{m}' = \mathbf{M}_c^{-1}\mathbf{M}_p^{-1}\mathbf{M}_o^{-1}\mathbf{M}_v^{-1}[u \;\; v \;\; -1 \;\; 1]^{\mathrm{T}} \tag{4.13}$$

To estimate the 3D location of a pedestrian, we employ the fact that a point $\mathbf{m}$ that projects to $\widetilde{\mathbf{m}}_i$ in the i^{th} camera lies at the intersection of lines $\mathbf{p}(\lambda_i) = \mathbf{o}_i + \lambda_i\mathbf{d}_i$, where $i \in [1, n]$, n (> 1) is the number of cameras in which the point $\mathbf{m}$ is visible, $\mathbf{o}_i$ is the center of projection of the i^{th} camera, $\mathbf{d}_i = \left(\mathbf{m}_i^{'} - \mathbf{o}_i\right)/|\mathbf{m}_i^{'} - \mathbf{o}_i|$, and $0 \leq \lambda_i \leq \infty$ (see Figure 4.12(b)). Here, $\mathbf{m}_i^{'}$ are computed from $\widetilde{\mathbf{m}}_i$ using (4.13).

The intersection of lines cannot be computed in closed-form as the probability of lines actually intersecting is a probability zero event due to imaging artifacts and numerical imprecisions. We instead cast the intersection estimation as an optimization problem. The point of intersection of n lines is then $(1/n)\sum_i \mathbf{p}(\lambda_i^*)$, where the λ_i^* are computed as

$$\arg\min_{\lambda} \sum_{\substack{i,j,i<j \\ i,j\in[1,n]}} |\mathbf{p}(\lambda_i) - \mathbf{p}(\lambda_j)|^2. \tag{4.14}$$

We noticed that the intersection process was highly sensitive to the errors in pedestrian segmentation and tracking. Errors during the segmentation and tracking stage get amplified during the intersection process, and sometimes even a small error in one of the cameras results in a large deviation from the "true" position of the pedestrian. We propose the following Random Sample Consensus (RANSAC) [Fischler and Bolles 1981] based strategy to resolve this issue:

RANSAC-Based Intersection of Lines: Assuming that a pedestrian is visible in n cameras, the 3D position of the pedestrian lies at the intersection of n lines, $\mathbf{p}(\lambda_i) = \mathbf{o}_i + \lambda_i\mathbf{d}_i$, $i \in [1, n]$.

(Step 1) Randomly pick $m < n$ lines and estimate their intersection point $\mathbf{p}_k$ using the technique described above.

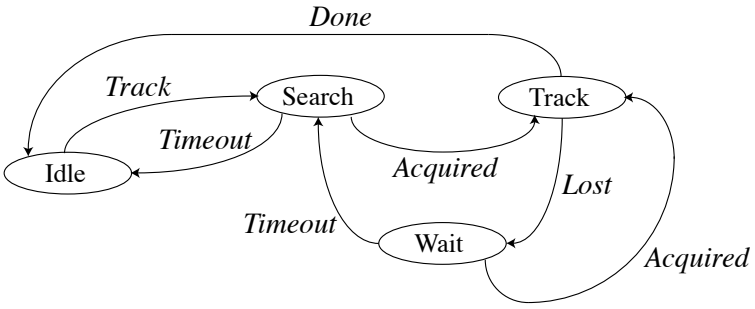


Figure 4.15: Camera behavioral controller.

(Step 2) Compute the perpendicular distance d_i of the point $\mathbf{p}_k$ from the i^{th} line as

$$d_i = \frac{|\mathbf{d}_i \times (\mathbf{o}_i + \mathbf{d}_i - \mathbf{p}_k)|}{|\mathbf{d}_i|} \tag{4.15}$$

(Step 3) Let n_c be the number of lines for whom d_i is below the user-specified threshold.

(Step 4) If n_c/n is greater than a user defined threshold, exit with *success*.

(Step 5) Repeat steps $1, \cdots, 4$ at most k times.

(Step 6) Exit with *failure*.

4.4 PTZ Active Camera Controller

We treat every PTZ active camera as a behavior-based autonomous agent. The overall behavior of the camera is determined by the pedestrian tracking module and the current task. The camera behavioral controller, which we model as an augmented finite state machine (Figure 4.15), enables an autonomous camera to achieve its high-level sensing goals as determined by the current task. Typical sensing goals might be, "look at the pedestrian i at location (x, y, z) for t seconds," or "track the pedestrian whose appearance signature is h." Our approach severs the ubiquitous master-slave relationship between the originator of the sensing goal and the camera

in the sensor network that will perform the sensing action [Zhou et al. 2003]. Communication requirements and scalability considerations aside, the master-slave relationship between multiple cameras is undesirable as it requires the camera network to be calibrated. Unfortunately, active PTZ cameras are notoriously difficult to calibrate; moreover, the calibration deteriorates over time and needs to be recomputed. Our camera network model does not require calibrated active cameras, so it is easier to change the topology of the network by adding/removing/modifying cameras.

When carrying out a new sensing request, the camera selects a suitable FOV setting and either chooses an appropriate gaze direction using the estimated 3D location of the pedestrian (Section 4.5), or performs an exploratory sweep when the pedestrian's 3D location is unavailable. Upon the successful identification of the pedestrian within the FOV, the camera uses fixation and zooming algorithms to follow the subject. The fixation and zooming routines are image driven and do not require any 3D information such as camera calibration or a global frame of reference. Furthermore, the 3D location of the pedestrian is not required by a PTZ camera for the purposes of fixation/zooming/tracking.

We discovered that traditional proportional derivative (PD) controllers generate unsteady control signals resulting in jittery camera motion. The noisy nature of tracking forces the PD controller to strive to minimize the error metric continually without ever succeeding, so the camera keeps servoing. Hence, we model the fixation and zooming routines as dual-state controllers. The states are used to activate/deactivate the PD controllers. In the *Act* state the PD controller tries to minimize the error signal; whereas, in the *Maintain* state the PD controller ignores the error signal altogether and does nothing.

The fixate routine brings the region of interest—e.g., the bounding box of a pedestrian—into the center of the image by tilting the camera about its local X and Y axes (Figure 4.16, Row 1). We provide the *fixate* routine in Figure 4.17. The zoom routine controls the FOV of the camera such that the region of interest occupies the desired percentage of the image. This is useful in situations where, for example, the operator desires a closer look at a suspicious

Figure 4.16: Row 1: A fixate sequence. Row 2: A zoom sequence. Row 3: Camera returns to its default settings upon losing the pedestrian; it is now ready for another task.

Require: Size of the video image: $w \times h$ {*Origin of the image coordinate space coincides with the left-bottom corner of the image*}
Require: Region of interest (ROI): left-bottom (l, b) and right-top (r, t) corners defined in the image space
Require: Camera's field of view (FOV) settings along the image's x and y axes: θ_x and θ_y, respectively
Require: Camera's pan and tilt settings (rotation about local y and x axis): α and β, respectively
  Center of the image, $\mathbf{c}_{\text{I}} = (1/2)(w, h)$
  Center of the region of interest, $\mathbf{c}_{\text{ROI}} = (1/2)(l + r, t + b)$
  Rectangle r_1 with corners $\mathbf{c}_{\text{I}} \mp \eta_1(w, h)$
  Rectangle r_2 with corners $\mathbf{c}_{\text{I}} \mp \eta_2(w, h)$
  if ROI is enclosed within r_1 **then**
    State = *Maintain* {**No change**}
  else if ROI is not enclosed within r_2 **then**
    State = *Turn*
    Error, $\mathbf{e} = \mathbf{c}_{\text{I}} - \mathbf{c}_{\text{ROI}}$
  end if
  if state is *Maintain* **then**
    $\alpha_{\text{new}} = \alpha$
    $\beta_{\text{new}} = \beta$
  else
    $\alpha_{\text{new}} = -g_p e_1 \theta_x / w - g_d \Delta e_1 / \Delta t$
    $\beta_{\text{new}} = g_p e_0 \theta_y / h + g_d \Delta e_0 / \Delta t$
  end if

Figure 4.17: The fixation routine. Here, $0 < \eta_1 < \eta_2 < 1$. We have empirically selected $\eta_1 = 0.025$ and $\eta_2 = 0.225$. Δt is the time-elapsed between two control cycles. g_p and g_d are empirically selected proportional and derivative gains.

pedestrian (Figure 4.16, Row 2). The details of the *zoom* routine are given in Figure 4.18.

Fixation and zooming routines operate independently of each other, each trying to achieve its respective goals. Consequently, an active PTZ camera can simultaneously zoom and fixate on the pedestrian being tracked. When a tracking failure is detected, the camera controller goes into a recovery mode where the fixation routine is deactivated and the camera begins to increase its FOV setting with the aim to keep the pedestrian within the field of view. Visual analysis is performed to reacquire the pedestrian using the stored pedestrian signature. If unsuccessful, the camera reports a failure and returns to its default state (Figure 4.16, Row 3).

4.5 Learning the Gaze Direction

Require: Size of the video image: $w \times h$ {*Origin of the image coordinate space coincides with the left-bottom corner of the image*}
Require: Region of interest (ROI) defined in the image space
Require: Camera's field of view (FOV) setting along the image's y axis (vertical FOV): θ
Require: Desired coverage factor $d_r \in [0, 1]$ {*e.g. ROI covers the entire image when* $d_r = 1$}
Compute the smallest rectangle that encloses ROI and whose aspect ratio is w/h: r
Error, $e = \text{area}(r)/(wh) - d_r$
if $|e| < \eta_1$ **then**
State = *Maintain*
else if $|e| > \eta_2$ **then**
State = *Zoom*
end if
if state is *Maintain* **then**
$\theta_{\text{new}} = \theta$ {**No change**}
else
if $e > 0$ **then**
$\theta_{\text{new}} = \theta + g_p \min(1, e) + g_d \Delta e/\Delta t$, where g_p and g_d are empirically selected proportional and derivative gains, respectively {**Zoom out**}
else
Rectangle r_{valid} with corners $(1/2)(w, h) \mp \delta(w, h)$
if r is not enclosed within r_{valid} **then**
$\theta_{\text{new}} = \theta$ {**No change**}
else
Let (v_x, v_y) be the corner of r that is farthest from the point $(1/2)(w, h)$
Let $s = \begin{cases} 2v_x/w & \text{if } v_x \geq v_y \\ 2v_y/h & \text{if } v_x < v_y \end{cases}$
Let $s' = 1 - s^2/(a + s^2)$
$\theta_{\text{new}} = \theta - (g_p \min(1, s'|e|) + g_d \Delta e/\Delta t)$ {**Zoom in.** *The change in FOV during a zoom-in phase depends upon the current FOV settings with smaller changes for small FOV settings to avoid zooming in too much, too quickly, which can result in tracking failures.*}
end if
end if
end if

Figure 4.18: Zooming Algorithm. Here, $0 < \eta_1 < \eta_2 < 1$ and $\delta \in [0, 1]$. Specifically, we have empirically selected $\eta_1 = 0.01$, $\eta_2 = 0.03$, $\delta = 0.45$, and $a = 0.4$.

Computing an appropriate gaze direction in order to bring the desired pedestrian within the FOV of a camera requires a map between 3D locations in the scene and the associated internal gaze-direction parameters (i.e., the pan-tilt settings) of the camera. This map is learned automatically by observing a pedestrian directed to move around in the scene during an initial learning phase. While the active PTZ cameras are tracking and following the pedestrian, the 3D location of the pedestrian is estimated continuously through triangulation using the calibrated, passive FOV cameras. A lookup table is computed for each PTZ camera that associates the estimated 3D scene location of the target pedestrian with the corresponding gaze angles of the camera. Specifically, for PTZ camera i, this yields tuples of the form $(x^j, y^j, z^j, \alpha^j, \beta^j)$, where j indexes over the scene locations (x, y, z) and corresponding camera pan and tilt angles (α, β). The lookup table constitutes the training dataset used for learning a continuous map $\mathcal{M} : \Re^3 \rightarrow \Re^2$ between the locations and associated angles. We tried two approaches for learning $\mathcal{M}$, which we explain in the next two sections.

4.5.1 Nearest Neighbor Approximation to $\mathcal{M}$

Given any 3D input point $\mathbf{p} = (x, y, z)$, the system estimates the values for α_i and β_i of any camera i that can observe $\mathbf{p}$ as follows: Let S_k be the set of k nearest neighbors to $\mathbf{p}$ in $\{\mathbf{p}^j = (x^j, y^j, z^j)\}$, where proximity is computed using the L_2 norm $\|\mathbf{p} - \mathbf{p}^j\|$. Then

$$\alpha_i = \frac{1}{k} \sum_{j:\, \mathbf{p}^j \in S_k} \alpha^j; \tag{4.16}$$

$$\beta_i = \frac{1}{k} \sum_{j:\, \mathbf{p}^j \in S_k} \beta^j. \tag{4.17}$$

This provides only a coarse map between the 3D scene points and the camera pan-tilt angles, but the map is accurate enough in practice to bring the pedestrian within the field of view of the camera. We noticed, however, that the learnt map deteriorates rapidly for points far from those in the lookup table (Figure 4.20, Rows 1 and 3). As the PTZ cameras end up performing a visual search operation to locate the pedestrian, poor mapping results in longer lead times (the time it takes for an active PTZ camera to lock onto the desired pedestrian).

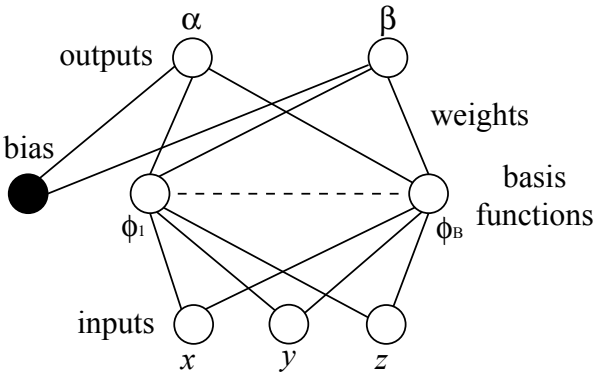


Figure 4.19: The architecture of a radial basis function network. Each basis function acts as an hidden unit. The lines from basis function ϕ_j to the inputs represent the corresponding elements of μ_j. The weights $w_{j\alpha}$ are represented by lines from the basis functions to the output α, and $w_{j\beta}$ are represented by lines from the basis functions to the output β. The output of the bias is fixed at 1.

4.5.2 Radial Basis Function Approximation to $\mathcal{M}$

We addressed the above issue to some extent by modeling the mapping $\mathcal{M}$ as a radial basis function (RBF) network, since such networks are known to have excellent interpolation properties and are relatively easy to train. Referring to Figure 4.19, the network is given by

$$\alpha(\mathbf{p}) = \sum_{j=1}^{B} w_{j\alpha}\phi_j(\mathbf{p}) + w_{\alpha 0}; \tag{4.18}$$

$$\beta(\mathbf{p}) = \sum_{j=1}^{B} w_{j\beta}\phi_j(\mathbf{p}) + w_{\beta 0}, \tag{4.19}$$

where B is the number of basis functions, the weights w are defined in the figure, and

$$\phi_j(\mathbf{p}) = \exp\left(-\frac{|\mathbf{p} - \boldsymbol{\mu}_j|^2}{2\sigma_j^2}\right), \tag{4.20}$$

where $\boldsymbol{\mu}_j$ and σ_j determine the center and standard deviation of the basis function ϕ_j, respectively. The RBF networks are trained by choosing the number of hidden units (basis functions), choosing their centers and standard deviations, and training the output layer weights w. We train the RBF using an adaptive approach that adds nodes and estimates their means and variances successively, until the approximation error falls below a user-specified threshold. When dealing with large training data sets, the resultant RBF network can potentially have a large

number of hidden nodes. We did not encounter this issue; however, one way to deal with it is to sub-sample the training set.

4.5.3 Results and Comparison

Figure 4.20 shows the results obtained for the k-nearest neighbor and the RBF network approaches. Each of the plots shows a plan view of a 3D space, covering the XY-plane ($Z = 0$). The red dots denote 3D points used in learning the map between the 3D world and the pan-tilt settings of the active, PTZ camera. The optical center of this camera is indicated by the green dot at the upper right of each plot. The red points indicate the triangulation-estimated locations of the visually tracked pedestrian, where the active PTZ camera fixating on the pedestrian has stored an associated pan-tilt setting. Note that the training uses no information about the camera's 3D location. Rows 1 and 3 in Figure 4.20 show results obtained for k-nearest neighbor; where as, Rows 2 and 4 show results obtained for the RBF network approach.

Each plot shows the squared error between the true and estimated quantities, where brightness is proportional to the magnitude of the error. To generate the plots, we regularly sampled points on the XY-plane and used the position of the camera to compute the true pan-tilt settings for each of the sampled points (true α and β), which constitutes the ground truth. Next, we used the learned map to estimate the camera's pan-tilt settings for every sampled point (estimated α and β).

The first two columns (from left) in Figure 4.20 plot the squared error in degrees between the true and estimated pan-tilt settings. The last column in Figure 4.20 plots the Euclidean distance between the true 3D location and the point where the XY-plane intersects the optical axis of the PTZ camera when the camera's pan-tilt angles are set to the estimated values. As expected, we observed that the estimate exhibits smaller error in the vicinity of the training data used to learn the map and that it improves over a larger area when the PTZ camera has had the chance to observe longer, more varied, pedestrian tracks like the one shown in the bottom two rows of Figure 4.20. Notice also that the RBF approach outperforms the k-nearest neighbor

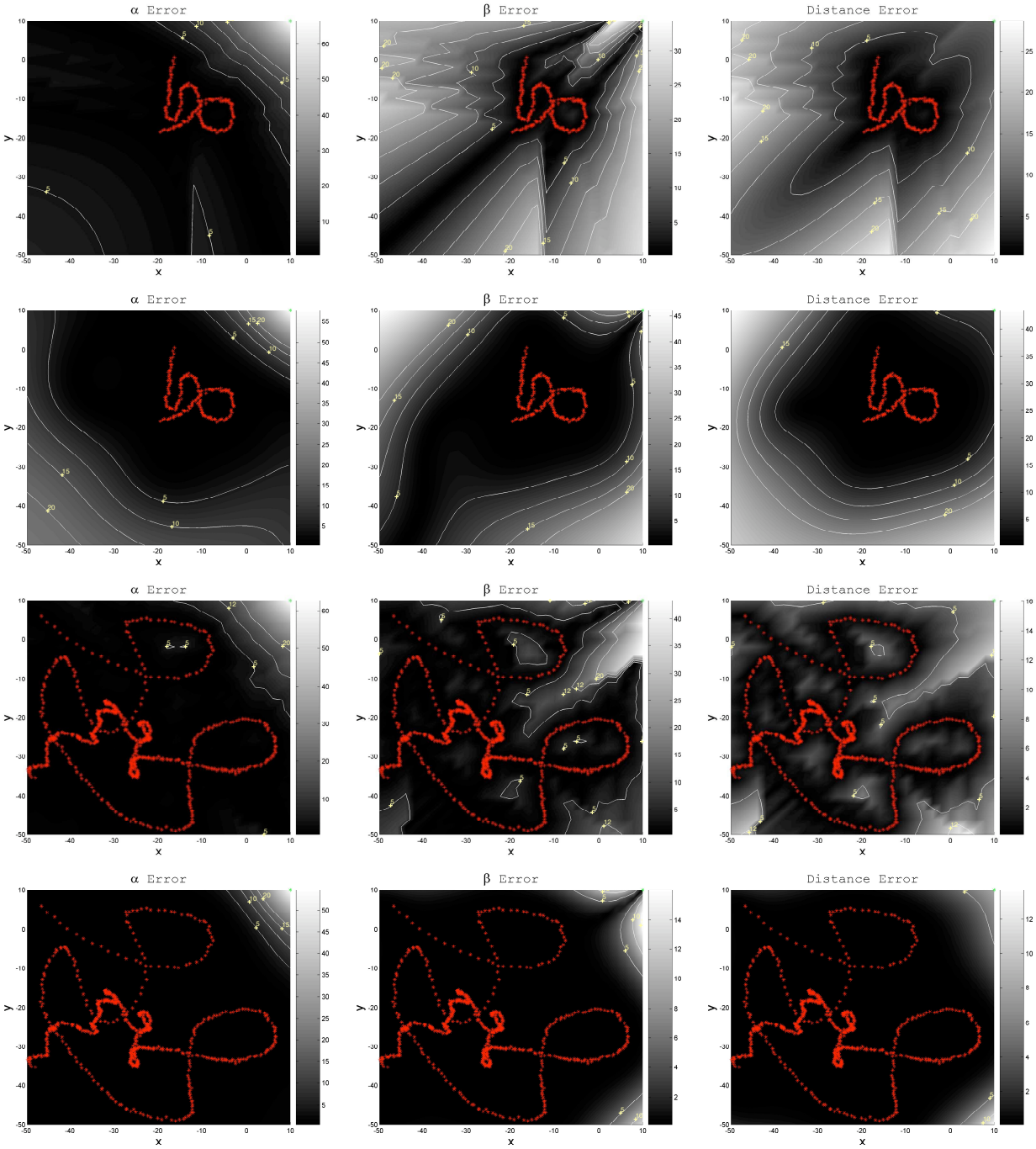


Figure 4.20: The map from 3D world locations to PTZ camera gaze direction using (Rows 1 and 3) k-nearest neighbor and (Rows 2 and 4) RBF approaches. Plots of the squared error between the true and estimated (Column 1) pan α and (Column 2) tilt β settings. (Column 3) Euclidean distance between the true 3D location and the point where the XY-plane ($Z = 0$) intersects the optical axis of the PTZ camera when its pan-tilt angles are set to the estimated α and β.

approach.

For each camera, we store a subsampled version of the lookup table, which is later used to estimate the accuracy of the estimated pan-tilt settings. The distance of $\mathbf{p}$ from the nearest (x, y, z) in the stored sub-sampled lookup table is a good indicator of the accuracy of the computed angles. If this distance is large, the PTZ camera invokes a search behavior to locate the pedestrian. In order to minimize the reliance on the initial learning phase, the lookup table is continously updated, and the RBF network re-trained, when the PTZ camera is following a pedestrian whose 3D location is known.

4.6 Summary

In this chapter, we have developed the low-level sensing machinery for our synthetic cameras:

- Synthetic video capture.
- Pedestrian segmentation via an automatically learned background model.
- Pedestrian tracking using color-based signatures for both passive and active cameras.
- Image-based fixation and zooming routines for active PTZ cameras.
- A strategy for estimating the 3D locations of the pedestrians present in the scene using passive wide-FOV cameras.
- A machine-learning based approach to learn the mapping between the 3D locations and the pan-tilt settings of active PTZ cameras.

The visual processing routines developed here will satisfy the low-level sensing requirements of the prototype camera networks presented in the next two chapters.

Chapter 5

Scheduling Active Cameras

Wide-FOV passive cameras are well-suited for providing continual, low-resolution coverage of a large area, and the video collected through these cameras is suitable for pedestrian detection and tracking. It is, however, unfit for activity recognition and pedestrian identification tasks, which require much higher resolution video. It is not feasible to provide high-resolution visual coverage over a large region using only passive cameras, and many researchers have suggested camera networks comprising both, wide-FOV passive and PTZ cameras to collect the necessary data [Collins et al. 2001; Zhou et al. 2003; Costello et al. 2004; Hampapur et al. 2003].

Low-resolution video from the wide-FOV cameras is used to build an estimate about pedestrians' locations within the designated region and the PTZ cameras are coordinated to collect high-resolution videos of the pedestrians present in the scene. Controlling PTZ cameras to observe the pedestrians present in the scene is a challenging task, which we cast as a resource allocation/scheduling problem. We adopt ideas from the scheduling literature and propose a novel camera scheduling strategy capable of controlling multiple uncalibrated active PTZ cameras to observe the pedestrians present in the scene.

In this chapter, Section 5.1 overviews our system, Section 5.2 introduces our scheduling strategy, we present results in Section 5.3, and Section 5.4 concludes with a summary of our contributions.

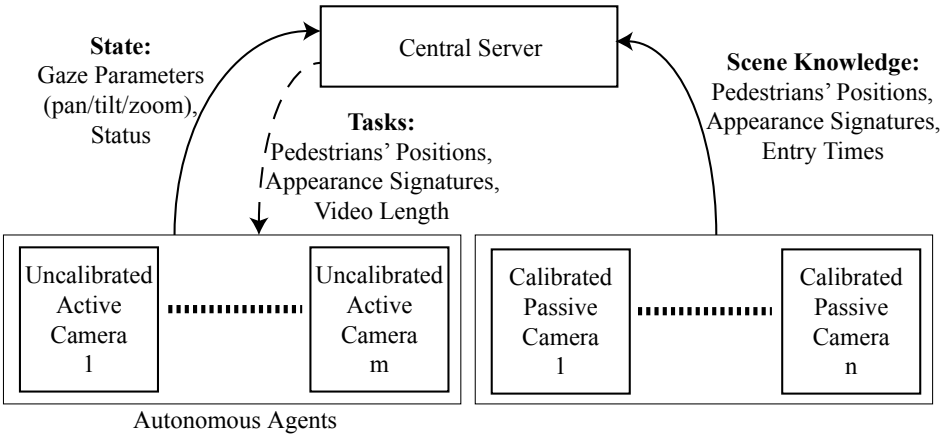


Figure 5.1: A central controller uses the scene information collected by calibrated, static cameras to schedule active PTZ cameras for recording close-up videos of the pedestrians in the region of interest. The PTZ cameras are autonomous agents that attempt to achieve the tasks assigned to them by the central controller. The dashed arc indicates the intermittent flow of instructions from the central controller to the PTZ cameras.

5.1 Surveillance System Overview

Our surveillance system is realized within the virtual Penn Station. The network comprises calibrated wide-FOV static cameras and uncalibrated active PTZ cameras. A central server acquires environmental information, such as the 3D positions of the pedestrians, through the static cameras and employs this information to schedule the active cameras in order to visually examine every pedestrian in the scene (Figure 5.1).

For the purposes of this active camera scheduling, we assume that the scene is monitored by more than one calibrated wide-FOV passive camera plus at least one PTZ active camera. Each camera has its own suite of Local Vision Routines (LVRs) that support pedestrian recognition, identification, and tracking. The LVRs are computer vision algorithms that directly operate upon the synthetic video generated by the virtual cameras and the information available from the 3D virtual world (See Chapter 4 for the details). At present, we assume that pedestrians can be reliably identified across multiple cameras. The static cameras track the 3D locations of pedestrians through *triangulation* (Section 4.3). An offline machine learning scheme learns the map between the gaze direction parameters (i.e., pan-tilt angles) of the PTZ cameras and target

3D world locations (Section 4.5). Each PTZ camera is modeled as an autonomous agent capable of recording the detailed video of the pedestrian of interest without relying on continuous feedback from the central controller. The PTZ camera uses image-based fixation and zooming routines to follow a pedestrian reliably (Section 4.4).

5.2 Camera Scheduling

The sensor network maintains an internal world model that reflects the current knowledge about the world. The model stores information about the pedestrians present in the scene, including their arrival times and the most current estimates of their positions and headings. The world model is available to the scheduling routine that assigns cameras to the various pedestrians in the scene. Using the 3D information stored in the world model, the cameras choose an appropriate gaze direction when viewing a particular pedestrian. The scheduling algorithm must find a compromise between two competing objectives: 1) to capture high-quality video for as many as possible, preferably all, of the pedestrians in the scene and 2) to observe each pedestrian for as long or as many times as possible, since the chances of identifying a pedestrian improve with the amount of data collected about that pedestrian. At one extreme, the camera follows a pedestrian for their entire stay in the scene, essentially ignoring all other pedestrians, whereas at the other extreme, the camera briefly observes every pedestrian in turn and repeatedly, thus spending most of the time transitioning between different pan, tilt, and zoom settings.

Following the reasoning presented in [Costello et al. 2004; Qureshi and Terzopoulos 2005a], the camera scheduling problem shares many characteristics with the network packet routing problem. Network packet routing is an online scheduling problem where the arrival times of packets are not known *a priori* and where each packet must be served for a finite duration before a deadline, when it is dropped by the router. Similarly, in our case, the arrival times of pedestrians entering the scene is not known *a priori* and a pedestrian must be observed for

some minimal amount of time by one of the PTZ cameras before he leaves the scene. That time serves as the deadline.

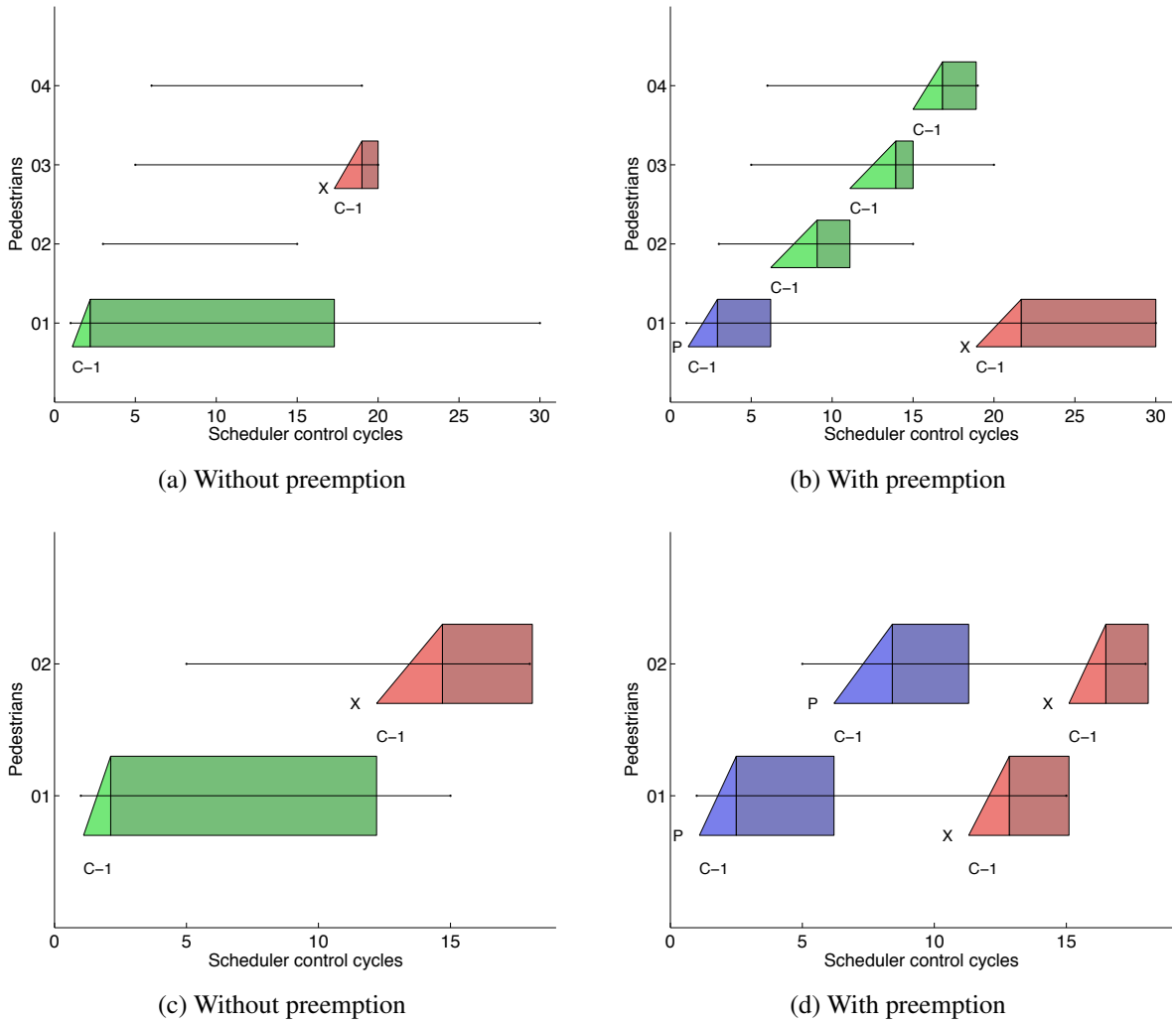
However, our problem differs from the packet routing problem in several significant ways. First, continuing with network terminology, we have multiple "routers" (one for every PTZ camera) instead of just one. This aspect of our problem is better modeled using scheduling policies for assigning jobs to different processors. Second, we typically must deal with additional sources of uncertainty: 1) it is difficult to estimate when a pedestrian might leave the scene and 2) the time period during which a PTZ camera should track and follow a pedestrian to record high-quality video suitable for further biometric analysis can vary depending on multiple factors; e.g., a pedestrian suddenly turning away from the camera, a tracking failure, an occlusion, etc.

Consequently, camera scheduling is an *online scheduling* problem (in scheduling jargon, a PTZ camera is a "processor" or "machine" and a pedestrian is a "job") for which 1) jobs are released over time, 2) have deadlines, 3) have different processing requirements, 4) can be assigned to one of many processors, and 5) the scheduler must schedule them without any knowledge of the future. Our approach is loosely related to the Multi Level Feedback Algorithm used for process scheduling in the Unix and Windows NT operating systems [Sgall 1998].

5.2.1 Online Scheduling Paradigms

An online scheduling algorithm does not have access to the entire input instance as it makes its decisions [Fiat and Woeginger 1998]. Thus, at each time t, the scheduler must decide which job(s) to run at t. Here, jobs typically have release times, and the scheduler is not aware of the existence of a job until its release time.

An online scheduler is referred to as "clairvoyant" when the processing times of the jobs are available to the scheduler upon their arrival. Such schedulers are modeled within an *online-time* paradigm. Alternatively, the lack of job processing time information is called "nonclair-



(a) Without preemption

(b) With preemption

(c) Without preemption

(d) With preemption

Figure 5.2: Scheduling a single camera to observe multiple pedestrians with and without preemption. (a), (b) A camera is scheduled to observe 4 pedestrians. (c), (d) A camera is scheduled to observe 2 pedestrians. Triangles represent lead times, rectangles represent processing times. Green indicates a successful recording, blue (or 'p') indicates a preemptive abort. Red (or 'x') indicates a failed recording due to the departure of a pedestrian. C-1 refers to Camera 1.

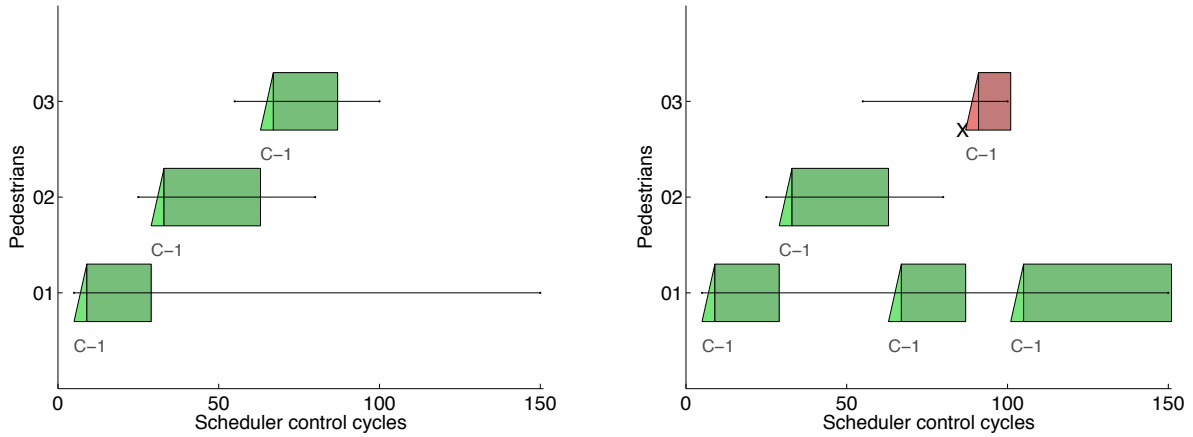
voyance" and such schedulers are modeled using the *online-time-nclv* paradigm. Since the exit times of pedestrians are difficult to predict, our scheduler is nonclairvoyant.

Another issue that arises in scheduling algorithms is that of *preemption*; i.e., interrupting a running job to process another job. A common scheme for implementing preemption is to pause an active job and later resume it on the same processor. Another possibility, which only exists in the online setting and is meaningless for offline scheduling algorithms, is to stop an existing job and restart it from the beginning on the same or a different processor. It is well known that if preemption is not allowed for problems in either the *online-time* or *online-time-nclv* model, and jobs can have arbitrary processing times, then the schedules produced by any online scheduler will be far from optimal [Sgall 1998]. This is why most online schedulers generally allow preemption, unless all jobs have similar processing times. Preemption incurs an overhead as job swapping (pausing and resuming jobs, saving and restoring job states, etc.) consumes resources.

In the camera scheduling application, for example, job swapping involves, among other overheads, locating the other pedestrian and initializing the tracking routine. Preemption is especially relevant to our application, as without it a PTZ camera can potentially track a pedestrian indefinitely without ever succeeding in recording video suitable for further biometric analysis. We therefore allow preemption. When there are multiple pedestrians in the scene, a PTZ camera is assigned to a pedestrian for some fixed maximum duration, after which the PTZ camera can be reassigned even if the video recording was unsuccessful.[1] The pedestrian whose video recording is terminated is treated as a new arrival and, later, the same or a different camera can attend to the pedestrian.

Figure 5.2 shows examples of the scheduling of a single camera to observe multiple pedestrians with and without preemption. Without preemption, the camera observes Pedestrian 1 until sufficient video is collected, ignoring all other pedestrians (Figure 5.2(a)). By contrast,

[1] An observation is deemed successful when the recorded video is suitable for further biometric analysis. The duration of the video depends on a number of factors, including the requirements of the biometric analysis routine, the quality of the recorded video, etc.



(a) Single observation

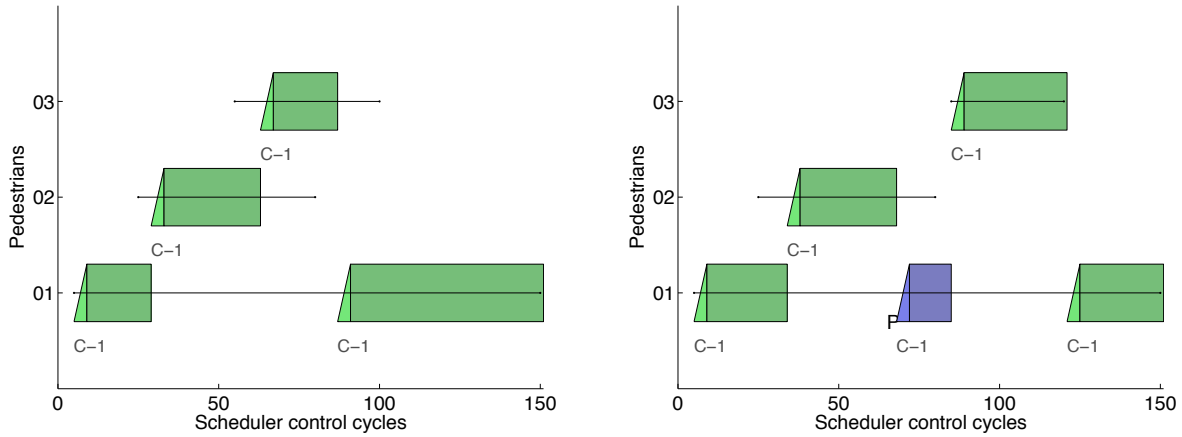
(b) Multiple observations

Figure 5.3: Scheduling a single camera to observe 3 pedestrians in single-observation and multiple-observations modes. A pedestrian's position in the priority sequence is computed using the FCFS policy. C-1 refers to Camera 1.

preemption prevents the camera from being monopolized by Pedestrian 1, so the camera can attend to Pedestrians 2, 3, and 4 even before sufficient video of Pedestrian 1 has been acquired (5.2(b)). The camera later resumes observing Pedestrian 1; however, the pedestrian leaves the scene before suitable video has been collected. Preemption is not advantageous in all situations and it can have unintended side effects. In Figure 5.2(c), Pedestrian 1 was successfully observed; however, with preemption, none of the pedestrians were observed long enough in 5.2(d).

For more effective preemption, we have adopted a multi-class pedestrian (job) model (Figure 5.4(a)–(b)). Every pedestrian is assigned to a class based on how many times the pedestrian has been observed successfully by a PTZ camera. Each sensed pedestrian is initialized as a member of Class 0 and advances to the next higher class after each successful observation. The class numbers of pedestrians together with their arrival times determine their positions in the priority queue, with priority given to pedestrians in lower classes. For example, cameras currently observing pedestrians belonging to Classes $1, 2, 3, \cdots$, are immediately reassigned to Class 0 pedestrians that have not yet been recorded (Figure 5.4(b)).

Figures 5.3 and 5.4 compare the naive FCFS-based priority calculation against the multi-



(a) Mutli-class priority

(b) Multiclass preemption

Figure 5.4: A single camera is scheduled to observe 3 pedestrians in multiple observation mode. The pedestrians' positions in the priority sequence is calculated using their arrival times and class memberships.

class priority computation scheme. The multi-class priority scheme is irrelevant when each pedestrian is observed only once. In single observation mode, each pedestrian is observed exactly once; e.g., in Figure 5.3(a) the camera is free after Pedestrian 3's departure, yet the camera is not assigned to observe Pedestrian 1 who is still present. However, it is desirable to observe a pedestrian multiple times when possible, in order to collect more data. In Figure 5.3(b), the camera is instructed to observe every pedestrian for as many times as possible, ergo Pedestrian 1 is observed twice. When the camera finishes observing Pedestrian 2, Pedestrian 1 is ahead of Pedestrian 3 in the priority sequence since the arrival time of Pedestrian 1 preceded the arrival time of Pedestrian 3.[2] The scheduler uses the naive FCFS priority computation, which does not differentiate between Pedestrians 1 and 3 based on their class memberships: Pedestrians 1 and 3 belong to Classes 1 and 0, respectively. Therefore, the camera is assigned to observe Pedestrian 1 *again* and Pedestrian 3 goes unnoticed.

Multi-class priority calculation resolves the above issue (Figure 5.4(a)). After the camera has successfully observed Pedestrian 2, priority is given to Pedestrian 3 (Class 0) over Pedes-

[2]In the multiple observations setting, the arrival time of a pedestrian is updated after each successful recording; e.g., in Figure 5.4(b), the arrival time of Pedestrian 1 was changed from the true arrival time, 5, to 23 after the pedestrian was successfully recorded.

trian 1 (Class 1), even though the arrival time of Pedestrian 1 precedes that of Pedestrian 3. Consequently, the camera is assigned to Pedestrian 3 and the camera successfully observes all three pedestrians. Furthermore, multi-class priority calculation allows more intelligent preemption. Consider Figure 5.4(b): The camera is observing Pedestrian 1 for the second time when Pedestrian 3 enters the scene. Upon sensing the new arrival, the camera immediately aborts observing Pedestrian 1 (Class 1) and attends to the previously unseen pedestrian (Pedestrian 3, Class 0).

5.2.2 Camera Scheduling Problem Formulation

The standard three-field notation for describing scheduling problems [Graham et al. 1997] is $\alpha|\beta|\gamma$, where α describes the processing environment, β encodes the job characteristics, and γ is the performance criterion for the scheduling algorithm. Using this notation, we can describe our camera scheduling problem as

$$P \,|\, r_j, \text{online-time-nclv}, \text{pmtn} \,|\, \sum U_j; \tag{5.1}$$

i.e., find a schedule on m processors that minimizes the total unit penalty when jobs j with deadlines d_j are released at time r_j. The jobs require arbitrary processing times and preemption (pmtn) is allowed. Minimizing the total unit penalty is akin to maximizing the number of jobs successfully completed prior to their deadlines. If C_j is the completion time of a job j, then

$$U_j = \begin{cases} 0 & \text{if } C_j \leq d_j; \\ 1 & \text{otherwise.} \end{cases} \tag{5.2}$$

The complexity of problem (5.1) is not known. However, the simpler problem $P2 \,|\, r_j, \text{pmtn} \,|\, \sum U_j$,[3] is at least NP-hard [Du et al. 1992]. Hence, our formulation of the camera scheduling problem is likely NP-hard. Consequently, we resort to a greedy algorithm for scheduling cameras to observe pedestrians.

[3] I.e., find a schedule on two processors that minimize the total unit penalty under release time constraints r_j, where release times are known *a priori* and preemption is allowed.

The obvious nonclairvoyant online algorithms are Round Robin (RR) and Shortest Elapsed Time First (SETF). RR devotes identical processing resources to all jobs, whereas SETF devotes *all* resources to the job that has been processed the least. As SETF is known to perform poorly when jobs are not fully parallelizable [Dobson 1984], we use weighted RR, a variant of RR. The weighted RR scheduling scheme is used to assign jobs to multiple processors with different load capacities. Each processor is assigned a weight indicating its processing capacity and more jobs are assigned to the processors with higher weights. We model each PTZ camera as a processor whose weights, which quantify the suitability of a camera with respect to observing a pedestrian, are adjusted dynamically. The weights are determined by two factors: 1) the number of adjustments the camera needs to make in the PTZ coordinates to fixate on the pedestrian and 2) the distance separating the pedestrian from the camera. In order to ensure fairness, we use a *First Come, First Served* (FCFS) priority scheme to select jobs that should be assigned to a processor (preemption forces job priorities to vary over time). Additionally, FCFS is said to be optimal when the optimization criterion is to minimize the maximum flow time, which is a measure of the quality of service defined as $C_j - r_j$ [Fiat and Woeginger 1998].

On the one hand, a camera that requires small adjustments in the PTZ coordinates to fixate on a pedestrian usually needs less *lead time* (the total time required by a PTZ camera to fixate on a pedestrian and initiate video recording) than a camera that needs to turn more drastically in order to bring the pedestrian into view. Consequently, we assign a higher weight to a camera that needs less redirection in order to observe the pedestrian in question. On the other hand, a camera that is closer to a pedestrian is more suitable for observing this pedestrian, since such an arrangement can potentially avoid occlusions, tracking loss, and subsequent re-initialization, by reducing the chance of another pedestrian intervening between the camera and the subject being recorded. The camera weights with respect to a pedestrian are computed as

$$w = \begin{cases} \exp\left(-\frac{(\theta-\hat{\theta})^2}{2{\sigma_\theta}^2} - \frac{(\alpha-\hat{\alpha})^2}{2{\sigma_\alpha}^2} - \frac{(\beta-\hat{\beta})^2}{2{\sigma_\beta}^2}\right) & \text{if the camera is free;} \\ 0 \quad \text{if the camera is busy,} \end{cases} \tag{5.3}$$

where $\hat{\theta} = (\theta_{\text{min}} + \theta_{\text{max}})/2$, $\hat{\alpha} = (\alpha_{\text{min}} + \alpha_{\text{max}})/2$, and $\hat{\beta} = (\beta_{\text{min}} + \beta_{\text{max}})/2$, and where θ_{min} and

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

Figure 5.5: (a)–(d) Images from wide-FOV passive cameras situated at the 4 corners of the main waiting room in the train station. (e) Image from a fish-eye camera mounted at the ceiling of the waiting room. These static cameras are calibrated, enabling the 3D positions of observed pedestrians to be estimated through triangulation.

θ_{max} are extremal field of view settings, α_{min} and α_{max} are extremal rotation angles around the x-axis (up-down), and β_{min} and β_{max} are extremal rotation angles around the y-axis (left-right). The values of the variances σ_θ, σ_α, and σ_β associated with each attribute are chosen empirically (in our experiments, we assigned $\sigma_\theta = \sigma_\alpha = \sigma_\beta = 5.0$). Here, α and β are the gaze parameters corresponding to the 3D location of the pedestrian as computed by the triangulation process, and θ is an approximate measure of the distance between the camera and the pedestrian. The distance between the camera and the pedestrian can also be approximated by declination angle, which can be estimated from α, under a ground-plane assumption.

A danger of using weighted round-robin scheduling is the possibility that a majority of the jobs will be assigned to the processor with the highest weight. We avoid this situation 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. The FCFS policy breaks ties on the basis of arrival times and pedestrian classes. Pedestrians in Class 0 (i.e., never observed by a PTZ camera) have the highest priority. Among pedestrians belonging to Class 0, the pedestrian that entered the scene first is selected. Pedestrians belonging to Classes 1 or higher are similarly selected on the basis of their arrival times. The arrival times of the pedestrians are maintained by the sensor network and are made available to the PTZ cameras.

The amount of time a PTZ camera will observe a pedestrian depends upon the number of pedestrians in the scene. However, we have specified a minimum time that a PTZ camera must

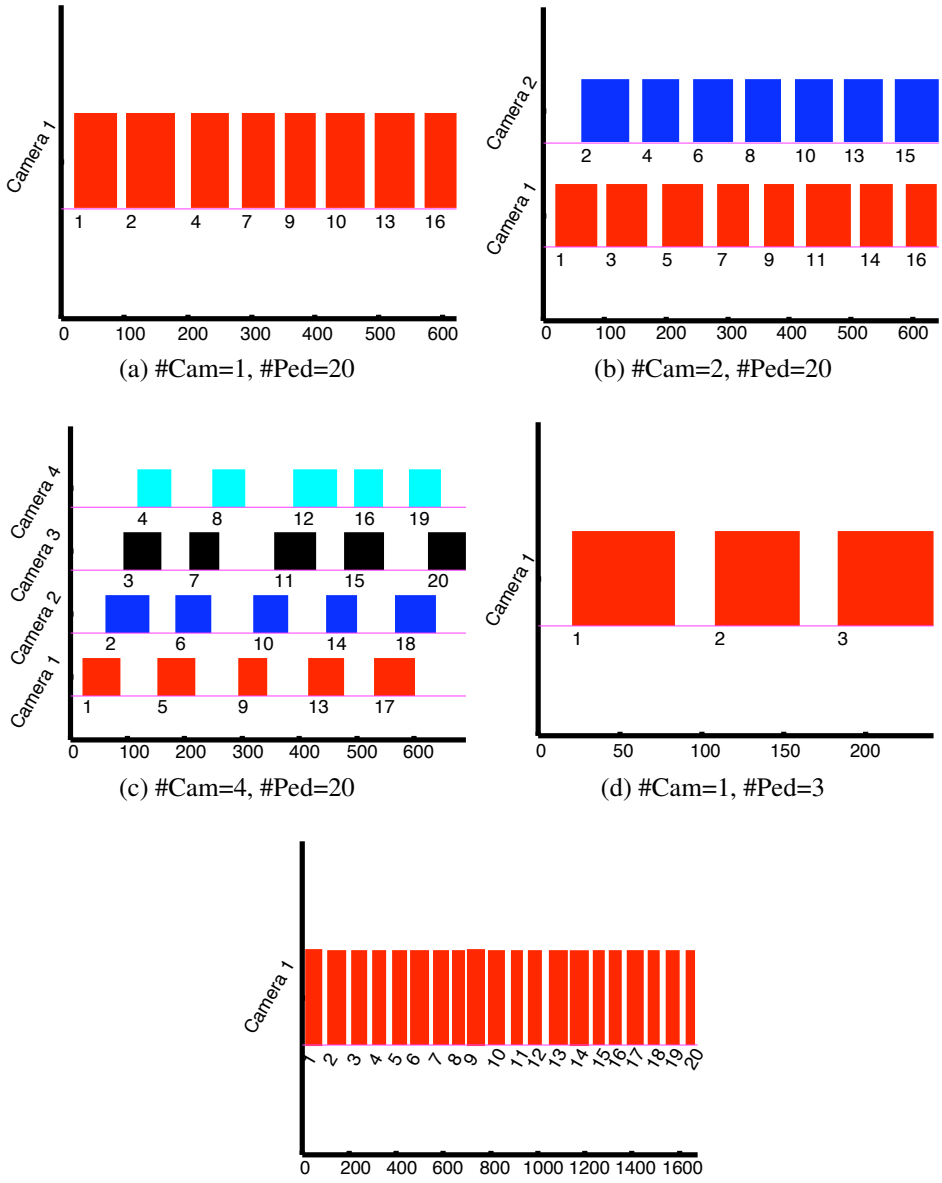
Figure 5.6: Sample close-up images captured by the PTZ active cameras.

spend observing a pedestrian. This is determined by the minimum length of video sequence required by the biometric routines that perform further evaluation plus the average time it takes a PTZ camera to lock onto and zoom into a pedestrian. To implement preemption, we specify the maximum time that a camera can spend observing a pedestrian when there are multiple pedestrians in the scene. The proposed scheduling scheme strikes a balance between the two often competing goals of following a pedestrian for as long as possible and observing as many pedestrians as possible.

5.2.3 Camera Scheduling Algorithm

We now present our *Preemption, Multiple Observations, Multi-class* (PMOMC) camera scheduling algorithm. All other variants discussed here can be realized within PMOMC.

Require: L_{busycam} comprises of the PTZ cameras currently assigned to the different pedestrians
Require: L_{freecam} contains the PTZ cameras that are currently available
Require: L_{unsched} comprises of the pedestrians that are currently not assigned a PTZ camera
Require: L_{sched} consists of the pedestrians that are currently being followed by a PTZ camera
Require: Initially L_{busycam}, L_{unsched}, and L_{sched} are empty
Require: Initially L_{freecam} consists of all available PTZ cameras
  for time $= 0, 1, 2, \cdots$ **do**
    Remove the pedestrians that appear to have left the scene from L_{sched} and L_{unsched}, and move the corresponding cameras from L_{busycam} to L_{freecam}
    for all New arrivals p **do**
      Set p's timestamp equal to the current time
      Set p's times-recorded count equal to 0
      Add p to L_{unsched}
    end for
    for all Cameras c in the L_{busycam} **do**
      Camera c is assigned to pedestrian p



(a) #Cam=1, #Ped=20

(b) #Cam=2, #Ped=20

(c) #Cam=4, #Ped=20

(d) #Cam=1, #Ped=3

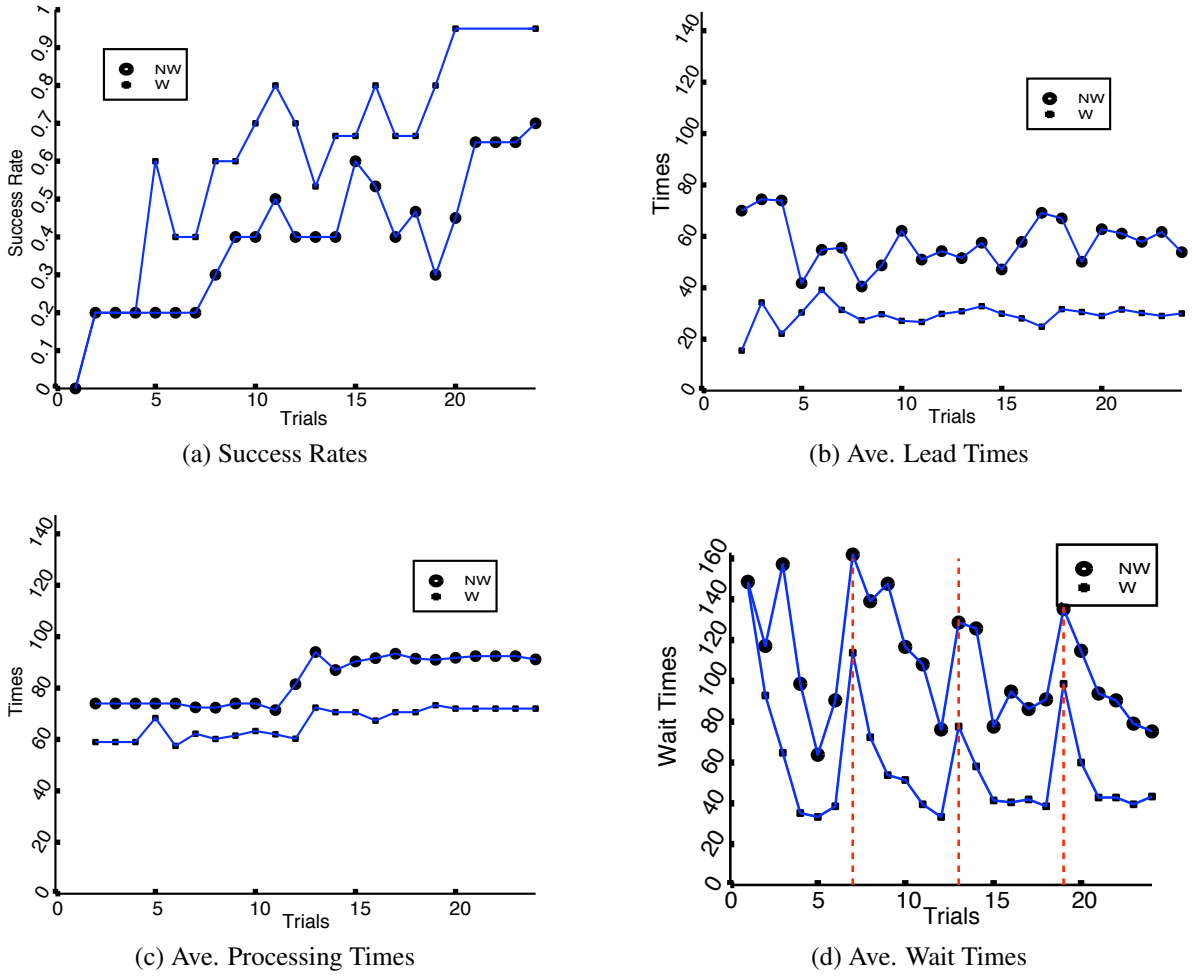
(e) #Cam=1, #Ped=20

Figure 5.7: Pedestrians are assigned unique identifiers based on their entry times; e.g., Pedestrian 1 always enters the scene at the same time or before the arrival of Pedestrian 2. (a)–(c) Twenty pedestrians are present in the scene. (a) The scheduling policy for one camera: Camera 1 successfully recorded Pedestrians 1, 2, 4, 7, 9, 10, 13, and 16. (b)–(c) Adding more cameras improves the chances of observing more pedestrians. Only Pedestrians 12, 17, 18, 19, and 20 go unnoticed when two cameras are available. With four cameras all pedestrians are observed. (d) The scene is populated with 3 pedestrians. (e) Twenty pedestrians, who tend to linger. The chances of a set of cameras to observe the pedestrians increase when (d) there are fewer pedestrians or when (e) pedestrians tend to linger.

if Camera c has finished recording video **then**
Set p's timestamp equal to the current time
Increment p's times-recorded count
Move c from L_{busycam} to L_{freecam}
Move p from L_{sched} to L_{unsched}
else if (Camera c has spent more than the allotted time on pedestrian p **and** L_{unsched} is not empty **and** Camera c is relevant to at least one of the pedestrians in L_{unsched}) **or** (p's times-recorded count ≥ 1 **and** A pedestrian p' in L_{unsched} has times-recorded count equal to 0 **and** Camera c is relevant to p') **then**
Set p's timestamp equal to the current time {*A camera is considered relevant to a pedestrian when the pedestrian weight values with respect to camera is above a prescribed threshold. Weights are computed using Equation 5.3*}
Move c from L_{busycam} to L_{freecam}
Move p from L_{sched} to L_{unsched}
end if
end for
for all Cameras c in L_{freecam} **do**
Compute $L_{\text{relevantped}}$, which consists of the pedestrians in L_{unsched} that are relevant to c
if $L_{\text{relevantped}}$ is empty **then**
Continue
else
Pick pedestrian p from $L_{\text{relevantped}}$ with the least times-recorded value
Assign c to p
Move c from L_{freecam} to L_{busycam}
Move p from L_{unsched} to L_{sched}
end if
end for
end for

5.3 Results

We populated the virtual train station with up to twenty autonomous pedestrians that enter, go about their business, and leave the waiting room of their own volition. We tested our scheduling strategy in various scenarios using from 1 to 18 PTZ active cameras. For example, Figure 5.5 shows our prototype surveillance system utilizing five wide-FOV static cameras situated within the waiting room of the train station. Figure 5.6 shows sample close-up images captured by four PTZ active cameras. The system behaved as expected in all cases, correctly tasking the available cameras using weighted round-robin scheduling with an FCFS priority policy.



(a) Success Rates

(b) Ave. Lead Times

(c) Ave. Processing Times

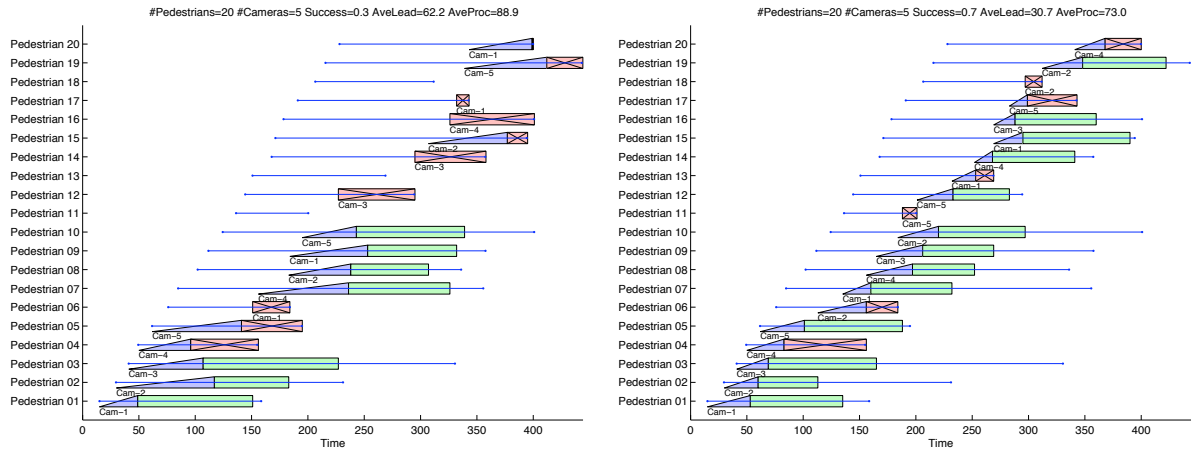
(d) Ave. Wait Times

Figure 5.8: A comparison of weighted (W) and non-weighted (NW) scheduling schemes. Equation (5.3) is used to compute camera weights (relevances) with respect to a pedestrian. The weighted scheduling strategy outperforms its non-weighted counterpart as is evident from its higher success rates (a) and shorter lead (b), processing (c), and wait (d) times. The displayed results are averaged over several runs in each trial scenario. Trials 1–6 involve 5 pedestrians and from 1 to 6 cameras, respectively. Trials 7–12 involve 10 pedestrians and from 3 to 8 cameras, respectively. Trials 13–18 involve 15 pedestrians and 5, 6, 9, 10, 11, and 12 cameras, respectively. Trials 19–24 involve 20 pedestrians with 5, 8, 10, 13, 15, and 18 cameras, respectively.

In a typical experiment (Figure 5.7(a)), when only one PTZ camera is available, Pedestrians 1, 2, 4, 7, 9, 10, 13, and 16 were recorded, but Pedestrians 3, 5, 6, 8, 11, 12, 14, 15, 17, 18, 19, and 20 go unnoticed since they left the scene before the camera had an opportunity to observe them. Figure 5.7(b) and (c) shows the results from the same run with 2 and 4 active cameras, respectively. In the 2-camera case, even though the performance has improved significantly from the added camera, Pedestrians 12, 17, 18, 19, and 20 still go unnoticed. With 4 PTZ cameras, the system is now able to observe every pedestrian. As expected, the chances of observing multiple pedestrians improve as more cameras become available.

For Figure 5.7(d), we have populated the virtual train station with only 3 autonomous pedestrians, leaving all other parameters unchanged. Given that there are now only 3 pedestrians in the scene, even a single camera successfully observes them. Next, we ran the simulation with 20 pedestrians (Figure 5.7(e)). This time, however, we changed the behavior settings of the pedestrians so they tend to linger in the waiting room. Here too, a single camera successfully observed each of the 20 pedestrians. We conclude that even a few cameras can perform satisfactorily when there are either few pedestrians in the scene or when the pedestrians tend to spend considerable time in the area.

In Figure 5.8, we compare the scheduling scheme that treats all cameras equally with the weighted scheduling scheme that takes into account the suitability of any camera in observing a pedestrian. As expected, the weighted scheduling scheme outperforms its non-weighted counterpart, exhibiting higher success rates (the fraction of pedestrians successfully recorded) and lower average lead time, processing time (the time spent recording the video of a pedestrian), and wait time (the time elapsed between the entry of a pedestrian and when the camera begins fixating on the pedestrian). The lower average lead and processing times are a consequence of the use of (5.3) in computing the suitability of a camera to recording a pedestrian. As expected, the average wait times typically decrease as we increase the number of cameras. In some cases, however, the average wait times increase slightly when a small number of cameras are added (e.g., see Trials 5 and 6 in Figure 5.8(d)). The variation in the recording durations

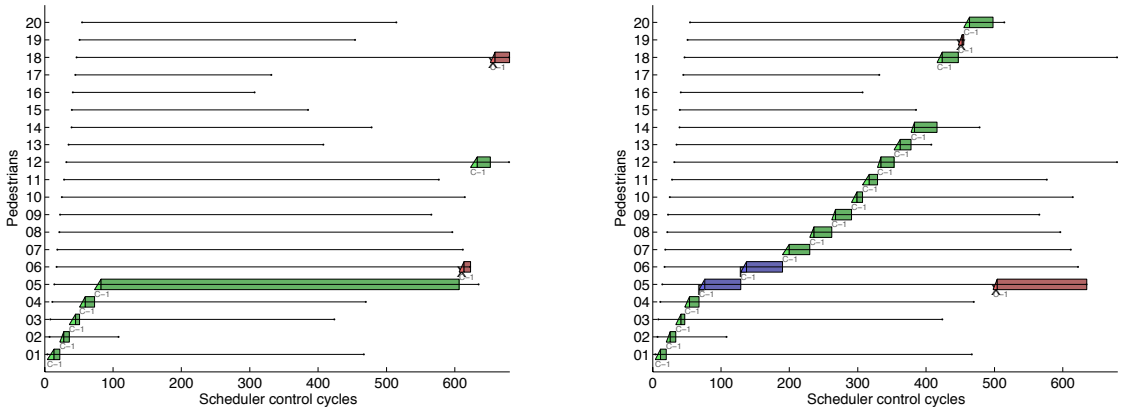


(a) Non-weighted (b) Weighted

Figure 5.9: Scheduling results for Trial #19. Non-weighted (a): Success Rate=0.3, Average Lead Time=62.2, Average Proc. Time=88.9. Weighted (b): Success Rate=0.7,Ave. Lead Time=30.7, and Ave. Proc. Time=73.0. Blue lines represent the entry and exit times, the Blue triangles represent the lead times, the Green rectangles represent the processing times, and the Red crossed rectangles represent an aborted attempt at capturing the video of a pedestrian.

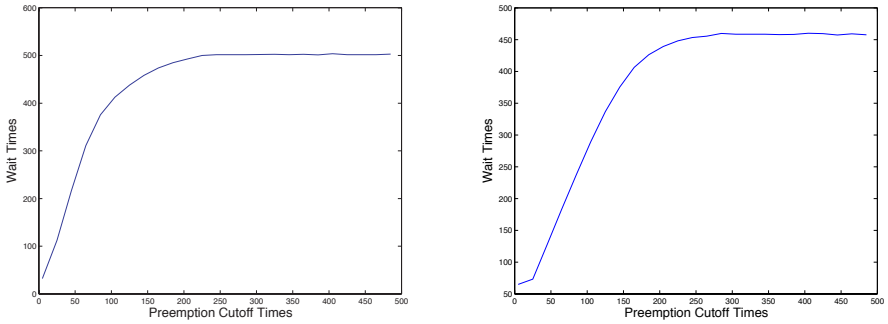
of different pedestrians accounts for this observation. The video recording duration depends upon multiple factors, including when a camera is assigned to a pedestrian, which camera is assigned to the pedestrian, what the position of the pedestrian is with respect to the camera and the other pedestrians present in the scene, and the position of the obstacles present in the scene. Figure 5.9 shows detailed results for the scenario with 20 pedestrians and 5 cameras.

The lack of preemption can impact the overall performance of the scheduler. In Figure 5.10(a), for example, the camera continues to observe Pedestrian 5 when preemption is not allowed—disregarding all other pedestrians in the scene. On the other hand, the camera decides to stop recording Pedestrian 5 and attend to the other pedestrians when preemption is enabled (Figure 5.10(b)). Choosing an appropriate value for the preemption cutoff time is critical to achieving the balance between the competing goals of observing as many pedestrians as possible and observing each pedestrian for as long as possible. Too small a value will result in the cameras spending most of their time transitioning between pedestrians, whereas too large a value will result in cameras myopically dwelling on pedestrians (Figure 5.11). We found that the average camera assignment time is a good indicator of the preemption cutoff time. In



(a) (b)

Figure 5.10: A single camera is scheduled to observe 20 pedestrians. Without preemption, the camera stays occupied with observing pedestrian 5, essentially ignoring all other pedestrians in the scene. Preemption enables the camera to relinquish observing pedestrian 5 and attend to other pedestrians in the scene. Triangles represent lead times and Rectangles represent recording durations. Lines represent entry and exit times. Colors Green, Blue, and Red encode a successful observation, an incomplete observation due to preemption, and a failed observation due to departure, respectively.



(a) #cam = 2 (b) #cam = 4

Figure 5.11: Two and four cameras are scheduled to observe 20 pedestrians. Smaller preemption cutoff times—i.e., the duration after which a camera is reassigned to observe another pedestrian even if its current assignment has not yet completed—results in smaller wait times; i.e., the time after which a camera is assigned to a new pedestrian. On the other hand, large preemption cutoff times results in a large wait times.

the current setting, the camera assignment time equals lead time plus processing time, and the preemption cutoff time should be no less than the average camera assignment time. However, if the variation in camera assignment times is large, then the average assignment time is a poor indicator of preemption cutoff times. Average assignment times can be computed on the fly using; e.g., using a running average.

We compared the proposed camera scheduling algorithm in the following six configurations in (Figure 5.12):

1. No preemption, single observation, single class (NPSOSC)
2. No preemption, multiple observation, single class (NPMOSC)
3. No preemption, multiple observation, multi-class (NPMOMC)
4. Preemption, single observation, single class (PSOSC)
5. Preemption, multiple observation, single class (PMOSC)
6. Preemption, multiple observation, multi-class (PMOMC)

For these tests, 1 to 4 cameras were scheduled to observe up to 20 pedestrians. The pedestrians enter the main waiting room of the station in groups of 10. Each group precedes the next by about 150 scheduler control cycles. Each pedestrian spends anywhere from 850 to 1500 scheduler control cycles in the waiting room. The preemption cutoff time is set to 170.

Preemption appears to be useful when the camera to pedestrian ratio is small or when there is a potential for cameras to continually record videos of the assigned pedestrians without success (a surprisingly common occurrence in camera scheduling due to tracking/occlusion issues). Scheduling without preemption has similar, or in some cases even higher, success rates than those achieved with preemption when the camera-to-pedestrian ratio is large (e.g., in Figure 5.12(a), when 3 or 4 cameras are available to observe up to 20 pedestrians). The success rates for no-preemption scheduling, however, drop rapidly as the number of available cameras

is reduced, and no-preemption scheduling is outperformed by its preemption-savvy counterpart; e.g., when 1 or 2 cameras are scheduled to observe 10 or 20 pedestrians (Figure 5.12(a)).

Single observation (SO) scheduling features better success rates than the corresponding multiple observation (MO) scheduling, which can be attributed to the longer wait times for the latter (Figure 5.12(b)), except when the multi-class pedestrian model is employed. Multi-class pedestrian modeling can reduce the wait times as it enables the scheduler to focus on the pedestrians that have been overlooked. When using MO scheduling, the lowest wait times are obtained by combining multi-class pedestrian models with preemption. PMOMC scheduling, for example, achieves wait times comparable to PSOSC with the added advantage of multiple observations. Our tests also confirm the intuition that multiple observation scheduling yields better data (i.e., longer duration video tracks) than single observation scheduling (Figure 5.12(d)). PMOMC offers the highest lead times as a result of the highest number of preemptions combined with high success rates (Figure 5.12(c)).

Our results suggest that PMOMC has the best overall performance: high success rates, low wait times, and longer recorded video durations. NPSOSC has better success rates under favorable circumstances—a good camera to pedestrian ratio and when all pedestrians have similar processing requirements.

5.4 Summary

In this chapter, we introduced a scheduling strategy for intelligently managing multiple, uncalibrated, active PTZ cameras, supported by several static, calibrated cameras, in order to satisfy the challenging task of automatically recording close-up, biometric videos of pedestrians present in a scene. We have found the PMOMC (preemption, multiple observation, multi-class) scheduling scheme to be the most suitable one for this purpose. At present, predicting pedestrian behaviors is at best an inexact science, so we have intentionally avoided scheduling policies that depend on such predictions.

In the next chapter, we will examine the problem of perceptive scene coverage for surveillance and propose a sensor network framework particularly suitable for designing camera networks for surveillance applications.

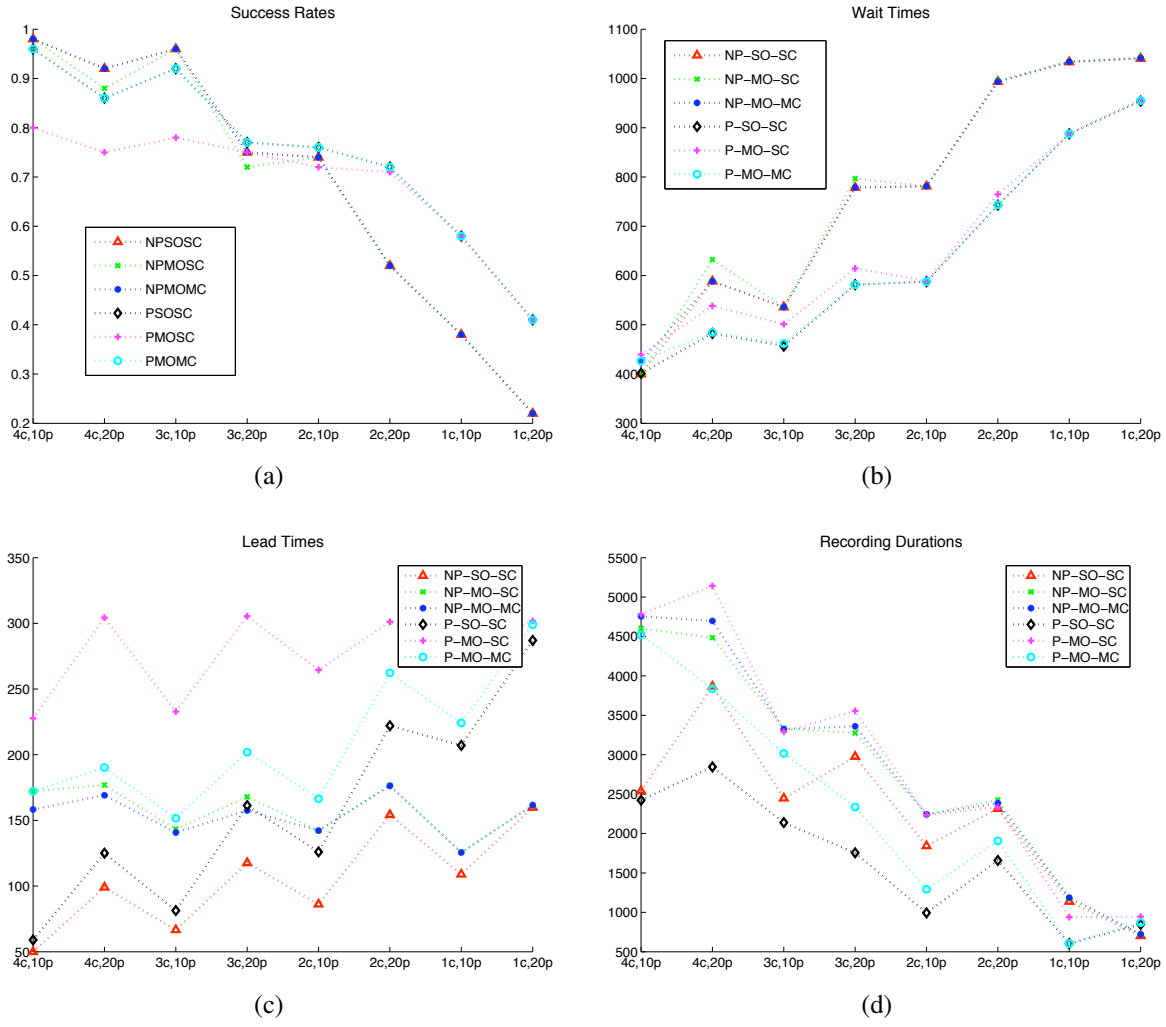


Figure 5.12: Comparison of various scheduler configurations. One to four cameras are scheduled to observe 10 or 20 pedestrians. Along the X-axis, letters 'c' and 'p' refer to the number of cameras and pedestrians, respectively. E.g, '4c,10p' denotes the test consisting of scheduling 4 cameras to observe 10 pedestrians. The results are averaged over 5 trials each.

Chapter 6

Perceptive Scene Coverage

Effective visual coverage of large public spaces, such as a train station or an airport, requires multiple cameras to work together towards common sensing goals. As the size of the camera network grows and the level of activity in the public space increases, it becomes infeasible for human operators to monitor the multiple video streams and identify all events of possible interest, or even to control individual cameras in performing advanced surveillance tasks, such as closely monitoring a pedestrian of interest as he meanders through the field-of-view (FOV) of multiple cameras, or zooming in on a particular subject to acquire one or more facial snapshots. It is, therefore, desirable to design camera networks that are capable of performing visual surveillance tasks autonomously, or at least with minimal human intervention.

Many of the characteristics and challenges associated with sensor networks are relevant to the work presented here. A fundamental issue in sensor networks is the selection of sensor nodes that participate in a particular sensing task [Zhao et al. 2003]. The selection process must take into account the information contribution of each node against its resource consumption or potential utility in other uses. Distributed approaches for node selection are preferable to centralized approaches that compromise what are perhaps the greatest advantages of networked sensing—robustness and scalability. Also, in a typical sensor network, each sensor node has local autonomy and can communicate with a small number of neighboring nodes that are within

the radio communication range.

Mindful of these issues, we propose a novel camera network control strategy that does not require camera calibration, a detailed world model, or a central controller. The overall behavior of the network is the consequence of the local processing at each node and inter-node communication. The network is robust to node and communication link failures; moreover, it is scalable due to the lack of a central controller. Visual surveillance tasks are performed by groups of one or more camera nodes. These groups, which are created on the fly, define the information sharing parameters and the extent of collaboration between nodes. A group keeps evolving—i.e., old nodes leave the group and new nodes join it—during the lifetime of the surveillance task. One node in each group acts as the group leader and is responsible for group level decision making. We also present a new constraint satisfaction problem formulation for resolving group-group interactions.

We explain the low-level vision emulation and behavior models for camera nodes in Section 6.1. Section 6.2 introduces the sensor network communication model. In Section 6.3, we demonstrate the application of this model in the context of visual surveillance. We present our results in Section 6.4.

6.1 Camera Node Behaviors

Each camera node is an autonomous agent capable of communicating with nearby nodes. The LVRs determine the sensing capabilities of a camera node, whose overall behavior is determined by the LVR (bottom-up) and the current task (top-down). We model the camera controller as an augmented hierarchical finite state machine (Figure 6.1).

In its default state, *Idle*, the camera node is not involved in any task. A camera node transitions into the *ComputingRelevance* state upon receiving a *queryrelevance* message from a nearby node. Using the description of the task that is contained within the *queryrelevance* message and by employing the LVRs, the camera node can compute its relevance to the task.

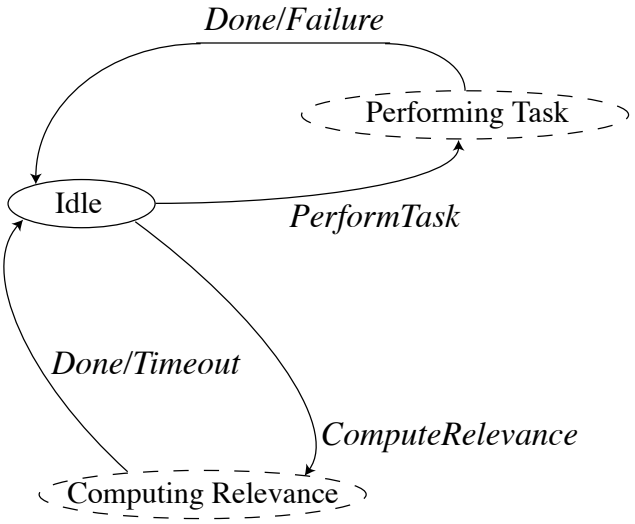


Figure 6.1: Top-level camera controller. Dashed states indicate embedded child finite state machines similar to the one shown in Figure 4.15.

For example, a camera can use visual search to find a pedestrian that matches the appearance-based signature passed by the querying node. The relevance encodes the expectation of how successful a camera node will be at a particular sensing task. The camera returns to the *Idle* state when it fails to compute the relevance due to the fact that it can not find a pedestrian that matches the description. On the other hand, when the camera successfully finds the desired pedestrian, it returns the relevance value to the querying node. The querying node passes the relevance value to the leader (leader node) of the group, which decides whether or not to include the camera node in the group. The camera goes into *PerformingTask* state upon joining a group where the embedded child finite state machine hides the sensing details from the top-level controller and enables the node to handle short-duration sensing (tracking) failures. Built-in timers allow the camera node to transition into the default state instead of being frozen in a state waiting forever for a message from another node. An expected message might never arrive due to a communication error or node failure.

Each camera can *fixate* and *zoom* in on an object of interest. Fixation and zooming routines

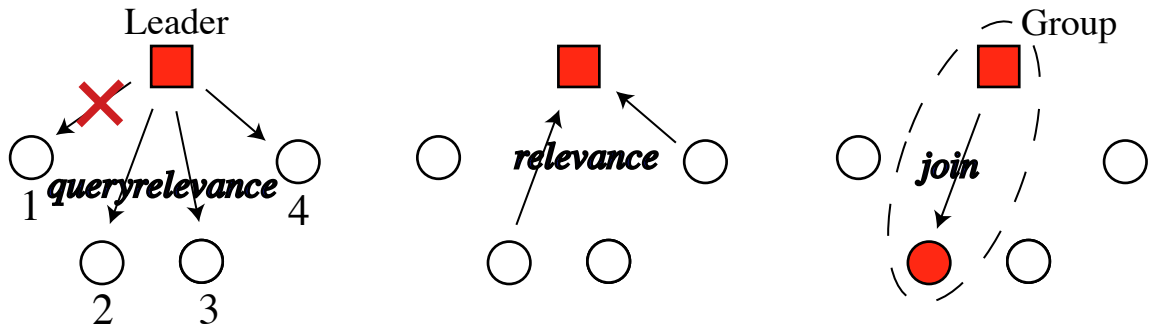
are image driven and do not require any 3D information, such as camera calibration or a global frame of reference. We discovered that traditional Proportional Derivative (PD) controllers generate unsteady control signals, resulting in jittery camera motions. The noisy nature of tracking forces the PD controller to try continuously to minimize the error metric without ever succeeding, so the camera keeps servoing. Hence, we model the fixation and zooming routines as dual-state controllers. The states are used to activate/deactivate the PD controllers. In the *act* state the PD controller tries to minimize the error signal; whereas, in the *maintain* state the PD controller ignores the error signal altogether and does nothing.

The *fixate* routine brings the region of interest—e.g., the bounding box of a pedestrian—into the center of the image by tilting the camera about its local x and y axes (Figure 4.16, Row 1). The *zoom* routine controls the FOV of the camera such that the region of interest occupies the desired percentage of the image. We provide the details of the *fixate* and *zoom* routines in Algorithms 4.17 and 4.18, respectively.

A camera node uses the *reset* routine to return to its default stance after finishing a task. The *reset* routine is modeled as a PD controller that attempts to minimize the error between the current zoom/tilt settings and the default zoom/tilt settings. We expect that for real PTZ cameras this would not be enough due to parameter drifting: a common occurrence in real hardware where the "true" setting of a parameter is not necessarily equal to the "dial" setting. We do not expect it to be a major issue though, as the true and dial settings typically follow each other closely; moreover, an image-based approach can drive the last stages of the reset operation and discover the dependencies between the dial and true settings.

6.2 Sensor Network Model

We now explain the sensor network communication scheme that enables task-specific node organization. The idea is as follows: A human operator presents a particular sensing request to one of the nodes. In response to this request, relevant nodes self-organize into a group with



(a) Announcement (b) Bidding (c) Selection

Figure 6.2: Node grouping via task auction. Filled square nodes represent leaders, filled circle nodes represent group nodes, and empty circle nodes represent nodes that are currently idle. The red cross indicates a lost message.

the aim of fulfilling the sensing task. The *group*, which formalizes the collaboration between member nodes, is a dynamic arrangement that keeps evolving throughout the lifetime of the task. At any given time, multiple groups might be active, each performing its respective task. Group formation is determined by the local computation at each node and the communication between the nodes. We require each node to compute its relevance to a task in the same currency. Our approach draws inspiration from behavior-based autonomous agents where the popularly held belief is that the overall intelligent behavior is a consequence of the interaction between many simple processes, called behaviors, rather than being the result of some powerful central processing facility. We leverage the interaction between the individual nodes to generate global task-directed behavior.

¿From the standpoint of user interaction, we have identified two kinds of sensing queries: 1) where the queried sensor itself can measure the phenomenon of interest—e.g., when a human operator selects a pedestrian to be tracked within a particular video feed—and 2) when the queried node might not be able to perform the required sensing and needs to route the query to other nodes. For instance, an operator can request the network to count the number of pedestrians wearing Green shirts. Currently, the network supports only the first kind of queries, which are sufficient for setting up collaborative tracking tasks.

6.2.1 Node Grouping

Node grouping commences when a node n receives a sensing query. In response to the query, the node sets up a named task and creates a single-node group. Initially, as node n is the only node in the group, it is chosen as the leader node. To recruit new nodes for the current task, node n begins by sending *queryrelevance* messages to its neighboring nodes, N_n. This is the announcement phase. A subset N' of N_n respond by sending their relevance values for the current task (*relevance* message). This is the bidding phase. Upon receiving the relevance values, node n selects a subset M of N' to include in the group, and sends *join* messages to the chosen nodes. This is the selection phase. When there is no resource contention between groups (tasks)—e.g., when only one task is active, or when multiple tasks that do not require the same nodes for successful operation are active—the selection process is relatively straightforward; node n picks those nodes from N' that have the highest relevance values. On the other hand, a conflict resolution mechanism is required when multiple groups vie for the same nodes; we present a scheme to handle this situation in the next section. A node that is not already part of any group can join the group upon receiving a *join* message from the leader of that group. After receiving the *join* message, a subset M' of M elect to join the group. Figure 6.2 illustrates this process.

For multinode groups, if a group leader decides to recruit more nodes for the task at hand, it instructs group nodes to broadcast task requirements. This is accomplished via sending *queryrelevance* to group nodes. The leader node is responsible for group-level decisions, so member nodes forward to the group leader all the group-related messages, such as the *relevance* messages from potential candidates for group membership. During the lifetime of a group, group nodes broadcasts *status* messages at regular intervals. Group leaders use *status* messages to update the relevance information of the group nodes. When a leader node receives a *status* message from another node performing the same task, the leader node include that node into its group. The leader node uses the most recent relevance values to decide when to drop a member node (Figure 6.3(a)–(b)). A group leader also removes a node from the group if it

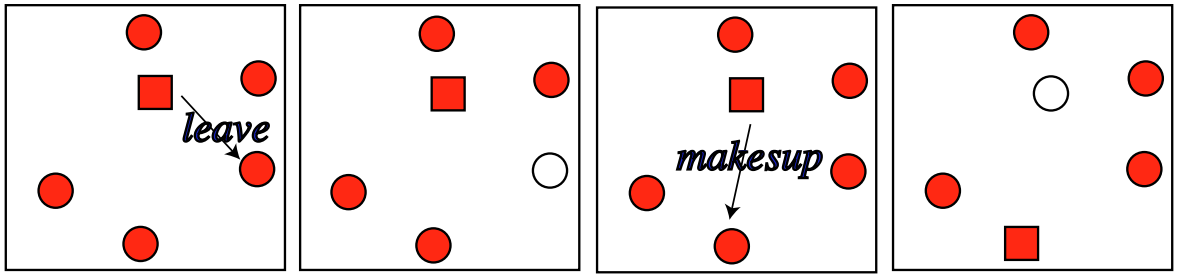


Figure 6.3: Group evolution. (a)-(b) A node leaves a group after receiving a leave message from the group leader. (c)-(d) Current leader selects a new group leader and leaves the group.

has not received a *status* message from the node in some preset time limit.[1] Similarly, a group node can choose to stop performing the task when it detects that its relevance value is below a certain threshold. When a leader detects that its own relevance value for the current task is below the predefined threshold, it selects a new leader from amongst the member nodes and leaves the group (Figure 6.3(c)–(d)). The group vanishes when the last node leaves the group.

Neighborhood of a node n typically consists of nodes that are at one hop distance from n. This, however, is an implementation detail, as we can as easily define neighborhoods to include nodes that are at multiple hop distance from n. It suggests that neighborhoods can be dynamically defined, e.g., a node can grow its neighborhood to include faraway nodes, if any of the adjacent nodes become irrelevant to the task at hand. One must bear in mind that larger group sizes can have negative effect on the overall performance of the network. Larger groups can potentially lead to more conflicts (with other groups) and larger groups have longer response times due the time it takes for a message to be seen by every node in the group. For the pedestrian following task, we expect group sizes to not exceed 10 or 12 cameras.

6.2.2 Conflict Resolution

[1]The relevance value of a group node is decayed over time in the absence of new *status* messages from that node. Thus, we can conveniently model node dependent timeouts; i.e., the time duration during which at least one *status* message must be received by the node in question.

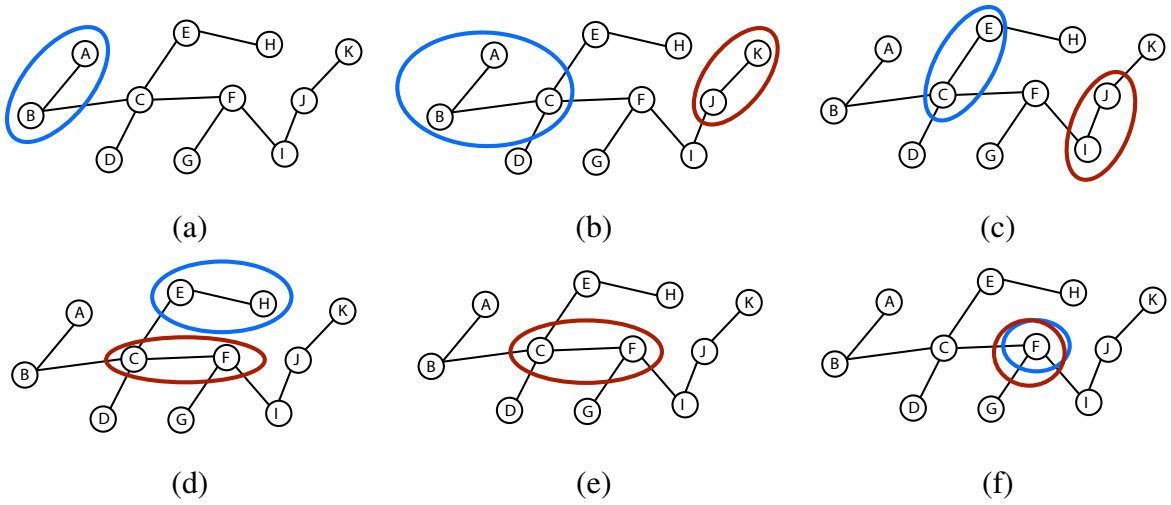


Figure 6.4: Grouping and conflict resolution. (a) Group 1: A and B; possible candidate: C (b) Group 1: A, B, and C; possible candidates: E, F, and D. Group 2: J and K; possible candidate: I. (c) Group 1: E and C; possible candidates: H, F, D, and B. Group 2: J and I; possible candidate: F and K. (d) Group 1: E and H; possible candidate C. Group 2: C and F; possible candidates: B, D, G, I, and E. (e) Group 1 and 2 require the same resources, so Group 1 vanished; task failure. (f) A unique situation where both groups successfully use the same nodes; e.g., imagine two groups tracking two pedestrians that started walking together.

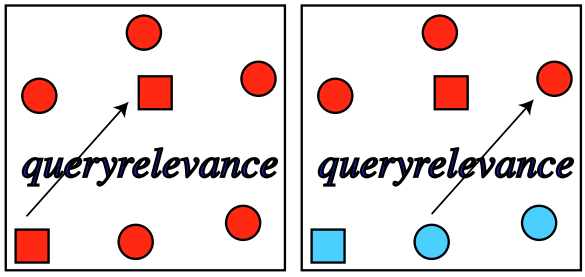


Figure 6.5: (a) A leader node detects another leader performing the same task; leader/supervisor demotion commences. (b) Conflict detection between two resources.

A conflict resolution mechanism is needed when multiple groups require the same resources (Figure 6.4(e)–(f)). The problem of assigning sensors to the contending groups can be treated as a Constraint Satisfaction Problem (CSP) [Pearson and Jeavons 1997]. Formally, a CSP consists of a set of variables $\{v_1, v_2, v_3, \cdots, v_k\}$, a set of allowed values Dom$[v_i]$ for each variable v_i (called the domain of v_i), and a set of constraints $\{C_1, C_2, C_3, \cdots, C_m\}$. The solution to the CSP is a set $\{v_i \leftarrow a_i | a_i \in \text{Dom}[v_i]\}$, where the a_is satisfy all the constraints.

We treat each group g as a variable, whose domain consists of the non-empty subsets of the set of sensors with relevance values (with respect to g) greater than a predefined threshold.[2] The constraints restrict the assignment of a sensor to multiple groups. Assume, for example, a group g and a set of nodes $\{n_1, n_2, n_3\}$ with relevance values (with respect to g) $\{r_1, r_2, r_3\}$, respectively. If r_3 is less than the predefined threshold, the set of nodes that will be considered for assignment to g is $\{n_1, n_2\}$, and the domain of g is the set $\{\{n_1\}, \{n_2\}, \{n_1, n_2\}\}$. We define a constraint C_{ij} as $a_i \cap a_j = \{\Phi\}$, where a_i and a_j are sensor assignments to groups g_i and g_j, respectively; k groups give rise to $k!/2!(k-2)!$ constraints.

We can then define a CSP problem $P = (G, D, C)$, where $G = \{g_1, g_2, \cdots, g_k\}$ is the set of groups (variables) with non-empty domains, $S = \{\text{Dom}[g_i] | i \in [1, k]\}$ is the set of domains for each group, and $C = \{C_{ij} | i, j \in [1, k], i \neq j\}$ is the set of constraints. To solve P, we employ *backtracking* to search systematically through the space of possibilities. When P does not have a solution, we recursively solve smaller CSP problems $P' = (G', D', C')$, where $G' \subset G$ and D' and C' are defined accordingly, until we find a solution to a smaller problem P'.

A node initiates the conflict resolution procedure upon identifying a group-group conflict; e.g., when it intercepts a *queryrelevance* message from multiple groups, or when it already belongs to a group and it receives a *queryrelevance* message from another group. The conflict resolution procedure begins by *centralizing* the CSP in one of the leader nodes that uses

[2] We can reduce the domain of a group g by drawing upon other sources of information. For example, domain of a group g can be restricted based upon the minimum and maximum number of sensors that must be assigned to it.

backtracking to solve the problem. The result is then conveyed to the other leader nodes.

CSPs have been studied extensively in the computer science literature and there exist more powerful variants of the basic backtracking method; however, we employ the naive backtracking approach in the interest of simplicity. Naive backtracking can easily cope with the size of problems encountered in the current setting. A key feature of the conflict resolution scheme proposed here is centralization, which requires that all the relevant information is gathered at the node that is responsible for solving the CSP. For smaller CSPs, the cost of centralization is easily offset by the speed and ease of solving the CSP. One can perhaps avoid centralization by using a scheme for distributed CSPs [Yokoo 2001]. Methods for solving distributed constraint satisfaction problems are in general more difficult to implement and have much higher communication requirements.

Solving CSP

Any solution of the above CSP problem P is a valid sensor node assignment; however, some solutions are better than others as not all nodes are equally suitable for any given sensing task. Node relevance value with respect to a group quantify the suitability of the node to the task performed by that group, and we can view the quality of a solution as a function of the quality of sensor assignments to different groups. In a restrictive setting, we can define the quality of a solution to be the sum of the quality of sensor assignments to individual groups.

When it is not possible to measure the quality of a partial solution to that of a full solution, we need to find all solutions, rank these solutions according to the relevance values for sensors (with respect to each group), and select the best solution to find the optimal assignments. This can be extremely costly and it is only feasible when the number of relevant nodes to a particular group is small. Results presented in Tables 6.1 and 6.2 confirm this expectation. For example, the total number of solutions remains manageable when we increase the number of interacting groups from 2 to 10, given that the average number of relevant sensors to a group stays at 4. On the other hand, the total number of solutions explodes and quickly becomes unmanageable as

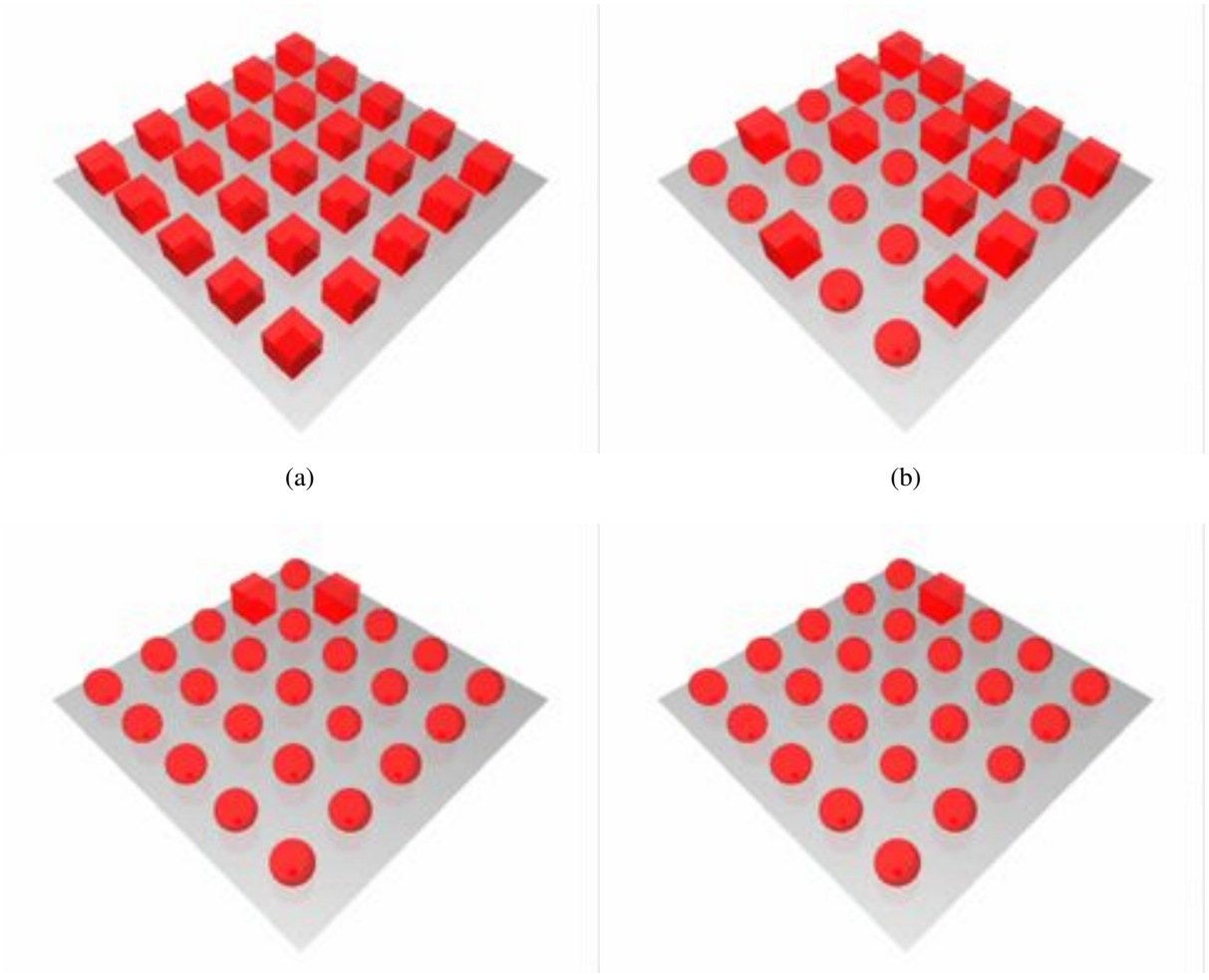
Test cases	1	2	3	4	5
Number of groups	2	4	6	8	10
Number of sensors per group	3	3	3	3	3
Average number of relevant sensors	4	4	4	4	4
Average domain size	4	4	4	4	4
Number of solutions	7	14	28	56	112
Nodes explored	12	54	138	306	642
Number of Backtracks	20	160	440	1000	2120
Solver used	*AllSolu*	*AllSolu*	*AllSolu*	*AllSolu*	*AllSolu*

Table 6.1: Finding optimal sensor node assignment. Three nodes each are assigned to upto 10 interacting groups. The average number of relevant nodes for each group is 4. *AllSolu* finds all solutions, rank them, and pick the best one.

we increase the number of relevant sensors, even when we are finding sensor assignments for two groups only. This is to be expected given the relationship between the number of relevant sensors to a group and the domain size for the corresponding variable in the CSP problem. When the number of relevant sensors to a group is n, the domain size for the corresponding variable in the CSP problem can be as large as $2^n - 1$.[3]

Alternately, when it is possible to compare the quality of a partial solution to that of a full solution, we can store the best result so far and backtrack whenever the current partial solution is of poorer quality. Using this strategy, we can guarantee an optimal solution under the assumption that the quality of solutions increase monotonically as values are assigned to more variables. For example, compare test cases 1 and 2 in Table 6.2. The goal was to assign 3 sensors each to the two groups. Optimal assignments were found in both cases; however, *BestSolu* that employs backtracking based on the quality of the partial solution visited only 175 nodes to find the optimal solution as opposed to *AllSolu* that visited 29290 nodes. The same trend is observed in columns 3 and 4 in Table 6.2. *BestSolu* solver clearly outperforms *AllSolu* solver in finding the optimal node assignment. Of course, we can always choose the first solution or pick the best solution after a pre-determined number of nodes have been explored when operating under time/resource constraints.

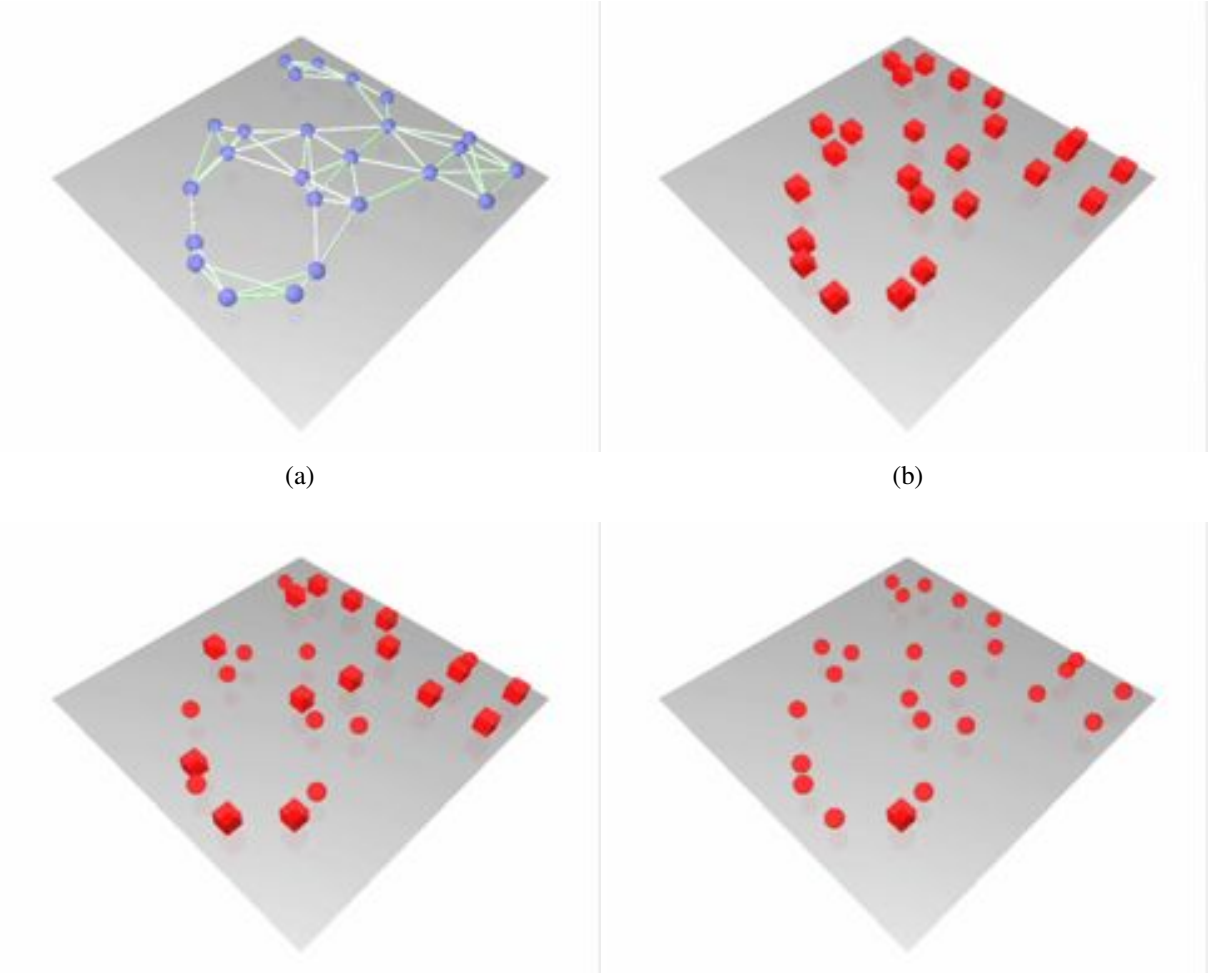
[3]A set with n elements has 2^n subsets including the Null set.



(a) (b)

(c) (d)

Figure 6.6: 25 single node groups merge to form one 25 node group with a single leader. Each node can directly communicate with its four neighbors. Leader demotion / group merging was carried out using the strategy outlined in Figure 6.8. Cube nodes represent leader nodes and spherical nodes represent group nodes.



(a) (b)

(c) (d)

Figure 6.7: 25 single node groups merge to form one 25 node group with a single leader. Communication graph is shown in (a). Leader demotion / group merging was carried out using the strategy outlined in Figure 6.8. Cube nodes represent leader nodes and spherical nodes represent group nodes. Blue nodes represent idle nodes.

Test cases	1	2	3	4
Number of groups	2	2	2	2
Number of sensors per group	3	3	3	3
Average number of relevant sensors	12	12	16	16
Average domain size	220	220	560	560
Number of solutions	29290	9	221347	17
Nodes explored	29511	175	221908	401
Number of Backtracks	48620	36520	314160	215040
Solver used	*AllSolu*	*BestSolu*	*AllSolu*	*BestSolu*

Table 6.2: Finding optimal sensor node assignment. The problem is to assign three sensors each to two groups. The average number of relevant nodes for each group is 12 and 16. *AllSolu* finds all solutions, rank them, and pick the best one; whereas, *BestSolu* computes the optimal solution by storing the best solution so far and backtracking when partial assignment yields a poorer solution. As expected *BestSolu* outperforms *AllSolu*.

6.2.3 Node Failures & Communication Errors

The purposed communication model takes into consideration node and communication failures. Communication failures are perceived as sensor failures; for example, when a node is expecting a message from another node, and the message never arrives, the first node concludes that the second node is malfunctioning. A node failure is assumed when the leader node does not receive a *status* from the node during some predefined interval, and the leader node removes the problem node from the group. On the other hand, when a member node does not receive any message (*status* or *queryrelevance*) from the leader node during a predefined interval, it assumes that the leader node has experienced a failure and selects itself to be the leader of the group. An actual or perceived leader node failure can therefore give rise to multiple single-node groups performing the same task. Multiple groups assigned to the same task are merged by demoting all of the leader nodes of the constituent groups, except one. Consider, for example, a group comprising three nodes a, b, and c; node a being the leader node. When a fails, b and c form two single-node groups and continue to perform the sensing task. In due course, nodes b and c discover each other—e.g., when b intercepts a *queryrelevance* or a *status* message from c—and they form a new group comprising b and c, demoting node c in the process. Thus, our proposed communication model is able to handle node failures.

Demotion is either carried out based upon the unique ID assigned to each node—among the conflicting nodes, the one with the highest ID is selected to be the group leader—or when unique node IDs are not guaranteed, demotion can be carried out via the process shown in Figure 6.8. Figures 6.6 and 6.7 show two demotion sequences using the strategy outlined in Figure 6.8.

The following observations suggest that our leader demotion strategy is correct; i.e., only a single leader node survives the demotion negotiations, every other leader node is demoted.

- **Observation 1:** Demotion process between two leader nodes either succeeds or fails. It succeeds when one of the two nodes is demoted. Demotion between two nodes is based on the contention management scheme that was first introduced in ALOHA network protocol [Kuo 1995]. ALOHA network protocol was developed in the late 60s and it is a precursor to the widely used Ethernet protocol. In its basic version, ALOHA protocol states

  - if you have data to send, send it.
  - if there is a collision, resend after a random interval.

  We point the interested reader to [Murthy and Manoj 2004] for more details about the ALOHA network protocol. What is important here is to note that eventually one of the two leader nodes will be demoted, i.e., the demotion process between two nodes will eventually succeed.

- **Observation 2:** Demotion process between more than two nodes involves repeated (distributed and parallel) application of the demotion process between two nodes.

6.3 Video Surveillance

Assumptions: Nodes n and m are two leader nodes performing task 1.

Case 1: Node n receives a *queryrelevance* or *status* message from node m.

- Node n is not involved in demotion negotiations with another node **then** send *demote* message to node m after a random interval.

Case 2: Node n receives a *demote* message from node m.

- Node n has not sent a *demote* message to another node **then** demote node n and send *demoteack* message to node m.
- Node n has sent a *demote* message to node m **then** send *demoteretry* message to node m and send a *demote* message to node m after a random interval.
- Node n has sent a *demote* message to another node **then** send a *demotenack* message to node m.

Case 3: Node n receives a *demotenack* message from node m.

- Terminate demotion negotiations with node m.

Case 4: Node n receives a *demoteack* message from node m.

- Add m to node n's group.

Case 5: Node n receives a *demoteretry* message from node m.

- Send a *demote* message to node m after a random interval.

Figure 6.8: Demotion negotiations

Let us now consider how a sensor network of dynamic cameras may be used in the context of video surveillance (Figure 6.9). A human operator spots one or more suspicious pedestrians in one of the video feeds and, for example, requests the network to "track this pedestrian," "zoom in on this pedestrian," or "track the entire group." The successful execution and completion of these tasks requires intelligent allocation and scheduling of the available cameras; in particular, the network must decide which cameras should track the pedestrian and for how long.

A detailed world model that includes the location of cameras, their fields of view, pedestrian motion prediction models, occlusion models, and pedestrian movement pathways may allow

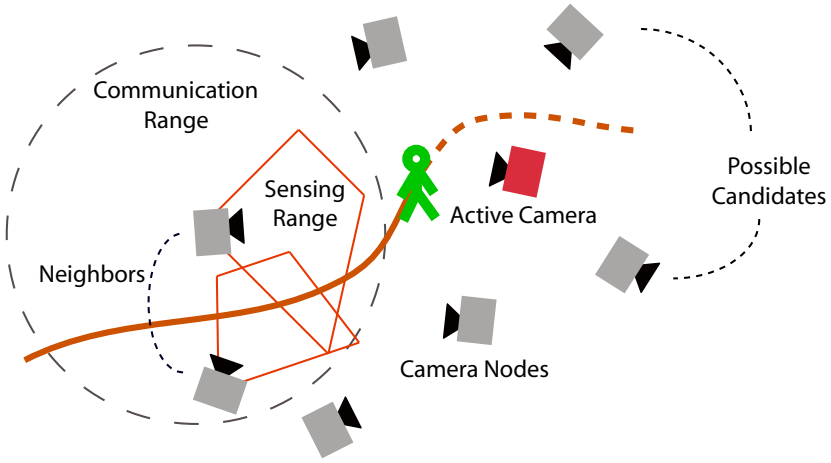


Figure 6.9: A camera network for video surveillance consists of camera nodes that can communicate with other nearby nodes. Collaborative tracking requires that cameras organize themselves to perform camera handover when the tracked subject moves out of the sensing range of one camera and into that of another.

(in some sense) *optimal* allocation and scheduling of cameras; however, it is cumbersome and in most cases infeasible to acquire such a world model. Our approach does not require such a knowledge base. We assume only that a pedestrian can be identified by different cameras with reasonable accuracy and that the camera network topology is known *a priori*. A direct consequence of this approach is that the network can easily be modified through removal, addition, or replacement of camera nodes.

6.3.1 Computing Camera Node Relevance

The accuracy with which individual camera nodes are able to compute their relevance to the task at hand determines the overall performance of the network. Our scheme for computing the relevance of a camera to a video surveillance task encodes the intuitive observations that 1) a camera that is currently free should be chosen for the task, 2) a camera with better tracking performance with respect to the task at hand should be chosen, 3) the turn and zoom limits of cameras should be taken into account when assigning a camera to a task; i.e., a camera that has more leeway in terms of turning and zooming might be able to follow a pedestrian for a longer time, and 4) it is better to avoid unnecessary reassignments of cameras to different tasks, as

Status	=	$s \in \{\text{busy}, \text{free}\}$
Quality	=	$c \in [0, 1]$
Fov	=	$\theta \in [\theta_{\text{min}}, \theta_{\text{max}}]$ degrees
XTurn	=	$\alpha \in [\alpha_{\text{min}}, \alpha_{\text{max}}]$ degrees
YTurn	=	$\beta \in [\beta_{\text{min}}, \beta_{\text{max}}]$ degrees
Time	=	$t \in [0, \infty)$ seconds
Task	=	$a \in \{a_i \mid i = 1, 2, \cdots\}$

Figure 6.10: The relevance metric returned by a camera node relative to a new task request. The leader node converts the metric into a scalar value representing the relevance of the node for the particular surveillance task.

that might degrade the performance of the underlying computer vision routines.

Upon receiving a task request, a camera node returns to the leader node a relevance metric—a list of attribute-value pairs describing its relevance to the current task along multiple dimensions (Fig. 6.10). The leader node converts this metric into a scalar relevance value r as follows:

$$r = \begin{cases} \exp\left(-\frac{(c-1)^2}{2{\sigma_c}^2} - \frac{(\theta-\hat{\theta})^2}{2{\sigma_\theta}^2} - \frac{(\alpha-\hat{\alpha})^2}{2{\sigma_\alpha}^2} - \frac{(\beta-\hat{\beta})^2}{2{\sigma_\beta}^2}\right) \\ \qquad\qquad\qquad\qquad \text{when } s = \text{free} \\ \frac{t}{t+\gamma} \quad \text{when } s = \text{busy} \end{cases} \tag{6.1}$$

where $\hat{\theta} = (\theta_{\text{min}} + \theta_{\text{max}})/2$, $\hat{\alpha} = (\alpha_{\text{min}} + \alpha_{\text{max}})/2$, and $\hat{\beta} = (\beta_{\text{min}} + \beta_{\text{max}})/2$, and where θ_{min} and θ_{max} are extremal field of view settings, α_{min} and α_{max} are extremal rotation angles around the x-axis (up-down), and β_{min} and β_{max} are extremal rotation angles around the y-axis (left-right). Here, $0.3 \leq \sigma_c \leq 0.33$, $\sigma_\theta = (\theta_{\text{max}} - \theta_{\text{min}})/6$, $\sigma_\alpha = (\alpha_{\text{max}} - \alpha_{\text{min}})/6$, and $\sigma_\beta = (\beta_{\text{max}} - \beta_{\text{min}})/6$. The value of γ is chosen empirically (for our experiments we have selected γ to be 1000).

The computed relevance values are used by the node selection scheme described above to assign cameras to various tasks. The leader node gives preference to the nodes that are currently free, so the nodes that are part of another group are selected only when an insufficient number of free nodes are available for the current task.

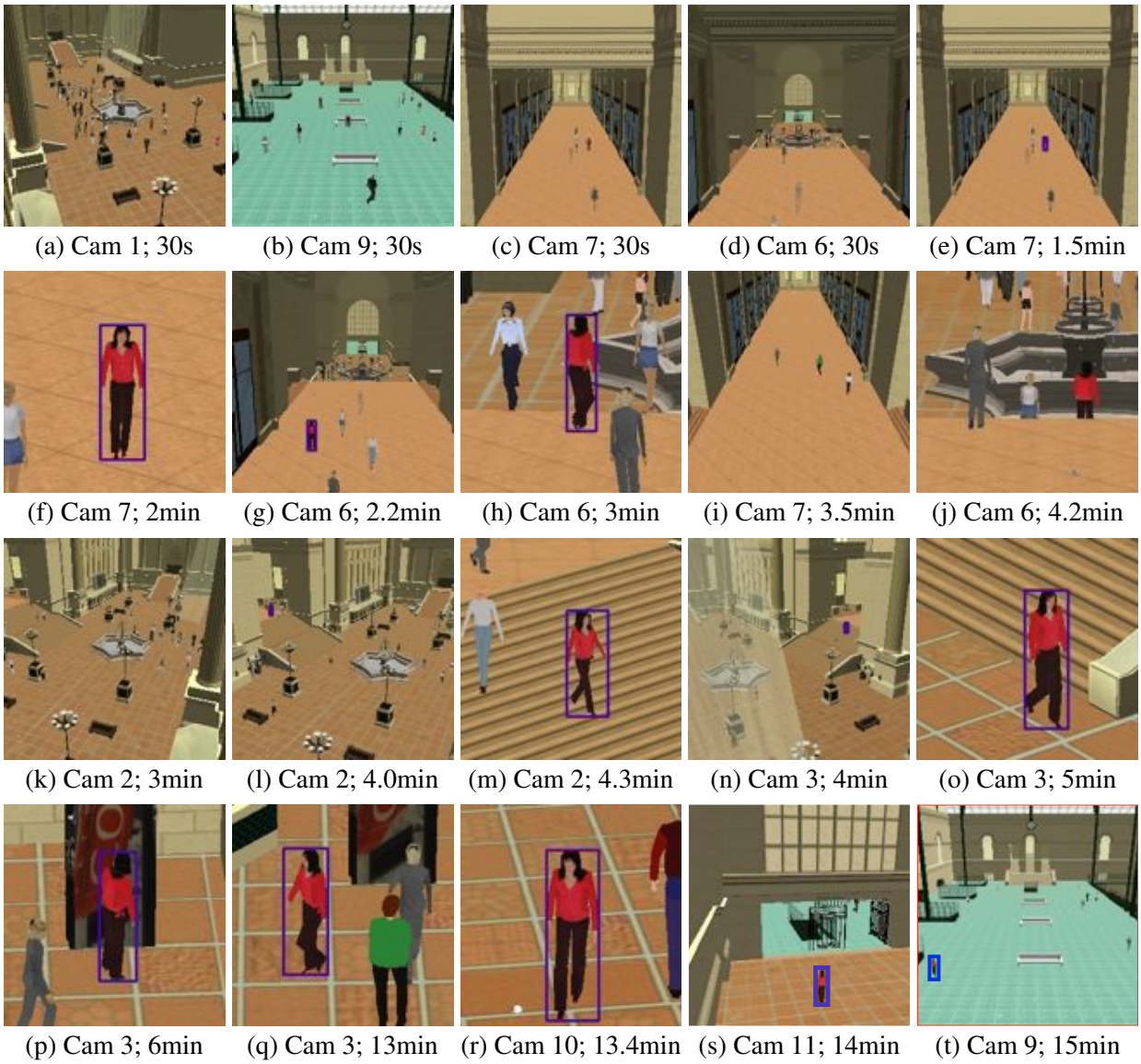
6.3.2 Surveillance Tasks

We have implemented an interface that presents the operator a display of the synthetic video feeds from multiple virtual surveillance cameras (c.f., Figure 2.2). The operator can select a pedestrian in any video feed and instruct the camera network to perform one of the following tasks: 1) follow the pedestrian, 2) capture a high-resolution snapshot, or 3) zoom-in and follow the pedestrian. The network then automatically assigns cameras to fulfill the task requirements. The operator can also initiate multiple tasks, in which case either cameras that are not currently occupied are chosen for the new task or some cameras are reassigned to the new task.

6.4 Results

To date, we have tested our visual sensor network system with up to 16 stationary and pan-tilt-zoom cameras, and we have populated the virtual Penn station with up to 100 pedestrians. The sensor network correctly assigned cameras in most cases. Some of the problems that we encountered are related to pedestrian identification and tracking. As we increase the number of virtual pedestrians in the train station, the identification and tracking module has increasing difficulty following the correct pedestrian, so the surveillance task fails (and the cameras just return to their default settings).

For the example shown in Figure 6.11, we placed 16 active PTZ cameras in the train station, as shown in Figure 2.3. An operator selects the pedestrian with the red shirt in Camera 7 (Figure 6.11(e)) and initiates the "follow" task. Camera 7 forms the task group and begins tracking the pedestrian. Subsequently, camera 7 recruits camera 6, which in turn recruits cameras 2 and 3 to track the pedestrian. Camera 6 becomes the leader of the group when camera 7 loses track of the pedestrian and leaves the group. Subsequently, camera 6 experiences a tracking failure, sets camera 3 as the group leader, and leaves the group. Cameras 2 and 3 track the pedestrian during her stay in the main waiting room, where she also visits a vending machine. When the pedestrian starts walking towards the concourse, cameras 10 and 11 take over the group from cameras 2 and 3. Cameras 2 and 3 leave the group and return to their default modes.



(a) Cam 1; 30s (b) Cam 9; 30s (c) Cam 7; 30s (d) Cam 6; 30s (e) Cam 7; 1.5min

(f) Cam 7; 2min (g) Cam 6; 2.2min (h) Cam 6; 3min (i) Cam 7; 3.5min (j) Cam 6; 4.2min

(k) Cam 2; 3min (l) Cam 2; 4.0min (m) Cam 2; 4.3min (n) Cam 3; 4min (o) Cam 3; 5min

(p) Cam 3; 6min (q) Cam 3; 13min (r) Cam 10; 13.4min (s) Cam 11; 14min (t) Cam 9; 15min

Figure 6.11: A pedestrian is successively tracked by cameras 7, 6, 2, 3, 10, and 9 (see Fig. 2.3) as she makes her way through the station to the concourse. (a-d) Cameras observing the station. (e) Operator selects a pedestrian in feed 7. (f) Camera 7 has zoomed in on the pedestrian, (g) Camera 6, which is recruited by camera 7, acquires the pedestrian. (h) Camera 6 zooms in on the pedestrian. (i) Camera 7 reverts to its default mode after losing track of the pedestrian—it is now ready for another task (j) Camera 6 has lost track of the pedestrian. (k) Camera 2. (l) Camera 2, which is recruited by camera 6, acquires the pedestrian. (m) Camera 2 tracking the pedestrian. (n) Camera 3 is recruited by the camera 6; camera 3 has acquired the pedestrian. (o) Camera 3 zooming in on the pedestrian. (p) Pedestrian is at the vending machine. (q) Pedestrian is walking towards the concourse. (r) Camera 10 is recruited by camera 3; camera 10 is tracking the pedestrian. (s) Camera 11 is recruited by camera 10. (t) Camera 9 is recruited by camera 10.

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

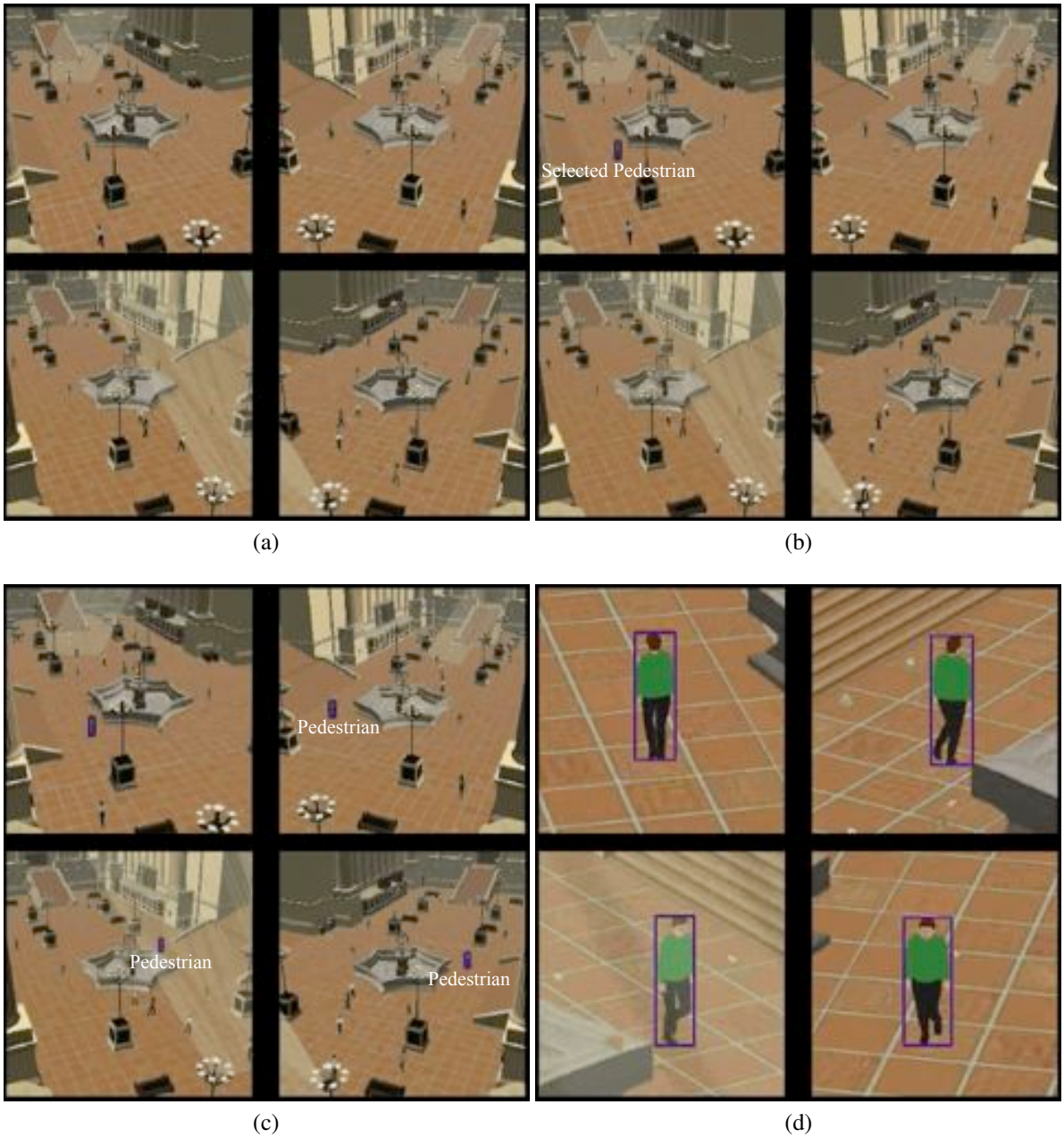
Figure 6.12: "Follow" sequence. (a) The operator selects a pedestrian in Camera 1 (upper row). (b) and (c) Camera 1 and Camera 2 (lower row) are tracking the pedestrian. (d) Camera 2 loses track. (e) Camera 1 is still tracking; Camera 2 has returned to its default settings.

Later camera 11, which is now acting as the group's leader, recruits camera 9, which tracks the pedestrian as she enters the concourse.

Figure 6.12 illustrates a "follow" task sequence. An operator selects the pedestrian with the green shirt in Camera 1 (top row). Camera 1 forms a group with Camera 2 (bottom row) to follow and zoom in on the pedestrian. At some point, Camera 2 loses the pedestrian (due to occlusion), and it invokes a search routine, but it fails to reacquire the pedestrian. Camera 1, however, is still tracking the pedestrian. Camera 2 leaves the group and returns to its default settings.

Figure 6.13 displays video surveillance panel showing video feeds from four uncalibrated active PTZ cameras situated at the corners of the waiting room of the virtual train station. The user selects a pedestrian in Camera 1 by drawing a rectangle around it (Top-left feed in Figure 6.13(b)). Camera 1 computes the appearance signature (i.e., color histogram) of the pedestrian and broadcasts it to the neighboring cameras. Cameras 2, 3, and 4 successfully locate the pedestrian using its appearance signature (Figure 6.13(c)). The four cameras form a group and zoom in on the pedestrian to capture close-up video (Figure 6.13(d)).

Figure 6.14 displays video surveillance panel showing video feeds from four calibrated



(a) (b)

(c) (d)

Figure 6.13: Video surveillance panels showing video feeds from four uncalibrated active PTZ cameras located at the corners of the waiting room in the virtual train station (counter-clockwise from top-left is video feed from cameras 1, 3, 4, and 2).

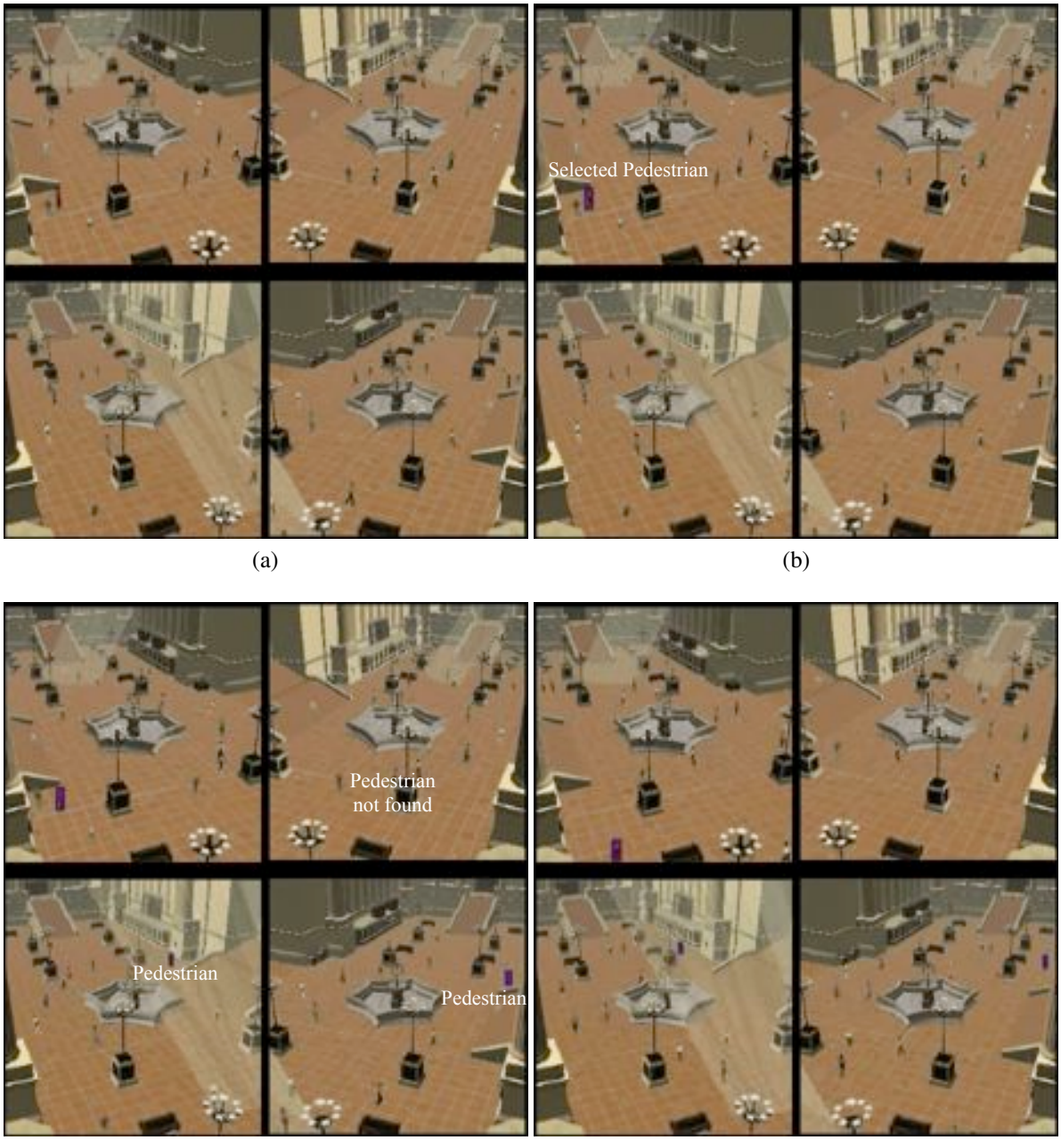


Figure 6.14: Video surveillance panels showing video feeds from four calibrated wide-FOV cameras located at the corners of the waiting room in the virtual train station (counter-clockwise frop top-left is video feed from cameras 1, 3, 4, and 2).

wide-FOV cameras situated at the corners of the waiting room of the virtual train station. The user selects a pedestrian in Camera 1 by drawing a rectangle around it (Top-left feed in Figure 6.14(b)). Camera 1 computes the appearance signature (i.e., color histogram) of the pedestrian and broadcasts it to the neighboring cameras. Cameras 3 and 4 successfully locate the pedestrian using its appearance signature (Figure 6.14(c)). Cameras 1, 3, and 4 form a group and track the pedestrian.

6.4.1 Larger Sensor Network Simulations

We now present several simulations of larger sensor networks outside our virtual vision simulator.

We have tested the sensor network communication model under various conditions with up to 400 sensor nodes. Figure 6.15 shows a sensor network consisting of 400 nodes on a 20 by 20 uniform grid. Each node can communicate with its 4-neighbors. A node can communicate with any other node in the network via multi-hop messaging. The user tasks the bottom-left node to follow the target shown as a cyan cone.[4] The bottom left node elects itself as the leader and begins recruiting other sensor nodes (Figure 6.15(a)). Leader nodes are shown as squares (cubes). The group evolves by recruiting new nodes, dropping member nodes that are no longer relevant, and selecting new leaders in response to the change in target's location (Figure 6.15(b)). When the target splits into two, the group successfully splits into two groups, each with its own leader, to follow the target (Figure 6.15(c)–(e)). It is important to note that group splitting occurs naturally in our protocol and does not require any special handling. Furthermore, the two groups do not need to communicate with each other. As targets converge upon each other, the two groups discover each other and merge demoting one of the leaders (Figure 6.15(f)–(h)). The protocol successfully handled both leader and member node failures and kept evolving to follow the target (Figure 6.15(i)–(l)).

Figure 6.16 shows a sensor network similar to that shown in Figure 6.15. The user tasks

[4] It appears as the cyan circle in Figure 6.15.

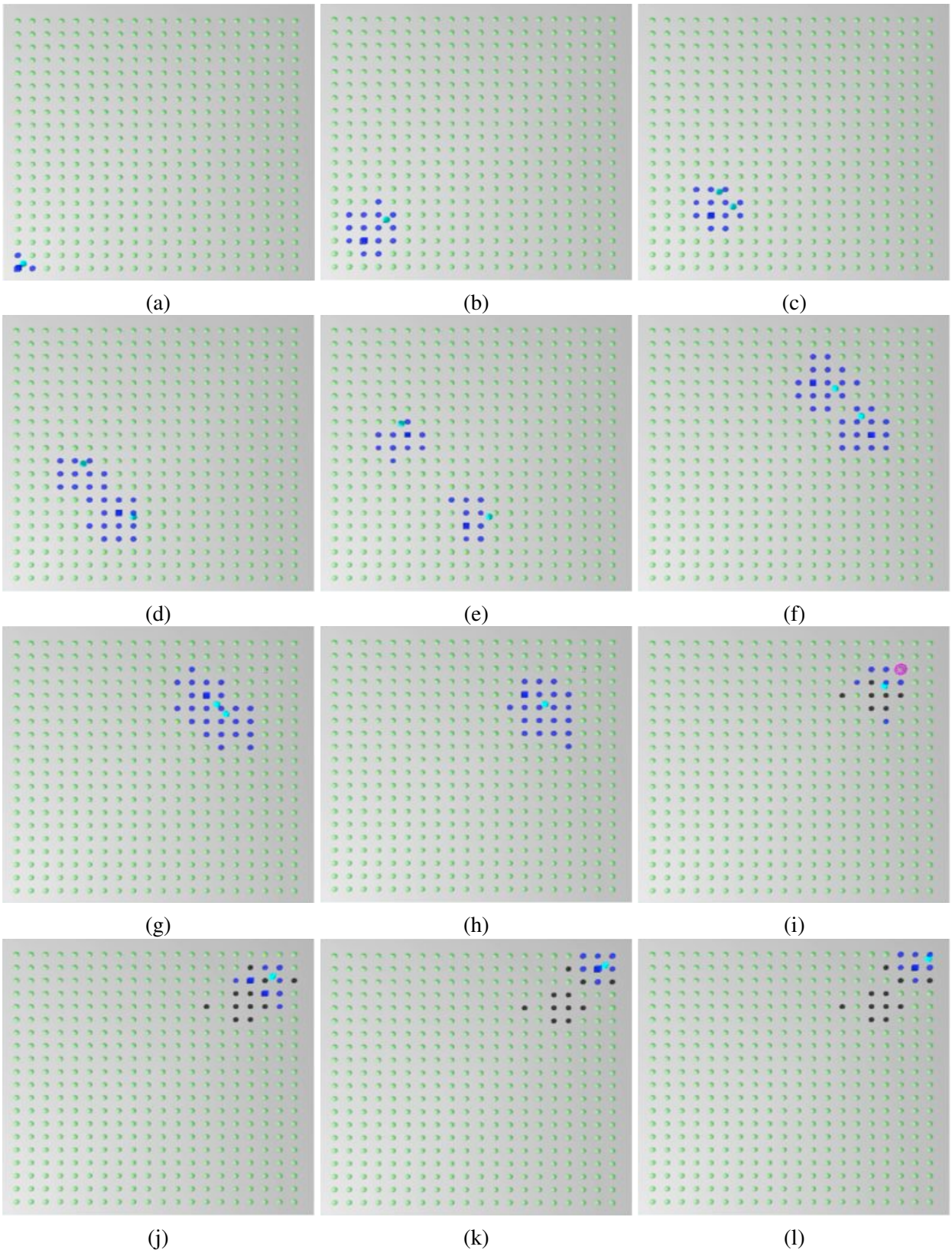
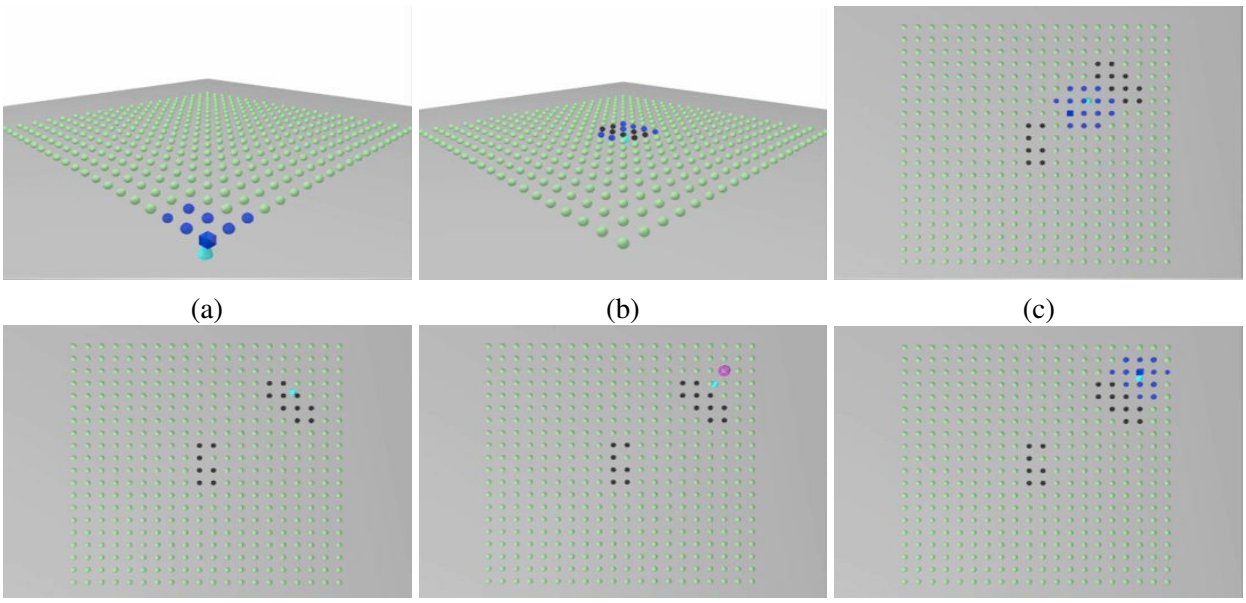


Figure 6.15: Group splitting and merging. Green nodes represent idle sensor nodes, blue nodes represent sensor nodes that are currently engaged in the task of following the target denoted by a cyan cone. The target appears as a cyan circle in the top view of the sensor grid shown here. Square nodes represent group leaders and black nodes represent failed nodes. Each node can communicate with its 4-neighbor.



(a) (b) (c)

(d) (e) (f)

Figure 6.16: Simultaneous node failures.

the bottom-left node to follow the target shown as a cyan cone (Figure 6.16(a)). The network successfully handled node failures shown as black nodes around the center of the image in Figure 6.16(b). These node failures were generated randomly along the path of the target. The group of black nodes in the top right corner of Figure 6.16(c) show coordinated node failures along the path of the target. Our system was not able to handle this error as seen in Figure 6.16(d) where no node is following the target. However, at this stage an operator can retask another node to observe the target Figure 6.16(e)–(f). Simultaneous failures of relevant nodes typically have catastrophic effects, as these sever the communication backbone of the sensor network.

Figure 6.17 shows a sensor network of 50 nodes placed randomly in a 25 square m area. The nodes that are within 5 m of each other can directly communicate with each other. Each node can communicate with another node in the network through multi-hop routing. Figure 6.17(a)–(e) shows group merging. When the leader of the group fails (Figure 6.17(f)), multiple member nodes assume leadership (Figure 6.17(g)). These nodes negotiate with each other to select a single leader (Figure 6.17(h)).

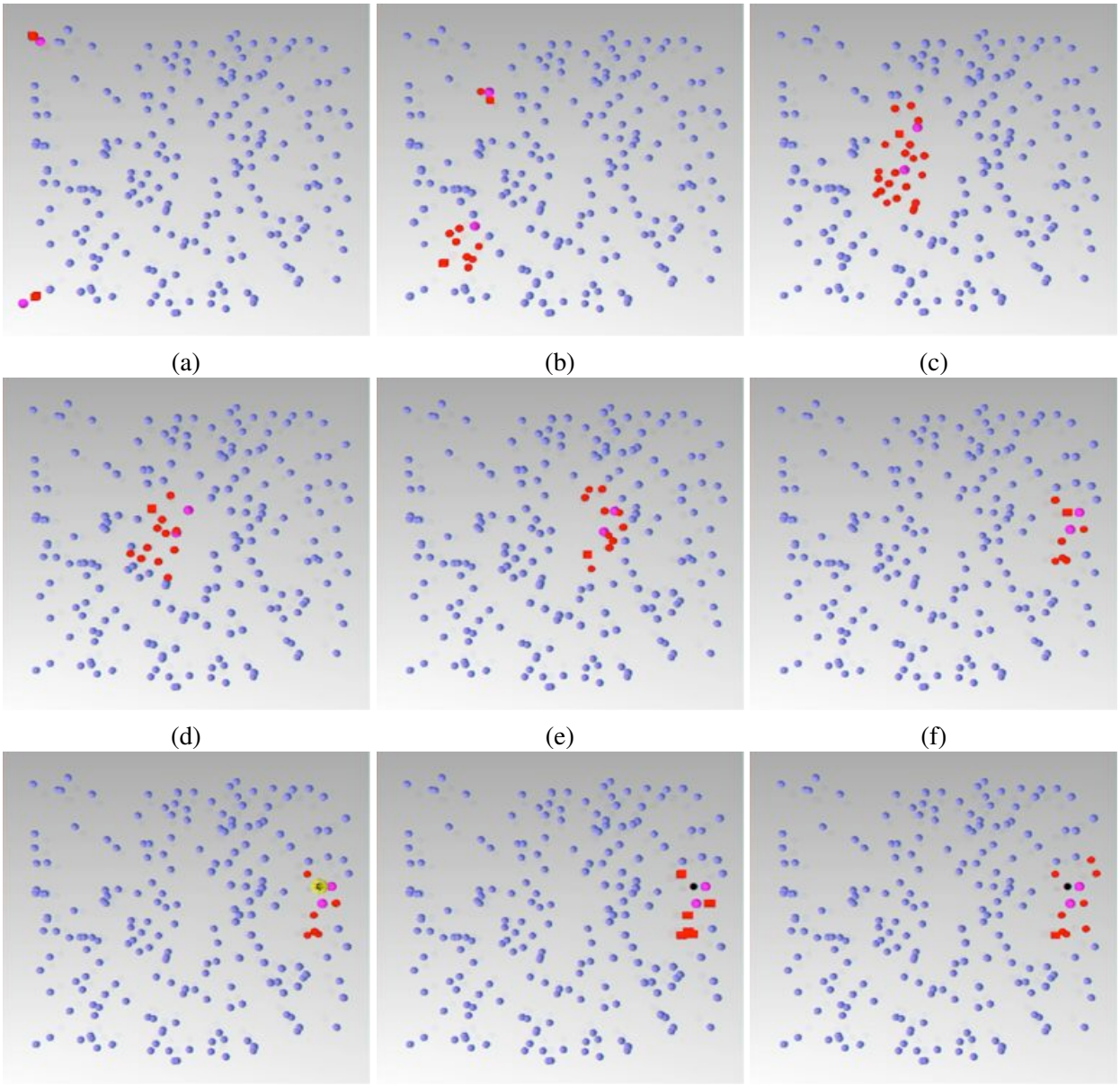


Figure 6.17: Group merging and leader failure. Blue nodes are idle. Red nodes are following the targets. Pink cones represent targets. Square nodes represent group leaders and black nodes indicate node failures.

6.4.2 Discussion

Given the above results, we make the following observations:

- The proposed protocol successfully forms camera groups to carry out various observation tasks. Cameras that belong to a single group collaborate with each other for the purposes of carrying out the observation task. Currently we support a small number of observation tasks that are of interest to the visual surveillance community. These are 1) taking snapshots of a pedestrian, 2) closely observing a pedestrian during his/her stay in the designated region, and 3) following a pedestrian across multiple cameras.

- Camera grouping does not require camera calibration or camera network topology information, which makes our system suitable for ad hoc deployment. This is not to say that the proposed protocol cannot take advantage of camera calibration and/or camera network topology information, if such information were available. For example, a leader node can use this information for targeted announcements of observation tasks to the relevant nodes, thereby reducing bandwidth consumption.

- Camera grouping is strictly a local negotiation between the relevant nodes. It is independent of the total number of nodes in the network, and it does not require a central controller.

- Cameras have a limited field of regard, and as long as the observation tasks are localized, we expect that the cameras that are relevant to an observation task will be close to one another. This suggests that camera groups are spatially local arrangements. It is important to note that groups that are spread over a large area are undesirable as they have much higher maintenance costs (e.g., in terms of communications).

- Camera groups are dynamic and transient arrangements that evolve in order to perform an observation task. Like group formation, group evolution is a negotiation between the relevant nodes.

- Camera handoff occurs naturally during negotiations.

- The proposed protocol can deal with node and message failures. Failed nodes are dropped from the group and failed leaders are replaced by new leaders, who themselves select a single leader through leader demotion (group merging) activity. This suggests that the network protocol can handle addition and removal of camera nodes during the lifetime of an observation task.

- Assuming that each (camera) node is a perfect sensor, the proposed protocol still might fail to carry out an observation task. It can happen when a large fraction of nodes fail or a significant fraction of messages are lost. Large scale catastrophic node failures sever the communication paths in the network and can only be dealt with by replacing failed nodes. We can handle message loss by drawing upon the techniques developed in the computer networks.

- The proposed protocol might also fail to carry out an observation task even when each (camera) node is assumed to be a perfect sensor if the group evolution cannot keep up with a fast changing observation task (e.g., a pedestrian who is walking too fast and who spends little time in the field of regard of any single camera). Group formation and group evolution are not instantaneous processes. Response times for group formation and evolution are intimately tied to the characteristics of the underlying communication system. Longer message delays will result in sluggish overall performance.

- Smaller group sizes are preferable than larger group sizes, as larger group sizes have slower responses. For example, group formation requires at least three messages (*queryrelevance*, *relevance*, and *join*) to be exchanged between two adjacent nodes, more when multiple nodes are involved or if the two nodes are at multi-hop distance from each other. Similarly, group evolution requires multiple messages. The failure of a leader node begins leader demotion (group merging) activity that requires anywhere from a single message for the two node case to hundreds of messages when multiple nodes

are involved (the pathological scenarios shown in Figures 6.6 and 6.7 exchange roughly 1500 messages to form a single leader group comprising 25 nodes). The conflict resolution scheme also requires multiple messages for centralization and sending out the assignments to various nodes.

- The CSP formulation of the conflict resolution strategy can handle multiple interacting groups as long as the number of relevant sensors for any group remains small (fewer than 10). This is in line with our previous observations that small group sizes are preferable. The conflict-resolution-driven sensor assignment strategy proposed in this thesis will be unable to find an optimal sensor assignment when a large number of nodes are involved. Our scheme can find the optimal assignment without enumerating all feasible assignments when the quality of an assignment increases monotonically as more tasks are assigned cameras.

- Camera node aggregation is fully distributed and lacks a central controller, so it is scalable. Sensor assignment in the presence of conflicts, however, is centralized over the involved groups. Our scheme, therefore, lies somewhere between a fully distributed and a fully centralized system. In the interest of scalability, group sizes should be kept small.

Part II

Intelligent Perception for Space Robotics

Chapter 7

Introduction and Motivation

Since the earliest days of the field, computer vision researchers have struggled with the challenge of effectively combining low-level vision with classical artificial intelligence (AI). Some of the earliest work led to robots that combined image analysis and symbolic AI [Roberts 1965; Nilsson 1984]. The vision problem is hard, however, and these early attempts met with limited success; hence, the focus of vision research shifted from developing vertically-integrated vision systems to the development of faster, more robust low-level vision modules. Such modules were combined with behavior-based control mechanisms to create autonomous robotic agents that inhabit and pursue their agendas in the real world [Brooks 1986b; Arkin 1990]. Using their sensors and low-level vision modules, these autonomous agents were able to perceive their surroundings and they are able to react appropriately using their actuators. Unlike the earliest systems, however, behavioral robots did not aspire to be truly intelligent in complex, unpredictable environments, because they could not acquire knowledge about themselves and their environments and exploit this knowledge to reason and plan their actions. Interest began to shift to cognitive robotics [Levesque and Reiter 1998; Giacomo et al. 1998], whose top-down deliberative processes are able to support reasoning and planning. This has resulted in more intelligent autonomous robots, such as MINERVA [Burgard et al. 1999].

There is a pressing need to develop autonomous robots for space robotics. For example,

on-orbit satellite servicing operations are currently carried out manually; i.e., by an astronaut. However, manned missions are usually very costly and there are human safety concerns.[1] Furthermore, it is currently impracticable to carry out manned on-orbit servicing missions for satellites in geosynchronous equatorial orbit (GEO), as the space shuttle cannot reach them. Unmanned, tele-operated, ground-controlled missions are infeasible due to communications delays, intermittence, and limited bandwidth between the ground and the servicer. A viable alternative is to develop the capability of autonomous on-orbit satellite servicing.

A critical first phase of any on-orbit satellite servicing mission, be it for the purpose of refueling, reorbiting, repairing, etc., involves rendezvousing and docking with the satellite. From the perspective of the software responsible for controlling the sensory apparatus and robotic manipulator, the rendezvousing step is the most interesting and challenging. Once the satellite is secured, we can assume a static workspace and handle the remaining steps using more primitive scripted controllers [Gillett et al. 2001]. Most national and international space agencies realize the important role of autonomous rendezvous and docking (AR&D) operations in future space missions and now have technology programs to develop this capability [Wertz and Bell 2003; Gurtuna 2003].

Autonomy entails that the on-board controller be capable of estimating and tracking the pose (position and orientation) of the target satellite and guiding the robotic manipulator as it 1) approaches the satellite, 2) maneuvers itself to get into docking position, and 3) docks with the satellite. The controller should also be able to handle anomalous situations, which might arise during an AR&D operation, without jeopardizing its own safety or that of the satellite. Another requirement that is desirable for space operations is that of *sliding autonomy*, where a human operator can take over the manual operation of the robotic system at any level of the task hierarchy [Sellner et al. 2006; Brookshire et al. 2004]. Sliding autonomy enhances the reliability of a complex operation and it expands the range and complexity of the tasks that a

[1] The Hubble telescope captured the imagination of the public during its highly publicized repair missions, which were carried out by astronauts. By some estimates, these repairs cost taxpayers as much as $12 billion.

robotic system can undertake.

7.1 Vision-based AR&D System and CoCo

In this part of the thesis, we will develop a visually-guided AR&D system and validate it in a realistic laboratory environment that emulates on-orbit lighting conditions and target satellite drift. To our knowledge, ours is the only AR&D system that uses vision as its primary sensory modality and can deal with an uncooperative target satellite. Other AR&D systems either deal with target satellites that communicate with the servicer craft about their heading and pose, or use other sensing aids, such as radar and geostationary position satellite systems [Polites 1998].

Our system features the cognitive controller, called CoCo, a new hybrid robot control framework that combines a behavior-based reactive component and a logic-based deliberative component. CoCo is designed from the ground up, keeping in mind the special needs of a vision-based robot, such as real-time mental state maintenance, and it draws upon prior work in bio-mimetic virtual characters [Tyrrell 1993; Tu and Terzopoulos 1994; Funge et al. 1999] and hybrid reactive/deliberative autonomous robots [Gat 1992; Connell 1992]. Its motivation comes from the fact that humans, who are sophisticated autonomous agents, are able to function in complex environments through a combination of reactive behavior and deliberative reasoning. We demonstrate that CoCo is useful in advanced robotic systems that require or can benefit from highly autonomous operation in unknown, non-static surroundings, especially in space robotics where large distances and communication infrastructure limitations render human teleoperation exceedingly difficult. In a series of realistic laboratory test scenarios, we subject our CoCo AR&D system to anomalous operational events, forcing its deliberative component to modify existing plans in order to achieve mission goals. The AR&D controller demonstrates the capacity to function in important ways in the absence of a human operator.

Our AR&D prototype meets the operational requirements by controlling the visual process and reasoning about the events that occur in orbit. The system functions as follows: First,

captured images are processed to estimate the current position and orientation of the satellite (Figure 1.4). Second, behavior-based perception and memory units use contextual information to construct a symbolic description of the scene. Third, the cognitive module uses knowledge about scene dynamics encoded using the *situation calculus* to construct a scene interpretation. Finally, the cognitive module formulates a plan to achieve the current goal. The scene description constructed in the third step provides a mechanism to verify the findings of the vision system. Its ability to plan enables the system to handle unforeseen situations.

The performance of the system results from the cooperation of its components, including low-level visual routines, short and long-term memory processing, symbolic reasoning, and the servo controllers of the robotic arm used to capture the satellite. Competent, reliable low-level visual routines are essential for meaningful higher-level processing. Consequently, the AR&D system depends upon the reliable operation of the low-level object recognition, tracking, and pose-estimation routines. The AR&D system is able to handle transient errors in the low-level visual routines, such as momentary loss of tracking, by using short-term memory facilities. The high-level routines can also invoke visual search behaviors to re-acquire the target satellite, and the mission can proceed as planned if the target is successfully re-acquired. However, the system cannot accomplish the task when the low-level vision algorithms altogether fail to track the satellite, in which case the high-level routines abort the mission.

Stable servoing routines that account for the manipulator's dynamics are vital for a successful AR&D mission. Hence, the AR&D prototype developed here assumes that the robotic arm can servo competently under the guidance of the higher level modules. Although we have not proved the correctness of the reasoning module, it appears in practice to meet the task requirements—autonomous and safe satellite rendezvous and docking.[2]

The performance of an intelligent vision system is closely tied to the capabilities of its components: low-level visual routines, short- and long-term memory processing, and sym-

[2]Our reasoning modules are GOLOG programs (Section 9.5). Quoting [Levesque et al. 1997], "A GOLOG program macro-expands to a situation calculus sentence, we can prove the properties of this program (termination, correctness, etc.) directly within situation calculus."

bolic reasoning. Reliable low-level visual routines are essential for meaningful higher-level processing. Early attempts at designing high-level vision systems failed because of the lack of competent low-level visual algorithms. Consequently, the cognitive vision system depends upon the reliable operation of the object recognition, tracking, and pose-estimation routines. The cognitive vision system is able to handle short-duration errors in the low-level visual routines, such as momentary loss of tracking, by using short-term memory facilities. However, it cannot accomplish the task when the low-level vision algorithms altogether fail to track the satellite, in which case the high-level routines aborts the mission. The reasoning module meets the task requirements in practice: autonomous and safe satellite rendezvous and docking.

Chapter 8

Related Work

In this chapter, we first briefly review the state of the art in space robotics, then we examine the interplay between vision and robotic control, and we follow that by a detailed examination of the relevant literature in robot control architectures, ethological modeling for virtual characters, and cognitive robotics. We conclude the chapter by differentiating our CoCo architecture from prior robot control architectures.

8.1 Space Robotics

Robots are an invaluable tool for space exploration, where distances are too large and environments too hostile to send humans. Furthermore, robots are also indispensable in manned missions. Before the famous Apollo Moon Missions, Surveyor landers were sent to the moon to send back images and soil analysis data (circa 1966–1968). Surveyor landers were remotely operated from Earth and were equipped with a robotic are to collect soil samples from the moon surface.

In 1970, the U.S.S.R. sent the first space rover, Lunakhod 1, to explore the surface of the moon. Lunakhod was remotely operated by Soviet scientists using onboard television cameras. Upon detecting a possible tip-over, it would automatically stop and wait for a signal from the ground situation room. This is perhaps the first use of onboard autonomy in space robotics.

Remotely operated vehicles are not restricted to space landers (Surveyor) and space rovers (Lunakhod). They also include unmanned deep space probes, such as the Voyager spacecraft. Launched in 1977, Voyager has already travelled 2.8 billion miles! Unmanned deep space probes are continuously monitored and controlled from earth, yet the large distances render real-time control infeasible. Consequently, these vehicles have limited capabilities of autonomous operation.

In addition to remotely operated vehicles, remote manipulator systems have found wide spread used in space missions. Remote manipulator systems are robotic arms, equipped with cameras and grappling mechanisms, that are remotely controlled by earthbound humans or astronauts. To date, robotic arms have been employed in a wide variety of situations, ranging from collecting extraterrestrial soil samples, in-orbit assembly tasks, satellite orbiting and retrieving tasks, and as a positioning and anchoring device for astronauts during Extra Vehicular Activity (EVA) missions. We point the interested reader to [NASA 2005], which presents a historic account of the important milestones in space exploration.

8.1.1 The Need for Increased Autonomy

Currently, nearly every spacecraft is closely monitored and controlled by highly skilled personnel—earthbound operators or astronauts in space. The dominant view is that every aspect, however minor, of a space mission should be controlled by a human operator. Increased autonomy is seen as a means to reduce dependence on human operators. Here, human operators set high-level mission goals, and the spacecraft takes care of low-level details to achieve the goals, handling anomalous situations as they arise. Autonomous spacecraft promise to increase space mission capabilities by many orders of magnitude. While in almost all cases increased autonomy is advantageous as it reduces dependencies on a human operator; in others, it is a necessity. For example, on-orbit satellite servicing is a class of space missions that stands to gain significantly from increased onboard autonomy.

Historically, space agencies have been suspicious of AI technologies, as they threaten to re-

place human judgement in space operations. However, more recently various research projects have been initiated to develop AI technology for space operations. We now briefly survey autonomy-related projects carried out by the space exploration community over the last two decades.

8.1.2 Autonomy Initiatives in Space Exploration

Telerobotics Testbed Project: In the mid 1980s, NASA started the Telerobotics Testbed project. The goal was to research and develop proof-of-concept projects for supervised, automatic satellite servicing [Schenker 1988]. A robotic arm was demonstrated automatically docking with a mockup satellite in a realistic, zero-gravity setting. An array of five cameras estimated the position and orientation of the target satellite and servoed the robotic manipulator to dock with the target. The Telerobotic Testbed was a large research effort, consisting of many disparate modules, and was eventually abandoned in favour of smaller, more focused manipulator teleoperation projects [Backes et al. 1993]. Very little of the Telerobotic Testbed project survives today.

Microrovers for Extraterrestrial Exploration: Around the same time, a number of efforts were underway to design a semi-autonomous mobile rover for extraterrestrial exploration. The desire was to go beyond remotely teleoperated rovers, such as Lunakhod, the Soviet rover that had visited Moon some 15 years earlier. The first Mars Rover, Robby [Wilcox et al. 1992], followed the Sense-Model-Plan-Act (SMPA) paradigm and was soon abandoned in favour of smaller, cheaper mobile robots, called Microrovers [Hayati et al. 1997]. Microrovers are reactive robots capable of semi-autonomous operation, and they turned out to be much more competent than Robby. A microrover, Sojourner, reached Mars on the Pathfinder mission in 1997 [Matijevic and Shirley 1997]. Sojourner is a teleoperated robot capable of way-point navigation and obstacle avoidance. However, it cannot wander very far from its mothership (< 5m), the Pathfinder lander, as the waypoints are selected in the 3D stereo images from the

Pathfinder lander.

Spirit and Opportunity Rovers: The state of the art in space rovers is the Mars exploration rovers, Spirit and Opportunity, that reached Mars in January 2004 [NASA 2004]. Both rovers were sent to sites on opposite sides of Mars to search for answers regarding the history of water on Mars. The initial goal for each rover was to travel 1 kilometer, which has been exceeded by far. These are autonomous robots that are capable of taking pictures, driving around while avoiding collisions, and carrying out scientific experiments in response to commands transmitted from the ground, but they lack any cognitive or reasoning abilities. Unlike the Sojourner rover, Spirit and Opportunity can automatically carry out scientific experiments using their onboard instruments. Each rover is equipped with three monochrome cameras that allow them to see their environments and navigate around obstacles.

On-orbit Satellite Servicing: As argued earlier, the space community is interested in autonomous on-orbit satellite servicing capabilities. During the course of our presence in space, only a tiny fraction of satellites have been serviced in orbit. Currently available teleoperated and manned approaches are either infeasible or too costly for most satellite servicing operations. Consequently, on-orbit satellite servicing has been restricted to some of the most expensive space infrastructure, such as the Hubble Space telescope and the International Space Station. Less expensive satellites, or those that reside in higher orbits, are generally disregarded.

ETS-7 Mission: The ETS-7 mission carried out by the Japanese AeroSpace Exploration Agency (JAXA) is the first on-orbit satellite rendezvous mission that demonstrated the viability of unmanned, automated satellite capture. A chaser satellite (Hikoboshi) equipped with a robotic arm and a target satellite (Orihme) were launched together [Kasai et al. 1999].[1] GPS

[1]Hikoboshi and Orihme is Japanese for "hunter boy" and "weaver girl", two lovers from an old Japanese love story. Separated by the milky way, they are allowed to meet each other only once per year.

receivers (GPSR), rendezvous laser radar (RVR), and a proximity sensor (PXS) are used to estimate the relative position of the two satellites. The on-board Guidance Control Computer (GCC) can handle sensor and communication failures by invoking a collision avoidance behavior that moves the chaser satellite away from the target. Lack of deliberation and reliance on other GPS constellations for estimating the position and orientation of the target renders this scheme brittle. Nevertheless, this is first in-orbit, automated satellite capture demonstration. We refer the reader to [Kasai et al. 1999] for more details.

Teleoperated On-orbit Servicing: Prior to the ETS-7 mission, the focus was on improving teleoperated on-orbit servicing missions via accounting for the communication lag between the ground station and the servicer craft [Fagerer et al. 1994]. Similarly, efforts since ETS-7 have focused on teleoperated on-orbit servicing. We point the reader to [Landzettel et al. 2006] for more details.

Orbital Express: NASA initiated the Orbital Express program to research and develop autonomous on-orbit servicing capabilities [Kennedy 2006]. The work presented in the second part of this thesis was done while working with MDRobotics Ltd. (now MDA Space Missions) in Brampton, Ontario, and it helped Boeing[2] win the Orbital Express program.[3]

8.2 Vision and Control

Computer vision and autonomous robots are among the earliest applications of artificial intelligence. Vision-guided robots has been a major field of research for the last three decades. To operate in its blocks' world, Shakey—the first robot that reasoned about its actions—employed TV cameras in addition to tactile and sonar sensors. Vision, as a sensing modality, has many

[2]Boeing: `www.boeing.com` (Last accessed on 25 January 2007)

[3]News item: `www.mdacorporation.com/news/pr/pr2002051301.html` (Last accessed on 25 January 2007)

desirable properties, including its long range, as well as the fact that it is a passive, non-invasive modality. Consequently, vision has appeared in many mobile robotic systems.

Mobile robotic systems have employed vision for the purposes of visual navigation [Horswill 1995; Pomerleau 1997], vision-based localization and mapping [Karlsson et al. 2005; Se et al. 2001], ego-motion estimation [Yagi et al. 2000], and searching and tracking objects. Generally speaking, mobile robotic systems employ additional sensors to assist with the perception process, such as sonars and laser range-finders. In some cases visual sensing is employed to handle situations that cannot be addressed using, say, sonars. For example, RHINO uses vision to identify noise absorbing obstacles [Buhmann et al. 1995]. Reference [Desouza and Kak 2002] provided a recent survey of vision-based mobile robots.

The work on vision-based mobile robots falls into two broad categories. At one extreme, the focus has been to understand and solve "the vision problem" from the point of view of an active observer; whereas, at the other extreme, vision, which is but one of many sensory modalities, is relegated to a secondary position. PLAYBOT [Dickinson et al. 1993; Tsotsos et al. 1998] and Autonomous Land Vehicle [Dickinson and Davis 1988; Dickinson and Davis 1990; Davis et al. 1992] are notable exceptions; they combine bottom-up, data-driven visual processing with top-down, goal-directed behavior.

As early as the 1970s, vision was used to control robotic manipulators in assembly jobs, inspection tasks, pick-and-place tasks, and material handling jobs, to name a few. Blind robots that are unable to perceive their environments are brittle and they can only operate in highly structured, engineered environments. Consequently, researchers propose equipping manipulators with sensory capabilities—typically tactile or visual—that would allow the robots to perceive their environment and act accordingly. As stated earlier, early work on integrating vision and control followed the SMPA cycle. These initial attempts at integrating vision with control are called a static "Look and Move." Weiss demonstrated that visual feedback can provide closed-loop position control of a robotic manipulator, bypassing the complex scene interpretation stage. This style of control is called *visual servoing* [Hill and Park 1979; Weiss 1984].

Reference [Kragic and Christensen 2002] provides a survey of the visual servoing literature.

Visual servo systems can be classified depending upon the number of cameras and their positions. Single camera systems are cheaper, whereas multi-camera systems can help with the vision problem. Cameras can be mounted on the end-effector to construct what are called *eye-in-hand* systems. Alternately, in *out-of-hand* configurations, cameras observe the robotic manipulator from a third person's perspective. One can also design *hybrid* systems by combining *eye-in-hand* and *out-of-hand* configurations. Our satellite capture system has an *eye-in-hand* stereo camera configuration. Visual servo control structures fall into one of the following three categories:

Image-based 2D image measurements directly drive the servo mechanism. Typical examples include tracking and fixating on objects using uncalibrated cameras. Image-based servo schemes provide long range motor skills for our satellite capture system described in the next chapter.

Position-based The 3D pose of the target relative to the camera/robot is computed through visual analysis and the servo task is defined in the 3D pose space. These are typically used in industrial applications and require calibrated stereo cameras. A Position-based visual servo provides the medium/short range motor skills for our satellite capture system.

2.5D Combine direct image measurements and position estimates to drive the servo mechanisms. See [Malis et al. 1998] for more details.

Activity planning in visual servo systems is typically concerned with robot/camera trajectory planning during the visual servo task [Mezouar and Chaumette 2000; Thuilot et al. 2002]. For example, to ensure that the target remains within the field of view at all times (especially for eye-in-hand configurations). Trajectory planning is also required during pick and place operations. Similarly, trajectory planning is needed to avoid gimbal lock situations and stuck joints.

Vision and control have also crossed paths in the context of active vision research; e.g., view point planning [Madsen and Christensen 1997], gaze holding [Coombs and Brown 1991; Coombs and Brown 1993], controlling attention for vision [Clark and Ferrier 1988], and grasping moving objects [Allen et al. 1993]. Active vision systems typically employ eye-in-hand configurations.

8.3 Robotic Control Architectures

Robots are quintessential autonomous agents of the artificial kind. Building robotic autonomous agents has proven to be a formidable challenge. Regardless of the reliability of the available sensors and effectors, the real world defies complete and accurate modeling, nor can a robot anticipate the effects of all its possible actions on the world. These are fundamental problems whose solution requires more than brute force compute power.

To address these problems, researchers have proposed various robot control architectures. The architectures fall into one of three categories: Deliberative, reactive, or hybrid (i.e., deliberative/reactive). Deliberative controllers maintain an internal world model and use it to plan a course of action. They typically cannot keep pace with the real world, because all but the most trivial planning takes significant time. By contrast, reactive controllers operate at the speed of the environment, exploiting the interaction of many real-time processes. Unfortunately, they cannot accomplish tasks that require any amount of planning. Humans are sophisticated autonomous agents that combine reactivity and deliberation. This observation has led robotics researchers to propose hybrid, deliberative/reactive control architectures. Hybrid architectures appear to hold the greatest promise for building autonomous robots and other autonomous agents with human-like abilities.

8.3.1 Deliberative Agent Architectures

Early attempts at designing autonomous robotic agents employed a sense-model-plan-act (SMPA) architecture with limited success [Fikes and Nilsson 1971]. Here, the agent maintains an explicit model of its goals, abilities, and the world. The sensing module keeps the internal model consistent with the external reality, the planning or reasoning module uses the internal model to plan a course of action, and the execution module executes the plan. These agents do not perform well in the real world, which is hard to model and waits for no one. Their sensing and planning modules fail to keep up, thereby rendering the agent unsafe. Even though newer SMPA architectures support interleaved planning and execution, they too have difficulty attaining the desired performance [Ambros-Ingerson and Steel 1988].

Shakey, built by the Stanford Research Institute in the early 1970s, is the most famous deliberative robot, and it illustrates the strength and weakness of deliberative architectures [Nilsson 1984]. Shakey occupied a simple blocks world where, on a good day, it could formulate and execute over a period of hours, plans involving moving from place to place and pushing blocks to achieve a goal. Shakey's sensors consisted of a camera and a laser range-finder, and its reasoning module comprised a STRIPS-based planner [Fikes and Nilsson 1971]. The planner represents the current world-state as a set of logical statements that specify what is true in the world. It encodes the effect of the actions by maintaining add-delete lists for every action. The add list specifies conditions that become true after executing the action, and the delete list specifies conditions that will cease to be true after the execution of the action. It represents a goal as a desired world state and employs means-end analysis to plan the sequence of actions that will achieve the goal by transforming the current world state into the desired world state. At each instant, Shakey (1) constructs a spatial model of its world through edge processing of camera images and laser-range measurements, (2) updates the internal world model by adding or deleting logical statements, (3) computes the plan, and (4) attempts to execute this plan. Shakey's success was limited to its blocks world. Its SMPA cycle was slow and it would fail in more dynamic settings.

Integrating Planning and Execution

Learning from the shortcomings of the strict sense-model-plan-act paradigm, researchers proposed to interleave sense-model-plan and act cycles. Integrated Planning and Execution Monitoring (IPEM), proposed by Ambros-Ingerson and Steel [1988], implements a control strategy to decide when to plan and when to execute. IPEM is able to detect when the current plan becomes invalid because of a failed execution of an action or due to some unexpected event in the world. It then backtracks to the last decision point that is still valid and replans. If multiple actions are possible, a scheduler is used to arbitrate between them. Even this variation of the sense-model-plan-act paradigm does not yield the desired performance.

Emphasis on Internal Representations

Building and maintaining a detailed internal world model is difficult and time-consuming, and often counter-productive. A detailed internal world model will always lag behind the outside world due to perception delays; this situation worsens with the increasing complexity of the internal world model. Planning systems assume that the world behaves in a predictable way. For example, each action has a well-defined effect on the world. It also assumes that the world will not change significantly during the course of planning, as otherwise the plan will become invalid. Both assumptions do not hold in the real world.

Knowledge-Granularity

There is another, perhaps more fundamental, reason why pure SMPA agents do not perform well in the real world. These agents assume that actions and world states are discrete, which is not true for a world that obeys Newtonian physics. Discrete actions and world states are theoretical constructs that make planning tractable. In the continuous world, however, activities blend into each other, and more than one of them can be active simultaneously, which makes it harder to define precisely the effect of an action/activity.

We require the means to abstract away unnecessary details from the real world and present

a discrete and predictable world to the deliberative module. Reactive systems provide us with such a mechanism.

8.3.2 Reactive Agent Architectures

Animals are natural autonomous agents that are able to make decisions that prolong their survival and achieve their goals by taking into account external factors[4] and internal motivations[5]. To this day no robot can match all the abilities of even simple animals. Even insects, such as ants, that do not appear to have deliberative capabilities, perform better than any existing robot. This observation, combined with a disillusionment with the traditional AI approach, led researchers to explore alternative approaches for robotic agent design. The reactive or behavior-based approach appeared in the 1980s. These robots operate in real-time and can handle the dynamic events of the world better than their deliberative counterparts. They accomplish this by tightly coupling sensing to action through simple processes, called *behaviors*. Every behavior handles its own perception, modeling, and execution needs, and is only responsible for a small portion of the overall behavioral repertoire of the robot. The interaction of these behaviors gives rise to the desired overall behavior, which is often called the *emergent behavior*.

Tight Connection between Stimulus and Action

The earliest instance of a behavior-based agent is the schematic sowbug[6] that was presented by the psychologist Tolman in 1939 [Tolman 1939]. Tolman put forward the theory of purposive behaviorism and claimed that a tight connection exists between stimulus and response. Furthermore, he said that high-level factors, such as motivation, experience, and purpose, affect

[4]Imagine a zebra trying to get away from a lion. It is safe to assume that the zebra's actions are influenced by the proximity of the lion.

[5]A giraffe makes its way to a pond to quench its thirst. In this scenario, the actions of the giraffe are influenced by external factors, such as the location of the pond, and its internal motivation, which includes its thirst.

[6]Tolman's schematic sowbug exhibits phototactic behavior. For an implementation see [Endo and Arkin 2000].

this connection. He called this connection *behavior*.

Complex Behavior Does Not Imply Complex Internal Structure

Complex behavior of an agent does not necessarily indicate a complex internal structure, so it is possible to realize robots having elaborate behaviors using simple components. For example, Walter's [1950; 1951; 1953] mechanical turtles, Elmer and Elsie, displayed complex behavior despite their simple design, which consists of a light sensor, a touch sensor, two vacuum-tube analog computers, a steering motor, and simple behaviors such as *MoveTowardsLight*. The interaction with the world results in an interesting and hard to predict behavior. They appeared curious as they explored their environment in a speculative manner; Walter therefore named them *machina speculatrix*.

The World is its Own Best Model

In spite of these early forays, the behavior-based approach did not catch on in the robotics research community until Brooks introduced the *subsumption* architecture, which demonstrated that the interaction of independent components can produce coherent "intelligence". Brooks claimed that "the world is its own best model," and argued that it can simply be sensed as necessary. He rejected the necessity of a symbolic world model for intelligence. He built many subsumption-based insect-like robots [Brooks 1986b; Brooks 1986c; Brooks 1986a; Brooks 1990b; Brooks 1990a], which could explore their environment, build maps, navigate, and interact with people—feats no previous robot had accomplished. The subsumption architecture is a network of message passing augmented finite state machines (AFSMs) that are hierarchically organized into *layers*. There is no central world model and all layers take care of their own perception and actuation. Inhibition and suppression between layers produce the desired overall behaviors. New behaviors can be added to the behavior repertoire by adding more layers.

Emergent Behavior

Around the same time when Brooks was building reactive robots, other researchers found evidence supporting the basic assumption of the behavior-based approach, i.e., interaction of simple rules can produce elaborate and coherent overall behavior. Reynold's virtual *boids* exhibit elaborate global behavior (flocking) while using simple local rules [Reynolds 1987]. Each boid follows three rules: maintain a safe distance from all nearby boids, move towards their centroid, and match their average velocity. The interaction of these rules produces realistic, life-like flocking, schooling, and herding behaviors[7]. Braitenberg's imaginary *vehicles* [Braitenberg 1984], which consist of simple components, appear to have emotions like love, fear, and logic. Here too, the interplay of simple components among themselves and with the environment gives rise to elaborate behavior. In his book, *The Society of Mind*, Minsky suggests that the human mind is organized as a collection of specialists, and the competition and cooperation of these specialists produce the overall behavior [Minsky 1985]. Ethologists, who have independently reached the same conclusion, propose that the behavior of an animal is an outcome of competition and interaction among many behaviors [Lorenz 1973]. Ethology, especially, has proven useful for designing behavior-based agents, as it provides the necessary blueprint required for identifying and designing behaviors and managing their interaction.

The Role of Deliberation

An argument in favor of behavior-based AI is that most of the time many robots just perform routine tasks, such as recharging their batteries and moving around safely [Agre and Chapman 1987].[8] These tasks require little or no abstract reasoning. Deliberation is only required to perform novel tasks, and most tasks, once learned, can be accomplished using purely reactive means. This might be why deliberative agents have failed to match the success of behavior-based agents.

[7] Reynold's model belongs to the class of individual-based models where local interaction of the members of a population generate the overall behavior for that population.

[8] This also holds for humans.

Implications of the Lack of Internal Representation

Purely reactive systems—Brooks' subsumption architecture, Walter's *machina speculatrix*, Reynold's boids, and Braitenberg's imaginary vehicles—eschews an internal mental model, which suggests that these systems might never scale up to human level intelligence. Tsotsos [1995] cogently argues about the limitations of a strict behavior-based controller—one that does not allow any internal mental model. He shows that strict behavior-based controllers cannot exhibit human-level intelligence. In the strict behavior-based controller, a behavior is a stimulus-action pair, and at each instant, the controller searches for relevant behaviors to activate. Tsotsos shows that unbounded stimulus-action search is NP-complete; thereby showing that strict behavior-based architecture is not enough. Furthermore, he shows that an internal world model, explicit goal representation, and a hierarchy among behaviors are required to reduce the complexity of the stimulus-action search. He also shows that unbounded vision is intractable. This too excludes the possibility of designing a subsumption style agent with human-level capabilities.

Computation Centric View of a Reactive Robotic Agent

Before we turn our attention to ethologically-inspired behavior-based systems, we take a moment to describe Robot Schema (RS), proposed by Lyon and Arbib, which takes a computation centric view of designing a reactive robotic agent [Lyons and Arbib 1989]. It is an attempt to capture the computational characteristics of a purely reactive robotic agent. RS defines a formal process composition algebra for composing simple reactive, parametrized processes. These processes can execute in parallel or in sequence. Recursion is supported and processes can communicate with each other via port-to-port mapping. They have shown RS successfully controlling a kitting robot. Process algebra provided by RS allows one to analytically verify the properties of the reactive module. Still, it is unclear how RS can deliver reactive robotic agents with capabilities similar to those obtained through the ethologically-inspired approaches. Perhaps because, RS is mute on several issues addressed by the ethologically-inspired approaches,

such as process grouping, process arbitration, process composition, attention-based perception, and short-term memory.

Ethologically-Inspired Behavior Based Systems

In the 1990s, several researchers demonstrated impressive examples of ethologically-based autonomous agents (see, e.g., [Tu and Terzopoulos 1994], [Blumberg 1997], and [Arkin et al. 2001]). These attempts showed that an internal world model does not necessarily hamper reactivity. It is, however, important for the internal mental state within the reactive module to be able to operate in real-time, as otherwise it defeats the main purpose of the reactive module: real-time response. One way to do that is to keep the internal mental state simple, i.e., it should only store information relevant to the task at hand. For example, Tu's fishes use an attention-based perception system that filters out irrelevant details [Tu and Terzopoulos 1994]. Attention-based perception systems are biologically plausible, and they have found broad acceptance in the behavior-based AI community. More recent behavior-based architectures allow for a small working memory, or short-term memory.

Tu and Terzopoulos' Artificial Fishes: Tu and Terzopoulos established that it is possible to build complex ethologically-inspired behavior-based agents by developing virtual, artificial fishes that "live" in a physics-based virtual sea. These biomimetic agents exhibit a broad behavioral repertoire, including behaviors such as foraging, predation, schooling, and mating. They are a departure from the strict subsumption style agents, as they include a simple mental state, an attention-based perception mechanism, and an elaborate action-selection scheme. The fishes are driven by motivations, such as fear, and desires, such as hunger and libido, which are recorded in their mental state. The action-selection mechanism, which is implemented as a decision tree, takes into account sensory information, motivations, and desires to choose an appropriate behavior. The strict hierarchy among behaviors, the continuity of mental state, and the constancy in habits provides persistence among behaviors.

Sony's Robot Dog, Aibo: As stated earlier, ethological models provide a basis for the kinds of behaviors we should realize within a robot. A testament to this statement is Sony Corporation's life-like robot dog Aibo [Arkin et al. 2001]. Aibo can operate competently in the real-world. It exhibits interesting behaviors that are commonly associated with a pet dog: it appears to yearn for its master's approval, it gets hungry, it exhibits moodiness, and it likes to play with bright balls (or things that look like bright balls). Aibo's behavior controller implements a subset of the complete ethogram (categorization of behavioral patterns that span the range of an animal) of a dog. Its behavior arbitrator takes into account external stimuli (such as the proximity of food) and internal motivations (the urge to play with a bright ball) and uses a homeostasis regulation scheme to choose appropriate behaviors [Arkin 1988]. For behavior coordination (i.e., to resolve behavior-dither and implement behavior-persistence), it uses Ludlow's [1976] model of lateral inhibition.

Arkin's Schema Architecture: In Arkin's Schema-based architecture, each behavior (called Motor Schemas) uses relevant perceptual features, which are computed by what are called Perceptual Schemas, to compute action vectors that are subsequently combined to produce the net output. Behaviors are asynchronous processes that execute in parallel and their outputs are combined via weighted vector summation. The schema-based approach has been used successfully to develop the reactive component for the Aura architecture that placed first in the first two American Association for Artificial Intelligence (AAAI) mobile robot competitions, 1994 and 1997.

Engineering Concerns for Behavior Based Systems

Designing behavior-based controllers is a tricky engineering endeavour, as we must deal with emergent properties that defy formal modeling. Some of the issues faced when designing behavior-based system are:

- Behavior arbitration and behavior coordination

- Behavior preference unification
- Behavior dither avoidance
- Level of interest modeling to avoid behavior lock out
- Transitions between behaviors

Behavior Arbitration and Behavior Coordination: Managing behavior interaction to produce the desired overall behavior is quite challenging, as there are possibilities of dead-locks, race-conditions, and the ubiquitous behavior-dither. Behavior interaction involves both behavior selection and behavior preferences unification.

A *behavior arbitration* scheme selects appropriate behaviors given the current sensory inputs and internal motivations. It can be implemented in a single specialized behavior arbitration module, or it can be the consequence of the design of individual behaviors. In the first approach, the behavior arbitrator takes into account the sensory inputs and internal motivations and activates appropriate behaviors. Here, the design of individual behaviors is simpler; however, the behavior arbitrator becomes increasingly complex with the addition of new behaviors, which can adversely affect the real-time performance of the robot. In the second approach, all behaviors compute their own "relevance" (in a predetermined common currency) to the current situation. Here, adding a new behavior is easier, as it does not effect other behaviors; however, the new behavior must be able to compute its relevance in the common currency. Hierarchy among behaviors [Tyrrell 1993; Tu and Terzopoulos 1994] and behavior-groups [Minsky 1985; Blumberg 1994] simplify behavior interactions, thus making behavior-arbitration easier. Another commonly used scheme for behavior arbitration is the "homeostasis regulation rule." Here, a number of internal (usually motivational) variables along with their allowable ranges are specified, and the behavior arbitrator selects behaviors to keep the values of interval variables within the allowable ranges [Arkin 1988].

Behavior Preference Unification: In the subsumption architecture only one behavior issues preferences for the motors (for example, either the *GotoLoc* or the *AvoidCollision* behavior can issue motor commands at any instant), so it does not fully utilize the interaction of multiple behaviors.

Payton remedied this problem by allowing multiple behaviors to issue preferences for a common control [Payton 1990]. The net preference is then the sum of the active behaviors' preferences weighted by their respective relevances. Tyrell modifies Payton's approach, and proposed that preferences for a given control should be calculated as a weighted average of the highest single preference and the sum of all preferences for that control [Tyrrell 1993]. Combining behavior preferences is especially easy in Arkin's motor-scheme framework [Arkin 1998]. Here, a behavior is a mapping from sensor vectors to motor vectors. Behavior preferences, which are vectors, are combined through simple vector addition. We need to keep in mind that in all of the strategies for combining behavior preferences, the most difficult element is to weigh different preferences correctly. Unfortunately, there are no fixed rules/algorithms to resolve this issue, and the designers have to rely upon their intuition and experience to come up with an appropriate set of weights that result in the desired overall behavior.

Behavior Dither Avoidance: Persistence among behaviors is necessary for a coherent and useful overall behavior; a robot that constantly alternates between two goals appears undecided, and it might never achieve either goal. Behavior-dither is a consequence of a poor behavior-arbitration scheme—one that swings back and forth between different behavior choices. The careful use of internal mental states can avoid this undesirable situation. For example, in the case of Tu's artificial fishes [Tu and Terzopoulos 1994], the continuity of mental state and the constancy in habits provides persistence among behaviors. Another approach is that the currently active behavior inhibits a non-active behavior by a factor that varies inversely as the duration of the active behavior. For example, behavior A becomes active at time t_0, then at time t, where $t > t_0$, the inhibition for behavior B will be proportional to $1/f(t - t_0)$. This

simple scheme ensures that once activated, a behavior will remain active for some minimum duration.

Level of Interest Modeling to Avoid Behavior Lock Out: A related problem is how to avoid one behavior from taking over and never relinquishing control. This is a common occurrence in situations where the robot forever tries to achieve some goal. The solution here is to associate a boredom factor with the active behavior that increases with its duration. The boredom factor decreases the relevance of the active behavior over time, so even if some goal is not achieved, the behavior-arbitrator will activate another behavior. This way, a behavior cannot be active for more than some maximum duration.

Transitions Between Behaviors: Transitions between behaviors require extra care, as the controller may exhibit choppy, or worse, incorrect overall behavior. This is especially true for behaviors that share resources, such as motors, which is a common situation for low-level behaviors. One solution is to force the active behavior to return the shared resource in a consistent state before shutting off [Blumberg 1997], which can be achieved through "action buffering" [Perlin and Goldberg 1996]. Another, more involved scheme is to implement fuzzy transitions where the state of the shared resource depends upon both the old and new behaviors. See [Minsky 1985] and [Blumberg 1997] for further details.

Summary

To summarize, behavior-based agents can survive in a complex and unpredictable environment for prolonged periods without human intervention, but they cannot reason about the goals which are implicit in their designs. Behavioral agents cannot "think" ahead and perform tasks that require deliberation, such as path planning. Also, they are not easily amenable to formal analysis, which is a big concern in safety-critical applications. It appears unlikely that a purely behavior-based agent can ever demonstrate human-level capabilities.

8.3.3 Combining Reactive and Deliberative Agent Architectures

Purely deliberative and purely reactive architectures are limited in their own ways, and architectures that support reactivity and deliberation promise greater utility, reliability, and performance. Two approaches have emerged over the last 20 years to support reactivity and deliberation in robotic agents. The first approach takes a unified view of reactivity and deliberation and provides for both using the same computational structures, whereas the second approach relies on specialized reactive and deliberative modules. These controllers are sometimes referred to as hybrid controllers. The unified approaches are scientifically elegant, yet in practice they have not been as successful as their hybrid counterparts.

Unified Approaches to Robotic Agent Design

Soar: Soar [Laird and Rosenbloom 1990] is a cognitive architecture that combines deliberation and reactivity. It has been used successfully to control virtual characters in simulated worlds[9]; however, it has only met with limited success in the domain of real robots. Soar makes no distinction between deliberative and reactive activities. It stores all knowledge as *productions*, and its problem-solving cycle consists of activating the relevant productions till it reaches *quiescence*. Soar embeds deliberative activity within the productions (some production may lead to planning), so reaching quiescence can take an arbitrarily long time, which rules out reactive performance. Soar performs deliberation whenever it fails to reach quiescence and it saves the result of deliberation as a production, thus avoiding deliberation in similar situations in the future. This built-in learning mechanism is called *chunking*, and it increases the ability of the agent over time. This, however, has an undesirable side-effect: Now Soar has to consider more productions, so its performance decreases.

[9]SOAR has successfully controlled Quakebot—the main character in the popular first-person shooter game Quake by Id Software. SOAR is also used to design a realistic aerial war simulation.

Spreading Activation Networks: Spreading Activation Networks is another agent architecture that supports distributed goal-directed behavior [Maes 1990]. Maes demonstrated her system performing STRIPS-like deliberation. The lack of internal representation would give rise to loops during the action selection process. However, she dismisses this concern and argues that a changing world would overcome such an impasse. Maes' approach combines behaviors to support deliberative activity.

Hierarchical Abstract Behavior Network Architecture: Mataric constructed Toto, a subsumption style office wandering robot capable of automatic map construction through landmark detection and map-based navigation, and showed that it is indeed possible to maintain internal representations in a purely behavior-based system [Mataric 1992]. Distributed deliberation emerges through spreading activation in the behavior network. More recently Mataric and Nicolescu have proposed Abstract Behavior Networks that supports reactivity and deliberation [Nicolescu and Mataric 2002; Monica Nicolescu 2001]. These can be seen as augmented spreading activation networks where each primitive behavior is paired with another concurrent process called Abstract Behavior. Abstract behaviors, which can activate or inhibit their corresponding primitive behaviors, are connected to each other via activation/inhibition/task-completion links. STRIPS-like deliberation is realized within the network of abstract behaviors.

The S* Proposal: Tsotsos addresses the shortcomings of the subsumption architecture in the S* framework [Tsotsos 1997; Rotenstein 2003]. S* allows an agent to have an internal mental model, and it generalizes the notion of a behavior by adding modeling and planning components to each behavior. The behaviors can read from sensors (i.e., the real world) or from the internal world model. Similarly, the behaviors can write to actuators or to the internal world model. S* provides general guidelines about how to design a controller, but its main contribution is to point out the importance of perceptual attention, internal models, and explicit goal representations when designing an intelligent agent.

Limitations of the Unified Approaches: Rotenstien argues that these approaches suffer from the Scaling Problem. Maintaining internal representations via behavior states suggests that the number of behaviors is unbounded, which most likely renders these schemes impractical. Additionally, these schemes support only minimal deliberation capabilities. Deliberation occurs when a situation is encountered in the world, and no decision-making is done about future actions.

Hybrid Controllers

Hybrid controllers treat deliberation and reactivity as two separate activities that require different computational mechanisms. In practice, these controllers have shown the most promise. This is perhaps because they can take full advantage of specialized deliberative (e.g., a symbolic planner) and reactive (e.g., a behavior-based controller) computational structures. The common theme among hybrid architectures is that the reactive component deals with the continuous, detailed, and uncertain aspects of the world, and that it mediates between the world and the deliberative component. The deliberative component experiences the outside world as a well-behaved, discrete phenomenon—actions are discrete and have well-defined effects, and world states, which are represented by a set of symbols (fluents), are akin to snapshots in time. It maintains a detailed mental model of the agent and its environment and it uses this model to form strategies that will achieve the long-term goals of the agent. We point the reader to [Kortenkamp et al. 1998] that present a survey of hybrid architectures used for mobile robots.

Constructing Abstract Actions: We begin by discussing the relationship between discrete actions and continuous activities. A deliberative module (planner) constructs plans, which are usually construed as lists of actions. The quality of a planner (i.e., its efficiency and the reliability of its plans) depends upon the actions that are available to it. These actions are called *primitive actions*, and they should have well-defined effects and unambiguous success and failure conditions. In practice, planners fare better when dealing with more abstract primitive

actions. Abstract primitive actions hide much of the complexity of the world, so planning takes less time.[10] Moreover, they can also make a plan more robust to the changes in the world by handling "routine" error situations (at run-time) that would otherwise invalidate the plan.[11]

As stated earlier, primitive actions do not exist in the real-world; however, we can define them in terms of activities, which are real-world equivalents of actions. The associated difficulty, which is considerable to begin with, is greater for more abstract primitive actions. The upside is that the reactive approach is well-suited for constructing these actions correctly and robustly.

¿From the reactive module's point of view, primitive actions are just high-level behaviors. In simplest terms, the actions of the deliberative module are tied to appropriate behaviors in the reactive module. This connection forms the basis of the interface between the two modules with vastly different properties and structures. For any hybrid architecture, this interface is a crucial design decision that affects both its ability and its performance. Too much reliance on the deliberative module renders it incapable of handling time-critical situations, whereas a bloated reactive module makes it fragile, as reactive modules tend to get stuck in "local minima."

Reactive Action Package: Firby's Reactive Action Package (RAP) framework provides an elegant scheme for constructing abstract primitive actions on top of a behavior-based reactive module [Firby 1989; Firby 1992; Firby 1994]. It consists of two components: (1) a library of RAPs and (2) an execution module. Each primitive action is encoded as a RAP, which is a control structure that can encode multiple situation-dependent execution, error-handling, and termination-evaluation strategies for that action by specifying the interaction of various activities. The execution of a RAP (and hence the associated primitive action) is handled by the RAP executor, which chooses the most appropriate method for execution at run-time. This

[10] When dealing with low-level actions, the search-space for possible plans is larger.

[11] In general, low-level primitive actions fail, and as a result, invalidate the plan more often than their more abstract counterparts.

situation-dependent execution paradigm makes a primitive action more robust to the changes in the environment than is otherwise possible. Moreover, it removes the burden from the deliberative module of handling every contingency in the environment, so the deliberative module can construct sketchy plans—plans that constitute high-level actions—and let the actions figure out the details at run-time. The ability of describing a RAP in terms of other existing RAPs is especially powerful, and it makes this framework ideally suited for describing hierarchical task networks.

Firby [1992] mentions how the RAP framework can provide the necessary interface between the deliberative and reactive modules, but he does not describe any specific planner. He views RAPs as pre-coded hierarchical plans that can be expanded into subgoals, other RAPs, and activities at runtime. Within this view, the executor activates the RAP that will achieve the current goal of the robot, and the RAP remains active until its success (the goals for this RAP have been achieved) or failure (the goals for this RAP can never be achieved) conditions are met. No on-the-fly planning is performed, which severely limits a RAP-based robot's ability to handle novel situations. Gat [1992] addresses this limitation and shows how RAPs can connect a classical AI planner with a control theory based reactive module.

Atlantis: A Three-Tiered Architecture: Gat [1992] proposes a three-tiered control architecture called ATLANTIS[12] that consists of three heterogeneous, asynchronous modules: controller, sequencer, and deliberator. The controller uses classical control theory techniques to implement motor controls, such as *MoveForward*, and low-level behaviors, such as *FollowPath*. The sequencer is a RAP module that interfaces the other two modules. Finally, the deliberator consists of a classical AI planner, and is responsible for planning to perform high-level tasks of the robot. A noteworthy feature of ATLANTIS is the relationship between the deliberator and the sequencer. It follows Agre and Chapman's [1987] theory of plans-as-communications, and the deliberator module merely advises the sequencer. This proves

[12] ATLANTIS stands for "A Three-Layer Architecture for Navigating Through Intricate Situations."

to be a powerful mechanism for combining reactivity and deliberation, as it allows deliberation without affecting reactivity—a feature previously missing from hybrid architectures. An ATLANTIS-based robot can boast reactive performance similar to that of a subsumption-based robot and deliberative capabilities similar to those of Shakey.

Servo, Subsumption, Symbolic Architecture: Connell's [1992] SSS, which stands for "servo, subsumption, symbolic," architecture combines a servo-control layer, a subsumption layer, and a symbolic layer. The lower-most layer implements servo-loops for various actuators. The middle layer is a subsumption style controller, which consists of behaviors that handle "events of interests" by choosing appropriate *setpoints* for the servo-loops, such as the desired speed for the wheel-speed servo-loop. The symbolic layer maintains a coarse geometric map of the world, which is used to plan a route between the source and destination. To traverse this route, the symbolic system enables/disables appropriate behaviors present in the subsumption layer before and after each segment of the route. The subsumption layer has considerable freedom for following the current segment; however, it requires immediate response from the symbolic system for the events that it cannot handle (for example, the end of the current segment, or if it is stuck). This imposes an undesirable constraint on the symbolic layer, which now must perform in real-time. The symbolic layer accomplishes this by using special-purpose data structures called *contingency tables*, which contain pre-compiled responses to various events that the symbolic layer must handle. Here, the decoupling between the symbolic layer and the subsumption layer is insufficient; a better approach is to use the deliberative layer only as an adviser.

Autonomous Robot Architecture: Autonomous Robot Architecture (AuRA) is a hybrid architecture for autonomous robots capable of both deliberative planning and reactive control [Arkin 1998; Arkin 1992; Arkin 1990]. AuRA principles have been applied in a variety of domains like navigation, robot competitions, multi-robot teams, and vacuuming. A multi-agent implementation of AuRA won the 1994 robotic competition. Aura consists of four modules:

a mission planner, a spatial reasoner, a plan sequencer, and a schema-based reactive module. The mission planner sets high-level goals for the robot, and the spatial reasoner constructs plans that will achieve those goals. The plan sequencer executes these plans over the reactive module by invoking appropriate behaviors. Aura does not have true deliberative capabilities; i.e., it does not have a planning module. Aura's plans, which are represented as Finite State Acceptors (FSA), are hand-coded. Each state represents a specific combination of behaviors that accomplish one step of the task (i.e., one path leg). The reactive module is a schema-based control system.

Procedural Reasoning System: Georgeff and Lansky [1987] propose the Procedural Reasoning System (PRS) for combining reactivity and deliberation. A notable feature of the PRS approach is that, unlike hybrid architectures described so far, reactive processes control deliberative activity. It subscribes to the view of planning as a *least commitment strategy* and tightly couples perception to planning. The robot is considered a "rational" agent endowed with psychological attitudes of belief, desire, and intention. Beliefs represent the informational state of the agent, desires (or goals) encode the motivational state of the agent, and intentions represent an agent's commitments (i.e., what the agent has "chosen to do"). The system alters its plans on-the-fly in response to a change in its beliefs, which suggests that the system is both reactive and deliberative; however, the response of the system depends upon the speed of the deliberative mechanism.

PLAYBOT: PLAYBOT is a visually-guided robot that helps physically disabled children in play [Tsotsos et al. 1998]. PLAYBOT uses its robotic arm to interact with the environment, e.g., picking and dropping objects. Its vision system [Dickinson et al. 1993], which combines reactive behavior and planning, is capable of performing complex visual tasks, such as object search, recognition, and localization. The vision system consists of two layers that run independently of each other. The lower layer, which consists of reactive behaviors, is always active and performs object recognition. It continuously extracts information about the world from the

images and uses this information to construct and update the internal world model. The top layer consists of a planner that maintains the high-level goals of the vision system. It uses the world model to reason about the task at hand and directs the recognition layer. For example, it can direct the recognition layer to look in a particular direction, to look for a particular object, or to look for objects in a particular location, etc. In addition, it can help the recognition layer in the task of recognizing objects; e.g., by suggesting a better viewpoint so as to disambiguate an object. Apart from showing how perception systems capable of complex visual tasks can be realized by combining reactivity and deliberation, this work proposes a straightforward scheme of injecting high-level advice into the reactive modules; the planner controls the recognition layer through its state variables. It also furthers the idea that hybrid architectures should implement an adviser-client relationship between the deliberative and reactive module.

ARK: Autonomous Robot for a Known Environment: The ARK project has designed a series of autonomous mobile robots capable of safely navigating within industrial environments. These robots employ a sensor called *Laser Eye* that combines vision and laser ranging for navigation and self-localization [Jasiobedzki 1993]. Indoor navigation and localization is accomplished by laser ranging, while navigation in open areas is carried out by visually detecting landmarks. The control architecture consists of two modules. The high-level module is responsible for planning robot actions, path planning, selecting landmarks for sighting, and user interactions. The low-level is a subsumption style reactive module that implements motion commands, such as go-straight and turn-left. To ensure that the robot moves around safely, the reactive module implements a collision avoidance behavior that detects obstacles and navigate around them. ARK project illustrates how vision can be used to navigate a robot in an unstructured environment, thereby showing that vision is indeed a viable sensory modality for physical robots.

RHINO and MINERVA: Cognitive Robots with Subsumption Style Reactive Modules: RHINO and MINERVA, two recent examples of robots that combine deliberation and reac-

tivity, have garnered kudos. Both have gained popularity as museum tour-guides. RHINO conducted tours at Deutsches Museum Bonn (Germany) in mid-1997 [Burgard et al. 1999]. It successfully interacted with visitors, planned tours, and navigated at high-speeds through dense crowds. Its control architecture consists of two modules: (1) a high-level module that uses symbolic logic to perform planning, and (2) a low-level module that takes care of the reactive needs of the robot (i.e., sensing, motor-skills, and low-level behaviors). The reactive module uses probabilistic techniques to perform localization (determining the current position of the robot in the world) and mapping (the current location of the obstacles).[13] GOLEX, or GOLOG executor [Hhnel et al. 1998; Levesque et al. 1997], connects the two modules. Similar to RAP, GOLEX can expand each primitive action of a linear plan returned by GOLOG into a pre-specified sequence of commands to be executed by the reactive module. GOLEX can also expand GOLOG actions into pre-specified conditional plans, and it has limited capability of handling such plans. It also monitors the execution of these plans, and can request the top-level module to replan upon failure.

MINERVA, which conducted tours at Smithosonian's National Museum of American History in Washington (USA), is more capable than RHINO [Thrun et al. 2000]. Unlike RHINO, it can learn maps from scratch, and within these maps, it uses a *coastal planner* algorithm for path-planning. Such paths take into account the amount of information the robot is expected to receive at different locations in the environment, which helps in localization. In addition to occupancy maps, it uses ceiling-mosaic maps to perform localization. MINERVA uses facial expression, gaze direction, and voice features to interact with people. This interaction depends upon, and conveys, its current "emotional state," or mood, which ranges from happy to angry. A four state stochastic finite state automaton controls its mood by taking into account several external factors, such as proximity of people and whether people are in the way. In MINERVA, the high-level controller is developed using RPL, and is able to compose tours on-the-fly (for

[13] RHINO's uses modified *Markov Localization* method to find the current pose of the robot, and uses *occupancy grid algorithm* to find the locations of nearby obstacles.

example, it can choose not to visit all the exhibits if time is running out, say, because it spent more time than he expected on one exhibit). GOLEX reactively executes the plans returned from the high-level controller.

Intelligent, Autonomous Virtual Characters

Duffy the Merman: Funge et al. [1999] combine deliberation and reactivity to construct quasi-intelligent autonomous virtual characters. These characters inhabit complex virtual worlds, such as the undersea world of Tu's fishes and the prehistoric world that is inhabited by a Tyrannosaurus and Velociprators. They perform high-level tasks using their deliberative/cognitive abilities. For example, in one of the scenarios, the merman Duffy evades a shark using his superior intelligence even though the shark is bigger and faster than him.

Here, the deliberative module manages the knowledge that a character possesses, such as the position of the nearest rock, and it uses this knowledge to plan. It uses interval valued epistemic fluents to represent the uncertainty about this knowledge, and when this uncertainty increases beyond a certain threshold, it can plan to (1) decrease the uncertainty through sensing and (2) accomplish the task using the newly acquired knowledge. The controller can automatically initiate re-planning when the current knowledge becomes out-dated. Unlike the hybrid architectures discussed so far, here the deliberative module constructs *partial* plans (it only plans a limited number of moves in the future). Funge proposes the Cognitive Modeling Language (CML) and uses it to implement the deliberative module. CML can be seen as GOLOG on steroids; it provides GOLOG-like syntax, structures, and facilities. One difference is that it compiles the high-level controller into the C language. The reactive module implements high-level behaviors using the scheme proposed by Tu and Terzopoulos [1994]. For example, the reactive module of Duffy implements the behaviors *following* and *evading*. The reactive module is competent on its own and prevents the character from doing anything stupid in the absence of advice from the deliberative module; an adviser-student relationship exists between the deliberative and reactive modules.

Funge's implementation is geared towards graphics and animation applications and cannot be used to control a physical robot: 1) it does not use real-time control structures, and 2) the interface between the deliberative and reactive modules is ill-defined—it appears as if deliberative activity is embedded within the reactive control-loop[14]. It has nevertheless shown how a deliberative/cognitive module on top of an ethologically-inspired behavior-based substrate can yield powerful high-level controllers for autonomous characters (and robots). It also serves another purpose; it shows that simulated environments can prove useful for designing, implementing, and testing controllers for physical robots, as a designer can focus on the design of the controller itself rather than being distracted and preoccupied by hardware issues.[15]

8.3.4 Summary

Hybrid control architectures seem to be the most suitable for designing autonomous robots. It also appears that controllers that consist of heterogeneous modules fare better than homogeneous hybrid controllers. Furthermore, the adviser-student relationship between the deliberative and reactive modules yields the best results.

The success of Sony's Aibo robot has clearly shown that it is possible to construct a life-like robot using purely reactive (or behavior-based) control strategies. Reactive robot controllers model biological processes, such as low-level motor facilities, perception, attention, memory, and emotion, whose combination comprises a reactive module. The biological sub-field of ethology provides the necessary guidance for designing reactive robot controllers. Although behavior-based robots can operate in the real world for long periods without human intervention, they are hard to formalize. Researchers have yet to devise a mathematical theory of reactive systems. Reactive systems exhibit emergent functionality that is hard to model formally, making it difficult to come up with a theory within which one can prove properties about them.

In the absence of such a theory, one can only study and test these systems empirically.

[14]These are limitations of the implementation, and not of the theory itself.

[15]We are not saying that hardware issues are irrelevant, only that high-fidelity simulation environments are useful for designing and testing physical devices.

One cannot prove that a reactive controller is "safe"; i.e., there are no guarantees that it will perform its mission satisfactorily. Moreover, reactive systems are inherently limited because they are incapable of deliberative planning. Unlike reactive controllers, deliberative controllers are amenable to formal analysis, so we can prove a priori that a deliberative module is correct. However, experience has shown that deliberative controllers perform poorly in real-time situations.

Despite initial speculations in the domain of behavior-based robotics that purely reactive controllers are sufficient, it is now becoming clear that they are not, and that control architectures that combine reactive and deliberative strategies fare better. For example, behavior-based systems are inherently myopic; therefore, they tend to get stuck in local minima. A solution is to endow the robot with deliberative abilities such that it can reason about its goals and guide its reactive module, which provides the necessary interface between the deliberative module and the world, towards achieving them.

Existing deliberative modules are designed either as finite state machines, which represent pre-coded plans, or as a single monolithic planner. The plans that are computed offline are in general less robust to unanticipated situations; however, they can boost the performance of an agent, as now the agent need not waste any time computing the plan. Perhaps a better approach is to combine these two strategies within a single framework. Here, the agent will have access to a suite of pre-coded plans to handle the frequent situations encountered by the agent. The agent will also have a planning module that can jump in whenever none of the pre-coded plans can handle the current situation satisfactorily.

The interface between the two modules is a critical component that determines the overall performance of the controller. It must allow the deliberative module to advise the reactive module without affecting the real-time properties of the agent. Reactivity and deliberation have different characteristics and hence require different computational structures, so it is not surprising that heterogeneous hybrid architectures fare better than those that use the same computational mechanism for reactivity and deliberation.

Thus, our hypothesis is as follows: A heterogeneous hybrid architecture that implements an adviser-client relationship between the deliberative and reactive modules is well suited for designing intelligent controllers for autonomous robotic agents.

8.4 Comparison of the CoCo Architecture

Like ATLANTIS, CoCo consists of both deliberative and reactive modules, featuring a reactive module that performs competently on its own and a deliberative module that guides the reactive module. CoCo was originally inspired by experience implementing self-animating graphical characters for use in the entertainment industry. In particular, our approach was motivated by the "virtual merman" of Funge et al. [1999], which augments a purely behavioral control substrate [Terzopoulos et al. 1994] with a logic-based deliberative layer employing the situation calculus and interval arithmetic in order to reason about discrete and continuous quantities and plan in highly dynamic environments.

CoCo differs in the following ways: First, its deliberative module can support multiple specialized planners such that deliberative, goal-achieving behavior results from the cooperation between more than one planner. The ability to support multiple planners makes CoCo truly taskable. Second, CoCo features a powerful and non-intrusive scheme for combining deliberation and reactivity, which heeds advice from the deliberative module only when it is safe to do so. Here, the deliberative module advises the reactive module through a set of motivational variables. Third, the reactive module presents the deliberative module with a tractable, appropriately-abstracted interpretation of the real world. The reactive module constructs and maintains the abstracted world state in real-time using contextual and temporal information.

Chapter 9

Autonomous Satellite Rendezvous and Docking

In this chapter, we present our research in the domain of space robotics. In particular, we design a visually guided robotic system capable of autonomously performing the challenging task of capturing a non-cooperative, free-flying satellite for the purposes of on-orbit satellite servicing. Our innovative system features object recognition and tracking combined with high-level symbolic reasoning within a hybrid deliberative/reactive computational framework, called the *Cognitive Controller* (CoCo).

The chapter is organized as follows: In the next section we present the CoCo framework. Section 9.3 describes the satellite recognition and tracking module. Section 9.2 explains the visual servo behaviors for the task of satellite capturing. We explain the reactive module in Section 9.4. Section 9.5 and Section 9.6 describe the deliberative and the plan execution and monitoring modules, respectively. Section 9.7 describes the physical setup and presents results.

9.1 The CoCo Control Framework

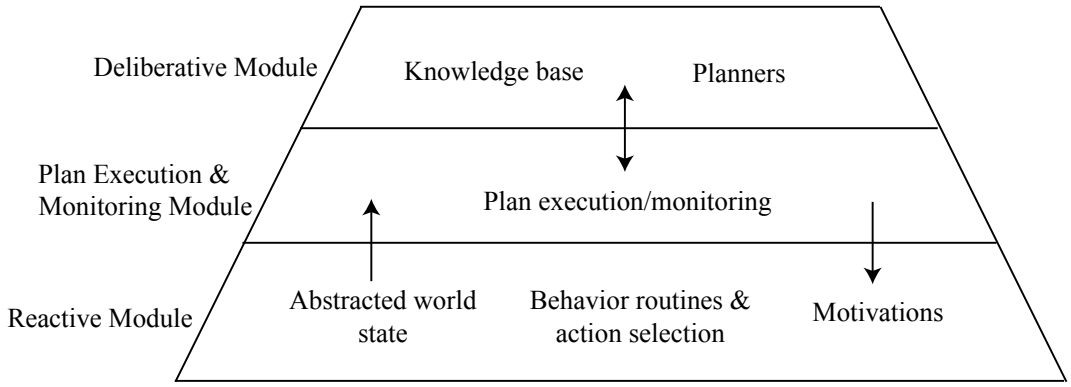


Figure 9.1: The CoCo three-tiered architecture.

CoCo is a three-tiered control framework that consists of deliberative, reactive, and plan execution and monitoring modules (Figure 9.1). The deliberative module encodes a knowledge-based domain model and implements a high-level symbolic reasoning system. The reactive module implements a low-level behavior-based controller with supporting perception and memory subsystems. The reactive module is responsible for the immediate safety of the agent. As such, it functions competently on its own and runs at the highest priority. At the intermediate level, the plan execution and monitoring module establishes an adviser-client relationship between the deliberative and reactive modules.

In typical hybrid control frameworks, the reactive module serves as a mechanism to safely execute commands produced through high-level reasoning [Connell 1992; Jenkin et al. 1994] (a notable exception is [Arkin et al. 1987]). A reactive module is capable of much more as is shown by Tu and Terzopoulos [Tu and Terzopoulos 1994], Blumberg [Blumberg 1994], and Arkin [Arkin et al. 2001], among others. Agre and Chapman [Agre and Chapman 1987] observe that most of our daily activities do not require any planning whatsoever; rather, deliberation occurs when a novel, previously unseen situation is encountered. This further highlights the importance of a reactive module in any autonomous robot. CoCo features an ethologically inspired behavior based reactive system fashioned after those developed for autonomous characters in virtual environments [Tu and Terzopoulos 1994; Shao and Terzopoulos 2005a].

In CoCo, the deliberative module advises the reactive module on a particular course of

action through motivational variables. In contrast to other control architectures where the deliberative module replaces the action selection mechanism built into the reactive module [Gat 1992], our approach provides a straightforward mechanism for providing high-level advice to reactive behavior without interfering with the action selection mechanism built into the reactive module.

Figure 9.2 illustrates the AR&D system realized within the CoCo framework. The satellite recognition and tracking routines compute the position and orientation of the satellite and supply the perceptual input to the reactive module, where the servo behaviors that control the kinematic and dynamic actions of the robotic manipulator provide the relevant motor skills.

9.2 Motor Skills: Visual Servo Behaviors

Satellite rendezvous and docking operations, like all space missions, place stringent requirements on the safety of both the astronauts and the equipment. Therefore, these missions adhere to strict operational guidelines and fully scripted and rehearsed activities. The *Mobile Servicing Systems Guide for International Space Station Robotic Systems* [SSP 1997] defines approach trajectories and envelopes as well as mission stages for robotic manipulators during contact operations. During a manned satellite capture operation, an astronaut controls the robotic arm and moves the end-effector through a sequence of way points, which are defined relative to the target satellite. These way points are defined so as to reduce the time that the end-effector spends in close proximity to the target satellite.

AR&D operations will most likely follow the operational guidelines developed for manned operations, especially the concepts of way points, incremental alignment, and stay-out zones. Jasiobedzki and Liu [Liu and Jasiobedzki 2002] divide a satellite capture operation into six phases (Figure 9.3): 1) visual search, 2) monitor, 3) approach, 4) stationkeep, 5) align, and 6) capture, which comply with the robotic manipulator approach guidelines prescribed in [SSP 1997]. During the *visual search* phase, the cameras are pointed in the direction of the target

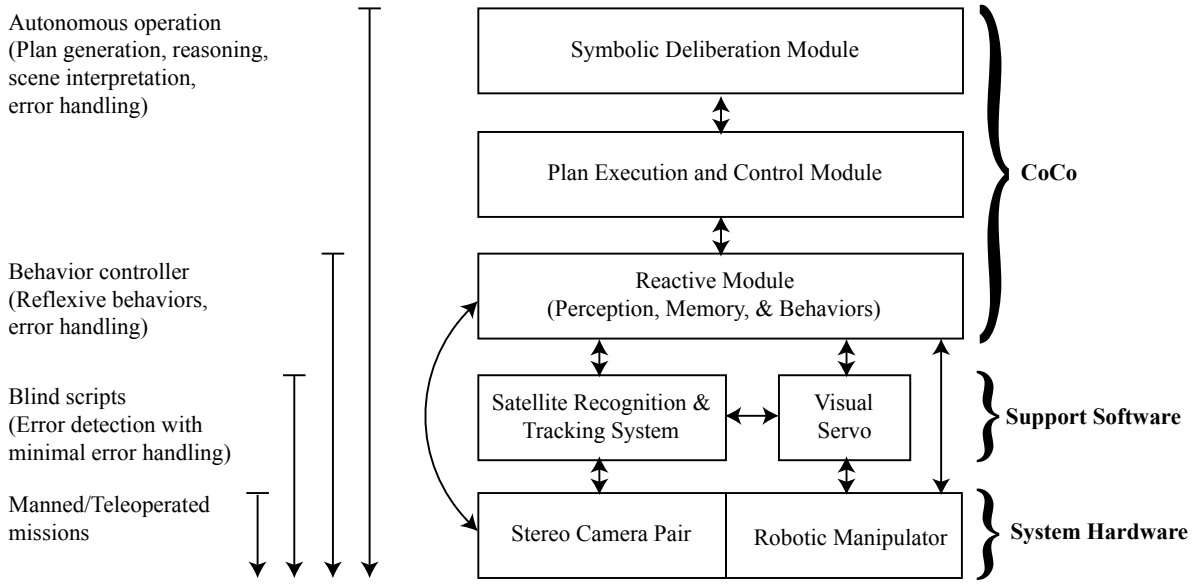
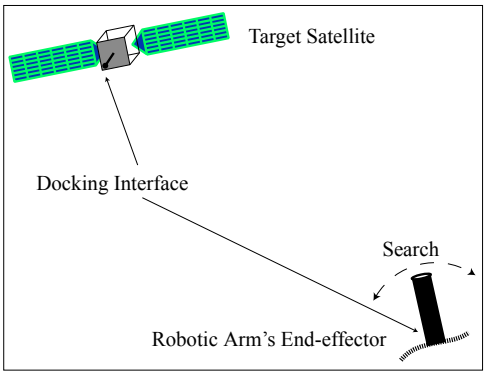
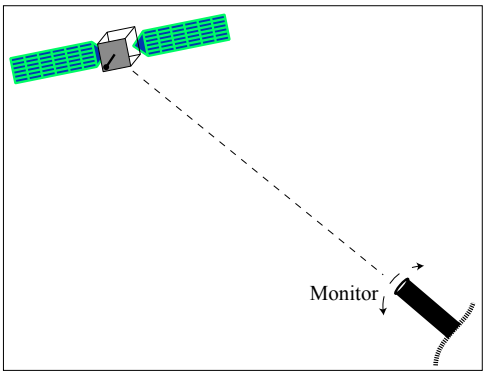


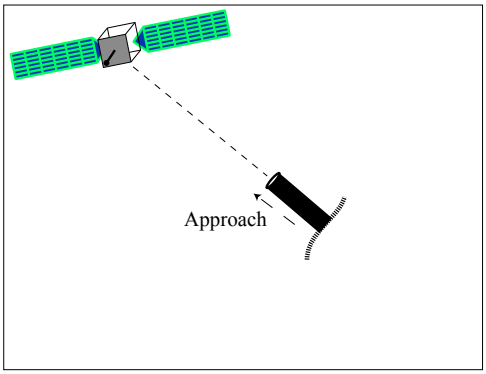
Figure 9.2: CoCo system architecture. The satellite rendezvous and docking system comprises an ethologically-inspired behavior module guided by a deliberative module with high-level reasoning abilities. We list the mission capabilities corresponding to different levels of control on the left. The degree of autonomy increases as we add more levels of control. At the lowest level, for example, an operator uses the live video feed from the stereo camera pair to teleoperate the robotic manipulator (a.k.a. the chaser robot or the servicer) in order to dock with the target satellite. At the next level of control, visual servo routines that depend upon the target satellite pose estimation module enable the robotic manipulator to automatically capture the target satellite. Here, the lack of error handling capability renders the operation brittle at best. Furthermore, the system requires a detailed mission script. The addition of the reactive module results in a more robust autonomous operation, as the reflexive behaviors allow the system to respond to the various contingencies that might arise in its workspace. However, the system still cannot formulate plans through deliberation to achieve its goals. Consequently, the system requires a mission script. The top-most level of control boasts the highest degree of autonomy. Here, the symbolic deliberation module enables the system to generate the mission script on the fly through reasoning.



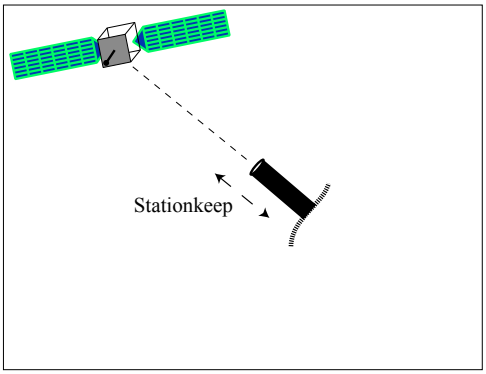
(a) Search



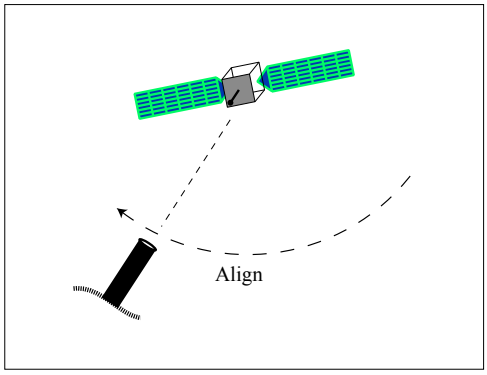
(b) Monitor



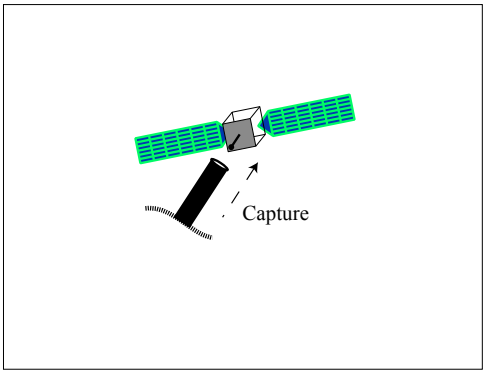
(c) Approach



(d) Stationkeep



(e) Align



(f) Capture

Figure 9.3: Six phases during a satellite rendezvous and docking operation.

satellite, and images from the cameras are processed to compute an initial estimate of the position and orientation of the satellite. The *monitor* phase fixates the cameras on the detected satellite while maintaining distance between the satellite and the end-effector. The *approach* phase reduces the distance between the end-effector and the target satellite while keeping the cameras focused on the target. During *stationkeeping*, the distance between the end-effector and the target is preserved and the cameras are kept locked onto the target. The *align* phase controls all six degrees of freedom, aligning the end-effector with the docking interface of the target satellite. Finally, in the *capture* phase, the end-effector moves in to dock with the satellite.

Jasiobedzki and Liu [Liu and Jasiobedzki 2002] also developed visual servo behaviors corresponding to the six phases identified above. Pose-based servo algorithms that minimize the error between the desired pose and the current pose of the end-effector implement the visual servo behaviors. Poses are defined with respect to the end-effector or the target satellite. During tele-operated missions, the desired poses are set by the operator, whereas during autonomous operation the desired poses are selected by the active servo behavior, such as monitor, approach, etc. Likewise, the higher-level controller can also set the desired poses, especially when the vision system is failing, to move the end-effector along a particular trajectory. The vision system, which estimates the transformation between the current pose and the desired pose, yields the error signal for the pose-based servo routines. The visual servo behaviors provide the motor skills that are essential for successful satellite rendezvous and docking missions.

9.3 Visual Sensors: Satellite Recognition and Tracking

The satellite recognition and tracking module (Figure 9.4) processes images from a calibrated passive video camera-pair mounted on the end-effector of the robotic manipulator and estimates the relative position and orientation of the target satellite [Jasiobedzki et al. 2002]. It supports medium and short range satellite proximity operations; i.e., approximately from 6m

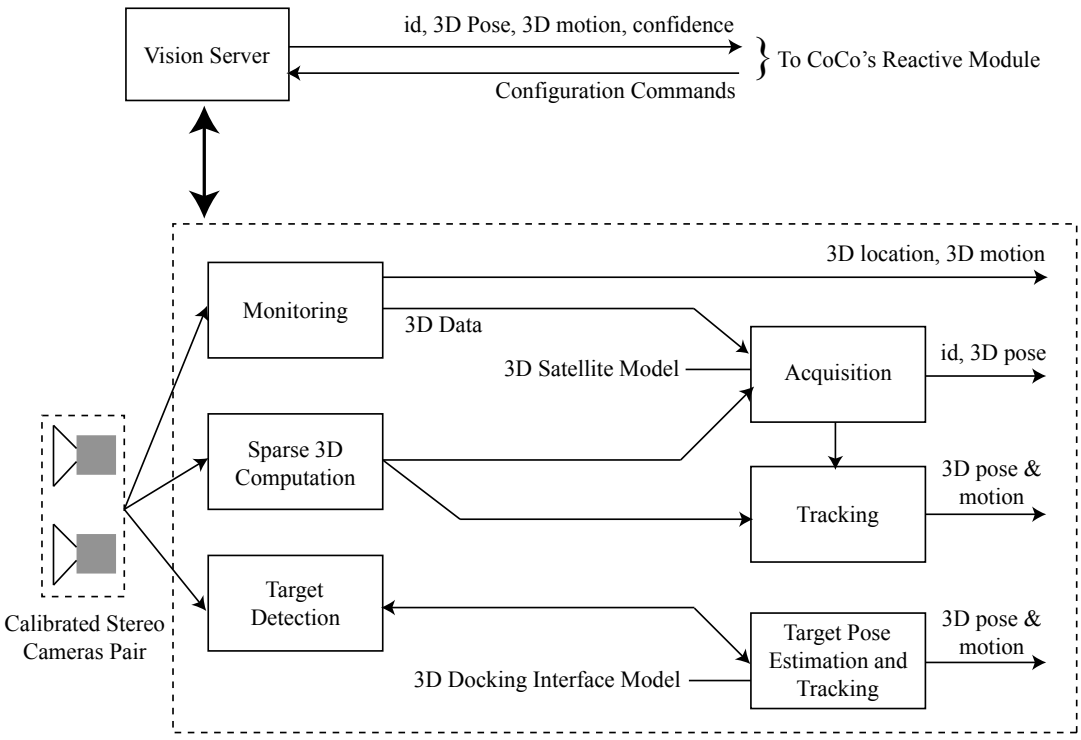


Figure 9.4: The satellite recognition and tracking system.

to 0.2m. The minimum distance corresponds to the separation between the camera and the satellite in contact position.

The vision algorithms implemented rely mostly on the presence of natural image features and satellite models. During the medium range operation, the vision system cameras view either the complete satellite or a significant portion of it (left image in Figure 9.5), and the system relies on natural features observed in stereo images to estimate the motion and pose of the satellite. The medium range operation consists of the following three configurations:

- **Model-free motion estimation:** In the first phase, the vision system combines stereo and structure-from-motion to indirectly estimate the satellite motion in the camera reference frame by solving for the camera motion, which is the opposite of the satellite motion [Roth and Whitehead 2000].

- **Motion-based pose acquisition:** The second phase performs binary template matching to estimate the pose of the satellite without using prior information [Greenspan and Ja-

Figure 9.5: Images from a sequence recorded during a docking experiment (left image at 5m; right at 0.2m)

siobedzki 2002]. It matches a model of the observed satellite with the 3D data produced by the previous phase and computes a 6 degree-of-freedom (DOF) rigid transformation that represents the relative pose of the satellite. The six DOFs are solved in two steps. The first step, which is motivated by the observation that most satellites have an elongated structure, determines the major axis of the satellite. The second step solves for the remaining 4 DOFs—the rotation around the major axis and the 3 translations—through exhaustive 3D template matching over the 4 DOFs.

- **Model-based pose tracking:** The last phase tracks the satellite with high precision and update rate by iteratively matching the 3D data with the model using a version of the iterative closest point algorithm [Jasiobedzki et al. 2001]. This scheme does not match high-level features in the scene with the model at every iteration. This reduces its sensitivity to partial shadows, occlusion, and local loss of data caused by reflections and image saturation. Under normal operating conditions, model-based tracking returns an estimate of the satellite's pose at 2Hz with an accuracy on the order of a few centimeters and a few degrees.

The short range operation consists of one configuration, namely *visual target-based pose acquisition and tracking*. At close range, the target satellite is only partially visible and it cannot be viewed simultaneously from both cameras (the center and right images in Figure 9.5); hence, the vision system processes monocular images. The constraints on the approach trajec-

tory ensure that the docking interface on the target satellite is visible from close range. Markers on the docking interface are used to determine the pose and attitude of the satellite efficiently and reliably at close range [Jasiobedzki et al. 2002]. Here, visual features are detected by processing an image window centered around their predicted locations. These features are then matched against a model to estimate the pose of the satellite. The pose estimation algorithm requires at least 4 points to compute the pose. When more than four points are visible, sampling techniques choose the group of points that gives the best pose information. For the short range vision module, the accuracy is on the order of a fraction of a degree and 1mm right before docking.

The vision system returns a 4×4 matrix that specifies the relative pose of the satellite, a value between 0 and 1 quantifying the confidence in that estimate, and various flags that describe the state of the vision system.

The vision system can be configured on the fly depending upon the requirements of a specific mission. It provides commands to activate/initialize/deactivate a particular configuration. At present, this module can run in four different configurations, which may run in parallel. Each configuration is suitable for a particular phase of the satellite servicing operation and employs a particular set of algorithms. Active configurations share the sensing and computing resources, which reduces the mass and power requirements of the vision system, but can adversely affect its overall performance.

9.4 The Reactive Module

CoCo's reactive module is a behavior-based controller that is responsible for the immediate safety of the agent. As such, it functions competently on its own and runs at the highest priority. At each instant, the reactive module examines sensory information supplied by the perception system, as well as the motivational variables whose values are set by the deliberative module, and it selects an appropriate action. Its selection thus reflects both the current state of the world

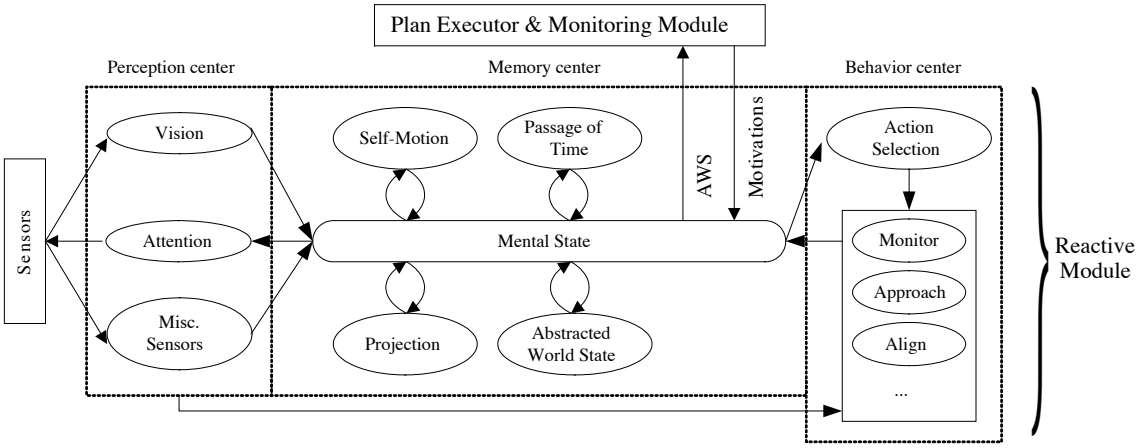


Figure 9.6: Functional decomposition of the reactive module, which is realized as a set of asynchronous processes.

Class	Input	Output	Functional Unit
1	External	External	Behavior Center (Reflex actions)
2	External	Internal	Perception Center (Sensing)
3	Internal	External	Behavior Center (Motor commands)
4	Internal	Internal	Memory Center (Mental state maintenance) Behavior Center (High level behaviors) Perception Center (Sensor Fusion)

Table 9.1: Four classes of asynchronous processes (behaviors) constitute the reactive module.

and the advice from the deliberative module. The second responsibility of the reactive module is to abstract a continuum of low-level details about the world and present a tractable discrete representation of reality within which the deliberative module can effectively formulate plans. CoCo's reactive module comprises three functional units: perception, memory, and behavior (Figure 9.6). This functional decomposition is intuitive and facilitates the design process. The reactive module is implemented as a collection of asynchronous processes (Table 9.1), which accounts for its real-time operation.

9.4.1 Perception Center

¿From an implementational point of view, one can imagine two extremes: one in which a single process is responsible for computing every feature of interest, and the other in which every

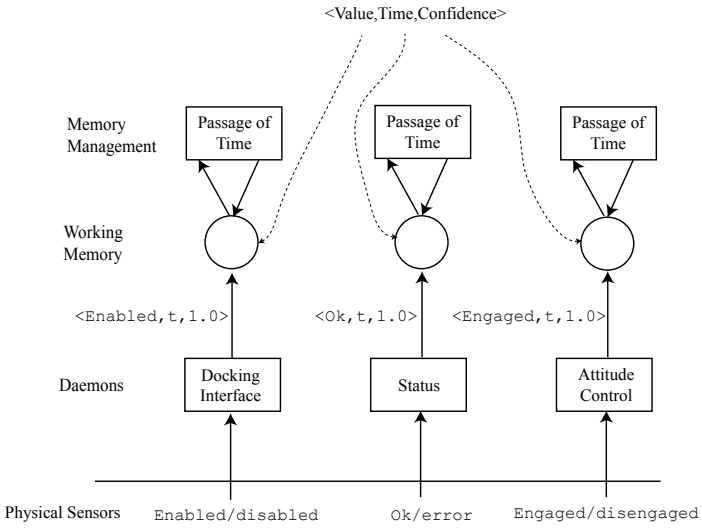


Figure 9.7: Daemon processes for monitoring and reading servicing satellite attitude control, docking interface, and robotic arm status sensors. The daemons that collect information from the sensors are associated with the perception center (bottom row), whereas those that operate upon the working memory belong to the memory center (top two rows).

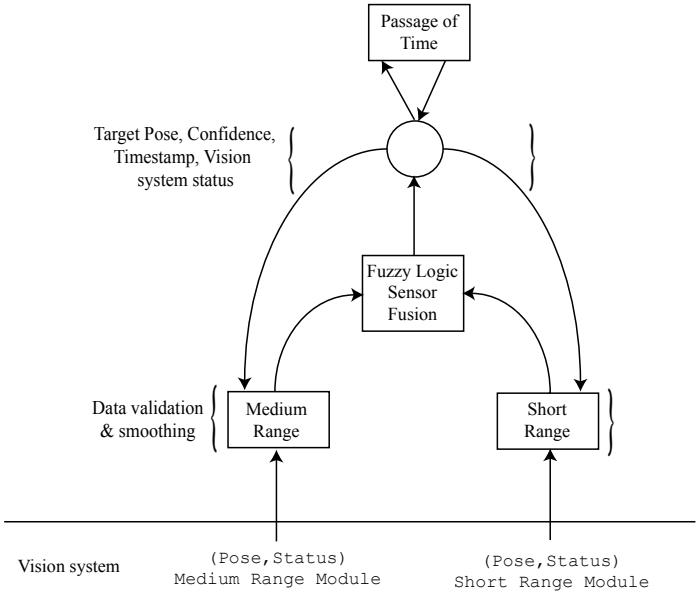


Figure 9.8: The perception center is in charge of the vision system that implements satellite identification, recognition, and tracking routines. The deamon processes associated with the visual processing awakens when new vision readings become available and copy the new readings into the working memory. The vision readings are validated and smoothed using an $\alpha\beta$ tracker. A fuzzy logic based sensor fusion scheme combines the readings when multiple vision configurations are active. A passage-of-time behavior associated with the satellite pose information implements a forgetting mechanism, which prevents the reactive system from using outdated information.

feature is assigned its own sensing process. In the first scenario, the overall speed of sensing is determined by the feature that takes the longest time to compute, whereas a higher process management overhead is associated with the second scenario to ensure that the sensed values are coherent. For a particular application, it is up to the designer to decide how best to implement the perception system. We chose the second approach where various routines process different perceptual inputs asynchronously in order to compute higher order features, which are then immediately available for subsequent processing. Each data item is assigned a timestamp and a confidence value between 0 and 1, and it is managed by the memory center, which is responsible for preventing other processes from using outdated or incorrect information.[1]

The perception center manages the vision system, which was described in Section 9.3. It decides which vision modules to activate and how to combine the information from these modules depending on their characteristics, such as processing times, operational ranges, and noise. In addition, the perception center incorporates an attention mechanism that gathers information relevant to the current task, such as the status of the satellite chaser robot, the docking interface status, and the satellite's attitude control status. The perception center processes the raw perceptual readings that appear at its inputs, constructs appropriate perceptual features, and stores them in the working memory (memory center) for later use by the behavior center during action selection and behavior execution. A perceptual reading is either from an actual physical sensor (e.g., the docking interface sensor) or the result of a multistage operation (e.g., the target satellite's position and orientation). Each perceptual reading is processed independently. Consequently, different perceptual features become available to the reactive module as soon as they are computed.

The perception center includes daemon processes for every perceptual input (Figs. 9.7 and 9.8). The daemon processes, which awaken whenever new information arrives at their input ports, assign a confidence value to the readings, timestamp them, and push them up the perception pipeline for subsequent processing. The confidence value for a reading is in the range $[0, 1]$,

[1] Working memory is sometimes referred to as the *short term memory* or STM.

where 0 reflects a total lack of confidence and 1 reflects absolute certainty. It is computed either by the associated daemon or by the process responsible for producing the perceptual reading in the first place. For instance, the vision routines determine the confidence for the estimated position and orientation of the satellite and the daemon responsible for the docking interface sensor assigns a value of 1 to each new reading it receives from the sensor.

Communicating with the Vision Module

Figure 9.8 shows the interface to the vision sub-system. Long range vision operates anywhere between 20m to 5m, and the maximum separation between the mock-up satellite and robotic arm is roughly 6m. To estimate the position and orientation of the satellite, the perception center uses contextual information, such as the current task, the predicted distance from the target satellite, the operational ranges of the various configurations, and the confidence values returned by the active configurations. The perception center is responsible for the transitions between the different vision configurations, and it also performs a sanity check on the operation of the vision sub-system. A decision about whether or not to accept a new pose reading from an active vision module is made by thresholding the confidence value of the reading. The minimum acceptable confidence value for a medium range estimate is 0.3 and it is 0.6 for a short range estimate. These threshold values reflect the expected performance characteristics of the vision system and are selected to impose more stringent performance requirements on the vision system when the robotic arm is in close proximity to the target satellite.

An $\alpha\beta$ tracker validates and smoothes the pose readings from the vision configurations (Appendix A). The validation is done by comparing the new pose against the predicted pose using an adaptive gating mechanism. When new readings from the vision system consistently fail the validation step, either the vision system is failing or the satellite is behaving erratically and corrective steps are needed. The $\alpha\beta$ tracker thus corroborates the estimates of the visual routines. In addition, it provides a straightforward mechanism for compensating for visual processing delays by predicting the current position and orientation of the target satellite.

Visual Processing Handover

In the final stages of a successful satellite capture operation, the distance between the robotic arm and the target satellite can vary anywhere from around 6m to 0m. The perception center is responsible for transitioning the visual tracking task from the medium to the short range module as the robotic arm approaches the target satellite and vice versa as it pulls back. The perception center uses the estimated distance of the target satellite and the confidence values returned by the active vision configurations to decide which vision module to activate/deactivate.

The strategy for controlling the transition between medium and short range vision modules is based on the following intuitions:

- Since the vision modules are designed to perform reliably only in their operational ranges, a vision module whose estimate falls outside of its operational range should not be trusted.

- When the estimates returned by the active vision module nears its operational limits, activate the more reliable vision module. The operational range of a vision module and the estimated distance of the target satellite determines the suitability of the vision module. For example, when the medium range vision module is active and the target distance estimate is less than 2m, the short range vision module is activated. The short range vision module uses the current pose of the satellite as estimated by the medium range module to initialize satellite tracking.

- A vision module that is currently tracking the target satellite should not be deactivated unless another vision module has successfully initiated target tracking.

- Avoid unnecessary hand-overs.

We describe the hand-over strategy between different vision modules in Appendix B. Figure 9.9 shows the operational status of the vision module during a typical satellite capture mission. Initially, the medium range vision module is tracking the target; however, as the

robotic arm approaches the satellite and the distance to the satellite decreases below 2m, the short range module is activated. Once the short range vision module successfully locks onto the satellite and commences visual tracking, the medium range vision module is deactivated to conserve energy.

Target Pose Estimation using Multiple Visual Processing Streams

To improve the quality of target pose estimates and to ensure smooth transition between different vision modules, we have implemented a fuzzy logic based sensor fusion scheme that combines pose estimates from active vision modules (Appendix C). The sensor fusion scheme takes into account target pose estimates along with their associated confidences and the operational ranges of the vision modules to compute a weighted sum of the pose estimates from the active modules. Currently, it works only with the short and medium range vision modules.[2]

The position $\mathbf{p}$ of the satellite is given by

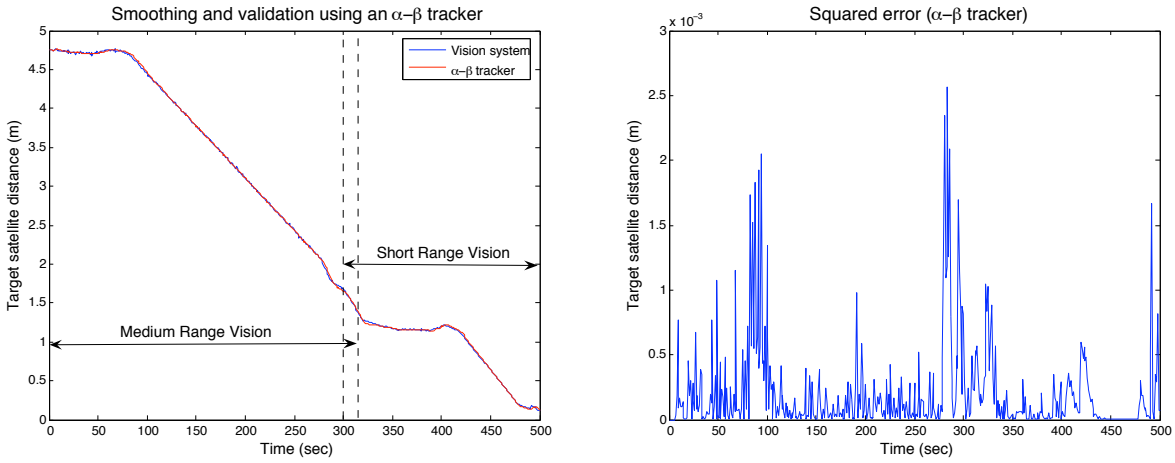
$$\mathbf{p} = w\mathbf{p}_s + (1 - w)\mathbf{p}_m, \tag{9.1}$$

where $0 \leq w \leq 1$ is the weight assigned to the short-range module's estimate and which is determined by the fuzzy logic based controller, and $\mathbf{p}_s$ and $\mathbf{p}_m$ are the position estimates for the short and medium range modules, respectively. Similarly, we combine the orientation estimates from the short and medium range vision modules by expressing the orientation as quaternions and interpolating between them using w:

$$\mathbf{q} = (\mathbf{q}_s\mathbf{q}_m^{-1})^w\mathbf{q}_m, \tag{9.2}$$

where $\mathbf{q}_s$ and $\mathbf{q}_m$ are the rotation estimates from the short and medium range vision modules, respectively (Appendix D). When w is 0 the computed pose of the target is the medium range estimate, whereas when w is 1, it is the estimate returned by the short range module. The details of the sensor fusion module are provided in Appendix C.

[2]The long range vision module is used only initially to locate and identify the target. Once the target is identified, the medium range module takes over. At present, the long and medium range vision modules do not operate concurrently.



(a) (b)

Figure 9.9: (a) Medium range vision hands over target tracking to the short range vision module as the chaser moves closer to the target. (b) The prediction error of the $\alpha\beta$ tracker. The fuzzy logic based sensor fusion scheme fuses the information from active vision modules to form a single coherent target pose estimate.

9.4.2 Behavior Center

The behavior center manages the reactive module's behavioral repertoire. This by no means trivial task involves arbitration among behaviors. The reactive module supports multiple concurrent processes, and arbitrates between them so that the emergent behavior is the desired one. We have, however, benefited from dividing the reactive module into three components (perception, behavior, and memory), minimizing behavior-interaction across the components, thus simplifying the management of behaviors.

At each instant, the action selection mechanism chooses an appropriate high level behavior by taking into account the current state of the world and the motivations. The chosen action then activates lower level supporting behaviors, as necessary. The current state of the world takes precedence over the motivations; i.e., the reactive module will follow the advice from the deliberative module only when the conditions are favourable. When no motivation is available from the deliberative module, the action selection mechanism simply chooses a behavior that is the most relevant, usually one that ensures the safety of the agent.

Motivational Variables

The behavior controller maintains a set of internal mental state variables, which encode the motivations of the robotic arm: 1) *search*, 2) *monitor*, 3) *approach*, 4) *align*, 5) *contact*, 6) *depart*, 7) *park*, 8) *switch*, 9) *latch*, 10) *sensor*, and 11) *attitude control*. The mental state variables take on values between 0 and 1, and at each instant the action selection mechanism selects the behavior associated with the motivational variable having the highest value. Priority among the different motivations resolves behavior selection conflicts when multiple motivations have the same magnitudes. Once the goal associated with a motivational variable is fulfilled, the motivational variable begins to decrease asymptotically to zero.[3] A similar approach to action selection is used by Tu and Terzopoulos [1994] in their artificial fish and by Shao and Terzopoulos [2005a] for their autonomous pedestrians.

We model a *level-of-interest* to prevent one behavior from excluding other behaviors while it infinitely pursues an unattainable goal [Blumberg 1994]. A maximum cutoff time is specified for each motivational variable and if, for whatever reason, the associated goal is not fulfilled within the prescribed cutoff time, the value of the motivational variable starts to decay to 0 (Figure 9.10). We also employ a Minsky/Ludlow [Minsky 1985] model of mutual inhibition to avoid behavior dither; a situation where the action selection keeps alternating between two goals without ever satisfying either of them. Mutual inhibition is implemented by specifying a minimum duration for which a behavior must remain active and by initially increasing the value of the associated motivation variable (Figure 9.11).

The values of the motivational variables are calculated as follows:

$$m_t = \begin{cases} \max\left(0, m_{t-1} - d_a \Delta t \left(1 - d_b m_{t-1}^2\right)\right) & \text{when } t > t_c \text{ or the associated behavior} \\ & \text{achieves its goal} \\ \min\left(1, m_{t-1} + g_a \Delta t \left(e^{m_{t-1}^2} - g_b\right)\right) & \text{when the associated behavior is first initiated,} \end{cases}$$

[3] This is consistent with the "drive reduction theory" proposed by Hull [Hull 1943], whose central theme is that drive (motivation) is essential in order for a response to occur; furthermore, a response is chosen so as to reduce (or satisfy) the most pressing drive.

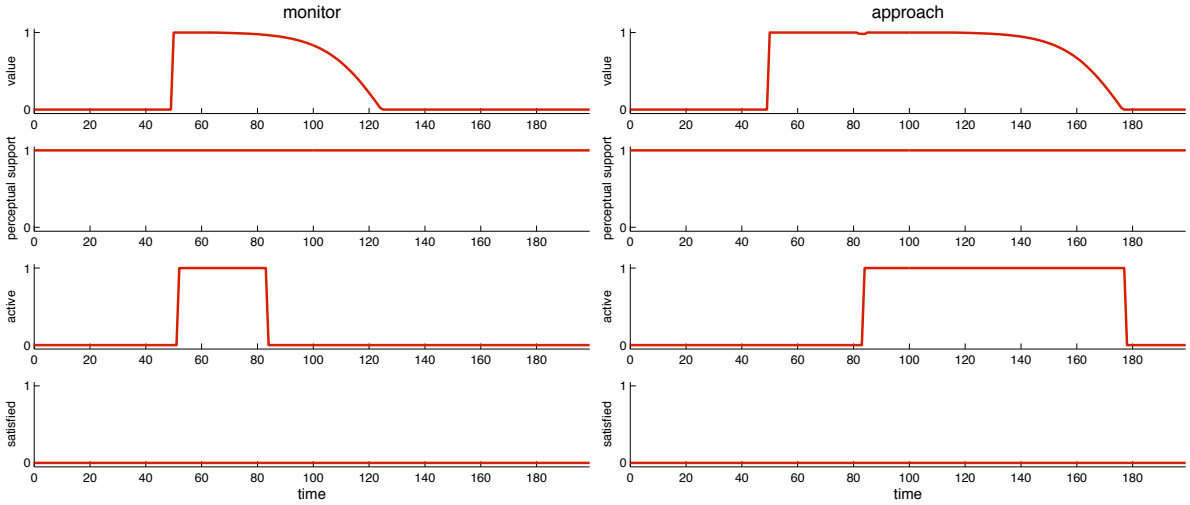


Figure 9.10: Priority among motivations and level-of-interest modeling. The deliberative module sets the value of motivational variables *monitor* (a) and *approach* (b) to 1. The action selection mechanism selects the *monitor* behavior, which has a higher priority than the *approach* behavior. The *monitor* behavior fails to achieve its objectives within the prescribed time, the motivational variable *monitor* begins to decay to 0. When the value of the *monitor* variable is less than that of the *approach* variable, the *approach* behavior is activated. In this particular scenario, the *approach* behavior did not meet its objectives within the prescribed time and the *approach* variable decreases to zero. In either case, the decay in the motivational variables is due to the level-of-interest modeling.

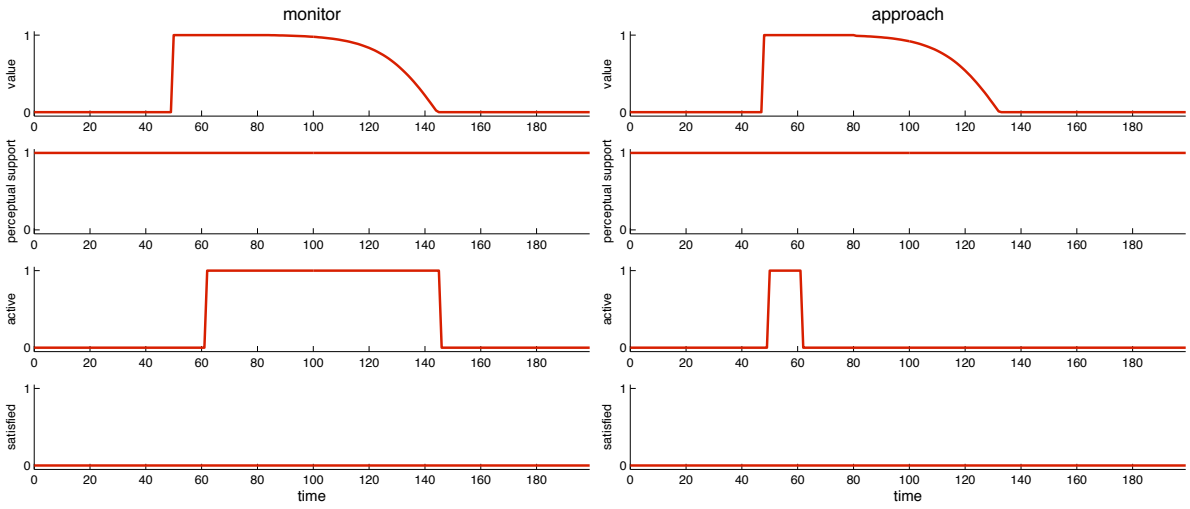
where $0 \leq g_a$,g_b,d_a,$d_b \leq 1$ are the coefficients that control the rate of change in the motivational variables, which are set empirically to 0.5, 0.99, 0.05, and 0.99, respectively.[4] The time step is Δt. The values of a motivational variable at time t and $t - \Delta t$ are m_t and m_{t-1}, respectively. The associated cutoff time is t_c. The cutoff time for a particular motivation depends upon two factors: the motivation in question and whether or not other motivational variables are greater than 0:

$$t_c = \begin{cases} t_1 & \text{when other motivational variables} > 0, \\ t_2 & \text{otherwise,} \end{cases}$$

where $0 < t_1 < t_2 < \infty$.

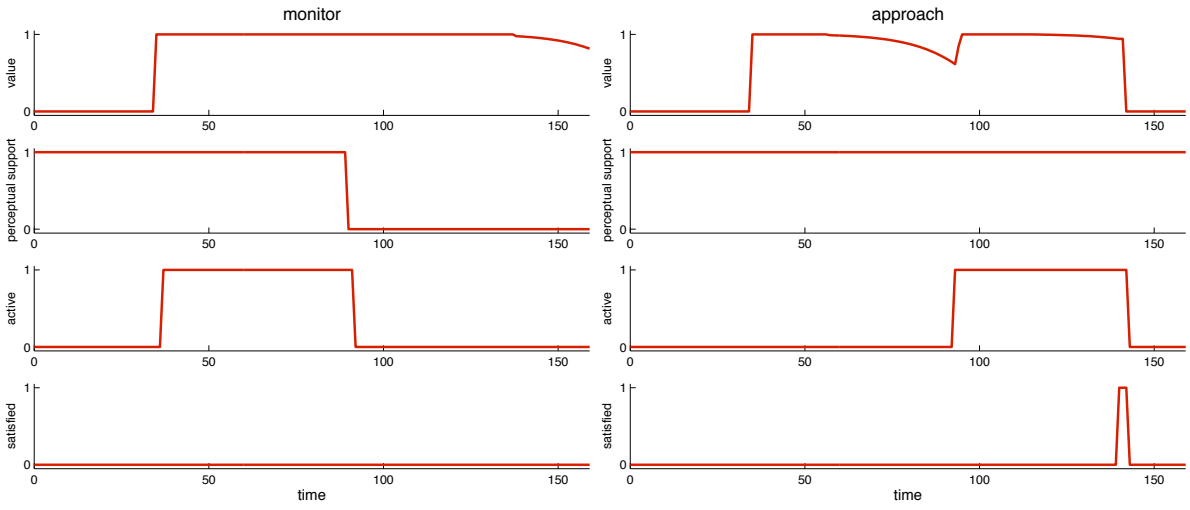
The higher-level deliberative module suggests an action to the reactive module by setting

[4] In a more general setting the values of the coefficients can be chosen on a per-motivation basis.



(a) (b)

Figure 9.11: Mutual inhibition. The *monitor* behavior has a higher priority than the *approach* behavior; however, when the *approach* behavior is active, it inhibits the *monitor* behavior for some prescribed time and prevents the *monitor* behavior from becoming active. After the inhibition period, the *monitor* behavior becomes active, deactivating the *approach* behavior.



(a) (b)

Figure 9.12: (a) Perceptual support: *monitor* behavior is deactivated when perceptual support for the *monitor* behavior vanishes. (b) The *approach* behavior initially increases the *approach* variable to encourage persistence, and the *approach* variable decreases as the *approach* behavior is doing its job.

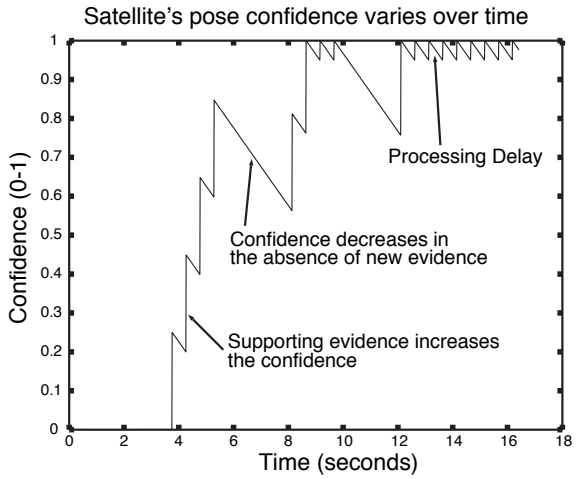


Figure 9.13: The confidence in the satellite's pose decreases in the absence of supporting evidence from the vision system.

the relevant motivational variable(s) to 1 or 0. Any parameters associated with the suggested action are passed directly to the behavior linked to the motivational variable. It is up to the reactive module to decide whether or when to execute the suggested action by activating the associated behavior. Furthermore, the reactive module is not responsible for communicating its decision or status to the deliberative module. The plan execution and monitoring module determines whether or not the suggested action was ever executed or that it failed or succeeded through the abstracted world state.

A consequence of the design proposed here is that the behavior-based reactive module is oblivious to the existence of the deliberative and the plan execution and monitoring modules. The sole agenda of the reactive module is to minimize the internal motivational variables by activating appropriate behaviors. The system operates at a diminished capacity when higher level modules are disabled. Built into the reactive module is a provision for overriding the action selection mechanism during a teleoperated mission; i.e., when the system is being controlled by an astronaut.

9.4.3 Memory Center

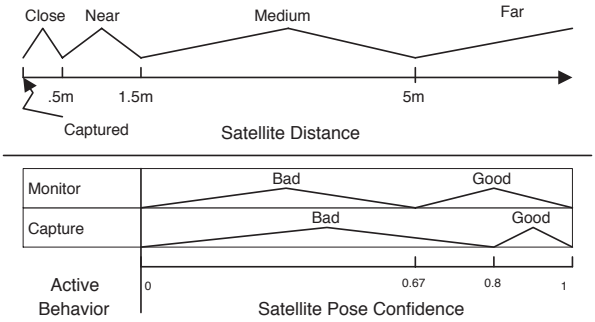


Figure 9.14: The abstracted world state represents the world symbolically. For example, the satellite is either *Captured*, *Close*, *Near*, *Medium*, or *Far*. In the memory center, the conversion from numerical quantities to symbols takes into account the current state of the agent.

The memory center manages the short-term memory of the agent. It holds the relevant sensory information, motivations, state of the behavior controller, and the abstracted world state. At each instant, it copies whatever new sensory information is available at the perception center, and it provides a convenient way of handling perception delays. At any moment, the memory center has a time-delayed version of the sensory information, and it projects this information to the current instant. Thus, the behavior center need not wait for new sensory information; it can simply use the information stored in the memory center, which is responsible for ensuring that this information is valid.

The memory center uses two behavior routines (per feature), *self-motion* and *passage-of-time*, to ensure the currency and coherence of the information. The robot sees its environment egocentrically. External objects change their position with respect to the agent as it moves. The *self-motion* behavior routine constantly updates the internal world representation to reflect the current position, heading, and speed of the robot.

Each perceptual feature is represented as a tuple $\langle$*Value*, *Timestamp*, *Confidence*$\rangle$ in the working memory. *Value* represents the present value of the feature, *Timestamp* stores the time at which the feature was generated, and *Confidence* $\in [0, 1]$ is the current confidence value of the feature. In the absence of new readings from the perception center, the confidence in the world state should decrease with time (Figure 9.13). How the confidence in a particular feature decreases depends on the feature (e.g., the confidence in the position of a dynamic object

decreases more rapidly than that of a static object) and the penalty associated with acting on the wrong information.[5]

The ability to forget outdated sensory information is critical to the overall operation of the reactive module, providing a straightforward mechanism to prevent it or the deliberative module from operating upon inaccurate, or worse, incorrect information, and can be used to detect sensor failures. The confidence value for a perceptual feature tends to zero in the absence of fresh information from the relevant sensor. The lack of new information from a sensor can be construed as a malfunctioning sensor, particularly for sensors such as the docking interface status sensor that periodically send new information to the perception center.

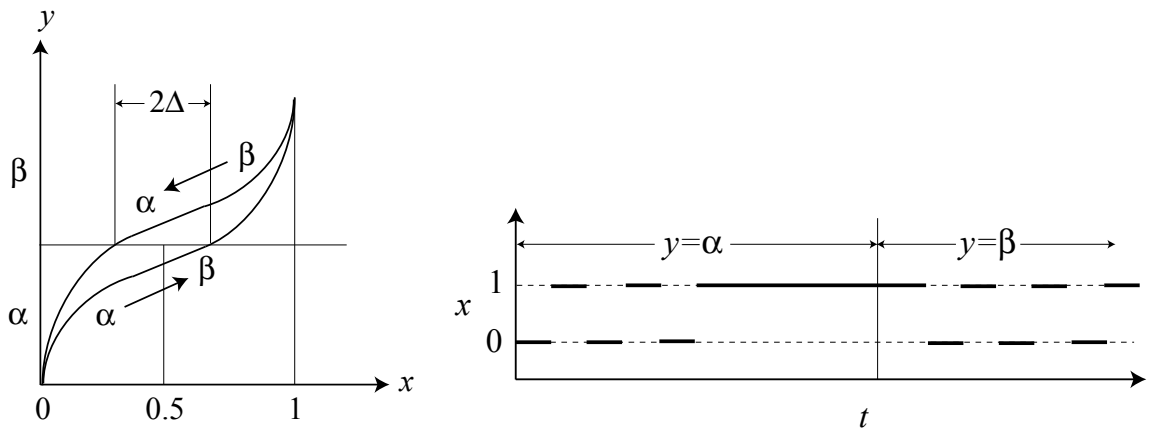
Abstracted World State (AWS)

The reactive module requires detailed sensory information, whereas the deliberative module employs abstract information about the world. The memory center filters out unnecessary details from the sensory information and generates the abstracted world state which expresses the world symbolically (Figure 9.14). The abstracted world state is a discrete, multivalued representation of an underlying continuous reality.

Discretization involves dividing a continuous variable into ranges of values and assigning the same discrete value to all values of the continuous variable that fall within a certain range. Discretization, however, is not without its problems. When the value of the continuous variable hovers about a discretization boundary, the discretized value can switch back and forth between adjacent discrete values, which can pose a challenge for a process that relies on the stability of a discrete variable.

Consider, for example, mapping a continuous variable x whose values lie between 0 and 1

[5]Decreasing confidence values over time is motivated by the decay theory for short term (working) memory proposed by Peterson and Peterson in 1959 [Peterson and Peterson 1959]. In their experiments, 50 participants were shown trigrams and then counting down from 50, the participants were asked to recall them. A long count back caused poor recall, suggesting decay in short term memory. Proactive interference could also explain the poor recall due to counting of numbers before recall.



(a) (b)

Figure 9.15: Emulating hysteresis during discretization. (a) y is a discretization of x that takes values between 0 and 1. If y is α and the value of $x > 0.5 + \Delta$, y becomes β. Otherwise, if y is β and the value of $x < 0.5 - \Delta$, then y becomes α. (b) $x \in [0, 1]$ is mapped to $y \in [\alpha, \beta]$. The state $y = \alpha$ indicates that $x = 0$ and $y = \beta$ indicates that $x = 1$. The variable y resists changing its value from α to β and vice-versa, which allows y to exhibit more stable behavior by ignoring spurious changes in x.

to a discrete variable y that can take one of the two possible values α and β,

$$y = \begin{cases} \alpha & \text{if } 0 \leq x < 0.5 \\ \beta & \text{if } 0.5 \leq x \leq 1 \end{cases}.$$

Here, when the value of x hovers around 0.5, the value of y keeps alternating between α and β.

We address this problem by imitating hysteresis during the discretization operation. We illustrate our strategy in Figure 9.15(a), where variable y resists a change from α to β and vice-versa; thereby, avoiding alternating between the two values when the value of x fluctuates about 0.5. A related approach is taken when converting binary (or discrete multi-valued) sensory information to binary (or multi-valued) fluents. Consider, for example, mapping a binary variable $x \in [0, 1]$ to $y \in [\alpha, \beta]$,

$$y = \begin{cases} \alpha & \text{if } x = 0 \\ \beta & \text{if } x = 1 \end{cases}.$$

The value of y does not faithfully follow the value of x. Rather, the value of y is only switched from α to β when the value of x is consistently 1. Similarly, the value of y is switched from β

Fluents/arity	Values	Description
fStatus/1	on|off	Status of the servicer
fSatPosConf/1	yes|no	Confidence in the estimated pose of the satellite
fSatPos/1	near|medium|far|contact	Distance from the satellite
fSatSpeed/1	yes|no	Whether the satellite's relative speed is within the acceptable limits
fLatch/1	unarmed|armed	Status of the latch (docking interface)
fSatCenter/1	yes|no	Whether the satellite is in the center of the field of view
fSatAlign/1	yes|no	Whether the servicer is aligned with the docking interface of the satellite
fSensor/2	short|medium, on|off	Current configuration of the vision system
fError/1	sensor|shadow|any|no	Error status
fSatContact/1	false|true	Whether the satellite is already docked
fSatAttCtrl/1	on|off	Whether or not the satellite's attitude control is active
fSun/1	front|behind	Location of the Sun relative to the servicer
fRange/1	near|far	Distance from the satellite

Table 9.2: The abstracted world state for the satellite servicing task. The choice of fluents describing the abstracted world state depends upon the target application.

to α when the value of x stays at 0.

Using the above scheme, we convert continuous sensory information, such as the estimated distance from the satellite and the estimated speed of the satellite, as well as binary values, such as the status of the latch, to appropriate fluents that comprise the abstracted world state. The list of fluents is provided in Table 9.2.

9.5 The Deliberative Module

The deliberative module endows our agent with the ability to plan its actions, so that it can accomplish high-level tasks that are too difficult to carry out without "thinking ahead." To this end, the deliberative module maintains a set of planners, each with its own knowledge base and planning strategy. Generally, the world behaves much more predictably at higher levels of abstraction. Hence, each planner understands the world at an abstract level, which makes

reasoning tractable, as opposed to ill-conceived attempts to formulate plans in the presence of myriad low-level details. The lowest level of abstraction for a particular planner is determined by the reactive module explicitly through the abstracted world state and implicitly through the behaviors that it implements. The latter constitute the basis (grounded actions) of the plans generated by the deliberative module. For any application, it is essential to choose the right level of abstraction (Table 9.2).

Symbolic logic provides the right level of abstraction for developing high level planners that elegantly express abstract ideas. We advocate using a high-level agent language, such as GOLOG [Levesque et al. 1997], to develop planners for the deliberative module. Consequently, the deliberative module comprises high-level, non-deterministic GOLOG programs whose execution produce the plans for accomplishing the task at hand. GOLOG is a logic programming language for dynamic domains with built-in primitives (fluents) to maintain an explicit representation of the modeled world, on the basis of user supplied domain knowledge. The domain knowledge consists of what actions an agent can perform (primitive action predicates), when these actions are valid (precondition predicates), and how these actions affect the world (successor state predicates). GOLOG provides high level constructs, such as if-then-else and non-deterministic choice, to specify complex procedures that model an agent and its environment. A GOLOG program can reason about the state of the world and consider various possible courses of action before committing to a particular choice, in effect performing deliberation. The GOLOG language has been shown to be well suited to applications in high-level control of robotic systems, industrial processes, software agents, etc. An advantage of GOLOG over traditional programming languages like C is that programs can be written at a much higher level of abstraction. GOLOG is based on a formal theory of action specified in an extended version of the situation calculus [Reiter 2001], so GOLOG programs can be verified using theorem proving techniques. We treat GOLOG programs as planners; hence, in the remainder of this thesis we will use the term planner and GOLOG program interchangeably.

The symbolic reasoning module comprises two *specialist* planners. Planner A is responsi-

Actions/#args	Arguments' Values	Description
aTurnon/1	on|off	Turns on the servicer
aLatch/1	arm|disarm	Enables/disables the latching mechanism
aErrorHandle/1		Informs the operator of an error condition
aSensor/2	medium|near, on|off	Configures the vision system
aSearch/1	medium|near	Initiates a medium/short visual search sequence
aMonitor/0		Initiates the monitor phase
aAlign/0		Initiates the align phase
aContact/0		Moves in to make contact
aGo/3	park|medium|near, park|medium|near, vis|mem	Moves to a particular location using either the current information from the vision system (if vision system is working satisfactorily) or relying upon the mental state
aSatAttCtrl/1	off|on	Asks the ground station to turn off the satellite attitude control
aCorrectSatSpeed/0		Informs the operator that the satellite is behaving erratically

Table 9.3: The primitive actions available to the planner that creates plans to accomplish the goal of safely capturing the target satellite.

ble for generating plans to achieve the goal of capturing the target satellite. Planner B attempts to explain the changes in abstracted world state. It effectively produces high level explanations of what might have happened in the scene (workspace). The primitive actions available to the two planners are listed in Table 9.3 and 9.4, respectively. The planners experience the world through the fluents (AWS) listed in Table 9.2.

On receiving a request from the plan execution and monitoring module, the deliberative module selects an appropriate planner, updates the planner's world model using the abstracted

Actions/#args	Arguments' Values	Description
aBadCamera/0		Camera failure
aSelfShadow/0		Self-shadowing phenomenon
aGlare/0		Solar glare phenomenon
Sun/1	front|behind	The relative position of the Sun
aRange/1	near|medium	Distance from the satellite

Table 9.4: The primitive actions available to the planner that constructs abstract, high-level interpretations of the scene by explaining how the AWS is evolving.

world state, and activates the planner. The planner computes a plan, which is a sequence of zero (when the planner cannot come up with a plan) or more actions, to the deliberative module, which then forwards it to the plan execution and monitoring module. Each action of an executable plan contains execution instructions, such as which motivational variables to use, and specifies its preconditions and postconditions.

9.5.1 Scene Interpretation

The cognitive vision system monitors the progress of the current task by examining the AWS, which is maintained in real-time by the perception and memory module. Upon encountering an undesirable situation, the reasoning module tries to explain it by constructing an interpretation. If the reasoning module successfully finds a suitable interpretation, it suggests appropriate corrective steps; otherwise, it suggests the default procedure for handling anomalous situations. The default error handling procedure for our application, like all space missions, is to safely abort the mission; i.e., to bring the robotic manipulator to its rest position while avoiding collisions with the target satellite. The procedure for finding explanations is as follows:

Construct plans that account for the current error conditions by using the knowledge encoded within the error model.
Sort these plans in ascending order according to their length. (We disregard the default plan, which usually has a length of 1.)
for all Plans **do**
  Simulate plan execution; this consists of querying the perception and memory unit or asking the operator.
  if The execution is successful **then**
    The current plan is the most likely explanation.
    Break
  end if
end for
if No explanation is found **then**
  The default plan is the most likely explanation.
end if
Generate a solution based on the current explanation; this requires another round of reasoning.
if The solution corrects the problem **then**
  Continue doing the current task.

else
Abort the current task and request user assistance.
end if

A fundamental limitation of the proposed scene interpretation strategy is that it requires a detailed error model—i.e., a knowledge base of what might go wrong and how—and for a general scene it might be infeasible to acquire this knowledge. Space missions, however, can benefit from the approach, since they are usually studied in detail for months and sometimes, years by a team of engineers and scientists who run through all the foreseeable scenarios. Indeed, on-orbit missions are carefully planned and highly scripted activities. Furthermore, they generally take place in *uncluttered* environments, so the number of possible events can be managed. Therefore, our framework appears to be useful for vision-based robotic systems for AR&D. The proposed scene interpretation scheme can be extended to other domains as long as the knowledge base that encodes the error conditions is complete. Such knowledge bases are available for space missions and they might be available in some other situations as well; e.g., in industrial settings.

9.5.2 Cooperation Between Active Planners

The planners cooperate to achieve the goal—safely capturing the satellite. The two planners interact through a plan execution and monitoring unit to avoid undesirable interactions. Upon receiving a new "satellite capture task" from the ground station, the plan execution and monitoring module activates Planner A, which generates a plan that transforms the current state of the world to the goal state—a state where the satellite is secured. Planner B, on the other hand, is only activated when the plan execution and monitoring module detects a problem, such as a sensor failure. Planner B generates all plans that will transform the last known "good" world state to the current "bad" world state. Next, it determines the most likely cause for the current fault by considering each plan in turn. After identifying the cause, Planner B suggests corrections. Possible corrections consist of "abort mission," "retry immediately," and "retry after a

Starting world state: fStatus(off) ∧ fLatch(unarmed) ∧ fSensor(all,off) ∧ fSatPos(medium) ∧ fSatPosConf(no) ∧ fSatCenter(no) ∧ fAlign(no) ∧ fSatAttCtrl(on) ∧ fSatContact(no) ∧ fSatSpeed(yes) ∧ fError(no) **Execution result:** aTurnon(on) → aSensor(medium,on) → aSearch(medium) → aMonitor → aGo(medium,near,vis) → aSensor(short,on) → aSensor(medium,off) → aAlign → aLatch(arm) → aSatAttCtrl(off) → aContact

Table 9.5: A linear plan generated by the GOLOG program to capture the target.

Starting world state: fRange(unknown) ∧ fSun(unknown) ∧ fSatPosConf(yes) **Proposed explanation 1:** aBadCamera **Proposed explanation 2:** aSun(front) → aGlare **Proposed explanation 3:** aRange(near) → aSun(behind) → aSelfShadow

Table 9.6: Planner B uses the error model to determine possible explanations of an error condition. The plan execution and monitoring module executes these plans in sequence to pick the most likely cause of the error. A solution is suggested once the cause of the error is identified.

random interval of time" (the task is aborted if the total time exceeds the maximum allowed time). Finally, after the successful handling of the situation, Planner A resumes.

9.6 Plan Execution and Monitoring Module

The Plan Execution and Monitoring (PEM) module interfaces the deliberative and reactive modules. It initiates the planning activity in the deliberative module when the user has requested the agent to perform some task, when the current plan execution has failed, when the reactive module is stuck, or when it encounters a non-grounded action that requires further elaboration. The execution is controlled through preconditions and postconditions specified by the plan's actions. Together, these conditions encode plan execution control knowledge. At each instant, active actions that have either met or failed their postconditions are deactivated, then un-executed actions whose preconditions are satisfied are activated (Figure 9.16).

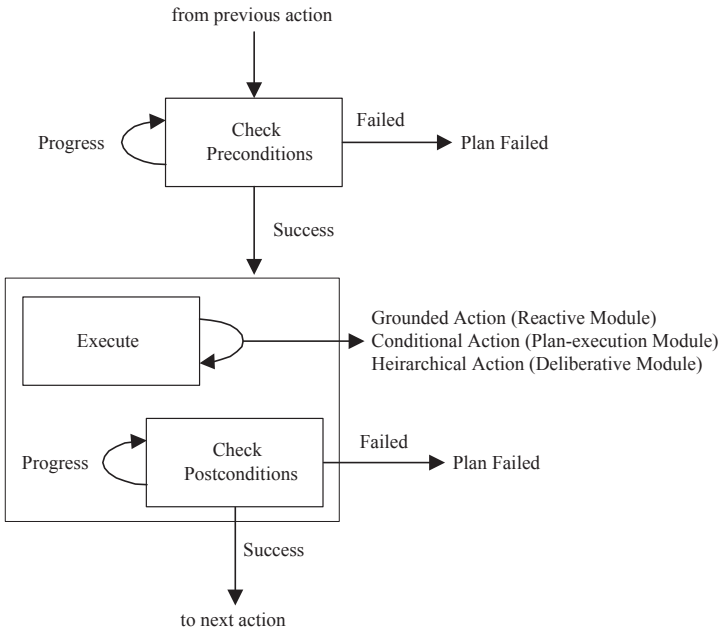


Figure 9.16: The plan execution and monitoring module sequentially executes each action. It checks the current action's preconditions until they succeed or fail. If they succeed, it enters the current action's execution/postcondition-check loop, wherein it activates the current action's execution code until the postconditions either succeed or fail. Upon success, it proceeds to the next action.

Together the preconditions and postconditions constitute the plan execution control knowledge.

The PEM module can handle linear, conditional, and hierarchical plans, thereby facilitating the *sliding autonomy* capability of the overall controller (Figure 9.17). Plans constructed by the deliberative module have a linear structure. Every action of the plan is directly executable on the reactive module, and each action must succeed for the plan to achieve its objectives. Scripts uploaded by human operators usually have a conditional/hierarchical structure. For conditional plans, it is not sufficient to execute each action in turn, rather the outcome of an action determines which of the remaining actions to execute next. On the other hand, in hierarchical plans some actions acts as macros that represent other plans that have to be computed at runtime. The plan execution and monitoring module handles linear, conditional, and hierarchical plans through the following *plan linearization* process (Figure 9.17):

if Current action is "grounded" (i.e., directly executable on the reactive module) **then**
  Send to the reactive module.
else if Current action is "conditional" (e.g., a sensing action, etc.) **then**

Evaluate condition and pick the next action based on the outcome.
else if Current action is "non-grounded" (i.e., it requires further elaboration) **then**
Perform elaboration and replace the hierarchical action with the outcome (plan stitching).
else
Unknown action type. Plan execution failure.
end if

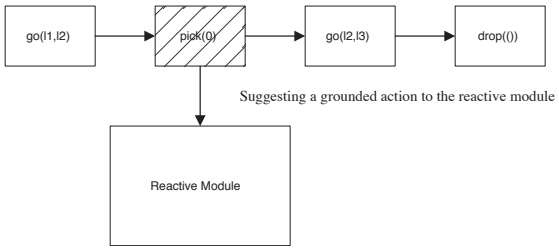
The PEM module can execute multiple actions concurrently; however, it assumes that the plan execution control knowledge for these plans will prevent race conditions, deadlocks, and any undesirable side affects of concurrent execution.

9.6.1 Plan Execution Control Knowledge

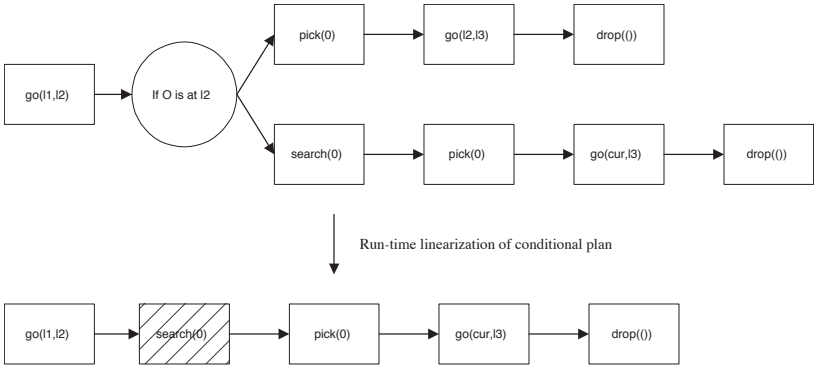
The PEM relies upon execution control knowledge to properly execute a plan. Execution control knowledge is defined over the abstracted world state, and it consists of conditions that must hold before, during, or after an action (or a plan). Some of these conditions span the entire plan while others are action-dependent. Together these conditions answer the following questions that are vital for the correct execution of a plan:

- Plan validity (a plan might become irrelevant due to some occurrence in the world).
- Action execution start time.
  - Now.
  - Later.
  - Never; the plan has failed.
- Action execution stop time.
  - Now; the action has either successfully completed or failed.
  - Later; the action is progressing satisfactorily.

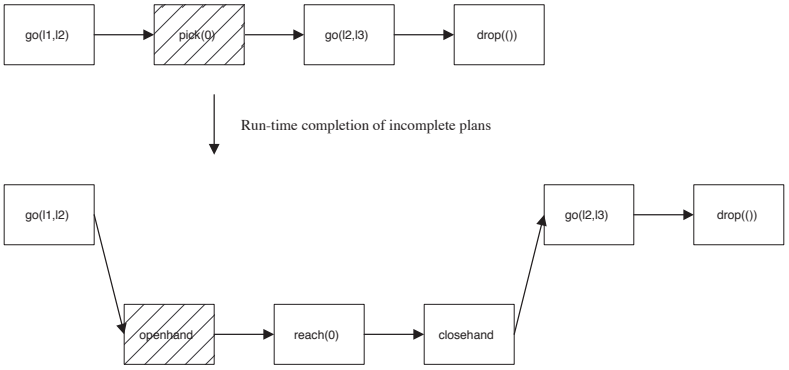
For our application, the plan execution control knowledge is readily available in the form of precondition action axioms and successor state axioms.



(a) PEM suggests a grounded action to the reactive module



(b) PEM evaluates the condition, and the outcome determines which actions are chosen



(c) At run-time, the PEM expands the non-grounded action as a linear plan

Figure 9.17: PEM module executing linear, conditional, and hierarchical plans.

Vision System Errors	Hardware Errors
Camera Failure	Grapple Fixture Error
Self Shadowing	Joints Error (critical)
Solar Glare	Satellite's Attitude Control Error
Failed transition between vision modules	

Table 9.7: CoCo handled these error conditions that were randomly generated during various test runs.

9.7 Results

We have developed and tested the CoCo AR&D system in a simulated virtual environment, as well as in a physical lab environment at MDA Space Missions, Ltd., which faithfully reproduces on-orbit movements and illumination conditions—strong light source, very little ambient light, and harsh shadows. The physical setup consisted of the MD Robotics, Ltd., proprietary "Reuseable Space Vehicle Payload Handling Simulator," comprising two Fanuc robotic manipulators and the associated control software. One robot with the camera stereo pair mounted on its end effector acts as the servicer. The other robot carries a grapple-fixture-equipped satellite mock-up and synthesizes realistic satellite motion.

The capture procedure is initiated by a single high-level command from the ground station. Upon receiving the command, the system initializes the long-range vision module to commence a visual search procedure. Once the satellite is found, and its identity confirmed, the system guides the robotic arm to move closer to the satellite. The performance of the long-range vision module deteriorates as the separation between the robotic arm and the satellite decreases due to the fact that the cameras are mounted on top of the end-effector. In response, the cognitive vision system turns on the medium range vision module and turns off the long-range vision module to conserve power once the medium range system is fully initialized and reliably tracking the satellite. Next, the robotic is arm tries to match the satellite's linear and angular velocities, a procedure known as *station keeping*. Then, short-range vision processing is initiated, and a message is sent to the ground station to turn off the satellite's attitude control system. The robotic arm should not capture a satellite whose attitude control system is func-

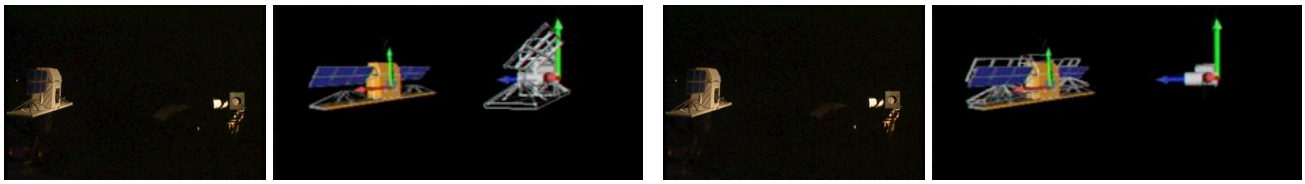
tioning, as that might destroy the satellite, the robotic arm, or both. When the attitude control system is inactive, the satellite begins to drift; however, the robotic arm follows it by relying upon the short-range vision system. Upon receiving a confirmation from the ground station that the satellite's attitude control system is off, the robotic arm moves in to make contact.

We performed 800 test runs in the simulated environment and over 25 test runs on the physical robots. For each run, we randomly created error conditions (see Table 9.7), such as a vision system failure and/or hardware failures. The cognitive controller gracefully handled all of them and met its requirements; i.e., safely capturing the satellite using vision-based sensing (Figure 9.5 shows example sensed images) while handling anomalous situations. The controller never jeopardized its own safety nor that of the target satellite. In most cases, it was able to guide the vision system to re-acquire the satellite by identifying the cause and initiating a suitable search pattern. In situations where it could not resolve the error, it safely parked the manipulator and informed the ground station of its failure.

Figure 1.5 shows a satellite capture sequence in the lab, where the servicer was able to capture the satellite without incident.

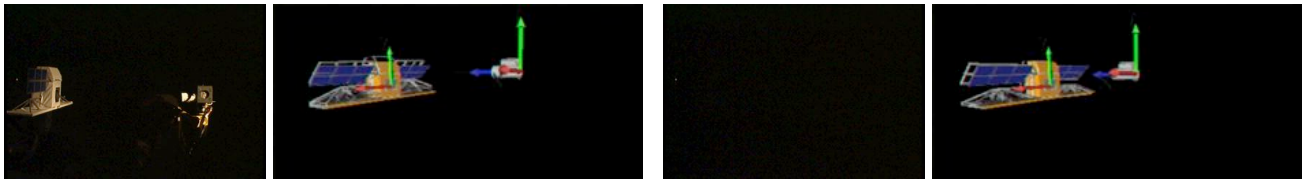
During the capture sequence shown in Figure 9.18, we simulated a vision system failure. The servicer gracefully handled the error by relying upon its cognitive abilities and successfully captured the satellite. When there is an error condition, such as a vision system failure, the reactive system responds immediately and tries to increase its separation from the satellite. In the absence any new perceptual information, the system relies upon its time-aware and context-sensitive mental state. Meanwhile, the deliberation module is using its knowledge base to explain the error and suggest a recovery.

Figure 9.19 shows a simulated satellite rendezvous sequence. Upon receiving a *dock* command from the ground station, the servicer initiates a visual search behavior, which points the cameras towards the incoming satellite (Figure 9.19(a–b)). Once the satellite's identity is confirmed, the servicer begins to approach it (Figure 9.19(c)). Initially, the servicer only has information about the position of the satellite; however, as it approaches the satellite, it activates



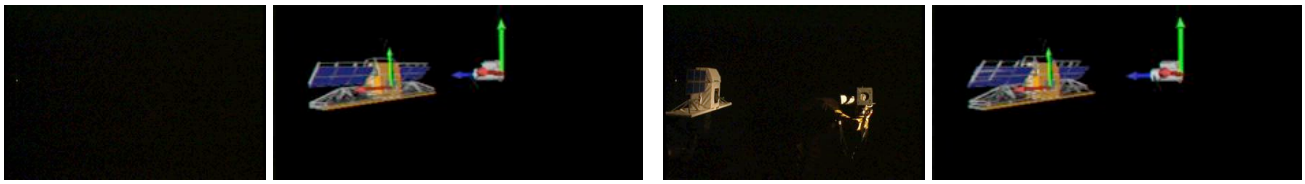
(Frame 1) Recieved a *capture* command.

(Frame 2) Target acquired.



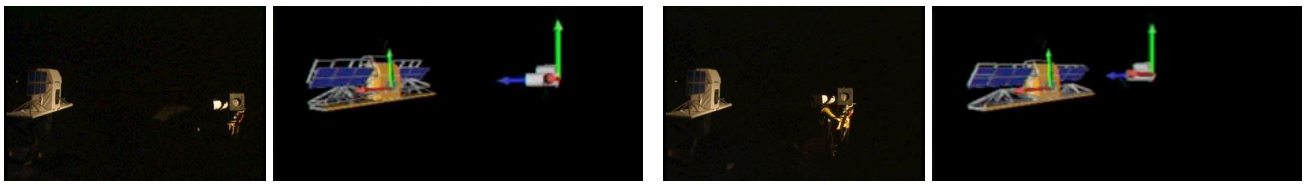
(Frame 3) Approaching target.

(Frame 4) Vision system failure.



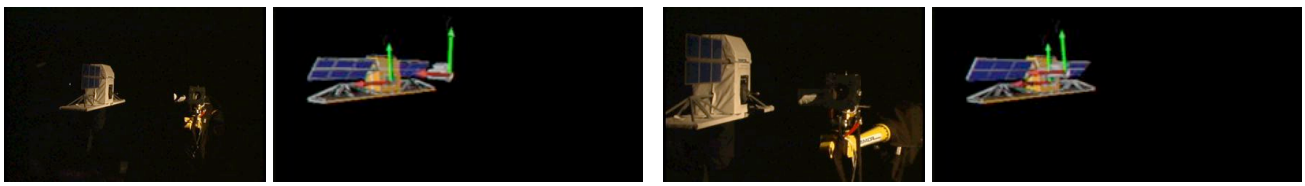
(Frame 5) Safety manuever: departure.

(Frame 6) Target re-aquired.



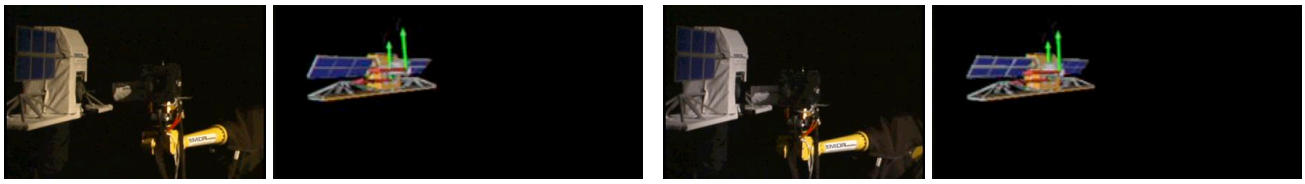
(Frame 7) Approaching target.

(Frame 8) Aligning with the docking interface.



(Frame 9) Station keeping.

(Frame 10) Moving in to make contact.



(Frame 11) Accounting for satellite drift.

(Frame 12) Successful contact.

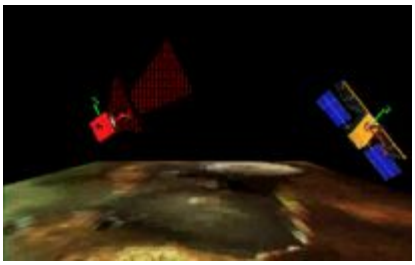
Figure 9.18: Despite a simulated vision system failure, the servicer robot captures the satellite using vision by applying its cognitive abilities. The right part of each frame shows the view from the simulation environment that runs the controller code. The simulation environment communicates with the physical robots over the network. The wireframe model represents the position of the satellite as estimated by the robotic arm. The 3D model of the satellite represents the actual position of the satellite according to the sensors on the Fanuc robot arm. The gray cylinder represents the position of the chaser robot end-effector according to the telemetery information. Note that the estimated position is maintained in the absence new perceptual information (Frames 3 and 4). A vision failure was induced by shutting off the ambient light.

the medium range vision module. Medium range visual processing estimates the orientation of the satellite. The servicer aligns itself with the grapple fixture of the satellite, following an arc around the satellite to avoid the delicate solar panels (Figure 9.19(d-f)). The servicer initiates stationkeeping, where it matches the position and orientation of the satellite (Figure 9.19(g)). At this stage, the servicer's view is limited to the grapple fixture mounted on the satellite, as the cameras are mounted on the docking mechanism of the servicer. Therefore, the servicer activates the short range vision module. Finally, it moves towards the satellite to make contact and capture it (Figure 9.19(h)).

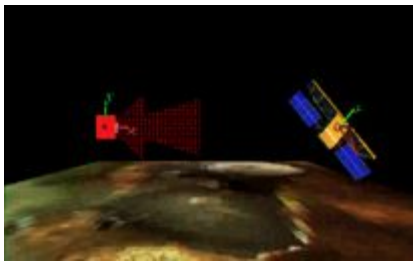
9.7.1 CoCo Outperformed CSA's Controller

The Canadian Space Agency (CSA) also developed an autonomous satellite rendezvous and docking system as part of MDRobotics's efforts to support Boeing's bid for the Orbital Express contract. The CSA controller took a more conventional and accepted approach towards designing intelligent controllers for space missions. It relied on CORTEX, a hierarchical finite state machine based autonomy engine. CORTEX does not support deliberation and is unable to construct error explanations. Our CoCo-based controller and CORTEX rely upon the same motor primitives and sensors provided by MDRobotics Ltd.'s proprietary "Reuseable Space Vehicle Payload Handling Simulator," comprising two Fanuc robotic manipulators and associated control software. One robot with a camera stereo pair mounted on its end effector acts as the servicer. The other robot carries a grapple-fixture-equipped satellite mock-up and it synthesizes realistic satellite motion.

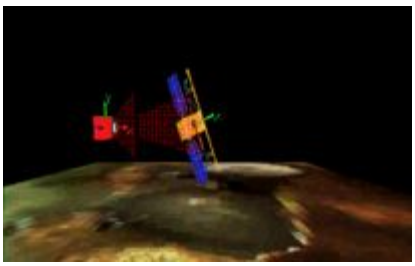
Both controllers were demonstrated during the Space, Vision, and Advaned Robotics Workshop held at MDRobotics on June 11, 2002. Our CoCo-based controller outperformed CSA's controller by capturing the mockup satellite while handling a variety of anomalous situations, such as hardware failures and vision-system failures. CSA's controller could not manage to handle anomalous events and it failed to capture the mockup satellite.



(a) Searching for the target

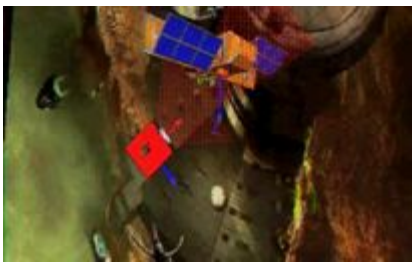


(b) Target identified; Long range vision

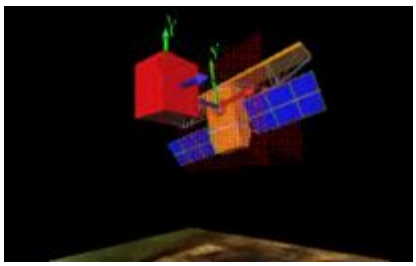


(c) Moving towards the target

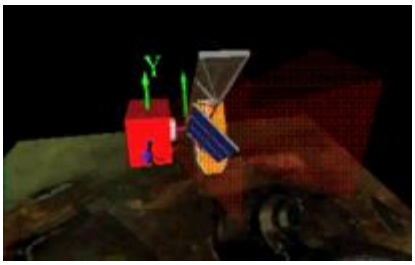
(d) Aligning with the docking interface while avoiding the solar panels



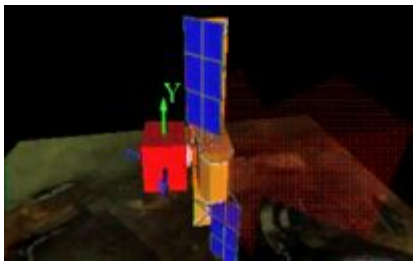
(e) Aligning with the docking interface



(f) Aligning with the docking interface



(g) Making contact



(h) Target captured

Figure 9.19: Satellite rendezvous simulation.

Part III

Conclusion

Chapter 10

Summary and Research Directions

10.1 Space Robotics and CoCo

Like the earliest machine vision systems [Roberts 1965; Nilsson 1984], future applications of vision will require more than just image analysis. They will also need a high-level AI component to guide the vision system in a deliberate, task-appropriate manner, to diagnose sensing problems, and to take corrective actions. The AI system must rely on a low-level reactive component responsible for sensorimotor control.

With this in mind, we developed and demonstrated in the latter part of the thesis a first-of-its-kind vision-based robotic system whose objective is to tackle a challenging space-robotics problem: autonomous satellite rendezvous and docking. To our knowledge, our work is the first attempt at designing an AR&D system that relies solely upon computer vision for its sensing needs. Existing AR&D systems typically rely on RADAR, GPS, or proximity-detector sensors. Ours is also the first use of deliberation in the context of satellite rendezvous and docking scenarios.

Through our work, MDRobotics supported Boeing's successful bid for DARPA's 12 million dollar Orbital Express project. To this end, we proposed an artificial life inspired control architecture, called the Cognitive Controller, or CoCo. CoCo is our attempt to deal with the

challenges that we faced while working on this real-world space robotics problem. CoCo combines low-level visual analysis and high-level reasoning within a hybrid deliberative/reactive control framework. Bridging the gap between, at times, two distinct streams of autonomous agents research—robotics and virtual characters—CoCo appears suitable for visually-guided agents capable of intelligent, autonomous operation in dynamic settings. CoCo's reactive module interacts with an agent's environment through sensors and actuators, and it presents the higher level modules (plan executor and the deliberative module) with an abstracted view of an agent's environment. The deliberative module uses the abstracted world state and the knowledge base to reason about an agent's actions. And the plan executor mediates between the two modules. The plan executor interacts with the reactive module by setting motivations. We have found this to be a powerful and non-intrusive scheme of injecting higher-level advice into the reactive processing, as it does not interfere with the behavior arbitration mechanisms available in the reactive module.

CoCo differs from its predecessors in the following important ways: 1) its reactive module is ethology-based as opposed to subsumption-based, 2) its plan execution and control module interacts with the reactive module through the affective states (i.e., the motivations), and 3) its deliberative module, which sees the world at a higher level of abstraction comprises multiple specialist deliberative routines, as opposed to the ubiquitous single, monolithic deliberation mechanism.

In our space robotics research, there are multiple opportunities for future work. Currently, we assume unlimited power and unbounded computational resources. These are naive assumptions. In fact, power and computational resources are at a premium in space robotics. Indeed, this was one of the motivations for developing a vision-based satellite servicing system. Vision is a passive sensory modality with very little power requirements when compared to that of an active sensor, such as a radar. Therefore, a space-worthy controller must be able to operate under limited resources. Future work on our space robotic system should include analyzing its resource requirements. Additionally, a possible direction of future work is to build

energy resource awareness within CoCo. Energy/computational resource aware control architectures will be most useful in space robotics. These architectures would adapt their cogitation-action cycle to optimize power consumption and computational resources utilization. Exploring ideas from the resource bounded reasoning community should prove advantageous in this regard [Zilberstein 1996].

CoCo relies on a handcrafted, detailed error-model for constructing scene interpretations. It is feasible to construct such a knowledge base for space missions; however, it might be infeasible to acquire this knowledge for a general scene. It is crucial to investigate and remove this limitation before CoCo's promise can be fully exploited in terrestrial robotics.

Although CoCo can deal with hierarchical, conditional, and linear plans, to date its deliberate module can only generate linear plans. It would be worthwhile to investigate more sophisticated planning strategies to augment the deliberative module. These might include conditional planning, anytime planning, and iterative plan repair.

Developing a computer-aided software engineering (CASE) tool to design and study CoCo-based robotic agents would be a valuable future contribution. An initial promise of subsumption architecture was behavior reuse, but this has rarely been the case in practice. A well-designed CASE tool can help maintain behavior libraries that will significantly speed up the initial design process. Additionally, in the absence of a theory of CoCo that would allow us to prove the correctness of a CoCo controller, CASE tools can help us empirically validate a particular CoCo controller.

In a head-to-head competition within a lab environment, our CoCo-based AR&D controller outperformed the conventional, non-deliberative controller developed by the Canadian Space Agency. Although this is an encouraging result, the ultimate test of our AR&D controller would be to deploy it in space. Therefore, an exciting direction for future work would be to make our system spaceworthy. This will likely take many years, and it will be a large-scale, collaborative effort with engineers, scientists, and technicians that specialize in space hardware and software.

10.2 Virtual Vision and Visual Sensor Networks

In the first part of the thesis, we argued that future surveillance systems will comprise networks of static and active cameras capable of providing perceptive coverage of extensive environments with minimal reliance on a human operator. Such systems will require not only robust, low-level vision routines, but also novel sensor network methodologies. Our work represents another step toward their realization.

However, the cost associated with deploying a camera network and experimenting with it on a large scale in a big public space is prohibitive. There are also legal and privacy concerns that must be addressed before experimental camera networks can be deployed in a public space. Consequently, setting up and experimenting with larger scale networks of cameras is beyond the reach of most vision researchers.

In the first half of this thesis, we developed the Virtual Vision paradigm—a unique synthesis of virtual world modeling, computer graphics, computer vision, and sensor network technologies—for camera networks research. Virtual vision democratizes camera network research: any researcher can study and develop camera senor network protocols. All that is required is a high-end commodity PC. We showed virtual vision to be particularly useful for studying camera networks, especially in the study of high-level camera control problems. It is fair to say that virtual vision has enabled us to make substantial progress on such problems in a relatively short period of time. Our approach provides a viable alternative to installing a physical surveillance system, at least during the R&D and evaluation phase.

The success of the virtual vision paradigm for camera network research hinges upon one crucial factor: faithful emulation of physical vision systems. In order to capture the performance characteristics of a physical vision system—failures due to occlusions, lighting variations, targets with similar appearance, etc.—we implemented an appearance based pedestrian tracker that operates upon the synthetic footage captured by the virtual cameras situated in the virtual train station. To this end, we adapted well-known computer vision algorithms. A unique feature of our pedestrian tracker is that it works for both active PTZ and passive wide-

FOV cameras. We have shown that imaging artifacts that are of interest to camera network researchers can be simulated through more advanced rendering techniques. Specifically, we simulate compression artifacts and interlacing. We have developed image-driven PTZ controllers for active cameras. These controllers eschew camera calibration. Additionally, we have developed machine learning based techniques that allow an uncalibrated active PTZ camera to learn the mapping between the 3D locations and its internal pan/tilt settings by fixating a single pedestrian during an initial learning phase.

We examined the problem of perceptive scene coverage for surveillance and proposed a sensor network framework particularly suitable for designing camera networks for surveillance applications. The overall behavior of our prototype camera sensor network is governed by local decision making at each node and communication between the nodes. Our approach is new insofar as it does not require camera calibration, a detailed world model, or a central controller. We have intentionally avoided multi-camera tracking schemes that assume prior camera network calibration which, we believe, is an unrealistic goal for a large-scale camera network consisting of heterogeneous cameras. Similarly, our approach does not expect a detailed world model which, in general, is hard to acquire. Since it lacks any central controller, we expect the proposed approach to be robust and scalable.

Little attention has been paid to the problem of controlling or scheduling active cameras when there are more people to be monitored in the scene than there are available cameras. We introduced a scheduling strategy for intelligently managing multiple, uncalibrated, active PTZ cameras, supported by several static, calibrated cameras, in order to satisfy the challenging task of automatically recording close-up, biometric videos of pedestrians present in a scene. Other novel features of our work is that we assume a non-clairvoyant model, we assume that recording durations of the pedestrian is not known a priori, we do not assume perfect tracking, our scheduling scheme supports preemption, and we allow multiple observations of the same pedestrian. We have found the PMOMC (preemption, multiple observation, multi-class) scheduling scheme to be the most suitable one for this purpose. At present, predicting

pedestrian behaviors is at best an inexact science, so we have intentionally avoided scheduling policies that depend on such predictions.

Smart cameras are self-contained vision systems, complete with image sensors, power circuitry, communication interface, and on-board processing capabilities. They promise to revolutionize next generation camera networks. Two problems are currently being investigated in the context of smart camera networks: automatic camera handoff and automatic camera aggregation.

We developed a simulated network of smart active PTZ cameras capable of task-dependent camera grouping and assignment. We proposed an auction-based network protocol capable of dealing with camera handoff and camera aggregation. Prior work on camera handoff typically deals with passive cameras only, it requires either network topology information and/or assumes camera calibration. By contrast, our work can deal with both active and passive cameras and it does not assume camera and/or network calibration. The camera aggregation protocol proposed by Mallet [2006] organizes cameras into locality based groups, whereas our protocol supports leader nodes (one for each group) that manage group memberships and group-group interactions (camera assignments). We account for node failures and imperfect communications, and we propose a sensor assignment scheme in the presence of conflicts. Furthermore, our protocol does not require dedicated group destruction nodes.

Our virtual vision research offers numerous opportunities for future work. The future of advanced simulation-based approaches appears to be promising in computer vision for the purposes of low-cost design and experimentation.

Currently, our virtual world consists of a large train station, but imagine an entire city, consisting of indoor and outdoor scenes, subway stations, automobiles, shops and market places, houses, and public spaces, all richly inhabited by autonomous virtual humans. Such large-scale virtual worlds will provide unprecedented opportunities for studying large-scale camera sensor networks in ways not currently possible in our train station simulator. Future work on virtual vision research will therefore benefit from long-term efforts to increase the complexity

of virtual worlds.

Additionally, there is a need to simulate more realistic environmental conditions to capture higher fidelity synthetic video. For example, we can simulate object shadows, specularities, and transparency, as well as lens flare, image vignetting, depth-of-field effects, and varying illumination conditions. Among other things, this would include the use of more realistic finite aperture camera models. More sophisticated low-level computer vision routines will probably be required to handle virtual environments where more realistic environment conditions are simulated. Specifically, a more robust pedestrian tracking algorithm would be needed. Our pedestrian recognition and tracking module relies on color signatures, thus it is sensitive to illumination variations. Our color pedestrian recognition and tracking algorithm should be combined with other robust cues, such as gait signatures, SIFT features, and spatial and temporal constraints to minimize the effects of varying light conditions. Camera network color calibration can also be exploited to improve the performance of pedestrian tracking and recognition algorithm.

Currently, we assume that pedestrians can be uniquely identified in multiple cameras. This is a strong assumption. A possible direction for future work would be to relax this assumption. Furthermore, we ignore the problem of distinguishing objects with similar appearance. This problem is not unique to computer vision since humans will have a hard time distinguishing between identical twins wearing the same clothes. Fortunately, PTZ cameras can help avoid these situations by zooming in on the pedestrian of interest.

Scalability is an issue when dealing with a large number of cameras spread over a large area. We hope to tackle scalability by investigating distributed scheduling strategies. Currently, we have proposed a non-clairvoyant scheduling strategy, as it does not require predictions about the exit times of the pedestrians. However, clairvoyant algorithms are known to perform poorly when these predictions are poor. A possible direction of future work is to develop pedestrian traffic models that allow the scheduler to make better predictions. This would require algorithms that can construct high-level interpretations capable of answering Who, What, Where,

and When questions.

Another challenge is to combine camera scheduling with camera organization. The sensor assignment problem can be seen as single-task agents, multi-agent tasks, time-extended assignment (ST-MR-TA) problems that are have been studied in the literature on multi-agent task allocation. Organization (or coalition formation) and scheduling occur naturally in ST-MR-TA problems. Currently, finding optimal solutions to these problems is infeasible, as it requires considering all possible schedules for all coalitions. One approach to address these problems is to employ a leader-based approach that dynamically forms coalitions and build task allocations for them [Dias and Stentz 2002].

Another direction of future work is to investigate the Cognitive Modeling approach to camera node organization and assignments. Here, each node is endowed with reasoning/planning abilities and can apply them to make deliberative choices. For example, cameras can learn their sensing capabilities and limitations over time, and exploit that knowledge when deciding whether or not to take on an observation task. Sensor networks can learn and exploit topological information to pre-task camera nodes through deliberation. Currently, our sensor network model does not support pre-tasking, and this should be remedied in future work.

Of course, the acid test for algorithms and systems developed in our virtual vision simulator is to see if they perform as expected in the real world. Therefore, an important direction for future research would be to connect the virtual vision work to physical multi-camera systems. This will probably require collaboration with researchers who engineer surveillance system hardware and those who deploy them in real-world installations.

Part IV

Appendices

Appendix A

$\alpha\beta$ Tracker for Satellite Pose Validation

An $\alpha\beta$ tracker is a fixed gain formulation of a Kalman filter. It is widely used in track-while-scan and single target tracking applications, since it is easy to implement and performs well in practice [Bar-Shalom and Li 1998; Barton 2004]. It has been shown that the output noise variance at steady state is minimized when the gain coefficients are related as $\beta = \alpha^2/(2 - \alpha)$. The main disadvantage of the fixed gain $\alpha\beta$ tracker is that it estimates the position of an accelerating target with a constant lag. The magnitude of this lag can easily be estimated and the filter can use adaptive gains to improve the RMS tracking error. We decided in favor of an $\alpha\beta$ tracker instead of a Kalman filter for pose tracking in the perception center, since the pose values supplied by the visual routines are already fairly stable. We use an $\alpha\beta$ to validate the target position estimates from satellite tracking routines. Currently, we rely on the underlying vision system to predict/validate the orientation of the target satellite.

A.1 Formulation

Validation:

$$\text{Accept } \mathbf{p}^t, \text{ if } \|\mathbf{p}_p^t - \mathbf{p}^t\| < \frac{\tau}{2}$$

Smoothing:

$$\begin{aligned} \mathbf{p}_s^t &= \mathbf{p}_p^t + \alpha\left(\mathbf{p}^t - \mathbf{p}_p^t\right) \\ \mathbf{v}_s^t &= \mathbf{v}_p^t + \frac{\beta}{\Delta t}\left(\mathbf{p}^t - \mathbf{p}_p^t\right). \end{aligned}$$

Prediction:

$$\begin{aligned} \mathbf{p}_p^{t+\Delta t} &= \mathbf{p}_s^t + \mathbf{v}_s^t \Delta t \\ \mathbf{v}_p^{t+\Delta t} &= \mathbf{v}_s^t. \end{aligned}$$

Where:

$\mathbf{p}^t$: position reading received at time t

$\mathbf{p}_p^t$: the predicted position estimate at time t

$\mathbf{p}_s^t$: the smoothed position estimate at time t

$\mathbf{v}_p^t$: the predicted velocity estimate at time t

$\mathbf{v}_s^t$: the the smoothed velocity estimate at time t

$\mathbf{p}_p^{t+\Delta t}$: the predicted position estimate at time $t + \Delta t$

$\mathbf{v}_p^{t+\Delta t}$: the predicted velocity estimate at time $t + \Delta t$

α, β: gains

τ: context sensitive gating threshold.

Appendix B

Vision Module Handover

The following procedure controls the handover between the medium and short range modules. Here, we assume that the target is initially being tracked by the medium range module. Range A is defined between 2m to 6m, range B is defined as between 2m to 1.5m, and range C is defined as between 1.5m to 0m. The medium range module works reliably when the target's distance is in ranges A and B, whereas the short range module works reliably when the target's distance is in ranges B and C (Figure B.1).

B.1 Handover Procedure

```
1:  if distance is in A then
2:      ensure medium range module is active
3:      if short range module is active then
4:          deactivate it
5:      end if
6:  end if
7:  if distance is in C then
8:      ensure short range module is active
9:      if medium range module is active then
10:         deactivate it
11:     end if
12: end if
13: if distance is in B then
14:     if moving towards the target then
15:         if short range module is not active then
```

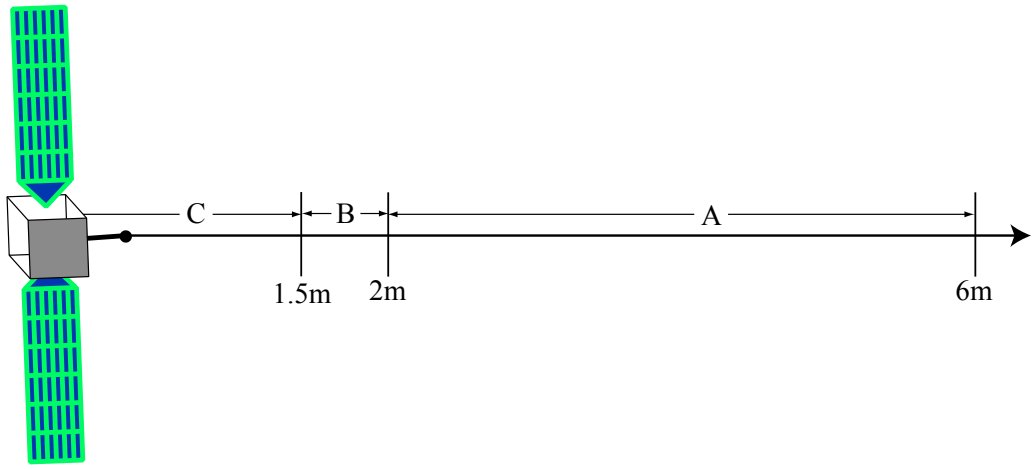


Figure B.1: Satellite tracking handover: Operational ranges of short and medium range vision modules. Short range is active anywhere from 0.2m to 2m (ranges B & C), whereas medium range is active from 1.5m to 6m (ranges A & B).

```
16:          initialize short range module
17:       else if short range is active AND tracking the target competently then
18:          deactivate medium range module
19:       end if
20:    else
21:       if medium range module is not active then
22:          intialize medium range module
23:       else if medium range is active AND tracking the target competently then
24:          deactivate short range module
25:       end if
26:    end if
27: end if
```

Appendix C

Fuzzy Logic Based Sensor Fusion

We employ a fuzzy logic based sensor fusion technique to compute the pose of the target satellite by combining target pose estimates from short and medium range vision modules. The proposed scheme takes into account the estimated distance from the target and the confidence values returned by the short and medium range modules to determine a weight ($\in [0, 1]$) associated with the short range target pose estimation.

Our fuzzy inference system has 3 inputs, 1 output, and 8 rules (Figure C.1). The membership functions corresponding to the three inputs and the output are shown in Figure C.2. We employ max-min inferencing and centroid defuzzification schemes to generate a crisp output. We point the interested reader to [Klir and Yuan 1995] for an introduction to fuzzy logic. We list the fuzzy rules for our system below.

Fuzzy Rules

Rule 1: if distance is not medium **then** use is all-short

Rule 2: if distance is not short **then** use is all-medium

Rule 3: if distance is short **and** short-confidence is excellent **then** use is all-short

Rule 4: if distance is short **and** short-confidence is not excellent **and** medium-confidence is excellent **then** use is short

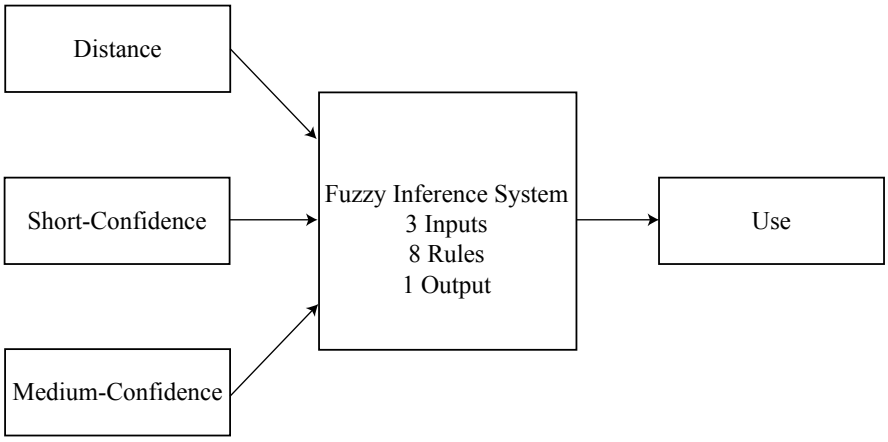


Figure C.1: Sensor Fusion: Fuzzy Inference System with 3 inputs, 1 output, and 8 rules.

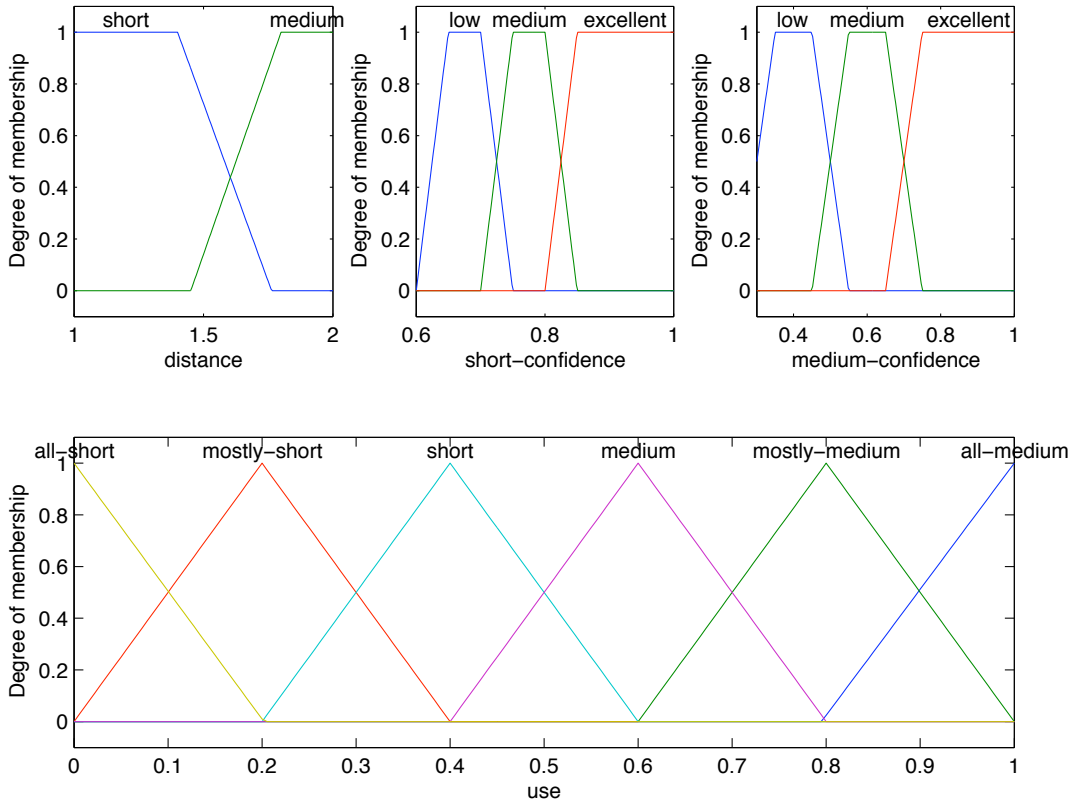


Figure C.2: Sensor Fusion: Fuzzy sets corresponding to three input and one output quantities. (*Top-Left*) Fuzzy concept of medium and short distance from the satellite. Medium range vision module can reliably track the target when the distance from the target is "medium." Similarly, the optimum operational distance for the short range vision module is "short." (*Top-Middle*) Fuzzy performance of the short range vision module and (*Top-Right*) fuzzy performance of the medium range vision module. Output variable "use" determines the weight associated with the target pose estimation from the short range module.

Rule 5: if distance is medium **and** medium-confidence is excellent **then** use is all-medium

Rule 6: if distance is medium **and** medium-confidence is not excellent **and** short-confidence is excellent **then** use is medium

Rule 7: if distance is short **and** short-confidence is not excellent **and** medium-confidence is not excellent **then** use is mostly-short

Rule 8: if distance is medium **and** short-confidence is not excellent **and** medium-confidence is not excellent **then** use is mostly-medium

Appendix D

Quaternion Representation for Rotations

Every rotation can be expressed as a right handed rotation of $-2\pi < \theta < 2\pi$ radians about an arbitrary axis $\mathbf{a}$. We associate with this rotation a unit quaternion

$$\mathbf{q} = \begin{bmatrix} cos(\frac{\theta}{2}) \\ \mathbf{a}sin(\frac{\theta}{2}) \end{bmatrix}. \tag{D.1}$$

Representing rotations as quaternions eliminates the well-known problem of singularities associated with the Euler angle representation. The rotation matrix associated with a unit quaternion $[w, x, y, z]^T (= [w, \mathbf{v}^T]^T$, where $\mathbf{v} = [x, y, z]^T)$ is

$$\begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix}. \tag{D.2}$$

Equations (D.1) and (D.2) can be used to convert from the matrix representation of a rotation to its quaternion representation, and back. We define the power operator to represent scaling in rotation

$$\begin{bmatrix} cos(\frac{\theta}{2}) \\ \mathbf{a}\sin(\frac{\theta}{2}) \end{bmatrix}^{\alpha} = \begin{bmatrix} cos(\frac{\alpha\theta}{2}) \\ \mathbf{a}\sin(\frac{\alpha\theta}{2}) \end{bmatrix}, \tag{D.3}$$

where $\alpha \in [0, 1]$. Here, $\mathbf{q}^{\alpha}$ describes a series of rotations starting from the null rotation (for $\alpha = 0$) and ending at the complete rotation $\mathbf{q}$ (for $\alpha = 1$). Quaterions $\mathbf{q}$ and $-\mathbf{q}$ represent

the same rotations and are called *antipodal* points: $-\mathbf{q}$ is really $\mathbf{q}$ rotated through $\theta - 2\pi$. Quaternions allow us to compose rotations through quaternion multiplication, which is defined as

$$\mathbf{q}_1\mathbf{q}_2 = \begin{bmatrix} w_1 \\ \mathbf{v_1} \end{bmatrix} \begin{bmatrix} w_2 \\ \mathbf{v_2} \end{bmatrix} = \begin{bmatrix} (w_1 w_2 - \mathbf{v}_1.\,\mathbf{v}_2) \\ (w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \end{bmatrix}. \tag{D.4}$$

Then the inverse of a quaterion is

$$\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}^{-1} = \begin{bmatrix} w \\ -x \\ -y \\ -z \end{bmatrix}. \tag{D.5}$$

A 3D vector $\mathbf{p}$ can be rotated through a rotation represented by a quaterion $\mathbf{q}$ by quaterion multiplication

$$\begin{bmatrix} 0 \\ \mathbf{p}' \end{bmatrix} = \mathbf{q} \begin{bmatrix} 0 \\ \mathbf{p} \end{bmatrix} \mathbf{q}^{-1}, \tag{D.6}$$

where $\mathbf{p}'$ is the rotated vector. Quaternions are the representation of choice when interpolating between two rotations. Given the starting and ending quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$, we can interpolate from the former to the latter by

$$\mathbf{q}(\alpha) = (\mathbf{q}_2\mathbf{q}_1^{-1})^{\alpha}\mathbf{q}_1, \tag{D.7}$$

where $\mathbf{q}(\alpha)$ is the interpolated rotation, $\mathbf{q}_1\mathbf{q}_2^{-1}$ is the transition rotation from $\mathbf{q}_1$ to $\mathbf{q}_2$, and $\alpha \in [0, 1]$ is the interpolation parameter. $\mathbf{q}(0) = \mathbf{q}_1$ and $\mathbf{q}(1) = \mathbf{q}_2$. To ensure that $\mathbf{q}$ falls on the short route from $\mathbf{q}_1$ to $\mathbf{q}_2$, we must use the antipodal quaterion of the transition quaterion when the w of the transition quaterion is negative. This method is called spherical interpolation or *slerp* and it is the preferred method for rotation interpolation. For more information about quaternions see [Shoemake 1985; Hearn and Baker 1996; Weisstein].

Bibliography

AGRE, P., AND CHAPMAN, D. 1987. Pengi: An implementation of a theory of activity. In *Proceedings of American Association of Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA.

AKNINE, S., PINSON, S., AND SHAKUN, M. F. 2004. An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems 8*, 1 (Jan), 5–45.

AKYILDIZ, I., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. A survey on sensor networks. *IEEE Communications Magazine 40*, 8 (Aug), 102–114.

ALLEN, P., TIMCENKO, A., YOSHIMI, B., AND MICHELMAN, P. 1993. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation 9*, 2 (Apr), 152–165.

AMBROS-INGERSON, J., AND STEEL, S. 1988. Integrating planning execution and monitoring. In *Proc. of the seventh national conference on artificial intelligence (AAAI 88)*.

ARKIN, R. C., RISEMAN, E. M., AND HANSON, A. R. 1987. ArRA: An architecture for vision-based robot navigation. In *Proceedings of the DARPA Image Understanding Workshop*.

ARKIN, R. C., FUJITA, M., TAKAGI, T., AND HASEGAWA, R. 2001. Ethological modeling and architecture for an entertainment robot. In *Proc. of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)*, IEEE, Seoul, South Korea, 453–458.

ARKIN, R. C. 1988. Homeostatic control for a mobile robot: dynamic replanning in hazardous environments. In *Proc. of SPIE, Vol 1007, Mobile Robots III*, W. J. Wolfe, Ed., 407–+.

ARKIN, R. C. 1990. Integrating behavioural, perceptual and world knowledge in reactive navigation. *Journal of robotics and autonomous systems(1-2), June 1990 6*.

ARKIN, R. C. 1992. AuRA: A hybrid reactive/hierarchical robot architecture. In *Proc. of the IEEE Int. Conf. on Robotics and Automation, Workshop on Architecture for Intelligent Control Systems*.

ARKIN, R. C. 1998. *Behavior-Based Robotics*. The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts.

ATAMTURK, A., NEMHAUSER, G., AND SAVELSBERGH, M. 1995. A combined lagrangian, linear programming and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics 1*, 247–259.

BACKES, P., LONG, M., AND STEELE, R. 1993. The modular telerobot task execution system for space telerobotics. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA' 93)*, vol. 3, 524–530.

BAR-NOY, A., GUHA, S., NAOR, J., AND SCHIEBER, B. 2002. Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing 31*, 2, 331–352.

BAR-SHALOM, Y., AND LI, X.-R. 1998. *Estimation and tracking: principles, techniques, and software*, 2nd edition ed. No. ISBN: 096483121X. Storrs.

BARTON, D. 2004. *Radar System Analysis And Modeling*. No. ISBN: 1580536816. Artech House, October.

BERTAMINI, F., BRUNELLI, R., LANZ, O., ROAT, A., SANTUARI, A., TOBIA, F., AND XU, Q. 2003. Olympus: An ambient intelligence architecture on the verge of reality. In *Proc. 12th International Conference on Image Analysis and Processing*, 139–145.

BIDDLE, P., ENGLAND, P., PEINADO, M., AND WILLMAN, B. 2003. The darknet and the future of content distribution. In *Proc. of the 2nd ACM Workshop on Digital Rights Managements*, Springer, New York, NY, USA, 155–176.

BIRMAN, K., CONSTABLE, B., HAYDEN, M., HICKEY, J., KREITZ, C., RENESSE, R. V., RODEH, O., AND VOGELS, W. 2000. The horus and ensemble projects: Accomplishments and limitations. In *Proc. DARPA Information Survivability Conference & Exposition (DISCEX '00)*, IEEE Computer Society, Los Alamitos, CA, USA, vol. 1, 149–161.

BIRREN, F. 1976. *Color Perception in Art*. Van Nostrand Reinhold Company.

BLUMBERG, B. M. 1994. Action selection in hamsterdam: Lessons from ethology. In *Proceedings of the 3rd International Conference on the Simulation of Adaptive Behaviour*, The MIT Press, Cambridge, MA, Brighton.

BLUMBERG, B. M. 1997. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. PhD thesis, Massachusetts Institute of Technology.

BRAITENBERG, V. 1984. *Vehicles*. MIT Press, Cambridge MA.

BROOKS, R. A. 1986. Achieving Artificial Intelligence through building robots. AI Memo 899, MIT, Cambridge, MA, May.

BROOKS, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics & Automation, March 1986, Also, MIT AIM 864, Sept.1985 - draft form RA-2*, 1.

BROOKS, R. A. 1986. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, vol. RA-2 (1).

BROOKS, R. A. 1990. The behavior language; user's guide. Technical Report A. I. MEMO 1227, Massachusetts Institute of Technology, A.I. Lab., Cambridge, Massachusetts, Apr.

BROOKS, R. A. 1990. A robot that walks: Emergent behaviors from a carefully evolved network. In *Artificial Intelligence at MIT, Expanding Frontiers*, P. H. Winston and S. A. Shellard, Eds. MIT Press, Cambridge, Massachusets.

BROOKSHIRE, J., SINGH, S., AND SIMMONS, R. 2004. Preliminary results in sliding autonomy for assembly by coordinated teams. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, 706–711.

BUHMANN, J., BURGARD, W., CREMERS, A. B., FOX, D., HOFMANN, T., SCHNEIDER, F., STRIKOS, J., AND THRUN, S. 1995. The mobile robot rhino. *AI Magazine 16*, 2, 31–38.

BURGARD, W., CREMERS, A. B., FOX, D., HAHNEL, D., LAKEMEYER, G., SCHULZ, D., STEINER, W., AND THRUN, S. 1999. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence 114*, 1-2, 3–55.

CAMPBELL, J., GIBBONS, P. B., NATH, S., PILLAI, P., SESHAN, S., AND SUKTHANKAR, R. 2005. Irisnet: An internet-scale architecture for multimedia sensors. In *Proc. of the 13th annual ACM international conference on Multimedia (MULTIMEDIA '05)*, ACM Press, New York, NY, USA, 81–88.

CHANG, J.-H., AND TASSIULAS, L. 2000. Energy conserving routing in wireless adhoc networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 22–31.

CHEVALEYRE, Y., DUNNE, P. E., ENDRISS, U., LANG, J., LEMAITRE, M., MAUDET, N., PADGET, J., PHELPS, S., RODRIGUEZ-AGUILAR, J. A., AND SOUSA, P. 2006. Issues in multiagent resource allocation. *Informatica 30*, 3–31.

CLARK, J., AND FERRIER, N. 1988. Modal control of an attentive vision system. In *Proc. of the Second International Conference on Computer Vision*, 514–523.

COLLINS, C. A., AND WILLIAMS, A. D. 1961. Noise and intermodulation problems in multichannel closed-circuit television systems. *AIEE Transactions (Communication and Electronics) 80* (Nov), 486–491.

COLLINS, R., LIPTON, A., KANADE, T., FUJIYOSHI, H., DUGGINS, D., TSIN, Y., TOLLIVER, D., ENOMOTO, N., AND HASEGAWA, O. 2000. A system for video surveillance and monitoring. Tech. Rep. CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May.

COLLINS, R., LIPTON, A., FUJIYOSHI, H., AND KANADE, T. 2001. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE 89*, 10 (Oct.), 1456–1477.

COLLINS, R., AMIDI, O., AND KANADE, T. 2002. An active camera system for acquiring multi-view video. In *Proc. International Conference on Image Processing*, 517–520.

COMANICIU, D., RAMESH, V., AND MEER, P. 2000. Real-time tracking of non-rigid objects using mean shift. In *Proc. of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 00)*, vol. 2, 142–151.

COMANICIU, D., BERTON, F., AND RAMESH, V. 2002. Adaptive resolution system for distributed surveillance. *Real Time Imaging 8*, 5 (Oct), 427–437.

COMANICIU, D. 2003. An algorithm for data-driven bandwidth selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 2, 281–288.

CONNELL, J. 1992. SSS: A hybrid architecture applied to robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*.

CONWAY, L., LESSER, V. R., AND CORKILL, D. G. 1983. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine 4*, 3, 15–33.

COOMBS, D., AND BROWN, C. 1991. Cooperative gaze holding in binocular vision. *IEEE Control Systems Magazine 11*, 4 (Jun), 24–33.

COOMBS, D., AND BROWN, C. 1993. Real-time binocular smooth pursuit. *International Journal of Computer Vision 11*, 2, 147–164.

COSTELLO, C. J., AND WANG, I.-J. 2005. Surveillance camera coordination through distributed scheduling. In *Proc. 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC '05)*, 1485–1490.

COSTELLO, C. J., DIEHL, C. P., BANERJEE, A., AND FISHER, H. 2004. Scheduling an active camera to observe people. In *Proc. 2nd ACM International Workshop on Video Surveillance and Sensor Networks*, ACM Press, New York, NY, 39–45.

CRAMTON, P., SHOHAM, Y., AND STEINBERG, R., Eds. 2006. *Combinatorial Auctions*. The MIT Press.

CUCCHIARA, R., GRANA, C., PICCARDI, M., AND PRATI, A. 2003. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Analysis 25*, 10, 1337–1442.

DAVINIC, N., ARKUS, A., CHAPPIE, S., AND GREENBERG, J. 1988. Cost-benefit analysis of on-orbit satellite servicing. *Journal of Reducing Space Mission Cost 1*, 1 (March), 27–52. rosa-coco, http://dx.doi.org/10.1023/A:1009955909481.

DAVIS, R., AND SMITH, R. G. 1983. Negotiation as a metaphore for distributed problem solving. *Aritificial Intelligence 20*, 1 (Jan), 63–109.

DAVIS, L., DEMENTHON, D., DICKINSON, S., AND VEATCH, P. 1992. Algorithms for road navigation. In *Vision-Based Navigation*, I. Masaki, Ed. Springer-Verlag, New York, NY, USA, 83–110.

DE HAAN, G., AND BELLERS, E. 1998. Deinterlacing–an overview. *Proceedings of the IEEE 86*, 9 (September), 1839–1857.

DESOUZA, G. N., AND KAK, A. C. 2002. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 2 (Feb), 237–267.

DEVARAJAN, D., RADKE, R. J., AND CHUNG, H., 2006. Distributed metric calibration of ad-hoc camera networks. To appear in ACM Transactions on Sensor Networks.

DIAS, M. B., AND STENTZ, A. 2002. Opportunistic optimization for market-based multi-robot control. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS02)*, 2714–2720.

DICKINSON, S., AND DAVIS, L. 1988. An expert vision system for autonomous land vehicle road following. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 826–831.

DICKINSON, S., AND DAVIS, L. 1990. A flexible tool for prototyping alv road following algorithms. *IEEE Transactions on Robotics and Automation 6*, 2 (Feb), 232–242.

DICKINSON, S., STEVENSON, S., AMDUR, E., TSOTSOS, J., AND OLSSON, L. 1993. Integrating task-directed planning with reactive object recognition. *Proc. of SPIE, Vol. 2055, Intelligent Robotics and Computer Vision XII: Algorithms and Techniques* (Aug), 212–224.

DOBSON, G. 1984. Scheduling independent tasks on unrelated processors. *Journal of the ACM*, 13, 705–716.

DU, J., LEUNG, J.-T., AND WONG, C. 1992. Minimizing the number of late jobs with release time constraints. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 11, 97–107.

ELGAMMAL, A., HANVOOD, D., AND DAVIS, L. 2000. Nonparametric model for background subtraction. In *Proc. European Confernece on Computer Vision (ECCV'00)*, 751–767.

ENDO, Y., AND ARKIN, R. C. 2000. Implementing tolman's schematic sowbug: Behavior-based robotics in 1930's.

ESTRIN, D. L., BORRIELLO, G., COLWELL, R. P., FIDDLER, J., HOROWITZ, M., AND KAISER, W. J. 2001. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. National Academy Press, Washington, DC, USA.

FAGERER, C., DICKMANNS, D., AND DICKMANNS, E. 1994. Visual grasping with long delay time of a free floating object in orbit. *Autonomous Robots 1*, 53–68.

FIAT, A., AND WOEGINGER, G. J., Eds. 1998. Online Algorithms, The State of the Art, vol. 1442 of *Lecture Notes in Computer Science*, Springer.

FIEGUTH, P., AND TERZOPOULOS, D. 1997. Color-based tracking of heads and other mobile objects at video frame rates. In *Proc. of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '97)*, IEEE Computer Society, Washington, DC, USA, 21–??

FIKES, R. E., AND NILSSON, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence 5*, 2.

FIRBY, R. 1989. *Adaptive Execution in Complex Dynamics Worlds*. PhD thesis, Computer Science Department, Yale University.

FIRBY, R. 1992. Building symbolic primitives with continuous control routines. In *Proc. of the First International Conference on AI Planning Systems*, 62–69.

FIRBY, R. 1994. Task networks for controlling continuous processes. In *Artificial Intelligence Planning Systems*.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6, 381–395.

FOLEY, J., VAN DAM, A., FEINER, S., AND HUGHES, J. K. 1990. *Computer Graphics: Principles and Practice*. Addison-Wesley, Boston, MA.

FORESTI, G. L., AND MICHELONI, C. 2003. A robust feature tracker or active surveillance of outdoor scenes. *Electronic Letters on Computer Vision and Image Analysis*, 1, 21–34.

FORSYTH, D. A., AND PONCE, J. 2002. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference.

FUNGE, J., TU, X., AND TERZOPOULOS, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Proc. ACM SIGGRAPH 99*, A. Rockwood, Ed., 29–38.

GANDHI, T., AND TRIVEDI, M. M. 2004. Calibration of a reconfigurable array of omnidirectional cameras using a moving person. In *Proc. 2nd ACM International Workshop on Video Surveillance and Sensor Networks*, ACM Press, New York, NY, 12–19.

GAREY, M. R., AND JOHNSON, D. S. 1978. "strong" npcompleteness results: Motivation, examples, and implications. *Journal of the ACM 25*, 3, 499–508.

GAT, E. 1992. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of National Conference on Artificial Intelligence*.

GAVRILA, D. M. 1999. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding 73*, 1, 82–89.

GEORGEFF, M., AND LANSKY, A. 1987. Reactive reasoning and planning. In *Proc. of the AAAI-87*, 677–682.

GERKEY, B., AND MATARI, M. 2004. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research 23*, 9, 939–954.

GIACOMO, G., REITER, R., AND SOUTCHANSKI, M. 1998. Execution monitoring of high-level robot programs. In *Principles of Knowledge Representation and Reasoning*.

GIBBONS, P. B., KARP, B., KE, Y., NATH, S., AND SESHAN, S. 2003. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing 2*, 4 (Oct–Dec), 22–33.

GILLETT, R., GREENSPAN, M., AND L. HARTMAN, E. DUPUIS, D. T. 2001. Remote operation with supervised autonomy (rosa). In *Proc. 6th International Conference on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2001)*.

GIVAN, R., CHONG, E., AND CHANG, H. 2002. Scheduling multiclass packet streams to minimize weighted loss. *Queueing Systems: Theory and Application 41*, 3 (July), 241–270.

GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K., AND KAN, A. H. G. R. 1997. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics 5*, 287–326.

GREENSPAN, M., AND JASIOBEDZKI, P. 2002. Pose determination of a free-flying satellite. In *Motion Tracking and Object Recognition (MTOR02)*.

GURTUNA, O., 2003. Emerging space markets: Engines of growth for future space activities. `www.futuraspace.com/EmergingSpaceMarkets_fact_sheet.htm`.

HAMPAPUR, A., PANKANTI, S., SENIOR, A., TIAN, Y.-L., BROWN, L., AND BOLLE, R. 2003. Face cataloger: Multi-scale imaging for relating identity to location. In *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance*, 13–21.

HAMPAPUR, A., BROWN, L., CONNELL, J., EKIN, A., HAAS, N., LU, M., MERKL, H., AND PANKANTI, S. 2005. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. *IEEE Signal Processing Magazine 22*, 2 (Mar), 38–51.

HARITAOGLU, I., HARWOOD, D., AND DAVIS, L. S. 1998. W4s: A real-time system detecting and tracking people in 2 1/2d. In *Proc. European Conference on Computer Vision (ECCV98)*, 877–892.

HAYATI, S., VOLPE, R., BACKES, P., BALARAM, J., WELCH, R., IVLEV, R., THARP, G., PETERS, S., OHM, T., PETRAS, R., AND LAUBACH, S. 1997. The rocky 7 rover: a mars sciencecraft prototype. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'97)*, vol. 3, 2458–2464.

HE, T., KRISHNAMURTHY, S., STANKOVIC, J. A., ABDELZAHER, T., LUO, L., STOLERU, R., YAN, T., GU, L., HUI, J., AND KROGH, B. 2004. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys '04: Proc. of the 2nd international conference on Mobile systems, applications, and services*, ACM Press, New York, NY, USA, 270–283.

HEARN, D., AND BAKER, M. P. 1996. *Computer Graphics: C Version, 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall.

HHNEL, D., BURGARD, W., AND LAKEMEYER, G. 1998. Golex: Bridging the gap between logic (golog) and a real robot. In *Proc. of the 22nd Annual German Conference on Advances in Artificial Intelligence (KI-98)*, O. Herzog and A. Gnter, Eds., vol. 1504 of *Lecture notes in Artificial Intelligence*, 165–176.

HILL, J., AND PARK, W. 1979. Real time control of a robot with a mobile camera. In *Proc. of the 9th International Symposium on Industrial Robots*, 233–246.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. 2000. System architecture directions for networked sensors. *SIGOPS Oper. Syst. Rev. 34*, 5, 93–104.

HORSWILL, I. 1995. Visual routines and visual search: A real-time implementation and an automata-theoretic analysis. In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*, 56–63.

HU, W., TAN, T., WANG, L., AND MAYBANK, S. 2004. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics (Part C) 34*, 3 (Aug), 334–352.

HUANG, H.-Y., KHENDRY, A., AND ROBERTAZZI, T. 2002. Layernet: a self-organizing protocol for small ad hoc networks. *IEEE Transactions on Aerospace and Electronic Systems 38*, 2 (Apr), 378–387.

HULL, C. 1943. *Principles of Behavior*. Appleton-Century-Crofts, New York.

IHLER, A., FISHER, J., MOSES, R., AND WILLSKY, A. 2004. Nonparametric belief propagation for self-calibration in sensor networks. In *Proc. Third International Symposium on Information Processing in Sensor Networks*, 225–233.

INTANAGONWIWAT, C., GOVINDAN, R., ESTRIN, D., HEIDEMANN, J., AND SILVA, F. 2003. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking 11*, 1 (Feb.), 2–16.

JASIOBEDZKI, P., GREENSPAN, M., AND ROTH, G. 2001. Pose determination and tracking for autonomous satellite capture. In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space (i-SAIRAS 01)*.

JASIOBEDZKI, P., GREENSPAN, M., ROTH, G., NG, H., AND WITCOMB, N. 2002. Video-based system for satellite proximity operations. In *7th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2002)*.

JASIOBEDZKI, P. 1993. Active image segmentation using a camera and a range finder. In *Proc. of SPIE, Vol. 1964, Application of Machine Vision and Artificia Intelligence*, K. L. Boyer and L. Stark, Eds., 92–99.

JAVED, O., RASHEED, Z., ALATAS, O., AND SHAH, M. 2003. Knight: a real time surveillance system for multiple and non-overlapping cameras. In *Proc. International Conference on Multimedia and Expo (ICME03)*, vol. 1, 649–652.

JENKIN, M., BAINS, N., BRUCE, J., CAMPBELL, T., DOWN, B., JASIOBEDZKI, P., JEPSON, A., MAJARAIS, B., MILIOS, E., NICKERSON, B., SERVICE, J., TERZOPOULOS, D., TSOTSOS, J., , AND WILKES, D. 1994. Ark: Autonomous mobile robot for an industrial environment. In *Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS '94)*, 1301–1308.

KAHN, J. M., KATZ, R. H., AND PISTER, K. S. J. 1999. Next century challenges: mobile networking for "smart dust". In *MobiCom '99: Proc. of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, ACM Press, New York, NY, USA, 271–278.

KANG, J., COHEN, I., AND MEDIONI, G. 2003. Multi-views tracking within and across uncalibrated camera streams. In *Proc. First ACM SIGMM International Workshop on Video Surveillance*, ACM Press, New York, NY, 21–33.

KARLSSON, N., DI BERNARDO, E., OSTROWSKI, J., GONCALVES, L., PIRJANIAN, P., AND MUNICH, M. 2005. The vslam algorithm for robust localization and mapping. In *Proc. of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, 24–29.

KARMANN, K.-P., AND VON BRANDT, A. 1990. *Time Varying Image Processing and Moving Object Recognition*, vol. 2. Elsevier Science Publishers, Amsterdam, The Netherlands, ch. Moving Object Recognition Using an Adaptive Background Memory.

KARUPPIAH, D. R., ZHU, Z., SHENOY, P., AND RISEMAN, E. M. 2001. A fault-tolerant distributed vision system architecture for object tracking in a smart room. In *Second International Workshop on Computer Vision Systems (ICVS 2001)*, vol. 2095 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, Vancouver, BC, Canada, Jul, 201–219.

KASAI, T., ODA, M., AND SUZUKI, T. 1999. Results of the ets-7 mission - rendezvous docking and space robotics experiments. In *Proc. of the Fifth International Symposium Artificial Intelligence, Robotics and Automation in Space (ISAIRAS '99)*.

KENNEDY, L. C. F., 2006. Orbital express space operations architecture. `http://www.darpa.mil/tto/programs/oe.htm` (accessed on 5 Oct 2006).

KETTNAKER, V., AND ZABIH, R. 1991. Bayesian multi-camera surveillance. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 99).*, 2, Ed., 253–259.

KHAN, S., AND SHAH, M. 2003. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 10 (Oct.), 1355–1360.

KLIR, G. J., AND YUAN, B. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall.

KOLLER, D., WEBER, J., HUANG, T., MALIK, J., OGASAWARA, G., RAO, B., AND RUSSELL, S. 1994. Towards robust automatic traffic scene analysis in real-time. In *Proc. International Conference on Pattern Recognition (ICPR94)*, 126–131.

KORTENKAMP, D., BONASSO, R. P., AND MURPHY, R., Eds. 1998. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. AAAI Press, Menlo Park, CA, USA.

KRAGIC, D., AND CHRISTENSEN, H. I. 2002. Survey on visual servoing for manipulation. Tech. Rep. ISRN KTH/NA/P–02/01–SE CVAP259, Computer Vision and Active Perception Laboratory, Centre for Autonomous Systems, Numerical Analysis and Computer Science, Kungl Tekniska Hgskolan, Stockholms Universitet, Jan.

KULKARNI, P., GANESAN, D., AND SHENOY, P. 2005. The case for multi–tier camera sensor networks. In *NOSSDAV '05: Proc. of the international workshop on Network and operating systems support for digital audio and video*, ACM Press, New York, NY, USA, 141–146.

KULKARNI, P., GANESAN, D., SHENOY, P., AND LU, Q. 2005. Senseye: a multi-tier camera sensor network. In *MULTIMEDIA '05: Proc. of the 13th annual ACM international conference on Multimedia*, ACM Press, New York, NY, USA, 229–238.

KUO, F. F. 1995. The aloha system. *ACM SIGCOMM Computer Communication Review 25*, 1, 41–44. Special twenty-fifth anniversary issue. Highlights from 25 years of the Computer Communication Review.

LAIRD, J. E., AND ROSENBLOOM, P. S. 1990. Integrating execution,planning,learning in SOAR for external environments. vol. 2.

LANDZETTEL, K., ALBU-SCHFFER, A., PREUSCHE, C., REINTSEMA, D., REBELE, B., AND HIRZINGER, G. 2006. Robotic on-orbit servicing - dlr's experience and perspective. In *Proc. (IROS 06)*, 8 Pages (Electronic).

LEE, Y.-B., YOU, B.-J., AND LEE, S.-W. 2001. A real-time color-based object tracking robust to irregular illumination variations. In *Proc. IEEE International Conference on Robotics and Automation (ICRA01)*, vol. 2, 1659–1664.

LESSER, V. R., AND CORKILL, D. D. 1981. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics SMC-11*, 1 (Jan), 81–96.

LESSER, V., AND ERMAN, L. 1980. Distributed interpretation: A model and experiment. *IEEE Transactions on Computers C-29*, 12 (Dec), 1144– 1163.

LESSER, V. R. 1991. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics 21*, 6 (Nov/Dec), 1347–1362.

LEVESQUE, H., AND REITER, R. 1998. High-level robotic control: Beyond planning. a position paper. In *AIII 1998 Spring Symposium: Integrating Robotics Research: Taking the Next Big Leap*.

LEVESQUE, H. J., REITER, R., LESPÉRANCE, Y., LIN, F., AND SCHERL, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming 31*, 1-3, 59–83.

LEVIS, P., MADDEN, S., GAY, D., POLASTRE, J., SZEWCZYK, R., WOO, A., BREWER, E., AND CULLER, D. 2004. The emergence of networking abstractions and techniques in tinyos. In *Proc. the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 1–14.

LIM, S.-N., DAVIS, L. S., AND ELGAMMAL, A. 2003. A scalable image-based multi-camera visual surveillance system. In *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance*, IEEE Computer Society, Washington, DC, USA, 205–212.

LING, T., AND SHROFF, N. 1996. Scheduling real-time traffic in ATM networks. In *Proc. IEEE Infocom*, 198–205.

LIU, M., AND JASIOBEDZKI, P. 2002. Behaviour based visual servo controller for satellite capture. In *Proc. 7th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2002)*.

LO, B., AND VELASTIN, S. 2001. Automatic congestion detection system for underground platforms. In *Proc. ISIMP2001*, 158–161.

LORENZ, K. 1973. *Foundations of Ethology*. Springer-Verlag, New York.

LUDLOW, A. 1976. The behavior of a model animal. *Behavior 58*, 1–2.

LYONS, D. M., AND ARBIB, M. A. 1989. A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics and Automation 5*, 3 (Jun), 280–293.

MADSEN, C. B., AND CHRISTENSEN, H. I. 1997. A viewpoint planning strategy for determining true angles on polyhedral objects by camera alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 2, 158–163.

MAES, P. 1990. Situated agents can have goals. MIT Press, Cambridge, MA, USA, 40–71.

MALIS, E., CAHUMETTE, F., AND BOUDET, S. 1998. Positioning a coarse-calibrated camera with respect to an unknown object by 2d1/2 visual servoing. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA98)*, vol. 1, 1352–1359.

MALLETT, J. 2006. *The Role of Groups in Smart Camera Networks*. PhD thesis, Program of Media Arts and Sciences, School of Architecture, Massachusetts Institute of Technology.

MARCENARO, L., OBERTI, F., FORES TI, G., AND REGAZZONI, C. 2001. Distributed architectures and logical-task decomposition in multimedia surveillance systems. *Proc. of the IEEE 89*, 10 (Oct), 1419–1440.

MARCH, J., AND SIMON, H. 1958. *Organizations*. John Wiley & Sons.

MARINAKIS, D., DUDEK, G., AND FLEET, D. 2005. Learning sensor network topology through Monte Carlo expectation maximization. In *Proc. IEEE Intl. Conf. on Robotics and Automation*.

MARKATOS, E. P. 2002. Tracing a large-scale peer to peer system: An hour in the life of gnutella. In *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '02)*, IEEE Computer Society, Washington, DC, USA, 65–74.

MATARIC, M. J. 1992. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation 8*, 3 (Jun), 304–312.

MATIJEVIC, J., AND SHIRLEY, D. 1997. The mission and operation of the mars pathfinder microrover. *Control Engineering Practice 5*, 6 (Jun), 827–835.

MEZOUAR, Y., AND CHAUMETTE, F. 2000. Path planning in image space for robust visual servoing. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)*, vol. 3, 2759–2764.

MICHELONI, C., FORESTI, G., AND SNIDARO, L. 2005. A network of co-operative cameras for visual surveillance. In *IEE Proc. Vision, Image and Signal Processing*, vol. 152, 205–212.

MIDDLETON, R., WALTZ, D., AND SCHROCK, S., Eds. 1984. Satellite Servicing Technology Development Missions.

MINSKY, M. 1985. *The Society of Mind.* Simon and Schuster, New York, NY.

MODI, P. J., SHEN, W.-S., TAMBE, M., AND YOKOO, M. 2006. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence 161*, 1–2 (Mar), 149–180. Elsevier.

MONICA NICOLESCU, M. J. M. 2001. Learning and interacting in human-robot domains. *Special Issue of IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 31*, 5 (Sep), 419–430.

MURTHY, C., AND MANOJ, B. 2004. *Ad Hoc Wireless Networks Architectures and Protocols*. Prentice Hall.

NASA, J. P. L., 2004. Mars exploration rover mission home. `marsrovers.nasa.gov` (accessed on 5 Oct 2006).

NASA, J. S. C., 2005. Space educators' handbook home page. `vesuvius.jsc.nasa.gov/er/seh/` (accessed on 5 Oct 2006).

NICOLESCU, M., AND MATARIC, M. J. 2002. A hierarchical architecture for behavior-based robots. In *Proc. First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 227–233.

NILSSON, N. J. 1984. Shakey the robot. Tech. Rep. 323, Artificial Intelligence Center. SRI International, Menlo Park, CA.

OLIVER, N. M., ROSARIO, B., AND PENTLAND, A. 2000. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (??), 831–843.

OPPELT, U. 1995. New possibilities for video applications in the security field. In *Proc. IEEE 29th Annual 1995 International Carnahan Conference on Security Technology*, 426–435.

ORWELL, J., REMAGNINO, P., AND JONES, G. 2000. From connected components to object sequences. In *Proc. of the First IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS'00)*, 72–79.

OTSU, N. 1979. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics 9*, 1, 62–66.

PAVLIDIS, I., MORELLAS, V., TSIAMYRTZIS, P., AND HARP, S. 2001. Urban surveillance systems: From the laboratory to the commercial world. *Proc. of IEEE (Special Issue on Video Communications, Processing, and Understanding for Third Generation Surveillance Systems) 89*, 10 (Oct), 1478–1497.

PAYTON, D. W. 1990. Internalized plans: A representation for action resources. *International Journal of Robotics Systems 6*, 1-2 (June).

PEARSON, J. K., AND JEAVONS, P. G. 1997. A survey of tractable constraint satisfaction problems. Tech. Rep. CSD-TR-97-15, Royal Holloway, University of London, July.

PEDERSINI, F., SARTI, A., AND TUBARO, S. 1999. Accurate and simple geometric calibration of multi-camera systems. *Signal Processing 77*, 3, 309–334.

PERLIN, K., AND GOLDBERG, A. 1996. IMPROV: A system for scripting interactive actors in virtual worlds. In *SIGGRAPH 96 Conference Proceedings*, Addison Wesley, H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH. held in New Orleans, Louisiana, 04-09 August 1996.

PETERSON, L., AND PETERSON, M. 1959. Short-term retention of individiual verbal items. *Journal of Experimental Psychology 58*, 3 (September), 193–198.

PICCARDI, M. 2004. Background subtraction techniques: a review. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 3099–3104.

POLITES, M. 1998. An assessment of the technology of automated rendezvous and capture in space. Tech. Rep. NASA/TP-1998-208528, Marshall Space Flight Center, AL.

POMERLEAU, D. A. 1997. Efficient training of arificial neural networks for autonomous navigation. *Neural Computation 3*, 88–97.

PORIKLI, F. 2003. Inter-camera color calibration by cross-correlation model function. In *Proc. of the IEEE International Conference on Image Processing (ICIP03)*, vol. 2, 133–136.

PUN, T. 1980. A new method for gray-level picture thresholding using the entropy of the histogram. *Signal Processing 2*, 223–237.

QURESHI, F., AND TERZOPOULOS, D. 2005. Surveillance camera scheduling: A virtual vision approach. In *Proc. Third ACM International Workshop on Video Surveillance and Sensor Networks*, 131–139.

QURESHI, F., AND TERZOPOULOS, D. 2005. Towards intelligent camera networks: A virtual vision approach. In *Proc. The Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS05)*, 177–184.

QURESHI, F., AND TERZOPOULOS, D. 2006. Surveillance camera scheduling: A virtual vision approach. *ACM Multimedia Systems Journal 12* (Dec), 269–283. Special Issue on "Multimedia Surveillance Systems".

QURESHI, F., AND TERZOPOULOS, D. 2006. Virtual vision and smart camera networks. In *Working Notes of the International Workshop on Distributed Smart Cameras (DSC 2006)*, B. Rinner and W. Wolf, Eds., 62–66. Held in conjunction with The 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006).

QURESHI, F. Z., TERZOPOULOS, D., AND GILLETT, R. 2004. The cognitive controller: A hybrid, deliberative/reactive control architecture for autonomous robots. In *Innovations in Applied Artificial Intelligence. 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert System (IEA/AIE 2004)*, Springer-Verlag, Ottawa, Canada, B. Orchard, C. Yang, and M. Ali, Eds., vol. 3029 of *Lecture notes in Artificial Intelligence*, 1102–1111.

QURESHI, F. Z., TERZOPOULOS, D., AND JASIOBEDZKI, P. 2004. A cognitive vision system for space robotics. In *Applications of Computer Vision Workshop. European Conference of Computer Vision*, Czech Technical University, Prague, Czech Republic, Clabian, Smutny, and Stanke, Eds., 120–128.

QURESHI, F., MACRINI, D., CHUNG, D., MACLEAN, J., DICKINSON, S., AND JASIOBEDZKI, P. 2005. A computer vision system for spaceborne safety monitoring. In

Proc. 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space.

QURESHI, F., TERZOPOULOS, D., AND JASEIOBEDZKI, P. 2005. Cognitive vision for autonomous satellite rendezvous and docking. In *Proc. IAPR Conference on Machine Vision Applications*, 314–319.

RABIE, T., AND TERZOPOULOS, D. 2000. Active perception in virtual humans. In *Vision Interface (VI 2000)*, 16–22.

RABIE, T., SHALABY, A., ABDULHAI, B., AND EL-RABBANY, A. 2002. Mobile vision-based vehicle tracking and traffic control. In *Proc. of the IEEE 5th International Conference on Intelligent Transportation Systems (ITSC 2002)*, 13–18.

REITER, R. 2001. *Knowledge in Action—Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Cambridge Massachusetts.

REMAGNINO, P., ORWELL, J., GREENHILL, D., JONES, G., AND MARCHESOTTI, L. 2001. *Multimedia Video Based Surveillance Systems: Requirements, Issues and Solutions*. No. ISBN/ISSN 0-7923-7927-6. Kluwer Academic Publishers, ch. An Agent Society For Scene Interpretation, 108–117.

REYNOLDS, C. W. 1987. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics 21*, 4 (July). Description of Reynold's bottom-up system for the study of flocking behavior.

ROBERTS, L. 1965. Machine perception of 3-D solids. In *Optical and Electro-Optical Information Processing*, MIT Press, J. Trippit, D. Berkowitz, L. Chapp, C. Koester, and A. Vanderburgh, Eds., 159–197.

ROTENSTEIN, A. M. 2003. *Supporting Deliberation within Behavior-Based Systems*. Master's thesis, Department of Computer Science, York University.

ROTH, G., AND WHITEHEAD, A. 2000. Using projective vision to find camera positions in an image sequence. In *Vision Interface (VI 2000)*, 87–94.

SACCHI, C., REGAZZONI, C. S., AND DAMBRA, C. 1999. Use of advanced video surveillance and communication technologies for remote monitoring of protected sites. In *Advanced Video-Based Surveillance Systems*, C. S. Regazzoni, G. Gabri, and G. Vernazza, Eds. Kluwer, Norwell, MA, USA, 154–164.

SANTUARI, A., LANZ, O., AND BRUNELLI, R. 2003. Synthetic movies for computer vision applications. In *Proc. 3rd IASTED International Conference: Visualization, Imaging, and Image Processing (VIIP 2003)*, no. 1, 1–6.

SCHENKER, P. S. 1988. Nasa research and development for space telerobotics. *IEEE Transactions on Aerospace and Electronic Systems 24*, 5 (Sep), 523–534.

SCOTT, D. W. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization*. No. 0-471-54770-0. Wiley, August. non-parametric density models.

SE, S., LOWE, D., AND LITTLE, J. 2001. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proc. of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)*, vol. 2, 2051–2058.

SEITNER, F. H., AND HANBURY, A. 2006. Fast pedestrian tracking based on spatial features and colour. In *Proc. of the Computer Vision Winter Workshop 2006 (CVWW'06)*, Czech Society for Cybernetics and Informatics, Prague, Czech Republic, O. Chum and V. Franc, Eds., 105–110.

SELLNER, B., HIATT, L., SIMMONS, R., AND SINGH, S. 2006. Attaining situational awareness for sliding autonomy. In *Proc. 1st Annual Conference on Human-Robot Interaction (HRI2006)*.

SGALL, J. 1998. Online scheduling: A survey. In *On-Line Algorithms: The State of the Art, Lecture Notes in Computer Science*. Springer-Verlag, 192–231.

SHAO, W., AND TERZOPOULOS, D. 2005. Autonomous pedestrians. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 19–28.

SHAO, W., AND TERZOPOULOS, D. 2005. Environmental modeling for autonomous virtual pedestrians. In *Proc. SAE Digital Human Modeling Symposium*.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. In *Proc. of the 12th annual conference on Computer graphics and interactive techniques (SIGGRAPH'85)*, ACM Press, New York, NY, USA, 245–254.

SIEBEL, N. T. 2003. *Designing and Implementing People Tracking Applications for Automated Visual Surveillance*. PhD thesis, Dept. of Computer Science. The University of Reading., UK.

SIGAL, L., SCLAROFF, S., AND ATHITSOS, V. 2004. Skin color-based video segmentation under time-varying illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 7, 862–877.

SINOPOLI, B., SHARP, C., SCHENATO, L., SCHAFFERT, S., AND SASTRY, S. 2003. Distributed control applications within sensor networks. *Proc. of the IEEE 91*, 8 (Aug), 1235–1246.

SMITH, R. G. 1980. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transctions on Computers C-29*, 12 (Dec), 1104–1113.

HTTP://WWW.SPACEREF.COM. 1997. *Mobile Servicing System (MSS) to User (Generic) Interface Control Document*.

STAUFFER, C., AND GRIMSON, W. 1999. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, 246–252.

STAUFFER, C., AND GRIMSON, W. E. L. 2000. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8, 747–757.

STEWART, D. B., AND KHOSLA, P. K. 1992. Real-time scheduling of sensor-based control systems. In *RealTime Programming*, W. Halang and K. Ramamritham, Eds. Pergamon Press Inc., Tarrytown, NY, USA, 139–144.

STILLMAN, S., TANAWONGSUWAN, R., AND ESSA, I. 1998. A system for tracking and recognizing multiple people with multiple cameras. Tech. Rep. GIT-GVU-98-25, Georgia Institute of Technology, GVU Center.

SWAIN, M. J., AND BALLARD, D. H. 1991. Color indexing. *International Journal of Computer Vision 7*, 1 (Nov), 11–32.

TERZOPOULOS, D., AND RABIE, T. 1997. Animat vision: Active vision in artificial animals. *Videre: Journal of Computer Vision Research 1*, 1 (Sept.), 2–19.

TERZOPOULOS, D., TU, X., AND GRZESZCZUK, R. 1994. Artificial fishes with autonomous locomotion, perception, behavior, and learning in a simulated physical world. In *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems* $ArtificialLifeIV$, MIT Press, Cambridge, MA, R. A. Brooks and P. Maes, Eds.

TERZOPOULOS, D. 2003. Perceptive agents and systems in virtual reality. In *Proc. 10th ACM Symposium on Virtual Reality Software and Technology*, 1–3.

THRUN, S., BEETZ, M., BENNEWITZ, M., BURGARD, W., CREMERS, A., DELLAERT, F., FOX, D., AHNEL, D., ROSENBERG, C., ROY, N., SCHULTE, J., AND SCHULZ, D. 2000.

Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research 19*, 11 (Nov), 972–999.

THUILOT, B., MARTINET, P., CORDESSES, L., AND GALLICE, J. 2002. Position based visual servoing: Keeping the object in the field of vision. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'02)*, 1624–1629.

TOLMAN, E. C. 1939. Prediction of vicarious trial and error by means of the schematic sowbug. *Psychological Review 46*.

TOYOMA, K., KRUMM, J., BRUMITT, B., AND MEYERS, B. 1999. Wallflower: Principles and practice of background maintainence. In *Proc. International Conference on Computer Vision (ICCV99)*, 255–261.

TRIVEDI, M., HUANG, K., AND MIKIC, I. 2000. Intelligent environments and active camera networks. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 804–809.

TSOTSOS, J., VERGHESE, G., DICKINSON, S., JENKIN, M., JEPSON, A., MILIOS, E., NUFLO, F., STEVENSON, S., BLACK, M., METAXAS, D., S. CULHANE, Y. Y., AND MANN, R. 1998. PLAYBOT: A visually-guided robot for physically disabled children. *Image and Vision Computing 16*, 4, 275–292.

TSOTSOS, J. 1995. On behaviorist intelligence and the scaling problem. *Artificial Intelligence 75*, 2 (Jun), 135–160.

TSOTSOS, J. 1997. Intelligent control for perceptually attentive agents: The s* proposal. *Robotics and Autonomous Sytems 21*, 1, 5–27.

TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94*, ACM Press, Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH. ISBN 0-89791-667-0.

TYRRELL, T. 1993. *Computational Mechanisms for Action Selection*. PhD thesis, Centre for Cognitive Science, University of Edinburgh.

WALLACE, G. 1991. The JPEG still picture compression standard. *Communications of the ACM 34*, 4 (April), 30–44.

WALTER, W. G. 1950. An imitation of life. *Scientific American 182*, 5.

WALTER, W. G. 1951. A machine that learns. *Scientific American 185*, 2.

WALTER, W. G. 1953. *The Living Brain*. W. W. Norton, New York, NY.

WEISS, L. E. 1984. *Dynamic Visual Servo Control cf Robots: An Adaptive Image-Based Approach*. PhD thesis, Department of Electrical and Computer Engineering, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA.

WEISSTEIN, E. W. Quaternion. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Quaternion.html.

WERTZ, J., AND BELL, R. 2003. Autonomous rendezvous and docking technologies—status and prospects. In *SPIE's 17th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*.

WILCOX, B., MATTHIES, L., GENNERY, D., COOPER, B., NGUYEN, T., LITWIN, T., MISHKIN, A., AND STONE, H. 1992. Robotic vehicles for planetary exploration. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'92)*, vol. 1, 175–180.

WILSON, S. W. 1990. The animat path to ai. In *Proc. of the first international conference on simulation of adaptive behavior on From animals to animats*, MIT Press, Cambridge, MA, USA, 15–21.

WREN, C., AZARBAYEJANI, A., DARRELL, T., AND PENTLAND, A. 1997. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 7 (July), 780–785.

XING, G., WANG, X., ZHANG, Y., LU, C., PLESS, R., AND GILL, C. 2005. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks (TOSN) 1*, 1, 36–72.

YAGI, Y., SHOUYA, K., AND YACHIDA, M. 2000. Environmental map generation and egomotion estimation in a dynamic environment for an omnidirectional image sensor. In *Proc. of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000)*, vol. 4, 3493–3498.

YANG, B., AND GARCIA-MOLINA, H. 2003. Designing a super-peer network. In *Proc. 19th International Conference on Data Engineering (ICDE'03)*, IEEE Computer Society, Los Alamitos, CA, USA, 49–60.

YOKOO, M. 2001. *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Springer-Verlag, Berlin, Germany.

ZHAO, F., SHIN, J., AND REICH, J. 2002. Information-driven dynamic sensor collaboration for tracking applications. In *IEEE Signal Processing Magazine*, vol. 19. Mar., 61–72.

ZHAO, F., LIU, J., LIU, J., GUIBAS, L., AND REICH, J. 2003. Collaborative signal and information processing: An information directed approach. *Proceedings of the IEEE 91*, 8, 1199–1209.

ZHOU, X., COLLINS, R. T., KANADE, T., AND METES, P. 2003. A master-slave system to acquire biometric imagery of humans at distance. In *Proc. First ACM SIGMM International Workshop on Video Surveillance*, ACM Press, New York, NY, 113–120.

ZHU, Z., RAJASEKAR, K. D., RISEMAN, E. M., AND HANSON, A. R. 2000. Panoramic virtual stereo vision of cooperative mobile robots for localizing 3d moving objects. In *Proc. IEEE Workshop on Omnidirectional Vision (OMNIVIS'00)*, IEEE Computer Society, Los Alamitos, CA, USA, 29.

ZILBERSTEIN, S. 1996. Resource-bounded reasoning in intelligent systems. *ACM Computing Surveys (CSUR) 28*, 4, 15.