2008

Development of an Active Vision System for the Remote Identification of Multiple Targets

Christopher D. W. Ward
Western University

Development of an Active Vision System for the Remote Identification of Multiple Targets

(Spline title: Active Vision System for Remote Target Identification)

(Thesis format: Monograph)

by

Christopher D. W. Ward

Graduate Program in Engineering Science

Department of Mechanical and Materials Engineering

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Engineering Science

Faculty of Graduate Studies

The University of Western Ontario

London, Ontario, Canada

Abstract

This thesis introduces a centralized active vision system for the remote identification of multiple targets in applications where the targets may outnumber the active system resources. Design and implementation details of a modular active vision system are presented, from which a prototype has been constructed. The system employs two different, yet complimentary, camera technologies. Omnidirectional cameras are used to detect and track targets at a low resolution, while perspective cameras mounted to pan-tilt stages are used to acquire high resolution images suitable for identification. Five greedy-based scheduling policies have been developed and implemented to manage the active system resources in an attempt to achieve optimal target-to-camera assignments. System performance has been evaluated using both simulated and real-world experiments under different target and system configurations for all five scheduling policies. Parameters affecting performance that were considered include: target entry conditions, congestion levels, target to camera speeds, target trajectories, and number of active cameras. An overall trend in the relative performance of the scheduling algorithms was observed. The Least System Reconfiguration and Future Least System Reconfiguration scheduling policies performed the best for the majority of conditions investigated, while the Load Sharing and First Come First Serve policies performed the poorest. The performance of the Earliest Deadline First policy was seen to be highly dependent on target predictability.

Keywords: modular, omnidirectional, fisheye, active camera, vision system, multi-camera system, surveillance, resource scheduling.

Acknowledgements

Foremost, I would like to express my gratitude to my supervisor, Dr. Michael Naish. His expertise and guidance has played a critical role in the development and refinement of the ideas presented in this work. The understanding and patience that he has shown has added considerably to my graduate experience.

I have been fortunate to work with many helpful and talented people during my time here at The University of Western Ontario. In particular, I would like to recognize Dave Lunn, for his continuous technical and moral support; and Eugen Porter, for his flawless electrical advice and countless consults. I would also like to thank my colleagues, Andrew Lyle and Anita Jain, whose friendship, encouragement and advice have been invaluable.

I thank my friends and family, Philip Ward; Thomas Ward; Adrian and Anna Wolters; Zachary Taylor; Miranda Restorick; Alex Pieprzak; Vulindlela and Katherine Ndlovu; Jesse Mark; Kristyn Leitch; and George Konstantinopoulos, for always being there for me — a constant source of support, inspiration and encouragement.

I would especially like to thank my parents, Joseph and Carolyn Ward. Their love and support have been unyielding, I doubt that I will ever be able to convey my appreciation fully.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

2D	Two-dimensional
3D	Three-dimensional
CAD	Computer aided design
DOF	Depth of field
EDF	Earliest deadline first
FCFS	First come first serve
FLSR	Future least system reconfiguration
FOV	Field of view
GUI	Graphical user interface
HSV	Hue, saturation, value
IEEE	Institute of Electrical and Electronics Engineers
LSR	Least system reconfiguration
NEMA	National Electrical Manufacturers Association
PID	Proportional-Integral-Derivative
PTZ	Pan-tilt-zoom
UML	unified modeling language

Chapter 1

Introduction

1.1 Motivation

A renewed interest in surveillance techniques and methodologies has been sparked by recent world events. The miniaturization of vision technologies, in conjunction with the ready availability of computing power, has seen an increase in vision systems being deployed to perform complex surveillance tasks. Many of these surveillance tasks require high levels of image detail that are usually difficult to acquire using cameras having fixed perspectives. For example, in unconstrained environments without choke points (restricted entry and exit points), it can be extremely challenging to collect images of people's faces for identification. An expensive solution to this problem would be to litter the environment with thousands of cameras to ensure that a high resolution image is captured of every person.

A more practical solution has been found using pan-tilt-zoom (PTZ) cameras to vary the perspective of the images collected. This enables a much smaller number of cameras to provide the video needed for tasks such as activity monitoring and target identification. It has also been shown that, in many applications, the addition of omnidirectional camera(s) can greatly enhance the performance of the active PTZ system. The perspective PTZ cameras have an inherently narrow field of view (FOV), producing a rather large "blind spot" that limits the cameras' ability to detect and observe moving targets. Alternatively, by virtue of their large FOV, omnidirectional cameras can be used to detect and track multiple targets at a lower resolution.

Although PTZ vision systems are already used in many surveillance applications, the majority of these systems are deployed using operator control. One or more humans watch video from the camera network and then direct cameras to view a desired target or region. This method of deployment becomes impractical in high throughput applications, where the resource management and control strategies used can significantly impact the overall performance and robustness of the system.

1.2 Research Objectives

The goal of this thesis is to develop a centralized active vision based solution to the problem of remote identification of multiple targets in high throughput applications, where there are more targets than cameras. The development of the identification algorithms themselves is outside of the scope of this thesis; instead, it is simply assumed that high resolution target images must be acquired for identification.

The primary objectives of the thesis are as follows:

1. The development and implementation of a centralized active vision system. The system must be capable of detecting, locating and imaging targets autonomously with a high level of detail.

2. The development of resource scheduling policies. Active system resources must be managed in a manner that maximizes the overall system performance. In most applications, the probability of a successful identification will increase with the number of images that are taken of the object being identified. As such, the objective of a scheduling policy can be broken into two competing goals. One goal is to collect high resolution images of as many targets as possible. The second goal is to collect high resolution images of each target, for as long or as many times as possible.

3. The development of a methodology that, given a choice, can prioritize targets according to observed target behaviours. The ability to prioritize targets greatly enhances the flexibility and robustness of the overall solution.

4. The development and implementation of a simulated system environment, whose behaviour closesly resembles that of the physical system. This environment will be used to evaluate the performance of the developed scheduling policies under different target and system configurations.

The system is envisioned to be applied to the task of remotely identifying people in an unconstrained environment; however, the proposed approach to target identification may be suitable for any application where the targets are of adequate size, and the target motion does not surpass the physical limitations of the system.

1.3 Thesis Outline

This thesis is organized into six chapters. Chapter 2 focuses on the design and implementation details of a modular active vision system. It begins with an overview of related works demonstrating other multi-camera vision systems, followed by a description of the chosen system concept and its theory of operation. The remainder of the chapter outlines the key design features of the physical system modules, and provides details of the implemented hardware and software architectures.

Chapter 3 is dedicated to the topic of resource scheduling. First, relevant scheduling policies are reviewed. The scheduling problem is then defined, followed by a description of five heuristic on-line scheduling policies that have been implemented. The scheduling algorithms for each policy are also detailed.

Chapter 4 provides details of the experimental procedures used in the evaluation of the five scheduling policies. A description of the simulator that has been developed to aid in this evaluation is presented, followed by an outline of the target and system parameters considered in the simulated evaluation. Finally, a description of the experiments that were performed using the real-world system is given.

Results obtained from this evaluation are presented in Chapter 5. Observations and trends from both the simulated and real-world experiments are discussed.

The final chapter gives a summary of the main conclusions drawn from this work. The chapter also highlights recommendations and improvements for future work.

Chapter 2

System Design

This chapter presents the key design features of the active modular omnidirectional vision system that has been developed. The system implemented here is based on a previously developed concept and is composed of two different, yet complementary, camera types that can be integrated in various configurations to provide a robust modular vision system. A review of relevant vision systems is provided in Section 2.1 to reinforce the advantages of the chosen system type. The strengths and improvements of the new design are outlined in Section 2.2, and the rest of the chapter describes the key design features and components of the implemented system in more detail.

2.1 Multi-camera System Review

The idea of integrating multiple cameras into a single robust vision system is not a new idea. A large body of literature exists for multi-camera systems covering a multitude of applications and system configurations. Currently, available camera technology can be decomposed into two types: perspective cameras and omnidirectional cameras. The following subsections elaborate on how these camera types are used in existing systems and highlight their distinctive features. Specifically, system functionality is discussed in the context of two main configurations: centralized and distributed. In a centralized system, the cameras are restricted to a single, localized device. Alternatively, in distributed systems, the cameras are positioned at strategic locations throughout the environment.

2.1.1 Perspective Camera Systems

Ukita and Matsuyama [1] present a real-time cooperative multi-target tracking system comprising a network of Pan Tilt Zoom (PTZ) cameras. The cameras are distributed throughout the environment such that they have overlapping Fields of View (FOV), and all cameras can communicate with one another. Each camera is given the tasks of detecting and tracking target objects. For a target to be detected, it must first pass through the FOV of at least one camera — to this end, cameras that are not tracking a target are set to 'roam' the environment searching for new targets. A similar approach is also taken by Lim *et al.* [2].

Bakhtari *et al.* present a reconfigurable active vision system consisting of a single fixed observation camera and a number of distributed active perspective cameras [3–5]. The system is used to track single or multiple maneuvering targets through its workspace. The same system is also used by Naish to experimentally verify sensor-system planning techniques [6].

2.1.2 Omnidirectional Camera Systems

Omnidirectional camera systems have also been developed to tackle the problems of target localization and tracking. Sogo *et al.* developed a distributed omnidirectional camera network for real-time surveillance and detection [7]. By using multiple omnidirectional cameras, they were able to not only view large regions of the environment, but were also able to eliminate many of the problems associated with target occlusions. Centralized systems composed of two omnidirectional cameras vertically aligned have also been developed [8, 9]. These systems have the ability to see, and therefore to track an object in an almost uninterrupted spherical FOV. Depth perception is also possible where the views overlap, similar to a stereo head. Although these systems are capable of continuously monitoring large areas, they are not well suited to the task of identification as they lack the ability to resolve a high level of detail.

2.1.3 Combined Perspective and Omnidirectional Camera Systems

There has been a recent trend towards developing visual surveillance systems that combine at least one omnidirectional camera and one perspective camera. The omnidirectional camera(s)

allow large regions to be continuously monitored for target detection and tracking, while the perspective cameras enable images to be captured with high levels of detail — these complementary features serve to enhance the overall robustness of the surveillance system. The majority of the existing systems incorporating both perspective and omnidirectional cameras use a distributed configuration; the most common of which uses PTZ cameras to view/track objects of interest, directed by one or more omnidirectional cameras [10–15].

Scotti *et al.* [16] present a centralized surveillance system where a PTZ camera is situated directly below a catadioptric camera. This configuration reduces the complexity of directing the perspective camera to observe the target being seen by the omnidirectional camera; there exists a direct relationship between the target location in the omnidirectional image and the corresponding pan/tilt angles required to bring the target into view. Several other centralized systems with omnidirectional and perspective camera have also been developed [17–19], but these were mostly developed to mimic features of human vision; i.e., having depth perception while simultaneously providing wide peripheral FOV with a detailed central view.

A modular centralized system is presented by Jankovic and Naish [20–22]. Their approach addresses many of the limitations that are evident in the reviewed prior work. It consists of a combination of omnidirectional and perspective camera modules that are arranged vertically. The modularity of the system permits the construction of different system configurations by stacking basic camera modules. This feature allows the system to be tailored to suit specific applications and, under certain configurations, can even provide a spherical FOV. The vertical arrangement of the modules grants each camera module an unobstructed 360° horizontal view of its surroundings, and as in [16], allows the omnidirectional camera(s) to directly guide an active camera to view a target point. The proposed system can be reconfigured with up to three different camera module types: a fisheye camera module, a catadioptric camera module and an active pan/tilt module. A system prototype was constructed using one of each type of module.

2.2 Design Concept

The approach used in this thesis is a refinement of the modular vertically stacked concept used by Jankovic. The modular approach allows the system to be scalable and provides a high level of functionality and flexibility for the surveillance task at hand. The new system design proposed in this thesis addresses some of the limitations of Jankovic's implemented system prototype [20]. First, the overall size of each of the module types is greatly reduced and true modularity has been preserved, thereby increasing the system's scalability. Second, customized motion controllers were developed to again improve the overall system modularity. The controllers were tailored to the motor and encoder setups employed and allow the active modules to be fully self contained. Third, a cable spooling device was designed to allow the active modules to make multiple revolutions while still allowing a direct connection with the active perspective cameras. This allows the video feed to remain digital, thereby reducing the losses incurred by analog to digital conversion, as well as analog transmission. Fourth, the software and hardware architectures were revamped, allowing the system to track and prioritize multiple targets. The new software architecture was also designed to be split among multiple computers to increase its real-time performance. The following subsections will provide a short overview of the system layout and its theory of operation.

2.2.1 System Physical Configuration

The redesigned system makes use of two of the three camera module types implemented by Jankovic: the fisheye camera module and the active pan/tilt module. The inherent size, alignment difficulties and calibration complexity of the catadioptric module rendered it impractical for the current application. Using the two selected module types, a couple of configurations are of interest, as depicted in Figure 2.1. Configuration a) consists of only a single omnidirectional fisheye camera module and is suitable for ceiling mounting typical of most surveillance applications. The dual fisheye camera module shown in configuration b) on the other hand produces an almost spherical FOV. Not only does this increase the extent to which the surrounding environment can be monitored, but when in overlapping views a target's location can be triangulated to aid in guiding the active cameras to the correct viewing orientation.

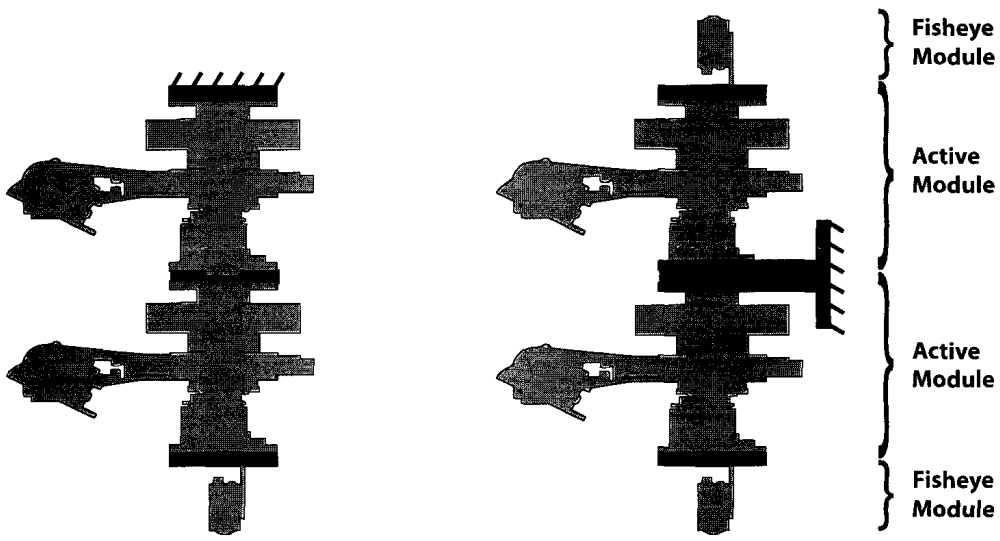


Figure 2.1: Potential system configurations. a) Ceiling mounted — hemispherical FOV. b) Wall mounted — spherical FOV.

2.2.2 Peripherally-Guided Active Vision

As mentioned Section 2.1.2, the omnidirectional camera(s) can be used to guide the active cameras to observe a target of interest. This is accomplished by mapping a point in the omnidirectional image to a corresponding point in the perspective image. The mapping accuracy is dependent on the calibration precision of the camera optics as well as the degree to which the relative camera positions are known. One of the major advantages of the vertically stacked structure is that it greatly simplifies the triangulation and guidance tasks as outlined below.

1. A target is first detected in an omnidirectional view. An epipolar line with azimuth angle ϕ can then be drawn from the image centre to the target point. The radial distance of the point from the image centre can be converted to an angle, θ, with respect to the vertical system axis as described in Section 2.6. Using these two angular quantities, a 3D line can be projected from the camera centre through the target as depicted in Figure 2.2.

2. Since the modules are vertically aligned, the rotational axis of the pan stage passes through the omnidirectional camera centre(s), co-linear with their principal axis. The pan stage can therefore be instructed to rotate the active camera such that it coincides with the radial line drawn through the target point in an omnidirectional image.

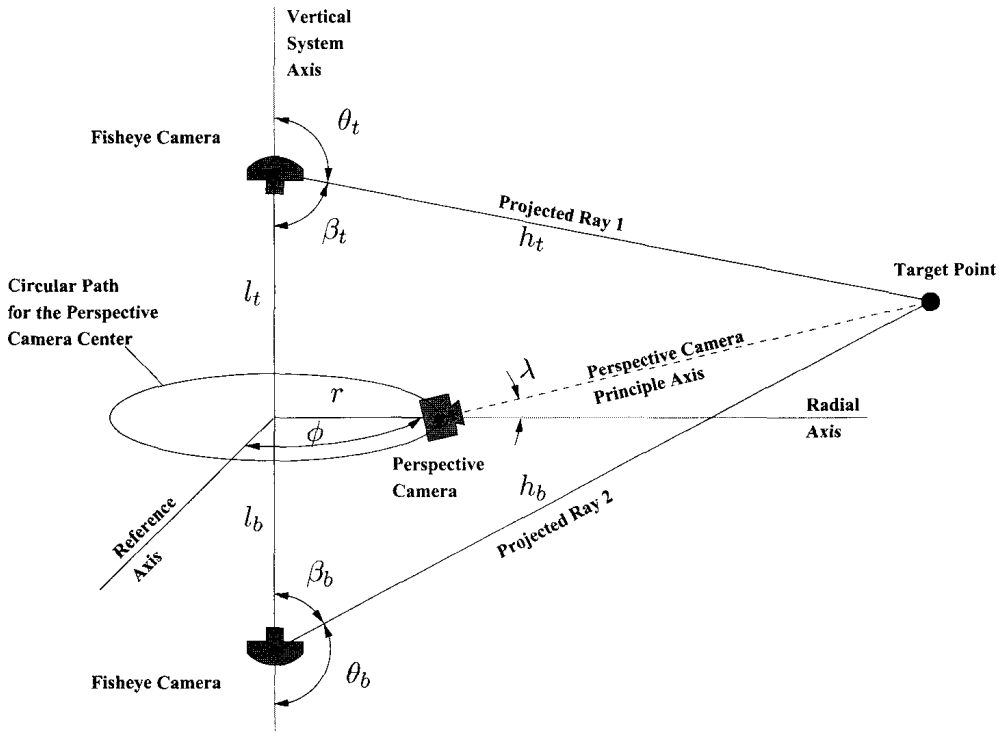


Figure 2.2: Triangulation and system geometry for two fisheye cameras and one perspective camera. Here, l_t, l_b and r are known from system measurements and calibration. θ_t and θ_b are determined from the top and bottom fisheye images respectively. Upon calculating β_t and β_b, the cosine law can be used to calculate h_t and h_b, from which ϕ and λ, the pan and tilt angles required to view the target point, can be found.

3. When only detected in a single omnidirectional view, the active module can tilt its camera such that its principle axis is aligned in the direction of the target as a function of the ray projected by the omnidirectional camera. This provides a good starting point for the active camera to start searching for the desired target, especially when acquiring distant targets or when the active camera is situated close to the omnidirectional module. If the target is not located immediately, the active camera can be made to rotate inward toward the omnidirectional camera until it is located. When detected in more than one omnidirectional camera view, the intersecting rays from the omnidirectional cameras, along with the relative position of the camera centers, can be used to triangulate the target's position and guide the active camera directly to the correct viewing orientation.

2.3 Physical Design

The main objectives in redesigning the physical system were to reduce the size of each of the module types and preserve the modularity of the original concept. A picture of the completed system consisiting of two active modules and two omnidirectional fisheye modules is shown in Figure 2.3. The sizes of the modules were greatly reduced from Jankovic's original prototype design [20]. The active module now stands 0.13 m tall, down from 0.26 m while the height of the fisheye module was reduced from 0.16 m to 0.06 m. Implementation and design details for the two module types are discussed in the following subsections.

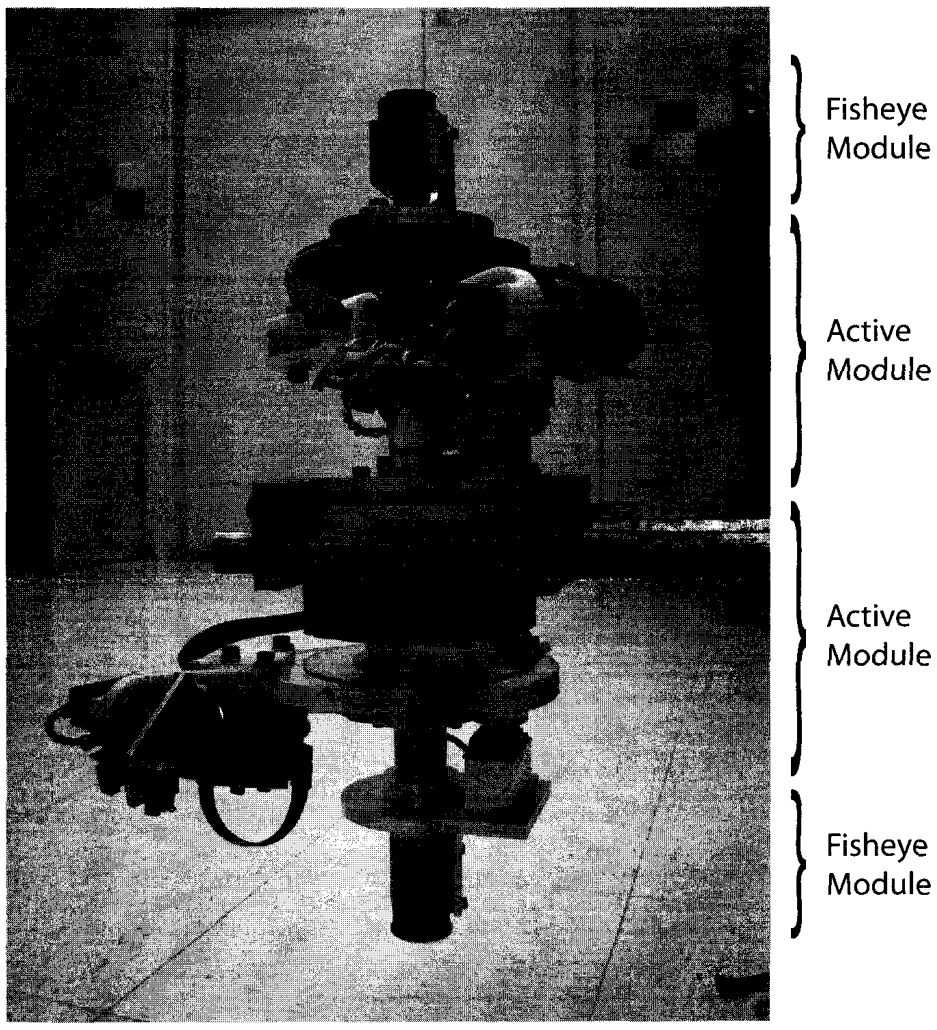


Figure 2.3: System prototype.

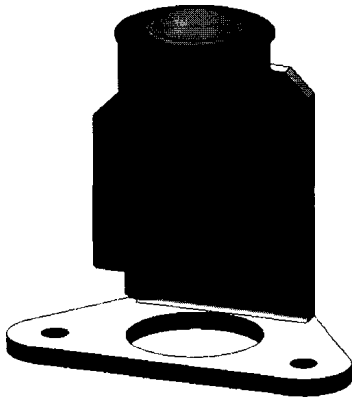


Figure 2.4: Fisheye module.

2.3.1 Fisheye Module

The fisheye module is the simpler of the two module types. Its most important feature is its ability to provide an unobstructed omnidirectional view of its surroundings. To achieve this, the fisheye module must be the last module at either end of the system. Its mechanical design is simple, consisting of an L-type bracket with the appropriate mounting and alignment holes for connecting to other modules. When assembled, the principle axis of the camera is aligned with the vertical system axis. A model of the assembled fisheye module is shown in Figure 2.4.

The camera that was selected for the fisheye module was Point Grey's 14S3C-C Flea2 camera. The 14S3C-C provides good image resolution, up to 1384×1024, can be externally triggered, and comes in a very compact, 29×29×30 mm, package.

To achieve an almost spherical system FOV, a fisheye lens with a FOV greater than 180° is required. A few lenses meeting this requirement are commercially available. Nikon has designed the FC-E8 fisheye lens converter [23], an add-on fisheye lens providing an 183° FOV. Alternatively, Omnitech Robotics offers a fisheye microlens that achieves a 190° FOV when used with a 1/3-inch image sensor [24]. Sunex Inc has also designed the DSL215A, a fisheye microlens that achieves a 185° FOV on a 1/2-inch image sensor [25]. The DSL215A was chosen here because it provided the most practical and cost effective solution. It is a miniature lens that unlike the Nikon lens adapter, requires no alignment and for which there exists a commercially available adapter for the Flea2's C-mount lens holder.

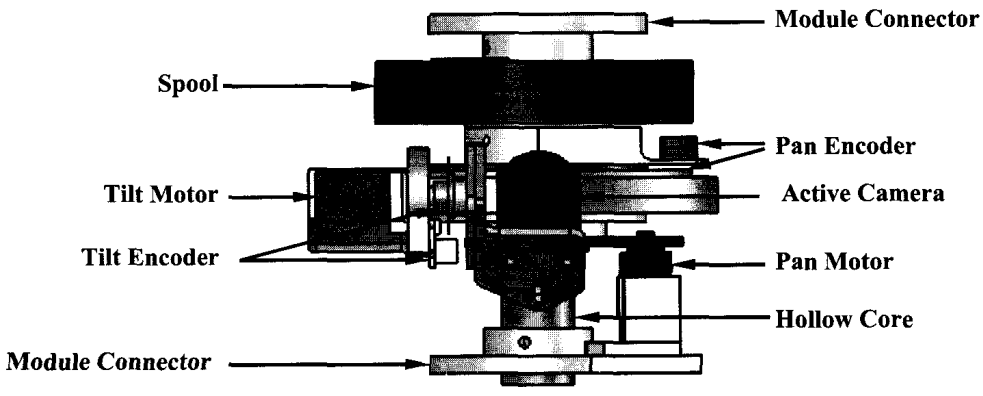


Figure 2.5: Active perspective module.

2.3.2 Active Perspective Module

The design of the active perspective module was dictated by the overall goal of obtaining a truly modular and compact system. The active module layout described by Jankovic was used in this design as it provides the most appropriate solution for the vertically stacked system concept [20]. Using this layout, the camera pans around the module's circumference and tilts about its optical centre. Extending the camera outside the module envelope allows the camera to tilt $\pm 90^{\circ}$ with respect to the horizon and significantly reduces obstruction caused by other system modules. Ensuring that the tilt axis passes through the camera's optical centre constrains the camera's perspective centre to always lie on a circle, the radius of which is fixed by the module geometry. This feature greatly aids in the peripheral guidance and triangulation tasks as is shown in Figure 2.2. Although the layout chosen for the active module resembles that presented by Jankovic on a conceptual level, it differs greatly in its implementation and design. The labeled CAD model in Figure 2.5 is included as a visual reference for the following discussion on the active perspective module's key design features.

2.3.2.1 Connectivity and Structural Design

To ensure modularity of the system, the active perspective module must provide mechanical and electrical connectivity to other system modules, irrespective of the module type. Since the camera rotates around the module circumference, structural support and electric cable routing was pro-

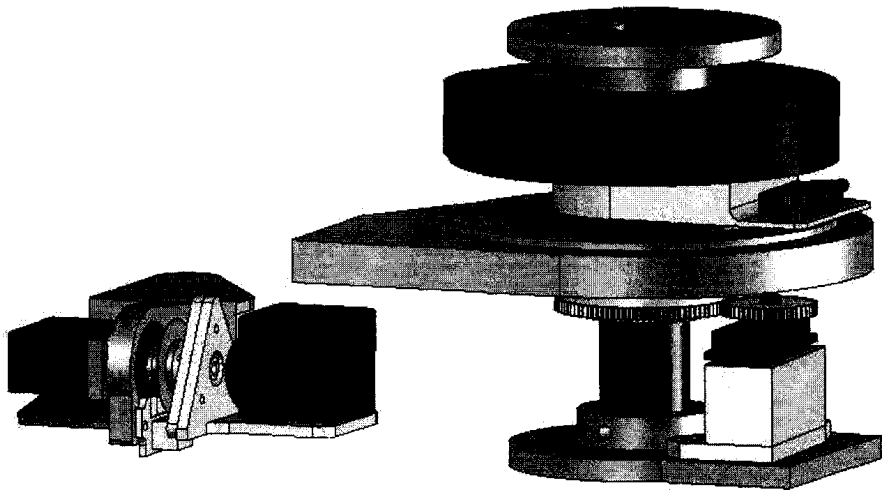


Figure 2.6: Active perspective module showing the tilt-axis subsystem (left) and pan-axis subsystem (right).

vided at the module center. This was achieved using a tubular core that was sized to house the required control and video cables while still providing support for the static and dynamic system loads. Mechanical connection with other modules is established using a removable 5-hole flange.

Direct electrical connection between the pan platform and the rest of the system was achieved through the use of a custom-designed cable spooling mechanism. The spooling approach allows digital video to be streamed across the rotating interface and provides a more simplistic and compact alternative to using an electrical or optical slip-ring; however, the disadvantage of using this approach over a slip-ring design is that spooling is limited to a finite number of platform revolutions, approximately ± 4 revolutions about its neutral, unwound state.

2.3.2.2 Pan Axis Subsystem

The use of a tubular module core prevents the use of an inline drive system to rotate the pan platform. Instead, an offset drivetrain utilizing a set of spur gears was developed. A wide variety of motor styles are available for driving the pan platform, including the two most common types, stepper motors and servo motors. Jankovic employed a stepper motor with a NEMA 23 ($57 \times 57 \times 52$ mm) frame size coupled to a 4:1 ratio gear head to rotate the pan platform in his prototype [20].

Although this setup provided a cost effective, and easily controllable solution, its large size proved prohibitive. Alternatively, Kollmorgen offers a frameless hollow-bore direct drive DC torque motor that could be used as a compact in-line solution to driving the pan stage [26]. Unfortunately, the power requirements, integration complexity and high cost meant that this was not a viable solution. Instead, a Hitec HS5245MG digital RC-servo motor was chosen [27]. The extremely small package ($17 \times 44 \times 31$ mm), high torque (5.5 kg·cm), low power consumption and low cost of this motor were ideally suited for this application.

A further 2:1 speed reduction was gained using a set of spur gears. A large bore 96-tooth gear mounted on the pan platform is driven by a 48-tooth pinion. Readily available self-lubricating 48 pitch Acetal gears with a pressure angle of 20° were used.

Backlash in both the external gear set and the servo's internal drivetrain meant that accurate platform positions could only be obtained with an encoder located on the platform itself. The size constraints imposed on the design, coupled with the size of module's central shaft greatly constrained the encoder selection. Heidenhain has developed the ERA-4000, a bearingless encoder kit with large bore hollow shafts [28]. The ERA-4000 provides great positional accuracy, but its performance is highly dependent on the precision to which the individual components can be aligned. MicroE Systems also offers a line of digital bearingless encoder systems. Their Mercury 1800 encoder system [29] was employed here. It offers a compact, low-profile solution providing high resolution and requiring relatively low alignment precision. A glass grating disk mounted to the platform passes under an encoder head that produces a 2-channel quadrature output with a resolution of 3,276,800 counts per revolution.

2.3.2.3 Tilt Axis Subsystem

The tilt axis subsystem is much simpler than the pan axis subsystem. The same Hitec HS5245MG RC-servo motor as used in the pan drive system is used here to rotate the camera. Since less torque is required to drive the tilt stage, the motor is connected directly to the main shaft without the need for any external gear reduction. A Renco RE201 bearingless optical encoder [30] is also mounted to the main shaft and provides positional feedback with a resolution of $16,384$ counts per revolution. The perspective camera is then attached to the shaft using a custom mounting bracket

that can be adjusted with 6-DOF. This allows the position of the camera to be fine tuned such that the tilt axis passes through the camera's optical centre. Point Grey's 08S2C-C Flea2 camera was chosen for the perspective cameras. The 08S2C-C is a lower resolution model to that used in the omnidirectional fisheye module, providing a maximum pixel resolution of 1024×768.

2.4 Hardware Architecture

A cost effective and truly modular hardware architecture has been developed and implemented. This hardware architecture allows the system to be easily reconfigured and/or expanded. A high level overview of the hardware layout is shown in Figure 2.7 and a description of the key components is given in the following subsections.

2.4.1 Communication

Data transfer between the system hardware and its computer network is accomplished using two different communication standards: video feeds from the active and omnidirectional cameras are transmitted using the IEEE1394b Firewire protocol, while control and positional data is transferred via an RS-232 serial link. A custom 8-bit encoded data transfer protocol was developed for transmission over the RS-232 bus. The protocol is loosely based on the ROBIN transfer protocol, initially developed for use on a RS-485 bus to provide robot component developers with a standardized communication protocol [31].

The number of cameras that can be connected on a single Firewire bus is restricted by the available bandwidth. It was found that when using the IEEE1394b protocol, a combination of either 4 low-resolution cameras, or 2 high-resolution cameras can be connected to the same bus while staying within the bandwidth limitations (800 Mb/sec). A two firewire-bus architecture was implemented for the prototype as shown in Figure 2.7, one bus for the omnidirectional cameras and one for the active cameras. A third bus can easily be added if more than 4 active modules are required.

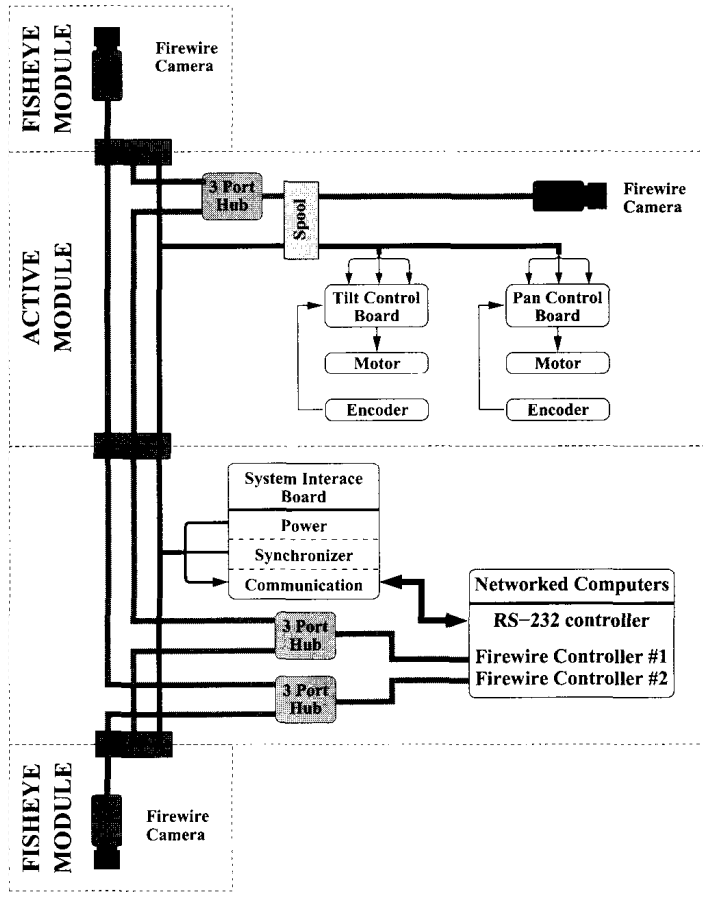


Figure 2.7: Modular hardware architecture.

2.4.2 System Interface Board

A power supply unit, a synchronization unit and a communication interface are all contained within a single system interface board.

1. Power supply unit: each active module is provided with a 5 V DC supply to power its control boards and encoders, as well as a 6 V DC supply for the pan and tilt motors.

2. Synchronization Unit: when triggered, the synchronization unit provides 5 V square pulse with a pulse-width of 1 ms. The pulse is used by the omnidirectional cameras to trigger an image capture and by the active modules' control boards to capture their respective encoder positions.

3. Communication interface: this is responsible for converting between the high RS-232 voltage levels required by the computers and the reduced voltage levels required by the Universal

asynchronous receiver/transmitter (USART) devices on the active module control boards.

2.4.3 Active Module Control Boards

Pan and tilt control boards were developed to interface with the encoders and drive motors. The customized designs provide a flexible and cost effective motion control solution that is tailored to the encoder/motor setup of each stage type. At a high level, the control boards for each stage type provide the same functionality. First, they can communicate with each other as well as with the system's computer network. This is facilitated by giving each board a unique node address. Second, both board types receive the encoder outputs, implement PID control and provide the stage motor with a Pulse Width Modulated (PWM) control signal. There are, however, some low-level differences between the pan and tilt control boards that are worthy of note.

2.4.3.1 Tilt Stage Control Board

The tilt stage control board is built around an 8-bit RISC ATmega168 microcontroller [32]. The microcontroller is responsible for communicating with the system and decoding system commands, as well as decoding the quadrature encoder input from the tilt encoder and implementing the PID control loop.

2.4.3.2 Pan Stage Control Board

For the pan stage control board, the tasks are split between two ATmega168 microcontrollers. Dual microcontrollers were required to manage the increased bandwidth requirements of the pan encoder. The extremely high resolution of the encoder meant that custom hardware had to be developed to decode its output at high panning speeds. One microcontroller is dedicated to reading the decoded encoder output and implementing the PID control while the second microcontroller is responsible for communicating with the rest of the system.

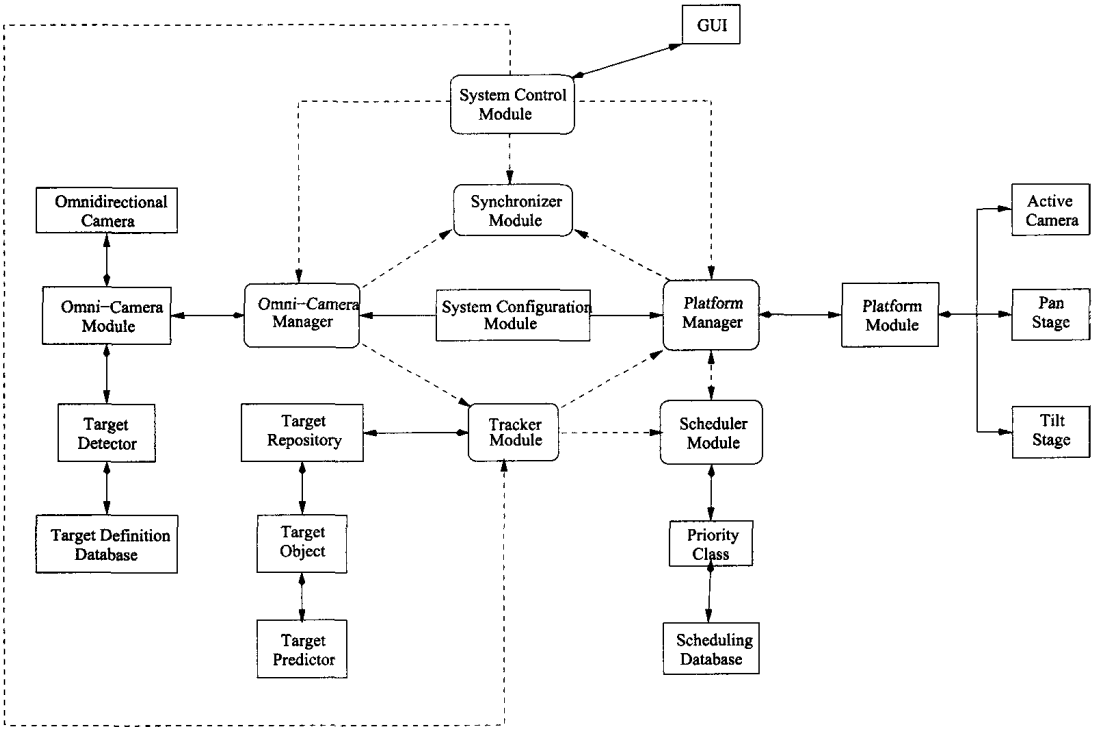


Figure 2.8: Process diagram for the overall software architecture. Boxes with rounded corners indicate separate processes and dashed lines indicate a network connection.

2.5 Software Architecture

For such a highly integrated and automated system to function effectively and achieve real-time performance, it must have a robust and efficient software architecture. This section provides a high level description of the processes and features that make up the system's software.

A modular approach was taken in the design of the software architecture, as is now standard practice in the software industry. This approach was adopted as it usually results in readable and maintainable code and because it can greatly simplify complex designs. The Unified Modeling Language (UML) process diagram in Figure 2.8 provides a high level overview of the static software structure and shows the relationship between the individual modules. In keeping with the adopted system philosophy, the software layout was even further abstracted; instead of running in a single process, the architecture was built around a User Datagram Protocol (UDP) network, allowing core modules to run in separate processes — and even further, on separate computers. This feature maintains the scalability of the system as a whole. A description of the software modules and their key features follow.

2.5.1 Target Detector

For a target to be identified, its presence in the scene must first be detected. A multitude of object detection algorithms have been previously developed, ranging from those using low level image analysis techniques to more complex statistical based algorithms. A good review of such algorithms can be found in [33, 34]. The target detection algorithm implemented in this work uses colour thresholding in the Hue, Saturation, Value (HSV) colour space [35]. This algorithm was chosen as it was quick to implement and easy to tune, and because it allowed multiple target definitions (i.e., colours) to be distinguished simultaneously. By implementing the target detector as a separate class, more complex and target specific detection algorithms can seamlessly be interchanged as required.

2.5.2 Omni-Camera Module

The Omni-Camera Modules provide an interface to the omnidirectional camera Firewire bus. A single module exists for each omnidirectional camera that is connected to the system. The modules are responsible for communicating with their respective cameras and provide a means of initializing and setting camera features. While the system is running, images captured by the cameras are transmitted over the Firewire bus to the corresponding Omni-Camera Module where they are preprocessed and passed to the target detector. The module contains a storage class that stores information for each target found by the target detector in the current image.

2.5.3 Omni-Camera Manager

The Omni-Camera Manager is responsible for creating and overseeing the operation of the Omni-Camera Modules, and for providing a means of communicating with the other system processes. Once a complete capture cycle has finished, the target information is read from each of the Omni-Camera Modules. Calibration parameters and system geometry information loaded from the System Configuration Module are then used to calculate the azimuth and polar angles of the ray projected from the corresponding camera centre through the target point. The ray angles for each target are then passed on to the Tracking Module. Each point is also tagged with its corresponding

omni-camera identifier.

2.5.4 Tracking Module

Once a target's entry into the scene has been detected, the challenge turns to that of tracking it through the scene from one image to the next. The difficulty of this task is greatly amplified in crowded environments where targets will often become partially or fully occluded. As with target detection, target tracking has received a great deal of attention in the research community. A thorough review of some of the more popular tracking algorithms is presented in [36, 37].

The method implemented here is based on the work carried out by Zhengyou Zhang [38]. Predicted positions of previously tracked targets are compared with target positions found in the most current capture. When an observed target position is within a given radius of the predicted position, a match is made and a target repository is updated with the new target position. The radius of the predicted region is determined based on prediction uncertainty. On entering the scene, a target's predicted region is large; the region decreases as the target trajectory is updated. In the case that multiple targets lie in the predicted region, the *best-first search* approach is used, whereby the target closest to the predicted position is taken as a match.

In the case of multiple omnidirectional cameras, tracking is performed separately for each camera. A search is then carried out to identify those targets that are in the overlapping views. If multiple views exist, triangulation is performed for the affected targets and the target repository is again updated.

2.5.5 Scheduling Module

Similar to an airport's air traffic control, the responsibility of the Scheduling Module is to coordinate available resources to achieve a desired task; in this case, assigning targets to active cameras in an effort to maximize target identification. The module takes on a multi-class structure, where by each class is given a different priority level, and in which a class can use a different scheduling algorithm independent of those used by other classes. The Scheduling Module is also updated with the most current synchronized target and platform positions. This multi-class configuration, and access to the current state of the system, provides great flexibility in the types of scheduling

algorithms that can be implemented. The target to camera assignments are then passed on to the Platform Manager which is responsible for guiding and controlling the active perspective camera(s) as described later in Section 2.5.6.

2.5.6 Platform Manager

The platform manager, as its name suggests, is a manager process that is responsible for creating and updating the Platform Modules, as well as maintaining a list of target to active camera assignments. The Platform Manager communicates directly with the Tracking Module and is updated with the most current target positions. This information, along with calibration and system geometry information provided by the System Configuration Module, is used to estimate the pan/tilt viewing angles using the peripheral guidance technique described in Section 2.2.2. The Platform Manager class also acts as a virtual interface to the physical system and is responsible for all communication with the system hardware.

2.5.7 Platform Module

The Platform Module acts as a virtual active perspective module. A container class within the module stores information related to the corresponding physical system module. Measured and desired pan and tilt angles, along with critical system information are all stored here. The Platform Modules also serve as an interface to their respective active cameras. They are responsible for communicating with their respective camera and provide a means of initializing and setting camera features. A PID controller is also implemented in the platform module. Once a platform has been guided by the omnidirectional cameras to the desired pan and tilt angles, the active camera is signaled to start transmitting images. When a target of interest enters the camera's view, control of the active module is relinquished to the platform module to try and bring the target into the image centre.

2.5.8 System Control Module

The System Control Module serves as a master control for the other system processes and provides the user with a system GUI. The module can probe the network to ensure that all other processes

are on-line and provides the user with a means of setting and initializing system and process parameters.

2.5.9 Synchronization Module

While the system is running, the Synchronization Module receives a signal from both the Omni-Camera Manager and the Platform Manager after the completion of each of their respective captures. Once both signals have been received, a command is sent to the physical system that triggers a synchronization pulse. This pulse triggers the omnidirectional cameras to start a new image capture while at the same time triggers the pan/tilt angles to be read from the platform encoders.

2.6 Calibration

A vision system that intends to extract geometric information from its images must first be calibrated to determine a relationship between the 2D images and the associated scene. The inherent difference in the FOVs of the camera technologies used in this system means that different techniques must be used for calibrating the fisheye and active perspective cameras. There are many prior works addressing the issue of camera calibration techniques for both camera types. A good review of perspective camera models and calibration methods can be found in [39–41]. Jankovic and Naish also provide a thorough review of the state of the art of omnidirectional camera calibration and present an integrated technique for calibrating an omnidirectional active vision system [20,22].

Developing a vision system to capture high resolution images of multiple targets for remote identification was the primary objective in this thesis. Providing the system with accurate depth perception for precise target localization, although very much achievable, is outside the scope of this work. As such, a less intense calibration process than that presented in [20] was employed. In this work, the purpose of calibration is to provide a relationship between the location of a point in the 2D omnidirectional images and the corresponding pan/tilt angles for the peripheral guidance task described in Section 2.2.2. The calibration process used to obtain the desired relationship is outlined below.

1. The calibration step determines the centre of the fisheye lens in (image coordinates). By covering the lens with a white cloth, the interface between the lens and its rim can clearly be identified; the imaging portion of the lens appears as a white circle in an otherwise black image. By applying standard circle fitting techniques, the centre of the lens can then be found. Once this centre is known, the polar angle, θ, of the ray projected through a point in the omnidirectional image can be estimated using $r = k \tan \frac{\theta}{2}$, where r is the distance of the point from the image centre and k is a constant, found using the technique described by Bakstein and Pajdla [42].

   Note that more complex lens models that can provide more accurate estimates of the polar angle do exist. These models account for external and internal camera parameters that must be found using more involved calibration procedures [42]. The gain in accuracy is unwarranted for this application.

2. Ideally, the pan stage can be instructed to rotate the active camera such that it coincides with the radial line drawn through the target point in the omnidirectional image; however, construction and assembly imperfections can result in a misalignment of the omnidirectional camera's principle axis and the vertical system axis. Thus, the relationship between the azimuth angle formed by a point in the omnidirectional image and its corresponding active-platform pan angle is achieved using a look up table (LUT). n calibration points and m test points are distributed evenly through out the scene. The azimuth angles, θ_c and θ_t, are extracted from the omndirectional views for each of the calibration and test points respectively. The active cameras are then guided manually to view the calibration points. For each point, the corresponding pan angles, θ_{pan}, are recorded to form a LUT for each platform. Pan angles for each point in the scene can then be found by interpolating linearly between the closest points in the LUT according to Equation 2.1: the grid size of the mapping is then verified using the test points. If the mapping fails for any of the test points, the size of the LUT is increased by adding more calibration points and repeating the above process.

$$\theta_{pan_i} = \theta_{pan_1} + (\theta_{pan_2} - \theta_{pan_1}) \frac{(\theta_{c_i} - \theta_{c_1})}{(\theta_{c_2} - \theta_{c_1})} \tag{2.1}$$

2.7 Summary

A centralized active vision system has been developed and a prototype constructed. The implemented system is based on a previously developed concept and is composed of two different, yet complementary, camera types that can be integrated in various configurations to provide a robust modular vision system. Improvements over the original design have been realized, including: miniaturization of the individual modules, improved system modularity, and a direct connection between the system's active cameras and the control and data acquisition computers. A modular software architecture has also been developed that provides the system with the ability to detect and track multiple targets. The active system resources are managed using a Scheduling Module that attempts to maximize the number of targets that can be viewed. Several scheduling policies used by the Scheduling Module are presented next in Chapter 3.

Chapter 3

Resource Scheduling

As stated in Section 1.2, the objective of this thesis is to develop a system for the remote identification of multiple targets in high throughput applications. The design and implementation details of the hardware and software used by the system have already been described in Chapter 2. The focus of this chapter is the development and implementation of scheduling policies that will maximize the use of the system's active resources. A review of the related previous work is first given in Section 3.1. The scheduling problem is then defined in Section 3.2, with the remainder of the chapter dedicated to the discussion of the implemented scheduling schemes.

3.1 Active Vision Resource Scheduling Review

Scheduling has been studied extensively in many forms and from many viewpoints, both in the on-line and off-line paradigms. Of interest in this thesis is the study of on-line scheduling schemes in applications with *a priori* unknown tasks. Inspired mostly by applications in practical computing, the field has developed into a theoretical area of study. A large amount of work has been done developing scheduling policies for processing packets in network routers [43, 44] as well as assigning jobs in single and multi-processor computing applications [45–48]. On-line scheduling schemes have also been adopted for reactive robotic systems [49–51]. A good overview of on-line scheduling in general can be found in [52, 53]. Of particular interest in this thesis is the use of on-line scheduling schemes as applied to vision based surveillance applications. Growing interest

in automated surveillance has lead to the development of several resource scheduling techniques for active vision systems. A review of the techniques related to this work follows below.

Costello *et al.* [13] approach the problem of scheduling a single active camera to observe people with the objective of maximizing the number of people that can be identified. A system comprised of an active PTZ camera and fixed wide-FOV camera was used. The key contributions from their work are a discussion of the problems presented by scheduling the active camera and the analysis of four multi-class greedy scheduling policies. These include: First Come First Serve (FCFS), Earliest Deadline First (EDF), random, and Current Minloss Throughput Optimal (CMTO) policies. The FCFS policy schedules targets in the order in which they enter the scene. Using the EDF scheme, people are observed in order of their estimated time remaining in the scene — those expected to leave the scene first are observed first. The CMTO policy was first introduced in [54] and attempts to minimize the total weight of the people unobserved before some future time horizon. Weights are assigned to people based on factors such as their estimated exit time and the number of times that they have been viewed. A comparative simulation-based evaluation was performed for the four policies, considering two objectives. It was shown that for the sub-goal of maximizing the number of people observed, the EDF algorithm performed the best under all tested trajectory types. The CMTO was shown to have to poorest performance when the trajectories were highly unpredictable, but showed significant performance increases as the trajectory predictability increased. Under the objective of maximizing the data collected of each person, however, the CMTO algorithm outperformed the other policies and the EDF policy showed the poorest performance.

Costello and Wang extend their work to consider the problem of coordinating a camera network for the purpose of identifying people in [14]. They formulate the problem to involve distributed camera networks. The networks consist of fixed, wide-FOV cameras, along with a number of non-overlapping PTZ cameras. By aiming to select which people are to be observed in a coordinated manner, they attempt to maximize the number of people observed over the network of active PTZ cameras. They propose two heuristic approaches to the problem. The first approach uses a localized EDF policy which schedules each camera independent of the global state of the system. A distributed load balancing scheduling policy which introduces inter-camera communication between neighboring pairs is also proposed in their latter approach. People are scheduled for

observation based on the 'congestion' of the areas to which they are heading. An empirical evaluation of the two policies was performed using simulations. It was concluded that the distributed load sharing algorithm outperforms the EDF policy in times of congestion but is comparable at times when people are distributed uniformly across the network.

The problem of scheduling multiple cameras to view pedestrians as they move through a designated area is also addressed by Qureshi *et al.* [55]. Similar to the system described by Costello and Wang [14], their system consists of a number of distributed static, wide-FOV cameras, and active, PTZ cameras. A multi-camera control strategy is presented with the goal of viewing each pedestrian at least once during their stay in the designated area. A single scheduling algorithm is presented in their work in which targets are scheduled using a weighted round-robin scheme with a static First Come First Serve priority policy. Weighting is determined by two factors: the amount of adjustment needed by the PTZ camera to view the pedestrian, and the distance of the pedestrian from the target, with those cameras closest to the target and requiring the least amount of reconfiguration being given the highest weight. Pedestrians are then assigned to the highest weighted 'free' camera using a non-preemptive FCFS policy. The performance of the scheduling algorithm is evaluated in simulation using a novel virtual environment that was presented in their work. The percentage of people observed along with the wait times and processing times of those observed were used in the evaluation. The effects of adding more active cameras to the network and changing the pedestrian behaviour were investigated.

Lim *et al.* propose the idea of including task-specific information into the scheduling process. Their work is motivated by the need for unobstructed images of an object that satisfy both temporal and positional constraints in many surveillance tasks. To accomplish this, they introduce the concept of Task Visibility Intervals or TVIs, and Multiple Task Visibility Intervals or MTVIs [15]. They describe these TVIs as those future time intervals in which objects are unobstructed and satisfy task specific constraints (i.e., resolution, direction of motion, duration). An MTVI is an extension to the aforementioned visibility intervals that allows multiple task specific constraints to be met simultaneously. The main contribution of their earlier work [15, 56] is in the development of the algorithms used to construct the time intervals that satisfy these visibility constraints on a per-camera basis. In later work, Lim *et al.* extend their approach to use the MTVIs in an

intelligent control system, where PTZ cameras are scheduled to take video only when task specific requirements can be met [57]. Two non-preemptive scheduling algorithms are presented. The first is a greedy algorithm that iteratively schedules the MTVI that covers the maximum number of unassigned tasks on a per-camera basis. The second scheduling algorithm uses a dynamic programming approach that takes into account the ability of other cameras in the network to cover tasks. The authors then evaluate the performance of the algorithms both in simulation and using a real world system. They show that the dynamic programming approach outperforms the simple greedy algorithm in simulation; however, for the real world experiments, only the greedy algorithm results were reported. This was as due to the greedy algorithm being faster than its dynamic programming counterpart.

Bimbo *et al.* cast the problem as a particular type of dynamic discrete optimization problem in which a Time Dependent Orienteering algorithm is used as a discrete solution to the Traveling Kinetic Salesman Problem [58–60]. Here the optimal camera tour was calculated on-line as a set of static optimization problems accounting for an object's predicted time of exit from the scene, its location and heading, as well as any necessary camera reconfiguration time. The proposed framework was developed for a master-slave camera configuration consisting of a static, wide FOV master camera and an active PTZ slave camera. It was assumed that while an object was in the scene its motion was predictable. The framework was also limited in the number of objects considered in each round of optimization due to its computational complexity. The scheduling policy was evaluated using a Monte Carlo simulation and was shown to outperform a simple EDF policy.

3.2 Problem Formulation

The scheduling algorithms developed and implemented in this work, although fine tuned for use with the centralized system described in Chapter 2, provide a general means of scheduling the active components of a visual surveillance system to observe multiple targets. The general problem formulation for which the algorithms were developed is as follows:

For a system comprising of m active cameras, the goal of a scheduling policy is to distribute

the system resources to view a set of n targets in a coordinated manner such that the total number of viewed targets is maximized. Targets can enter the scene at any time with the system having no *a priori* knowledge of the target's entry location, speed or trajectory through the scene. Let C be the set of active system cameras where $C = \{c_i : i = 1, 2, ..., m\}$. Also, let F be the set of free, or idle, cameras where $F \subset C$. Each target within the scene is contained within a global target set $G = \{g_i : i = 1, 2, ..., n\}$ and those targets not scheduled for viewing belong to U, where $U \subset G$. Multiple targets can be assigned to a single active camera. Each camera then has a unique set of assigned targets, S, that have yet to be viewed. Once a target has been scheduled, it is removed from the set of unscheduled targets, U. Correspondingly, when an assignment is made to an idle camera, it is removed from F. This leads to the following two properties: first, $S \cap U = \emptyset$ must hold true for each camera, and second, if a camera $c_i \in F$ then $S = \emptyset$ for camera c_i.

As mentioned previously in Section 2.5.5, the software scheduling module implements a multi-class structure. Each class within the module is given a priority level, such that those targets belonging to a higher priority class are scheduled before those in a lower priority class. Within a single class, however, targets are scheduled using one of the policies described below in Section 3.3, with each class having its own unscheduled target list, U. A target can be assigned to any class at any time, but cannot belong to more than one class at once, i.e. $U_{\text{class}_i} \cap U_{\text{class}_j} = \emptyset$.

3.3 Algorithm Descriptions

Five on-line scheduling policies are investigated in this thesis. Two of these policies are taken from the work reviewed in Section 3.1, while the remaining three policies are new to this work. The new scheduling schemes provide an intuitive heuristic approach to the resource scheduling problem being addressed. Common to each of the scheduling policies is the underlying use of a non-preemptive greedy algorithm that makes locally optimal decisions at each scheduling step. The greedy approach provides a computationally efficient solution, has been shown to have good real-time performance and scalability as compared to other dynamic programming and optimization techniques [14,57,58] and, in some cases, may provide an optimal solution [61]. The five scheduling schemes are described in the following subsections along with pseudo code of the implemented

Algorithm 3.1 Load Sharing algorithm.

```
for i = 1 : size of U do
  if F ≠ ∅ then
    minReconfiguration = 181°
    for j = 1 : size of F do
      reconfiguration = |ω_{f_j} − ω_{u_i}|
      if reconfiguration < min then
        minReconfiguration = reconfiguration
        f_closest = f_j
    assign target u_i to the free camera f_closest
  else
    minLoad = size of S for camera c_1
    c_LeastLoad = c_1
    for j = 2 : m do
      if size of S for camera c_j < minLoad then
        minLoad = size of S for camera c_j
        c_LeastLoad = c_j
    assign target u_i to camera c_LeastLoad
```

algorithms.

3.3.1 Load Sharing

The Load Sharing algorithm tries to distribute the target load evenly amongst the active cameras. Upon entering the scene, targets are scheduled in a non-preemptive manner in the order in which they enter. Targets are first assigned to the closest free camera. When all cameras have become occupied, targets are scheduled such that they are assigned to the active camera with the fewest number of unidentified, pre-assigned targets (i.e., with the smallest set, S). Pseudo code outlining this approach is shown in Algorithm 3.1.

3.3.2 First Come First Serve (FCFS)

The FCFS scheduling algorithm was proposed by Qureshi *et al.* [55] and uses a round robin approach to assign targets to cameras. A description of the algorithm is given in Algorithm 3.2. Targets are assigned to the closest free camera in the order in which they enter the scene. The 'closeness' of a target refers to the amount of reconfiguration required to bring the target into a camera's FOV. Targets are not assigned at the time of entry, as in the load sharing case, but

Algorithm 3.2 First Come First Serve algorithm.

```
sort U according to entry times
for i = 1 : size of U do
  if F ≠ ∅ then
    minReconfiguration = 181°
    for j = 1 : size of F do
      reconfiguration = |ω_{f_j} − ω_{u_i}|
      if reconfiguration < minReconfiguration then
        minReconfiguration = reconfiguration
        f_closest = f_j
    assign target u_i to the free camera f_closest
```

rather, are scheduled as cameras become available. In their implementation, Qureshi *et al.* also considered the Euclidean distance of the target from the cameras during the scheduling phase; however, this was omitted here due to the centralized nature of the system.

3.3.3 Earliest Deadline First (EDF)

The EDF scheduling algorithm was introduced by Costello *et al.* [13, 14]. As in the FCFS policy, targets are not scheduled as they enter the scene, but rather as active cameras become available; however, instead of being scheduled in the order in which they enter the scene, targets are scheduled in the order in which they are predicted to exit the scene. Those with the earliest expected deadline are scheduled first. The algorithm is shown below in Algorithm 3.3.

Algorithm 3.3 Earliest Deadline First algorithm.

```
for i = 1 : size of U do
  predict exit time t_{exit,u_i}
sort U according to exit times
for i = 1 : size of U do
  if F ≠ ∅ then
    minReconfiguration = 181°
    for j = 1 : size of F do
      reconfiguration = |ω_{f_j} − ω_{u_i}|
      if reconfiguration < minReconfiguration then
        minReconfiguration = reconfiguration
        f_closest = f_j
    assign target u_i to the free camera f_closest
```

Algorithm 3.4 Least System Reconfiguration algorithm.

Start
for $i = 1$: size of F **do**
  if $U \neq \emptyset$ **then**
    minReconfiguration = 181°
    for $j = 1$: size of F **do**
      reconfiguration = $|\omega_{f_i} - \omega_{u_j}|$
      if reconfiguration < minReconfiguration **then**
        minReconfiguration = reconfiguration
        $u_{\text{closest}} = u_j$
for $i = 1$: size of F **do**
  for $j = i + 1$: size of F **do**
    if u_{closest} for $f_i = u_{\text{closest}}$ for f_j **then**
      if $|\omega_{f_i} - \omega_{u_{\text{closest}}}| < |\omega_{f_j} - \omega_{u_{\text{closest}}}|$ **then**
        assign target u_{closest} to free camera f_i
      else
        assign target u_{closest} to free camera f_j
      goto *Start*
for $i = 1$: size of F **do**
  assign target u_{closest} for f_i to camera f_i

3.3.4 Least System Reconfiguration (LSR)

The Least System Reconfiguration algorithm is an intuitive heuristic approach to the scheduling problem and has not been previously explored in the reviewed literature. It is a greedy based algorithm that attempts to minimize the total system reconfiguration based on the state of the system and the position of targets in the scene at the time of scheduling.

In previous algorithms, the targets to be identified are taken from a sorted queue in the order that they enter or that they are predicted to exit the scene, and then assigned to the most appropriate camera. Alternatively, using the LSR approach, every camera-to-target pair is considered individually and the pair resulting in the least system reconfiguration is chosen. The proposed algorithm is shown in Algorithm 3.4

3.3.5 Least Future System Reconfiguration (LFSR)

The Least Future System Reconfiguration (LFSR) algorithm is a variant of the LSR algorithm that is described above in Section 3.3.4. The LFSR differs from the LSR approach in that it attempts to minimize the total system reconfiguration based on the *future* state of the system and predicted

Algorithm 3.5 Least Future System Reconfiguration algorithm.

Start
for $i = 1$: size of F **do**
  if $U \neq \emptyset$ **then**
    minReconfiguration = 181°
    for $j = 1$: size of U **do**
      time t_{ref} =time t_{current}
      time t_{previous} =time t_{future} = predicted exit time of target u_j
      $\Delta\omega_{f_i} = |\omega_{f_i,t_{\text{future}}} - \omega_{f_i,t_{\text{ref}}}|$
      while $\Delta\omega_{f_i} \neq |\omega_{u_j,t_{\text{future}}} - \omega_{f_i,t_{\text{ref}}}|$ **do**
        if $\Delta\omega_{f_i} > |\omega_{u_j,t_{\text{future}}} - \omega_{f_i,t_{\text{ref}}}|$ **then**
          $t_{\text{previous}} = t_{\text{future}}$
        else
          if $\Delta\omega f_i >$ minReconfiguration **then**
            break while
          $t_{\text{ref}} = t_{\text{future}}$
        $t_{\text{future}} = \frac{t_{\text{previous}}+t_{\text{ref}}}{2}$
      if $\Delta\omega f_i <$ minReconfiguration **then**
        minReconfiguration = $\Delta\omega f_i$
        $u_{\text{closest}} = u_j$
for $i = 1$: size of F **do**
  for $j = i + 1$: size of F **do**
    if u_{closest} for $f_i = u_{\text{closest}}$ for f_j **then**
      if $|\omega_{f_i} - \omega_{u_{\text{closest}}}| < |\omega_{f_j} - \omega_{u_{\text{closest}}}|$ **then**
        assign target u_{closest} to free camera f_i
      else
        assign target u_{closest} to free camera f_j
      goto *Start*
for $i = 1$: size of F **do**
  assign target u_{closest} for f_i to camera f_i

target positions. Pseudo code for the LFSR scheduling policy is given in Algorithm 3.5

3.4 Summary

Five greedy-based scheduling algorithms for target identification have been implemented: two previously developed and three new to this work. The greedy-based scheduling policies described in this chapter provide computationally efficient solutions to the real-time, *a priori* unknown, scheduling problem considered herein. A comparative evaluation of these five policies is outlined in Chapter 4.

Chapter 4

System Evaluation

Due to the complexity of developing an optimal offline algorithm for competitive analysis of the five scheduling policies described in Chapter 3, an empirical evaluation has been performed. A simulator has been developed to provide a repeatable experimental environment that addresses many of the inherent time and experimental setup limitations of using the real-world system. A description of the simulator and simulated experiments is given in Sections 4.1 and 4.2 respectively, while details of the real-world experiments are given in Section 4.3.

4.1 Simulator

A simulator has been developed to provide a controllable and repeatable environment in which to evaluate the scheduling algorithms. At a high level, the simulated system employs a similar modular layout as described in Chapter 2. There are, however, some implementation differences worthy of note. A global Target Repository has been added to the simulated system. The repository is responsible for creating and updating a set of simulated targets, each with a unique identification tag. The omnidirectional cameras scan this Target Repository at each 'capture' for targets that lie within their respective FOVs. The Tracking Module used in the simulated system also differs from that used in the physical system. In the real-world system, the Tracking Module uses the predicted positions of a target to track it from one image to the next; however, in the simulated system, the Tracking Module uses a target's identification tag to track its progress through the

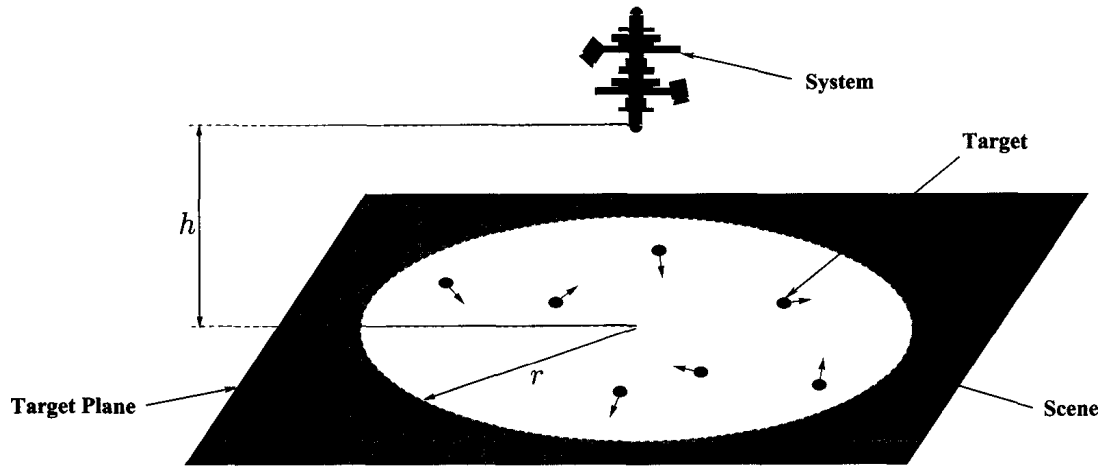


Figure 4.1: Simulated environment. The system centre is located $h = 5$ m above the target plane. The scene of radius $r \approx 14$ m is formed where the omnidirectional camera FOVs intersect the target plane.

scene. Using the identification tag has the advantage of allowing the scheduling algorithms to be evaluated directly without the influence of any tracking errors. The physical pan/tilt stages are also replaced using a virtual representation that simulates their motion.

4.2 Simulated Experiments

To gain some insight into the relative performance of the scheduling policies under different conditions, a total of 180 simulated experiments have been carried out — 36 scenarios were investigated for each of the five scheduling policies. For each experiment, targets were restricted to moving within a single plane situated $h = 5$ m below the system center. Additionally, without any loss of generality, the FOV of the omnidirectional cameras was set to 15 m. The intersection of the omnidirectional FOVs and the target plane results in a circular scene with a radius, r, of approximately 14 m, as depicted in Figure 4.1. For statistical purposes, each of the 180 experiments was run 100 times with independent target sets.

The performance of the scheduling schemes were investigated for different: target entry conditions, congestion levels, target to camera speeds, target trajectories, and number of active cameras. The details of each experiment are shown tabulated in Tables A.1, A.2 and A.3 in Appendix A. A description of these parameters is given in the following subsections.

4.2.1 Target Entry Condition

Two entry conditions are investigated; simultaneous entry, and continuous entry. The simultaneous entry condition is used to evaluate the performance of the scheduling schemes in applications where multiple targets enter the scene simultaneously; for example, monitoring an area outside of an arena after a large sporting event — a large number of people will exit the arena at roughly the same time. The performance of the scheduling policies is also investigated for applications where targets enter the scene continuously at random time intervals, as would be true of a monitored area in a shopping centre or airport.

Under the simultaneous entry condition, the experiments are run such that every target enters the scene at roughly the same time; however, for the continuous entry condition, targets are set to enter the scene according to a Poisson's process with entry rate, λ. An implementation of Knuth's Poisson random number generator [62] was used to obtain the target entry times as shown in Algorithm 4.1.

Algorithm 4.1 Knuth's Poisson random number generator.

let $L \leftarrow e^{-\lambda}$, $k \leftarrow 0$ and $p \leftarrow 1$
repeat
  $k \leftarrow k + 1$
  generate uniform random number U in $[0, 1]$ and let $p \leftarrow p \times u$
until $p \geq L$
return $k - 1$

4.2.2 Target Congestion

The influence of target congestion on the relative performance of the scheduling policies was also looked at. For the majority of experiments, the level of congestion was set such that the system is saturated; i.e., some of the targets passed through the scene without being viewed by an active camera. This was done to provide worst case comparisons between policies.

For the simultaneous entry condition experiments, the number of targets entering the scene was used to represent congestion. The greater the target number, the higher the congestion level. The number of targets employed ranges from 50 to 400 targets. For the continuous entry experiments, the entry rate, λ was used as a measure of congestion. A higher λ results in a faster rate of entry,

and thus a higher level of congestion. Two entry rates were investigated; $\lambda = 100$ targets/min and $\lambda = 200$ targets/min.

4.2.3 Target Trajectory

Experiments were performed for targets with linear and parabolic trajectories. Target trajectories were defined prior to an experiment by randomly assigning a set of starting conditions to each target (i.e. starting position, velocity, etc.), and then writing them to a trajectory file. Targets with linear trajectories were assigned a starting position (x_0, y_0, z_0), and velocity $(V_{x_0}, V_{y_0}, V_{z_0})$. As with the linear targets, the parabolic targets were created with an initial position (x_0, y_0, z_0) and velocity $(V_{x_0}, V_{y_0}, V_{z_0})$. In addition, the parabolic targets were also created with randomly chosen constants A, B and C which define the trajectory of the target according to the equation: $y = Ax^2 + Bx + C$. Targets were created such that they initially lay along the scene perimeter and moved in towards the scene. Fifteen linear and fifteen parabolic targets created in this way are shown in Figures 4.2(a) and 4.2(b) respectively.

At the start of each simulation, a predetermined trajectory file was loaded and a target was

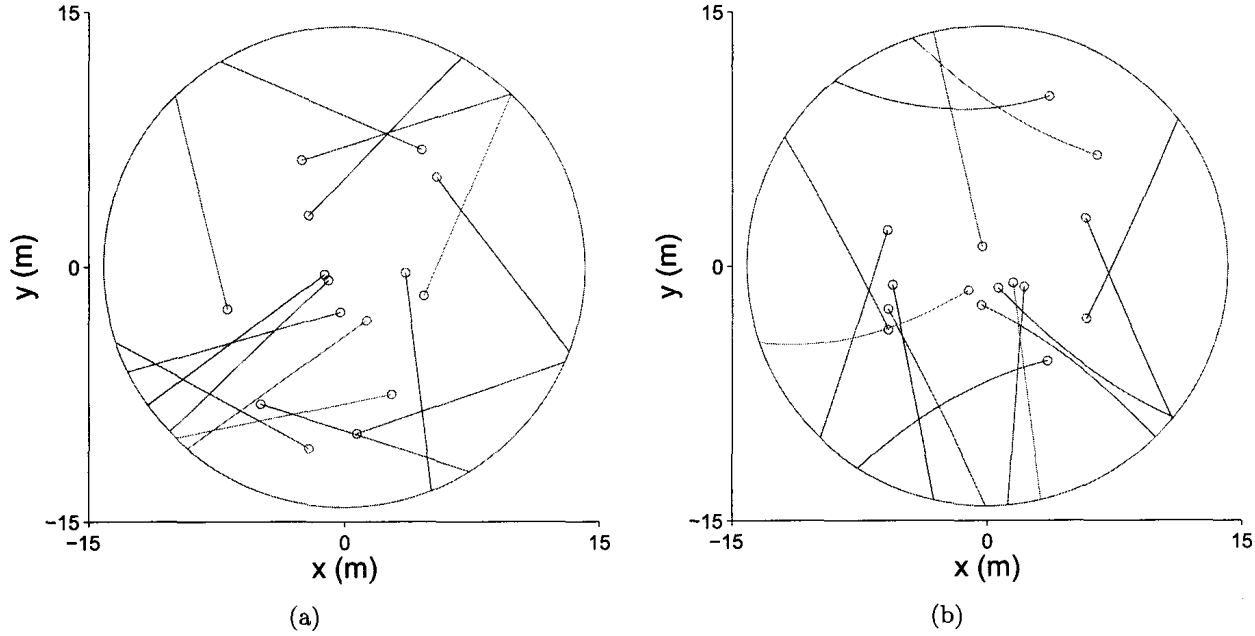


Figure 4.2: Example trajectories for (a) Simulated linear trajectory targets, and (b) Simulated parabolic trajectory targets.

created for each of its entries. The targets were stored in the global Target Repository (Section 4.1) where their positions were updated at each simulated time step. Having the ability to use the same trajectory file in multiple simulations provided a means of evaluating each of the five scheduling algorithms with the same target sets, thus allowing a direct comparison to be performed.

Linear and parabolic trajectories were chosen to investigate the effect of prediction accuracy on the performance of the scheduling schemes. In simulation, a target's future position was extrapolated using its current heading and speed, with the assumption that the target trajectory was linear. A target moving with a linear trajectory could therefore be predicted with 100 percent accuracy; however, a prediction error was introduced when the target trajectory was parabolic.

4.2.4 Target to Camera Speed

The effect of the ratio of target to camera speed on the relative performance of the algorithms was investigated. Different target to camera speeds were obtained by varying the speeds at which the targets moved through the scene, while keeping the camera speeds constant. At the slowest extreme, targets were set to move at an average pedestrian walking speed of 1.5 m/s. At the other extreme, the greatest target speed was set to 8 m/s — a quarter of the speed of the cameras at the midpoint between the system and its DOF. The third speed used is an average of the first two speeds, at 4.75 m/s.

4.2.5 Active Cameras

Simulations were run using two system configurations. The first configuration consisted of 2 active cameras while the second was made up of 4 active cameras. Each configuration used 2 omnidirectional cameras for detection and tracking. These configurations were chosen to investigate the scalability of both the scheduling policies as well as the system itself, while considering a practical limit, given the size of the physical modules.

4.2.6 Multi-Class Assignments

To demonstrate the advantages and flexibility of the multi-class Scheduling-Module structure, two different methods for assigning targets to classes have been investigated.

As is previously mentioned in Section 1.2, the objective of the Scheduling Module can be broken into two competing goals. One goal is to view as many targets as possible. The second goal is to view each target for as long or as many times as possible. To this end, the first multi-class implementation used, sets a target's priority level according to the number of times that the target has been viewed. Upon entering the scene, targets were assigned to the highest priority class; however, each time that they were viewed, the targets were demoted to the next class. In this way, targets continue to be serviced by the active cameras until either: the targets leave the scene, or they have been viewed some predefined maximum number of times. This maximum was set to 4 for these experiments.

In many applications, the ability of an automated surveillance system to prioritize targets based on their position in the scene, speed, etc., can greatly enhance its overall robustness. The advantages of using the proposed multi-class scheduling architecture in such applications was also investigated for each the five scheduling policies. Simulations were run using both a single-class and multi-class scheduling architecture. The same target set was used in both cases to provide a means of direct comparison. Each target was assigned one of two priority levels upon entering the scene — high or low, and unlike the previous multi-class implementation, targets were scheduled for viewing only once. For both the multi-class and single-class implementations, the simulations were run using a system comprising of two active cameras and two omnidirectional cameras. The target set used consisted of two hundred linear targets. A simultaneous entry condition was used, and every fourth target was assigned a high priority.

4.3 Real-World Experiments

The complexities associated with providing a controllable experimental environment for deploying the real-world system has restricted the extent to which the five algorithms could be evaluated through physical testing; instead, the focus of the real-world experiments was to validate the simulated system results while also serving as a proof of concept of the physical system itself.

4.3.1 Experimental Setup

The prototyped vision system comprising two active modules and a single omnidirectional fisheye module has been used, controlled by two networked system computers. The Omni-Camera Manager, Tracking and Synchronization modules were implemented on the first system computer, while the Scheduling, Platform Manager and System Control modules were run on the second system computer.

Fourteen static scene targets were implemented using fourteen 17" LCD computer monitors. Three different target types were employed, defined by the colour displayed on the corresponding LCD monitor — Type A (orange), Type B (green) and Type C (blue). Control of the fourteen targets was achieved using a system of three networked target computers. Each computer was responsible for controlling a single target type and was connected to its corresponding LCD monitors using a VGA video multiplexer. In this manner, each of the three target types could be displayed independent of the state of the other target types; however, all targets of the same type had to be displayed simultaneously.

A photograph of the experimental setup is shown in Figure 4.3, and the implemented layout showing the target type and coordinates of each of the fourteen scene targets is given in Figure 4.4.

4.3.2 Methodology

Experiments have been performed using the experimental setup described in Section 4.3.1 for each of the five scheduling algorithms. Precise and repeatable control of the entry and exit times of all fourteen scene targets was achieved using the aforementioned target implementation technique. To validate the results obtained in the simulated system evaluation (Section 4.2), simulations were performed for each of the real-world experiments using the same target positions, as well as entry and exit times. Validation was performed by comparing the results obtained using the simulated and real-world systems. All experiments were repeated five times using the same system and scene parameters.

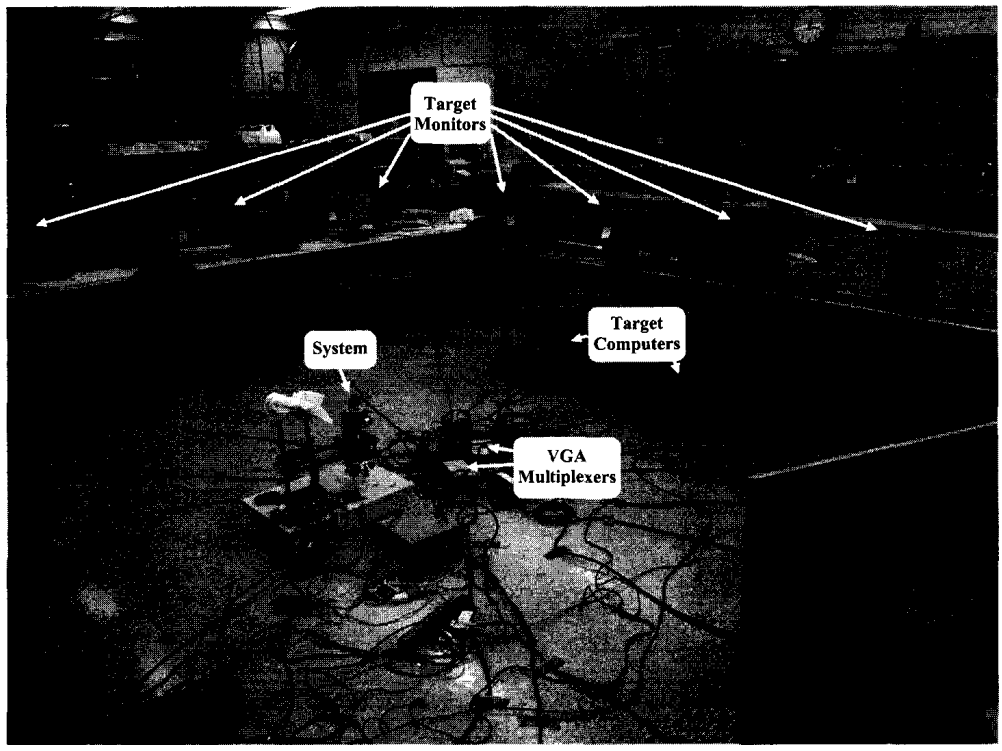
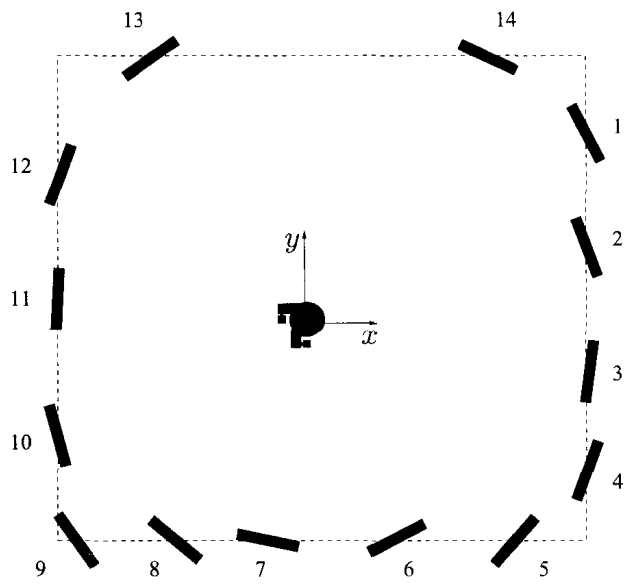


Figure 4.3: Experimental setup.

Two sets of experiments have been performed in this evaluation. First, the performance of each scheduling algorithm was investigated for a single scenario; the parameters for which were as follows:

- Simultaneous entry — targets were set to enter the scene simultaneously, ordered according to their type (colour) and position in the scene.
- Scene time — targets were set to stay in the scene for a predetermined length of time before exiting. The scene times were determined by target type and were chosen to ensure saturation of the system. Targets of Type A stayed in the scene for 8.0 seconds, targets of Type B for 7.0 seconds and targets of Type C for 6.0 seconds. These scene times were known to the Scheduling Module and were used by the EDF algorithm to determine the order in which targets were scheduled.
- Scheduling architecture — a single class architecture was used; i.e., all targets were of equal priority and were only scheduled for viewing once.



target	type	x	y	z
1	A	2.68 m	1.9 m	0.837 m
2	C	2.68 m	0.83 m	0.837 m
3	B	2.68 m	-0.33 m	0.837 m
4	A	2.68 m	-1.46 m	0.837 m
5	C	1.7 m	-1.96 m	0.857 m
6	A	0.8 m	-1.96 m	0.857 m
7	B	-0.25 m	-1.96 m	0.857 m
8	C	-1.13 m	-1.96 m	0.857 m
9	A	-2.08 m	-1.96 m	0.857 m
10	C	-2.22 m	-0.94 m	0.857 m
11	B	-2.22 m	0.14 m	1.107 m
12	C	-2.22 m	1.09 m	1.107 m
13	A	-1.45 m	2.44 m	0.827 m
14	B	1.57 m	2.6 m	0.977 m

Figure 4.4: Target layout showing target types and coordinates.

The performance of two scheduling policies (FCFS and LSR) was also investigated using both a multi-class and single-class scheduling architecture. These experiments were performed to demonstrate the advantages of the proposed multi-class scheduling architecture. The target and system parameters used in these experiments are as follows:

- Simultaneous entry — targets were set to enter the scene simultaneously, ordered according to their type (colour) and position in the scene.
- Scene time — all targets were set to stay in the scene for 8.0 seconds before exiting, therefore ensuring that the system was saturated.
- Scheduling architecture — experiments were performed for both a single class architecture as well as a multi-class architecture. For the single-class implementation, all targets were given an equal priority and were only scheduled once for viewing. Alternatively, for the multi-class implementation, targets of Type C were given a high priority while targets of Types A and B were both given a low priority; again, targets were only scheduled once for viewing.

4.4 Summary

Experiments have been performed in simulation to evaluate the relative performance of five scheduling policies under different target and system conditions. Parameters that have been investigated include: target entry conditions, congestion levels, target to camera speeds, target trajectories, and the number of active cameras. Experiments using the system prototype have also been performed to validate the results obtained through simulation, to demonstrate the advantages of using a multi-class scheduling architecture and to serve as a proof of concept of the physical system. The key findings from these experiments are presented in Chapter 5.

Chapter 5

Data Analysis and Discussion

This chapter presents the results obtained from the experimental evaluation (described in Chapter 4) of the five heuristic scheduling policies. The simulated performance of the system is shown in Section 5.1 for each of the five policies under varying target and system conditions. Section 5.2 presents results obtained using the system prototype.

The main performance measure used in the following discussion is the *identified target count*; defined as the number of targets that have been viewed at least once by an active camera. As described in Section 4.2, each experiment was run 100 times with independent target sets. Box plots are used in the following sections to illustrate the performance dispersion within a single experiment.

5.1 Simulated Experiments

The performances of the five scheduling policies under different simulated target and system conditions are presented in this section. The effects of target congestion, target to camera speed, target trajectory, and the number of active cameras (as described in Chapter 4) are investigated. Results showing the advantages of using the proposed multi-class scheduling architecture are also presented.

5.1.1 Target Congestion

The influence of target congestion on system performance for each of the five scheduling policies has been investigated. Simulations were run for congestion levels of 50, 100, 200 and 400 targets with simultaneous entry conditions. Simulations were also performed for 200 targets, entering the scene at a rate of $\lambda = 100$ targets/min, and $\lambda = 200$ targets/min. Interestingly, the changes in congestion levels had different effects on system performance for the two entry conditions.

Identified target counts for targets entering the scene simultaneously are illustrated in the box plots of Figures 5.1(a) and 5.1(b), for congestion levels of 400 and 100 targets respectively. Box plots provide a convenient method of summarizing the performance data for the 100 independent target sets used in each experiment. The plots were constructed as described in [63], such that the outliers (shown as crosses) lie at least 1.5 times the interquartile range above or below the upper quartile or lower quartile respectively.

The system performance does not appear to be affected by the level of congestion for any of the scheduling policies. This trend was observed for saturated congestion levels under different target speeds, target trajectories and active camera counts; i.e., once saturated, system performance appears to be unaffected by an increase in the congestion level.

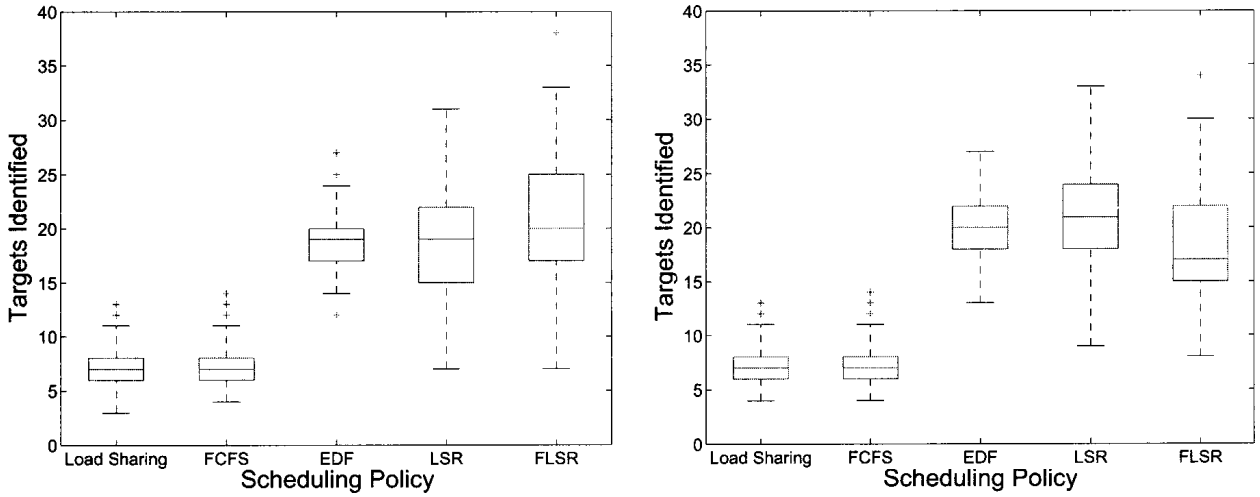


Figure 5.1: Identified target counts for a system comprising 2 active cameras for two congestion levels: (a) 400 targets, and (b) 100 targets. Here, each target follows a linear trajectory with a speed of 4.75 m/s and enters the scene simultaneously.

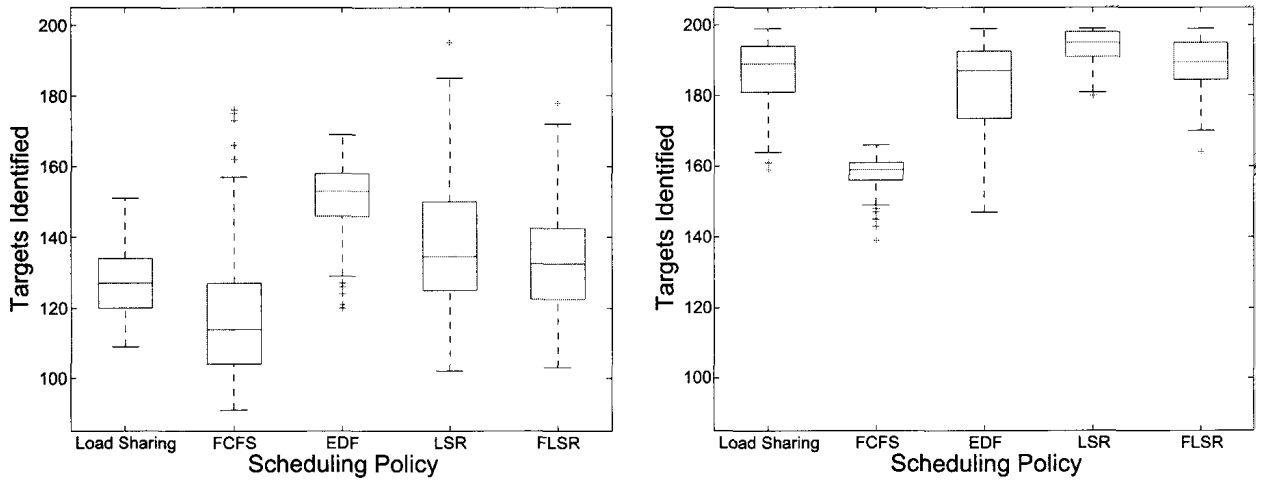


Figure 5.2: Identified target counts for a system comprising 2 active cameras for two congestion levels: (a) $\lambda = 200$ targets/min, and (b) $\lambda = 100$ targets/min. 200 targets were used, each following a linear trajectory with a speed of 4.75 m/s and entering the scene continuously.

Alternatively, under the continuous-entry target conditions, the system performance does appear to be affected by the level of congestion. Identified target counts for targets entering at a rate of $\lambda = 100$ targets/min, and $\lambda = 200$ targets/min are shown in Figures 5.2(b) and 5.2(a), respectively. This degradation in performance was seen to continue until some upper congestion level is reached, at which point a further increase in the entry rate will have no significant effect.

5.1.2 Target to Camera Speed

The effect of different target to camera speeds on system performance was also investigated for each of the five scheduling policies under different target trajectory types, active camera counts, entry conditions and congestion levels. A direct comparison between policies may be made as the targets used from one experiment to another differ only in their magnitude of velocity; i.e., corresponding targets move through the scene along the same path, differing only in how fast they move. Mean identified target counts for three different target speeds are presented in Figure 5.3. System performance was observed to decrease non-linearly with an increase in target speed for each policy.

The relative performance of the scheduling policies themselves also appear to be affected by target-to-camera speeds. The disparity in the level of performance between the scheduling policies

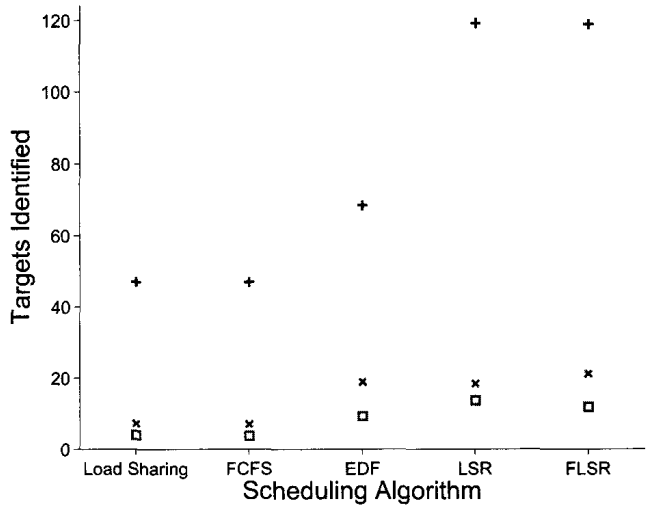


Figure 5.3: Mean identified target counts for a system comprising 2 active cameras for three target speeds, where boxes represent a target speed of 8.0 m/s, crosses a speed of 4.75 m/s and pluses a speed of 1.5 m/s. 400 targets were used, each following a linear trajectory and entering the scene simultaneously.

was seen to decrease as the target speeds increased. At the higher target speeds, the active cameras often can not 'chase-down' a target, thereby reducing the influence of the scheduling policy on the overall system performance.

5.1.3 Target Trajectory

The influence of target predictability on the performance of the scheduling policies was evaluated (as described in Section 4.2.3) by comparing the performance of the different policies for two trajectory types — linear and parabolic. Figure 5.4 shows the performance of each of the five scheduling policies for a single set of experiments using (a) linear, and (b) parabolic trajectories.

Interestingly, the performance of four out of the five scheduling policies were, to an extent, unaffected, if not enhanced, when using the more unpredictable parabolic targets. There was, however, a significant decrease in the performance of the EDF policy. This can be attributed to the method of queuing targets for scheduling used by the EDF policy. As discussed in Section 3.3.3, the EDF policy schedules targets in the order in which they are predicted to exit the scene. Inaccurate predictions are therefore detrimental to its performance.

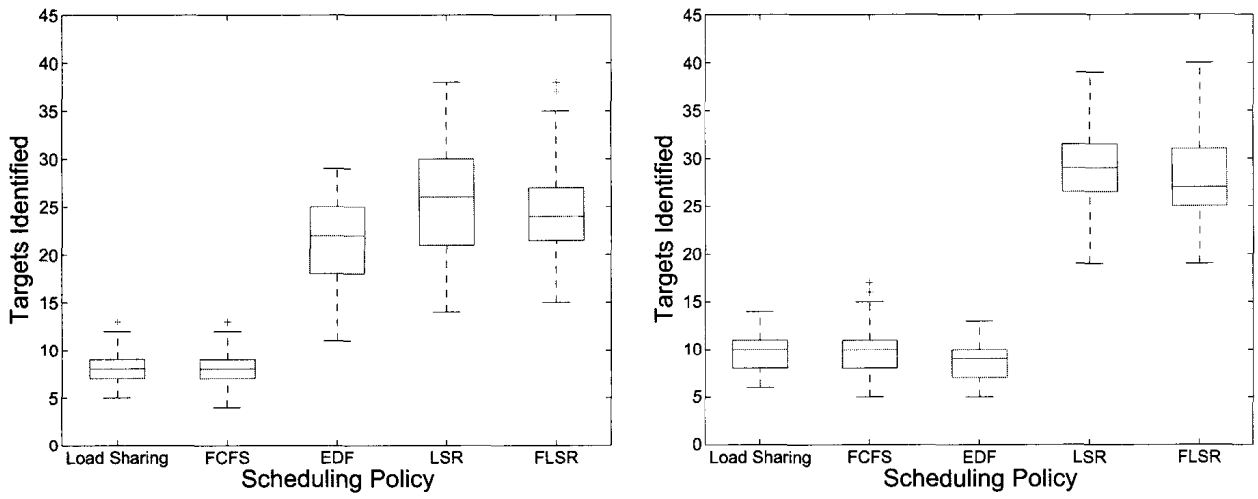


Figure 5.4: Identified target counts for a system comprising 4 active cameras for targets moving with (a) linear, and (b) parabolic trajectories. 400 targets were used, each moving at a speed of 8 m/s and entering the scene simultaneously.

5.1.4 Active Cameras

The scalability of the system is investigated for each scheduling policy. Figure 5.5 shows the mean identified target counts for experiments using systems comprising two and four active cameras. For each of the scheduling policies, the number of targets identified with the four camera system was approximately double that of the two camera system. This observation is also true for all other combinations of target trajectories, congestion levels, speeds and entry conditions.

5.1.5 Multi-Class Assignments

As detailed in Section 4.2.6, two sets of experiments were carried out to test the advantages of using the proposed multi-class scheduling architecture. In the first set of experiments, targets entering the scene were initially assigned to the highest priority class, and then demoted to a lower class each time that they were viewed. The identified target counts for one of these experiments are shown in Figure 5.6(a), while the average number of times that the viewed targets were seen by the active cameras is shown in Figure 5.6(b). The results demonstrate the ability of the multi-classed architecture to schedule targets for the competing identification goals outlined in Chapter 1.2.

Worthy of note is that no correlation was observed between the performance of the scheduling

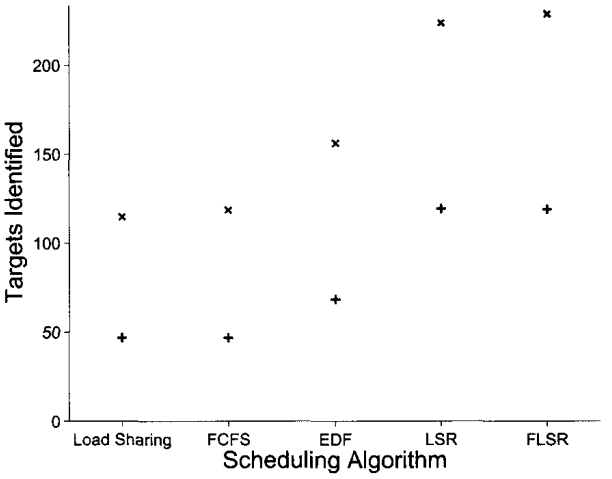


Figure 5.5: Mean identified target count for two system configurations, where crosses indicate the performance of a four camera system, and pluses that of a two camera system. 400 targets were used, each following a linear trajectory with a speed of 1.5 m/s and entering the scene simultaneously.

policies when using the two performance criteria. The LSR policy was seen to perform well in both cases, while the Load Sharing and EDF policies perform well under the first criteria but show the poorest performance under the second criteria. This observation is consistent with the findings of Costello *et al.* [13].

The second set of experiments was designed to demonstrate the ability of the multi-class architecture to prioritize targets according to external factors, such as their position in the scene,

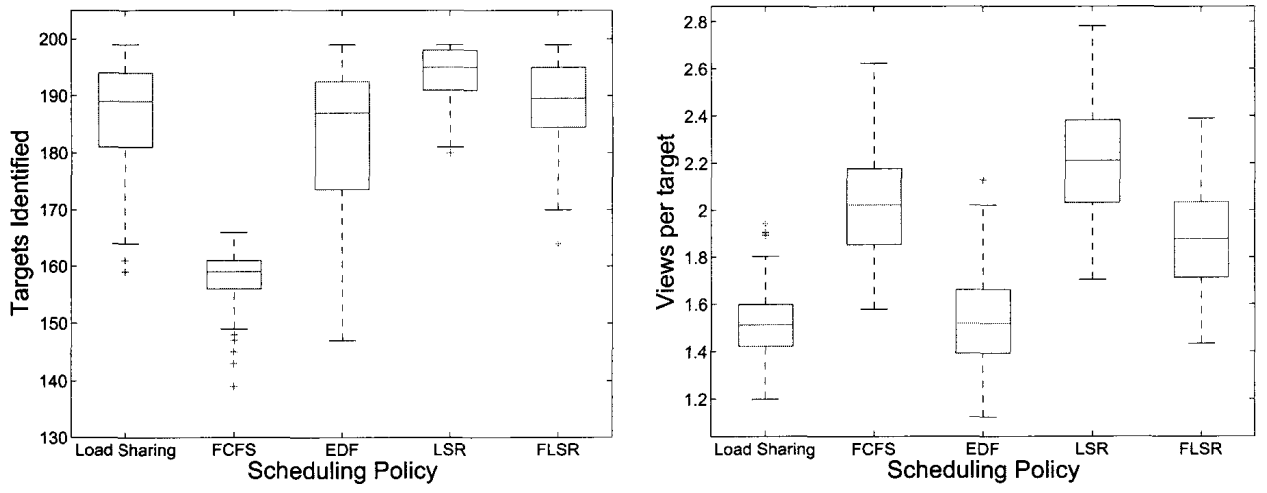


Figure 5.6: (a) Identified target counts, and (b) average number of views per target for a system comprising 2 active cameras using 200 targets, each following a linear trajectory with a speed of 4.75 m/s, and entering the scene continuously at a rate of 100 targets/min.

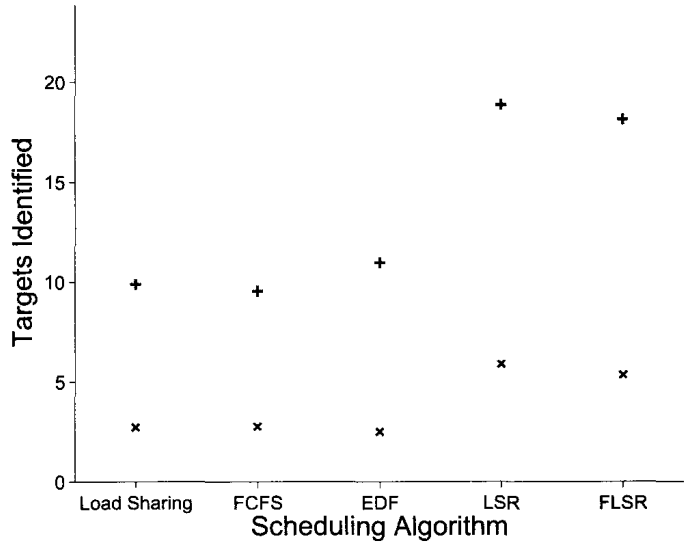


Figure 5.7: Mean identified target counts for a system comprising of 2 active cameras using 200 targets, each following a linear trajectories at a speed of 4.75 m/s and entering the scene simultaneously. Crosses show the performance using a single-class architecture, while pluses show that of a multi-class structure.

speed, etc.. To accomplish this, simulations were run using a multi-class architecture, and then again using a single-class architecture. In both cases, every fourth target entering the scene was given a high priority; the number of identified high priority targets was then used to compare the two architectures. The results from these experiments show that the multi-class architecture greatly outperforms its single-class counterpart in such applications. This is illustrated in Figure 5.7.

5.1.6 Relative Performance

An overall trend in the relative performance of the five scheduling algorithms could be seen in the data collected from the 36 simulated scenarios. For the majority of these scenarios, the Load Sharing and FCFS policies appeared to perform the poorest, while the LSR and FLSR scheduling policies consistently performed the best, by a considerable margin; however, the disparity in performance between policies was reduced at higher target-to-camera speed ratios (as shown in Section 5.1.2). When scheduling linear targets, the performance of the EDF policy typically fell in between that of the LSR and FLSR policies, and the FCFS and Load Sharing policies; however, its performance appeared compromised when scheduling the more unpredictable parabolic targets (shown in Section 5.1.3).

5.2 Real-World Experiments

A comparison between the performances of the five scheduling algorithms using the real-world and simulated systems has been performed for a single scenario as described in Chapter 4. The system prototype has also been used in experiments to showcase the advantages of using the proposed multi-class scheduling architecture for the prioritization of targets based on external factors. Fourteen static scene targets were implemented using LCD computer monitors, arranged in a rectangular pattern around the system as laid out in Figure 4.4. A simultaneous target entry condition was used in every experiment; i.e., all fourteen of the targets appeared at the same time. Targets were detected and tracked at a low resolution in the omnidirectional view (Figure 5.8), while high resolution target images, required for identification, were taken using the two active perspective cameras.

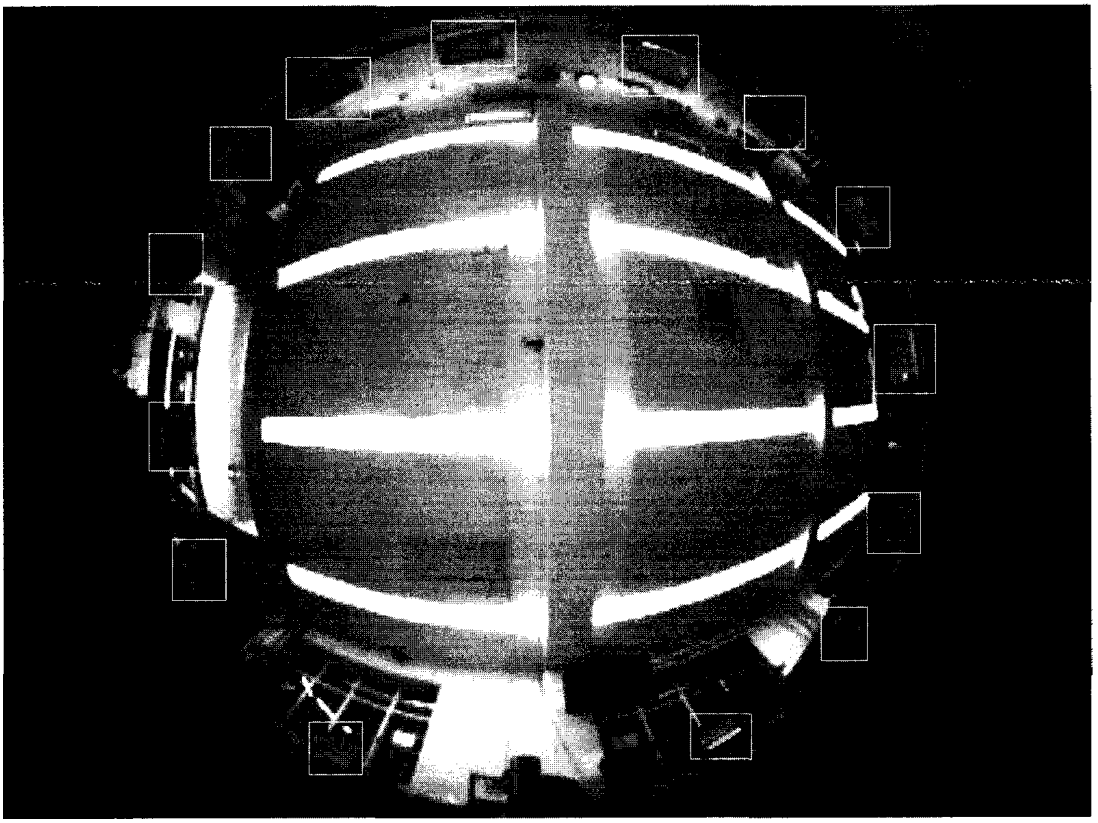


Figure 5.8: Omnidirection view from the top fisheye camera showing the fourteen scene targets.

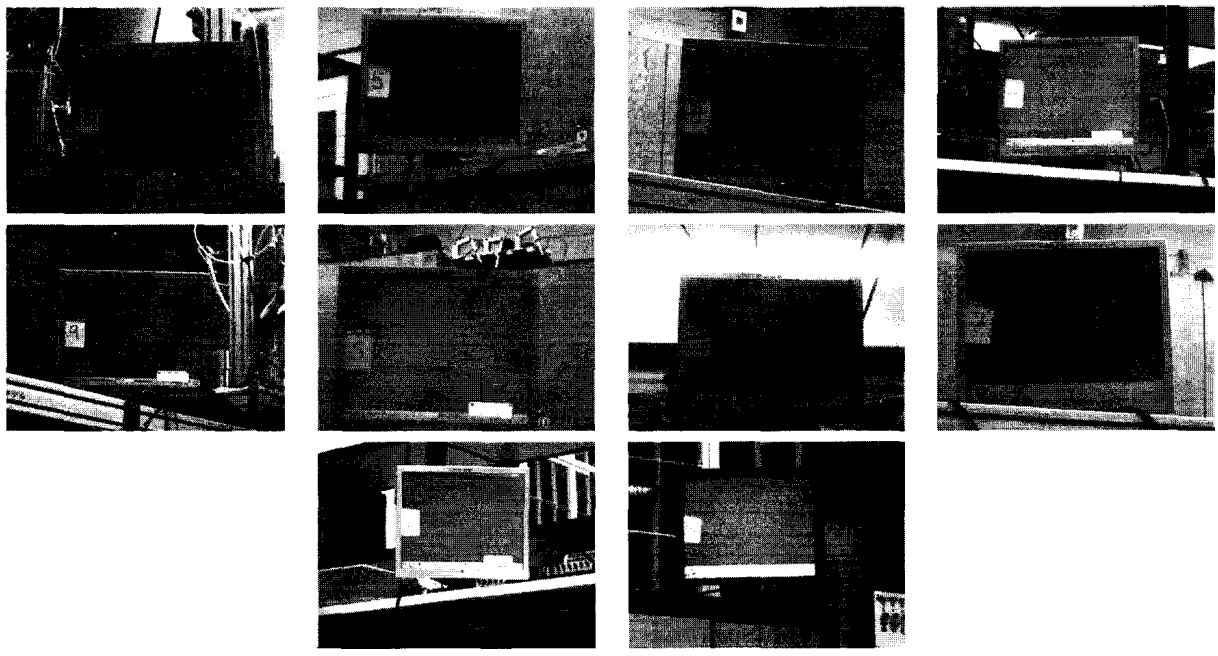


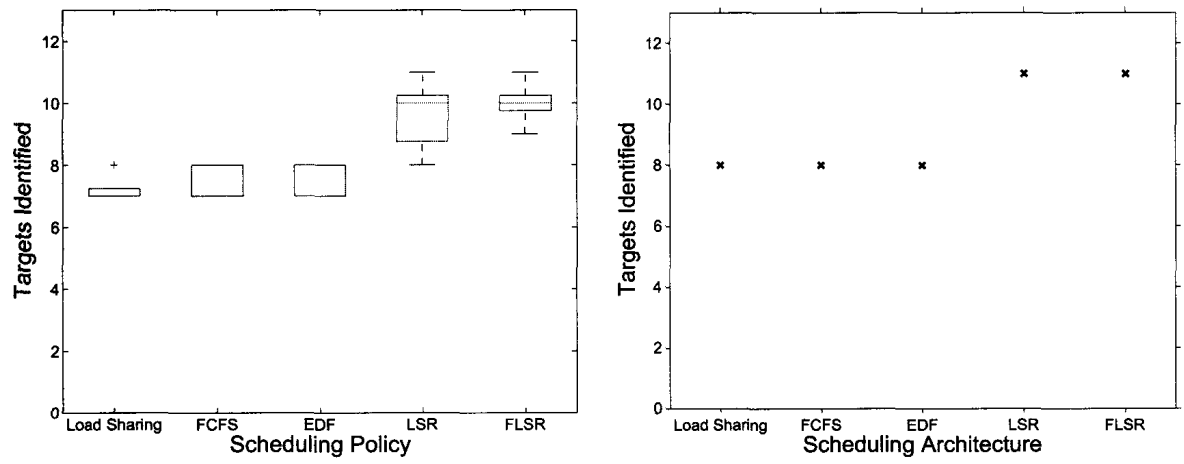
Figure 5.9: Target images taken by the active perspective cameras during one run of the real-world experiments.

A label was placed on each target monitor displaying its number in the scene. Once an active camera was centred on a target, six target images were taken and saved to memory; the target was considered to be identified if its target label was easily recognizable in one or more of these images. It was found that approximately 98 percent of the targets viewed in these experiments could be identified in at least one of the images taken. A sample of high resolution target images taken in one of the experiments is shown in Figure 5.9.

5.2.1 Real-World versus Simulated System Performance

To validate the simulated system results outlined in Section 5.1, experiments were performed for a single scenario using both the real-world and simulated systems. To account for real-world variability, every experiment was repeated five times. Identified target counts obtained using the real-world system are shown in Figure 5.10(a) for each of the five scheduling policies, while the corresponding identified target counts obtained through simulation are shown in Figure 5.10(b).

Some disparity was seen in the overall simulated and real-world system performances. For each of the scheduling policies, the simulated system performance was, on average, better than



(a) (b)

Figure 5.10: Identified target counts for (a) the real-world system, and (b) the simulated system.

that obtained using the physical system. The better simulated performances can be attributed to the idealized approach used in the simulated system implementation. The main goal of the simulated evaluation (Section 5.1) was to gain insight into the relative performance of the five scheduling policies; as such, many of the system parameters were modeled using an ideal case. The primary factor contributing to the disparity in performance between the real and simulated systems appears to be the target identification period; i.e., the time elapsed between the scheduling and identification of a target. In simulation, the cameras were accurately guided to the target locations and, once there, the cameras remained fixated on the target for precisely six 'image captures.' Imperfections in the real system's peripheral guidance calibration meant that visual servoing of the active stages was required to centre the targets within the FOV of the perspective cameras. This resulted in the identification period of the physical system cameras being greater than that of the ideal simulated case.

The variation in performance observed between repeated experiments for the Load Sharing, FCFS and EDF policies was the result of varying identification periods (targets were scheduled in the same order from run to run and in the same order as observed in simulation); however, some differences in the scheduling order were observed with the LSR and FLSR policies. As described in Chapter 3, scheduling is performed for the LSR and FLSR policies based on the state of both the system and the targets at the time of scheduling. Although targets were initially scheduled

in the same order for each run (and as in simulation), variations in the system state at times of scheduling, caused the target order to differ in a number of the experiments.

Although some differences were seen in the absolute performance between the simulated and real-world systems, strong similarities were observed in the relative performance of the five scheduling policies. This would suggest that, although not suitable for a direct characterization of the system, the simulated results presented in Section 5.1 provide a good means of evaluating the relative performance of the five scheduling policies.

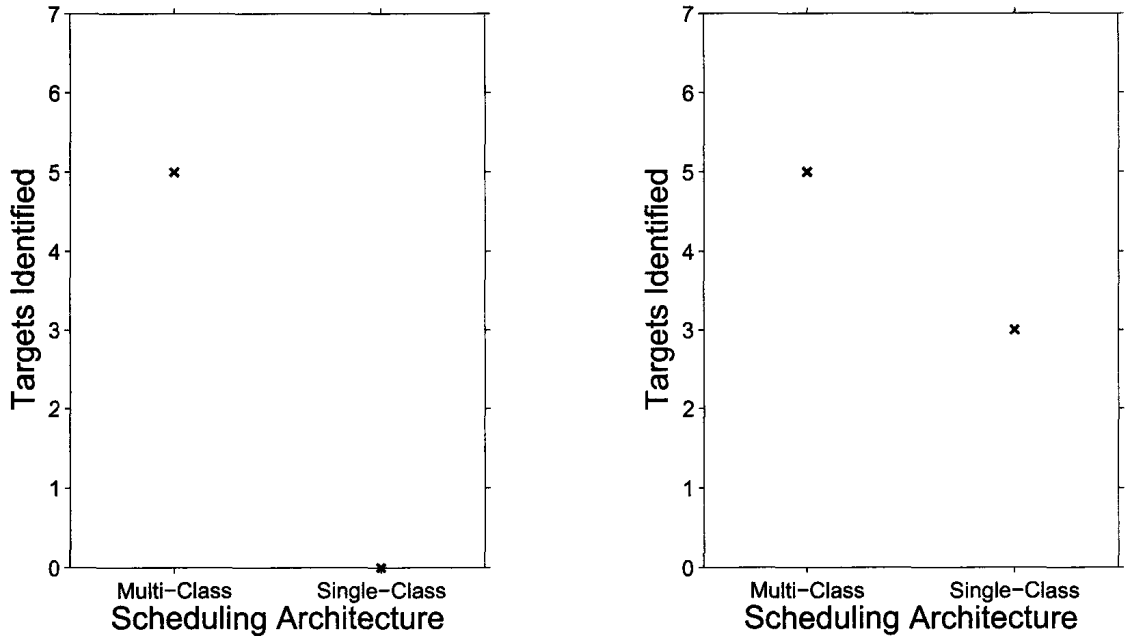


Figure 5.11: Identified target counts for multi-class and single-class scheduling architectures using (a) a FCFS scheduling policy, and (b) a LSR scheduling policy.

5.2.2 Multi-class Prioritization

To further characterize the operation of the system, the performance of two scheduling policies (FCFS and LSR) was investigated using both a multi-class and single-class scheduling architecture. For the single-class implementation, all targets were given an equal priority. Alternatively, for the multi-class implementation, targets of Type C were given a high priority while targets of Types A and B were both given a low priority. The number of targets of Type C identified using the

FCFS policy is shown for both class implementations in Figure 5.11(a) while those obtained using the LSR policy is shown in Figure 5.11(b). As in the simulated results described in Section 5.1.5, the results from these experiments show that the multi-class architecture greatly outperforms its single-class counterpart in its ability to prioritize targets.

5.3 Summary

A comparative evaluation of the five scheduling algorithms has been performed for thirty six different simulated system and target scenarios. In the majority of cases, the LSR and FLSR scheduling policies consistently performed the best, while the Load Sharing and FCFS policies appeared to perform the poorest. The EDF policy showed good overall performance under predictable target conditions, however, under unpredictable conditions, a drastic decrease in performance was observed. Experiments performed using the prototyped system serve to validate these findings and demonstrate the applicability of the multi-classed architecture.

Chapter 6

Concluding Remarks

This thesis presents a centralized active vision based solution to the problem of remote target identification in high throughput applications, where the targets to be identified outnumber the system resources. The following sections serve to highlight the contributions of this work and make recommendations for future work.

6.1 Summary and Conclusion

The contributions of this thesis may be summarized as:

1. Development of a centralized active vision system. Although based on a previously conceived concept, all aspects of the system hardware and software have been redesigned, resulting in a more cost effective, compact and modular system. Significant improvements over the previous design include:

    - Miniaturization — a significant reduction in the size of the system modules has been achieved. The active modules have been reduced in height from 0.26 m to 0.13 m, and the omnidirectional fisheye modules now measure 0.06 m, down from 0.16 m.
    - Self contained control hardware — by eliminating the need for expensive standalone control modules, the development of custom control hardware for the pan and tilt drive systems has enhanced the system's modularity, while significantly reducing its overall cost.

- Cable spooling — direct connection between the pan/tilt cameras and the system computers has been made possible through the design of a low-profile spooling mechanism. The direct connection allows the video feed to remain digital, thereby reducing the losses incurred by analog to digital conversion, as well as analog transmission. Unlike the original design, however, spooling has limited the active modules to ± 4 revolutions about the system axis.
- Software architecture — the software architecture developed in this work allows the system to detect, track and schedule multiple targets simultaneously.

2. Formulation of the resource scheduling problem for a centralized active vision system in an unconstrained environment with no *a priori* knowledge of target entry and exit points, trajectories, etc..

3. Development of three heuristic resource scheduling policies. Similar to the approach used in several of the reviewed scheduling policies, the policies developed in this thesis employ greedy-based algorithms to make locally optimal target to active-camera assignments. Two scheduling policies that appeared repeatedly in the reviewed literature were also implemented to serve as a benchmark for comparison.

4. Development of a multi-class scheduling architecture that extends the functionality of the system to allow targets to be prioritized based on factors such as location, speed and number of times viewed.

5. Development of a simulated system environment, whose behaviour closely resembles that of the physical system. The simulated system provides a means of quickly evaluating relative system performance under different system and target conditions, many of which would be prohibitively expensive and time consuming to investigate using the physical system.

Together, these contributions form the basis of an overall active vision solution to the remote identification of multiple targets. The scheduling policies that have been used in this thesis, along with findings from their simulated and real-world evaluations are elaborated upon below.

Five greedy-based scheduling policies have been implemented to coordinate the system's active vision resources for viewing multiple targets. EDF and FCFS policies were implemented based on work done in the reviewed literature. Load Sharing, LSR and FLSR scheduling policies, that are new to this work, have also been developed and implemented.

A comparative evaluation of the five scheduling policies has been performed for thirty six different simulated system and target scenarios. The parameters that were varied in this evaluation include: target entry conditions, target congestion levels, target to camera speeds, target trajectories, and the number of active cameras. An overall trend in the relative performance of the five scheduling algorithms was observed. In the majority of cases, the Load Sharing and FCFS policies appeared to perform the poorest, while the LSR and FLSR scheduling policies consistently performed the best, by a considerable margin. Interestingly, however, it was observed that the disparity in performance between policies was reduced at higher target-to-camera speed ratios. It was also noted that the performance of the EDF policy was highly dependent on target predictability.

The prototype system was used in a number of real-world experiments to validate the results obtained through simulation, while also serving as a proof of concept. Although strong similarities between the relative performances of five scheduling policies in the simulated and real-world experiments were observed, the simulated system appeared to, on average, outperform the real-world system. This suggests that, although not suitable for a direct characterization of the system, the simulated system provides a suitable means of evaluating the relative performance of the five scheduling policies.

As discussed in Chapter 1, the objective of the Scheduling Module can be broken into two competing goals. One goal is to view as many targets as possible. The second goal is to view each target for as long or as many times as possible. Simulated experiments have been performed that demonstrate the applicability of using a multi-class scheduling architecture under these objectives. Experiments using the prototype and simulated systems have also been performed in which the multi-classed scheduling approach was shown to greatly outperform its single-class counterpart in its ability to prioritize targets according to external factors, such as their position in the scene and speed.

6.2 Recommendations and Future Work

Vision System

Several recommendations can be drawn from the construction, calibration and evaluation of the prototype. Even though the prototyped system fulfills its purpose within the scope of this thesis, areas of improvement in aspects of both the hardware and software implementation have been identified.

Although continuous pan rotation is made possible through the use of a custom spooling mechanism, the system is still limited to ± 4 revolutions about its 'unwound' state. Measures have been taken in software to ensure that this limit is never reached in normal operation; however, situations may arise in which the software limiter may fail. To provide more robust protection of the spool and active module components at all times, the implementation of a dedicated hardware limiter is recommended.

The prototyped system employs target detection and tracking algorithms that have been greatly simplified. Target detection is currently performed using a simple colour thresholding technique. This approach proved useful during the scheduling policy development stage and for a proof of concept, allowing multiple target definitions (colours) to be implemented quickly and easily. Although acceptable within the scope of this thesis, a more robust target specific approach is needed if the system is to be used in real-world surviellance applications.

The target tracking technique can also be improved to increase the overall robustness of the system. The method used in this work uses the predicted positions of a target to track it from one image to the next. Again, this method proved to be acceptable given the scope and objectives of this thesis; however, implementing a more robust tracking method is recommended to overcome the limitations of the current method in situations when a target may become fully or partially occluded.

Without addressing task and algorithm specific requirements, it was simply assumed that high resolution target images were required for successful identification; the development and/or implementation of the identification algorithms themselves has been left for future work. It is noted that by adding this next step, other, task specific, factors may be introduced into the scheduling

problem; factors such as target heading and periods of predicted occlusion may play a role in determining an optimal scheduling policy.

The stacked multi-camera system concept that is used in this work has been previously proven to have good depth perception and target localization capabilities. This has been shown to be achievable through precise calibration of the active perspective cameras, the omnidirectional fisheye cameras, as well as relative camera positions. By improving the calibration technique used, the functionality of the system could be extended to allow for the extraction of 3D positional data from the environment.

Resource Scheduling

A heuristic greedy-based approach has been taken in this thesis to address the task of scheduling the system's active resources to view multiple targets. Although this approach provides a computationally efficient solution that is easily implemented, there is still much room for further investigation.

In each of the scheduling policies implemented in this work, targets are scheduled in a non-preemptive fashion (within a single priority class). Once scheduled, a camera will remain occupied until either, the assigned target is viewed for some predetermined length of time, the target leaves the scene, or another target is assigned a higher priority level. This non-preemptive approach can limit the ability of the system to react to changes within the environment; as such, an investigation into the affect of extending the policies to allow for preemptive scheduling is warranted.

When predicting future simulated target positions during scheduling, it was simply assumed that targets move with a linear trajectory; i.e., a target's future position was predicted using its heading and speed at the time of prediction. The simulated experiments that were performed using targets with parabolic trajectories served to illustrate the extent to which the performance of the EDF policy depends on the accuracy of predicted future target states. As was confirmed in the reviewed literature, this is an inherent problem of policies that rely on predicted target states. Intuitively, and as reported in previous work, prediction-based policies can often outperform those policies that only consider the current state of a target, in applications where target motion is highly predictable. This, however, is a problem in real-world applications where target motion is

often erratic and unpredictable. Although outside the scope of this thesis, developing a method of providing reasonable target predictions could allow prediction-based algorithms to perform in such real-world applications; for example, learning techniques may be used to build target models based on the trajectories of previously observed targets.

References

[1] N. Ukita and T. Matsuyama, "Real-time cooperative multi-target tracking by communicating active vision agents," *Computer Vision and Image Understanding*, vol. 97, pp. 137–179, 2005.

[2] S.-N. Lim, L. Davis, and A. Elgammal, "A scalable image-based multi-camera visual surveillance system," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*, (Washington, DC, USA), pp. 205–212, Jul. 2003.

[3] A. Bakhtari, M. Naish, M. Eskandari, E. Croft, and B. Benhabib, "Active-vision-based multisensor surveillance — an implementation," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 36, pp. 668–680, Sept. 2006.

[4] A. Bakhtari, M. D. Naish, and B. Benhabib, "Active vision for the autonomous surveillance of dynamic, multi-object environments," in *Proceedings of the 2004 ASME International Mechanical Engineering Congress and Exposition*, (Anaheim, California), Nov. 11–15 2004.

[5] A. Bakhtari, M. Eskandari, M. Naish, and B. Benhabib, "A multi-sensor surveillance system for active-vision based object localization," in *Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, (Washington, D.C.), pp. 1013–1018, Oct. 5–8 2003.

[6] M. D. Naish, *Sensing-System Planning for the Surveillance of Moving Objects.* PhD thesis, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, 2004.

[7] T. Sogo, H. Ishiguro, and M. M. Trivedi, "Real-time human tracking system with multiple omnidirectional vision sensors," in *Systems and Computers in Japan*, vol. 35, pp. 79–90, New York, NY, USA: Wiley-Interscience, 2004.

[8] H. Koyasu, J. Miura, and Y. Shirai, "Real-time omnidirectional stereo for obstacle detection and tracking in dynamic environments," in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, (Maui, HI), pp. 31–36, Oct. 29 – Nov. 3 2001.

[9] T. Matsuoka, M. Araoka, T. Hasegawa, A. Mohri, M. Yamamoto, T. Kirik, N. U. amd Takuya Sugimoto, J. Inoue, and Y. Yamaguchi, "Localization and obstacles detection using omni-directional vertical stereo vision," in *RoboCup 2001: Robot Soccer World Cup V (Lecture Notes in Computer Science)*, vol. 2377/2002, pp. 101–114, Springer Berlin / Heidelberg, 2001.

[10] D. Motonori and A. Yutaro, "Real-time video surveillance system using omni-directional image sensor and controllable camera," in *Proceedings of SPIE Real-time Imaging VII*, vol. 5012, (Santa Clara, CA), pp. 1–9, Jan. 22–23 2003.

[11] M. Greiffenhagen, V. Ramesh, D. Comaniciu, and H. Niemann, "Statistical modeling and performance characterization of a real-time dual camera surveillance system," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, (Hilton Head Island, SC, USA), pp. 335–342, Jun. 13–15 2000.

[12] D. Karuppiah, Z. Zhu, P. J. Shenoy, and E. M. Riseman, "A fault-tolerant distributed vision system architecture for object tracking in a smart room," in *ICVS '01: Proceedings of the Second International Workshop on Computer Vision Systems*, (London, UK), pp. 201–219, Springer-Verlag, 2001.

[13] C. J. Costello, C. P. Diehl, A. Banerjee, and H. Fisher, "Scheduling an active camera to observe people," in *Proceedings of the ACM 2nd International Workshop on Video Surveillance & Sensor Networks*, (New York, NY, USA), pp. 39–45, Oct. 15 2004.

[14] C. J. Costello and I.-J. Wang, "Surveillance camera coordination through distributed scheduling," in *Proceedings of the 4th IEEE Conference on Decision and Control*, (Seville, Spain), pp. 1485–1490, Dec. 12–15 2005.

[15] S.-N. Lim, L. S. Davis, and A. Mittal, "Constructing task visibility intervals for video surveillance," *Multimedia Systems*, vol. 12, pp. 211–226, 2006.

[16] G. Scotti, L. Marcenaro, C. Coelho, F. Selvaggi, and C. Regazzoni, "Dual camera intelligent sensor for high definition 360 degrees surveillance," in *IEE Proceedings of Vision, Image and Signal Processing*, vol. 152, pp. 250–257, Apr. 2005.

[17] J. P. Barreto and H. Araujo, "A general framework for the selection of world coordinate systems in perspective and catadioptric imaging applications," *International Journal of Computer Vision*, vol. 57, no. 1, pp. 23–47, 2004.

[18] T. Wilhelm, H. J. Böhme, and H. M. Gross, "A multi-modal system for tracking and analyzing faces on a mobile robot," *Robotics and Autonomous Systems*, vol. 48, pp. 31–40, Aug. 2004.

[19] Y. Yagi, K. Egami, and M. Yachida, "Map generation for multiple image sensing sensor miss under unknown robot egomotion," *Proceedings of the 1997 IEEE/RS International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1024–1029, Sept. 1997.

[20] N. D. Jankovic, *Development of a Modular Active Omnidirectional Vision System.* M.E.Sc. thesis, Department of Mechanical and Materials Engineering, The University of Western Ontario, London, Ontario, 2005.

[21] N. D. Jankovic and M. D. Naish, "Developing a modular active spherical vision system," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, (Barcelona, Spain), pp. 1234–1239, Apr. 18–22 2005.

[22] N. D. Jankovic and M. D. Naish, "Calibrating an active omnidirectional vision system," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Edmonton, Alberta), pp. 3093–3098, Aug. 2–6 2005.

[23] Michigan Tech, "Nikon fisheye converter FC-E8." [on-line], Sept. 2008. Available from: `http://www.cs.mtu.edu/~shene/DigiCam/User-Guide/lens/lens-fish.html`.

[24] Omnitech Robotics, "ORIFL190-3: 190 degree field of view fisheye lens for 1/3" format cameras." [on-line], Mar. 2007. Available from: `http://www.omnitech.com/pdf/ORIFL190_3%20rev%208.pdf`.

[25] Sunex, "Miniature megapixel fisheye lens." [on-line], Mar. 2007. Available from: `http://www.optics-online.com/Detail.asp?PN=DSL215A`.

[26] Danaher Motion, "Direct drive rotary f-series (frameless) brushless servomotors." [on-line], Feb. 2007. Available from: `http://www.danahermotion.com/website/common/download/document/F_Series_%Part_Number_TB.pdf`.

[27] Hitec RCD USA, "Announced specification of HS-5245MG digital mini metal gear servo." [on-line], Mar. 2007. Available from: `http://www.hitecrcd.com/product_file/file/73/HS5245MG.pdf`.

[28] Heidenhan, "Angle encoders without integral bearing." [on-line], Apr. 2007. Available from: `http://wwwpdb.heidenhain.com/ansicht/download.php?FilePath=..%2Fansicht%t%2FHeidenhain%2Fmedia%2Fimg%2F&FileName=606_136-22.pdf`.

[29] MicroE Systems, "Mercury TM1800 digital output encoder systems factory set resolution to 50 nanometers or 100 nanometers." [on-line], Apr. 2007. Available from: `http://www.microesys.com/pdf/M1800_Data_Sheet.pdf`.

[30] Renco, "RE201 kit incremental encoder." [on-line], Apr. 2007. Available from: `http://wwwpdb.renco.com/ansicht/Renco/media/img/RE201_647165.pdf`.

[31] B. S. Dean, "ROBIN — ROBot independent network, a simple network for RS485." [online], Sept. 15 2008. Available from: `http://www.bdmicro.com/code/robin/`.

[32] Atmel, "8-bit microcontroller with 8K bytes in-system programmable flash: ATmega48/V, ATmega88/V, ATmega168/V." [on-line], Sept. 2006. Available from: `http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf`.

[33] Y. Amit, *2D Object Detection and Recognition: Models, Algorithms, and Networks.* MIT Press, Nov. 2002.

[34] V. Mariano, J. Min, J. Park, R. Kasturi, D. Mihalcik, H. Li, D. Doermann, and T. Drayer, "Performance evaluation of object detection algorithms," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 3, (Washington, DC, USA), pp. 965–969, 2002.

[35] L. G. Shapiro and G. C. Stockman, *Computer Vision.* Prentice-Hall, 2001.

[36] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM computing surveys*, vol. 38, no. 4, pp. 1–45, 2006.

[37] O. Javed and M. Shab, "Tracking and object classification for automated surveillance," in *Lecture Notes in Computer Science*, vol. 2353/2002, pp. 439–443, Springer Berlin / Heidelberg, Jan. 1 2002.

[38] Z. Zhang, Z. Zhang, P. Robotique, and P. Robotvis, "Token tracking in a cluttered scene," *Image and Vision Computing*, vol. 12, pp. 110–120, 1994.

[39] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, second ed., 2004.

[40] A. Gruen and T. S. Huang, eds., *Calibration and Orientation of Cameras in Computer Vision*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.

[41] J. Salvi, X. Armangue, and J. Batlle, "A comparative review of camera calibrating methods with accuracy evaluation," *Pattern Recognition*, vol. 35, no. 7, pp. 1617–1635, 2002.

[42] H. Bakstein and T. Pajdla, "Panoramic mosaicing with a 180° field of view lens," in *Proceedings of the Third Workshop on Omnidirectional Vision*, (Washington, DC, USA), p. 60, 2002.

[43] S. Balasubramanian and D. Aksoy, "Adaptive online scheduling for asymmetric wireless sensor networks," in *2006 International Symposium on Computer Networks*, pp. 73–78, Jun. 2006.

[44] K. Li, "Average-case performance analysis and validation of online scheduling of independent parallel tasks," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, (Los Alamitos, CA, USA), pp. 2–, Apr. 26–30 2004.

[45] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, "On the competitiveness of on-line real-time task scheduling," in *Proceedings of the Twelfth Real-Time Systems Symposium*, (San Antonio, TX, USA), pp. 106–115, Dec. 4–6 1991.

[46] C. Franke, J. Lepping, and U. Schwiegelshohn, "Greedy scheduling with complex obejectives," in *IEEE Symposium on Computational Intelligence in Scheduling*, (Honolulu, Hawaii), pp. 113–120, Apr. 1–5 2007.

[47] K. Anastasova and M. Dror, "Intelligent scheduler for processing help requests on unrelated parallel machines in a computer support administration system," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, pp. 372–377, Oct. 11–14 1998.

[48] D. Shmoys, J. Wein, and D. Williamson, "Scheduling parallel machines on-line," in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science.*, (Los Alamitos, CA, USA), pp. 131–140, Oct. 1991.

[49] J. Miura and Y. Shirai, "Parallel scheduling of planning and action for realizing an efficient and reactive robotic system," in *7th International Conference on Control, Automation, Robotics and Vision*, vol. 1, pp. 246–251, Dec. 2–5 2002.

[50] R. Mattone, L. Adduci, and A. Wolf, "Online scheduling algorithms for improving performance of pick-and-place operations on a moving conveyor belt," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2099–2105, May 16–20 1998.

[51] H. Murao and S. Kitamura, "Online scheduling of a multi-robot system by using genetic algorithms," in *Proceedings of the IEEE International Symposium on Industrial Electronics*, vol. 2, (Pretoria, South Africa), pp. 709–713, Jul. 7–10 1998.

[52] S. Engell and M. Moser, "A benchmark problem in online scheduling," in *Proceedings of the 31st IEEE Conference on Decision and Control*, vol. 1, (Tucson, AZ, USA), pp. 386–391, Dec. 16–18 1992.

[53] J. Sgall, *On-line scheduling.* Springer Berlin / Heidelberg, 1998.

[54] R. L. Givan, E. K. P. Chong, and H. S. Chang, "Scheduling multiclass packet streams to minimize weighted loss," *Queueing Systems*, vol. 41, pp. 241–270, Jul. 2002.

[55] F. Z. Qureshi and D. Terzopoulos, "Surveillance camera scheduling: a virtual vision approach," in *Proceedings of the Third ACM International Workshop on Video Surveillance & Sensor Networks*, (Hilton, Singapore), pp. 131–140, Nov. 11 2005.

[56] S.-N. Lim, A. Mittal, and L. S. Davis, "Task-driven video collection," Tech. Report, CS Dept., University of Maryland and Siemens Corporate Research, Jan. 2006.

[57] S.-N. Lim, L. Davis, and A. Mittal, *Computer Vision — ACCV 2007.* Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007.

[58] A. D. Bimbo and F. Pernici, "Distant targets identification as an on-line dynamic vehicle routing problem using an active-zooming camera," in *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, (Beijing), pp. 97–104, Oct.15–16 2005.

[59] A. D. Bimbo and F. Pernici, "Saccades planning with kinetic tsp for distant targets identification," in *The IEEE International Symposium on Imaging for Crime Detection and Prevention*, (London, UK), pp. 145–149, Jun. 7–8 2005.

[60] A. D. Bimbo and F. Pernici, "Towards on-line saccade planning for high-resolution image sensing," *Pattern Recognition Letters: special issue on Vision for Crime Detection and Prevention*, vol. 27, pp. 1826–1834, November 2006.

[61] J. R. Jackson, "Scheduling a production line to minimize maximum tardiness," Tech. Rep. Research Report Number 43, Management Science Research Project, UCLA., 1955.

[62] D. E. Knuth, *Seminumerical Algorithms, The Art of Computer Programming*, vol. 2. Addison Wesley, 1969.

[63] R. L. Scheaffer and J. T. McClave, *Probability and Statistics for Engineers.* Massachusetts, USA: PWS-KENT Publishing Company, third ed., 1990.

Appendix A

Simulation Details

To gain some insight into the relative performance of the scheduling policies under different target and system conditions, a total of 180 simulated experiments have been carried out — 36 scenarios were investigated for each of the five scheduling policies. The parameters varied in these experiments include: target entry conditions, congestion levels, target to camera speeds, target trajectories, and number of active cameras. Table A.1 shows the parameters used in the simulated simultaneous target entry condition experiments. Table A.2 shows the parameters used in the simulated continuous target entry condition experiments. Finally, the parameters used in the simulated multi-class and single-class experiments are shown in Table A.3.

Table A.1: **Simultaneous entry experiments.**

Simulation	Scheduling Algorithm	# targets	# Cameras	Target Speed (m/s)	Target Trajectory
1	Load Sharing	400	2	1.5	Linear
2		400	4	1.5	
3		50	2	4.75	
4		100	2	4.75	
5		200	2	4.75	
6		400	2	4.75	
7		400	4	4.75	
8		400	2	8	
9		400	4	8	
10	FCFS	400	2	1.5	Linear
11		400	4	1.5	
12		50	2	4.75	
13		100	2	4.75	
14		200	2	4.75	
15		400	2	4.75	
16		400	4	4.75	
17		400	2	8	
18		400	4	8	
19	EDF	400	2	1.5	Linear
20		400	4	1.5	
21		50	2	4.75	
22		100	2	4.75	
23		200	2	4.75	
24		400	2	4.75	
25		400	4	4.75	
26		400	2	8	
27		400	4	8	

continued...

Table A.1: **Continued from previous page.**

Simulation	Scheduling Algorithm	# targets	# Cameras	Target Speed	Target Trajectory
28		400	2	1.5	
29		400	4	1.5	
30		50	2	4.75	
31		100	2	4.75	
32	LSR	200	2	4.75	Linear
33		400	2	4.75	
34		400	4	4.75	
35		400	2	8	
36		400	4	8	
37		400	2	1.5	
38		400	4	1.5	
39		50	2	4.75	
40		100	2	4.75	
41	FLSR	200	2	4.75	Linear
42		400	2	4.75	
43		400	4	4.75	
44		400	2	8	
45		400	4	8	
46		400	2	1.5	
47		400	4	1.5	
48		50	2	4.75	
49		100	2	4.75	
50	Load Sharing	200	2	4.75	Parabolic
51		400	2	4.75	
52		400	4	4.75	
53		400	2	8	
54		400	4	8	

continued...

Table A.1: **Continued from previous page.**

Simulation	Scheduling Algorithm	# targets	# Cameras	Target Speed	Target Trajectory
55	FCFS	400	2	1.5	Parabolic
56		400	4	1.5	
57		50	2	4.75	
58		100	2	4.75	
59		200	2	4.75	
60		400	2	4.75	
61		400	4	4.75	
62		400	2	8	
63		400	4	8	
64	EDF	400	2	1.5	Parabolic
65		400	4	1.5	
66		50	2	4.75	
67		100	2	4.75	
68		200	2	4.75	
69		400	2	4.75	
70		400	4	4.75	
71		400	2	8	
72		400	4	8	
73	LSR	400	2	1.5	Parabolic
74		400	4	1.5	
75		50	2	4.75	
76		100	2	4.75	
77		200	2	4.75	
78		400	2	4.75	
79		400	4	4.75	
80		400	2	8	
81		400	4	8	

continued...

Table A.1: **Continued from previous page.**

Simulation	Scheduling Algorithm	# targets	# Cameras	Target Speed	Target Trajectory
82	FLSR	400	2	1.5	Parabolic
83		400	4	1.5	
84		50	2	4.75	
85		100	2	4.75	
86		200	2	4.75	
87		400	2	4.75	
88		400	4	4.75	
89		400	2	8	
90		400	4	8	

Table A.2: **Continuous Entry Experiments.**

Simulation	Scheduling Algorithm	# targets	# Cameras	Target Speed (m/s)	Entry Rate (tar/min)	Target Trajectory
91		200	2	4.75	100	
92		200	4	4.75	100	
93		200	2	8	100	
94	Load	200	4	8	100	Linear
95	Sharing	200	2	4.75	200	
96		200	4	4.75	200	
97		200	2	8	200	
98		200	4	8	200	
99		200	2	4.75	100	
100		200	4	4.75	100	
101		200	2	8	100	
102	FCFS	200	4	8	100	Linear
103		200	2	4.75	200	
104		200	4	4.75	200	
105		200	2	8	200	
106		200	4	8	200	
107		200	2	4.75	100	
108		200	4	4.75	100	
109		200	2	8	100	
110	EDF	200	4	8	100	Linear
111		200	2	4.75	200	
112		200	4	4.75	200	
113		200	2	8	200	
114		200	4	8	200	
115		200	2	4.75	100	
116		200	4	4.75	100	
117		200	2	8	100	
118	LSR	200	4	8	100	Linear
119		200	2	4.75	200	
120		200	4	4.75	200	
121		200	2	8	200	
122		200	4	8	200	

continued...

Table A.2: **Continued from previous page.**

Simulation	Scheduling Algorithm	# targets	# Cameras	Target Speed (m/s)	Entry Rate (tar/min)	Target Trajectory
123		200	2	4.75	100	
124		200	4	4.75	100	
125		200	2	8	100	
126	FLSR	200	4	8	100	Linear
127		200	2	4.75	200	
128		200	4	4.75	200	
129		200	2	8	200	
130		200	4	8	200	
131		200	2	4.75	100	
132		200	4	4.75	100	
133		200	2	8	100	
134	Load	200	4	8	100	Parabolic
135	Sharing	200	2	4.75	200	
136		200	4	4.75	200	
137		200	2	8	200	
138		200	4	8	200	
139		200	2	4.75	100	
140		200	4	4.75	100	
141		200	2	8	100	
142	FCFS	200	4	8	100	Parabolic
143		200	2	4.75	200	
144		200	4	4.75	200	
145		200	2	8	200	
146		200	4	8	200	
147		200	2	4.75	100	
148		200	4	4.75	100	
149		200	2	8	100	
150	EDF	200	4	8	100	Parabolic
151		200	2	4.75	200	
152		200	4	4.75	200	
153		200	2	8	200	
154		200	4	8	200	

continued...

Table A.2: **Continued from previous page.**

Simulation	Scheduling Algorithm	# targets	# Cameras	Target Speed (m/s)	Entry Rate (tar/min)	Target Trajectory
155		200	2	4.75	100	
156		200	4	4.75	100	
157		200	2	8	100	
158	LSR	200	4	8	100	Parabolic
159		200	2	4.75	200	
160		200	4	4.75	200	
161		200	2	8	200	
162		200	4	8	200	
163		200	2	4.75	100	
164		200	4	4.75	100	
165		200	2	8	100	
166	FLSR	200	4	8	100	Parabolic
167		200	2	4.75	200	
168		200	4	4.75	200	
169		200	2	8	200	
170		200	4	8	200	

Table A.3: **Scheduling architecture experiments.**

Simulation	**Scheduling Algorithm**	**# targets**	**# Cameras**	**Target Speed (m/s)**	**Target Trajectory**	**Architecture**
171	Load Sharing	200	2	4.75	Linear	Multi-class
172	FCFS					
173	EDF					
174	LSR					
175	FLSR					
176	Load Sharing	200	2	4.75	Linear	Single-class
177	FCFS					
178	EDF					
179	LSR					
180	FLSR					