UNIVERSITY OF CALIFORNIA
Santa Barbara

Toward Persistent Tracking and Identification in Camera Sensor Networks

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

Doctor of Philosophy

in

Electrical Engineering

by

Michael James Quinn

December 2008

The Dissertation of
Michael James Quinn is approved:

Professor Upamanyu Madhow

Professor Ken Rose

Professor Amit Roy-Chowdhury

Professor João Hespanha

Professor B. S. Manjunath, Committee Chairperson

September 2008

Toward Persistent Tracking and Identification in Camera Sensor Networks

To my family

Acknowledgements

I would first like to thank my advisor Professor Manjunath for the opportunity to work with him at UCSB. His help and guidance have proved invaluable in my work here. Though I didn't always like the questions, he always asked the right ones to make me think about the big picture of my work.

I'd like to also thank my committee: Professor Madhow, Professor Roy-Chowdhury, Professor Rose, and Professor Hespanha. I'd also like to thank the other faculty and staff who provided guidance and support. Special thanks to Dr. Kenneth Noisewater for the ongoing guidance and inspiration.

I'd also like to thank all of the members of the VRL for their companionship, help, and support: Xinding, Dmitry, Marco, Jiyun, Anindya, Thomas, Nhat, Laura, Zefeng, Luca, Emre, Jae, Jiejun, Jim, Emily, Elisa, and Pratim. Special thanks to Jiyun, Laura, Marco, and Nhat for the added support over the past 6 months.

Thanks to my housemates Steve, Tom, Ed, Laura, Chiou-Wei, Greg, Colleen, Rama, and Kavir who helped make my time away from work fun and, in cases such as the reptile breeding operation in my house, quite educational. Also thanks to Greg and Rainy for providing cheap, flexible housing during my past 6 weeks here.

Thanks also to my various friends I've made during my time here. Lindsay (she wins at yelling), Luke (Bowser!), Joey, Evan, Jonas, Brandon, Rachael, Liz, Pakpoom, Elizabeth, Luke, Phu, Tan, and everyone else I can't think of right this second, You've made life fun out here.

Thanks also to the many representatives of Los Bike scattered throughout the world. You've always had time and space for me when I needed a break. Doug and Lorraine, thanks for the long, icy bike rides and the burritos on the beach in LaJolla. Thanks to everyone in and around the various incarnations of the LBHQ: Josh, Sarah, Ed, Amy, Pat, Erin, Lisa, Brad, Raj, Doug, Dawn, Tom, Dave, Kelly, Jenn, and Bill (as well as the corresponding offspring). You always had a bike, a couch, and a gin & tonic waiting for me when I visited. I'd like to offer special thanks to John and Erin. Long ago I lost count of the number of weekends spent in Berkeley discussing extra solar planets, east Asian cuisine, and the Aqua Teens. Also thanks to Sarah and Cisco for always having room for me on short notice trips down to the border.

I'd also like to thank the UCSB LEAPS program not only for 2 years of funding, but for changing my views both on my own education but also science education in general. Because of you, I'm on a life-long quest to prove that science is cool.

Thanks to my family for the constant support throughout my time here. You've always been there for me and I appreciate your patience with me, the perpetual student. I'd also like to thank my nephew Cameron and my nieces Lily, Elena, and Lucy. You always have a warm welcome ready for Uncle Mike when he comes to visit.

Last, but certainly not least, I'd like to thank my girlfriend Rachel for her ongoing support and understanding through these last several months of my time here. At last "soon" has changed to "now."

Curriculum Vitæ

Michael James Quinn

Education

December 2008	**Doctor of Philosophy** Department of Electrical and Computer Engineering University of California Santa Barbara Santa Barbara, CA
May 1999	**Master of Science** Department of Electrical and Computer Engineering University of Missouri - Rolla Rolla, MO
May 1997	**Bachelor of Science** Department of Electrical and Computer Engineering University of Missouri - Rolla Rolla, MO

Experience

2007-present	**Graduate Research Assistant** Department of Electrical and Computer Engineering University of California - Santa Barbara Santa Barbara, CA
2003-2005	**IGERT Fellow in Interactive Digital Multimedia** Department of Electrical and Computer Engineering University of California Santa Barbara Santa Barbara, CA
2002-2005	**Graduate Teaching Assistant** Department of Electrical and Computer Engineering University of California Santa Barbara Santa Barbara, CA
1999-2002	**Software Engineer** GM Truck Group General Motors Pontiac, MI

1997-1999	**Graduate Research Assistant** Department of Electrical and Computer Engineering University of Missouri - Rolla Rolla, MO
1997-1998	**Graduate Teaching Assistant** Department of Electrical and Computer Engineering University of Missouri - Rolla Rolla, MO

Selected Publications

Michael J. Quinn, Raghuraman Mudumbai, Thomas Kuo, Zefeng Ni, Carter De Leo, and B.S. Manjunath, "VISNET: A Distributed Vision Testbed," Proceedings of the Second International Conference on Distributed Smart Cameras, Stanford, CA, Sept. 2008.

Michael J. Quinn, Thomas Kuo, and B.S. Manjunath, "A Lightweight Multiview Tracked Person Descriptor for Camera Sensor Networks," Proceedings of IEEE International Conference on Image Processing, San Diego, CA, Oct. 2008.

Abstract

Toward Persistent Tracking and Identification in Camera Sensor Networks

Michael James Quinn

In recent years, research in the area of camera sensor networks has accelerated dramatically with the increased availability of cheap sensing, processing, and communications hardware. Design, implementation, and most importantly the operation of camera networks provide numerous challenges for vision researchers.

The first challenge encountered is usually the implementation of a test system in which research can be performed. We provide a detailed overview of our work on the VISNET system. The VISNET system is a ten-node vision testbed located in UCSB's Harold Frank Hall. The system is composed of standard off the shelf hardware, utilizing PCs and IEEE 1394 cameras. The software is developed using freely available resources, including OpenCV and ffmpeg. We present two applications in the VISNET system: distributed network calibration and multicamera tracking.

We then approach the problem of sensor selection in a camera network. We first present a scoring system for selecting camera nodes for localization and tracking. We then extend this system to minimize the number of node activations and handoffs during the tracking process. Second, we present a view scoring system for multiview appearance model learning in camera networks. The system collects the best views from several poses of a tracked person and uses them to assemble a model which captures appearance variation as a function of view angle.

In summary, we present work which advances the current state of camera networks research by providing guidance on both test system construction and system operation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual surveillance is becoming more and more common as increasingly complex systems become commercially available and their use more socially acceptable. Recently, the notion of distributed sensing with a network of tiny sensors has driven both hardware and systems research not only for visual surveillance applications, but for any application which would benefit from a distributed architecture. Such other applications include seismic sensing, acoustic sensing, wildlife monitoring, and environmental health and safety.

The shift toward distributed sensing using small, low-power sensors has shifted the applications and systems research direction away from the traditional idea of processing everything at once. Rather, a new approach is required in order to both harness the benefits of the distributed nature and to take into account the limitations of the small sensors.

1.1 Introduction to Sensor Networks

Since the late 1990s, there has been increased attention given to distributed sensing applications, especially those employing small devices with wireless connectivity. Networks of such devices, more easily deployed due to their wireless nature, have made possible numerous applications which before were difficult or impossible, such as minefield or battlefield surveillance. Such a network is popularly referred to as a “sensor network.”

1.1.1 Sensor Networks Definition

A sensor network is defined as a network of small, wireless, smart sensors which work together to perform a desired task. By smart, we mean nodes capable of local processing and decision making. These networks can contain hundreds or even thousands of nodes and can span tens, hundreds, thousands, or even millions of square meters. Recent advances in hardware, manufacturing, power, and communications technology have made the production of such nodes a reality. While the definition of a sensor network is not rigid, there are several attributes which are commonly used in the definition:

Size: The nodes of a sensor network are typically envisioned as very small sensor-equipped devices. This size requirement leads to the vision of deploying a network of hundreds or thousands of nodes. Large nodes would be cumbersome and more difficult to deploy in a real setting.

Power: The nodes of a sensor network have self-contained power sources. This is typically a battery but could be an alternative source such as a solar cell. The use of battery power immediately gives rise to the problem of conserving power.

Communications: The communications between the nodes of a sensor network is wireless. Additionally, in keeping with the power conservation issues already discussed, a sensor network minimizes internode communication as much as possible.

Operation: A sensor network ideally operates autonomously with minimal human interaction. In addition, operation is distributed rather than centralized. This allows the network to be more fault-tolerant in that nearby nodes can quickly fill in for failed nodes if required.

1.1.2 A Brief Survey of Sensor Networks Research

The modern notion of a sensor network is often credited to the Distributed Sensor Networks (DSN) program at DARPA in the early 1980s [25]. The goal of the program was to develop networks of sensing nodes which operated autonomously. As with so much research, the hardware state of the art lagged far behind the research goals. It wasn't until the late 1990s when processing, communication, and power technology finally began to catch up, sensor networks research exploded into what we see today.

One of the more notable sensor networks projects is the Smart Dust project at UC Berkeley. Spanning from the late 1990s until 2001, the Smart Dust project's end-goal was "a cubic millimeter device with a sensor, power supply, analog circuitry, bidirectional optical communication, and a programmable microprocessor" [65], [36]. The Smart Dust project spawned such entities as the TinyOS operating system and the company Dust Networks which produces wireless sensor network products.

Research at the Center for Embedded Network Sensing (CENS) at the University of California - Los Angeles focuses on "developing wireless sensing systems and applying this revolutionary technology to critical scientific and societal pursuits." [12] The group has several active projects which include animal and plant observation systems, systems and software research, sensor design, and seismic and structural monitoring.

In fact, many research groups list *wireless sensor networks* as their overall focus, but the specific foci of the work are as varied as the people performing it. The Wireless Sensor Networks Laboratory (WSNL) at Stanford University focuses on distributed vision applications and has spawned work in pose and gaze estimation, multicamera gesture recognition systems, and the *MeshEye* smart camera platform [10]. The Robotics, Vision, and Sensor Networks Group (RVSN) at the Massachusetts Institute of Technology has worked with distributed localization and tracking, hardware such as the Cricket location system, and assistive wheelchair systems [5]. Of course, for every application one can dream up, new challenges present themselves which must be addressed in order to realize such a system.

1.1.3 Challenges in Sensor Networks

The first question which comes to mind when a new field such as sensor networks arises is "what is it good for?" or, more specifically, "what can we do with it?" Immediately after comes the question of "how do we accomplish X or Y?" Thus two general challenges of sensor networks are defined.

To answer the first, one needs only to answer the question "what needs to be sensed or measured or detected?" One common target application is building monitoring. If we can embed millions of tiny sensors throughout a building we can keep a constant measure of some aspect of the building's environment such as temperature, light levels, or perhaps concentrations of some chemical. If a sensor exists and can be manufactured in a small footprint, it is likely that a sensor network can be built from it using standard off-the-shelf hardware platforms.

The answer to the second question posed above is quite a bit more complex than that of the first. In fact, both the question and corresponding answer can be broken into parts such as:

- How do we manufacture the sensors needed to accomplish this task?

- How do we build and install the infrastructure needed for this network to work?
- How do we program the system to achieve our desired task?
- How do we optimally extract the desired data from the system in a timely and useful manner?

In fact, the list could go on and on as the tasks and requirements are broken down into finer and finer detail. In the end, we are faced with many problems which need to be addressed in order to have a successful deployment.

For this thesis, we answer the first question with “We would like to monitor people’s movement and activities.” In other words, we aim our sights on utilizing sensor networks in the realm of visual surveillance.

1.2 Camera Sensor Networks

One distributed sensing application which is quickly rising in ubiquity is the visual surveillance network. Such a network traditionally consists of several cameras positioned around a site such as a building or parking lot streaming video feeds to a central bank of televisions where a human operator monitors the activity. Such networks present many challenges to which sensor networks can be applied.

The logical progression of the visual surveillance network takes the form of a camera sensor network. A camera sensor network is just what its name implies: a sensor network composed of camera-equipped nodes. It is here that our challenges become more apparent. The reduced communications resources in a camera sensor network greatly constrain our options for operation.

1.2.1 A Brief Survey of Visual Surveillance Research

Ever since technology enabled the manufacture of video cameras of a manageable size, they have been increasingly used for visual surveillance. One of the first documented uses of closed circuit television (CCTV) surveillance was in the 1960s in England [24]. Since then, they have seen increasing use for human surveillance, in both public and private areas. The surveillance camera is nearly a staple in convenience stores and banks. Casinos have for years utilized sizable visual surveillance systems which allow substantial coverage of all areas of the premises.

One of the larger surveillance research projects in recent history is the Video Surveillance and Monitoring (VSAM) system developed by Carnegie-Mellon University and the Sarnoff Corporation between 1997 and 2001 [26]. The VSAM system aimed at providing a human operator the most useful information available from a variety of sensors. The system included ground-based, vehicle-based, and air-based video sensors. It performed automatic detection and tracking of both pedestrians and vehicles and used a geospatial site model to provide real-time 3D position information.

The W^4 System focused on the surveillance capabilities of single-camera systems. It focused on such areas as object detection, shape recognition, pose recognition, group detection, and carried object recognition [41]. The EasyLiving project at Microsoft Research was aimed at the development of intelligent environments equipped with unobtrusive computing systems. It focused mainly on using computer vision to understand a person's behavior and preferences in order to provide him or her with a comfortable experience in the space.

The $KNIGHT^M$ system was a multicamera surveillance system which allowed object detection, tracking, and activity recognition in cameras with overlapping views as well as those with disjoint fields of view without requiring system calibration. Each camera node reported to a processing server which handled the inter-node processing [48].

The Distributed Interactive Video Arrays (DIVA) at the University of California - San Diego has focused on real-time tracking and activity analysis for security and anti-terrorism applications as well as intelligent environments and smart cars [2]. The technology was installed in downtown San Diego and provided extra security during the 2003 Super Bowl [30].

More recently, the Robot Vision Lab at Purdue University has addressed several different aspects of visual surveillance. Their 18-node wired camera network is used for multiple person tracking research as well as markerless tracking research. One application of note in this network is human monitoring in assisted living situations. The group has also developed wireless camera nodes utilizing the Cyclops camera paired with the Mica-Z mote. These nodes perform color-based tracking in both a uniform manner where all nodes report to a base station as well as a dynamic manner, where an elected cluster head only reports to the base station. In addition to camera network and application development, the group has also developed a lookup table based approach to best-node selection. Their method learns the viewing frustrum of each camera in a network and can quickly ascertain which node is the best for viewing a given point in the space.

The University of Maryland has developed a constrained motion wireless camera node known as the ZipCam. The ZipCam consists of a processing unit fitted with a firewire camera and communications board. The ZipCam is able to move along a horizontal cable, allowing it to reposition itself to better view events. Additionally, the group has active projects addressing camera selection, human activity detection, appearance modeling, tracking, and activity recognition.

The Wireless Sensor Networks Laboratory (WSNL) at Stanford University has recently developed the MeshEye wireless camera mote. The MeshEye features a low-resolution stereo imaging system paired with a high resolution color camera along with a processing unit and wireless communications capabilities. In addition, the group has addressed camera network calibration, utilizing a moving beacon calibration object.

They also have current work on multiple object tracking across overlapping fields of view in multicamera networks. Their work here utilizes object features for ID maintenance. In addition to generic surveillance, the WSNL has branched into applications in both assisted living, virtual reality, and human computer interfaces.

The Visualization and Intelligent Systems Laboratory (VISLab) at the University of California - Riverside has recently begun development on an 80 node network of pan-tilt-zoom smart cameras. The network is intended to facilitate large-scale camera network research and incorporates acoustic, seismic, and vibration sensors for visual sensor triggering. The network will also eventually employ infrared sensing to complement the visual sensing operations. Projects utilizing the system have included calibration, activity classification, facial modeling, and tracking.

1.2.2 Challenges in Visual Surveillance

The most obvious challenge of visual surveillance systems is the reduction or the elimination of the need for a human operator. Unless operators and video feeds are paired one-to-one, the possibility exists of missing an important event. Even then, physical and mental fatigue can reduce an operator's effectiveness in just a short time. By automating even general tasks such as motion detection, the role of a human operator can be shifted to decision making rather than general monitoring. Thus visual surveillance research is constantly pushing toward fully-automated visual surveillance.

Another challenge in visual surveillance is the amount of data produced by constant video feeds. For example, a 640 by 480 pixel color video camera capturing at 15 frames per second produces $(3)(640)(480)(15) = 13824000$ bytes per second or 829440000 bytes per minute or 1.19×10^{-12} bytes per day (that's 1.2 Terabytes). Multiply this by the 10, 20, 50, or more cameras used in many surveillance operations and even with the video compression technology available, we see that the transmission or storage of such data is impossible.

The approach to this problem is twofold. First, we desire to extract, in real time, only the interesting portions of the surveillance video. Hours and hours of an empty hallway are not interesting from a surveillance point of view and so there is no need to store such video. The second half of the solution to the data size problem is to efficiently store those frames, objects, or events deemed important by the first part. If a person walks through a scene and is captured on $10,000$ frames, do we need to save all of these frames? Is there a better way to represent both the person's appearance as well as motion or activity?

Using the commonly-accepted rule of thumb that communication is significantly more expensive (between 1000:1 and 10000:1 according to [87]) than processing, we desire to contain as much processing as possible at the node and communicate only small, vital information between nodes. In addition to local processing of data, the selection of only the most vital nodes in a network functions to reduce the number of nodes working on any problem at once. At the broadest level, we would only assign those sensors which actually sense an event but further analysis might allow us to assign only a handful of the best-suited sensing nodes to a particular process.

Another challenge in surveillance networks is maintaining persistent identification of a tracked person as he or she moves through the system. In a multicamera system, the nodes must be able to determine if they are seeing the same person. Additionally, in the case of nodes with disjoint views, they must be able to determine if the person who just entered one region is the same person who recently left another. Thus some model of each person's appearance should be built in order to have a global reference for the system. Like memory in humans, this global reference allows a system to compare tracked people with already-known appearances and make a decision about identity.

1.3 Objectives

The objective of this dissertation is to address the problem of persistent tracking and identification in a visual sensor network. Specifically, we approach three problems which are detailed below.

Camera Network Testbed Design (Chapter 2)

We address the practical issues of designing and implementing a real-time camera network in a small-scale setting. The objective is to provide a practical reference for those wishing to develop a similar system for vision research.

Sensor Selection for Localization in Camera Networks (Chapter 3)

We address the issue of sensor tasking for localization and tracking applications in camera networks. The objective is to minimize the error of a specific application while at the same time minimizing node activity within the network.

Multiview Human Modeling in Camera Networks (Chapter 4)

For human tracking and identification applications, we propose a multiview appearance model for tracked humans in a camera network. The objective is to provide a lightweight model for transmission between nodes of such a network. Additionally, the model should be updateable at a central level in order to allow the fusion of multiple inputs.

1.4 Summary of Contributions

The main contributions of this dissertation are as follows:

- **Camera Sensor Network Testbed Design:** (Chapter 2) We discuss the design and development of a multinode camera network testbed which emulates a node cluster in a larger-scale network. Our network uses off the shelf components and freely-available standard software tools which make development move quickly.

We introduce a procedure for the distributed calibration of a such a network. We then present a simple tracking application which demonstrates the full functionality of the testbed network.

- **Sensor Tasking for Localization in Camera Sensor Networks:** (Chapter 3) We address the problem of assigning tasks to the nodes in a camera sensor network. We explore the factors affecting localization and tracking in a camera network and develop scoring criteria for choosing the best nodes for this task. The criteria are based upon the topology of the network and the position and movement of the tracked person.

- **Parametric Human Modeling in Camera Sensor Networks:** (Chapter 4) We address the issue of multiview appearance modeling of tracked people in camera networks. We first develop a view scoring system for multiview image collection. We then offer a lightweight appearance model for storage and transmission of the multiview appearance data.

1.5 Thesis Organization

The rest of this dissertation is organized as follows:

In Chapter 2, we discuss the design, development, and operation of the VISNET system. We discuss calibration and tracking applications developed for the VISNET system and show results. In Chapter 3, we present a multiview parametric appearance model for use in camera sensor networks. We discuss its utility and demonstrate its effectiveness for differentiating tracked people. In Chapter 4, we present a method for predicitive sensor selection for tracking in camera sensor networks. We demonstrate via simulation the advantage of this method over simply using the current tracked object state for sensor assignment. Finally, in Chapter 5, we provide conclusions and future directions.

Chapter 2

Camera Sensor Network Testbed Design

2.1 Introduction

Like any other field, research focusing on sensor networks and visual surveillance relies on simulation for algorithm development and demonstration. However, at some point the transition must be made from simulation to real-world implementation in order to both demonstrate feasibility and to address other issues which may not manifest themselves in simulation. Unfortunately, the design and development of a real-world system is often not trivial and a considerable amount of time can be spent on this stage before any algorithm implementation can be performed.

In this chapter, we discuss the design of small-scale camera networks. Specifically, we discuss the development of UCSB's VISNET system. We begin with a general discussion on large-scale sensor network operation and how its operation can be reduced to a much smaller scale due to the properties of typical sensors. We then present the motivation and design considerations for the VISNET system, followed by a discussion of the final system design.

We then present two projects performed using the VISNET system. First is a camera network calibration procedure which is demonstrated for 3D point localization. This is followed by a demonstration of a simple multicamera human tracking application which utilizes a graphical user interface for system operation monitoring.

2.1.1 Large-Scale Camera Network Operation

The current state of hardware and sensing technology makes possible the construction and operation of sensor networks on a large scale. Such a network might span thousands or even millions of square meters. Ideally, the data collection and processing of such a network would be centralized so that all data is taken into account on a global level. Such a centralized communications architecture is shown in Figure 2.1. In a network larger than a few tens of nodes, however, this approach is not typically feasible. The bandwidth necessary for such communication is prohibitive, especially when dealing with "heavy" data such as video in a camera network.

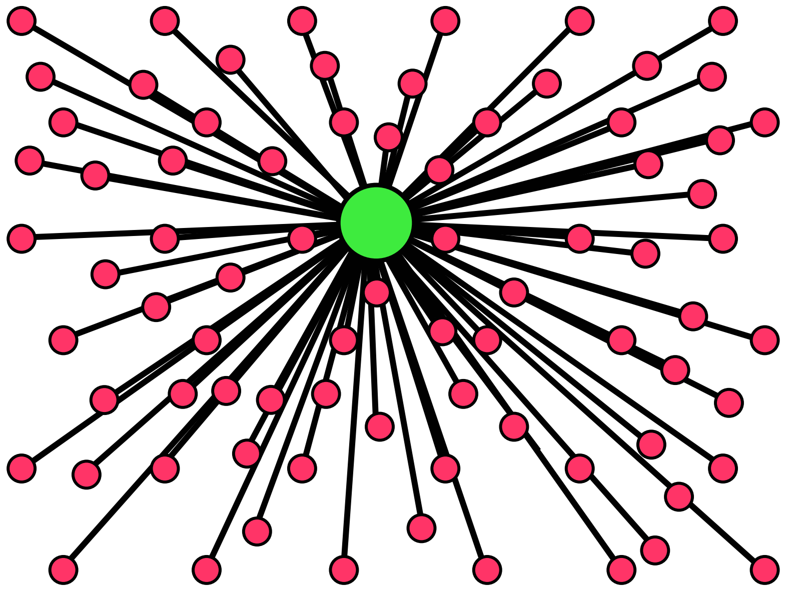


Figure 2.1: Centralized Network Architecture

Fortunately, most events in a sensor network are perceived by a small subset of the total nodes present in the system. This is especially true of camera nodes which have limited field of view and limited effective sensing range. By recognizing this limitation of the nodes, a system can adapt to sensed events in order to keep communications and processing of an event local to the event. We can think of this as a problem of how to cluster nodes in order to handle local events. One approach is to use fixed clusters, such as shown in Figure 2.2. This approach effectively partitions the network into local clusters, but may present a problem when sensing an event on the boundary of two clusters. In this case, inter-cluster collaboration is vital.

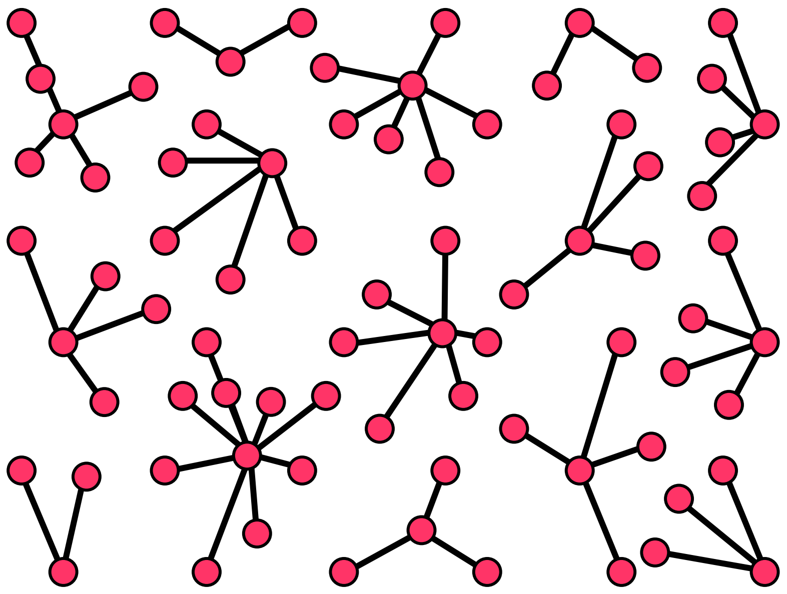


Figure 2.2: Sensor Network with Fixed Clusters

Another approach which has been used in sensor networks research involves the formation of dynamic node clusters which adapt with the movement of an object or event. Figure 2.3 illustrates this concept. Such a cluster of entities is often referred to as an *agency* in the realm of artificial intelligence [57] [14]. As an event moves through the network, nodes which no longer sense the event are removed from the cluster while newly-sensing nodes are added. The choice of the criteria for the inclusion or exclusion of a node from such a sensor cluster is itself a growing research problem.

Regardless of the sensing architecture adopted, the fact still remains that for each event only a small number of nodes (when compared with overall sensor node population) are involved in sensing, recording, or analyzing the event. Thus for any local

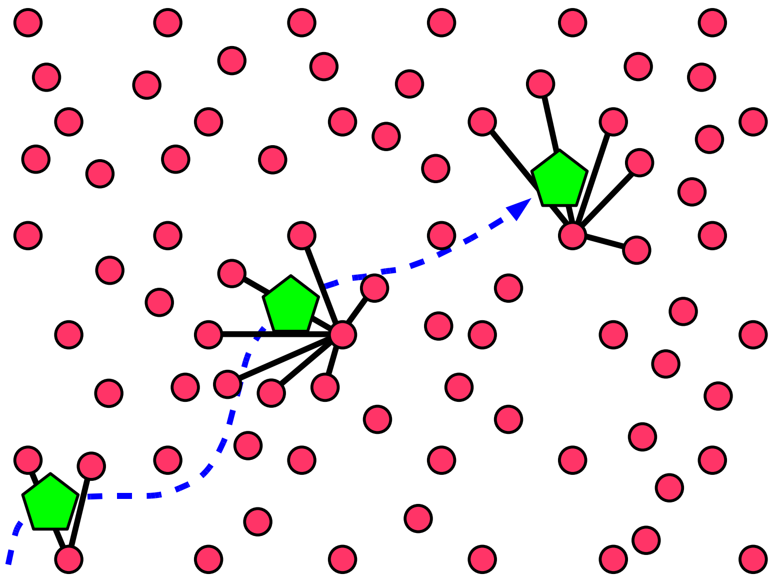


Figure 2.3: Dynamic Clustering in a Sensor Network. The nodes self-organize into clusters to provide covereage of the pentagonal object during its motion through the space.

sensing or vision tasks, we work solely with those sensors which are currently seeing or detecting the local events. We can model this subset or cluster as a small centralized network, as shown in Figure 2.4. It should be noted here that the central, or “sink,” node in this network is not necessarily a dedicated processing node. It can be a process which runs alongside the sensing process in a sensor node. However, for ease of discussion we treat this process as a separate entity.

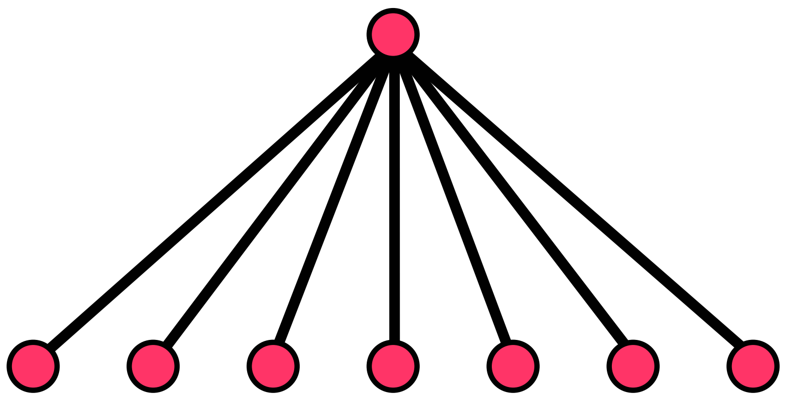


Figure 2.4: Centralized Sensor Cluster Architecture

2.1.2 Experimental Camera Networks

A very important stage of vision research is the demonstration of algorithms or processes with real-world data and, in some cases such as with visual surveillance, in real-time. This has prompted numerous research groups to design and build experimental networks for development and demonstration of vision algorithms.

One well-known project, VSAM (Video Surveillance and Monitoring) [26], incorporated multiple types of sensors including Pan-Tilt-Zoom (PTZ) cameras, infrared cameras, omnidirectional cameras, airborne cameras, and relocatable vehicle-mounted cameras. While the VSAM project yielded an impressive amount of research, a project of such scale is out of reach for most groups. In addition, the local nature of the data and processing in a camera network allow researchers to emulate a small subset of a larger system using a more manageable number of nodes.

More common is the smaller-scale network such as is found in a university research setting and contained within one or two small areas. An example of such a network is used in [32]. This network consisted of 8 PCs each with 2 cameras attached. The nodes reported to a ninth PC known as the cluster head. The $KNIGHT^M$ system [48] [47] [50] consisted of camera nodes consisting of a PC with attached camera and used cameras with both overlapping fields of view and cameras with non-overlapping fields of view for tracking and recognition work.

the EasyLiving system [51] used two PCs, each equipped with a stereo camera module. The cameras reported to a third PC which handled the tracking process. In [15] a single overhead wide-angle camera is used along with four semi-mobile PTZ cameras which are used for active vision applications.

One trait common to many experimental networks is the use of off-the-shelf hardware. In the interest of time, vision researchers typically prefer not to venture into the realm of hardware development instead opting for commercially-available products. In addition the use of PCs with attached cameras is more popular than the use of embedded systems. The use of more standard hardware allows the use of standard software and operating systems, speeding up the learning and development processes.

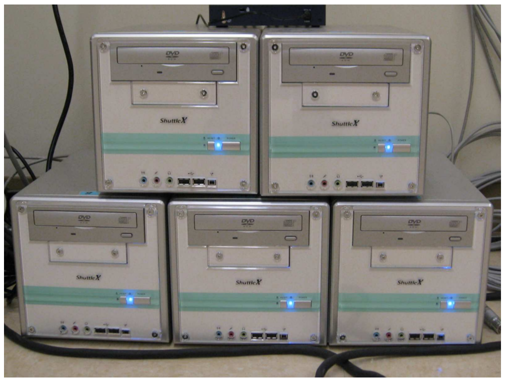
2.2 The VISNET System

In 2005, design began on UCSB's Visual Sensor Network (VISNET). VISNET is a ten-node experimental camera sensor network testbed located in Harold Frank Hall on the UCSB campus. VISNET was installed in early 2006 and has proven itself useful to the vision community at UCSB.

The motivation behind the VISNET system was the need for a multiple-node camera network for vision and surveillance research. The goal was an easy-to-use network with a minimal learning curve for users. The major constraint on the project was the available space for installation. The space is a room approximately 6 meters wide by 10 meters long by 2.8 meters tall. The size does not allow for expansive operation, but does provide enough space for local processing of tracked people or objects.

2.2.1 Hardware Construction

VISNET is comprised of ten networked sensor nodes and a single processing node. Each sensor node consists of a small-profile PC with an attached Point Grey Firefly2 IEEE 1394 webcam. The central processing node is also a standard PC. Figure 2.5 shows one of the VISNET nodes. The full VISNET system layout is illustrated in Figure 2.6. For ease of deployment and operation, the VISNET system uses a wired network for communications purposes.


(a) PC Nodes

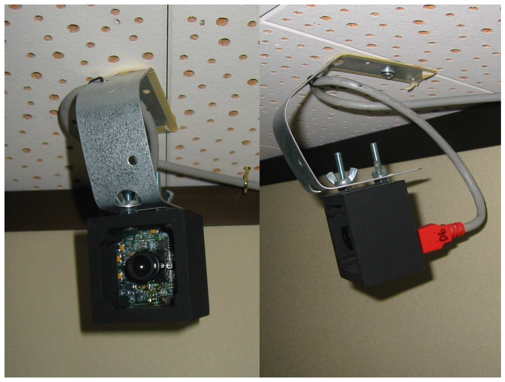

(b) Mounted Camera

Figure 2.5: VISNET Hardware Detail. Each node is a small form factor PC with attached webcam.

The physical VISNET network layout is shown in Figure 2.7. Figure 2.7(a) shows an overhead view of the installation and Figure 2.7(b) shows a 3D orthographic reconstruction of the system. The cameras are distributed fairly uniformly about the perimeter of the room and are mounted approximately 2.65 meters from the floor. Each camera has a slight downward rotation about its x-axis (defined in Appendix A).

Each camera is equipped with a wide angle lens and provides a good view of the space. Figure 2.8 shows the views from all ten VISNET nodes. We see that the cameras' fields of view overlap considerably, which fits well with our idea of a dense sensor network. A short wall is installed in one end, visible in nodes 4, 5, and 6 (Figures 2.8(d),2.8(e),and 2.8(f)). This is used to obscure operators and observers during operation and demonstration.

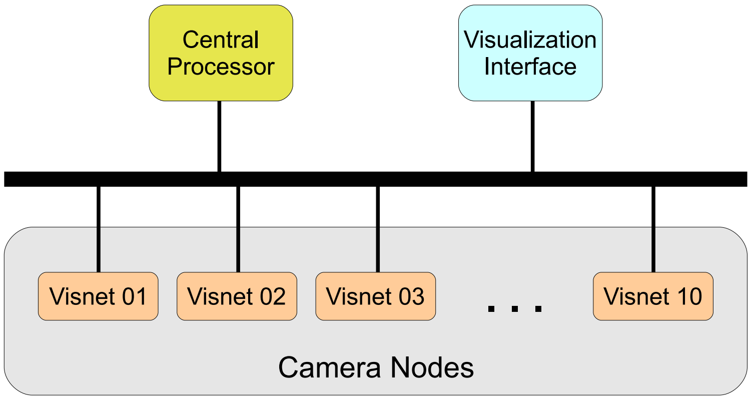
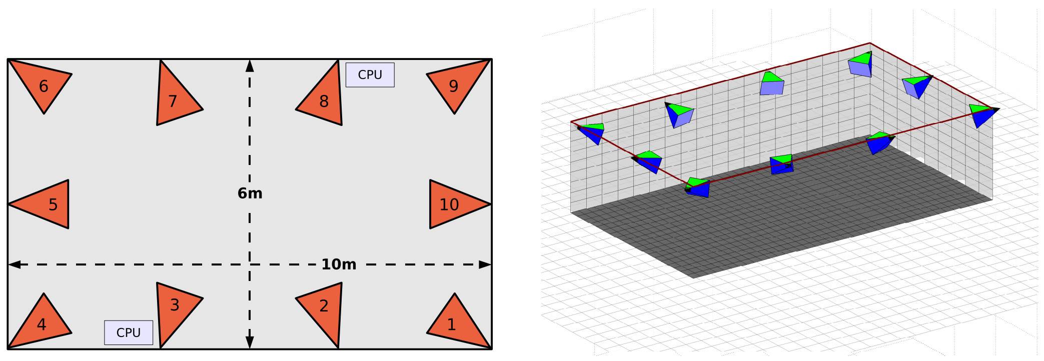


Figure 2.6: VISNET Block Diagram. The system consists of 10 sensor nodes, a processing node, and a system visualization node.

2.2.2 Software Design

The software design in VISNET is based upon that of Spheres of Influence [81], an installation in UCSB's Davidson Library. The nodes of VISNET run the Ubuntu linux operating system [11]. The use of linux allows easy administration of the machines. Additionally, it allows easy remote access, remote management, and remote execution of software.

All VISNET software is written in C using various free or open source resources. The general software architecture of each node is shown in Figure 2.9. The basic components are the capture module, the processing module, and the communications module.



(a) Overhead View (b) 3D View

Figure 2.7: VISNET Physical Layout

Video Frame Capture

The first step in any vision process is the acquisition of video. Camera control and capture in VISNET is performed using the *libdc1394* libraries available from [4]. These libraries allow direct control over camera parameters such as brightness, color balance, capture speed and capture format. Libraries for easy camera initialization and capture were designed in order to accelerate the development of applications in VISNET.

Video Processing

Once captured, the imagery from a camera node must be processed and analyzed. In VISNET, this processing is performed using the OpenCV open source computer vision libraries, originally developed by Intel. The libraries, available from [7], contain common functions for image and video processing as well as for image and video reading

(a) Node 1 View

(b) Node 2 View

(c) Node 3 View

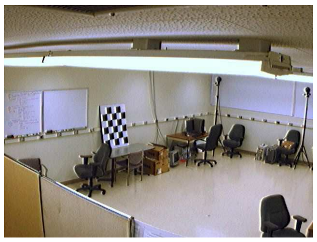
(d) Node 4 View

(e) Node 5 View

(f) Node 6 View

(g) Node 7 View

(h) Node 8 View


(i) Node 9 View

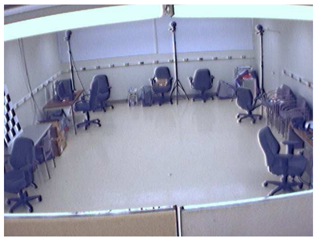

(j) Node 10 View

Figure 2.8: VISNET Node Views

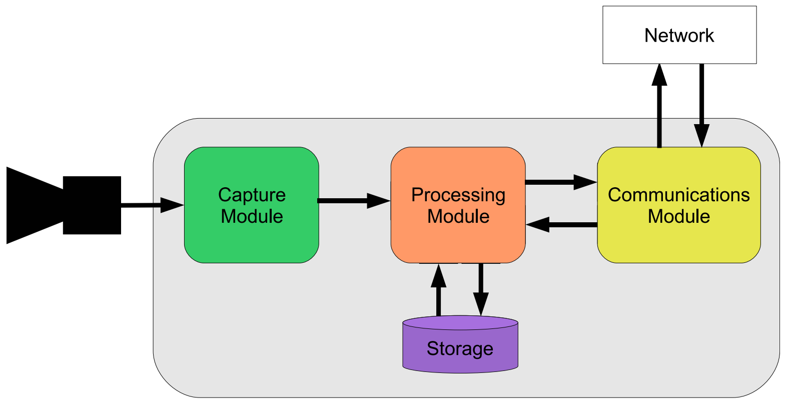


Figure 2.9: VISNET Node Software Architecture

and writing. OpenCV accelerates the development of real-time image and video processing applications by providing commonly-used lower-level functions such as edge detection, camera calibration, tracking, and optical flow.

Internode Communications

Internode communications in VISNET is accomplished using the Open Sound Control (OSC) protocol. Originally, intended for communication between musical instruments, OSC provides a compact, easy to use framework for communicating between networked computers. Using OSC, we have constructed a simple communications protocol for the operation of visnet. Each message in OSC begins with a prefix indicating the type of message. Table 2.1 lists the different message types developed for VIS-

NET. Table 2.2 fully describes a message describing a tracked object in a camera node. Descriptions of all messages use in the VISNET system can be found in Appendix C.

Message Prefix	Description	Node Send	Node Receive	Tracker Send	Tracker Receive
/extrinsics	Extrinsic Parameters	Y	Y	N	N
/relextr	Relative Extrinsic Parameters	Y	N	N	Y
/obj	Tracked Object Data	Y	N	N	Y
/ctrl	Node Control (Start / Stop)	N	Y	Y	N

Table 2.1: VISNET Message Types

Message Component	Description
/obj	Indicator of object data to follow
Node_ID	ID of originating node
time_sec	Timestamp of data (seconds since epoch)
time_usec	Timestamp of data (microseconds)
u	Local u coordinate of tracked object
v	Local v coordinate of tracked object

Table 2.2: VISNET Object Tracking Message Detail

Outgoing OSC communication at each node is handled by a node client and incoming communication is handled by a node server running in parallel to the main application. A similar setup runs on the tracking node.

Time Synchronization

In most applications involving the fusion of data from multiple nodes, correlation of data in time is a vital step in any application. Ideally, we would like to have direct control over the collection of data, ensuring exact synchronization of data. Unfortunately, the synchronization of data capture in all nodes in a large network is impossible. Instead, we choose to simply synchronize the system clocks of the nodes with each other. By doing so, we can attach a timestamp to data collected in each node, allowing the fusion process to determine which data were taken at similar times. Node clock synchronization is accomplished using the Simple Network Timing Protocol (SNTP) [6].

Used in systems such as that presented in [21], SNTP works by synchronizing the clocks of the network nodes with a server node. In our case, the nodes of VISNET are synchronized with the tracker node, which in turn is synchronized to the NTP server at *ntp.ubuntulinux.org*. By utilizing SNTP, we can achieve time synchronization for our applications. Figure 2.10 shows the clock offset of a VISNET node from several days of operation. We see that the maximum offset is $\pm 10ms$, with the variance $2.4ms$ which, even with 30 frames per second capture rate, is sufficient, given the typical moving speed of humans, our primary subject.

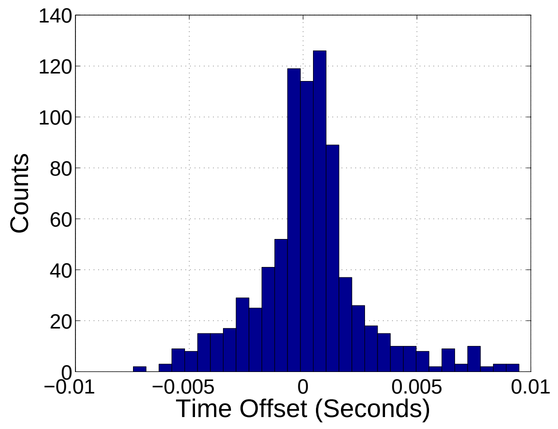
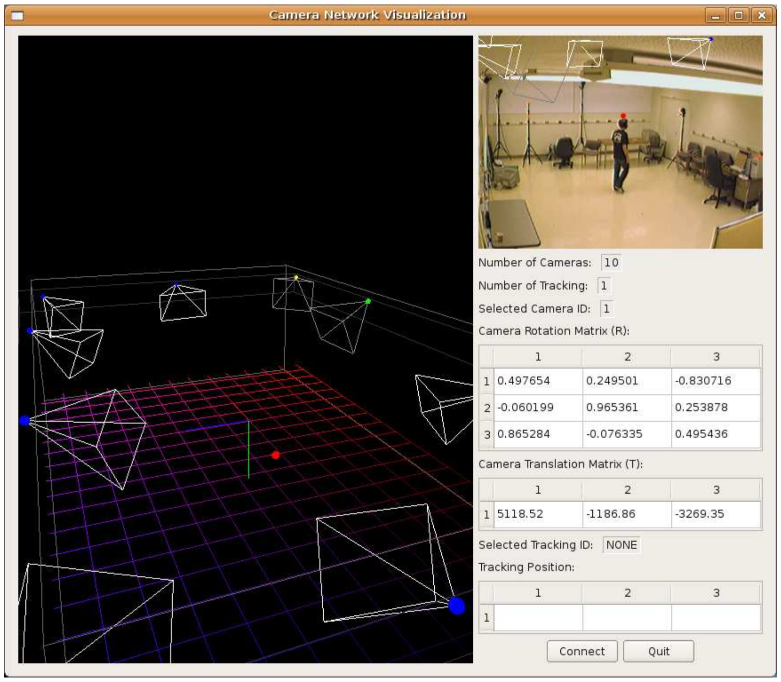


Figure 2.10: Node Clock Offset Using NTP

Network Operation User Interface

In order to provide user interaction for both development and demonstration purposes, a user interface was built which allows both control and monitoring of network operation. This was built using the QT toolbox [9] along with OpenGL [8] for the 3D portion. The visualization node communicates with the tracking node using a custom message protocol. Upon initialization, the visualization node requests the network calibration information, allowing it to visualize the layout of the system.

The user is able to see points tracked in the system displayed in real time. In addition, the user is able to select individual nodes in order to receive the current view from that node. These frames are encoded using MPEG-4 and transmitted to the visualization node for viewing. The user interface is shown in Figure 2.11. As discussed previously, during normal operation a camera network should not transmit video frames but for the visualization, we violate this rule with the understanding that it is for demonstration purposes only.



Number of Cameras: 10

Number of Tracking: 1

Selected Camera ID: 1

Camera Rotation Matrix (R):

	1	2	3
1	0.497654	0.249501	-0.830716
2	-0.060199	0.965361	0.253878
3	0.865284	-0.076335	0.495436

Camera Translation Matrix (T):

	1	2	3
1	5118.52	-1186.86	-3269.35

Selected Tracking ID: NONE

Tracking Position:

	1	2	3
1			

Figure 2.11: VISNET Visualization Interface

2.3 VISNET Calibration

In order to achieve multicamera operation, we require a calibrated network. The calibration of camera networks is a well-researched problem which has yielded various proposed solutions. However, many are centralized [76] [79] or require camera model simplification [77]. We present a simple, effective calibration procedure which lends itself well to distributed operation.

2.3.1 Relative Camera Calibration

The goal of our calibration procedure is to establish pairwise relationships between nodes in the network. Such a relationship is represented by the transformation between the two cameras' coordinate systems. The camera coordinate system is detailed in Appendix A. This transformation takes the form of a 3D rotation and a 3D translation, represented by R and T, respectively.

Our calibration approach builds upon the widely-used procedure from [86], which is summarized in Appendix A. We utilize a chessboard calibration pattern to define world points in our system. When cameras C_i and C_j both see the pattern, they are are able to calibrate themselves with the world coordinate system defined by the pattern. This process is illustrated in Figure 2.12 . The calibration yields the extrinsic parameters (R_{Wi}, T_{Wi}) and (R_{Wj}, T_{Wj}) for cameras i and j, respectively.

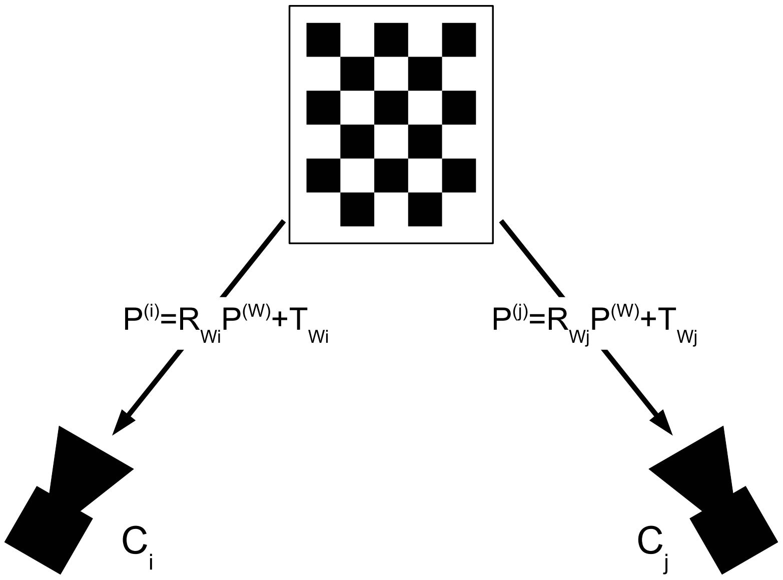


Figure 2.12: Pairwise Calibration Process

However, we are not interested in the cameras' positions with respect to the chess-board. We are instead interested in their relative relationships. Given the tranformation of a point in the world coordinate system $P^{(W)}$ into the cameras' coordinate systems:

$$P^{(i)} = R_{Wi}P^{(W)} + T_{Wi} \tag{2.1a}$$

$$P^{(j)} = R_{Wj}P^{(W)} + T_{Wj} \tag{2.1b}$$

we can calculate transformations between the points in the cameras' coordinate systems:

$$P^{(i)} = R_{Wi}(R_{Wj}^{-1}(P^{(j)} - T_{Wj})) + T_{Wi} \tag{2.2a}$$

$$P^{(j)} = R_{Wj}(R_{Wi}^{-1}(P^{(i)} - T_{Wi})) + T_{Wj} \tag{2.2b}$$

or, simplifying:

$$P^{(i)} = R_{ji}P^{(j)} + T_{ji} \tag{2.3a}$$

$$P^{(j)} = R_{ij}P^{(i)} + T_{ij} \tag{2.3b}$$

where

$$R_{ji} = R_{Wi}R_{Wj}^{-1} \tag{2.4a}$$

$$R_{ij} = R_{Wj}R_{Wi}^{-1} \tag{2.4b}$$

and

$$T_{ji} = -R_{Wi}R_{Wj}^{-1}T_{Wj} - T_{Wi} \tag{2.5a}$$

$$T_{ij} = -R_{Wj}R_{Wi}^{-1}T_{Wi} - T_{Wj} \tag{2.5b}$$

With this correspondence established, we are able to transform points viewed by cameras C_i and C_j into a common coordinate system for localization. This coordinate system may be either of the cameras' individual coordinate systems or a third system for which the transformation from at least one of the cameras is known.

2.3.2 Network Relative Calibration

It is not typically the case that all cameras in a network can see the same feature points and thus the use of a single set of stationary calibration points is not an option. Our approach utilizes a moving set of points - in our case the chessboard. By moving the chessboard through the network, pairwise geometric relationships can be established by cameras whose fields of view overlap.

Our calibration procedure is outlined in Algorithms 1 and 2. The procedure involves the movement of the chessboard pattern through the space (Figure 2.13) during the calibration period. If the chessboard pattern is detected by camera node i, the node performs its own calibration with respect to the chessboard's current position and saves the resulting extrinsic parameters along with the current time. This calibration yields the extrinsic parameters $R_{Wi}(t_n)$ and $T_{Wi}(t_n)$. This set of parameters is then broadcast across to the other nodes in the network.

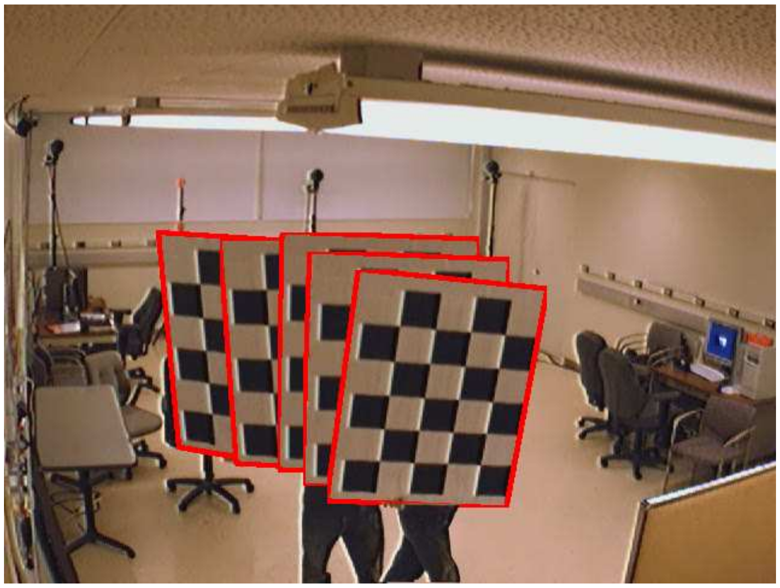


Figure 2.13: Chessboard Movement During Calibration Process

When a node receives a set of extrinsic parameters from another node in the network, it checks the timestamp of the incoming data to its own collection of parameters to check for a time match. If a match is made, the two sets of parameters are used to calculate a relative relationship, as in Equation 2.2. Over time, this process results in many sets of parameters between each pair of cameras with overlapping fields of view.

```
Input: Video Frames
Output: Local Extrinsic Parameters
$start\_time = current\_time$
$elapsed\_time = 0$
while $elapsed\_time < tmax$ do
    $curframe$ = Capture_ Frame()
    $detected$ = Detect_Chessboard()
    if $detected$ then
        $(R, T)$ = Compute_Extrinsics()
        Broadcast_Extrinsics()
    end
    $elapsed\_time$ = $current\_time$ - $start\_time$
end
```

Algorithm 1: Camera Network Calibration (Send) Process

Figure 2.14 shows a typical extrinsic parameter distribution result for a single pair of cameras.

Another option for this process is for each node to collect its entire collection of parametes during the calibration period and only then send out a single message containing them all. Both methods have yielded identical results in our experiments.

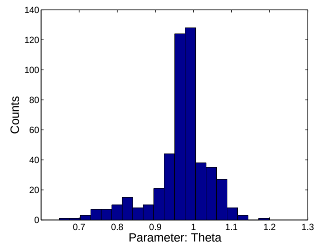
```
Input: Remote Extrinsic Parameters
Output: Relative Extrinsic Parameters
$start\_time$ = $current\_time$
$elapsed\_time$ = 0
while $elapsed\_time < tmax$ do
    Receive Incoming Messages
    if $timestamp\_match$ then
        $(R_{ij}(t), T_{ij}(t)) =$ Compute_Relative_Extrinsics()
        Save_Relative_Extrinsics()
    end
    $elapsed\_time$ = $current\_time$ - $start\_time$
end
Compute Mean Extrinsic Parameters
```

Algorithm 2: Camera Network Calibration (Receive) Process

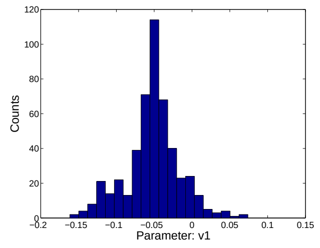
We take as our estimate for the cameras' relative parameters the mean of each individual parameter taken over the set collected over the calibration period. The accuracy of this estimate is discussed in the next section.

2.3.3 World Reference

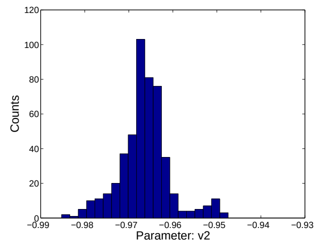
With knowledge of the relative positions of the cameras, we next compute a common reference for all nodes. As discussed in Appendix A, for analysis and visualization



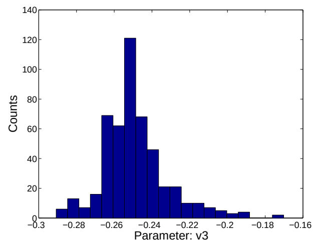
(a) θ



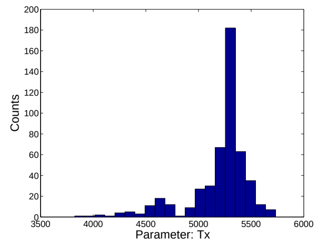
(b) v_1



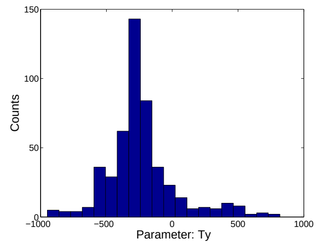
(c) v_2



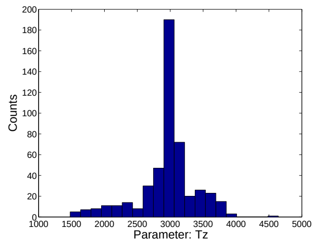
(d) v_3



(e) T_x



(f) T_y



(g) T_z

Figure 2.14: Pairwise Extrinsic Parameter Distributions. The parameters represent the rotation with a rotation angle θ and the axis of rotation $\vec{v} = [v_1\ v_2\ v_3]^T$. The translation is represented by the vector $T = [T_x\ T_y\ T_z]$.

purposes, we typically want this reference to have some meaning in the local environment. Thus we add an additional step to our calibration process. This step consists of placing the chessboard in a desired world origin and calibrating the network with it. What results is what is commonly called a *vision graph* [31]. The edges of the vision graph indicate successful pairwise calibration between the connected nodes. The final world reference calibration step puts an extra node, the "World" node, into the graph, yielding a graph similar to that in Figure 2.15. In order to establish a world reference for each node, we trace each node's shortest path to the world node, accumulating the relative transformations along the way. This process yields a set of extrinsic parameters $\{R_{Wi}, T_{Wi}\}_{i=1}^{N}$ where N is the total number of nodes in the network.

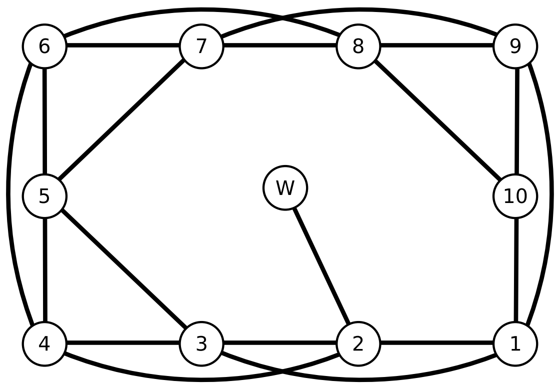


Figure 2.15: VISNET Vision Graph

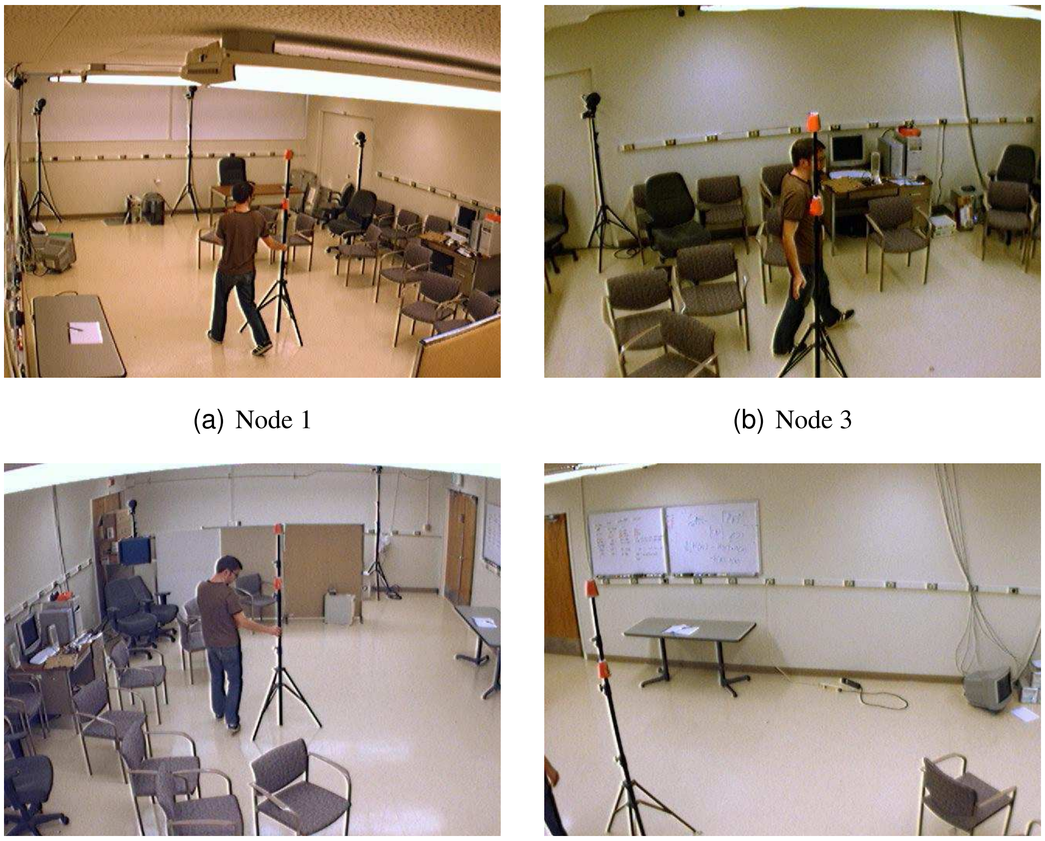
2.3.4 Results

To demonstrate the performance of our calibration process, we set up an experiment to measure localization error of the system. We moved a rigid structure around to 17 different positions (yielding 34 total points) in the space and recorded still images at each position. The structure, a modified tripod, had two points on it: one placed at

1500mm above the floor and one placed at 2000mm above the floor. Several frames of this data are shown in Figure 2.16, with the 3D ground-truth point positions shown in Figure 2.17. The 340 image points were manually extracted in order to obtain accurate image locations of the test points. Using the localization technique discussed in Appendix B, we found the 3D estimates of each of the 34 test points. Figure 2.18(a) compares the point estimates with the groundtruth positions. Figure 2.18(b) shows the 3D localization error for all 34 points.

For comparison, we took our calibration data and output and performed global bundle adjustment on it. Bundle Adjustment is a centralized global optimization process which takes as its input camera parameter estimates along with 3D point location estimates. It is a simultaneous refinement of both the 3D point estimates and the camera parameters. More information can be found in [79] and [54].

For our points for the bundle adjustment process, we used the chessboard points collected during the calibration process. The resulting camera network is reconstructed in Figure 2.19. Performing the localization experiment on the 34 previously-used data points, we arrived at the results shown in Figures 2.20(a) and 2.20(b). We see that the localization error before and after performing bundle adjustment are comparable.



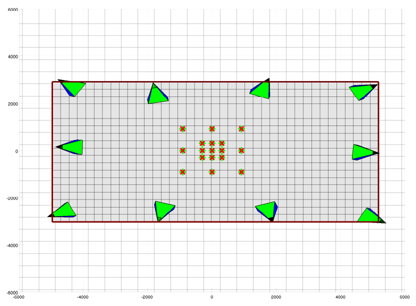
(a) Node 1 (b) Node 3

(c) Node 5 (d) Node 6

Figure 2.16: 3D Localization Experiment Snapshots

2.4 Human Tracking in VISNET

Conceived as a human surveillance testbed, VISNET's first application after network calibration was human tracking. The primary goal of this application was to establish a framework for future tracking applications in VISNET. The secondary goal was to demonstrate the accuracy of the system calibration procedure.


(a) Overhead View

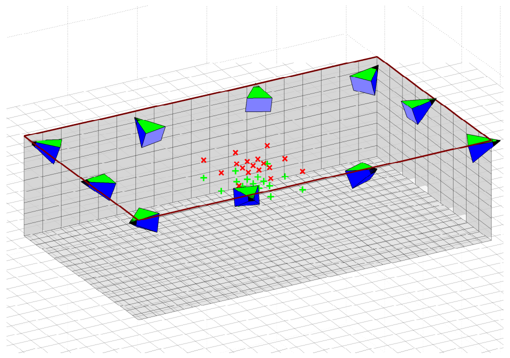
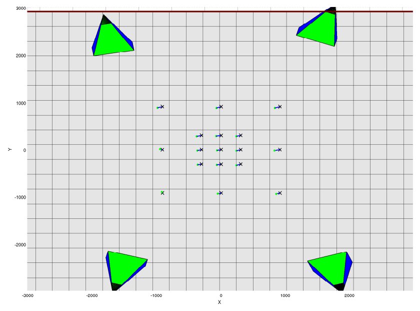
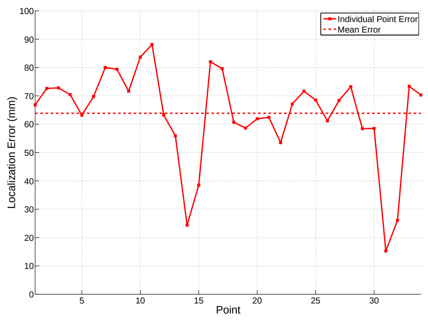

(b) Overhead View

Figure 2.17: 3D Localization Experiment Groundtruth


(a) Comparison with Groundtruth



(b) Localization Error

Figure 2.18: 3D Localization Experiment Results

2.4.1 Nodal Detection

Our approach to human segmentation and location in the camera nodes uses standard algorithms. For segmentation, we use the well-known process of background subtraction. Using a single Gaussian background model in the YUV color space, we

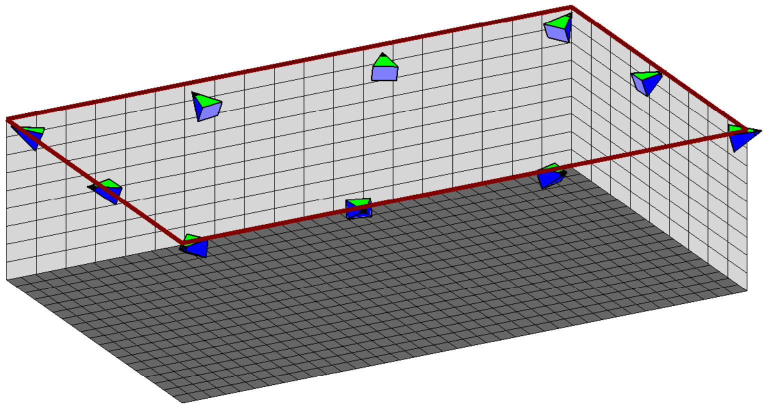
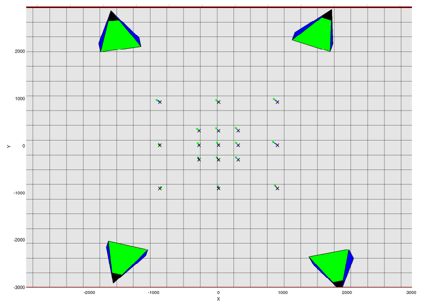
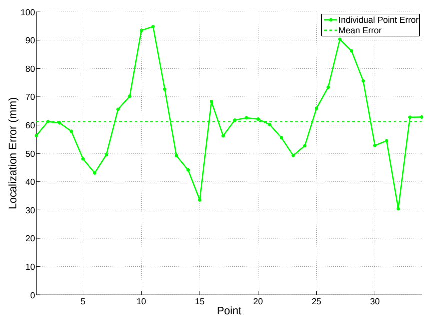


Figure 2.19: Network Visualization - After Bundle Adjustment



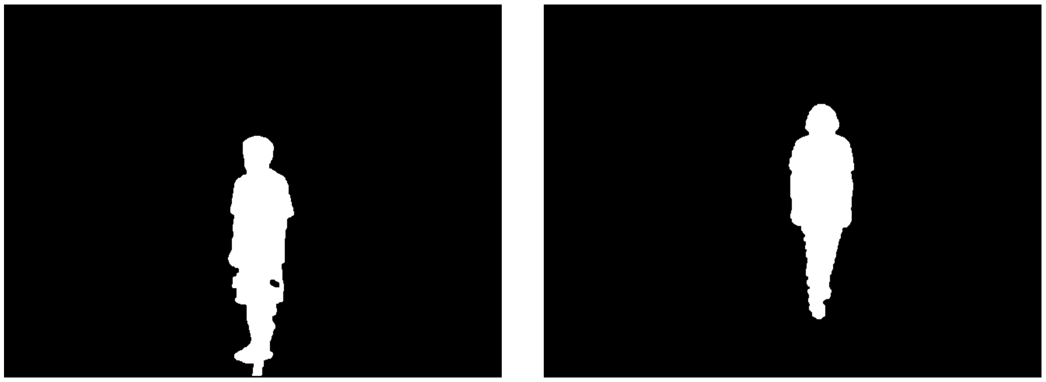
(a) Comparison with Groundtruth



(b) Localization Error

Figure 2.20: 3D Localization Experiment Results - After Bundle Adjustment

are able to produce an accurate silhouette of a moving person. Examples of the output are shown in Figure 2.21.



(a) Example 1 (b) Example 2

Figure 2.21: Background Subtraction Examples

The head location of the silhouette is then found by first taking the vertical projection of the silhouette to find the horizontal position. Then, along that position, The top edge of the silhouette is found using a simple edge detection filter. The results over time for a walking person are shown in Figure 2.22. It is this head point location which is then used at the central tracker for 3D localization and tracking.

2.4.2 Multicamera Tracking

Because data from multiple cameras is necessary for accurate 3D localization, we model tracking in VISNET as a centralized process. Each node segments the person and locates the head in the resulting foreground mask. This location is then reported to

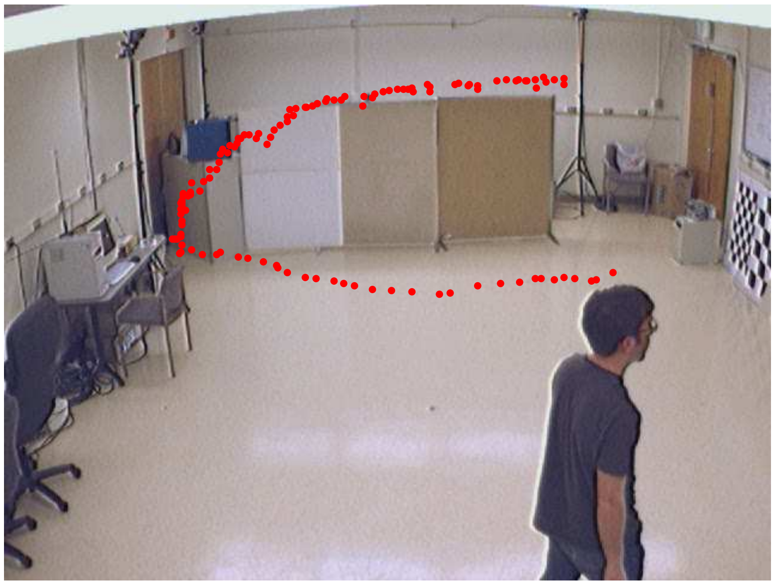


Figure 2.22: Head Location Over Time

the tracker via an OSC message, structured as shown in Table 2.2. The nodal operation is summarized in Algorithm 3.

At the tracker, incoming data from the reporting nodes is processed and assembled into the current 3D position estimate. The communications server, running in a background process, parses the incoming messages and places them into a current data queue for the localization process. The localization process first examines the timestamp of each node's data and discards old data. We currently use half the frame duration for this threshold. Timely data is then used to localize the point in the world's 3D coordinate system, as detailed in Appendix B. The tracking server process is detailed in Algorithm 4.

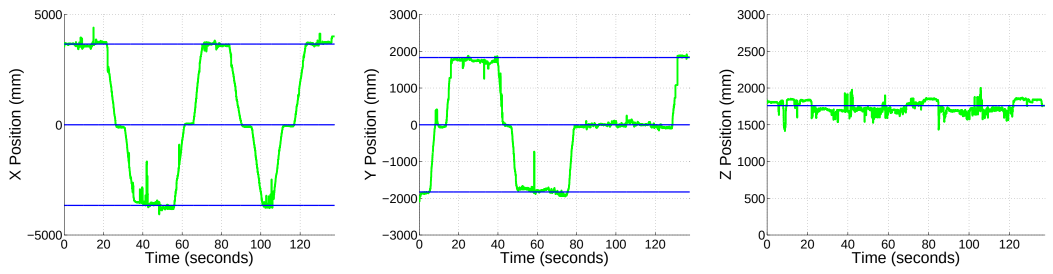
```
Input: Video Frames
Output: Local 2D Position
while $tracking == true$ do
    Capture_ Frame()
    $detected$ = Find_Head()
    if $detected$ then
        Send_Position()
    end
end
```

Algorithm 3: Nodal Tracking Process

```
Input: 2D Position Data
Output: 3D Position Estimate
while $tracking == true$ do
    Get_Data_Queue()
    Sort_Queue()
    if $NumSensingCams \geq 2$ then
        $X_{3D}$ = Localize 3D()
    end
    Wait()
end
```

Algorithm 4: Central Tracking Process

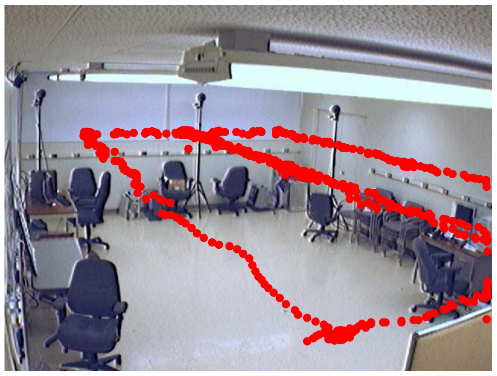
Figure 2.23 shows an example of raw output from the 3D localization process. In this experiment, the subject walked in a rectangle which spanned ± 3660 mm (± 12 feet) in the x direction and ± 1830 mm (± 6 feet) with height of 1760mm (5 feet 9 inches) as well as across the x-axis. These values are marked in the figure. Along the path, the subject paused briefly at all 4 corners of the rectangle as well as the halfway points of the sides. Figure 2.24 shows the view of nodes 1 and 5 from this experiment. In times of non-detection, no data is reported.



(a) X Coordinate (b) Y Coordinate (c) Z Coordinate

Figure 2.23: 3D Localization Raw Output

Figure 2.25 shows the path after smoothing by a Kalman filter. We see that the path is cleaned up considerably and gives a very accurate reconstruction of the path. The path is shown in 3D in Figure 2.26, with the groundtruth path indicated on the floor.


(a) Node 1 View

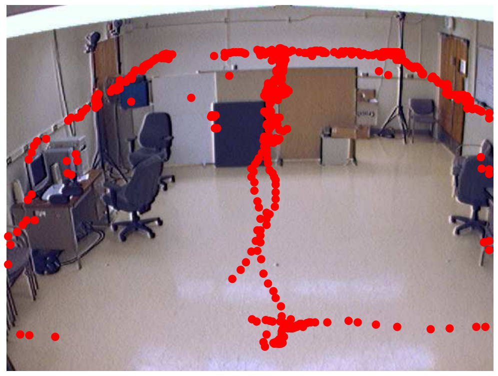
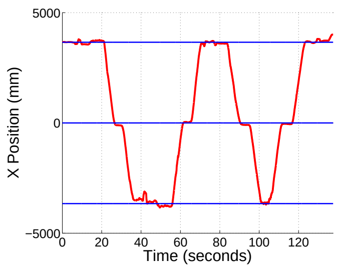
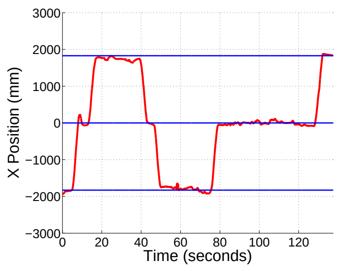

(b) Node 5 View

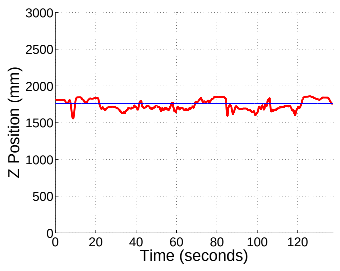
Figure 2.24: Individual Node Views



(a) X Coordinate



(b) Y Coordinate



(c) Z Coordinate

Figure 2.25: 3D Localization Smoothed Output

2.5 Summary

In this chapter we have discussed design considerations for small-scale vision network testbeds. We have presented UCSB's VISNET system as an example of a low-cost

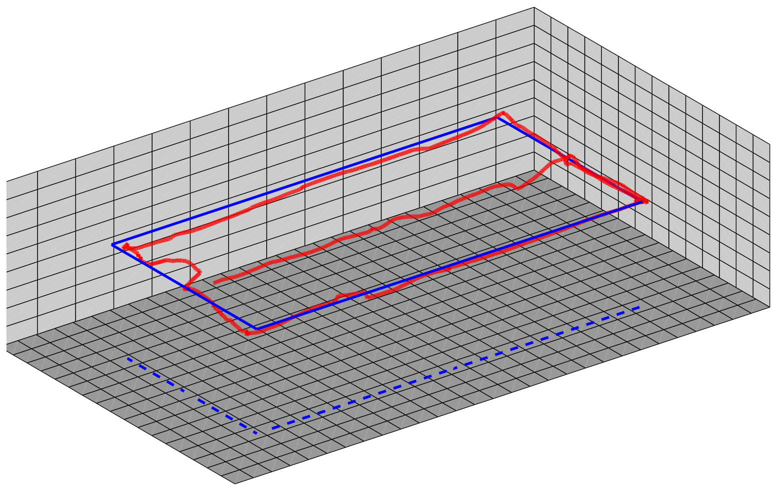


Figure 2.26: Smoothed 3D Location Output

easy to use testbed for vision research. We detailed the different software modules and their development using freely available open source tools.

We then presented two applications implemented in the VISNET system. First, we presented a simple procedure for calibrating distributed camera networks. The results showed that the localization error resulting from this method is comparable to the error resulting from running global bundle adjustment on the system.

We then detailed a simple single person tracking application implemented in the VISNET system. The person is segmented using standard background subtraction and the head located in the resulting foreground mask. The nodes' location estimates are then communicated via the network to the tracking node which filters them by their timestamps. The “good” data is then used for the current 3D location estimate. A sample tracking experiment was performed to demonstrate the functionality of the system.

While small in scale and simple in complexity, it is our hope that this description of the VISNET system may serve as a guide to others in the position of designing and developing a similar system, saving development time.

Chapter 3

Sensor Selection for Localization in Camera Sensor Networks

3.1 Introduction

With large-scale sensor networks, it is typically the case that a single event or object is sensed by only a subset of the entire network. In a dense network, however, even this sensing subset of the network may provide redundant information about the event or object. In the interest of power, bandwidth, or other concerns, we may decide to task only a few of the sensing nodes with a certain application, thus reducing the number of active nodes within a node cluster. This process is commonly referred to as *Sensor Tasking* or *Sensor Selection*.

Our motivation for minimizing the number of active nodes is twofold. First, we aim to minimize the use of resources. Having all nodes actively sensing and communicating at all times is much more expensive in terms of power than having only a subset active at any one time. This has the additional effect of reducing the amount of (likely useless) data being transmitted through the network. Second, we will reduce redundancy. While the very nature of a sensor network provides redundant data, it is not always necessary or even useful. We prefer to exert more control over which data is collected.

3.1.1 Objective

Out objective is a sensor selection process for localization and tracking in a camera network. This process will assign a score to each node based on its utility for tracking a given point. The score will be derived from geometric properties of the network and the tracked point.

3.1.2 Assumptions

Our first assumption is that we are dealing with stationary camera sensor nodes. This constrains the problem as we are unable to reposition the node (as with mobile sensors) or the camera (as with Pan/Tilt/Zoom cameras).

Second, we assume that we are working with a calibrated system. With a calibrated system, we are able to localize and track people and objects within the system in three dimensions.

Third, we assume that the tracking problem has been addressed and provides object/person state information as well as reliable state prediction information. Thus we have available tracked object features, current state information, and state prediction information.

3.2 Related Work

The problem of sensor selection in a sensor network can be thought of as the companion problem to the optimal sensor placement problem. In order to present a complete background, we discuss work in both of these areas.

3.2.1 Optimal Sensor Placement

When deploying a sensor network, we try to anticipate the activity in the covered area and position the sensors such that some measure of coverage is maximized. This measure is often application-driven and thus takes many forms.

The classic problem is that of simple visual coverage. This problem is epitomized in the Art Gallery Problem [63]. The goal of the Art Gallery Problem is to find the optimal coverage of a polygonal art gallery using a minimal number of guards. These guards are assumed to have omnidirectional vision with infinite viewing range.

While a good starting point for visual surveillance systems, the Art Gallery problem unfortunately suffers from its general nature. Real-world implementations typically involve directional sensors with finite sensing range. the method described in [45] projects the problem into 2D in the ground plane and uses linear programming to determine optimal coverage of a given space using a set number of sensors. A genetic algorithm approach is taken in [13] for the sensor selection problem in the realm of traffic monitoring.

A a simple coverage rate measure is presented in [71] for the purpose of finding the optimal coverage of a given polygon. This method also takes into account node direction, aiming to achieve coverage of the area from multiple directions. The algorithm works iteratively, optimizing the coverage rate at each step.

3.2.2 Optimal Sensor Selection

The companion to the sensor placement problem is the sensor selection problem. Here, the goal is to select some number of already-deployed sensors to perform a desired task, typically tracking.

[84] presents an algorithm which selects a fixed number of camera nodes for tracking objects in a plane. It accomplishes this by minimizing the visual hull of objects in

the cameras' fields of view. The algorithm is incremental in that it can be set to change only a small number (in their case, one) of node assignments per given time period.

[32] presents a similar approach, but models the situation as a single moving object surrounded by both static and dynamic occlusions. Their method minimizes the mean square error of the estimate of the object position based on predictions of occlusion of the object by the occluders. [64] presents a look-up table approach to sensor selection in a camera network. Each node's field of view is quantized into smaller volumes. The fields of view are distributed to the other networks nodes and their overlap computed. Using this information, each node then assembles a look-up table which is utilized for quick selection of the most favorable camera for a specified point.

[71] presents a quality-of-view metric for sensor selection in a camera network. This metric is calculated using the camera's distance from a tracked person as well as the camera's azimuth and zenith with respect to the person's position and pose. [43] scores cameras' views of tracked people based on head orientation with respect to the camera and the head size, which is a measure of distance to the camera. The goal of this work is to collect frontal face images for identification.

[37] presents a process for best node selection within node clusters. The network is divided into fixed node clusters which elect a cluster manager which is in charge of both tracking a detected person as well as the next assignment of the manager position. The selection is calculated using tracked blob geometry, tracked blob movement direction

(away from or toward camera center), and face detection results. The selection process is performed at a rate slower than the framerate of the nodes in order to produce more coherent video information to the user.

[60] uses a quality of service measurement which is simply the estimated size of the object at the current time step. This translates to assigning the nearest sensing camera to track each object. [62] uses pre-collected head models to choose the best view in a close-range multicamera network. Using 3 models of the face, side of the head, and back of the head, they compare color histogram information and choose the view which yields the best match for the face.

3.2.3 Mobile Sensors

The use of mobile sensors blurs the line between the two aforementioned problems. As such a network is more adaptable, its use alleviates the shortcomings of both by allowing gradual system reconfiguration in reaction to perceived activity.

[58] uses camera-equipped mobile sensors to monitor other mobile objects. The sensors use a score based on object distance as well as viewing angle to determine the best sensor position. [29] determines the optimal movement and placement of a network of mobile sensors based upon known event distributions.

3.3 Sensor Selection Factors

Many of the factors affecting a node's view of a point are dependent solely on the node's position relative to the point in question. As mentioned previously, we assume that we are working with a calibrated system and a tracker which provides reliable object state information. Here, we present a discussion of the most important factors affecting the localization and tracking of a point in a camera network.

3.3.1 Point Range

As with human vision, computer vision is typically improved when higher resolution images are available. With a fixed-lens, stationary camera, the only degree of freedom affecting image resolution is the distance from the camera to the object being imaged. It is obvious that we will normally prefer the subject to be larger in the frame, and thus nearer the camera. We will use 3D position information available to us and rank the resolution of an object via its distance from the camera. This distance is illustrated in Figure 3.1 and calculated by:

$$D_j^i = \|C_i - P_j\| \tag{3.1}$$

where C_i is the center point of camera i and P_j is the location of object j. This calculation can be performed in either the world coordinate system or the camera-centric 3D coordinate system.

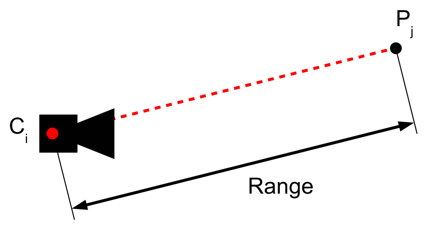


Figure 3.1: Point Range Definition

3.3.2 Point Visibility

The next factor we consider in the node selection process is simply whether or not a point is within the sensing range of a node. We wish to answer the question "Given otherwise ideal conditions, can this node sense that point?" With known camera parameters, it is easy to project a point into the image plane of each camera to determine this. In addition to determining whether a point lies within the camera's field of view, we can define a maximum range, d_{max}, beyond which we assume that a point is not visible or at least not useful to the node. This is illustrated in Figure 3.2.

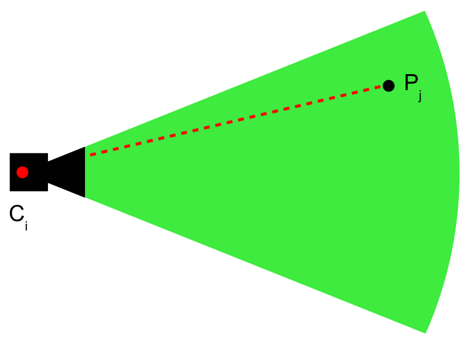


Figure 3.2: Point Visibility Definition

Based upon the potential visibility of a point in a specific camera, we define our visibility of a point P_j in camera i as:

$$V_j^i = \begin{cases} 1 & (0 \leq u_j < width_i) \bigcap (0 \leq v_j < height_i) \bigcap (Range_{ij} < d_{max}) \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where $[u_j \; v_j]^T$ indicates the projection of the point into camera and $width_i$ and $height_i$ indicate the image width and height of camera i.

3.3.3 Occlusion

Along the lines of a point being within a sensor's field of view is whether or not the sensor can actually *see* the point. A point located within the sensor's field of view may still be occluded by either another tracked person or a stationary object such as a sign or a partial wall. With prior knowledge about a scene culled from normal system operation up to the current time, it is fairly straightforward to predict occlusion for the current frame. Figure 3.3 illustrates an example of point occlusion. We define our occlusion measure of a point P_j in camera i as:

$$O_j^i = \begin{cases} 0 & \text{point } j \text{ is occluded in camera } i \\ 1 & \text{otherwise} \end{cases} \tag{3.3}$$

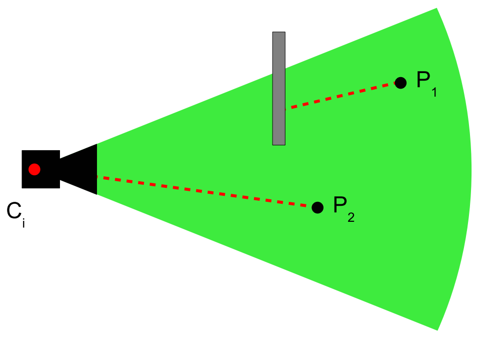
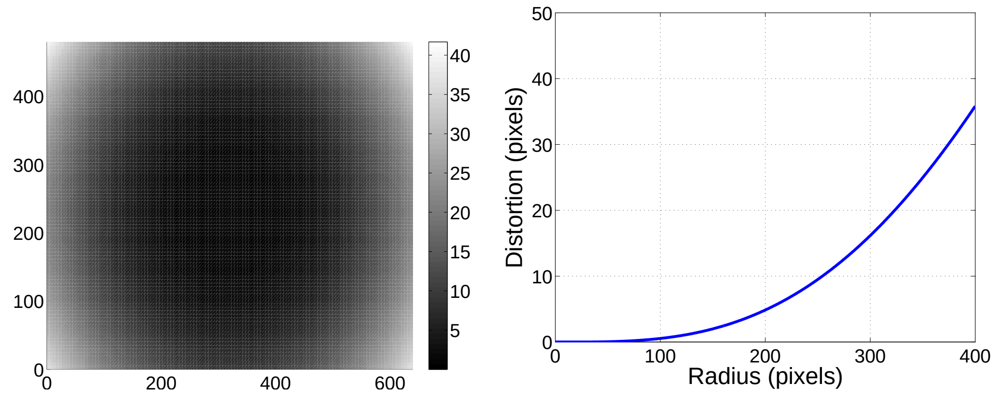


Figure 3.3: Point Occlusion Example

3.3.4 Frame Position

Because of distortion introduced by the camera lens, points near the edge of an image can experience significant change in their position with respect to their ideal position. The typical lens distortion model [86], [46], [16], [42], introduces both radial and tangential distortion. This distortion is detailed in Appendix A. Figure 3.4 shows lens distortion for a typical off-the-shelf web camera.



(a) Lens Distortion vs. Image Location (b) Lens Distortion vs. Radius

Figure 3.4: Lens Distortion Detail

By examining the distortion model in Appendix A, we see that it is dependent on a point's position with respect to the camera center. Thus we use the radius as an indicator of potential distortion:

$$R^i_j = ((u_j - c_x)^2 + (v_j - c_y)^2)^{\frac{1}{2}} \tag{3.4}$$

where $[u_j \ v_j]^T$ is the location of point P_j in the camera's image plane and $[c_x \ c_y]^T$ represents the camera principal point, as defined in Appendix A.

3.3.5 Relative View Angle

When two (or more) nodes are required for an application, their relative positions with respect to the point or object of interest may affect their utility for the application. Besides the previously defined factors, we look at the nodes' relative view angle. Assuming that both nodes i and k can see the point (otherwise we wouldn't be considering them for any application involving the point), we define their relative view angle θ_j^{ik} with respect to a point P_j as:

$$\theta_j^{ik} = cos^{-1}\left(\frac{(C_i - P_j)\cdot(C_k - P_j)}{\|C_i - P_j\|\|C_k - P_j\|}\right) \tag{3.5}$$

where C_i is the center of camera i and C_k is the center of camera k. This angle is illustrated in Figure 3.5.

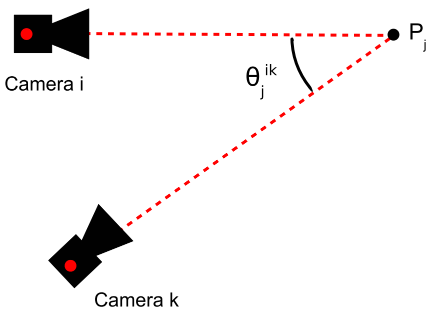


Figure 3.5: Relative View Angle Definition

3.4 Sensor Selection Scoring for Point Localization

As discussed previously, different applications have different criteria for what constitutes a ”best” view of a person, point, or object. It is unlikely that any single factor discussed here will serve any application perfectly. Rather, customized scores can be assembled after exploring how each factor might affect an application.

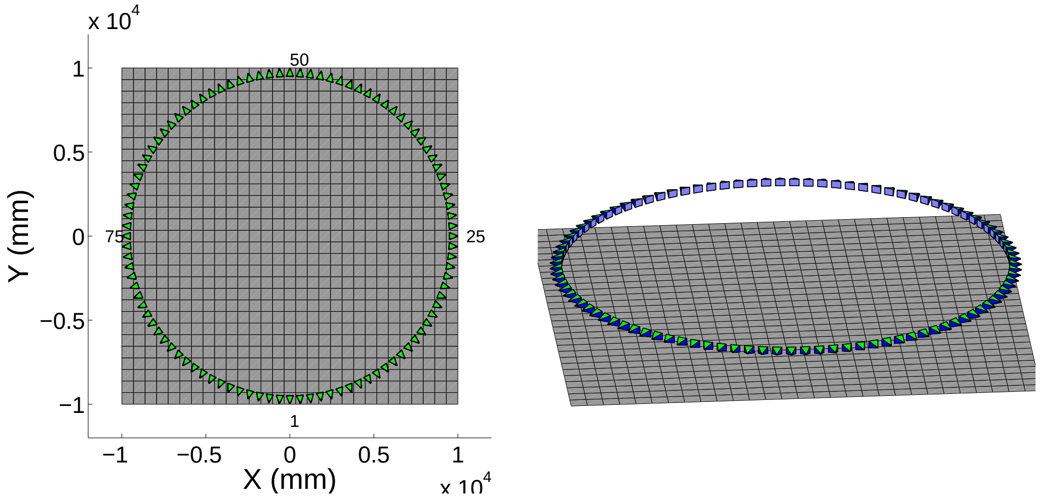
Here, we present a sensor selection score for the application of point localization in a sensor network. The process of point localization is discussed in Appendix B. Our score takes into account how the previously discussed selection factors affect the localization output.

3.4.1 Number of Sensing Nodes

It is natural to try and apply the *more is better* rule when working with sensing and especially cameras. However, as discussed earlier we have the conflicting requirement of minimizing the number of sensing nodes assigned to each tracking task. Thus we seek to maximize the accuracy of our process while still minimizing the number of nodes involved.

To illustrate the effect of the number of sensing nodes on localization error, we set up a simulation in Matlab. The simulation consists of 100 cameras placed uniformly around a circular space of radius 10 meters and shown in Figure 3.6. The intrinsic parameters are based on actual parameters of the cameras in VISNET, with a 640 by 480 image resolution and a centered principal point. The focal lengths are set to 800 and the distortion parameters are taken from one of the VISNET nodes.

The center point of the circle was projected into each camera with additive zero-mean Gaussian noise with variance 5 pixels. For each $N \in [1, 20]$ a random grouping of N cameras was chosen and the point $[0\ 0\ 0]^T$ was localized and its error calculated. This grouping and localization was performed 1000 times for each N.



(a) Overhead View (b) Orthographic View

Figure 3.6: Matlab Camera Network Simulation Layout

Figure 3.7 shows the results of the simulation. We see a definite descrease in the the mean error as more nodes are used for localization. Figure 3.8 shows the error distribution for $N = 2$, $N = 3$, $N = 4$, and $N = 5$. For the two camera case, we see that while the majority of the error is centered around 100mm, the upper tail of the distribution extends to the maximum of nearly 20 meters, drastically increasing the mean.

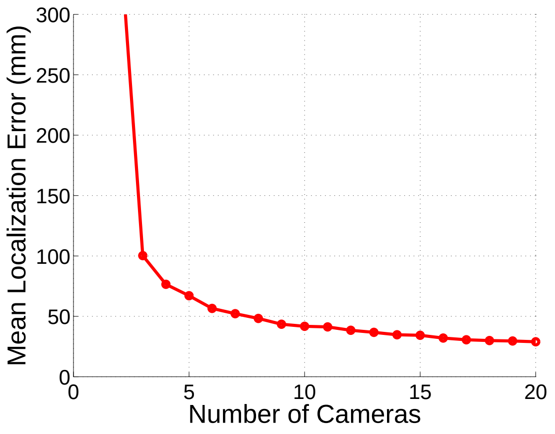


Figure 3.7: 3D Localization Error vs. Number of Cameras

This experiment was repeated using the VISNET system with the results shown in Figure 3.9. We note that the minimum error is influenced by the fact that as N increases, we have available fewer and fewer groupings of N cameras until $N = 10$ where we have just one grouping. Still, we see a similar trend for both the error versus the number of cameras and the error distribution for values of N.

Based upon the resulting error histograms, we see that even with two nodes we can achieve minimal localization error by careful choice of the nodes. Thus we focus on efforts on answering the question "which two nodes are the best for localizing a given point?"

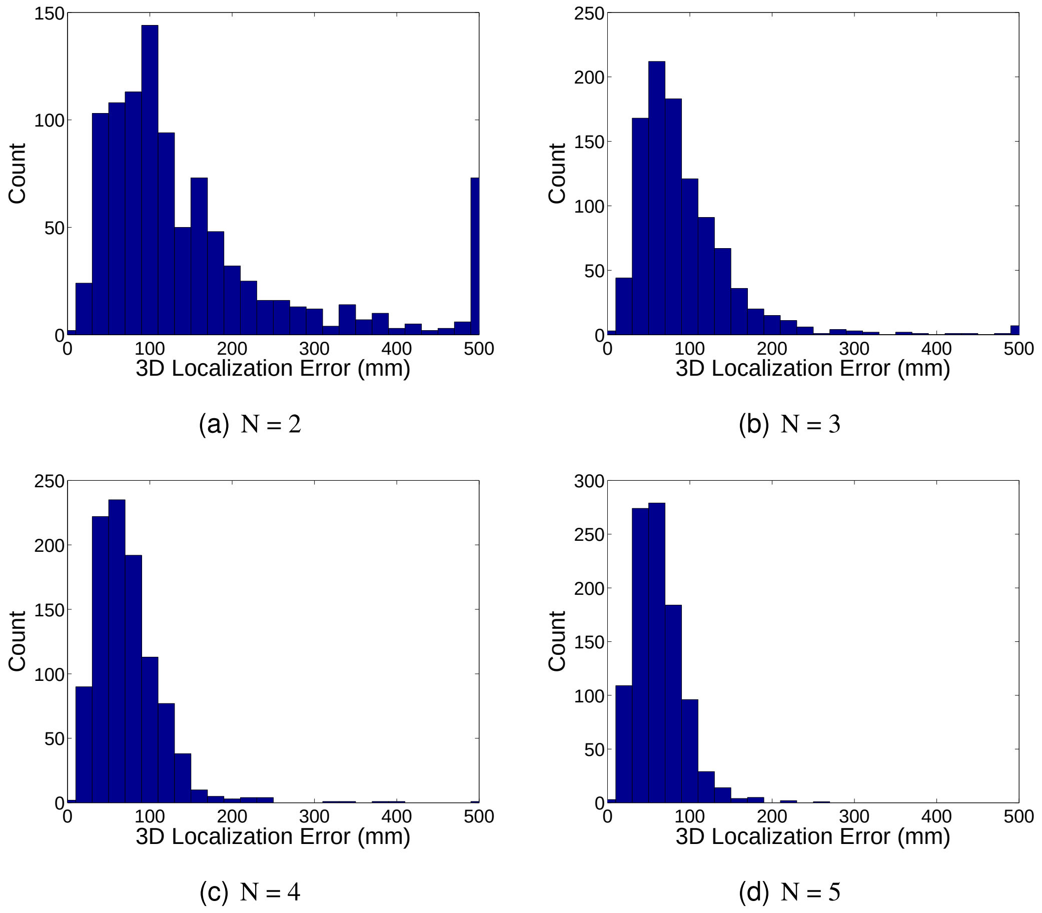


Figure 3.8: 3D Localization Error Distributions

3.4.2 Visibility and Occlusion Scores

The visibility of a point in a sensor is an obvious requirement for any application involving both. We define our visibility scores of a point P_j in camera i as:

$$\Psi_j^{i,V} = V_j^i \tag{3.6}$$

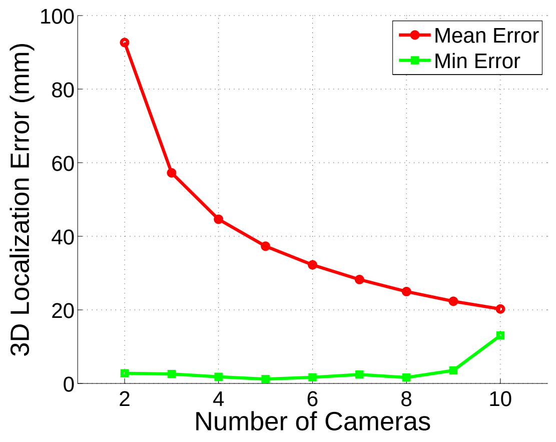


Figure 3.9: VISNET 3D Localization Error vs. Number of Cameras

and

$$\Psi_j^{i,O} = O_j^i \tag{3.7}$$

3.4.3 Point Range Score

Because 3D localization involves projecting a line from each camera's center through its local image projection of the 3D point, the process is susceptible to image noise. This image noise is "projected" outward into space, as illustrated by Figure 3.10. Thus we see that even with constant image noise, a point further away from the camera is more likely to experience localization error.

To demonstrate the effect of point range on localization error, a camera network simulation was built where camera location with respect to a point was varied. The

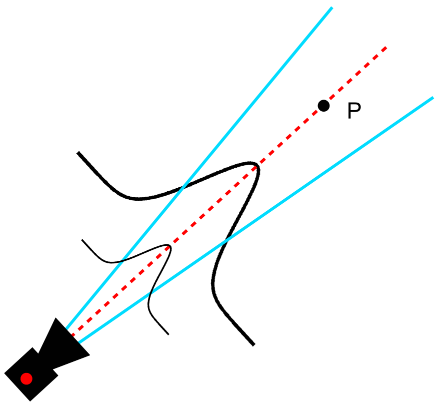


Figure 3.10: Projection of Image Plane Uncertainty

point was projected noisily into two cameras and then localized using the image positions. Figure 3.11 shows the localization error plotted versus point range.

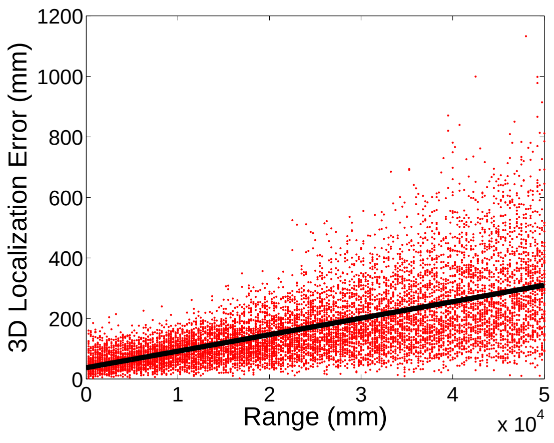


Figure 3.11: 3D Localization Error vs. Point Range

We see that localization error increases linearly with range from the sensing nodes. Based upon this relationship, we define our point range score as:

$$\Psi_j^{i,D} = 1 - \frac{D_j^i}{4 \cdot d_{max}} \tag{3.8}$$

where d_{max} is the maximum desired range, which is usually equal to the maximum effective range of the camera. The range score is shown in Figure 3.12.

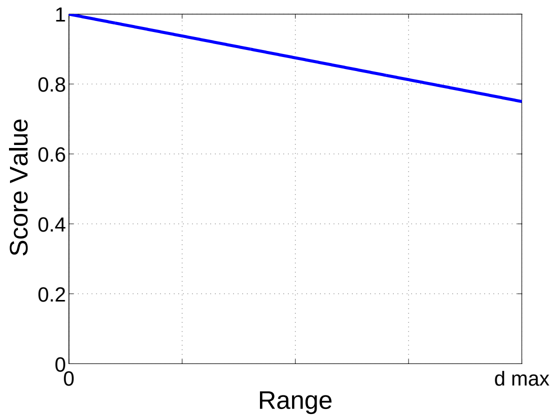
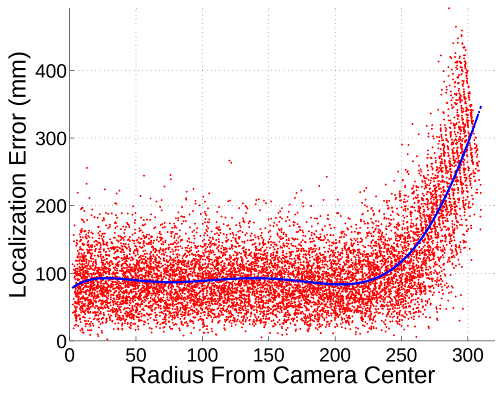


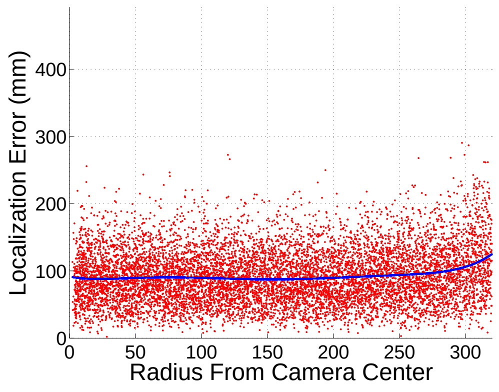
Figure 3.12: Point Range Score

3.4.4 Frame Position Score

It is easy to see that the use of points near the camera edge could introduce significant distortion into the localization process. However, by undisorting 2D object points prior to the localization process (as detailed in Appendix B), we can significantly reduce the localization error. Figure 3.13 shows the localization error for a pair of cameras as the point location is varied from the center (experiencing minimal lens distortion) to the edge (experiencing maximum lens distortion). Figure 3.13(a) shows the error without point undistortion and Figure 3.13(b) shows the error after first undistorting the 2D points.



(a) Error Without Distortion Correction



(b) Error With Distortion Correction

Figure 3.13: 3D Localization Error vs. Radius

We see that with proper correction of the lens distortion, only points on the extreme edges of the image contribute additional error to the localization process, and even then the error is minimal. Thus we define a point's confidence with respect to its position in the frame as:

$$\Psi_j^{i,R} = 1 - \left(\frac{R_j^i}{2 \cdot R_{max}} \right)^4 \tag{3.9}$$

The image position (radius) score is shown in Figure 3.14.

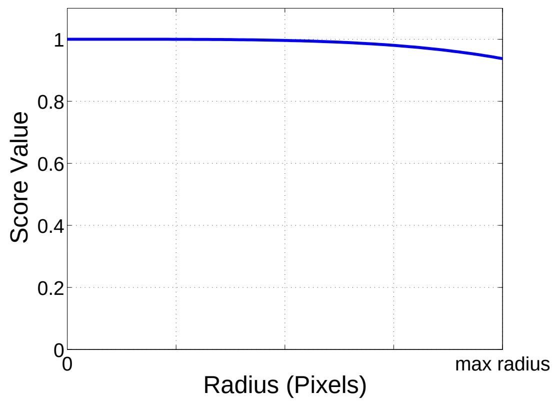
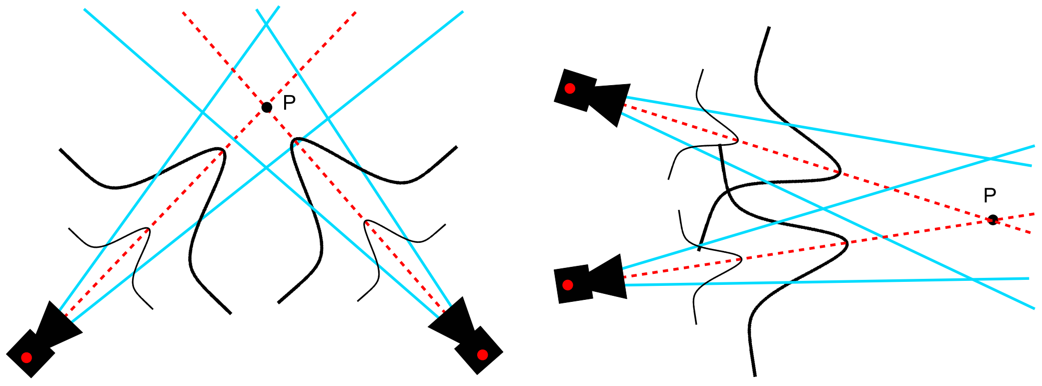


Figure 3.14: Point Radius Score

3.4.5 Relative View Angle Score

As described in Appendix B, the process of point localization with two cameras involves finding the intersection (or closest point thereto) of two lines in space. These two lines are constructed using the node-level point position information and thus is subject to imaging, segmentation, and localization noise. Assuming this noise to be stationary, we examine the effect of the relative view angle of the sensing cameras on the localization error.

If we model the image 2D localization error as an isotropic Gaussian random variable, the lines drawn out into space for 3D localization are characterized by a "cone" of uncertainty. This concept is illustrated in two dimensions in Figure 3.15(a). When computing the intersection of two similarly-constructed lines, as shown in the figure, we are presented with an intersection of the lines' uncertainty. In this scenario, a change (due to noise) in the image position will have an effect on the estimate of the point P. Figure 3.15(b) shows another scenario where the cameras are situated more closely together and thus even small changes in their image position estimates will result in larger changes in the estimate for P. This error is often encountered when using short baseline stereo camera setups.

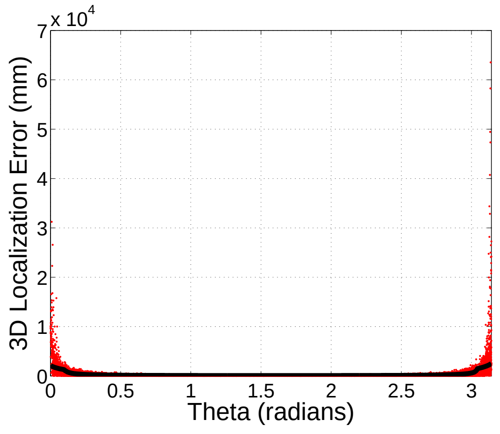


(a) 2 Cameras With Large Relative View Angle (b) 2 Cameras with Small Relative Angle

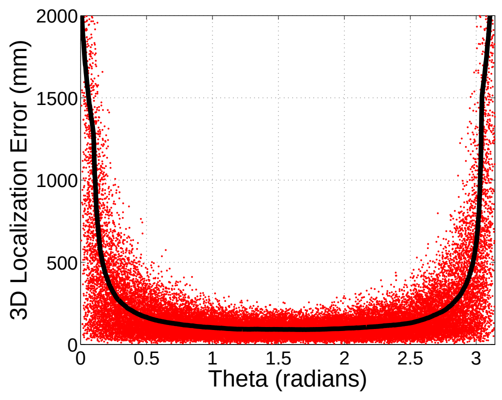
Figure 3.15: Camera Localization Uncertainty

In order to examine the error produced by the relative camera positions, we simulated a two-camera network with varying relative view angle between the nodes and the point to be localized. The relative view angle is defined in Equation 3.5.

Figure 3.16 shows the results of the simulation. The localization error reaches a minimum at $\frac{\pi}{2}$ and maxima near both 0 and π. Figure 3.17 shows the results for a similar experiment using the VISNET system. While the use of static cameras in the system reduces the variation in θ_j^{ik}, we see a similar trend in the localization error, shown in Figure 3.17.



(a) Error vs. Relative View Angle



(b) Error Detail

Figure 3.16: 3D Localization Error vs. Relative View Angle

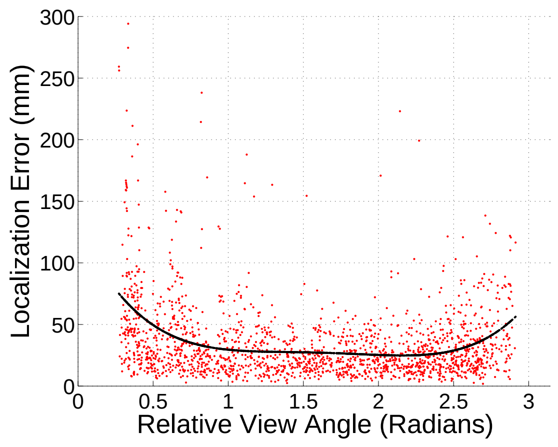


Figure 3.17: VISNET Localization Error vs. Relative View Angle

Based on the error trends observed in both the simulation and the VISNET system, we define the relative view angle scoring for a specific pair of cameras viewing a point as:

$$\Psi_j^{ik,\theta} = 1 - \left(\frac{2}{\pi}|\theta_j^{ik} - \frac{\pi}{2}|\right)^{14} \tag{3.10}$$

Following the error trend seen in Figure 3.16, the score highly penalizes relative angles close to $\frac{\pi}{2}$, as seen in Figure 3.18.

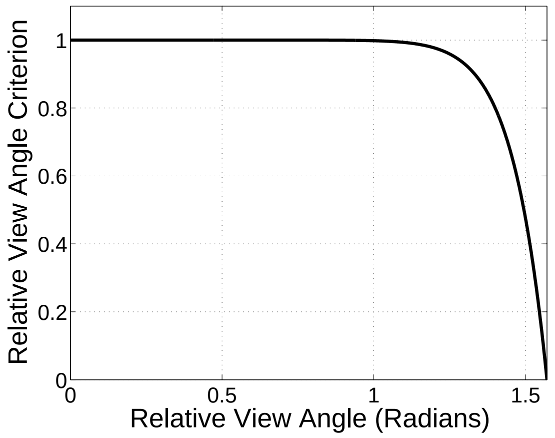


Figure 3.18: Two Camera Relative View Angle Score. Angle indicates deviation from $\frac{\pi}{2}$

3.4.6 Combined Sensor Selection Score

We now present the combined selection score for nodes to be used in point localization. First, we look at the scoring of single nodes based upon their Visibility, Occlusion, Range, and Radius scores. We define this combined score as:

$$\Psi_j^{i,L} = \Psi_j^{i,V} \Psi_j^{i,O} \left(w_R \Psi_j^{i,R} + w_D \Psi_j^{i,D} \right) \tag{3.11}$$

where $w_R + w_D = 1$.

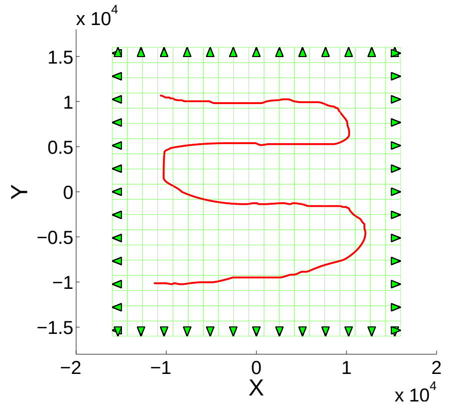
As they are binary, $\Psi_j^{i,V}$ and $\Psi_j^{i,O}$ are multiplied, causing the resulting score to be zero if either is zero. They are multiplied by a weighted sum of $\Psi_j^{i,R}$ and $\Psi_j^{i,D}$. As the range of a point has more effect on localization error than its radius, we weight $\Psi_j^{i,D}$ higher. Through experimentation, we found that values of $w_R = 0.4$ and $w_D = 0.6$.

For a pair of cameras (i and k), we now have the scores $\Psi_j^{i,L}$ and $\Psi_j^{k,L}$, respectively, as well as their relative view angle score, $\Psi_j^{ik,\theta}$. In order to arrive at a single score for the pair as well as to incorporate the relative view angle into our score, we calculate our pairwise localization score as:

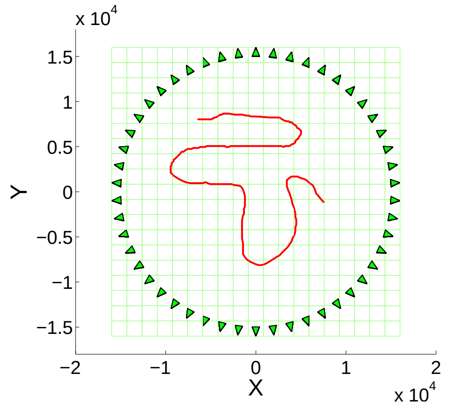
$$\Psi_j^{ik,Pair} = \frac{w_u}{2} (\Psi_j^{i,L} + \Psi_j^{k,L}) + w_b \Psi_j^{ik,\theta} \tag{3.12}$$

where w_u is the unary score weight, w_b is the binary score weight, and $w_u + w_b = 1$. Experimentation led us to set $w_b = 0.6$ and $w_u = 0.4$.

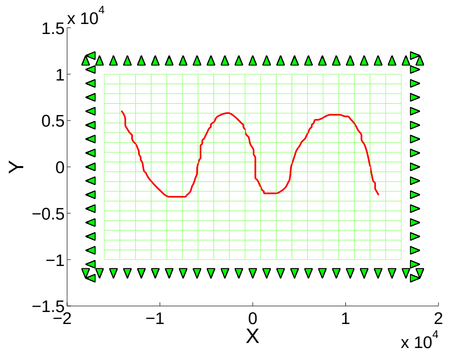
In order to test our selection score, we ran simulations in Matlab using 3 different network configurations. Each simulation consisted of a multicamera network and a single object moving through the network. The network layouts are shown in Figure 3.19.



(a) Square Layout - 52 Nodes



(b) Circular Layout - 50 Nodes



(c) Rectangular Layout - 84 Nodes

Figure 3.19: Point Localization Simulation Layouts

At each point along the object path, we localized the point using 6 different sensor selection methods, detailed below:

- **All Cameras**: All sensing nodes were used for localization
- **Random 2**: Two sensing nodes chosen at random were used for localization
- **Random 3**: Three sensing nodes chosen at random were used for localization

- **Distance 2**: The two nearest nodes were used for localization
- **Distance 3**: The three nearest nodes were used for localization
- Ψ^{Pair}: Two nodes chosen using the selection score defined in Equation 3.12 were used for localization

Table 3.1 details the results from these experiments. We see that using our localization selection score, we obtain improvement over random selection of 2 and 3 random nodes as well as over choosing the 2 or 3 nearest nodes. In one case, our method actually performed better, on average, than using all sensing cameras. In addition to the mean error, we were able to significantly reduce the error variance over random selection and nearest node selection.

We note here that our selection process outperformed using all sensing cameras in the circular simulation. Intuition leads us to attribute this to the nature of the circular layout - where the best relative view angle of a sensing group is typically less than ideal. One future direction for this work would be to explore the effect of camera layout on the localization error bounds.

The localization node selection experiment was then duplicated using the VISNET system, utilizing a set of 90 ground truth points in the room. The points were collected by moving a tripod with colored points around the room to specified positions, as shown

	Square		Circular		Rectangular	
Method	μ_{Error}	σ_{Error}	μ_{Error}	σ_{Error}	μ_{Error}	σ_{Error}
All Cameras	72	34	238	114	55	27
Random 2	408	485	426	622	492	606
Random 3	256	257	278	168	298	293
Nearest 2	305	252	345	295	344	313
Nearest 3	219	209	193	169	250	163
Ψ^{Pair}	99	65	135	74	141	139

Table 3.1: Simulated Localization Results

in Figure 2.16. The points were then hand-selected in each video frame at each node in order to reduce the amount of imaging and 2D localization noise.

Using our selection process, we localized the points in the room using the two best scoring nodes. As with the simulation, our selection score outperformed the other localization methods, including using all cameras. However, the error variance was higher than when utilizing all cameras. The results are shown in Table 3.2.

Method	μ_{Error}	σ_{Error}
All Cameras	174	134
Random 2	224	229
Random 3	184	159
Nearest 2	153	200
Nearest 3	161	153
Ψ^{Pair}	107	148

Table 3.2: VISNET Selection Experiment Results

3.5 Sensor Assignment Continuity

The sensor selection approach discussed so far has proven useful in reducing the number of active nodes at any single point in time. However, its momentary nature makes it susceptible to assignment fluctuations in cases where nodes' scores are very close to each other. For efficient system operation, we plan to minimize the number of camera transitions, or *handoffs* and thus introduce the idea of sensor assignment *inertia*.

3.5.1 Sensor Assignment Inertia

Inertia is defined as a resistance to change in state of motion. Thus our idea of node assignment inertia describes our desire to retain a node assignment for as long as possible, while still keeping with our goal of minimizing localization error. To accomplish this, we introduce an *Inertia* score for each node pair, defined as:

$$\Psi_j^{ik,Inertia} = \begin{cases} 1.0 & \text{Both node } i \text{ and node } k \text{ are currently assigned} \\ 0.95 & \text{Either node } i \text{ or node } k \text{ (but not both) is currently assigned} \\ 0.9 & \text{otherwise} \end{cases} \tag{3.13}$$

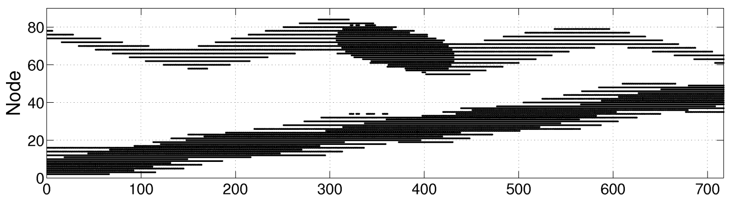
which rewards assignments where one or both sensors is already assigned to the localization task. This score is applied to the pairwise localization score, defined in Equation 3.12, resulting in a total selection score of:

$$\Psi_j^{ik,Total} = \Psi_j^{ik,Inertia} \Psi_j^{ik,Pair} \tag{3.14}$$

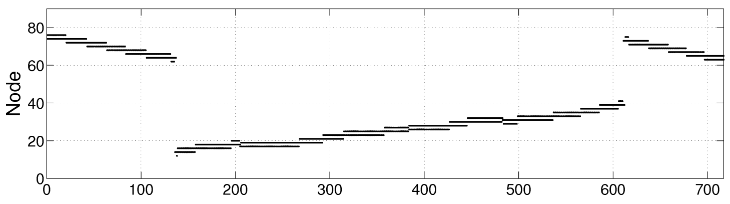
3.5.2 Continuity Demonstration

In order to demonstrate our selection process, we used the rectangular camera network simulation shown in Figure 3.19(c). The point path was tracked and node assignments were made based on path prediction information provided by the tracker. Figure 3.20(a) shows the node assignments over time when all sensing nodes were used for localization. We see that there is significant activity in the network with this method. Figure 3.20(b) shows the node activity when the two nearest nodes are used, and Figure 3.20(c) shows the activity when the 3 nearest nodes are used for localization. Again, we see that a significant number of nodes are activated over the course of the tracking.

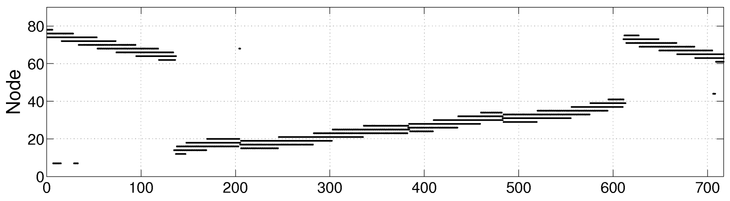
Figure 3.20(d) shows the results from our combination selection score defined in Equation 3.12. Again, we see significant node activity during the tracking process. Finally, using the inertial node score defined in Equation 3.14, we arrived at the results shown in Figure 3.20(e). We see a significant reduction in both the number of active



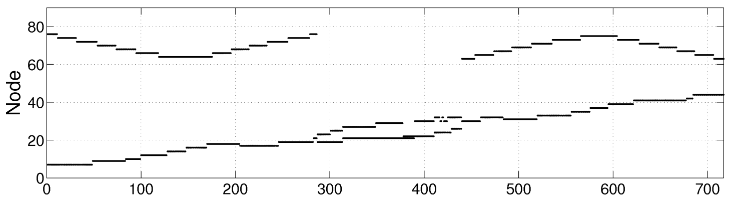
(a) All Sensing Nodes



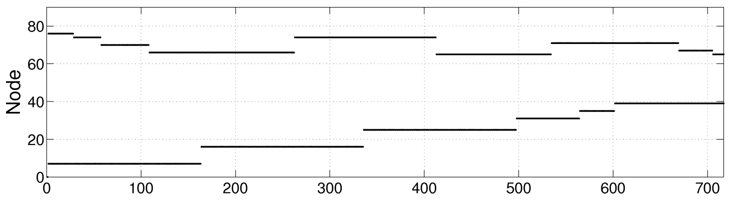
(b) 2 Nearest Nodes



(c) 3 Nearest Nodes



(d) Best Score Pair



(e) Best Score Pair With Inertia Score

Figure 3.20: Sensor Assignment Output

nodes and the number of activity transitions. Examining the error, we see only a small increase in the localization error as compared to the non-inertial optimal scoring.

Table 3.3 summarizes the results from this experiment. We see that our selection process significantly reduces the number of nodes utilized as well as the number of node handoffs performed during the tracking process. Compared to our momentary selection process, node activity is reduced by 78%. While the localization error increases slightly with the inertial assignment, it is not significant when compared to the reduction in sensor activity.

Method Method	Nodes Used	Node Transitions	μ_{Error}	σ_{Error}
All Cameras	78	90	55	27
Nearest 2	35	37	344	313
Nearest 3	44	43	250	163
$\Psi^{Combined}$	59	41	141	139
Ψ^{Total}	13	15	165	115

Table 3.3: Inertial Node Assignment Results

3.6 Summary

In this chapter, we discussed the factors affecting node utility for tracking in a camera network. These factors are:

- Point visibility

- Point range
- Point location in the sensing camera's image
- Relative view angle of two nodes with respect to the point

We then discussed the use of these factors for constructing a utility score for point localization. By examining the effect each factor has on point localization, we are able to construct an overall node scoring system aimed at minimizing 3D localization error.

We then demonstrated our localization score both in simulation and with real data from the VISNET system. Using our node utility score, we were able to reduce the mean localization error as well as the error variance as compared to localization using random groupings of 2 and 3 nodes as well as groupings of the 2 and 3 nearest nodes. In some cases, we were able to reduce the error below that of when all sensing cameras were used for localization.

In the interest of continuity of sensor assignment, we introduced an inertial selection score which factors nodes which are already assigned to a tracking task. When used with our node utility score, this assignment inertia provides a significant decrease in node activity.

Chapter 4

Active Multiview Appearance Modeling in Camera Sensor Networks

4.1 Introduction

The detection, identification, location, and tracking of any interesting object or phenomenon is a common application for camera networks. The surveillance of humans, however, is by far the most popular (and, to many, the most unpopular) application of such networks. Applications of human surveillance include detection, tracking, identification, activity recognition, and left object detection.

One challenge in visual surveillance is how to correlate a person viewed in different cameras. In an ideal situation, one would know *a priori* the appearance of each person in the network, making identification a simple task. Unfortunately, this is not typically the case and each person's appearance must be learned. A companion to the appearance modeling problem is the question of how we represent a person's appearance in a compact manner.

In this chapter, we discuss the learning of appearance models of tracked humans. Specifically, we address how to rank collected data from a camera network in order to obtain the best possible representation of the person. In addition, we address the storage and transmission issues involved with this process.

4.1.1 Motivation

In order to motivate our work, we begin with the scenario of a large-scale multicamera network in a setting such as an office building. Our primary task is to track people through the network, creating a database of movement and perhaps activity. This task presents us with with the following problems:

- In a multicamera network, it is often the case that two (or more) cameras see a person, but from different viewing angles. Depending on the person's appearance (mainly, his or her clothing), the different viewing angles may not be correlated and thus view correspondence is not guaranteed.

- In a multicamera network where there are disjoint groupings of cameras, such as on different floors of a building, we need a way to connect a person's appearance and activity in multiple clusters. We seek an appearance model which can aid in correlating tracked people in the different clusters.

- In a real-world scenario, we do not have the luxury of having *a priori* appearance information about tracked people. Instead, we must learn an appearance model as the person moves through the system. Thus we require a method for appearance data collection and scoring.

4.1.2 Assumptions

In order to focus on the modeling process, we make some assumptions about the system, which are detailed below.

Calibrated Network

Our first assumption is that we are dealing with a calibrated network of camera nodes. This calibration provides knowledge about the location and pose of each camera with respect to some global reference. This knowledge allows us to localize and track moving objects or people in real-world coordinates.

Object Tracking

We assume that reliable position and velocity data for each tracked person is available. Any issues stemming from occlusion or other complex situations are assumed to be addressed by the tracker. Another option would be to suspend the modeling process during times of occlusion or other uncertainty. The localization of points in a camera network is discussed in Appendix B.

Color Calibration

We assume that we have available to us a transformation from each node into a global color space. In order to combine and/or compare tracked person color features we must be confident that the comparison is meaningful. With this comes the assumption that illumination variations are also corrected. These issues are addressed in works such as [50] and [34].

4.2 Related Work

The modeling of tracked people in video varies over a wide spectrum of complexity. A chosen model is essentially a combination of features and is constrained only by the hardware on which the process is run. Here, we briefly discuss work in the area of appearance modeling. We first discuss modeling in a single camera and then discuss modeling in multicamera systems.

4.2.1 Single Camera Object Modeling

In order to model a tracked person, we must first concern ourselves with the detection of said person. Some methods detect and combine simple features in order to detect the human shape [82]. [68] finds the edges in a video frame, detects rectangular shapes, and combines them to build articulated human models.

Another approach to human modeling is to first use motion detection to segment areas or objects of interest in video. [72] uses corner features to choose strong points to track. These points are then tracked using an optical flow approximation. Optical flow essentially tries to estimate a point's change in position from one frame to the next. When this change is greater than some threshold, the point is considered to be of interest and further analysis can be performed.

Other detection methods such as background subtraction [44], [49], [53], [67], [74], [78], [55] yield what are known as "blobs" which can be thought of as the silhouettes of moving objects or people. These blobs can then be described and tracked by any combination of their various features. The simplest is a single point typically representing either the centroid or the top of the head. Geometric information is often used to supplement the position information. The person's silhouette bounding box is used by [40] as an additional tracking feature. Similarly, [73] uses the height and width aspect ratio to model a tracked blob. Instead of the bounding box, [52] and [88] model a tracked person with an ellipse.

In addition to position and geometric features, color information is commonly used to enhance the object model. This is an obvious extension as humans often use clothing, hair, and skin color to differentiate people. [52] uses the average color of the person for tracking purposes. The use of color for identification is extended by [90] which utilizes full RGB histogram information (quantized to 8 x 8 x 8 bins).

Texture information is also used to represent tracked people and objects. [18] uses correlograms, an extension of co-occurence matrices for color images, in addition to HSV histogram information. Edge information from inside the silhouettes is used in [41] to describe the tracked people . Both the Dominant Color Descriptor and the Edge Direction Histogram (defined in the MPEG-7 standard [70]) are used in [85] to describe tracked humans for retrieval purposes.

In lieu of using raw data, it has become popular to fit generative models to visual data in order to calculate parametric representations of tracked objects. By far the most popular method uses Gaussian Mixture Models (GMMs) for these representations. Mixture models are commonly used in image retrieval [20], [39], [38], [69] but have also made their way into tracking applications. [56] uses 2D mixture models to to track objects in video. [83] uses models spatial, color, and texture information with mixture models for tracking in single camera video. [80] uses mixture models in a mean shift tracking framework.

4.2.2 Multiple Camera Person Modeling

The use of multicamera networks presents new challenges to human surveillance. Aside from the obvious problem of calibration and 3D reconstruction, we are presented with the problem of identifying a person detected in one camera as the person appearing in another. Surprisingly, there is not a lot of work which directly addresses this issue.

In order to get a more accurate appearance model of a tracked person, [60] simply takes the mean color about all viewpoints of the person. The person is segmented into top and bottom regions, preserving some structure, but no care is taken to preserve view-specific features. Inter-camera color calibration is addressed by [50] but does not address multiple-view modeling of the tracked people. Their model consists of a separate histogram for the R, G, and B channels but again does not account for view-specific variation. [33] models tracked people in a multicamera environment using R, G, and B color distributions but again, does not address multiple viewpoints.

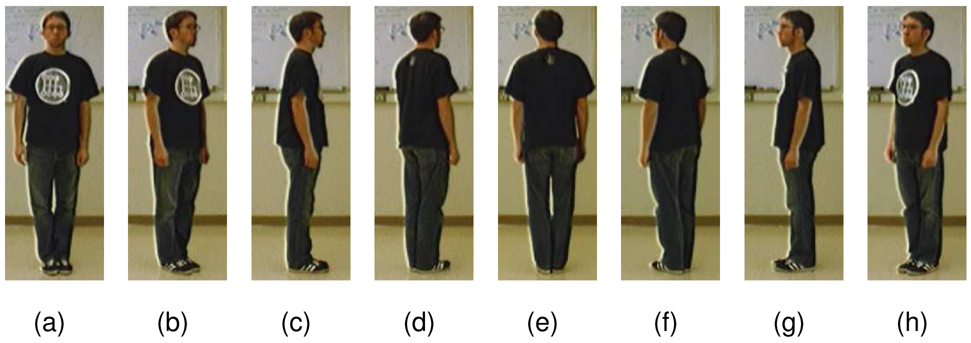


Figure 4.1: Multiview Appearance Model Definition. Views (a) through (h) represent view angles of 0 through $\frac{7\pi}{4}$

4.3 Multiple View Appearance Modeling

We are interested in an appearance model which will capture view-dependent variation in tracked humans. This is important for camera sensor networks for identification continuity throughout the network. Figure 4.1 shows an example of a person whose identification might hinge on the capturing of view-dependent appearance variations. A single, averaged color model would not provide the discriminatory power available with a multiview model. We demonstrated this power with our work in [66]. In addition, a compact model will allow efficient transmission and storage of people's appearance and activity within the network. While ideally we would have full video frames transmitted and stored, our bandwidth and power requirements force us to seek more efficient representations.

We define an 8-view model for our purposes. These views are defined in Figure 4.1. Ideally, we would have a person stand before a camera and spin so that the data could be properly captured. However, in a realistic scenario this is impossible. Rather, we are interested in the automatic collection of these views of a person as they are tracked. In addition, we are interested in acquiring the best possible view from each angle of the person. To this end, we have developed a scoring system by which a view can be ranked based upon several factors affecting its quality and usefulness for our modeling purposes.

4.4 View Quality Assessment For Appearance Modeling

When collecting visual data such as is found in surveillance networks, it is often necessary to assign a quality or utility value to the data. Because a lot of data in such systems is of little or no use, this value is used to "screen" the data so that only the most useful is retained. In the case of appearance modeling of humans, several factors affect the quality of the data. Here, we develop a view assessment score for human appearance modeling. For this process, we make the assumption that we have reliable calibration data, as discussed in Chapter 2. Additionally, we assume that we have available reliable state data about the person. Specifically, we require the person's location P and his or her velocity V. This information would be provided by a tracking process and could result from the selection and localization process presented in Chapter 3.

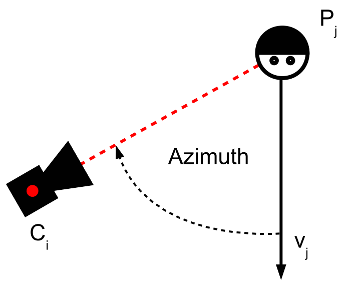


Figure 4.2: Azimuth Definition

4.4.1 Azimuth

When dealing with human surveillance, we are typically very concerned with which part of the person we are viewing. Most applications concern themselves only with frontal views, for obvious reasons. Face detectors are a popular method to assess the view of the person, as faces are by far the most interesting part of the body in surveillance applications.

We, however, take a more active approach and use tracking information to estimate the view angle in order to collect the proper data to populate our multiview appearance model, described above and shown in Figure 4.1. This view angle variation about the person's vertical axis is known as the *Azimuth*, and is illustrated in Figure 4.2.

The Azimuth is calculated from the camera pose and the person's current state information as:

$$Azimuth = cos^{-1}\left(\frac{(C_{xy} - P_{xy}) \cdot (V_{xy})}{\|C_{xy} - P_{xy}\|\|V_{xy}\|}\right) \tag{4.1}$$

with an extra check to obtain the direction of the azimuth from the reference.

As we are interested in several different poses of the modeled person, we define our azimuth score as dependent on the desired view, $\theta_{Az,desired}$:

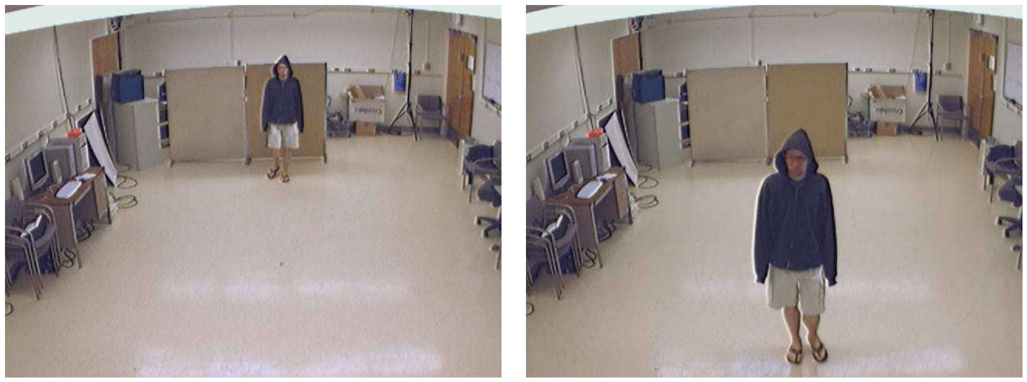
$$\Psi^{Az} = e^{-\frac{(\theta_{Az}-\theta_{Az,desired})^2}{\sigma_{Az}^2}} \tag{4.2}$$

where σ_{Az} is a tuning parameter for adjusting the azimuth tolerance.

4.4.2 Range

As with most applications, we require our data to be of the highest quality available. With vision and imaging, this translates to image resolution. The higher the resolution of an image, the more data is contained within it. Because of our assumption that the camera nodes are stationary and fitted with fixed lenses, we have no control over the resolution of the video. Thus we must score the images collected by each sensor based on the resolution of the captured person. Figure 4.3 illustrates two extremes of this situation taken from the VISNET system.

As discussed previously, resolution varies linearly with the range of the person from the camera, thus we can use a person's range as input to our scoring process. We define the range score as:



(a) Low Resolution (b) High Resolution

Figure 4.3: Range / Resolution Variation Examples

$$\Psi^D = 1 - \frac{min(D, d_{max})}{4 \cdot d_{max}} \tag{4.3}$$

where D is the distance from the point to the camera center, d_{max} represents the maximum useful distance or, conversely, the minimum useful resolution of an image of a person. This can be easily calculated using the geometry of the camera and the minimum desired resolution (in pixels) of the person.

4.4.3 Zenith

Another factor affecting the quality of view in camera networks is the zenith. The zenith describes the elevation angle of the camera with respect to the point or person it is viewing, and is illustrated in Figure 4.4. Figure 4.5 shows two examples of zenith variation when viewing people.

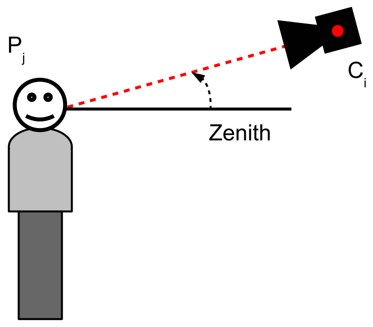


Figure 4.4: Zenith Definition

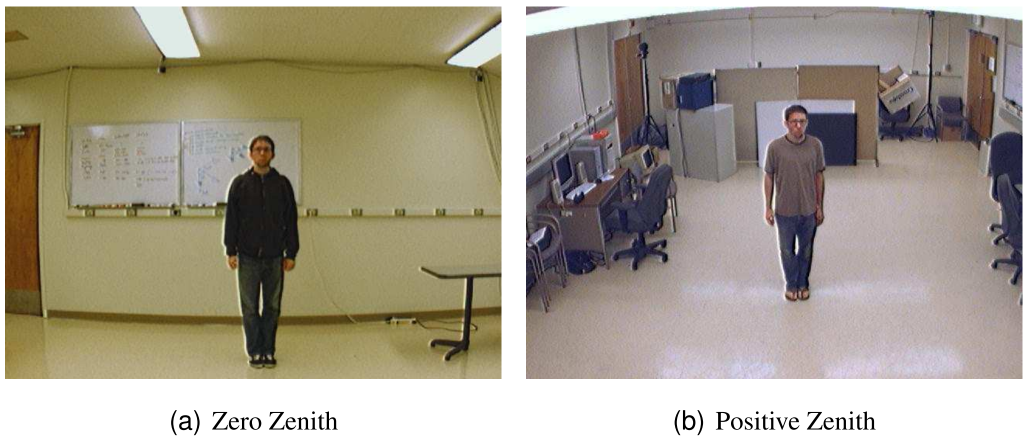


Figure 4.5: Zenith Variation Examples

While some experimental setups ([84]) assume side-mounted sensors (giving a constant zenith of zero), a more realistic scenario consists of high-mounted cameras. As such, we will assign a score which will favor sensors whose zenith is nearer zero. We define the zenith score as:

$$\Psi^{Zen} = e^{-\frac{(\theta_{Zen})^2}{\sigma_{Zen}^2}} \tag{4.4}$$

where θ_{Zen} is defined as:

$$\theta_{Zen} = cos^{-1}\left(\frac{(C-P)\cdot(C_{xy}-P_{xy})}{\|C-P\|\|C_{xy}-P_{xy}\|}\right) \tag{4.5}$$

where C_{xy} is the projection of the camera center onto the ground plane (here, $z = 0$), P_{xy} is the projection of the point P onto the ground plane, V is the current velocity of the point, and V_{xy} is the person's velocity projection onto the ground plane. σ_{Zen} represents a tuning parameter used to adjust the tolerance of the desired zenith value.

4.4.4 Radius

As with the point tracking scenario, the position of the person within the image affects the quality of the data. Aside from the obvious effect of lens distortion, the pose is more difficult to assess when the person is near the sides of the image. Figure 4.6 illustrates this with several examples. We see that when the person is nearer the edge of the image, the pose is more skewed as he is facing the camera center, not perpendicular

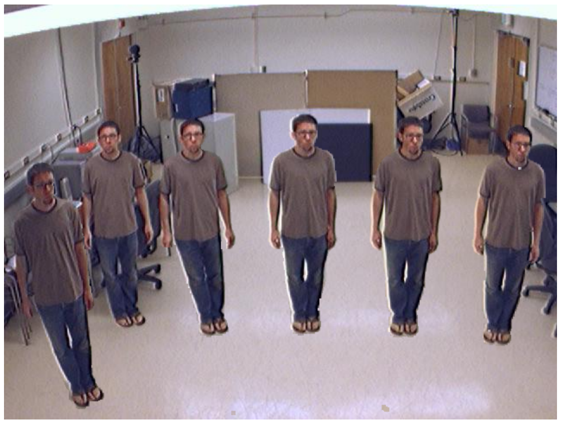


Figure 4.6: Radius Variation

to the image plane. While the difference is not drastic, we still prefer a more centered pose when possible.

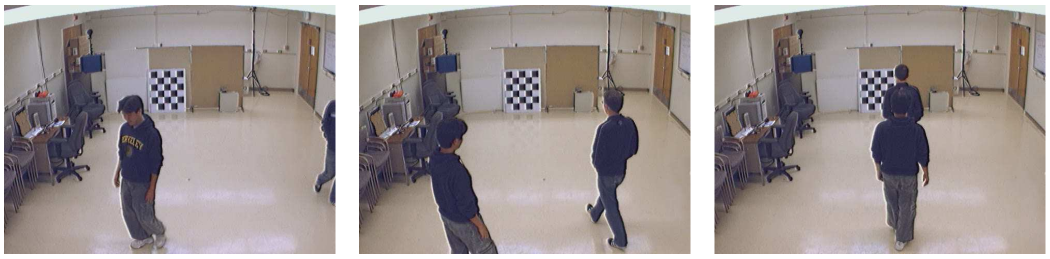
Based on this, we define our radius (image position) score as:

$$\Psi^{i,R} = 1 - \left(\frac{min(R_C^i, R_{max})}{2 \cdot R_{max}} \right)^4 \tag{4.6}$$

where R_C^i is the distance of the person's centroid from the image center, given by the camera's principle point.

4.4.5 Visibility

Because we are not dealing with a point object, we are concerned with whether or not the full person is visible in the frame. To ensure this visibility, we require that both the head and the feet are visible in the camera. We also combine the occlusion measure



(a) Head Not Visible (b) Feet Not Visible (c) Occlusion

Figure 4.7: Visibility Scenarios

in this score. If the person is partially or completely occluded by either another person, a permanent structure, or the edge of the frame, we consider him or her to be occluded and thus the data unusable. Figure 4.7 illustrates these 3 scenarios.

We define our visibility score as:

$$\Psi^{i,V} = \begin{cases} 1 & \text{The person's head and feet are visible in camera } i \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

4.4.6 Combined View Score

Having defined the different criteria which affect view quality and their corresponding scores, we want a single score which will allow us to easily rank the images collected in surveillance system. We define our overall view score as:

$$\Psi^{View} = \Psi^{V}\Psi^{Az}\Psi^{Zen}(w_D\Psi^{D} + w_R\Psi^{R}) \tag{4.8}$$

As it is not dependent on actual image analysis, this measure allows us to assign scores to video based solely on the location and pose of the tracked person.

4.5 Data Collection

In a surveillance scenario, we would like to collect our appearance information as soon as possible so that the model is available to other applications. When a person is discovered, the system should immediately begin tracking him or her, providing position and velocity information to a central controller. As described in Chapter 2, this central controller need not be a dedicated node, but only a process which can fuse local data and perhaps direct local operations.

4.5.1 Tracking Process

Our data collection is driven by the actual tracking process which estimates the state of the people in the system. We require position and velocity information for our view scoring. We model our tracking architecture as a group of sensing nodes reporting to the tracking process, as shown in Figure 4.8. 3D localization is accomplished as described in Appendix B. Tracking can be performed using standard tools such as a Kalman filter or particle filter.

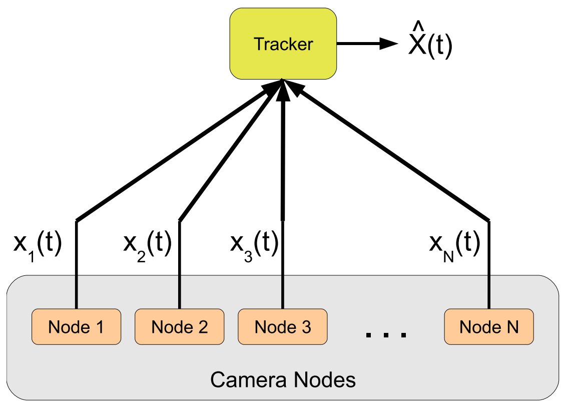


Figure 4.8: System Tracking Architecture

As the person is tracked, the tracker uses the path information to predict the view score for each camera within viewing range. When a desirable score is predicted, the sensing node is assigned the task of data collection and will capture and return the appropriate video frames. Algorithm 5 summarizes the data collection process.

4.6 Model Representation

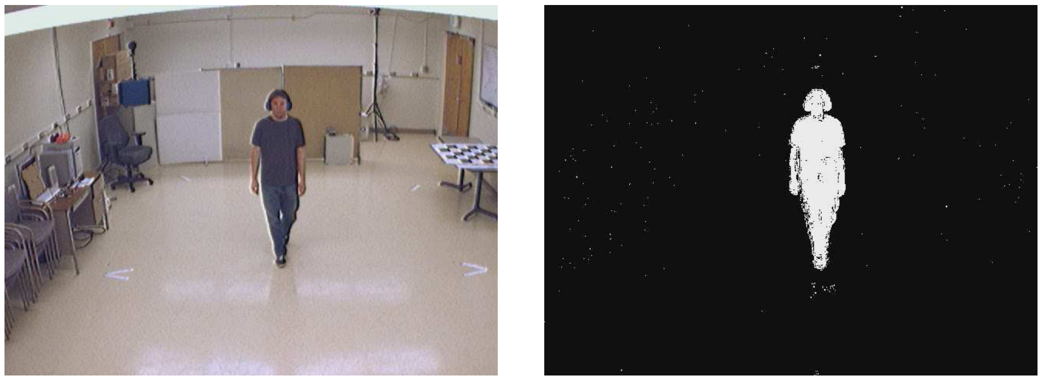
Rather than transmitting full or even cropped video frames through the camera network, we are interested in a more compact appearance representation which is minimal in size yet still retains important visual information about the modeled person. We used standard MPEG-7 descriptors in [66], but are interested in more adaptable and more compact format. We find these features with Gaussian mixture models.

```
while $tracking == TRUE$ do
    Receive_Data_From_Nodes()
    Localize_ Point()
    Compute_State_Prediction()
    Compute_View_Score_Predictions()
    if $predicted\ score > current\ score$ then
        Send Collection Instructions()
    end
end
```

Algorithm 5: Best View Collection Process

4.6.1 Person Segmentation

As discussed in Chapter 2, we segment people from our video using background subtraction. As we currently operate indoors in a somewhat controlled environment, this method has proven sufficient for our tracking and modeling purposes. Figure 4.9 shows that the background subtraction output yields a sufficient foreground mask. Figure 4.9(a) shows the video frame and Figure 4.9(b) shows the resulting foreground mask. It is this mask which we use to guide our modeling process, as it indicates the area of the frame which is the person.



(a) Current Image (b) Foreground Mask

Figure 4.9: Background Subtraction Example

4.6.2 Gaussian Mixture Modeling

Due to their ease of use and their wide applicability, Gaussian Mixture Models (GMMs) have been used extensively for the modeling of many types of data, including imagery [38] and audio data [61]. A GMM models a data distribution as a mixture, or summation, of several weighted Gaussian models. The general notation for a GMM is:

$$p(x) = \sum_{i=1}^{K} \pi_i p(x|i) \tag{4.9}$$

where $\sum_i \pi_i = 1$. π_i is the weight assigned to mixture component i, which is also the probability that a point x was generated from mixture component i. Each mixture component is represented by the probability:

$$p(x|i) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_i)\Sigma_i^{-1}(x-\mu_i)^T} \tag{4.10}$$

The choice of color space for our model is important, as we prefer that the model have meaning and, more importantly, the comparison of models be as truly discriminative as possible. We base our choice of color space on prior work in image retrieval [38] [69]. The use of the CIELAB and CIELUV color spaces (relatives of each other) offer the best discrimination of colors as compared to the human visual system. [59], [35], and [1] provide more information on the various color spaces.

Figure 4.10 provides a more detailed view of a person's appearance data (pixel color) in the CIELUV color space, with pairwise groupings of color channels showing the clustering of the data.

4.6.3 Joint Spatial Appearance Model

In order to preserve structural information from the modeled person, we include in our model spatial information from each pixel. The works in [38], [20], and others have shown that by incorporating spatial information with the color information, better differentiation is achieved. Due to the non-rigidity of the human silhouette, pixel location in the horizontal direction is variable and thus we use only vertical position of each pixel. The vertical pixel location is normalized to lie between 0 (the head) and 1 (the feet). This partitioning is akin to segmenting the body into regions and model-

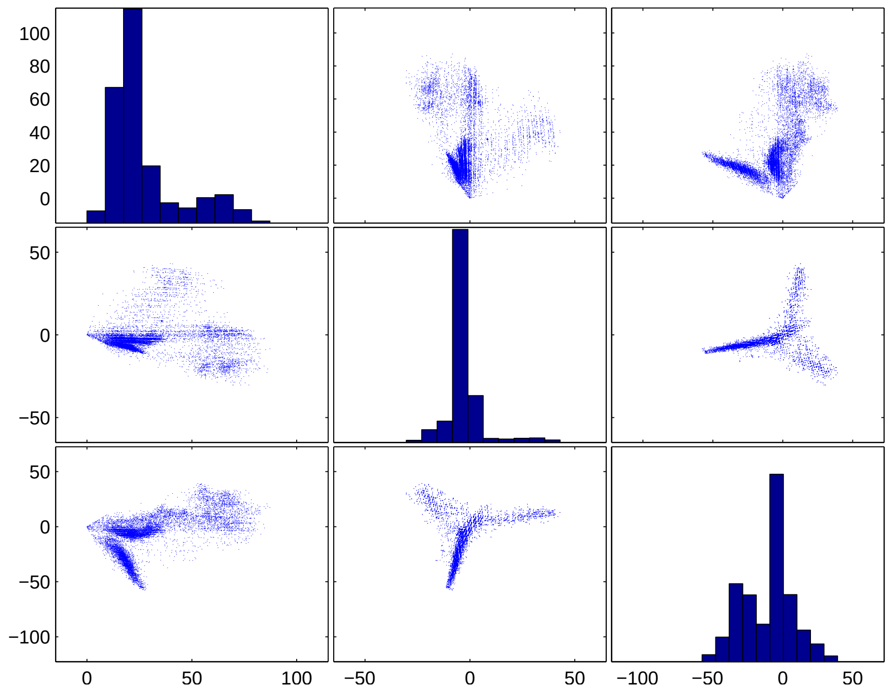


Figure 4.10: CIELUV Color Space Detail. Each row and column indicates one dimension of the colorspace. Here, the dimensions are L, U, and V (left to right) and L, U, and V (top to bottom). Each box shows the distribution of one feature versus another while the (i, j) boxes where $i = j$ show the histogram of each dimension.

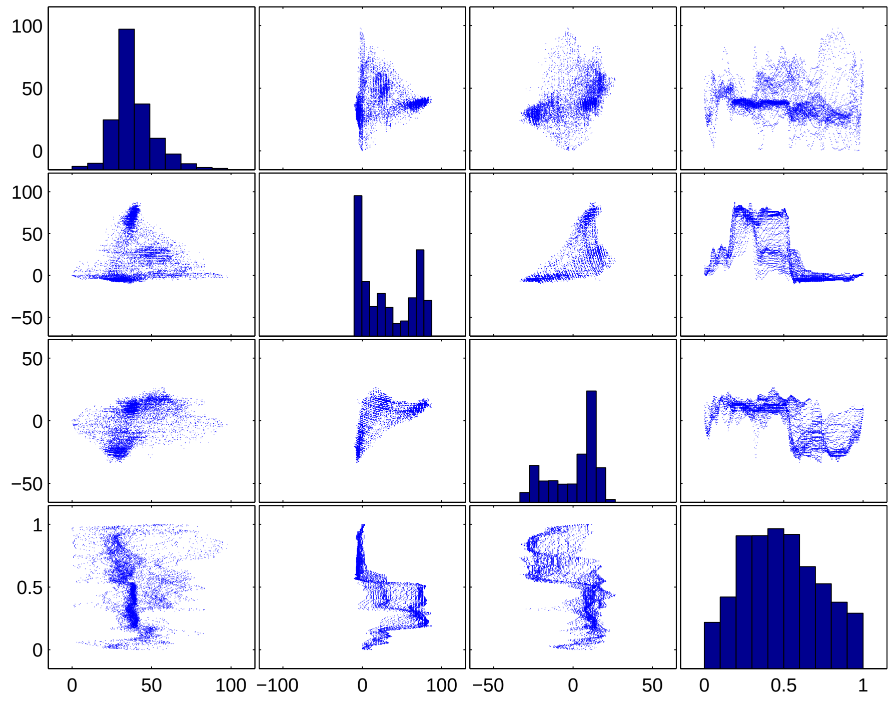


Figure 4.11: Joint Spatial and Color Pixel Distribution Detail. Each row and column indicates one dimension of the feature space. Here, the features are L, U, V, and position (left to right) and L, U, V, and position (top to bottom). Each box shows the distribution of one feature versus another while the (i, j) boxes where $i = j$ show the histogram of each dimension.

ing each individually, as in [23] and [66], but better encodes the feature variation, as in [38]. Figure 4.11 shows an example of the data distribution for all channels when spatial information is used to describe a tracked person.

4.6.4 Model Generation

With the color space chosen, we now move on to fitting a Gaussian Mixture Model to our data in order to significantly reduce the storage / transmission requirements.

Expectation Maximization (EM) is commonly used for this stage. The EM process functions to find unknown model parameters in a system. In the case of mixture model fitting, EM finds the parameters $\zeta = \{\mu_i, \Sigma_i, \pi_i\}_{i=1}^{K}$ of the mixture components in our model.

Expectation Maximization is a two-step iterative process which alternates the Expectation Step (E-Step) with the Maximization Step (M-Step) until a stop condition is met. The E-Step computes an expectation of the likelihood based on the current parameter estimate (Equation 4.11). The M-Step of the process utilizes the output of the E-Step to compute the maximum likelihood estimates of the parameters (Equations 4.12-4.14). The steps are alternated until a stop condition - often a minimal change in the likelihood - is reached. The result of the Expectation Maximization process is a model, in the form of the parameters $\zeta = \{\mu_i, \Sigma_i, \pi_i\}_{i=1}^{K}$, describing the distribution of data of our detected person.

$$q_{ji} = \frac{p_i(x_j)\pi_i}{\sum_{k=1}^{N} p_k(x_j)\pi_k} \tag{4.11}$$

$$\pi_i = \frac{\sum_{j=1}^{N} q_{ji}}{N} \tag{4.12}$$

$$\mu_i = \frac{\sum_{j=1}^{N} q_{ji}x_j}{N} \tag{4.13}$$

$$\Sigma_i = \frac{\sum_{j=1}^{N} q_{ji}(x_j - \mu_i)(x_j - \mu_i)^T}{N} \tag{4.14}$$

Silhouette Edge Point Suppression

Because the segmentation output can be noisy, binary image morphology is often used to fill gaps or holes in a detected silhouette. Because of the noise and subsequent processing, the exact edges of the tracked person are often themselves noisy. As a precaution, we suppress the effects of these edge points in order to produce a more reliable appearance model.

This suppression of edge data is sometimes used in color-based tracking [28], [19], [90]. In the first two works, a circular kernel, often the Epanechnikov kernel, is applied to a tracked patch of the image. In the case of a non-square region, the kernel is resized accordingly. Still, the kernel is defined over the entire rectangular region and does not take into account the shape or pose of the tracked object or person and thus the tracked object's shape is assumed to be elliptical. [90] uses, instead, a Gaussian kernel to weight the pixels near the edge of the tracked region.

However, when modeling a tracked person, we would like to exclude all non-person data in order to obtain a strong model of the person. Because the shape and pose of the tracked person varies as they move, we need a weighting kernel which adapts with this variation. [22] introduced the use of the normalized chamfer distance transform for generating a silhouette-guided weighting kernel for use in a mean-shift tracking framework. The distance transform provides a measure of a pixel's distance from the nearest edge - in this case the edge of the silhouette.

We desire a more even weighting of the interior pixel data, however, and want to suppress only the extreme edges of the silhouette. To this end, we introduce a piecewise weighting kernel for our modeling process. This kernel is based upon (and is actually a function of) the distance transform, but is applied only to the pixels near the edge of the silhouette. We define our kernel as:

$$K(x_i) = \begin{cases} 0 & d(x_i) \leq 0 \\ \frac{1}{2} sin(\pi(\frac{d-\frac{d_e}{2}}{d_e})) + \frac{1}{2} & 0 < d(x_i) \leq d_e \\ 1 & d(x_i) > d_e \end{cases} \tag{4.15}$$

where d_e is the desired width of the edge weighting of the silhouette, defined as a fraction of the bounding box width. The kernel profile is shown in Figure 4.12. Figure 4.13 shows an example of the data weighting kernel for a typical silhouette.

Weighted Expectation Maximization

As in [28], the use of the weighting kernel on the pixel data requires the modification of our modeling process. In our case, we must modify the Expectation Maximization process in order to account for the weights. The derivation of the Weighted EM (WEM)

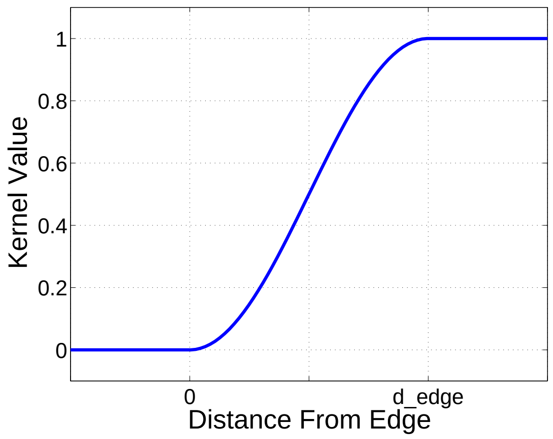


Figure 4.12: Silhouette Weighting Kernel Profile

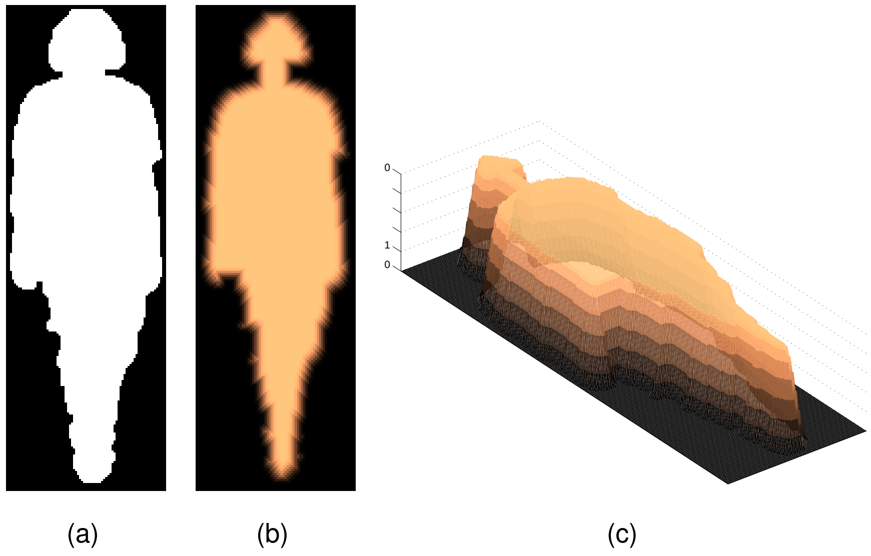


Figure 4.13: Silhouette-Guided Weighting Kernel. (a) Foreground Mask, (b) Weighting Kernel Overhead View, (c) Weighting Kernel Orthographic View

is derived by approaching the process from the viewpoint of inserting additional copies of the higher-weighted data. The resulting E-Step and M-Step equations of the WEM process are given in Equations 4.16 and 4.17-4.19, respectively. Here, λ_j represents the weight assigned to pixel j by our weighting kernel.

$$q_{ji} = \frac{p_i(x_j)\pi_i}{\sum_{k=1}^{N} p_k(x_j)\pi_k} \tag{4.16}$$

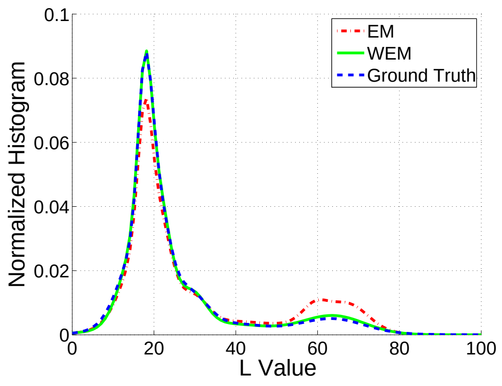
$$\pi_i = \frac{\sum_{j=1}^{N} \lambda_j q_{ji}}{\sum_{j=1}^{N} \lambda_j} \tag{4.17}$$

$$\mu_i = \frac{\sum_{j=1}^{N} \lambda_j q_{ji} x_j}{\sum_{j=1}^{N} \lambda_j} \tag{4.18}$$

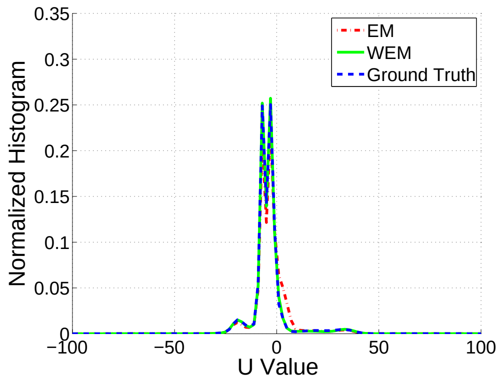
$$\Sigma_i = \frac{\sum_{j=1}^{N} \lambda_j q_{ji}(x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^{N} \lambda_j} \tag{4.19}$$

Figure 4.14 shows a comparison of the resulting pixel distribution of the WEM process as compared to regular EM. We see that by reducing the influence of the potentially errant border pixels, the distribution more closely resembles that calculated using the ground truth silhouette.

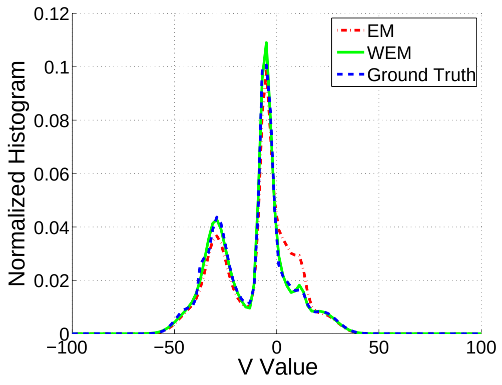
(a) Segmented Person



(b) L Channel



(c) U Channel



(d) V Channel

Figure 4.14: Weighted Expectation Maximization Example. Color channels are separated for easier comparison.

4.6.5 Model Order Selection

When fitting mixture models to data using the Expectation Maximization process, we must specify in advance the desired model order. In some applications ([89]) the model order is determined manually via prior investigation of the object appearance. In others, mixture component probabilities are ordered and summed from greatest to lowest with components left after some threshold sum is reached are discarded [83].

We desire a more systematic approach which will optimize the model fit to our data. We utilize the *Minimum Description Length (MDL)* criterion, described in [17]. The MDL criterion is defined as:

$$MDL(K,\zeta) = -log(\mathcal{L}(x|K,\zeta) + \frac{1}{2}L \cdot log(NM) \tag{4.20}$$

where $\mathcal{L}(x|K,\zeta)$ is the likelihood given the model order K and the model parameters ζ. N is the number of data points, M is the data dimension, and L represents the number of parameters required to represent the model. L is defined as:

$$L = K\left(1 + M + \frac{M(M+1)}{2}\right) + 1 \tag{4.21}$$

The first term of Equation 4.20 signifies the model fit to the data and (naturally) decreases with increasing K. The second term represents a penalty term, penalizing higher order models. The MDL criterion has similar form to the Akaike Information

Criterion (AIC) and the Bayes Information Criterion (BIC), with the penalty term varying between the different criteria [75]. The use of the MDL criterion ensures that we have the best model fit for the current data. Our experiments returned typical model orders between 12 and 15.

4.6.6 Mixture Model Distance Measure

In order to properly differentiate objects represented by a mixture model, we seek a proper measure of similarity. Histogram-based modeling methods such as [27] and [22] use binwise comparison such as the Bhattacharyya distance. However, with a parametric model, we no longer have access to raw data and instead seek a distance measure which will utilize our appearance model.

We adopt as our distance measure the Earth Mover's Distance (EMD) [69]. The EMD represents a measure of how much "work" is required to transform one distribution into another. The name comes from the analogy of using one distribution, the earth, to fill in holes represented by the second distribution. The EMD is useful for comparing two objects represented by lists of features, called signatures in most works, which are not necessarily the same length. In the case of GMMs, the signature is the set of Gaussian components in the mixture. [69] and [38] demonstrate its use for comparing mixture model representations of images in a database.

We adopt as our ground distance the Frèchet distance, used in [38]. The Frèchet distance between two single Gaussians N_1 and N_2 is defined as:

$$d_{Frechet}(N_1, N_2) = \left(\|\mu_1 - \mu_2\|^2 + Trace\left[\Sigma_1 + \Sigma_2 - 2\left(\Sigma_2 \Sigma_1\right)^{\frac{1}{2}} \right] \right)^{\frac{1}{2}} \tag{4.22}$$

4.6.7 Multiview Distance Measure

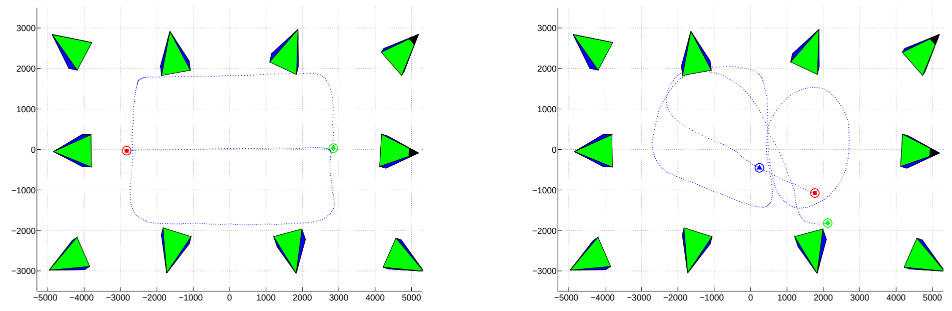
Our distance measure for the multiview appearance model is upon the individual views' distances when comparing two models. We define our multiview model distance measure as:

$$d_{MV}(M_1, M_2) = \max_{\theta \in \Theta} d(M_1(\theta), M_2(\theta)) \tag{4.23}$$

where Θ is the set of views for which both models have view data. Thus we are only computing differences when reliable data is available. By taking the maximum distance, we are looking at the worst view match between the two models.

4.6.8 Results

Using the VISNET system, we collected two video data sets. We refer to them as the *Rectangular* set and the *Random* set, indicating the path of motion of the subject within the system. The paths walked are shown in Figure 4.15, with the start and finish



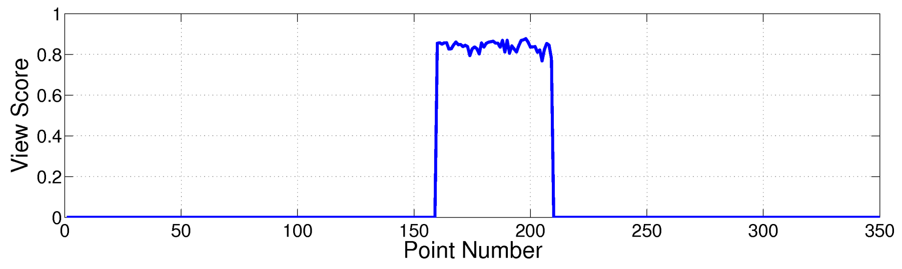
(a) Rectangular Path - 350 points (b) Random Path - 480 points

Figure 4.15: View Selection Experiment Paths. Nodes are numbered from the lower right with Node 1 and proceeding counterclockwise. Green indicates path start and red indicates path end. The blue mark indicates point 457 in the *random* sequence.

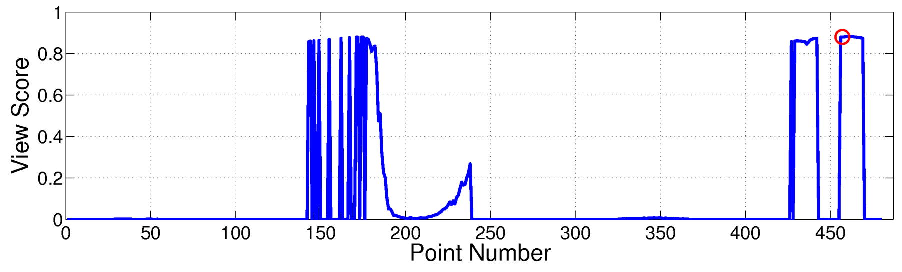
points indicated in green and red, respectively. As the person is tracked through the system, we apply our view prediction score from Equation 4.8 in order to rank and collect the best views of the person.

To demonstrate the functionality of the system, we first present the case for frontal view only, but will later discuss the results from the other 7 views in our model. Figure 4.16 shows the frontal view score from camera 1 for both data sets. Correlating with the points in Figure 4.15, we can easily see the evolution of the view score of this node.

Figure 4.17 shows the top 6 frames for the front view from camera 1 with associated view scores. We see that each provides a clear front view of the tracked person. Inspection shows that these frames are located near point 457 in the *Random* sequence, indicated by a blue dot in Figure 4.15(b).



(a) Rectangular Path



(b) Random Path

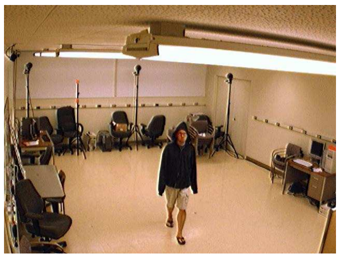
Figure 4.16: Camera 1 Frontal View Scores

(a) Score: 0.8814

(b) Score: 0.8814

(c) Score: 0.8810

(d) Score: 0.8808


(e) Score: 0.8806

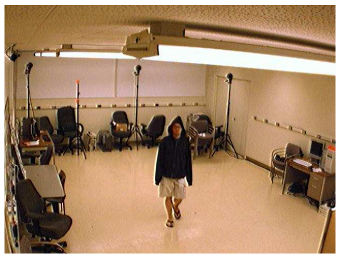
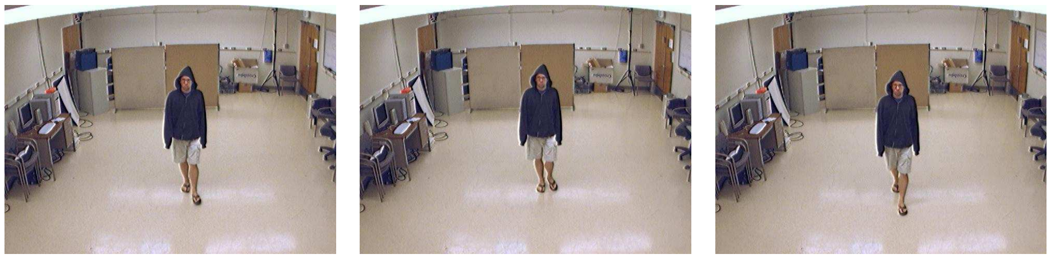

(f) Score: 0.8800

Figure 4.17: Camera 1 Frontal View Scores



(a) Node: 5 - Score: 0.8900 (b) Node: 5 - Score: 0.8900 (c) Node: 5 - Score: 0.8897

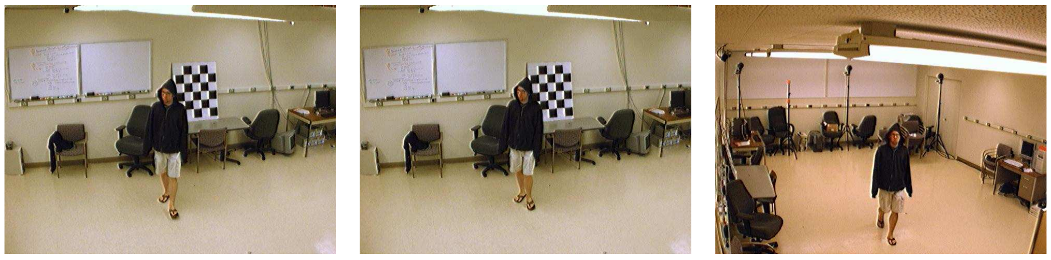
Figure 4.18: Global Best Scores - Rectangular Walk

On the global level, we compare the nodes' view scores in order to obtain the best view from any of the nodes. Here, we look at each sequence separately, with the 3 highest scoreing views of each shown in Figures 4.18 and 4.19. We see that the highest-scoring frame, from camera 5, gives us a large, clear view of the person.

Figure 4.20 summarizes the best views collected for each view angle in our model. The images shown are cropped but not rescaled, preserving the relative resolution. We see that they are similar in resolution and most are the maximum possible attainable given the geometry of the system.

In order to demonstrate the GMM representation and our multiview distance measure, we performed an experiment with 5 people collected in our system. The front views are shown in Figure 4.21.

We generated the mixture model for each view using the procedure described in Section 4.6 . The average mixture model order for the different views was 13. Table 4.1 compares the storage requirements for our model as compared to other options. We



(a) Node: 8 - Score: 0.8837 (b) Node: 8 - Score: 0.8823 (c) Node: 1 - Score: 0.8814

Figure 4.19: Global Best Scores - Random Walk

Method	Single-View Size (bytes)	8-View Size (bytes)
Raw Pixels	244800	1958400
JPEG Images	27964	223712
RGB 8-Bin	8192	65536
GMM 15 Component	1800	14400

Table 4.1: Model Size Comparison

see that our model is significantly smaller than both sending raw data and the widely-used mehtod of using RGB histograms.

Figure 4.22 shows the distances of the individual views of the different models as compared to subject 5. We see that our multiview model successfully captures the difference between subjects 5 and 4.

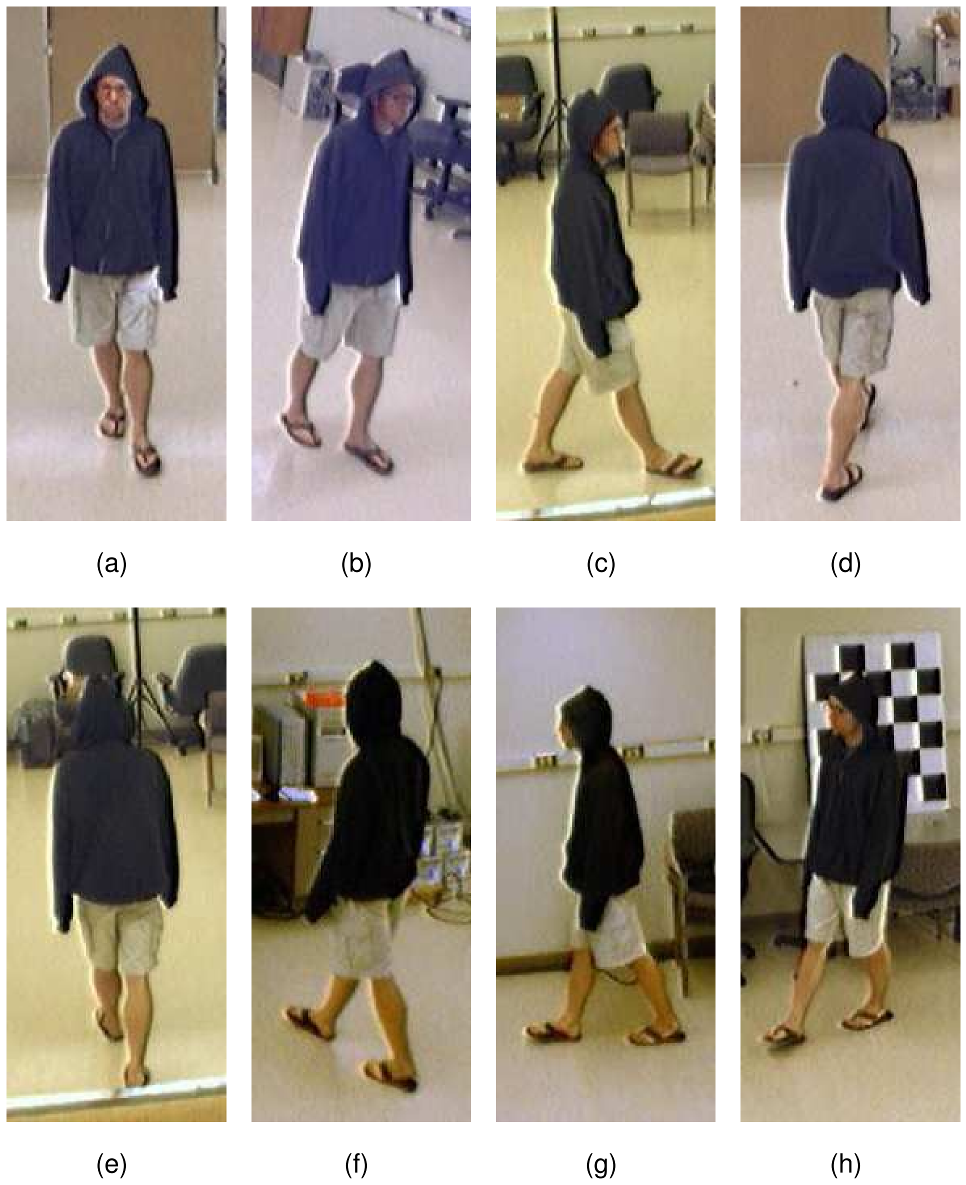


Figure 4.20: Global Best Views. Views (a) through (h) represent view angles 0 through $\frac{7\pi}{4}$, respectively.

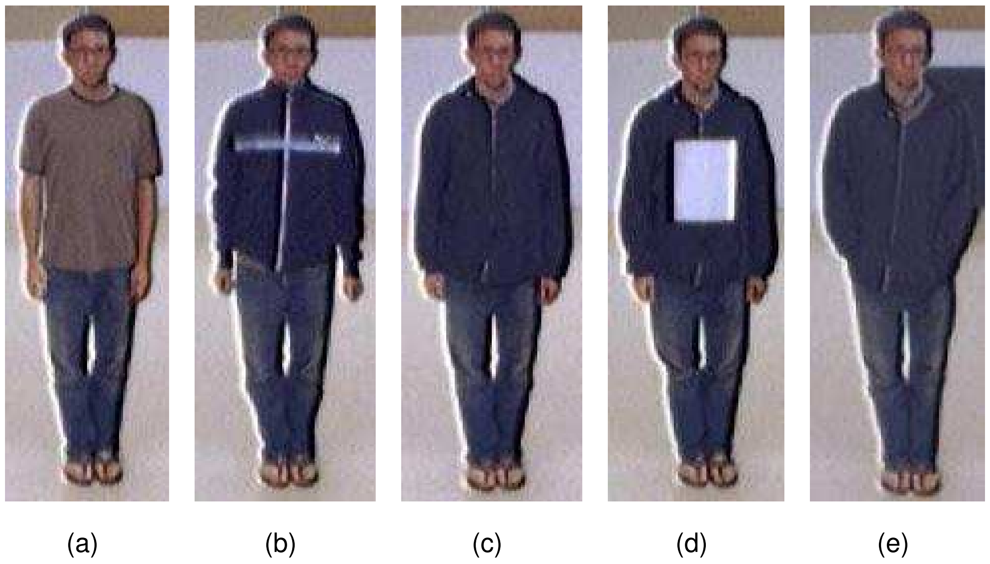


Figure 4.21: GMM Experiment Test Subjects - Front Views

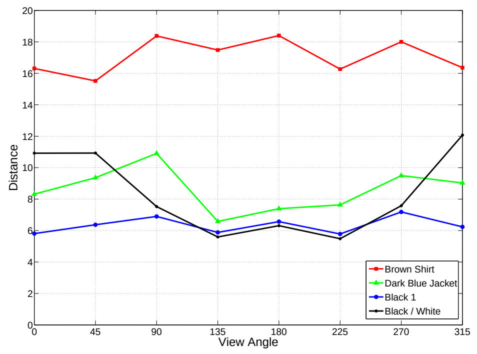


Figure 4.22: Multiview Model Distance

4.7 Summary

In this chapter, we introduced a multiview appearance modeling system for modeling tracked persons in camera networks. Our system is able to predict view quality based solely on tracking and motion prediction information, eliminating the need for costly image analysis of each frame. We introduced a view score which is dependent on:

- View angle (both azimuth and zenith)
- The person's distance from the camera
- The person's position within the frame
- The person's visibility within the frame

We demonstrated our view prediction and collection algorithm on real multicamera data with both a fixed rectangular path as well as a more random walking path. Our system returned clear, high-resolution views of the tracked person during operation.

We then introduced a modeling method by which the appearance data is modeled by a mixture of Gaussians, a common practice in image retrieval. The model allows efficient transmission and storage of appearance data, an important factor in sensor networks. As compared to sending cropped images of even RGB histograms, our model provides a drastic reduction in data size.

Chapter 5

Conclusion

In this dissertation, we discussed issues encountered in camera sensor networks research. In Chapter 2, we addressed the design and development of camera network testbeds. In Chapter 3 we introduced a method for sensor selection and assignment continuity for tracking in camera networks. In Chapter 4 we introduced a data scoring and collection method for multiview appearance modeling in camera networks.

5.1 Conclusions

The overall objective of this thesis was to address the problem of persistent tracking and identification in camera sensor networks. By this we mean that we wish to maintain coverage of an object or event of interest during its lifetime within our network.

In Chapter 2, we presented the design and operation of the VISNET system, a ten-node camera network testbed. Our work on this system involved hardware considerations, operating system choice, software development, and communication protocol development. In addition, we presented two basic applications developed for the VISNET system. We presented an easy to use distributed camera network calibration procedure which utilizes a moving calibration pattern. We then presented a basic multiple camera tracking application which has served as the basis for further tracking development within VISNET.

In Chapter 3 we addressed the issue of sensor selection for localization and tracking in a camera network. We presented a sensor scoring system by which the best pair of sensors for tracking a given position or predicted position can be assigned. This score is based upon different factors affecting the localization process. We demonstrated this method both with simulation and with the VISNET system. We then introduced an inertial assignment method which minimizes sensor activity and the number of handoffs during the tracking process. We demonstrated this in simulation and reduced the sensor activity by 78%.

In Chapter 4 we addressed the issue of appearance modeling in camera networks. Our goal was a human appearance model which captures view-dependent appearance variation as capture by the cameras in a camera network. We introduced a view quality prediction score for view data collection in a camera network. This score is calculated using position and movement information from the person along with camera position information. The use of this score allows predictive collection of data without frame by frame analysis like some other methods use. Implementation within VISNET yielded clear, high quality snapshots of a tracked person within the network. We then demonstrated the size reduction of our model by use of Gaussian mixture models, a common technique for image representation for database retrieval. Use of the GMMs reduced our model size drastically when compared to both the raw image data and the use of RGB histograms.

5.2 Future Directions

Because there are never enough hours in the day to implement every idea, we offer here some suggestions for extending the current work.

5.2.1 Software Library

In Chapter 2 we introduced our custom node communication protocol, based on the Open Sound Control standard. The development of a standard, expandable, easy to use communications library would benefit vision researchers working with camera networks such as VISNET. A lot of time is spent “reinventing the wheel” as the saying goes. A freely available open library would save researchers a lot of development time and allow them to focus on higher level functionality.

5.2.2 Multiple Object Selection

In Chapters 3 and 4, we demonstrated our work in single-object or single-person scenarios. Obviously, more realistic scenarios may involve several people or objects and thus require more adaptable sensor selection methods. The expansion of our work to work in such scenarios would greatly benefit vision applications such as we have presented.

5.2.3 Single Multiview Model

Another direction we have in mind involves the multiple view appearance model presented in Chapter 4. Currently, the model consists of eight separate views. We would like to expand the GMM used for model representation to include the view angle θ in it. This would provide a more compact model and likely describe the variation of the appearance better.

5.2.4 Color Calibration

One of the assumptions made in Chapter 4 was that the cameras's color responses were calibrated to a single global color reference. A set of identically-branded cameras will, in reality, produce as many different color responses. Camera network research would benefit from a standard systematic method for color calibration among the nodes, allowing more reliable color comparison between data taken with different nodes.

5.2.5 System Layout

In Chapter 3, we briefly discussed the effect of network layout on the bounds of localization error. One direction for future work in this area would analyze the network layout in order to establish these bounds in order to predict the best attainable error from the different selection methods.

In conclusion, we see that there are many, many research opportunities in the realm of camera sensor networks. The problems vary from the low-level like color calibration to higher levels like cluster organization.

Bibliography

[1] couleur.org (http://www.couleur.org/). 106

[2] DIVA web page (http://cvrr.ucsd.edu/tswg/diva.htm). 8

[3] (http://www.mip.informatik.uni-kiel.de/∼wwwadmin/software/doc/bias/html/matlabtbundistortion_8cpp-source.html). 149

[4] IEEE 1394 for linux (http://www.linux1394.org/). 26

[5] MIT Robotics, Vision, and Sensor Networks Group (http://rvsn.csail.mit.edu/). 5

[6] Network Timing Protocol (http://www.ntp.org/). 30

[7] OpenCV (http://opencvlibrary.sourceforge.net/). 26

[8] OpenGL (http://www.opengl.org/). 31

[9] QT (http://trolltech.com/products/qt/). 31

[10] Stanford wireless sensor network laboratory (http://wsnl.stanford.edu/). 4

[11] Ubuntu linux website (http://www.ubuntu.com/). 25

[12] UCLA Center for Embedded Networked Sensing (http://research.cens.ucla.edu/). 4

[13] R. Al-Hmouz and S. Challa. Optimal placement for opportunistic cameras using genetic algorithm. In *Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005*, pages 337–341, Dec. 2005. 56

[14] F. Amigoni, M. Somalvico, and D. Zanisi. A theoretical framework for the conception of agency. *International Journal of Intelligent Systems*, 14:449–474, 1999. 19

[15] A. Bakhtari and B. Benhabib. An active vision system for multitarget surveillance in dynamic environments. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 37(1):190–198, Feb. 2007. 22

[16] J. Bouguet. Camera calibration toolbox for matlab manual (http://www.vision.caltech.edu/bouguetj/calib_doc/), 2006. 62

[17] C. A. Bouman. Cluster manual (https://engineering.purdue.edu/ ∼bouman/ software/ cluster/ manual.pdf). 115

[18] R. Bourezak and G.-A. Bilodeau. Object detection and tracking using iterative division and correlograms. In *Proceedings of CRV 2006*, 2006. 91

[19] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. Technical report, Intel Corporation, 1998. 110

[20] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, Aug. 2002. 91, 106

[21] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. *SIGCOMM LA '01: Workshop on Data communication in Latin America and the Caribbean*, 31(2 supplement):20–41, 2001. 30

[22] Z. Chen, Z. L. Husz, I. Wallace, and A. M. Wallace. Video object tracking based on a chamfer distance transform. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 3, San Antonio, TX, Sept./Oct. 2007. 110, 116

[23] S.-Y. Chien, W.-K. Chan, D.-C. Cherng, and J.-Y. Chang. Human object tracking algorithm with human color structure descriptor for video surveillance systems. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 2097–2100, Toronto, Ont., July 2006. 108

[24] C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. In *Proceedings of the IEEE*, August 2003. 7

[25] C.-Y. Chong, F. Zhao, S. Mori, and S. Kumar. Distributed tracking in wireless ad hoc sensor networks. In *Proceedings of the Sixth International Conference of Information Fusion*, 2003. 3

[26] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Carnegie Mellon Institute, The Sarnoff Corporation, 2000. 7, 21

[27] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002. 116

[28] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003. 110, 111

[29] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, Apr. 2004. 58

[30] K. L. Dean. Smartcams take aim at terrorists (http://www.wired.com/science/discoveries/news/2003/06/59092), 2003. 8

[31] D. Devarajan, R. J. Radke, and H. Chung. Distributed metric calibration of ad hoc camera networks. *ACM Trans. Sen. Netw.*, 2(3):380–403, 2006. 40

[32] A. O. Ercan, A. E. Gamal, and L. J. Guibas. Camera network node selection for target localization in the presence of occlusions. In *Proceedings IPSN 2006*, 2006. 22, 57

[33] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, Feb. 2008. 92

[34] A. Gilbert and R. Bowden. Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity. In *Proceedings of ECCV 2006*, pages 125–136. Springer Berlin / Heidelberg, 2006. 89

[35] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, 2002. 106

[36] P. F. Gorder. Sizing up smart dust. *Computing in Science & Engineering*, 5(6):6–9, Nov.-Dec. 2003. 4

[37] R. Goshorn, J. Goshorn, D. Goshorn, and H. Aghajan. Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons. In *Proceedings ICDSC 2007*, pages 219–226, Vienna, Sept. 2007. 57

[38] H. Greenspan, G. Dvir, and Y. Rubner. Context-dependent segmentation and matching in image databases. *Computer Vision and Image Understanding*, 93:86–109, 2004. 91, 105, 106, 108, 116, 117

[39] R. Hammond and R. Mohr. Mixture densities for video objects recognition. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 71–75, Barcelona, Spain, 2000. 91

[40] M. Han, W. Xu, H. Tao, and Y. Gong. An algorithm for multiple object trajectory tracking. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages 864–871, June/July 2004. 90

[41] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000. 8, 91

[42] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 62

[43] K. Heath and L. Guibas. Facenet: Tracking people and acquiring canonical face images in a wireless camera sensor network. In *Proceedings ICDSC 2007*, pages 117–124, Vienna, Sept. 2007. 57

[44] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proc. IEEE Int"l Conf. Computer Vision "99 FRAME-RATE Workshop*, 1999. 90

[45] E. Horster and R. Lienhart. Approximating optimal visual sensor placement. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1257–1260, Toronto, Ont., July 2006. 56

[46] Intel. *Intel Open Source Computer Vision Library Reference Manual*. Intel, 2005. 62

[47] O. Javed. *SCENE MONITORING WITH A FOREST OF COOPERATIVE SENSORS*. PhD thesis, University of Central Florida, 2005. 22

[48] O. Javed, Z. Rasheed, O. Alatas, and M. Shah. Knight: A real time surveillance system for multiple overlapping and non-overlapping cameras. In *Proceedings. 2003 International Conference on Multimedia and Expo, 2003 (ICME '03)*, 2003. 8, 22

[49] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 22–27, Dec. 2002. 90

[50] O. Javed, K. Shafique, and M. Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005 (CVPR 2005)*, 2005. 22, 89, 92

[51] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for easyliving. In *Proceedings.Third IEEE International Workshop on Visual Surveillance, 2000.*, 2000. 22

[52] B. Lei and L.-Q. Xu. Real-time outdoor video surveillance with robust foreground extraction and object tracking via multi-state transition management. *Elsevier Pattern Recognition Letters*, 27:1816–1825, 2006. 91

[53] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, Nov. 2004. 90

[54] M. Lourakis and A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from http://www.ics.forth.gr/∼lourakis/sba. 42

[55] N. Martel-Brisson and A. Zaccarin. Learning and removing cast shadows through a multidistribution approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1133–1146, jul 2007. 90

[56] S. J. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 17:225–231, 1999. 91

[57] M. Minsky. *The Society of Mind.* Simon and Schuster, 1985. 19

[58] L. Navarro, J. Dolan, and P. Khosla. Optimal sensor placement for cooperative distributed vision. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 939–944, April 2004. 58

[59] A. N. Netravali and B. G. Haskell. *Digital Pictures: Representation, Compression, and Standards.* Plenum Press, 1995. 106

[60] N. T. Nguyen, S. Venkatesh, G. West, and H. H. Bui. Multiple camera coordination in a surveillance system. *ACTA Automatica Sinica*, 29 (3):408–422, 2003. 58, 92

[61] A. Nikseresht and M. Gelgon. Gossip-based computation of a gaussian mixture model for distributed multimedia indexing. *IEEE Transactions on Multimedia*, 10(3):385–392, Apr. 2008. 105

[62] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. J. V. Gool. Color-based object tracking in multi-camera environments. In *DAGM-Symposium*, pages 591–599, 2003. 58

[63] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987. 55

[64] J. Park, P. C. Bhat, and A. C. Kak. A look-up table based approach for solving the camera selection problem in large camera networks. In *Proceedings of ACM Sensys 2006*, 2006. 57

[65] K. Pister. Smart dust web page (http://robotics.eecs.berkeley.edu/ pister/smartdust/). 4

[66] M. J. Quinn, T. Kuo, and B. Manjunath. A lightweight multiview tracked person descriptor for camera sensor networks. In *Proc. IEEE International Conference on Image Processing*, Oct 2008. 93, 103, 108

[67] Y. Raja, S. J. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 228–233, Nara, Japan, Apr. 1998. 90

[68] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):65–81, Jan. 2007. 90

[69] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40:99–121, 2000. 91, 106, 116

[70] P. Salembier and T. Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley & Sons, Inc., New York, NY, USA, 2002. 91

[71] C. Shen, C. Zhang, and S. Fels. A multi-camera surveillance system that estimates quality-of-view measurement. In *Proceedings ICIP 2007*, volume 3, San Antonio, TX, Sept./Oct. 2007. 56, 57

[72] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Seattle, WA, June 1994. 90

[73] L. Snidaro, G. L. Foresti, R. Niu, and P. K. Varshney. Sensor fusion for video surveillance. In *Proceedings of the Seventh International Conference on Information Fusion*, volume 2, pages 739–746, June 2004. 91

[74] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, Fort Collins, CO, June 1999. 90

[75] P. Stoica and Y. Selen. Model-order selection. *IEEE Signal Processing Magazine*, 21(4):36–47, July 2004. 116

[76] T. Svoboda, D. Martinec, and T. Pajdla. Convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14:407–422, 2005. 32

[77] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue. Simultaneous localization, calibration, and tracking in an ad hoc sensor network. In *IPSN 2006*, pages 27–33, Apr. 2006. 33

[78] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *ICCV99*, pages 255–261, 1999. 90

[79] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000. 32, 42

[80] J. Tu, H. Tao, and T. Huang. Online updating appearance generative mixture model for meanshift tracking. In *Computer Vision - ACCV 2006 - 7th Asian Conference on Computer Vision, Proceedings*, volume 3851, pages 694–703, Hyderabad, India, Jan. 2006. 92

[81] A. Villacorta. Spheres of influence. In *SIGGRAPH '07: ACM SIGGRAPH 2007 art gallery*, page 254, New York, NY, USA, 2007. ACM. 25

[82] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63:153–161, 2005. 90

[83] H. Wang, D. Suter, K. Schindler, and C. Shen. Adaptive object tracking based on an effective appearance filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1661–1667, Sept. 2007. 91, 115

[84] D. B. Yang, J. Shin, A. O. Ercan, and L. J. Guibas. Sensor tasking for occupancy reasoning in a network of cameras. In *Proceedings of BROADNETS 2004*, 2004. 56, 99

[85] J. S.-C. Yuk, K.-Y. K. Wong, R. H.-Y. Chung, K. P. Chow, F. Y.-L. Chin, and K. S.-H. Tsang. Object-based surveillance video retrieval system with real-time indexing methodology. *Image Analysis and Recognition*, 4633:626–637, 2007. 91

[86] Z. Zhang. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, 1998. 33, 62, 146

[87] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich. Collaborative signal and information processing: an information-directed approach. *Proceedings of the IEEE*, 91(8):1199–1209, Aug. 2003. 11

[88] T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1208–1221, Sept. 2004. 91

[89] Y. Zhu and K. Fujimura. Driver face tracking using gaussian mixture model(GMM). In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 587–592, June 2003. 115

[90] Z. Zivkovic and B. Krose. An EM-like algorithm for color-histogram-based object tracking. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages 798–803, June/July 2004. 91, 110

Appendices

Appendix A

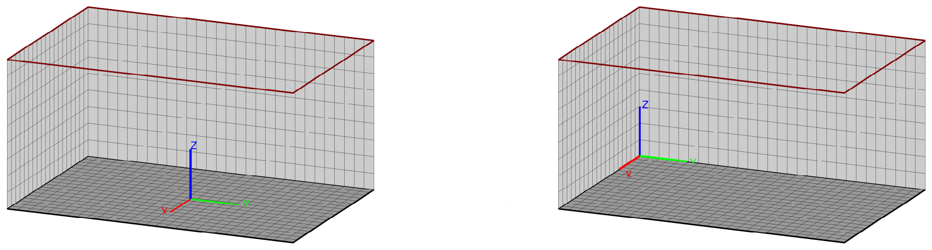
Pinhole Camera Model

The pinhole camera model is the model typically used in computer vision research. The model is named such as its geometry resembles that of the basic pinhole camera, a primitive form of photography. The camera model is described by two sets of parameters: the intrinsic parameters and the extrinsic parameters. The intrinsic parameters describe the projection of a point into the image plane. The extrinsic parameters describe the camera's location in the world with respect to some world 3D coordinate system.

In describing the projection of a point into the image plane, three coordinate systems are defined. These are the world 3D coordinate system, the camera 3D coordinate system, and the image coordinate system. The following sections detail these coordinate systems and the camera parameters.

A.1 Coordinate Systems

The world coordinate system is a 3D coordinate system defined with respect to some world origin. Often, this origin is defined with respect to a feature of the local environment such as the center of a room, a certain corner of the room, or some other landmark. This coordinate system normally gives the user a better reference point for data, as output aligns better with known geometry of the environment. Two possible definitions of the world coordinate system are illustrated in Figure A.1.



(a) Centered Coordinate System (b) Corner Coordinate System

Figure A.1: World Coordinate System Definition

The camera coordinate system describes a points location with respect to the camera center. This origin of this 3D coordinate system coincides with the camera's center point. The z-axis points straight out the front of the camera, with the x-axis pointing directly right and the y-axis pointing downward. This coordinate system is shown in Figure A.2.

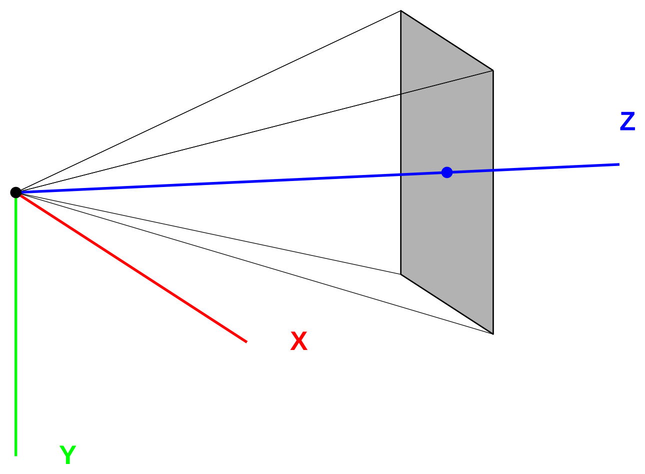


Figure A.2: Camera Coordinate System Definition

The image coordinate system describes a point's location in the image plane itself. This 2D coordinate system is centered at the upper left corner of the image. The u-axis points to the right and the v-axis points downward. The image coordinate system is illustrated in Figure A.3.

A.2 Extrinsic Parameters

A camera's extrinsic parameters describe its location with respect to the world coordinate system. In other words, they describe the transformation of a point's location from the world coordinate system into the camera coordinate system. Because both

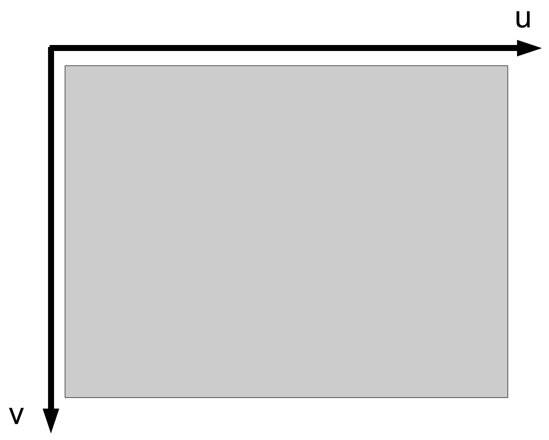


Figure A.3: Image Coordinate System Definition

systems are 3D, the extrinsic parameters take the form of a 3D rotation followed by a 3D translation. These are represented by a rotation matrix R_{WC} and a translation vector T_{WC}, respectively. The transformation is then given by:

$$P^{(C)} = R_{WC}P^{(W)} + T_{WC} \tag{A.1}$$

A.3 Intrinsic Parameters

A camera's intrinsic parameters describe the projection of a point from the camera coordinate system into the image coordinate system. This projection is illustrated in Figure A.4. In addition to the projection, the intrinsic parameters describe any distortion introduced by the camera lens. The intrinsic parameters are typically grouped into a projection matrix K and the distortion vector d. These are defined as

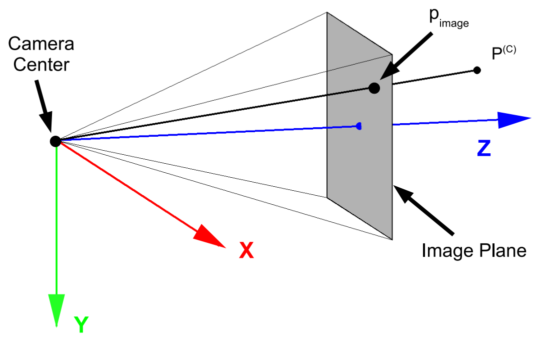


Figure A.4: Camera Projection

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

and

$$d = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 \end{bmatrix} \tag{A.3}$$

The projection of a point $P^{(C)} = [X^{(C)}Y^{(C)}Z^{(C)}]^T$ into its image plane representation $p^{(image)} = [uv]^T$ is a multistep process. First, $X^{(C)}$ and $Y^{(C)}$ are normalized by $Z^{(C)}$:

$$x' = \frac{X^{(C)}}{Z^{(C)}} \tag{A.4}$$

$$y' = \frac{Y^{(C)}}{Z^{(C)}} \tag{A.5}$$

Next, the lens distortion is applied:

$$x'' = x'(1 + k_1 r^2 + k_2 r^4) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \tag{A.6a}$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4) + 2p_2 x' y' + p_1(r^2 + 2y'^2) \tag{A.6b}$$

where $r^2 = x'^2 + y'^2$. Finally, u and v are calculated:

$$u = f_x x'' + c_x \tag{A.7a}$$

$$v = f_y y'' + c_y \tag{A.7b}$$

Sometimes this last stage is represented in matrix form:

$$\begin{bmatrix} u \\ vi \\ 1 \end{bmatrix} = K \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} \tag{A.8}$$

A.4 Camera Calibration

In order to determine the intrinsic and extrinsic parameters of a camera, a calibration procedure is performed. Simply put, the procedure calculates the mapping of 3D world points into their 2D image counterparts. The most widely-used calibration procedure is detailed in [86]. Given a set of known 3D world points $\{P_i^{(W)}\}_{i=1}^N$ and their corresponding projections $\{p_i^{(image)}\}_{i=1}^N$, we define the total reprojection error as:

$$E_{reproj} = \sum_{i=1}^{N} \|p_i^{(image)} - f(P_i^{(W)})\| \tag{A.9}$$

where $f(P_i^{(W)})$ denotes the projection of point $P_i^{(W)}$ into the image plane using the current parameter estimates. The Levenberg-Marquardt algorithm is commonly used to find the camera parameters which minimize Equation A.9.

Appendix B

3D Localization in Camera Networks

The process of object localization in a camera network involves the reconstruction of a 3D point from its 2D projections in 2 or more cameras. Here, we describe the process of 3D point localization in a camera network.

B.1 Two Camera Case

To illustrate the process of 3D point localization, we first look at the case of two cameras. We identify the detected 3D point as $P^{(W)} = [X(W)\ Y^{(W)}\ Z^{(W)}]^T$ and its location in each camera as $p^{(1)} = [u^{(1)}\ v^{(1)}$ and $p^{(2)} = [u^{(2)}\ v^{(2)}$ for cameras 1 and 2, respectively. This is illustrated in Figure B.1.

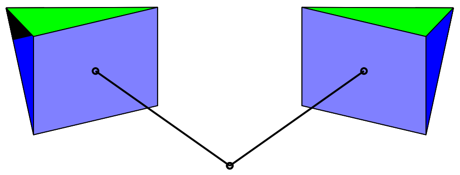
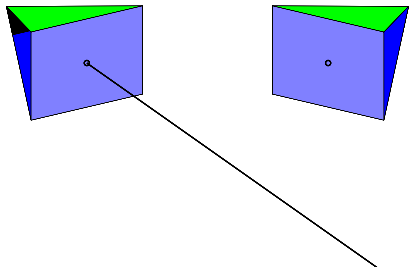


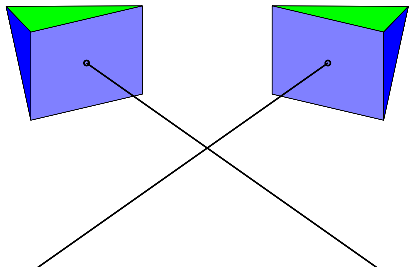
Figure B.1: Projection into Two Cameras

Essentially a bearing sensor, a single camera can only reconstruct a line in 3D space along which the point $P^{(W)}$ is located. This is illustrated in Figure B.2(a). In order to pinpoint the point's position along that line, we require at least one additional camera. By computing the intersection of the two cameras' "bearing lines," we can extract the location of $P^{(W)}$. This is illustrated in Figure B.2(b). Because we require a line intersection, the lines projected by the two cameras must not be parallel, otherwise no intersection will occur.

In practice, system noise significantly reduces the possibility of an actual intersection and we are forced to use an estimate of the point ($\hat{P}^{(W)}$) as our location estimate. The least squares localization process for two cameras is formulated as follows. We



(a) One Camera



(b) Two Cameras

Figure B.2: One and Two Camera Point Localization

first undistort the image points. We use an iterative process, detailed in Algorithm 6, to perform the undistortion [3].

```
**Input**: Distorted image point $x_{in} = [u\ v]^T$
**Output**: Rectified image point $x_{out} = [u'\ v']^T$

$x_{new}$ = $x_{in}$;
while $\Delta error > \epsilon$ do
    $x_{old}$ = $x_{new}$;
    $x_{dist}$ = distort($x_{old}$;
    $x_{new}$ = $x_{in}$ - ($x_{dist}$ - $x_{temp}$);
    $\Delta error$ = $|x_{new} - x_{old}|$;
end
$x_{out}$ = $x_{in}$;
```

Algorithm 6: Image Point Distortion Correction

Using the rectified image coordinates, we now reverse the projection process (sans the distortion step) which is detailed in Appendix A. We write the equations for $u'^{(i)}$ and $v'^{(i)}$ as:

$$u'^{(i)} = c_x^{(i)} + \frac{f_x^{(i)} r_{00}^{(i)} X^{(W)} + f_x^{(i)} r_{01}^{(i)} Y^{(W)} + f_x^{(i)} r_{02}^{(i)} Z^{(W)} + T_x^{(i)}}{r_{20}^{(i)} X^{(W)} + r_{21}^{(i)} Y^{(W)} + r_{22}^{(i)} Z^{(W)} + T_z^{(W)}} \tag{B.1a}$$

$$v'(i) = c_y^{(i)} + \frac{f_y^{(i)} r_{10}^{(i)} X^{(W)} + f_y^{(i)} r_{11}^{(i)} Y^{(W)} + f_y^{(i)} r_{12}^{(i)} Z^{(W)} + T_y^{(i)}}{r_{20}^{(i)} X^{(W)} + r_{21}^{(i)} Y^{(W)} + r_{22}^{(i)} Z^{(W)} + T_z^{(i)}} \tag{B.1b}$$

where

$$R_{Wi} = \begin{bmatrix} r_{00}^{(i)} & r_{01}^{(i)} & r_{02}^{(i)} \\ r_{10}^{(i)} & r_{11}^{(i)} & r_{12}^{(i)} \\ r_{20}^{(i)} & r_{21}^{(i)} & r_{22}^{(i)} \end{bmatrix} \tag{B.2}$$

and

$$T_{Wi} = \begin{bmatrix} T_x^{(i)} \\ T_y^{(i)} \\ T_z^{(i)} \end{bmatrix} \tag{B.3}$$

are the extrinsic parameters of camera i. Using simple algebra, we can reformulate this problem as a function of $X^{(W)}$, $Y^{(W)}$, and $Z^{(W)}$:

$$A \begin{bmatrix} X^{(W)} \\ Y^{(W)} \\ Z^{(W)} \end{bmatrix} = z \tag{B.4}$$

with

$$A = \begin{bmatrix} f_x^{(1)}[r_{00}^{(1)}\ r_{01}^{(1)}\ r_{02}^{(1)}] - (u'^{(1)} - c_x^{(1)})[r_{20}^{(1)}\ r_{21}^{(1)}\ r_{22}^{(1)}] \\ f_y^{(1)}[r_{10}^{(1)}\ r_{11}^{(1)}\ r_{12}^{(1)}] - (v'^{(1)} - c_y^{(1)})[r_{20}^{(1)}\ r_{21}^{(1)}\ r_{22}^{(1)}] \\ f_x^{(2)}[r_{00}^{(2)}\ r_{01}^{(2)}\ r_{02}^{(2)}] - (u'^{(2)} - c_x^{(2)})[r_{20}^{(2)}\ r_{21}^{(2)}\ r_{22}^{(2)}] \\ f_y^{(2)}[r_{10}^{(2)}\ r_{11}^{(2)}\ r_{12}^{(2)}] - (v'^{(2)} - c_y^{(2)})[r_{20}^{(2)}\ r_{21}^{(2)}\ r_{22}^{(2)}] \end{bmatrix} \tag{B.5}$$

and

$$z = \begin{bmatrix} T_z^{(1)}(u'^{(1)} - c_x^{(1)}) - T_x^{(1)} \\ T_z^{(1)}(v'^{(1)} - c_y^{(1)}) - T_y^{(1)} \\ T_z^{(2)}(u'^{(2)} - c_x^{(2)}) - T_x^{(2)} \\ T_z^{(2)}(v'^{(2)} - c_y^{(2)}) - T_y^{(2)} \end{bmatrix} \tag{B.6}$$

which can then be solved using the linear least squares method:

$$\hat{P}^{(W)} = (A^T A)^{-1} A^T z \tag{B.7}$$

B.2 3+ Cameras

The use of 3 or more cameras for point localization is straightforward as Equation B.4 scales easily. Figure B.3 illustrates a situation where 4 cameras are used to localize a point in 3D. Depending on the geometry of the situation and the system noise, additional cameras may or may not improve localization results.

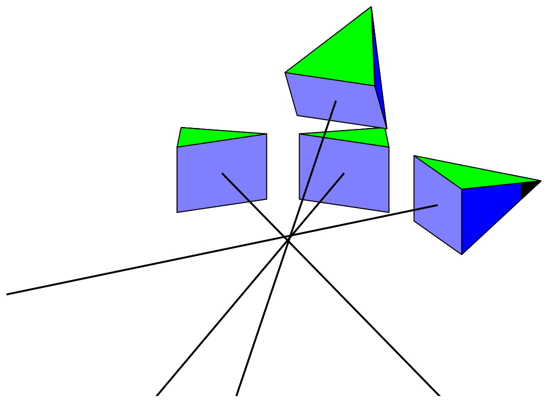


Figure B.3: 3D Point Localization with 4 Cameras

Appendix C

VISNET Message Formats

Message Component	Description
/extrinsics	Indicator of extrinsic parameters to follow
Node_ID	ID of originating node
time_sec	Timestamp of data (seconds since epoch)
time_usec	Timestamp of data (microseconds)
r_0 r_1 r_1	Rotation vector
t_x t_y t_z	Translation vector

Table C.1: VISNET Extrinsic Parameters Message Detail

Message Component	Description
/relextr	Indicator of relative extrinsic parameters data to follow
From_Node_ID	ID of originating node
To_Node_ID	ID of node to which the relative parameters refer
r_0 r_1 r_1	Relative rotation vector
t_x t_y t_z	Relative translation vector

Table C.2: VISNET Relative Extrinsic Parameters Message Detail

Message Component	Description
/obj	Indicator of object data to follow
Node_ID	ID of originating node
time_sec	Timestamp of data (seconds since epoch)
time_usec	Timestamp of data (microseconds)
u	Local u coordinate of tracked object
v	Local v coordinate of tracked object

Table C.3: VISNET Object Tracking Message Detail

Message Component	Description
/ctrl	Indicator of control data to follow
Control_Code	Control Command (0 = Stop, 1 = Start)

Table C.4: VISNET Node Control Message Detail