DESIGN AND PERFORMANCE OF MULTI-CAMERA NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Itai Katz
December 2010

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Andrea Goldsmith, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

John Haymaker, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Hamid Aghajan

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Martin Fischer

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Camera networks have recently been proposed as a sensor modality for 3D localization and tracking tasks. Recent advances in computer vision and decreasing equipment costs have made the use of video cameras increasingly favorable. Their extensibility, unobtrusiveness, and low cost make camera networks an appealing sensor for a broad range of applications. However, due to the complex interaction between system parameters and their impact on performance, designing these systems is currently as much an art as a science. Specifically, the designer must minimize the error (where the error function may be unique to each application) by varying the camera network's configuration, all while obeying constraints imposed by scene geometry, budget, and minimum required work volume. Designers often have no objective sense of how the main parameters drive performance, resulting in a configuration based primarily on intuition. Without an objective process to search through the enormous parameter space, camera networks have enjoyed moderate success as a laboratory tool but have yet to realize their commercial potential.

In this thesis we develop a systematic methodology to improve the design of multi-camera networks. First, we explore the impact of varying system parameters on performance motivated by a 3D localization task. The parameters we investigate include those pertaining to the camera (resolution, field of view, etc.), the environment (work volume and degree of occlusion) and noise sources. Ultimately, we seek to provide insights to common questions facing camera network designers: How many cameras are needed? Of what type? How should they be placed?

First, to help designers efficiently explore the vast parameter spaces inherent in multi-camera network design, we develop a camera network simulation environment to rapidly evaluate potential configurations. Using this simulation, we propose a new method for

camera network configuration based on genetic algorithms. Starting from an initially random population of configurations, we demonstrate how an optimal camera network configuration can be evolved, without *a priori* knowledge of the interdependencies between parameters. This numerical approach is adaptable to different environments or application requirements and can efficiently accommodate a high-dimensional search space, while producing superior results to hand-designed camera networks. The proposed method is both easier to implement than a hand-designed network and is more accurate, as measured by 3D point reconstruction error.

Next, with the fundamentals of multi-camera network design in place, we then demonstrate how the system can be applied to a common computer vision task, namely, 3D localization and tracking. The typical approach to localization and tracking is to apply traditional 2D algorithms (that is, those designed to operate on the image plane) to multiple cameras and fuse the results. We describe a new method which takes the noise sources inherent to camera networks into account. By modeling the velocity of the tracked object in addition to position we can compensate for synchronization errors between cameras in the network, thereby reducing the localization error. Through this experiment we provide evidence that algorithms specific to multi-camera networks perform better than straightforward extensions of their single-camera counterparts.

Finally, we verify the efficacy of the camera network configuration and 3D tracking algorithms by demonstrating their use in empirical experiments. The results obtained were similar to the results produced by the simulated environment.

Acknowledgements

I extend my gratitude and admiration to the Construction Metrology and Automation Group at the National Institute of Standards and Technology (NIST) for their support throughout my program at Stanford. In particular I'd like to thank group leader Alan Lytle, who gave me the utmost flexibility and provided the research environment that allowed me to work off-campus.

I'd also like to thank Hamid Aghajan who inspired me to study the field of camera networks. My work with him and the other members of the Wireless Sensor Networks Lab is the genesis of this work.

I am grateful to Andrea Goldsmith not only for lending her academic support as my advisor, but also for believing in me.

Working with John Haymaker has been a privilege. John's extensive knowledge of design and construction processes greatly influenced this dissertation by providing a real-world deployment scenario of the ideas described herein. His understanding and charisma made our discussions a pleasure and I admire his ability to turn any challenge into an opportunity.

I extend my appreciation to my orals committee, Andrea Goldsmith, John Haymaker, Hamid Aghajan, and the chair Martin Fischer for taking the time and effort to evaluate my work and to provide me with valuable suggestions to improve its quality.

Finally, my deepest gratitude goes to my family, Ilan, Orly, Roey, and Karen, who instilled in me a thirst for knowledge and the inspiration to excel. Their boundless support, in the form of late night draft reviews, encouragement, and unlimited cell phone minutes, made this work possible.

"Scholars multiply peace on earth"—*Babylonian Talmud, Y'vamot 122b*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"You can observe a lot just by watching."
Yogi Berra

1.1 Multi-camera Networks

Arrangements of multiple cameras have become commonplace in recent years. Consider the following scenarios:

- Intelligent transportation systems monitor video streams to detect trends in urban traffic [19]
- Surveillance cameras observe heavily trafficked transportation hubs to aid in crime prevention [85]
- Construction site camera networks allow foremen to supervise the flow of vehicles and personnel in crowded construction sites [44]

In each of these examples, a human operator interprets a collection of video streams in order to maintain *situation awareness*. Situation awareness is defined as the ability to perceive, comprehend, and predict the behavior of environmental elements in a given span of time and space. Military strategists often explain situation awareness as a continuing cycle of observations, orientation, decision, and actions (the "OODA Loop"). In most

cases, including those described above, the majority of the OODA loop is under machine control with the exception of one vital aspect: the decision phase. For decisions involving complex reasoning the use of human operators is inevitable, both at the present time and the foreseeable future. For spatial tasks that are subject only to geometry and physics, machines frequently deliver an advantage; the performance of a human in these tasks is often inconsistent. The quality of decisions an individual can make is governed by a variety of factors such as fatigue, skill, information load, and even temperament.

By removing the limitations that are inherent to human operators and replacing them with automated systems, camera networks can be built to perform tasks that would otherwise be impossible. Whereas humans are excellent at detecting anomalous events, for instance, they are relatively poor at making accurate measurements. A dense crowd of people can trivially be monitored for unusual incidents but the exact number of people in that crowd can be difficult to count. Similarly, an operator observing a highway traffic camera can identify stopped vehicles, but cannot measure the speed of traffic by visual inspection. Automating the decision phase with a collection of cameras can have broad benefits to measurement science; such systems are known as *multi-camera networks*.

The study of multi-camera networks is a quickly developing field with a wide breadth of promising applications. Due to recent improvements in computer vision and hardware costs, multi-camera networks have become an increasingly favorable sensor modality for 3D tasks. Localization and tracking are typical examples of the multi-camera network's utility. Research interest in the area has followed suit, as evidenced by the growing publication count: 25 publications in 2001, 128 in 2005, and 259 in 2008[1]. Part of the appeal stems from the technology's wide applicability: such diverse fields as industrial automation [71], ambient intelligent environments [61], and object tracking [10] can all benefit from an inexpensive and reliable 3D sensor.

[1]From a Google Scholar search for academic papers whose titles include the phrases "camera network", "multi-camera network", or "calibrated camera network".

1.2 Challenges

Despite their usefulness, automated multi-camera networks are largely confined to the laboratory setting with relatively few commercial examples. While meticulous error analysis is possible in a controlled research environment, this is not usually the case in deployment scenarios. The ability to predict the performance of a camera network with a given configuration is currently an open problem. This issue is particularly pressing in environments with stringent requirements (e.g., manufacturing) or where the environment is highly dynamic and cluttered (e.g., construction). Current methods of camera network design are *ad hoc*, in that the system is primarily configured by intuition, with a resulting *in situ* performance that is difficult to anticipate.

The performance of any particular camera network is difficult to generalize to all configurations. Unlike single-element sensors that can be mass produced, each camera network must be tailored to the particular environment under consideration. An environment with many occlusions (e.g., a parking garage with many columns) will need more cameras than an open environment in order to ensure an adequate line-of-sight to at least two cameras. Furthermore, the specific computer vision application being applied dictates the level of precision needed. Detecting the pose of an object with many articulated joints, for instance, requires far higher precision than simply localizing an object. Thus the critical parameters of a multi-camera network need to be determined on a per-installation basis. How many camera are needed? With what resolution? And how should they be posed to maximize system performance? In addition to the geometric limitations imposed by a physical system, the algorithms used to drive the camera network are an additional source of error. Most contemporary computer vision applications are designed with a single video stream in mind. Algorithms for camera networks are typically straightforward extensions to multiple views. This approach neglects to take advantage of multiple view geometry, while also ignoring its limitations. With regards to camera placement, developers typically rely on rules of thumb inherited from photogrammetry. These rules are sufficient for low-precision applications, but for multi-camera networks to live up to their promise, a more structured approach is needed.

Given the necessity of a theoretical foundation it is a reasonable question to ask why this line of research has not already been explored. There are several reasons for this. First, the study of multi-camera networks is still a burgeoning field, with serious consideration only being given in the last few years. It is not unusual for emerging technologies to be motivated by particular applications, with the theoretical analysis to follow. Modern computers, for instance, were developed as a practical tool long before computer science was established as a field of study in its own right.[2]

Another reason for the information gap is that multi-camera networks have their roots in a much older field. The predecessor to multi-camera networks, stereoscopy (two cameras), has been in use since the early days of photography. The theoretical underpinnings, in the form of photogrammatery, were well known by the early 20th century[3]. Once digital imaging became practical, applying computer vision techniques to stereoscopic configurations had been investigated by the 1980s. However, multi-camera networks have sufficiently unique challenges that merits extending this earlier work.

A secondary question should also be considered: why now? It should be noted that the multi-camera network is not a stand-alone technology. All camera network applications, and indeed *any* automated applications involving cameras, depend heavily on computer vision to convert raw sensor data into actionable knowledge. The science of detecting objects in a camera image is just now becoming mature. Only in the last few years has computer vision been robust enough to be incorporated into consumer products. With this crucial component in place, only now can commercial applications of multi-camera networks be considered.

Wider deployment of camera networks is critically dependent on the availability of inexpensive components, including computing equipment, communications gear, and cameras. Digital cameras, the key component of camera networks, have undergone a tremendous commercialization process from concept to consumer product. Figure 1.1 depicts this commercialization trend as a dramatic decrease in cost. This has made the possibility of

[2]The first computer whose logical design could be considered modern, Charles Babbage's Analytical Engine, was developed in the mid-19th century. The pioneering work of Claude Shannon, Alan Turing, and others who established computer science didn't occur until almost a century later.

[3]The earliest known use of photogrammetry came only a few years after the invention of photography, when scientist Aimé Laussedat used it to compile topographic maps for the French military in 1859 [3]—an old science indeed!

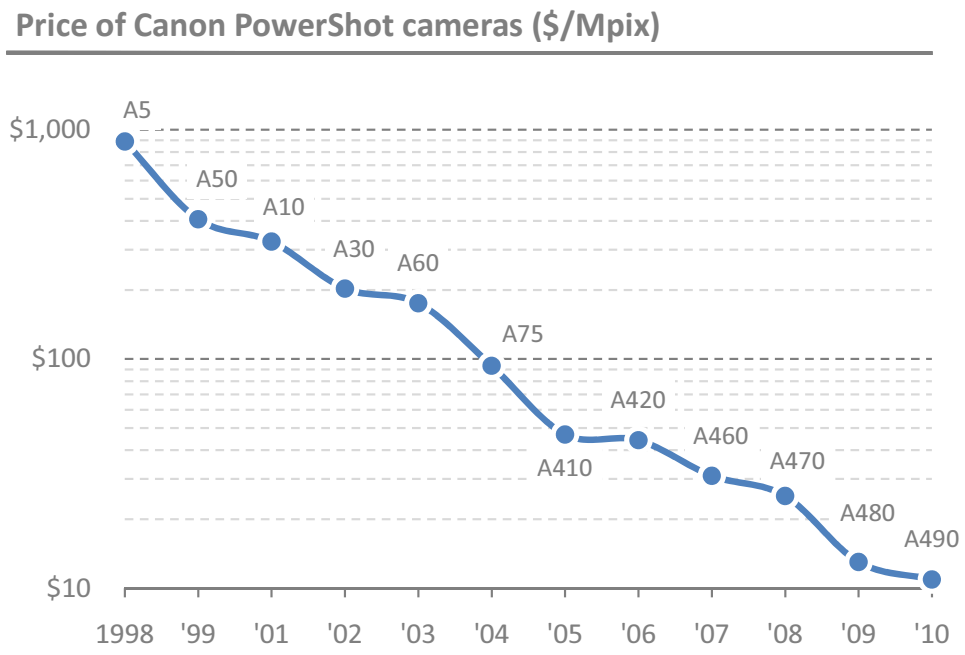


Figure 1.1: Dollars per megapixel, based on the recommended introductory retail price of Canon digital cameras (source: *Digital Photography Review* [1])

installing large numbers of cameras economically viable, which will hopefully spur further research and development.

1.3 Statement of Purpose

In this thesis we present a method for modeling camera networks of arbitrary configurations and environments. Through simulation and empirical verification, a model is developed which will help to better understand how design variables affect performance. The outcome of this investigation is twofold. First, this model will give verifiable insights to a field where design decisions are largely governed by intuition. A software implementation of this model will give hardware designers a tool to rapidly test potential configurations. Second, we will demonstrate how a better theoretical understanding of multi-camera networks can lead to sensor-specific algorithms which outperform the standard methods.

1.4 Research Questions

This thesis will primarily be concerned with two questions:

1. What is a process for understanding how system parameters affect the performance of multi-camera networks?

   The designer is typically faced with three constraints at the start of the design process: application, environment, and budget. Budget obviously affects the number of cameras, as well as associated computing and networking hardware and any installation costs. The demands of the environment and application influence many decisions, including the number, specification, and placement of cameras, as well as what algorithms to use. Since these criteria mutually affect each other, it is difficult to design optimal systems with intuition alone. Factors independent of the design also impact the overall performance—real-world deployments incur significant noise throughout the pipeline (see Figure 1.2, for example), including temporal misalignment between cameras and uncertainty in calibration. Developing a model that considers error as a function of these factors will give a deeper understanding of design trade-offs.

2. What process will allow designers to develop better multi-camera networks?

   With an adequate error model in place, we can turn our attention to processes that aid designers in developing improved multi-camera networks:

   - Because each system is closely associated with its environment and application, a one-size-fits-all solution is unlikely to be practical. Thus the designer is forced to consider each installation anew. A simulated environment could help the designer estimate how the system will behave *prior* to deployment, thereby delivering solutions with predictable, consistent performance.
   - Current computer vision algorithms are typically designed for single-camera operation. The interaction of multiple cameras introduces a number of complications that make standard algorithms perform sub-optimally. A better theoretical understanding can lead to algorithms that are designed with these considerations in mind, leading to higher reliability and accuracy.

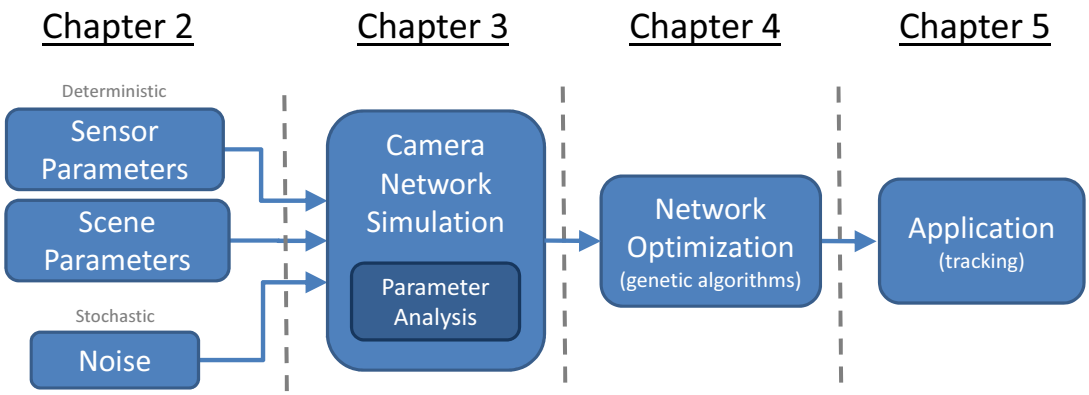


Figure 1.2: Thesis organization process model

1.5 Thesis Organization

This thesis began with a description of open problems associated with multi-camera networks. A process model outlining the relationship between subsequent chapters is illustrated in Figure 1.2. After introducing the research objectives, the remainder of the thesis proceeds as follows:

Chapter 2 defines the parameters associated with multi-camera networks and makes the comparison with other sensing modalities. This is followed by a mathematical primer on camera geometry.

Chapter 3 introduces a simulation environment for multi-camera networks and uses it to investigate how varying system parameters and error sources impact performance on 3D localization. To validate the simulator's correctness, a series of empirical studies are carried out, whose error is compared to that predicted by the simulator.

Chapter 4 presents a novel method for automated camera network configuration using genetic algorithms.

Chapter 5 uses the insights developed in preceding chapters to demonstrate an approach for 3D tracking designed specifically for use in multi-camera networks. This algorithm is shown to outperform several common approaches to 3D tracking.

Chapter 6 concludes this thesis with a summary and suggests potential avenues for future research.

Chapter 2

Camera Network Basic Principles

In this chapter we cover the background information necessary to support the chapters to follow. We begin with a description of relevant parameters that form the foundation of the analysis in Chapter 3, and continue with a comparison of multi-camera networks to other 3D sensing modalities, followed by an overview of multiple-view geometry.

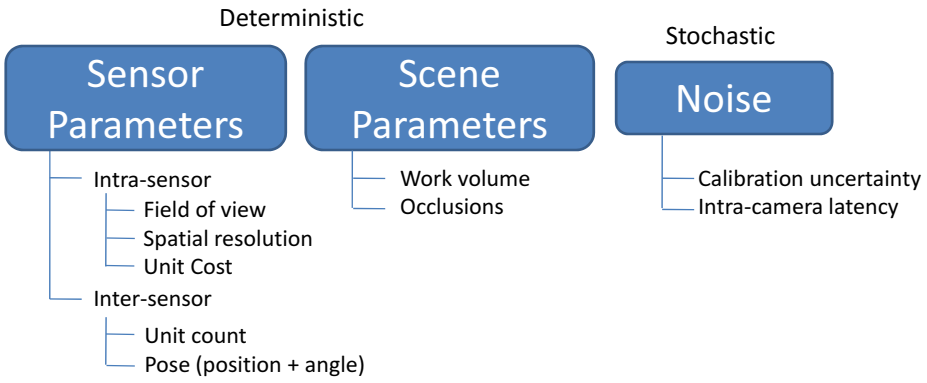


Figure 2.1: Common camera network parameters.

2.1 Camera Network Parameter Description

The performance of a given camera network is a function of a number of factors. Sensor parameters are those which pertain to the hardware itself. These are typically chosen by the

Parameter	***Units***	***Typical values***
Field of view	degrees	45°
Image resolution	pixels	1024 x 768 pixels
Unit cost	$	varies
Unit count	#	4-8
Intra-camera latency	msec	10-20 msec
Position (spatial)	$\{X, Y, Z\}$ m	varies
Orientation (angular)	$\{\alpha, \beta, \gamma\}$ degrees	varies

Table 2.1: Common camera network parameters and their nominal values.

designer. Scene parameters describe the environment which contains the camera network and the work volume it is observing. These values are usually fixed at design time and are application-dependent. Error sources are pseudorandom elements which are the result of practical limitations inherent in multi-sensor systems, and are the result of uncertainty in measuring the parameters. As a result, these values are fixed but difficult to measure precisely.

2.1.1 Sensor Parameters

Most of the design choices faced by camera network designers involve a tradeoff. Since a change in one parameters often influences another, these values need to be considered simultaneously. Furthermore, the optimal values are highly application dependent. Here we describe a few of the most important parameters.

Field of View This parameter describes the angular extent that is viewable in a single frame. A wide field of view provides greater coverage, while a narrow field of view provides greater detail (i.e., a "zoomed-in" image). Most camera networks tend towards the former, since fewer cameras are needed to provide adequate coverage across the entire work volume. One caveat for wide fields of view is the tendency to produce distorted images; a fish-eye lens is an extreme example of this phenomenon. For fixed lenses, the field of view needs to be determined at design time. Zoom lenses provide a mechanism to adjust the field of view after installation.

Image Resolution The number of pixels on the image sensor determines the image resolution. The lowest resolution typically encountered, 640 x 480 pixels (VGA resolution), is usually found in low-cost webcams. Higher resolutions are becoming more popular, with 1280 x 1024 not uncommon. From an image standpoint quality, "more pixels" is not strictly better. Since image quality is a function of both resolution and pixel size, a small sensor with high resolution tends to produce grainy images. For this reason, higher-end sensors are typically sized at 1/2" to 2/3" (measured diagonally). Resolution is a critical design consideration, as it is one of the major limiting sources in localizing a point in three dimensions.

Unit Count The number of cameras in the network is a function of the viewable work volume, which is typically specified as part of the application. The number and placement of cameras is subsequently configured to attempt to recreate the desired size and shape of the work volume. This recreation is usually only approximate: as the number of cameras increases, the marginal benefit to the size of the work volume decreases as the percentage of the viewable work volume approaches 100% (Figure 2.2).

Position and Orientation (Pose) Many simulations that attempt to find optimal camera pose treat these as free parameters. As a practical matter, the position and orientation are constrained by the shape of the environment and availability of mounting hardware. In an indoor installation, for instance, cameras are usually only placed on the walls and ceiling, unless tripods are to be used.

2.1.2 Scene Parameters

Work Volume The work volume describes the region in space in which a 3D point can be localized. Since at least two cameras are needed to observe a point, a work volume is effectively the region where two or more camera views overlap. For tracking objects that move freely in three dimensions, this volume is often a cubeoid. When tracking objects that are constrained to the ground plane (e.g., people, vehicles, etc.), the height of the work volume can be significantly constrained in the Z axis.

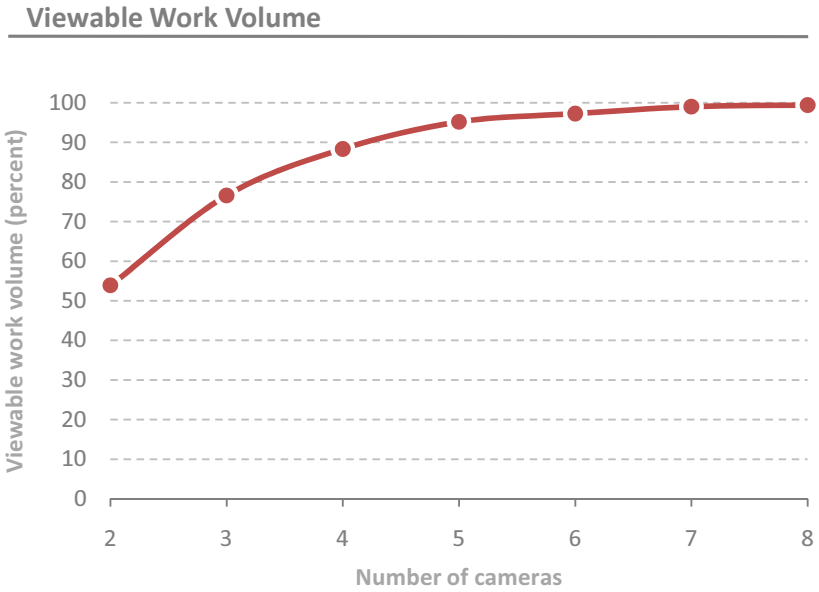


Figure 2.2: Percentage of work volume (covered by two or more cameras). As additional cameras are added, the percent covered asymptotically approaches 100%. Since system cost scales linearly with the number of cameras, how close to match the desired work volume is a design consideration.

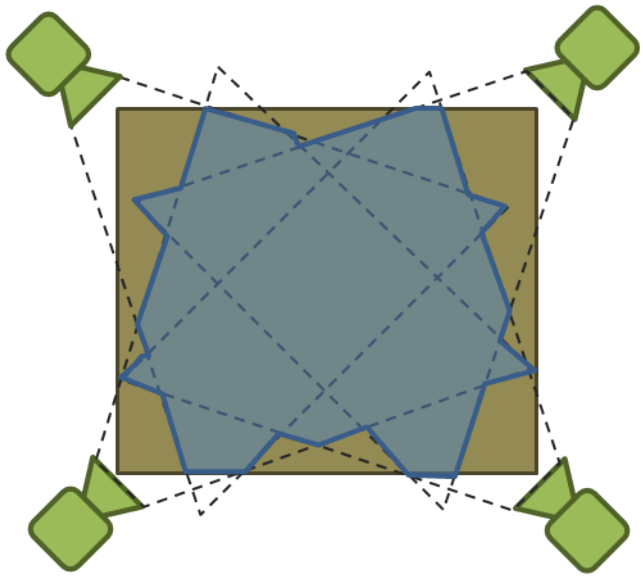


Figure 2.3: The work volume resulting from the overlap of two or more camera views (blue) might have a substantially different shape than the expected work volume (brown).

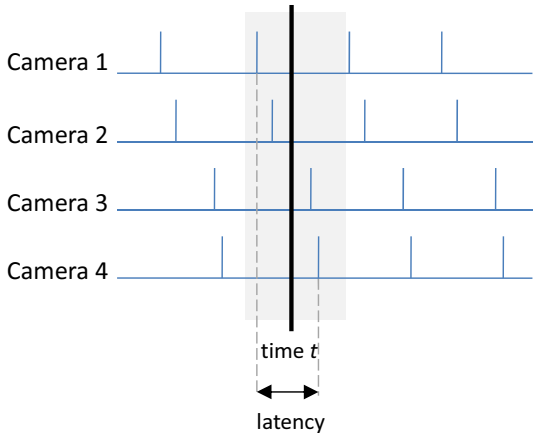


Figure 2.4: Intra-camera latency

Occlusions Most realistic scenes contain some number of occluding elements. These can be static occluders that are part of the scene itself (typically structural columns) as well as dynamic occluders caused by objects passing in front of one or more cameras. In some scenarios, occlusions can occupy a substantial amount of the scene, resulting in a large amount of visual "clutter". This is particularly true of construction sites, which contain a great deal of unstructured motion and whose scene is constantly changing.

2.1.3 Error sources

Intra-camera Latency This value refers to the longest period of time for any two cameras in the network to capture a simultaneous event (Figure 2.4). Due to errors in clock signal propagation, an event at time t will be captured by camera i at time $t + \Delta t_i$. For some applications, acquiring observations out of phase may be a desirable property (e.g., for simulating high frame rates). More commonly, intra-camera latency is detrimental and difficult to measure precisely so is usually treated as noise. The effects of poor synchronization between sensors can be significant. In a tracking application where cameras have a latency of 50 msec, a target moving at 2 m/s (typical walking speed) will result in a tracking error of 10 cm.

Calibration Uncertainty The uncertainty in calibration parameters can have a significant impact on localization accuracy. Most of the literature assumes that calibration parameters are known with perfect accuracy. While arbitrary accuracy can be achieved by incorporating more calibration measurements, these are expensive to capture, particularly in the field where rapid deployment is critical. In practice, camera pose can be off by several centimeters (translation) and several degrees (rotation). For targets far from the cameras, localization error is particularly prone to errors due to rotation uncertainty as these errors are amplified with distance.

2.1.4 Other Parameters

There are many other important parameters that will not be discussed in detail, but are mentioned here for completeness. Temporal resolution, or frame rate, is an important consideration when dealing with quickly moving objects. Typical resolution is between 15-30 Hz, which is adequate for tracking a target at human walking speeds. Specialized applications to track mechanized elements (e.g., manufacturing) may require much higher performance; 100 Hz is not unheard of.

Dynamic range, the levels of light a camera is capable of recording, should be matched to the ambient light level in the environment. Mismatched cameras may saturate in overly bright conditions, or exhibit excessive noise in overly dark conditions.

As a practical matter, the camera bandwidth should also be considered. With the avalanche of data that modern, high resolution cameras provide, bus limitations are frequently encountered, particularly when multiple cameras share a common bus.

2.2 Sensor Comparison

There is no shortage of technologies for 3D localization and tracking. Although the multi-camera network is new to the 3D imaging field, its extensibility and customizability makes it suitable for a wide variety of applications. Being a multi-sensor technology, camera networks offer several advantages over single-sensor devices. Individual sensors, in addition to being constrained to observe objects along their line of sight, are only as useful

as their effective viewable area. Camera networks can accommodate an arbitrarily-sized workspace, and can adapt as the application requirements evolve. While perhaps the most intuitive, cameras are not the only type of sensors employed in localizing and tracking applications:

- RFID uses a physical tag with an embedded ID which can be read from a reader stationed up to several meters away. Although designed to act as a binary proximity device, recent developments have enabled location sensing over RFID [62]. As a radio device it does not need a line of sight, but its range is limited; 20 feet is a nominal operating range [63].

- LADAR is a laser-based range measurement device that operates on a time-of-flight principle. It is designed primarily for scanning 3D objects and scenes, but has been applied to people tracking as well [30]. LADAR devices tend to be the most accurate class of tracking device, but are expensive. Recent years have seen a rapid rise in popularity, particularly for line scanners such as the ubiquitous SICK LMS-200 [88]. A new subclass, Flash LADAR, uses an array of infrared LEDs to provide 30 Hz, full-frame range images [56].

- SONAR enjoyed great success as a tracking technology through the 1990s [51]. Recently, however, its popularity has been declining due to poor sensitivity, low resolution, and the growing availability of laser devices.

2.3 Camera Network Taxonomy

Although collections of cameras have been used since the early days of photography, camera networks as a separate field of study is still a nascent field. As a result much of the terminology is still non-standardized. The wide variety of applications and configuration types result in a large number of ways camera networks can be described. In this section we attempt to develop a taxonomy from trends in the literature.

Distributed vs. Outward-facing vs. Inward-facing The way elements in a network are oriented is largely a function of the target application. Security and surveillance systems

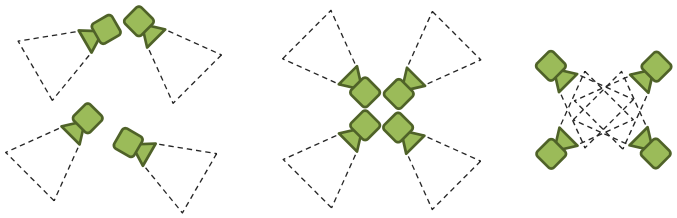
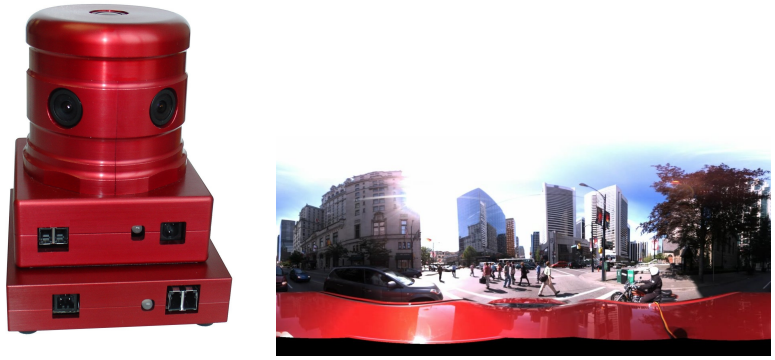


Figure 2.5: Camera network topologies (from l to r): distributed, outward-facing, inward-facing

are typically concerned with maximizing the coverage area and, for cost reasons, minimizing regions of overlapping coverage. *Distributed* networks accommodate this by placing cameras such that their fields of view are mutually exclusive. One common configuration used in large, unobstructed spaces is a grid of downward-facing ceiling mounted cameras. *Outward-facing* networks consist of a collection of elements co-mounted to the same structure, usually with a radial orientation. These types of systems are often used for mapping purposes and, due to their very large viewable area, have comparatively low sensitivity. Conversely, *inward-facing* networks have a much smaller viewable area but have high sensitivity.



(a) (b)

Figure 2.6: An example of an outward-facing camera network. (a) Point Grey Research LADYBUG2 (b) output. This camera was most famously used by the Google Streetview team for rapidly capturing nationwide observations of roadways [34].

Passive vs. Active In the context of camera networks, passive cameras are rigidly installed, while active cameras are free to rotate (for instance, by being mounted on a pan/tilt head) and possibly move by means of a mobile robotic platform, although this is rarely encountered. Note that this usage differs from the more common definition of "passive" and "active" with regards to sensors, which would classify all cameras as passive devices since they have no emmisive properties.

Homogeneous vs. Heterogeneous Most camera networks consist of elements with identical specifications. Some applications call for combinations of different sensors to achieve various resolution requirements. Foveated imaging is one such heterogeneous application, where a passive, wide angle camera captures the entire scene in low resolution and an active, zoom camera observes objects of interest in higher detail [36]. Homogeneous networks are relatively more common in practice.

Calibrated vs. Uncalibrated Calibrated sensors are those whose position, orientation and intrinsic parameters are known. Since the calibration step is time consuming (or impossible, if physical access to the cameras is limited), many applications are designed to work with uncalibrated cameras by making inferences on common features in the scene.

2.4 Practical Considerations

During our interaction with camera network development we came across many issues that have not been emphasized in the existing literature but result in significant obstacles in practical implementations.

Calibration Calibration is a significant subset of the computer vision field, as evidenced by the vast quantity of literature on the subject. Until recently, however, practical *implementations* that are both robust and easy to use have been more elusive. Although calibration is often the first step in any camera network application, how this step is performed is often omitted in the literature. Recent developments have made the process easier and we have found a number of tools to be particularly useful. For finding intrinsic parameters, the

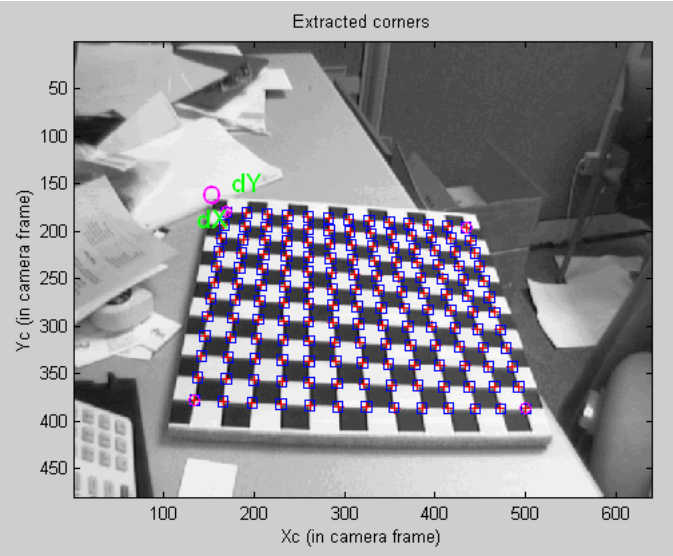


Figure 2.7: The Camera Calibration Toolbox for MATLAB

Camera Calibration Toolbox [11] for MATLAB provides an extremely intuitive interface (Figure 2.7) . For development in C/C++, the calibration functions provided in the OpenCV library [12] are amongst the most popular. These packages provide a high degree of automation in the calibration process, and are primarily based off the algorithm described in [92], which estimates parameters from a sequence of images of a checkerboard pattern.

For extrinsic calibration Direct Linear Transformation (DLT), first developed by photogrammatrists, was one of the earliest calibration algorithms. This algorithm attempts to derive the whole projection matrix from correspondences between 3D world points and 2D image points [2]. While the individual parameters can be extracted from this matrix, DLT's primary drawback is the requirement for dozens of accurately surveyed 3D points. Surveying points is an expensive and time-consuming process, so minimizing the number of parameters to solve for is highly desirable. With intrinsic parameters being so easy to calibrate separately, modern algorithms make use of this knowledge in order to require fewer correspondences. This is called the perspective-*n*-point (P*n*P) problem [26] and some solutions can find orientation parameters using only 4 correspondences (if intrinsic parameters and position are already known). OpenCV also has robust implementations of PnP solvers.

Triggering vs. Synchronization Triggering refers to the ability for multiple elements to make simultaneous observations by means of a common signal. When cameras are triggered they acquire images at nearly the same instant, usually within a few milliseconds of each other. This is a critical consideration for tracking dynamic targets, as a discrepancy of only a few tens of milliseconds can result in a localization error on the order of centimeters for objects moving at pedestrian speeds.

High-precision triggering is accomplished by means of a signal generator and a dedicated hardware port on the camera. This is typically only found on higher-end units. For low-cost devices such as webcams, this process must be accomplished through software and is dependent on a host computer being specially provisioned to meet real-time constraints.

Triggering is obviously a very desirable property but is surprisingly difficult to achieve in practice. How the bus is polled, how the CPU schedules threads, and clock skew resulting from long cable runs to the cameras all affect triggering in ways that are hard to predict. This can be mitigated with the use of one or more signal generators that provide a clock signal to each camera (precluding the possibility of an entirely wireless system and potentially increasing system cost significantly). For outdoor installations, or those with baselines on the order of tens of meters, this approach quickly becomes unmanageable. Software triggering requires the use of a local NTP server and a real-time operating system. For most realistic configurations, this is not possible within a reasonable budget. Despite this, most of the camera network literature omits this consideration and assumes perfect triggering.

Significantly easier to achieve is synchronization, where all elements maintain the same clock (through wireless NTP, for instance), but are free to take observations independently. With an accurate time stamp, latency can be taken into account when designing algorithms specific to camera networks, reducing the requirement for adequate triggering.

Field Installation Setting up a multi-camera network in a field deployment (that is, outdoors and temporary) presents several unique challenges for power, data, and physical installation. While most systems are permanently mounted in indoor environments, some applications call for outdoor setups. This is particularly true when conducting experiments

on wide-baseline configurations (e.g., 100 meters), where indoor testing would be impractical.

When operating away from mains power, the simplest approach to provide power is by running each node (camera, computing hardware, and networking equipment) off a battery or gasoline generator. For moderate baselines, data can be provided over wireless connections (long cable runs being unwieldy). If baselines are on the order of dozens of meters, it is more feasible to forgo networking and store video locally with each sensor, and running tracking algorithms in a post-processing phase. This would preclude real-time operation, but reduce cost and complexity significantly. Lack of networking also makes synchronization difficult. Clocks can be synchronized before installation, but clock drift quickly causes cameras to lose synchronization, often in a matter of hours.

Finally, without a building structure to attach to, cameras must be mounted on tripods. To provide an adequately-sized work volume for wide baselines these tripods often must be extended vertically by several meters. For all but the most rigid tripod masts sway is inevitable, introducing error in the extrinsic calibration parameters.

While outdoor installations have many interesting potential uses, examples of successful deployments are rare [9].

2.5 Mathematical Principles of Camera Networks

This section reviews the basic mathematical tools that describe multi-camera networks. This brief overview should give sufficient background needed for future chapters but is by no means comprehensive. The interested reader is encouraged to explore more thorough treatments on the topic.[1]

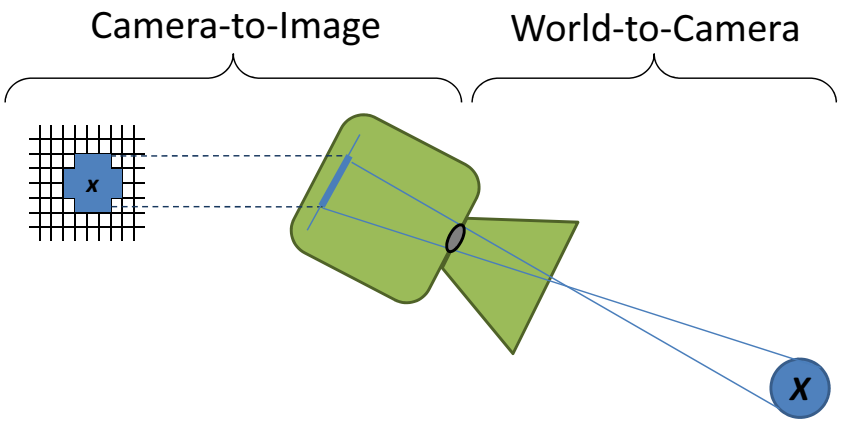


Figure 2.8: The perspective projection model

2.5.1 Image Formation

In the standard pinhole camera model, image formation is a $\mathbb{R}^3 \mapsto \mathbb{R}^2$ projection from global coordinates onto the image plane as shown in Figure 2.8. A point in space $\mathbf{X} = [X,Y,Z]^\top$ is related to a point in the image plane $\mathbf{x} = [u,v]^\top$ through a two stage transformation:

$$s\tilde{\mathbf{x}} = T_{image}^{cam} T_{cam}^{world} \tilde{\mathbf{X}} \tag{2.1}$$

where s is a scaling factor and $\tilde{\mathbf{x}} = [u,v,1]^\top$ and $\tilde{\mathbf{X}} = [X,Y,Z,1]^\top$ are $\mathbf{x}$ and $\mathbf{X}$ represented in homogeneous coordinates. T_{cam}^{world}, called the *extrinsic matrix* describes the camera's pose, a 6D value that encodes the position orientation in space:

$$T_{cam}^{world} = [\mathbf{R} \mid \mathbf{t}] \tag{2.2}$$

for a 3x3 rotation matrix $\mathbf{R}$ and a 3x1 translation vector $\mathbf{t}$. When camera network practitioners refer to the process of calibration, they are usually referring to determining the

[1] Multiple View Geometry in Computer Vision by Hartley & Zisserman [37] serves as the *de facto* reference on the mathematics behind multi-camera networks and is particularly well regarded. Introductory Techniques for 3-D Computer Vision by Trucco and Verri [80] provides an excellent primer on the theory of image formation.

6 parameters that constitute this matrix. A substantial amount of research effort has been focused on finding these parameters efficiently as it is usually the first step in setting up any camera application—every time a camera is physically moved this matrix needs to be recomputed. The extrinsic matrix is also highly pliable since, in theory, all values are physically realizable without regard to hardware constraints. This makes the extrinsic matrix the primary tool for camera network designers seeking optimal configurations.

T^{cam}_{image}, the *intrinsic matrix*[2], encodes the camera's internal parameters and has the following structure:

$$T^{cam}_{world} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

where

- f_x and f_y are the focal length, defined in terms of multiples of horizontal and vertical pixel lengths, respectively. For cameras with perfectly square pixels, $f_x = f_y$. In practice, most pixels are slightly rectangular as a result of CCD manufacturing tolerances.

- c_x and c_y define the pixel where the optical axis of the lens intersects the image plane. These values are nominally the central pixel of the sensor, but any misalignment between the lens and sensor will result in off-center values. This is typically the result of poor mounting between the lens and camera body, as is often the case in commodity webcams.

- s, the skew, describes the orthogonality of the u and v directions. Most texts include this factor for historical reference; modern cameras almost never have skew. For any commercial cameras, s can safely be assumed to be zero.

[2] In the literature this is frequently called the *camera matrix*, somewhat confusingly.

The vast majority of network camera applications deal with fixed (i.e., non-zoom) lenses. In this case, the intrinsic matrix is static quantity that need only be computed once.[3] Otherwise, the matrix can be precomputed at various zoom levels and stored in a lookup table for rapid access.

T_{image}^{cam} and T_{cam}^{world} are often multiplied together to form **P**, the canonical 3x4 *projection matrix*. Thus 2.1 is more commonly encountered as:

$$s\tilde{\mathbf{x}} = \mathbf{P}\tilde{\mathbf{X}} \tag{2.4}$$

2.5.2 Lens Distortion

Perspective projection, a linear model of image formation, is a suitable approximation in many cases. Most real cameras contain non-linear distortion which has the effect of making straight lines curved (Figure 2.9). The degree of distortion is a function of the lens size, lens curvature, and sensor size, but can be approximated in the following manner:

$$dx_{radial} = 1 + k_1 r^2 + k_2 r^4 + \cdots \tag{2.5}$$

where k_1 and k_2 are parameters, and $r = \sqrt{(x-c_x)^2 + (y-c_y)^2}$is the Euclidean distance between the pixel and the optical center. The undistorted pixel is then given as

$$x' = x + dx_{radial} \tag{2.6}$$

In addition to dx_{radial} there exists a $dx_{tangential}$ which is the result of misalignment between elements in a compound lens. Modern manufacturing tolerances are such that this is usually zero. For fixed lenses, distortion parameters need only be calculated once. Once the parameters are known, dx_{radial} can be computed for each pixel and the result stored into a lookup table. This makes undistorting entire images very efficient.

[3] Some manufacturers of high-end sensors take this to the extreme: each camera gets calibrated at the factory and ships with the intrinsic parameters stored in on board FLASH memory.

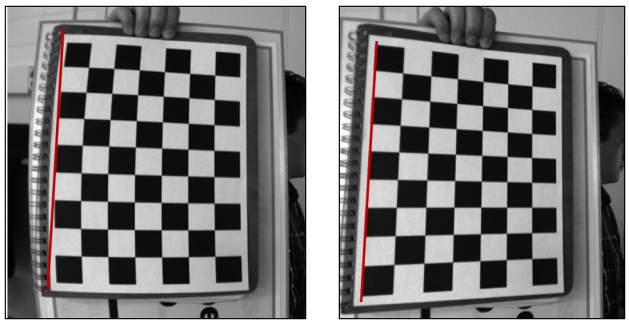


Figure 2.9: Radial lens distortion before and after correction.

2.5.3 3D Point Reconstruction

Given a set of n cameras with projection matrices P_i where $i \in 1 \ldots n$, a point $\mathbf{X}$ in 3D space will satisfy

$$s\mathbf{x} = \mathbf{P_i X} \tag{2.7}$$

for all i. Geometrically, $\mathbf{X}$ can be regarded as the intersection of n rays. However, because real pixels have a physical extent, a pixel projected into space is more accurately described as a square pyramid (Figure 2.10). The intersection of these pyramids forms a volume, in which a 3D point contained anywhere therein will reproject onto the same pixels. Even in a noiseless system (one free from calibration uncertainty, intra-camera latency, etc.) there will be some uncertainty associated with $\mathbf{X}$.

To reconstruct a 3D point, our goal is to minimize the error function, defined as the sum of distances (in pixel units) of the observed image point, and the reprojection of the 3D point we wish to reconstruct:

$$\arg\min_{\mathbf{X}} \sum_{i=1}^{n} \|P_i\mathbf{X} - \mathbf{x_i}\|_2 \tag{2.8}$$

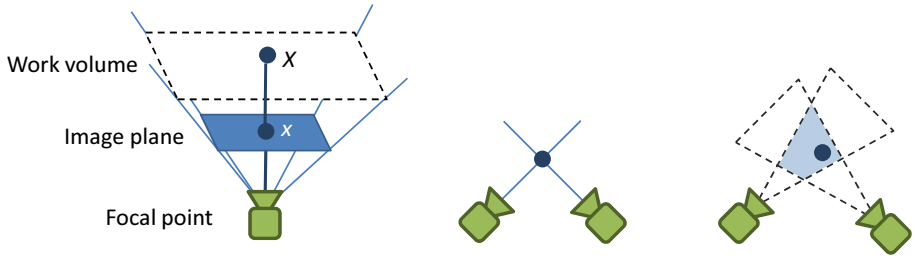


Figure 2.10: Because a pixel has a square extent, a point that projects to a particular pixel can lie anywhere inside the square pyramid that projects out from the camera. While two idealized cameras intersect an object at a point, a more realistic model intersects in a volume. Any point inside this volume projects onto the same set of pixels. This means that the limiting factor in localizing a point is pixel size.

As described in [53], by expanding P_i, and denoting the j-th row as P_i^j, Eq. 2.7 can be rewritten as two linearly independent equations:

$$\begin{aligned} \frac{P_{i,11}X + P_{i,12}Y + P_{i,13}Z + P_{i,14}}{P_{i,31}X + P_{i,32}Y + P_{i,33}Z + P_{i,14}} &= u_i \\ \frac{P_{i,21}X + P_{i,22}Y + P_{i,23}Z + P_{i,14}}{P_{i,31}X + P_{i,32}Y + P_{i,33}Z + P_{i,14}} &= v_i \end{aligned} \tag{2.9}$$

$$\begin{aligned} \left(u_i P_i^3 - P_i^1\right)\mathbf{X} &= 0 \\ \left(v_i P_i^3 - P_i^1\right)\mathbf{X} &= 0 \end{aligned} \tag{2.10}$$

Thus each camera contributes two linear equations, which can be aggregated into the $2N \times 4$ matrix

$$A_0 = \begin{bmatrix} \cdots \\ u_i P_i^3 - P_i^1 \\ v_i P_i^3 - P_i^1 \\ \cdots \end{bmatrix} \tag{2.11}$$

where $\mathbf{AX} = \mathbf{0}$ can be solved with a Singular Value Decomposition (SVD). For diagonal matrix Σ, and orthogonal matrices $\mathbf{U}, \mathbf{V}$, the SVD factors $\mathbf{A}$ such that

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\top}$$

where the least-squares solution to the homogenous system $\mathbf{AX} = \mathbf{0}$ is given by the column of $\mathbf{V}$ corresponding to the smallest diagonal value of Σ. Liu et al. [53] report better results when minimizing instead $\|\mathbf{AX} - b\|$, with $\mathbf{A}$ and b formed from the first three and last columns of $\mathbf{A}_0$, respectively. If $\mathbf{A}^{\top}\mathbf{A}$ is invertible, $\mathbf{X}$ can be solved with a linear least squares approach:

$$\mathbf{X} = \left(\mathbf{A}^{\top}\mathbf{A}\right)^{-1}\mathbf{A}^{\top}b \tag{2.12}$$

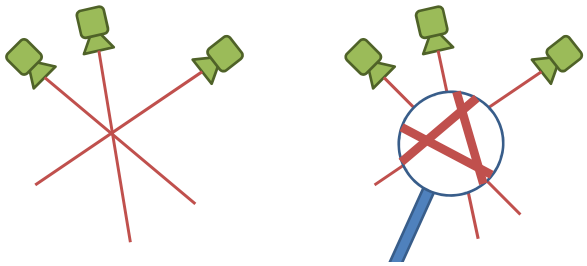
The least squares approach provides analytical backing to several intuitive insights on optimal camera network configuration:

1. At least two cameras are required to reconstruct a 3D point. When rearranged as in Eq. 2.11, $\mathbf{A}$ is a $2m$ x 4-matrix, for m cameras, with each camera contributing one row corresponding to the x and y dimensions. Since vector being optimized is three dimensional (or four, if using the more common homogeneous representation), a minimum of two cameras is required to localize a point in 3D space.

2. Cameras must be arranged so as to form a well conditioned system. If the cameras are collinear with respect to the 3D point, $A^{\top}A$ will not be invertible. Similarly, if they are close to collinear the system will be ill-conditioned. This is illustrated in Figure 2.12.

3. By incorporating additional cameras beyond two, the system is increasingly over-conditioned, reducing uncertainty in the reconstruction. This supports the intuition that in general, more cameras are better.

It should be noted that minimization with least squares is an algebraic approach which does not minimize a physically meaningful quantity. It is suggested that the result of this step should be refined using an iterative, non-linear method such as bundle adjustment [79], a procedure we follow in the subsequent chapters.



(a) (b)

Figure 2.11: On close inspection, intersecting rays tend to have some degree of skew. The least squares approach looks for the point which minimizes the residuals to these rays.

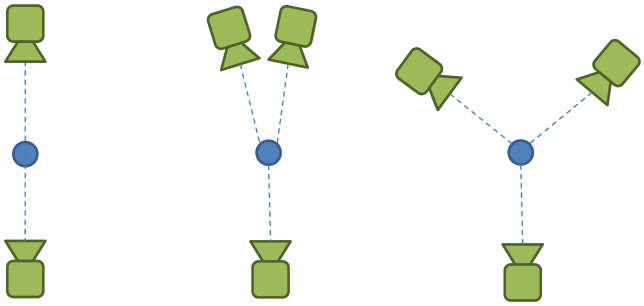


Figure 2.12: In configurations (a) the two cameras are collinear with respect to a 3D point, which cannot be reconstructed. Likewise, the cameras in configuration (b) are nearly collinear, resulting in a nearly singular $A^\top A$. Localization along the vertical axis is poor. Configuration (c) permits good measurements along both spatial dimensions.

Chapter 3

Analysis of Camera Network Parameters

3.1 Introduction

Designing efficient camera networks is a challenging task. Optimizing a network to fit an application requires the designer to meet a number of different criteria, and to choose parameters which often have performance trade-offs. Understanding the impact of these design choices is the goal of this chapter; in the next chapter we will apply this insight into an automated method for camera network design.

The considerations in creating a camera network are no different than designing any other systems namely, balancing functional requirements with design parameters. However, from a design perspective there are two complicating factors.

- Design parameters are *redundant* in that there are more parameters than there are requirements [77]. For example, precision is a function of camera resolution, position, latency, and a number of other factors. The work volume is a function of camera position, and field of view.

- Functional requirements are *coupled* in that two requirements may have opposite needs [77]. Optimizing a network to meet one criterion may be at the expense of others. An application that calls for a large work volume with overlapping views, for

		Design Parameters		
		Resolution	# Cameras	Field of View
Functional Requirements	Precision	1	1	1
	Work volume	0	1	1
	% Viewable	0	1	1
	Cost	0	1	0

Table 3.1: This matrix illustrates the influence of design parameters on functional requirements. Note that some requirements are functions of multiple parameters (redundancy) and some parameters affect multiple requirements (coupling). Adapted from [7]

example, may specify cameras with a wide field of view. An application that requires high precision meanwhile, will need cameras with narrower fields of view. For those needing both, an optimal balance must be found.

As a result of these complexities, having a clear understanding of a camera network's operating conditions is crucial. *A priori* knowledge of the application can provide additional important information. An ideal requirements specification will contain, in addition to a list of criteria, their tolerances and relative weights (See Table 3.2). Taken together this information can be combined into an objective function against which different configurations can be evaluated and ranked.

While extensive work has been done to characterize performance of individual cameras, estimating performance of multi-camera networks has only briefly been explored. The most comprehensive treatment to date (in [14]) focuses primarily on the construction of a physical testbed designed to analyze a select set of variables. While a physical simulation yields accurate results, it is cumbersome to extend to additional parameters or to rapidly evaluate a large number of candidate configurations. A software-based simulator is a promising alternative which will not only enable more rapid evaluation, but may also lower the barrier-to-entry for researchers for whom physical experiments may be impractical. Profiling multi-camera networks through simulation was explored in [39]. The authors

Functional Requirement	Value	(Tolerance)	Weight
Precision	5 cm	($\pm$ 20 %)	3
Work volume dimensions	10 m x 10 m x 4 m	($\pm$ 10%)	2
% Viewable work volume	95%	($\pm$ 5%)	1
% Not viewable due to obstruction	10%	($\pm$ 5%)	2
Real time operation	2 sec	($\pm$ 20%)	1
Cost	$10,000	($\pm$ 10%)	3

Table 3.2: An example requirement specification. Individual requirements can be aggregated in a weighted objective function to provide a means of comparing potential configurations.

develop a vision model for a specialized sterescopic camera, the Stanford MeshEye Mote [40], and examine the impact of varying parameters for a specific localization algorithm with fixed camera placement.

In this chapter we revisit these ideas by developing a camera network simulator with two goals in mind: first, by allowing arbitrary configurations we can rapidly investigate how performance is affected by varying parameters. This study will result in a list of tradeoffs providing general guidance on configuring camera networks. The simulator defines an extensible *process* rather than a particular implementation; in this way performance under different parameters and applications can be evaluated as needed. Second, the simulator will provide a development environment for testing the automatic placement and tracking algorithms presented in Chapters 4 and 5.

The topic of optimal camera network configuration raises a number of associated questions which must be addressed:

1. How is performance defined?

   The notion of "performance" is a relative measurement that depends on the context. The choice of a performance metric is as varied as the applications. Some metrics are application-specific, while others are theoretical. We discuss several types of error metrics and when they should be used.

2. To which parameters is performance most sensitive?

Given the large size of the configuration space, a sensitivity analysis is needed to understand which parameters have a significant impact on performance, and how related parameters are interdependent. A methodical sensitivity analysis would not only help designers optimize camera network configurations, but also provide decision makers with a traceable and reliable means to predict performance for mission critical applications.

3.2 Error Metrics

The concept of "performance" is a relative measurement that depends on the context. If the multi-camera network is being designed for a particular application, the most natural error metric is to use the target algorithm itself. For a surveillance system, for instance, this could be the percentage of occupants correctly identified. A system designed for manufacturing control might consider the localization error of a particular robotic manipulator.

When the particular computer vision algorithm is unknown at design-time, or when developing a more theoretical understanding, an application-agnostic error metric is needed. Here too, the designer has several options including the percentage of visible floor space (as in [42]), the number of cameras covering each point in the work volume, and the 3D point reconstruction error. Here we choose the latter metric , as it is easy to compute and is relatively analogous to common localization algorithms. This is also the approach taken by [64].

3.3 Camera Network Simulator

In order to examine the capabilities of the camera network through various configuration parameters, we created a simulation environment using Monte-Carlo techniques. The reference simulated environment consists of a work volume measuring 4 m x 4 m x 3 m observed by a network of four cameras. To ensure that the cameras are mounted in positions that could be realistically installed, they are constrained to lie on a ceiling-mounted, linear rail at a height of approximately 3m. Camera orientations are unconstrained. The sensitivity assessment will use this reference environment as a starting point to modify one design

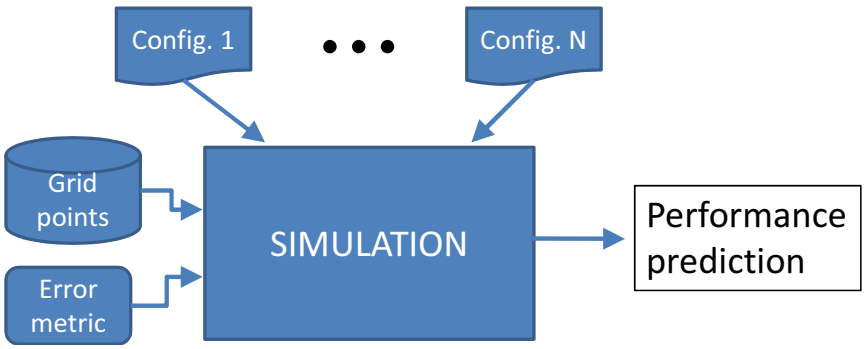


Figure 3.1: Camera network simulator block diagram

parameter at a time. For a given configuration of cameras and work volume, 2500 random 3D points are chosen within the work volume and evaluated as to the accuracy of their reconstruction by the cameras. Results are then averaged across all points for 3D localization error and work volume coverage. While straightforward and efficient, this approach has a number of limitations:

Limitations and Assumptions Multi-camera networks can assume a wide-variety of configurations. As this was a pilot study, we were only concerned with the most common examples. More esoteric setups will be left for future work. Thus the research described here is subject to the following limitations:

1. Cameras are inward facing with overlapping fields of view.

   This requirement ensures that cameras are focused on a common workspace so as to provide multiple views of the same scene. This is in contrast to a system that ensures maximum coverage of an area in exchange for minimizing overlapping views (often referred to as the "Art Gallery problem" [65]).

2. Camera orientations are static.

   In other words, the pan, tilt, and zoom parameters are set at installation time. This is known as "passive" cameras, as opposed to "active" cameras with integrated zoom lenses and pan/tilt platforms.

3. "Performance" is limited to the positional uncertainty in localizing a 3D object in the scene.

   There are many ways one could define "performance". Part of the difficulty stems from the fact that the term "performance", like the term "accuracy", is a qualitative descriptor and is thus necessarily different for each application. For this reason, the preferred measurement is "uncertainty". Furthermore, we restrict ourselves to uncertainty in measuring 3D localization of objects. While there are many tasks that can be accomplished with multi-camera networks, localization is a common enough task to provide a sufficient starting point.

4. We assume the image processing component is affected by a 2D Gaussian noise.

   In reality noise distribution and characteristics may not be purely Gaussian in nature and could be a function of the image processing algorithm.

5. All cameras are identical and are identically calibrated.

   In practice it is hard to calibrate even the same model cameras to the exact same values.

6. Lighting conditions are not a consideration in this simulation.

   Performance of image processing algorithms is highly dependent on the matching between the camera's dynamic range and scene brightness. Too much light will cause the sensor to saturate whereas too little light will introduce excessive noise.

These limitations and assumptions simplify the design of the camera network simulation while still allowing for a sensitivity analysis of key parameters, as detailed in the next section. However, nothing in this approach precludes incorporating higher fidelity models which do take into consideration these factors.

3.4 Analysis of Error Sources

Following the methodology outlined in the previous section, we conducted a series of simulation runs, each varying a single design parameter. Each run consisted of picking 2500

random points and calculating the resulting average error. The following sections describe the results of each of these simulation runs.

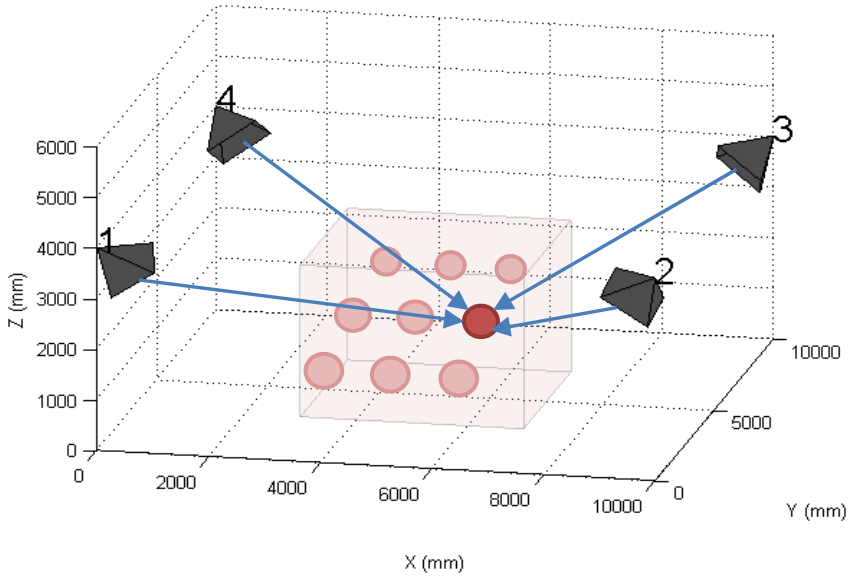


Figure 3.2: The camera network simulation environment in the reference configuration. The volume in the center delineates the work volume, which is discretized into a uniform grid of points. Uncertainty is calculated for each grid point and averaged.

Number of Cameras Several observations are evident from the results illustrated in Figure 3.3:

1. As expected, increasing the number of cameras decreases the reconstruction error; however, the marginal benefit decreases with each additional camera. The fact that each resolution has a similar decay constant suggests that this effect is stable across resolutions.

2. For each resolution, the error tends to converge towards a single value. This suggests that the ultimate performance of a camera network is limited not by the number of cameras but by resolution, as demonstrated in the next paragraph.

3. The marginal benefit decreases inversely with resolution. That is, beyond some point (640x480 pixels in this example), additional resolution provides little decrease in error. As expected, the total coverage exponentially approaches 100%. This result is dependent on both the size of the work volume and the field of view of the cameras. For similar environments we expect the function to behave similarly, albeit with a different exponential constant. Note that this result is for occlusion-free environments.

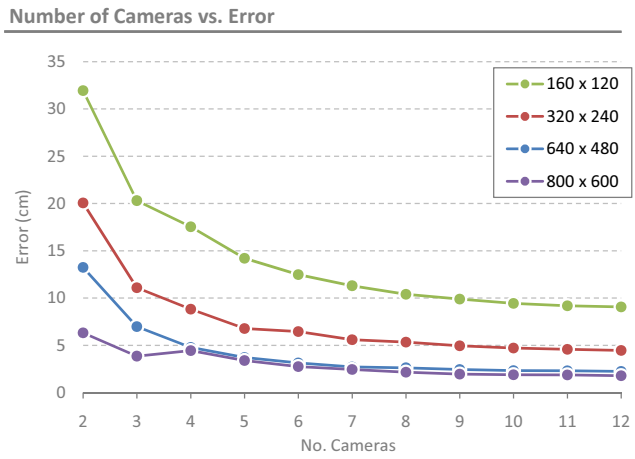


Figure 3.3: 3D point reconstruction error with varying numbers of cameras

Camera Sensor Resolution In Figure 3.4, the error decreases asymptotically as the resolution increases. The decrease in marginal benefit is outweighed by the exponential increase in cost per camera[1]. Note the small decrease in error between the 1600x1200 and 1920x1440 cases in light of nearly doubling the camera cost! For comparison, off-the-shelf webcams with a resolution of 640x480 are available for around $30 at the time of this writing.

[1]MSRP of Point Grey Research's Flea2 models at various resolutions [23]

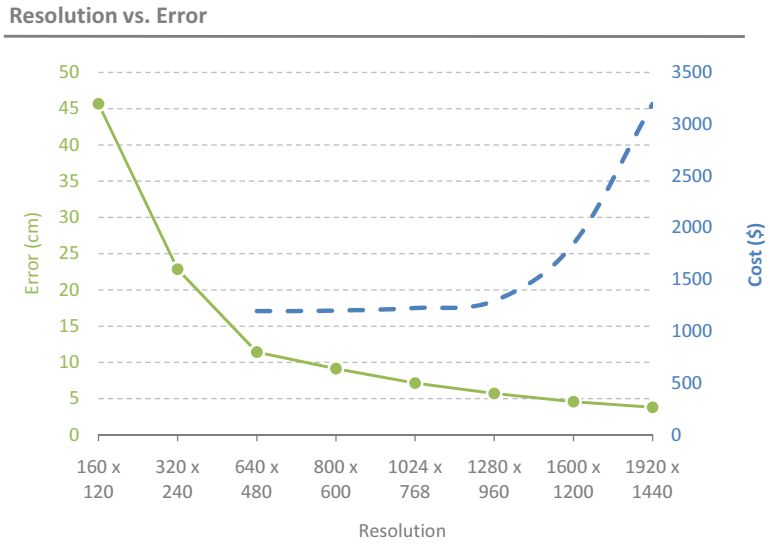


Figure 3.4: 3D point reconstruction error with varying resolution

Camera Field of View Field of view (FOV) refers to the angular extent, measured in degrees, that is observable in a single frame. It is closely related to the focal length, one of the intrinsic calibration parameters:

$$\alpha = \arctan \frac{d}{2f}$$

for field of view α, focal length f, and sensor width d (in mm) [59]. A wide field of view gives high coverage at the expense of decreased resolution. For comparison, a human's FOV is 180°, while typical binoculars might subtend 5° [2].

[2]The Hubble Space Telescope's Advanced Camera for Surveys has an extremely narrow FOV— just 0.00025°!

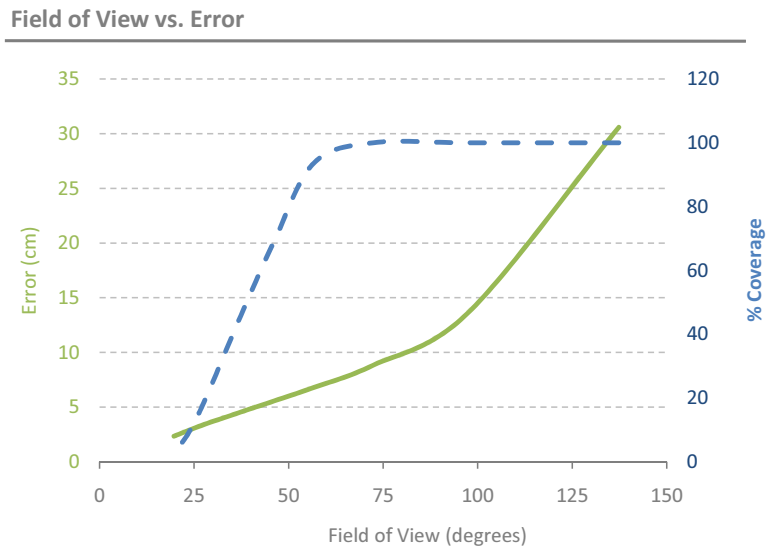


Figure 3.5: 3D point reconstruction error with varying field of view

Figure 3.5 demonstrates a FOV trade-off between error and work volume coverage. It should be noted that while error continues to increase with wider FOV, work volume coverage quickly saturates 100%, above which there is no further benefit to wider views.

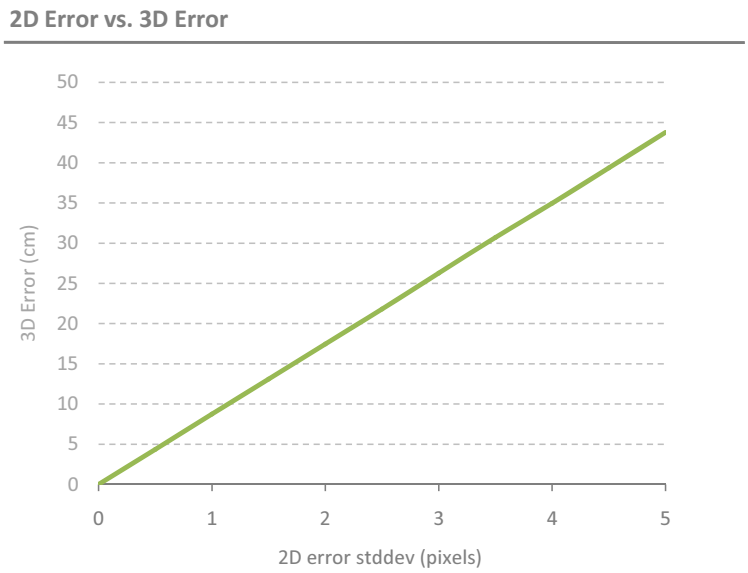


Figure 3.6: 3D point reconstruction error with varying image processing noise

Image Processing Noise Although in this work we are primarily concerned with hardware configuration, Figure 3.6 shows how significantly performance is also dependent on robust image processing. In this example, we used cameras with a resolution of 1024x768. At these settings, localizing a point to within a single pixel error (or $\approx$0.1% accuracy relative to the total image height) results in errors approaching 10 cm. Small errors in image processing can dramatically degrade performance. This error is magnified when objects are imaged from long distances.

3.5 Conclusion

In this chapter we used a simulation environment to develop insights on the impact of varying camera network parameters. Three trade-offs were identified:

- Number of cameras versus sensor resolution—while both increasing the number of cameras and increasing the resolution of each sensor are beneficial in terms of increased accuracy, when cost is factored, it is easily observed that cameras' unit cost rises linearly, while the cost of higher resolution sensors increases exponentially.
- Sensor resolution versus cost—Currently it appears that sensor resolutions over 1024 x 768 yield little advantage given the high cost of the more advanced components. However, as technology improves and prices fall, the break-even point might get pushed into higher resolution sensors in the not too distant future.
- Error versus the percentage of the work volume covered—Since work volumes can be fully covered with a relatively low number of cameras (assuming no occlusions), there is no need to accept a higher error in exchange for a wider field of view.

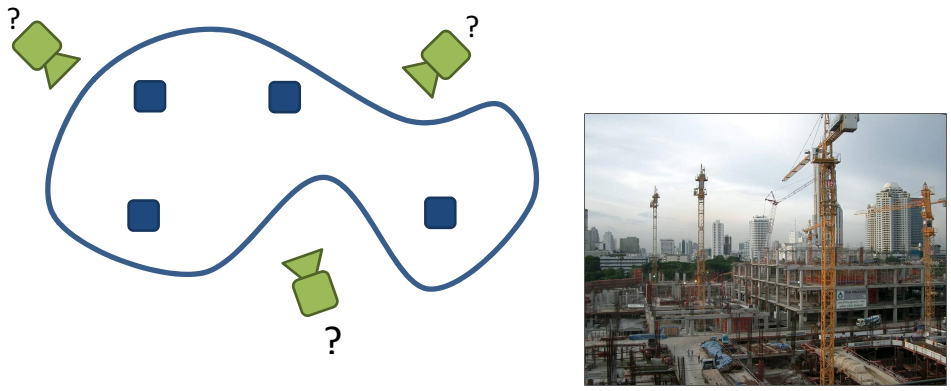
These observations are useful as qualitative guidelines but do not on their own address how to optimally configure camera networks. With an understanding of these parameters we can turn our attention towards applying them in an automated configuration tool, the topic of the next chapter.

Chapter 4

Optimal camera network configuration

The analysis in Chapter 3 highlights the complexities of optimally configuring multi-camera networks. As we concluded in the previous section, unfavorable sensor placement, low resolution, or poor fitting of any number of parameters can have detrimental effects on performance. At present, camera network configuration is primarily driven by intuition. As a result the technology has largely been confined to the laboratory. For small scale deployments, or where performance requirements are not stringent, a manual approach may suffice. As the commercialization of camera networks is still in its infancy complex environments are still the exception. For camera networks to transition from an experimental technique to a practical device, developing an automated method for optimal camera placement is crucial. Previously, we defined an optimal camera configuration as one that minimizes the average 3D reconstruction error. To be practical, however, a real system must also consider the percent of the work volume covered. In the extreme case, all cameras can be trained on a single point; such a system would be highly accurate but hardly useful! At the other extreme, the work volume can be comprehensively covered, but any individual point is only covered by a comparatively small number of cameras. As was demonstrated in the previous section, reconstruction error for a given 3D point can decrease substantially with additional observations, up to about 4-6 cameras.

For multi-camera networks to be commercially competitive an approach is needed to assist designers in setting parameters. Even with the aid of a simulation environment, this is still a difficult task. Some of the challenges include:



(a) (b)

Figure 4.1: Complex environments can arise from geometric constraints of the room or the requirements of the application. (a) In this irregular environment containing occluding columns, optimal camera placement is nontrivial. (b) Construction sites typically exhibit high degrees of visual clutter. Many occlusions from equipment and structural elements make it difficult to establish a line of sight to every point in the scene.

1. Interdependent parameters—The complex relationship between parameters means that an individual parameter can rarely be modified in isolation of the others. For example, one might want to increase the work volume by increasing the cameras' field of view. To maintain the same level of accuracy, however, the number of cameras in the network will need to increase, which in turn need to be optimally placed, and so on. Although in this work we only consider hardware parameters, software parameters need to be considered as well.

2. Designer expertise—The current state of camera network design is as much art as science. Due to the lack of a clearly defined design process, many decisions are subjective and depend on the expertise and experience of the designer. The quality of the output is the result of trade-offs such as the designer's choice between more cameras or higher resolution sensors.

3. Environment dependence—Camera placement in the presence of arbitrary occlusion presents a challenge. Because cameras are line-of-sight sensors, care must be taken

to ensure that the entire environment is adequately covered. Highly cluttered environments such as construction sites require different considerations than sparse manufacturing environments.

4. Application dependence—The configuration of the camera network must be appropriate for the precision requirements of the application. In this work we focus on localization and tracking, but other applications may require different levels of precision. Tracking a pedestrian may only require 10 cm precision, while localizing the end-effector of a robotic manipulator may demand much greater precision.

In simple cases these challenges can be easily addressed. For rectangular, obstruction-free environments a manual approach to optimal camera placement is relatively straightforward. In such environments very high accuracy can be obtained by skilled photogrammetrists: in a laboratory setting, using a network of 18 cameras, [31] reports measurement uncertainties on the order of one part per million! Outside of the laboratory, such photogrammetric techniques are impractical. Most of the literature on photogrammetry is concerned with small-scale camera networks characterized by close range observations and lack of occlusions. In the applications we consider in this work, observations are made from a distance of tens of meters. On these larger scales, occluders such as structural components, trees, or other environmental elements feature prominently and add to the complexity of the scene geometry. The varied and arbitrary environments one may encounter means that each installation must be designed independently. For a field-deployable, commodity system an approach that minimizes user interaction is necessary. An automated method would require only a description of the work volume (including any occluders) and the specifications of the cameras (number of units, field of view, and resolution).

In this chapter we implement a prototype method for automated camera placement.

4.1 Related Work

Camera calibration has been covered extensively in the computer vision literature [38, 76, 84, 92, 93]. Automating camera placement, in contrast, has only been addressed minimally. While the two problems seem superficially similar as they both deal with camera placement,

their objectives are fundamentally different. In camera calibration problems, the parameters of the camera (including its position) are fixed; the goal is to determine their values. In the camera placement problem, parameters must be chosen to optimize some global objective (for example, maximizing the viewable area for a given room configuration).

Optimizing camera placement has its genesis in the theoretical *art gallery problem* (AGP), a classic problem in computational geometry [65]. It poses the following question: given an art gallery defined as an n-sided simple polygon, what is the minimum number of guards needed to cover the entire space? AGP assumes the most basic of models: both the scene and observers are restricted to the 2D domain, and coverage is strictly binary based on a visibility computation. Unlike real cameras, resolution of the guards is assumed to be infinite. Subsequent investigations incorporate progressively higher fidelity visibility models. An early extension, described in [29], expands the problem space to 3D and assumes finite resolution. Their approach assigns unique textures to each surface in a model of the scene, renders images at hypothetical camera positions, and counts the number of visible textures.

In [42], the authors describe a mathematically optimal method based on binary (or, "0-1") integer programming (BIP), a subset of linear programming where variables are constrained to binary integers. The authors consider a number of objective functions, including one which looks for the minimum number of cameras such that every scene point is visible to at least N cameras. They later extend this model to allow for a heterogeneous camera network, at the expense of additional variables. While a binary integer programming approach delivers mathematically optimal results, modeling a realistic camera network in this way would result in a state space explosion. Nevertheless, it makes for a useful benchmark for evaluating the efficacy of more practical algorithms. BIP will be described in greater detail later in the next section, to compare against the proposed automated camera placement algorithm.

In [43], the visibility model is extended to include orientation. Rather than assuming objects of interest are rotationally symmetric, orientations are assigned to the targets as well as the cameras. This provides a realistic model for tasks such as face detection, which depend on having the target having a constrained rotation relative to the camera. As in

[42], binary integer programming is used with a greedy heuristic to cope with the high complexity.

This complexity is a common theme in the camera network literature. High dimensionality makes optimal camera network configuration a difficult problem. From a practical standpoint, the differing requirements of each application imply that there is unlikely to be a singular "best" optimization algorithm. While each of these methods are demonstratively effective for specific applications, extensibility remains a major issue. AGP has primarily been applied to theoretical problems and does not extend well to realistic camera models. BIP, which uses linear programming as a framework, is obviously constrained to linear objective functions. Many interesting objective functions require more generalizable tools. While minimization of 3D reconstruction error has been explored in [16, 87], these solutions are only suitable for a small number of cameras. [24] describes a method that incorporates a large number of cameras, but assumes that targets of interest lie on a 2D plane. An ideal approach would be robust enough to handle the complexity of a realistic problem, but flexible enough to adapt to the needs of each individual application.

Olague and Mohr [64] were the first to introduce the idea of genetic algorithms for optimal camera placement. Their prototype system, EPOCA ("Evolving Positions of Cameras"), uses an evolutionary approach to evaluate a set of camera network configurations. Over a series of generations, this set converges to a (hopefully optimum) configuration. Genetic algorithms are an elegant approach to the non-linear, high-dimensional problem in question but are particularly computationally intensive. This leads the authors to make several limiting assumptions (e.g., all cameras are equidistant, no occlusions) which reduce immediate practicality. A modern revisiting of this work with relaxed assumptions is warranted. In the following sections we explore the applicability of genetic algorithms to the camera placement problem.

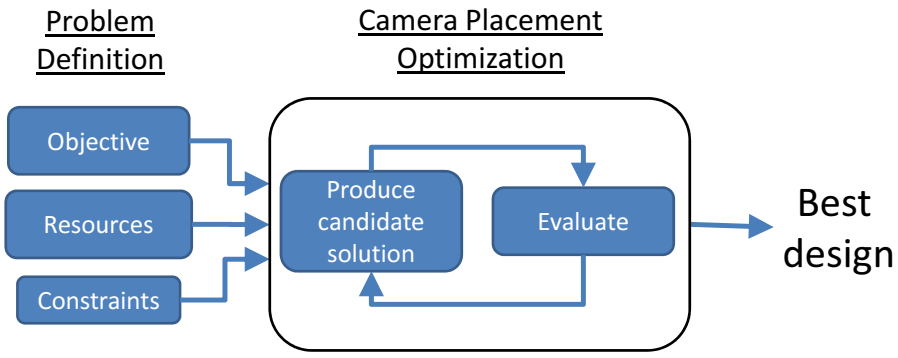


Figure 4.2: Camera placement algorithm block diagram

4.2 A Design Process for Camera Placement

Any of the approaches described above can be thought of as a black-box component in a larger design optimization framework (Figure 4.2). While the internals of these components differ across particular algorithms, their interfaces are identical. When undergoing an optimization, the designer must consider the following steps:

1. Defining the problem—The first step is to define what a "good" camera configuration looks like. More formally, this consists of choosing an appropriate objective function that allows candidate configurations to be evaluated and ranked. A number of objectives are in common use and their choice is dictated by the application in question. Objective functions can include ensuring a certain percentage of scene coverage or localizing an object within a guaranteed tolerance.

2. Identifying resources—Next, the designer must describe the components of the network. This includes descriptions of the sensors (with their associated parameters) and the scene geometry (both the shape of the room as well as the position of any occlusions).

3. Incorporating constraints—Each application may have a unique set of constraints that a potential solution must meet. Such constraints can be applied to the sensors (for example, a maximum number of cameras as determined by the project budget) or to the scene (cameras can only be placed at specific, predetermined mounting points).

4. Performing the analysis—This is the black-box component that contains the automated optimization algorithm. The algorithm takes the objective function along with resource and constraint definitions and outputs an optimized vector of camera positions and orientations.

5. Installation—Care must be taken to ensure that cameras are positioned and oriented as specified by the optimization algorithm. This can be done with the aid of ground truth devices or by observing scene points with known coordinates. Although this step is often disregarded in discussions of camera placement due to its laborious and somewhat unglamorous nature, it is critically important to the design of a precise camera network.

Using this design optimization framework we will demonstrate a new camera placement approach based on the genetic algorithm paradigm. This powerful but conceptually simple technique is adaptable and scales well with problem complexity. In following sections we will describe this approach and demonstrate its operation in simulated and empirical settings. To verify the validity of using genetic algorithms in this context we also compare it to a mathematically optimal method, binary integer programming.

4.3 An Introduction to Genetic Algorithms (GA)

Most optimization algorithms are designed for a specific class of problems. Specifically, these are problems where the objective can be described by an analytical, closed form expression whose derivative is readily available and the output is a real number (or perhaps a vector of real numbers). These optimization strategies (of which gradient descent and its variants are predominant) also impose smoothness constraints and require some knowledge of the structure of the solution. In this regard, optimization is concerned with fitting coefficients.

In contrast, there exists a large class of problems based on physical phenomena for which no analytical function exists and which must be evaluated through simulation. While

the structure of the solution is unknown, a candidate solution can be readily evaluated against a performance metric.[1]

4.3.1 The Genetic Algorithm Paradigm

A genetic algorithm is a search technique inspired by the evolutionary process of natural selection.[2] As in natural selection, GA maintains a population of candidate solutions (also termed *individuals*) which are repeatedly evaluated using an objective function over a series of generations. At each generation, the fittest individuals are selected, according to some heuristics, to seed the succeeding generation. Given a sufficiently diverse starting point, populations typically converge to an optimal solution within 50-100 generations. This approach is a subset of a class of search algorithms known as *evolutionary algorithms*, a broad class of techniques inspired by natural selection and genetics. The genetic algorithm was championed by John Holland at the University of Michigan in the 1970s [41]. Because of the demanding computation needed, the approach was slow to gain acceptance, with some researchers claiming GA reached its zenith in the 1980s. Genetic algorithms have also been criticized for their lack of theoretical background. Since the results are generated stochastically, it is said that their behavior can be unpredictable or difficult to verify. Nevertheless, a number of problem domains have been tackled with genetic algorithms including pharmaceutical design [58], financial forecasting [81], and bridge design [32].

4.3.2 Benefits

Despite the aforementioned shortcomings, genetic algorithms can excel where:

- The problem is of high dimensionality.
  For our purposes, a camera has three position (x, y, z) and three orientation parameters (α, β, γ). With another variable for field of view, and two for resolution (horizontal and vertical), each camera contributes 9 parameters.

[1] This is akin to saying "I know it when I see it", a phrase first popularized by Supreme Court Justice Potter Stewart in his 1964 ruling on Hollywood obscenity.

[2] One of the hallmarks of GA is its extensibility—there are as many flavors of GA as there are practitioners. This section serves as a brief overview of the process; more in-depth tutorials can be found in [60] and [35].

- Analytical methods are not available.
  A genetic algorithm is a *metaheuristic*. Metaheuristic optimization algorithms are those for which the objective function provides little heuristic information to aid the optimization, but when a good solution is found it can be readily identified as such [55]. Thus the objective function can be treated as a black box, which can be queried with a candidate configuration and returns a fitness score. No analytical expression is needed for the objective, as is required of many other optimization methods.

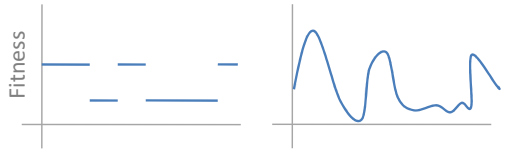
- The problem is different for each application.
  Due to the varied nature of camera network tasks, parameter requirements differ from one application to the next. In the genetic algorithm paradigm only the fitness function needs to be modified to fit a new problem.

- The solution space has many local minima.
  The genetic algorithm is a Monte Carlo approach. Each individual in the initial population of candidate solutions has a random starting point. For highly nonlinear problems this approach gives better results than gradient-based methods.

- Each application has a unique set of requirements.
  In designing a camera network, any number of constraints may be present. The designer may be limited to a fixed number or type of camera, for instance, or cameras can only be placed in specific mounting points. An ideal approach to automating the design would be able to adapt to these varying needs without fundamentally changing the analytical approach. The genetic algorithm is particularly strong in this regard: rather than describing a rigid set of steps, the genetic algorithm describes a problem solving framework where the application-specific portions are abstracted into two constructs, the encoding mechanism and the fitness function, which we define next.

4.3.3 Encoding

An encoding describes how an individual (in the genetic algorithm sense) is represented and manipulated. Typically this takes the form of a common data structure such as a vector



(a) (b)

Figure 4.3: (a) With a fitness function of low cardinality, it is difficult to find the global minimum. A better function (b) outputs values that allow any two individuals to be compared relative to each other.

or tree, where each element contains a parameter. More critical than the structure, however, is choosing an appropriate parametrization. An optimal representation is one where a small change in input yields a small change in output. In other words, individuals that are "close"[3] to each other in the representational space should behave similarly. The success of optimization largely rests on choosing an appropriate representation.

4.3.4 Fitness Function

The fitness function translates a genotype into a phenotype or, more formally, evaluates an individual using some user-defined method and returns a quality, or fitness value. When designing a fitness function, care should be taken to ensure that output values have high cardinality, that is, individuals can take on a wide variety of behaviors. A simple true/false output (typical of decision problems) gives a flat fitness landscape and makes it difficult or impossible for the population to converge (Figure 4.3). Problems of this type should be recast to provide a degree of optimality rather than a discrete label. This generally means that the output is a real number, although the fitness landscape itself need not be continuous.

The fitness function, together with a proper encoding, cleanly separate the specifics of the problem with the optimization approach itself.

[3] In an abuse of genetics terminology, some practitioners of genetic algorithms define the notion of "genotype" and "phenotype". The genotype describes how an individual is represented to the algorithm, while the phenotype describes how the individual performs (i.e., its fitness). When we say two individuals are close, we mean they have similar genotypes.

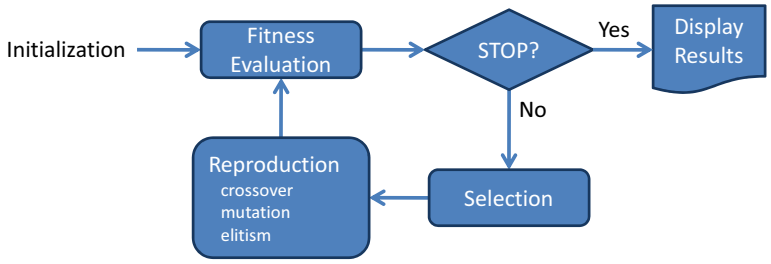


Figure 4.4: Genetic algorithm block diagram

4.3.5 Genetic Algorithm Walkthrough

An overview of this process is presented in Figure 4.4. The algorithm proceeds as follows:

Population Initialization A population of candidate solutions is generated by randomly choosing points in the solution space. The number of individuals in the population is dependent on the size of the problem as well as capability of the computational hardware, but typically consists of tens to thousands of individuals. If something is known about the problem beforehand, the initialization step could be biased to select from more favorable regions. More commonly, populations are initialized without this bias (also known as "seeding"). Although initial populations perform very poorly (due to their random nature), some will perform *slightly* better than others. Emphasizing these individuals through selective reproduction allows genetic algorithms to find a solution even when nothing is known about the solution space. A second reason for not biasing the population is because the designer's *prediction* of the solution space may be very different from the *actual* space.

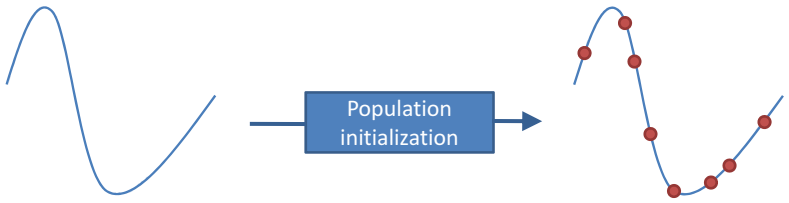


Figure 4.5

Evaluation Each individual in the population is evaluated according to an error metric and assigned a corresponding fitness score. This phase is by far the most time consuming, as this function needs to be called for all individuals at each generation. This demonstrates why computational expense is one of the primary drawbacks of genetic algorithms. Only in recent years have commodity-priced computers been sufficiently capable. The growing availability of parallel computing and multi-core processors have been a boon to the genetic algorithm community. Since each evaluation of individuals in a single generation are independent, speedup from parallel computation is nearly linear.

Figure 4.6

Selection After the individuals are ranked according to their fitness, the selection phase chooses those with better fitness to go on to the reproduction phase. Many selection operators have been proposed. The most common is k-way tournament, where k individuals are chosen at random, and the best performing gets flagged as a parent. The selection operator is repeated until a sufficient number of parents are chosen to form a child generation.

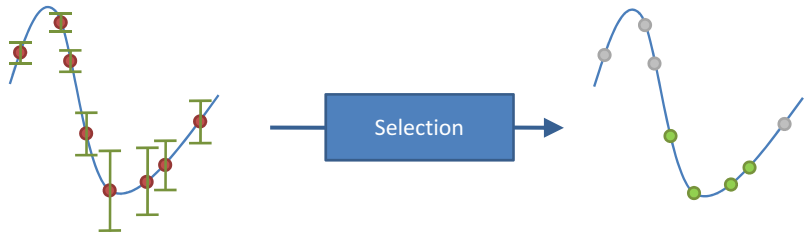


Figure 4.7

Reproduction In this stage, the parent individuals chosen in the previous step are recombinated to create a child generation. The hallmark of a reproduction operator is that the child retains some traits from the parents. Because of the large number of individuals in a population, the genetic algorithm does not need to "know" about these traits explicitly. Indeed, the algorithm designer may not predict certain beneficial traits. This highlights one of the appeals of genetic algorithms: the structure of the solution need not be known *a priori*.

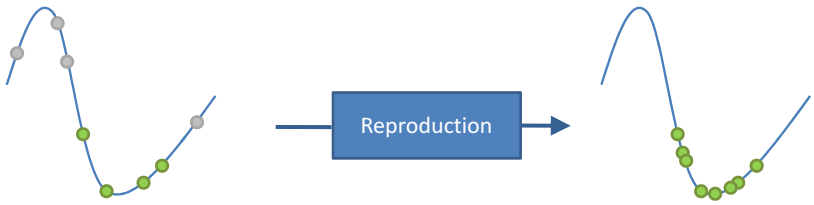


Figure 4.8

A large number of reproduction operators have been conceived; here we describe a few of the most common. Crossover, a mainstay of many evolutionary-based algorithms, takes fragments (often likened to "genes" to maintain the biological analogy) from two parents and splices them to form two offspring. Mutation modifies the parameters of a randomly selected gene from within a single parent. Elitism takes the fittest parents from the previous step and simply replicates them into the new generation, with the assumption that very fit individuals are worth keeping. Finally, individuals not selected as parents are replaced with a new, random offspring. These operators are illustrated in Table 4.1.

The evaluation, selection, and reproduction stages are repeated for either a fixed number of generations, or until no further improvement is noted. This usually occurs within 50-200 generations, with more complex problems falling into the higher range. A successful run will show an average fitness (measured across the entire population) that improves gradually. The score of the fittest individual will improve monotonically in a stair-step pattern, since a new "best" individual only occurs every few generations. After the run is complete, a gradient descent algorithm is applied to ensure that the solution falls into the nearest local minimum.

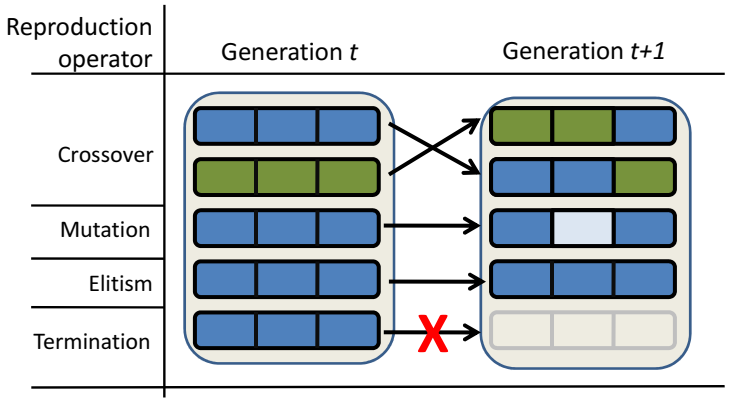


Table 4.1: Common reproduction operators

4.4 Validating the Genetic Algorithm Approach

In order to validate the efficacy of genetic algorithms in this domain we compare their performance to that of Binary Integer Programming (BIP), a mathematically optimal approach. We begin by briefly describing the BIP framework, followed by an application of BIP and GA to a simple problem.

4.4.1 Binary Integer Programming

As in the Art Gallery Problem, BIP begins the problem formulation by defining the space being observed as a 2D polygon.[4] A point on this space is considered *covered* if it can be observed by at least N cameras. If the target application is surveillance, N can be set to one. In our example we will set $N = 2$, the minimum number of camera observations needed to reconstruct an object's position. With these definitions we can consider the following objective:

> Given a rectangular space and a camera model with specific parameters, find the minimum number of cameras (and their poses) needed to cover every point in the space (for $N = 2$).

[4]For the purposes of example this space is square.

The camera model is defined simply as a triangular region with a field of view of α degrees and depth of field d. The camera pose consists of a 2D position, c_x, c_y, and an orientation, ϕ. Any point x, y in the space is observed by a camera with parameters α,d,c_x,c_y, and ϕ if the following inequalities hold (from [42]):

$$\begin{aligned}
\cos(\phi)\cdot(x-c_x)+\sin(\phi)\cdot(y-c_y) &\leq d \\
-\sin(\phi)\cdot(x-c_x)+\cos(\phi)\cdot(y-c_y) &\leq \frac{a}{2d}\cdot(\cos(\phi)\cdot(x-c_x)+\sin(\phi)\cdot(y-c_y)) \\
-\sin(\phi)\cdot(x-c_x)+\cos(\phi)\cdot(y-c_y) &\geq -\frac{a}{2d}\cdot(\cos(\phi)\cdot(x-c_x)+\sin(\phi)\cdot(y-c_y))
\end{aligned}$$

Due to the numerical nature of this approach, the ordinarily continuous space must be discretized into a grid of points according to a sampling frequency (s_x, s_y). Only points on this grid are considered for the purposes of evaluation. Note that this method will converge on the continuous case as $s_x, s_y \to \infty$. Binary integer programming problems take the form of

$$\min_x f^\top x \text{ such that } \begin{cases} A\cdot x \leq b, \\ Aeq\cdot x = beq \end{cases} \quad x \text{ is binary}$$

for binary solution vector x, and the (not necessarily binary) vectors f, b, and beq, and matrices A, and Aeq. To cast our problem in this form we define two additional variables:

$$\begin{aligned}
x_{ij\phi} &= \begin{cases} 1 & \text{if a camera is placed at grid point } (i,j) \text{ with orientation } \phi \\ 0 & \text{otherwise} \end{cases} \\
c_{i_1 j_2 \phi, i_2 j_2} &= \begin{cases} 1 & \text{if camera } c_{i_1 j_1 \phi} \text{ covers grid point } (i_2, j_2) \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

The problem then becomes to minimize the function

$$C = \sum_{\varphi}\sum_{i}\sum_{j} x_{ij\varphi}$$

subject to

$$\sum_{\varphi}\sum_{i_1}\sum_{j_1} c_{i_1 j_2 \phi, i_2 j_2} \cdot x_{i_1 j_1 \varphi} \geq N \; \forall i_2, j_2$$

The number of variables in this approach grows very quickly with the complexity of the problem. For example, assuming a modest 10×10 square grid (i.e., 100 grid points), and cameras can only be placed on the perimeter (38 points) with one of 10 possible poses, x is a vector with $38 \cdot 10 = 380$ elements, and c is a 100×380 matrix. The benefit to this approach is that, given sufficient time, it is capable of returning the optimal solution.

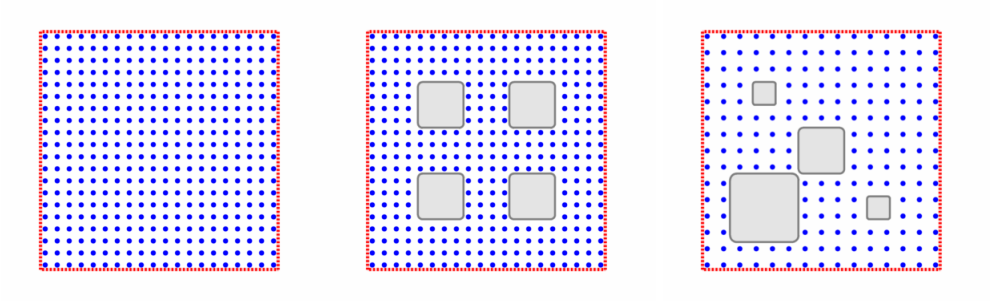
4.4.2 Comparing BIP and GA

We apply the BIP framework to the problem of finding an optimal camera placement in a simulated 2D scene. The scene consists of a square grid with camera mounting points along the perimeter. Three scenarios were tested (see Figure 4.9): an unoccluded room, a room with regular occlusions (simulating structural columns), and random occlusions (typical of occupants moving throughout the space). In MATLAB, BIP was implemented as described in subsection 4.4.1 and run on each of the three scenarios. Computation time is a function of scene complexity, and ranged from about a second for the unoccluded scenario, to several minutes from the scenario with random occlusions.

To fit the problem to the GA approach, we begin by choosing an appropriate encoding mechanism. In this case encoding is straightforward: candidate solutions are represented as a vector of pairs, with one pair representing an individual camera. A pair consists of real values P_i, describing the camera's parametrized position along the perimeter of the scene, and α_i, the camera's angle of rotation. Thus one vector represents an entire camera network. A population of 1000 such vectors is generated, a number empirically determined to balance population diversity

P_1	α_1	$\cdots$	P_i	α_i	$\cdots$	P_N	α_N

Figure 4.10: A candidate solution consists of a vector of $\{P, \alpha\}$ pairs



(a) No occlusions (b) Regular occlusions (c) Random occlusions

Figure 4.9: To verify the efficacy of the genetic algorithm approach, we compared it to binary integer programming in three simulated environments. Blue points represent regions to be observed by the camera network, while red points along the perimeter represent potential camera mounting points.

with computational time. As in BIP, the objective function is to find the camera placement that maximizes the number of observations at each grid point.

The resulting optimizations for BIP and GA are shown in table 4.2. Note the visual similarities between the two methods, particularly for the scene with no occlusions. As scene complexity increases, the two methods return somewhat varied results, although the gross structure remains similar. Presumably this is a result of the fact that in scenes with higher complexity, the objective function has many local minima which lie close to the global minimum. As is typically found in GA problems, increased problem complexity requires a larger initial population to ensure sufficient diversity. Although a population size of 1000 was chosen for the sake of example, larger population sizes are likely to yield results closer to the optimal solution.

Although not explicitly optimized for, the average number of observations per point can serve as a useful objective basis for comparison (table 4.3). Again, BIP and GA give identical results for the scene with no occlusions, with progressively divergent results as scene complexity increases.

If care is taken to ensure a sufficiently large population, GA can deliver competitive results to the mathematically optimal BIP.

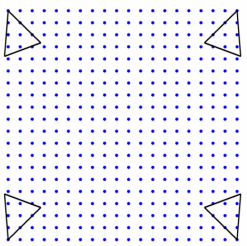
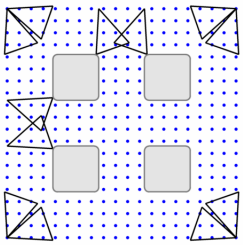
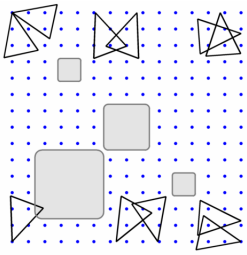
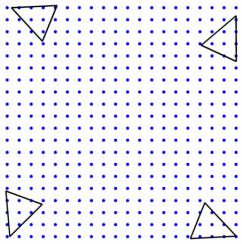
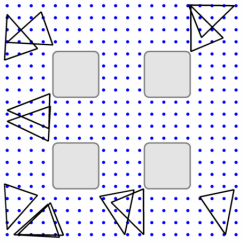
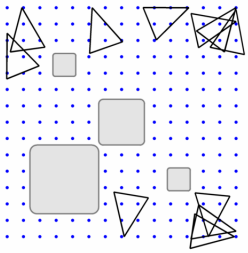
	No occlusion	Regular occlusion	Random occlusion
BIP			
GA			

Table 4.2: Comparison of GA and BIP in different scenes

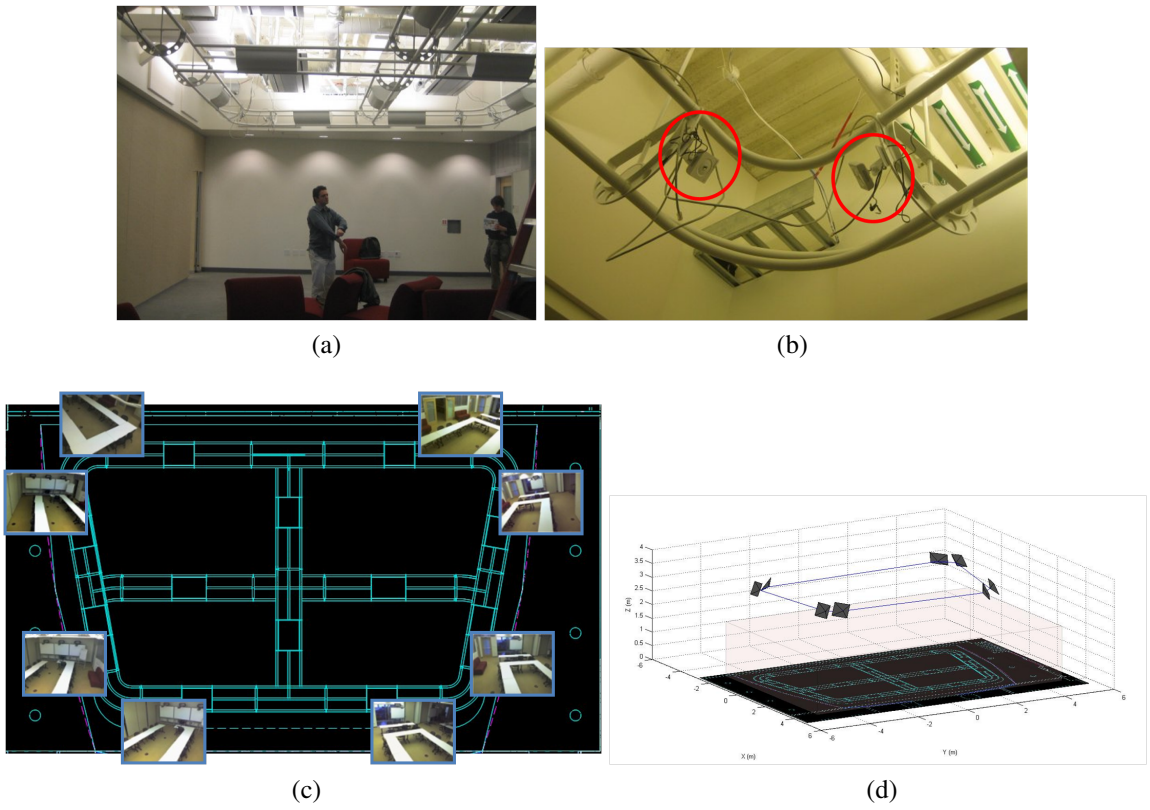
	No occlusion	Regular occlusion	Random occlusion
BIP	2.11	2.76	3.46
GA	2.10	2.68	2.80
# cameras	4	12	11

Table 4.3: Average number of observations per evaluation point

4.5 Camera Placement with Genetic Algorithms

We now consider the application of GA to more realistic problems on the scale one would typically encounter in the field. This differs from the previous example in two ways. First, we extend the 2D square grid to three dimensions, with a corresponding increase in the number of parameters (e.g., the angle parameter α in the previous example is replaced with three angles α, β, γ, corresponding to roll, pitch, and yaw, respectively). Second, whereas in the previous example we were limited to linear objective functions (due to the BIP framework), in this section we consider application-specific nonlinear objectives.

Our problem is motivated by a camera network installed at Stanford University's iRoom, a multi-purpose collaborative space in the Center for Integrated Facility Engineering (Figure 4.11). The task is to localize occupants within the room. Consequently, we seek to find a position and orientation of eight cameras which minimizes reconstruction of a 3D point in the work volume, a rectangular volume approximately 12 m x 7 m x 2 m. To provide power, data, and structural support, the cameras are constrained to a ceiling-mounted truss that encircles the work volume. The hardware consists of eight 640 x 480 resolution Panasonic (model #BL-C1) IP cameras.



(a) (b)

(c) (d)

Figure 4.11: Our task is to find an optimal configuration for a camera network mounted in an indoor environment. (a) The iRoom during construction. Note the ceiling mounted truss which constrains potential mounting points. (b) Two of the eight cameras installed on the truss. (c) A blueprint of the iRoom showing the truss section, with camera views placed at the position of the corresponding cameras. (d) The iRoom recreated in a simulation environment.

A simulation environment was prepared in MATLAB that recreates virtually the iRoom work volume (Figure 4.11d). In this virtual environment a camera network can be configured arbitrarily and the resulting projection of 3D points onto the image planes can be estimated. This simulation extends the one described in the previous chapter by the addition of a custom-designed genetic algorithm library. The following sections describe how the genetic algorithm was implemented for this task.

Initialization In our representation, the configuration parameters for an entire camera network are treated as a single individual, consisting of a vector of camera structures (Figure 4.12). In this experiment, the parameters to optimize over are the positions and orientations of each camera, with all other parameters being fixed. Furthermore, to ensure realistic mounting points, cameras were constrained to lie along a virtual ceiling-mounted rail that encloses the upper perimeter of the work volume. The initial population consists of 300 such randomly initialized eight-camera-long vectors. With six parameters (three rotation and three translation) and eight cameras, each individual is a 48-dimensional vector.

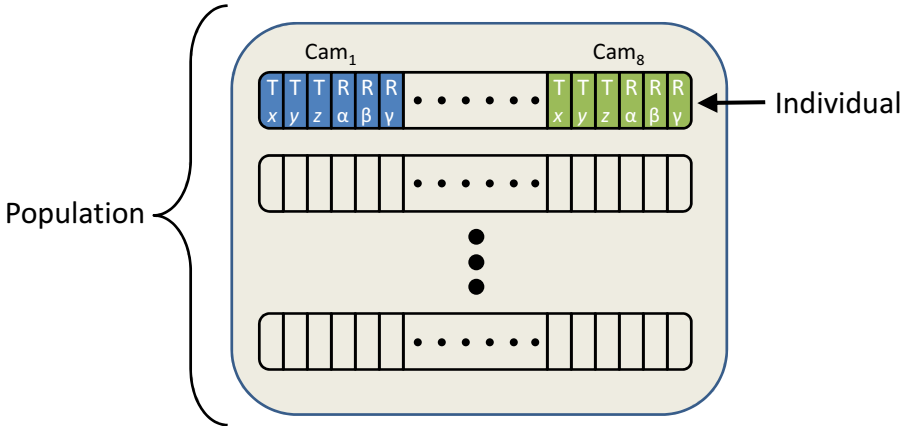


Figure 4.12: A population of camera network configurations. In the genetic algorithm paradigm, an individual (sometimes called a "chromosome") is representation of a candidate solution encoded into a vector of real numbers. In our experiment, one individual represents one possible camera network configuration, including translation and rotation parameters for each camera.

Evaluation To evaluate an individual, the work volume is first subdivided into a uniform 3D grid of voxels. Each voxel center, in turn, is reprojected into the image plane of each camera. A new 3D point is then reconstructed from these image points (as described in chapter 2) and compared to the voxel center. The fitness value is the average of the Euclidean distances between the voxel centers and their reconstructions:

$$f(i) = \frac{1}{n_x}\frac{1}{n_y}\frac{1}{n_z}\sum_{x=1}^{s_x}\sum_{y=1}^{s_y}\sum_{z=1}^{s_z}\mathbf{m_{xyz}} - \hat{\mathbf{m}}_{(xyz,i)} \tag{4.1}$$

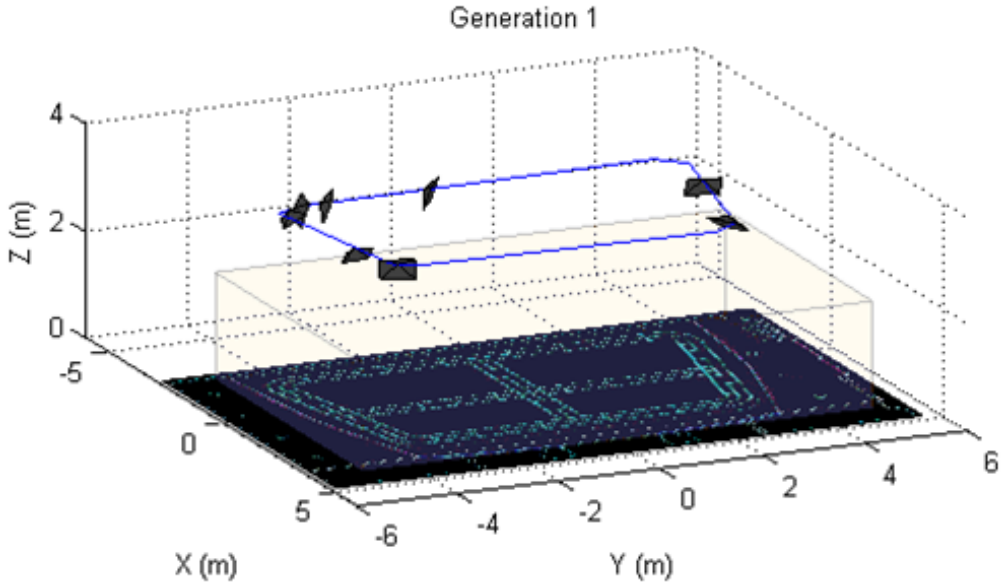
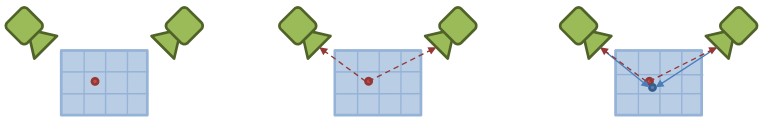


Figure 4.13: A member from the initial population. Note the random distribution and orientations. The rest of the individuals in this generation look characteristically similar.



(a) (b) (c)

Figure 4.14: A 2D schematic of the evaluation function. Select a voxel from the grid (a), reproject into each camera (b), and reconstruct the point (c). The error is the Euclidean distance between these points.

Algorithm 4.1 Crossover operator. Adapted from [55]

$\alpha \leftarrow$ `select first parent from population`
$\beta \leftarrow$ `select second parent from population`
$c \leftarrow$ `RAND`$(1 \cdots L)$ `for vector length` L

$\bar{\alpha} \leftarrow \alpha$
$\bar{\beta} \leftarrow \beta$
`for` i `from` c `to` L `do`

$\bar{\alpha}_i, \bar{\beta}_i =$ `swap`(α_i, β_i)

`return` $\bar{\alpha}, \bar{\beta}$

for individual i , 3D point m at voxel coordinate (x, y, z), and the corresponding reconstructed point $\hat{\mathbf{m}}_{(xyz,i)}$. In this experiment, 2500 voxel points were evaluated per individual.

Selection was done using 7-way tournament selection. The reproduction operators used were crossover and mutation, with probabilities of 90% and 10% respectively. Selection, evaluation, and reproduction were iterated for 50 generations.

4.6 Results

The genetic algorithm was run on a PC (2.0 GHz AMD Turion, 2GB RAM) with each generation taking one minute, for a total approximate runtime of one hour. Over 90% of the simulation time was taken by the evaluation phase.

Algorithm 4.2 Mutation operator.

```
α ← select parent from population
c ← RAND(0···1)
if c ≥ 1/2              // Mutate position
   d ← RAND(0···10)  // Amount to move along truss, in cm
   α ← MOVE(α,d)
else                    // Mutate orientation
   d ← RAND(0···10)  // Amount to rotate, in degrees
   α ← ROTATE(α,d)
end
return α
```

The result of the simulation is shown in Figure 4.15. The best performing individual in the initial population of 300 had a localization error of 1.73 cm. In other words, an arbitrary point in the work volume could, on average, be localized to within 1.73 cm in Euclidean distance. After a run of 50 generations (terminated after about 10 generations with no significant improvement), the localization error converged to **0.65 cm**. Following the best configuration with gradient descent showed no further improvement, suggesting that the genetic algorithm can effectively exploit local minima. Given the size of the work volume (12 m x 7 m x 2 m), this represents an error of less than 1 part in 1,000. A diagram of the best performing configuration is shown in Figure 4.16. Over the course of the algorithm run, an interesting emergent behavior was observed: camera network configurations tended to distribute uniformly and orient themselves so as to maximize the amount of overlap. For the simple, occlusion free environment recreated in this simulation, this seems like a reasonable evolution.

These results are quite promising; however, a few caveats should be noted. Our example assumes perfect image processing and that any arbitrary point in the work volume can be identified on the image plane to within one pixel. For experiments using known geometric objects this is approachable, but in typical deployment scenarios (e.g. pedestrian tracking) some degree of error due to image processing is unavoidable. When the target of interest is a complex object the error will likely be higher, however we expect that the level of improvement given by the genetic algorithm should be proportional. Finally, it should be

noted that due to the difficulty of precision on-site installation, this particular experiment was only carried out in simulation. An empirical demonstration of this technique (under laboratory conditions) is described in the next section.

We also compared the results against those from a hand-designed network, which resulted in an error of 3.67 cm. The large discrepancy between the manual and automated methods highlights a reality of physical hardware that to our knowledge has not been discussed in the literature. The hand-designed network was installed on real hardware, and the resulting calibration data was plugged into the simulation. Because of the difficulties associated with precisely aligning real cameras to high angular tolerances, our actual rotation uncertainty was on the order of several degrees. This undoubtedly introduced additional margin of error.

4.7 Empirical Verification

To verify the correctness of the simulation, an empirical study was carried out to measure the performance of a camera network with known configuration. At a facility at the US Dept. of Commerce National Institute of Standards and Technology (NIST), a camera network was installed and tasked with localizing a series of surveyed 3D points [45]. The system consisted of four cameras (Point Grey Research Flea2) with a resolution of 1024×768 pixels. We positioned and oriented the cameras manually in a rectangle of roughly 12 m $\times$ 12 m so as to maximize overlapping coverage. Finally, we mounted the cameras at a height of 3 m, providing a similar work volume to that in the simulation.

A set of 24 points at two different elevations (0 m and 1.2 m) were selected in the work volume and retroreflective targets were placed at each position. The 3D position of these targets was determined by a total station (Leica TPS1200+), a laser surveying device commonly used in construction applications. The model of total station used provides 2 mm spatial uncertainty, an estimated order of magnitude better performance than the camera network. Finally, the pixel position of the reflectors in each camera were manually identified by visual inspection of the images.

The same error metric described in the previous section was applied; each 3D point was reconstructed from its corresponding image points, and the result compared to the ground

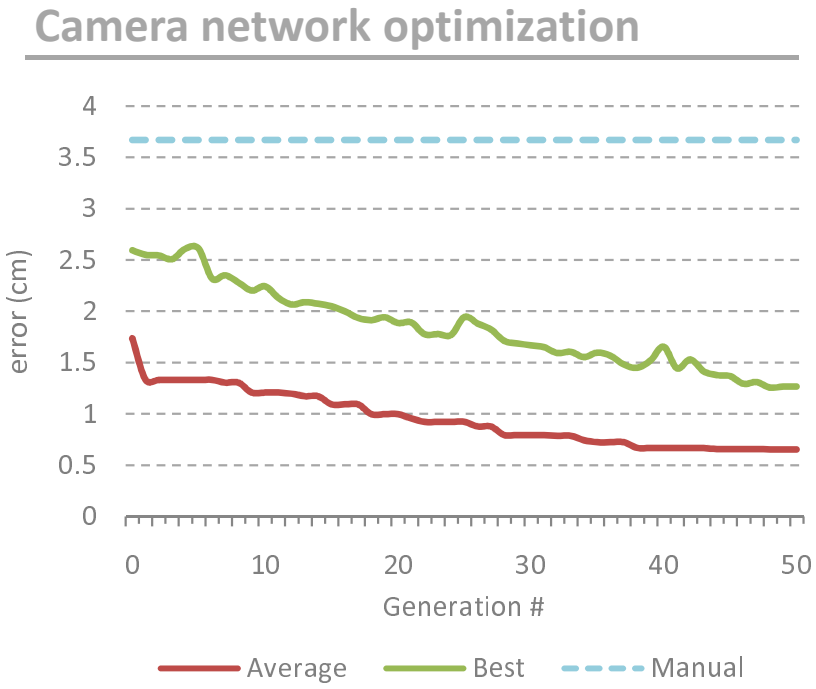


Figure 4.15: The results of running the genetic algorithm over a period of 50 generations. The green line represents the average 3D reconstruction error (in cm) across an entire population, while the red line represents the best performing individual for that generation. For comparison, the dashed blue line represents the performance of a hand-designed camera network.

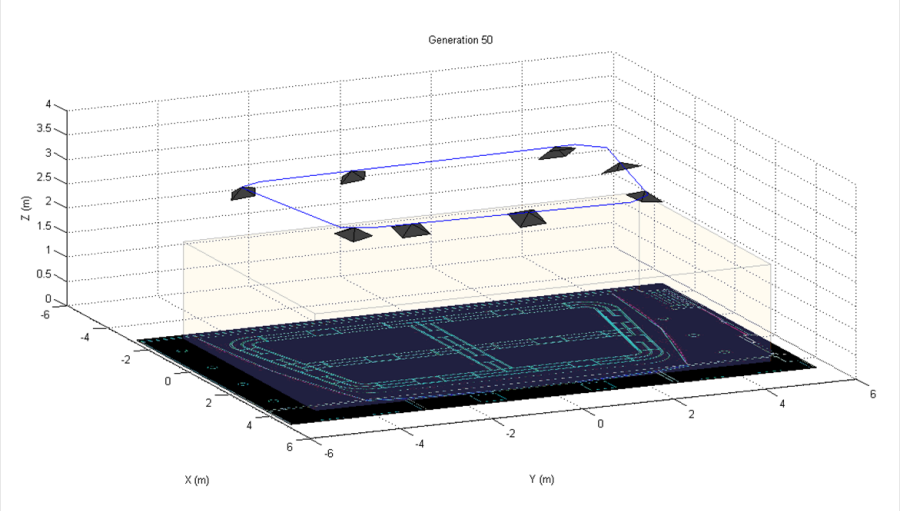


Figure 4.16: The test environment used in the simulation is a 12 m x 7 m x 2 m work volume (red cube) with a linear rail suspended at 3m on which cameras can be mounted. This diagram depicts the best performing camera network across the entire run (with an error of 0.65 cm). Note the near uniform distribution of cameras around the rail. Also, all camera are pointed nearly downward, to the effect of maximizing overlap in the viewable volume.

(a) (b)

Figure 4.17: (a) The Reconfigurable Camera Network Testbed at NIST prior to installation. The checkerboard in the center is a target for intrinsic calibration. (b) A view from one of the cameras after installation. On top of the tripod (center) is a retroreflector for use with a laser-based surveying device.

# Cameras	Trial 1		Trial 2		Trial 3	
	Error (cm)	Std dev (cm)	Error	Std dev	Error	Std dev
2	3.89	4.81	3.21	1.80	3.12	4.08
3	2.90	2.08	1.95	1.07	2.16	2.09
4	**2.40**	1.19	**1.54**	1.12	**1.74**	1.25

Table 4.4: 3D reconstruction results

truth provided by the total station. The resulting difference was averaged across all 24 points. The experiment was repeated 3 times. Results are shown in Table 4.4.

Several trends are evident from the results. First, as the number of cameras increases, mean error tends to decrease. This agrees well with the analysis in Chapter 3. Second, the error from four cameras (which across three trials averages to **1.89 cm**) is close to the average error evolved by the genetic algorithm simulation (**1.26 cm**). The slight difference is due to noise in identifying target points in the image. Since the simulation assumed perfect image processing, this additional error was not considered. Still, these results are sufficiently in agreement to verify the fidelity of the camera network simulation model.

4.8 Discussion

While the results of the simulated and empirical experiments are promising, there exist a number of avenues for potential improvement:

- In our experiment we considered localization of a 3D point as the optimization criterion. A more realistic task might to be optimize two objects simultaneously, cost and performance for instance. In this way a designer could generate a cost/performance Pareto frontier showing the optimal configurations at various price points (Fig 4.18).

- Our experiment considered all points in the work volume with equal weight. In some applications it may be desirable to require higher performance in some regions.

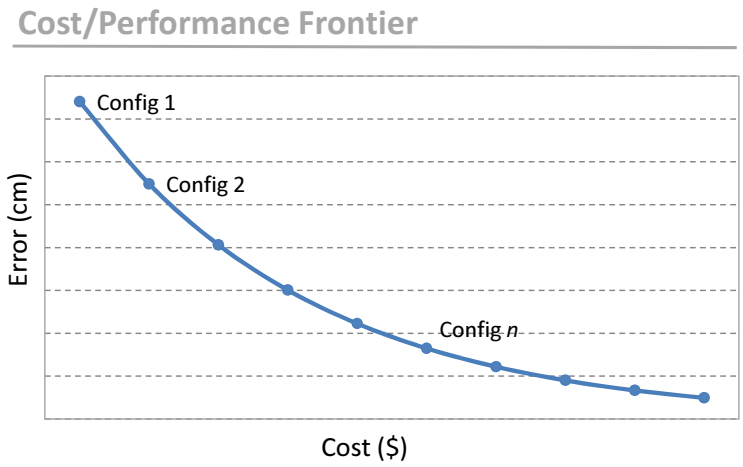


Figure 4.18: A theoretical Pareto curve which gives optimal configurations at various price points.

A manufacturing task may require high performance for tracking manipulators, but lower performance for tracking human operators.

- If the specific computer vision task is known *a priori*, more sophisticated error functions can be incorporated. A camera network could be directly optimized for say, identifying faces in a crowd, or tracking the end effector of a manipulator.

In this chapter we emphasized automation in the design phase. Of equal importance is developing a method for efficient *installation*. The usual approach to installing a camera network is to position and orient the cameras, and then determine their coordinates through calibration. The optimization approach described here requires the opposite. Position and orientation are determined first, followed by installation with the prespecified calibration parameters. In practice, accurate installation requires expensive surveying devices similar to the type used in the empirical verification. Most camera network installations, lacking these facilities, rely on manual placement, with a resulting position and orientation error measured in tens of centimeters and degrees. Clearly, for optimization to deliver its potential, a method needs to be developed for designers to install cameras accordingly, with a minimal amount of external metrology equipment. One potential solution is to mount

cameras on pan/tilt units and perform autocalibration. While this raises total system cost, the pan/tilt capability adds flexibility for potential future tasks.

Another solution which we leave for future work is to incorporate an error model for installation directly in the optimization algorithm. While the resulting configuration may not be theoretically optimal, it will be better able to predict performance after installation. One approach is to precompute error models corresponding to various levels of installation resources. A simple tape measure, for example, can be used to determine the position of calibration targets with several centimeters of uncertainty (for a room-sized system). Lower uncertainty can be achieved with more precise (and costlier) instruments such as a total station, but this likely precludes installation by the typical end-user[5]. Using a noise model may not overcome the installation problem, but at the least could give the operator a better understanding of the camera network's capabilities and limitations when installed using a particular surveying device.

4.9 Summary

At the present time, placement of cameras (position and orientation) is primarily driven by intuition. For small scale, experimental deployments or for environments that are relatively straightforward this approach may suffice. As the commercialization of camera networks is still in its infancy, complex environments are still the exception. For camera networks to transition from an experimental technique to a practical device, developing methods for optimal camera placement is crucial.

In this chapter we introduced a novel approach to the camera placement problem using genetic algorithms. Through simulation and empirical verification we demonstrated a substantial improvement in localization error over a hand-designed camera network. Furthermore, since automated camera placement tools are used at design-time (and therefore is performed only once), this tool represent a straightforward and relatively low cost method to improve performance.

[5] With current total station prices starting at $5000 [66], the cost of the installation device may exceed the cost of the system it is installing!

Chapter 5

Object Tracking for Multi-Camera Networks

Until now we have examined camera networks with abstract quantities: localizing points, maximizing coverage, and so on. These metrics serve as a useful foundation for theoretical analysis since they are easy to define mathematically. To be practical, however, we must also consider the application level. Since the early days of computer vision research, object tracking has been particularly well studied as it serves as a foundation for many important applications (Figure 5.1).

In this chapter we describe the object tracking pipeline and some common implementations. While tracking has traditionally been the domain of single cameras, extending these methods to multiple cameras offers new possibilities for improving robustness and accuracy. However, the added complexity of coordinating multiple, time-critical observations can also introduce new challenges. Care must be taken to ensure that these observations

- Crowd surveillance [25]
- Traffic monitoring [5]
- Human-computer interaction [20]
- Motion capture [78]
- Manufacturing automation [33]
- Vehicle navigation [82]
- Sporting event refereeing [69]
- Behavior recognition [57]

Figure 5.1: Multi-camera tracking applications

are combined effectively. Overcoming these challenges requires a new approach to tracking, one that treats the multi-camera network as a monolithic sensing device rather than a discrete collection of cameras. We propose a novel method for 3D tracking using multiple, calibrated camera views. We compare this approach, which we call Predictive 3D Tracking, with several existing methods and demonstrate its effectiveness in both simulated and empirical settings.

5.1 Introduction

To motivate our proposed approach we consider a case study of tracking workers on a construction site. The ability to automatically track personnel could have significant implications for project safety and productivity. Consider the following use cases:

- Workers operating under crane hooks or behind heavy machinery are susceptible to work site accidents. A robust tracking capable of localizing workers (and associated equipment) could provide real-time incident detection and mitigation.
- On large scale sites, personnel transportation time between work stations takes significant time. By analyzing the distribution of workers over time, project managers could optimize site layout to reduce this time.
- Understanding how personnel distribute themselves throughout a work site could allow foremen to optimize resource utilization.

Several radio-based technologies have been suggested to address this topic, including radio-frequency identification (RFID) [27] and ultra-wideband (UWB) [91], but these have the disadvantage of requiring personnel to carry tracking tokens. Token-based (or "active") tracking requires a higher degree of compliance from each worker and may raise privacy concerns. Additionally, large numbers of transient personnel make the distribution and management of an inventory of tokens impractical.

With the growing ubiquity of cameras and increasing computational power, video has also been explored as a possible sensor. Cameras are low-cost, familiar, and being passive devices they require no intervention by the personnel being tracked. Tracking using

camera images is not novel; while the problem has long been a mainstay of the computer vision literature, its application to the construction domain provides unique challenges. The high degree of visual clutter that is typical of construction sites, as well as the constantly evolving environment, leads to a deployment-ready tracking system whose requirements are substantially different than those of a laboratory prototype.

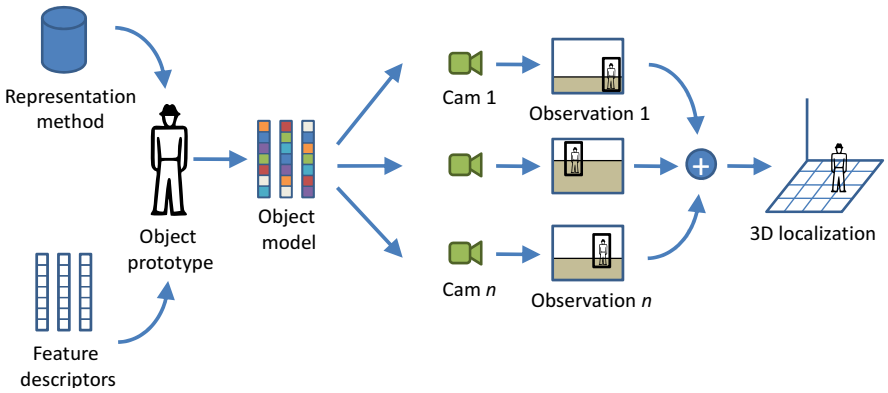
5.2 The Object Tracking Pipeline

While computer vision-based tracking has been a topic of research for a long time and has continuously maintained a strong presence in the literature, it still remains a difficult problem to solve. Many variations have been proposed, owing to the diversity of problem domains. In this section we highlight some state-of-the-art methods and recent trends, starting with single camera implementations[1] and continuing with multi-camera extensions.

The fundamental goal of object tracking is to localize and estimate the trajectory of an object, either in the image plane (2D), or in space (3D). All methods must answer two questions: How should an object be represented? What image features define the object? Additionally, multi-camera variants must consider how multiple observations should be combined.

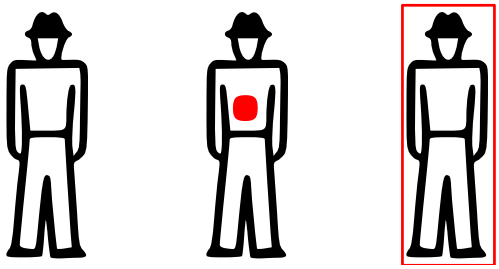
The choice of representation defines how an object is described. Our ultimate goal is a 2D or 3D coordinate. However, because of the diversity and complex geometry of the objects one may encounter, a single coordinate is rarely sufficient to provide sufficient robustness. For a highly articulated structure such as a human figure, what does it mean to say a person is at some coordinate? Choosing an appropriate representation depends on the needs of the application in question: for some a gross position is sufficient, others may need the position of each articulated joint. The simplest representation is a single *point* [83], typically the object centroid. *Geometric primitives* [17] such as a bounding box encodes more information (e.g., scale) and are useful for rigid objects. More complex representations, suitable for tracking humans, include *skeletonizations* [4], *contours* [18], and *silhouette*s [21].

[1] adapted from *Object Tracking: A Survey* [89]

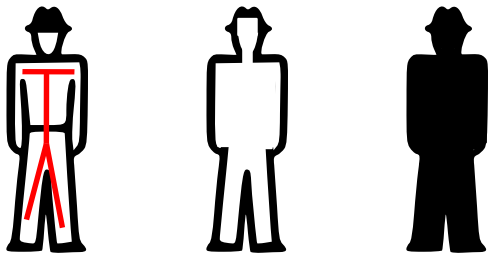


(a) (b)

Figure 5.2: The object tracking pipeline. (a) First, a model is built by applying a representation and descriptor to the object of interest. This step is typically done offline. (b) Next, In the tracking phase the previously learned model is applied to each camera observation, and the resulting 2D localizations are combined into a 3D localization estimate.



(a) (b) Centroid (c) Bounding box



(d) Skeletonization (e) Contour (f) Silhouette

Figure 5.3: Various object representation methods

After choosing a representation, the next component of a tracking algorithm is selecting a visual feature. These are closely related to the representation in that the latter determines which image locations are sampled, and the former describes what image property is recorded. An effective feature should encode some unique characteristic of the local neighborhood which represents the object being tracked. These building blocks of computer vision can include simple features such as color, edges, or corners. More abstract features can encode texture, or histograms of local pixel colors. Many simple features are sensitive to variation in the scale, orientation, or lighting of the object resulting in missed detections. Modern feature descriptors that are robust to these changes include the Scale-invariant Feature Transform (SIFT) [54] and Speeded Up Robust Features (SURF) [6]. For objects with a great deal of structure (a human face, for instance) these newer features and their variants have become predominant. Simpler objects (such as geometric primitives) can effectively be tracked with correspondingly simpler features.

If the application only calls for 2D tracking in the image plane, the choice of representation and visual feature are the two primary considerations. If a 3D coordinate is desired, there needs to be an additional mechanism to combine multiple observations. Generally speaking, there are three classes of tracking algorithms as described below:

1. Planar methods [74, 8, 73, 67]—If the object being tracked is restricted to a plane (vehicles, for instance, or pedestrians), a planar homography can be established between the image plane and the scene (Figure 5.4). This homography takes the form of a 3×3 matrix $\mathbf{H}$ which translate between an image point x_i and scene point X_i:

$$X = \lambda \mathbf{H} x$$

   for scale factor λ. This class of algorithm requires some *a priori* information about the scene where the tracked object resides—deriving $\mathbf{H}$ requires that the scene be surveyed and that the coordinates of at least 4 points are measured.

2. Triangulation [86, 28]—In this method the object is first localized independently in each 2D image plane. Next, a ray from each camera i's focal point is projected through its corresponding image pixel (x_i, y_i) and into the scene. The intersection

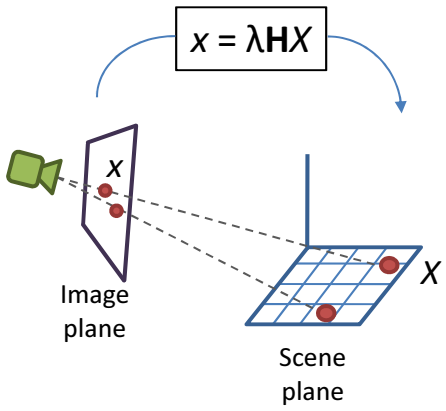
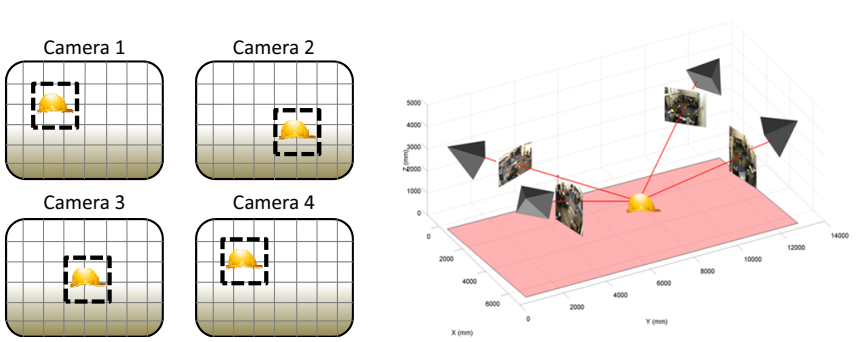


Figure 5.4: A planar homography transforms points in the image plane to the scene plane

of these rays represents the 3D coordinate of the tracked object as described in Eq. 2.12. This process is illustrated in Figure 5.5.

3. Model-based tracking [75]—This method takes a geometric description of the object in addition to the image features used in the other two methods. By projecting the model features (in their geometrically correct positions) onto the image, the object position and orientation can be estimated. This method has the benefit of being comparatively robust to occlusions and changes to orientation and scale (see Figure 5.6).

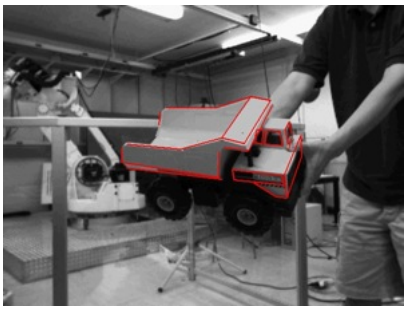
It should be noted that the planar and model-based approaches make some assumptions on the location of the object. In the former case, the object is constrained to lie on a plane in 3D space. In the latter case, an initial guess of the 3D position is required on which the success of the algorithm is highly dependent. Intersecting rays requires the least *a priori* knowledge and as such is the most general of the three; consequently it is the least robust and requires careful implementation to provide an adequate level of performance. Nevertheless, the appeal of a general approach makes triangulation the predominant method for combining multiple observations.



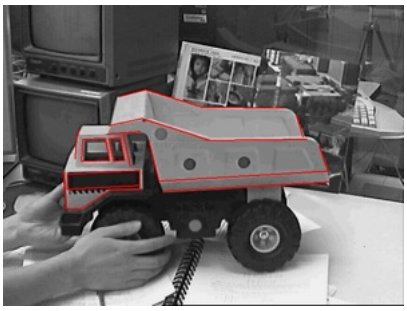
(a) Step 1: 2D localization (b) Step 2: Combining observations

Figure 5.5: The traditional approach to 3D object tracking with multiple cameras. (a) First, the object is tracked in each camera view using traditional 2D methods. (b) Next, the 3D position is reconstructed from the intersection of each pixel's ray projection.

(a)

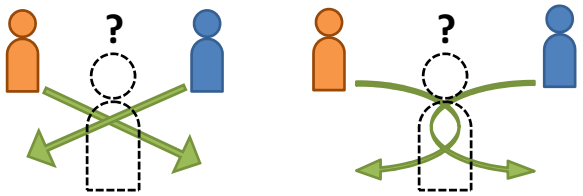


(b)



(c)

Figure 5.6: (a) Using model-based tracking, objects can be tracked while undergoing (b) rotation or (c) partial occlusion. (Images from [90])



(a) (b)

Figure 5.7: Mutual occlusion. When two objects follow intersecting trajectories, one temporarily occludes the other. Care must be taken to disambiguate cases (a) when the targets cross paths and (b) where the targets meet and then separate.

5.3 Challenges in 3D Tracking

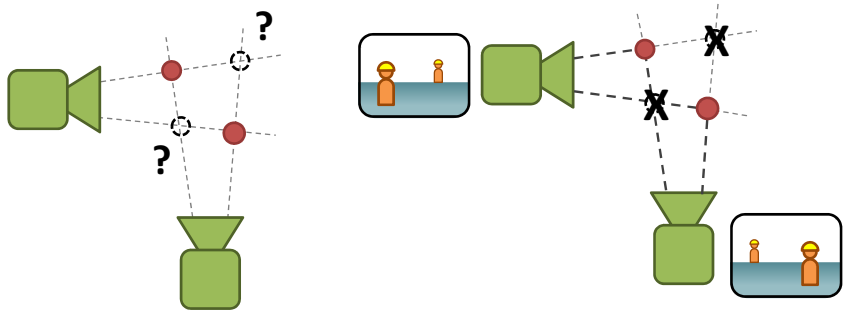
From the nature of the process described above it is clear that between the many variations of representations, image features, and ways to combine observations, the object tracking task is highly adaptable. This characteristic is in part due to the large degree of unpredictability found when considering objects in realistic settings. Tracking algorithms must contend with a number of challenges, including:

1. Occlusion. When the object is obscured by the environment or exits the frame edge in one or more cameras, the target will be lost. Care must be taken to identify obstructed cameras and exclude them from the reconstruction until the target is reacquired. Target reacquisition must be handled explicitly and is subject to additional sources of error (e.g., is the object that re-entered the scene the same as the one that previously exited?). Occlusion can occur when objects interact with static scene members, or as mutual occlusion between two tracked objects (Figure 5.7).

2. Outlier sensitivity. If tracking fails in any video stream, the resulting ray will be misprojected and the least square intersection of the rays will return an intermediate point which is not physically meaningful.

3. Scale selection. A classic issue in 2D tracking is determining scale, which is a function of the object's distance to the camera. When tracking is done independently

across cameras, a mechanism must be implemented to account for scale inconsistencies.



(a) (b)

Figure 5.8: The phantom target phenomenon. When tracking multiple objects simultaneously, care must be taken to ensure scale consistency between cameras or else depth ambiguity may result. (a) Two objects result in four ray intersections, and two possible solutions. (b) With proper scale selection, false solutions can be rejected.

4. Rotation-dependent error. If the target is not a rotationally symmetric, then as the target rotates, the apparent centroid in the image projection will drift relative to the true centroid.

5. Temporal inconsistency. Fusing observation data from multiple images assumes all observations are captured simultaneously. This is only an idealization. In practice some degree of latency between cameras is inevitable. The observations, being captured at slightly different times, are temporally inconsistent. The resulting error is a function of the object's speed and the degree of intra-camera latency present. This error can be significant, as we will show below. Although synchronizing clocks amongst cameras in the network can be accomplished easily in software (and thoroughly covered in [70]) true simultaneous triggering requires costly hardware (signal generators, repeaters, etc.).

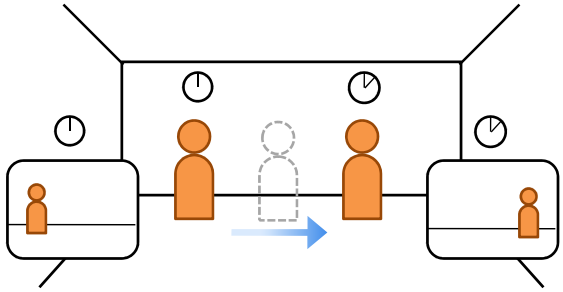


Figure 5.9: Result of temporal inconsistency. Reconstruction error occurs when cameras observe a moving object at different times.

Additional complications can result from tracking articulated objects, changes in scene illumination, and real-time processing requirements. Relying on intersecting rays to combine observations exacerbates these challenges, as a 2D localization is determined independently in each camera before being combined into a 3D position. Thus an error in one camera can strongly influence the final result; as the number of cameras increases (e.g., to increase the size of the work volume), the probability that any one camera will be in error increases. This is a major disadvantage to using triangulation, but since it is a straightforward extension of 2D tracking, this method enjoys the most widespread use.

Planar methods are computationally simple, but are not truly 3D as they simply map the image plane onto another arbitrary scene plane (or possibly, a non-linear 2D manifold). Thus planar methods are restricted to applications where the objects are constrained to the plane and have no substantial vertical extent.

Model-based methods have been successfully demonstrated for single cameras [49, 22], with some implementations offering substantial robustness to occlusion [68]. An extensive survey of monocular model-based tracking methods is available at [52]. Extensions of this work into the multi-camera domain have only briefly been explored, but offer appealing advantages. [15] describes an approach that relies on Kalman filtering and a Bayesian Belief Network to smooth trajectories of moving targets and fuse observations from multiple cameras. This allows identities of multiple targets to be retained even during occlusion or interaction between targets (mutual occlusion).

In the remainder of this chapter we describe a new approach to tracking, termed *Predictive 3D Tracking*, which iteratively aligns a 3D geometric model of an object with multiple images.

5.4 Predictive 3D Tracking

In Predictive 3D Tracking, the goal is to track an object by finding a 3D position of the target which best matches the cameras' observations. This method defers from other model-based methods in two ways:

1. The error function being minimized aggregates the error between camera observations and the projected model in *multiple* cameras, mitigating potential ambiguities in scale and position. Minimizing the error function in this away also obviates the need for explicit camera handoff as views get occluded; for each additional occlusion performance degrades gracefully.

2. Whereas existing tracking algorithms assume all observations are taken simultaneously, Predictive 3D Tracking takes each camera's observation time into account, compensating for error induced by intra-camera latency. In the case of fast moving objects, the amount of error could otherwise be significant.

In this section we describe the operation of the Predictive 3D Tracker, followed by a comparison to existing methods in simulated and empirical settings.

First, we create a model of the tracked object, M_{target}, which consists of a 3D point cloud describing its geometry along with a feature vector (texture, color, etc.). This model can be freely moved about and rotated in the target tracking space to simulate hypothetical positions or states. Using Eq. 2.12, we can project a 2D image plane of the model into each of the cameras' views. The resulting projected images then can be compared in feature space[2] to the observed images recorded by the cameras.

[2]In the computer vision literature, *feature space* commonly refers to the abstract, *n*-dimensional space that spans the features being compared. In this case, the features are the quantities derived directly from the image (texture or color, for instance). In contrast, *decision space* is often used to refer to the space of the target solution. In the case of object tracking, the decision space is the 3D Euclidean space that a physical object can occupy.

Based on the above process the Predictive 3D Tracking algorithm manipulates M_{target} to achieve a best fit across all camera observations, thereby pinpointing the object's true position in 3D space. Furthermore, the algorithm also takes into account factors such as velocity and trajectory to further improve tracking accuracy.

One of the most important advantages of this approach is that by using M_{target}, we can achieve consistency in position and scale. M_{target}'s projections into each camera's view are exact representations, albeit, theoretical, of the object had it been in its hypothesized position. By avoiding the scaling ambiguity of the baseline method, we do not need to explicitly define and incorporate compensating mechanisms for the error sources discussed in 5.3.

Another key feature of this algorithm is that it addresses intra-camera latency which is the inherent error introduced by imperfect triggering among cameras (See Chapter 2). Typically, such latencies can introduce time discrepancies as high as 50 milliseconds. With the Predictive 3D Tracking algorithm, as long as camera clocks are synchronized (an easily achievable objective), each camera can record images at slightly different times. M_{target} can then be manipulated in time as well as space for a best fit. This is much simpler than the practical problem of precisely triggering all cameras simultaneously.

In most applications, the target object being tracked is unlikely to be viewable by all cameras at all times, either due to occlusion or simply being out of a camera's field of view. Whereas the baseline method must handle this as a special case, the predictive 3D tracking method handles this implicitly, since for any M_{target} it is known if its projection is not visible to one or more cameras. In this case, the obstructed cameras can be simply ignored until the target is again within view.

This can be represented analytically as:

$$\mathbf{X} = \min_{\tilde{\mathbf{X}}} \sum_i \rho\left(P_i\tilde{\mathbf{X}}, \mathbf{x_i}\right) \tag{5.1}$$

for 3D coordinate $\mathbf{X}$, 2D coordinate $\mathbf{x}$, camera matrix P_i, and error function ρ. If ρ is the commonly used L_2-norm, the above equation becomes

$$\mathbf{X} = \min_{\tilde{\mathbf{X}}} \left\| f\left(P_1\tilde{\mathbf{X}}\right) - f\left(\mathbf{x}_1\right)\right\| + \cdots + \left\| f\left(P_N\tilde{\mathbf{X}}\right) - f\left(\mathbf{x}_N\right)\right\| \tag{5.2}$$

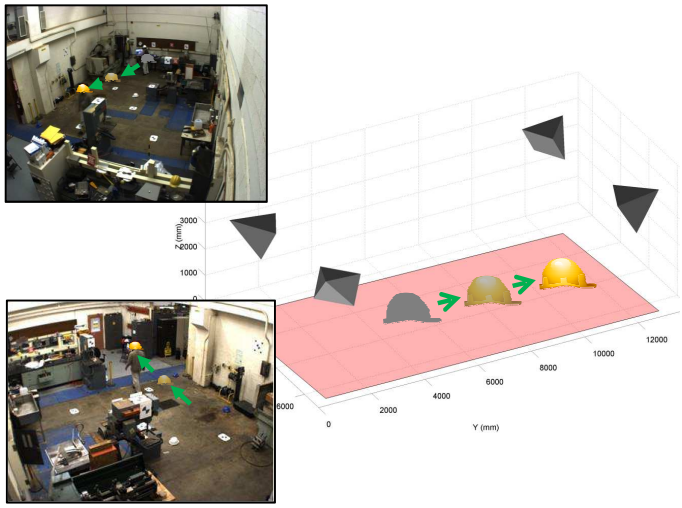


Figure 5.10: In the proposed method, a predicted position is iteratively updated until its projection matches the camera observations.

The function $f(\cdot)$ is a descriptor that describes the local neighborhood around a point using some low-level feature. This allows two image regions to be directly compared. Empirical testing found that a Hue-Saturation-Value (HSV) histogram works well.

In addition to optimizing for the position estimate $\mathbf{X}$ we also consider the velocity estimate $\dot{\mathbf{X}}$. When projecting the hypothesis position into the camera images in the optimization step, the location in the image is adjusted to account for velocity and latency (the time between corresponding frames). For camera i and latency δ, position $\mathbf{x}_i$ is given by:

$$\mathbf{x}_i = P_i\mathbf{X} + \delta_i\dot{\mathbf{X}} \tag{5.3}$$

5.5 Simulation

The MATLAB simulation first described in Chapter 3 was modified to include a target with a programmable trajectory, and an interface for different tracking algorithm implementations. The simulation tool is used to validate our 3D Predictive Tracking approach

Algorithm 5.1 Predictive 3D Tracking

```
P_i ←camera matrices for camera i
δ_i ←latency for camera i
I^t ←image streams for time t
(X,Y,Z) ←initial position
M_target ←model

do

    error←0
    for each P_i ∈ P do
        (x_i,y_i) ←PROJECT[P_i,δ_i,M_target(X,Y,Z)]
        error←error + COMPARE_FEATURES[M_target,P_i,(x_i,y_i)]
    (X,Y,Z) ←UPDATE_POSITION[(X,Y,Z)]

until (X,Y,Z) converges

return (X,Y,Z)
```

and to gather insights on the performance differences between it and two baseline methods under various operational conditions (object velocity, camera resolution and work volume dimensions). These baseline methods represent 3D tracking algorithms that rely on planar homographies and intersecting rays.

5.5.1 Baseline 1 (Triangulation)

The triangulation baseline method first localizes the object in each camera image, and then looks for the intersection of the resulting rays from each camera's focal point through the corresponding pixel. A specific implementation of the baseline method using 2D image tracking is provided by an implementation of CAMSHIFT [13], a well-studied algorithm provided by the OpenCV library [12]. As an extension of the ubiquitous mean shift algorithm, CAMSHIFT operates by color matching: iteratively searching for image regions whose local histograms match that of a model. The resulting pixel pairs (x_i^t, y_i^t) for camera i observed at time t are fused with the camera matrices P_i using Eq. 2.12. The pseudocode is listed in Algorithm 5.2.

Algorithm 5.2 Triangulation algorithm (for a single frame)

$P_i \leftarrow$`camera matrices for camera` i
$I^t \leftarrow$`image streams for time` t
$(x,y) \leftarrow$`initial position`

`for each` $P_i \in P$ `do`

$(x_i, y_i) \leftarrow$`CAMSHIFT`$[I_i^t, (x_i, y_i)]$

$(X^t, Y^t, Z^t) \leftarrow$`3D_RECONSTRUCT`$[P, (x,y)]$

`return` (X^t, Y^t, Z^t)

5.5.2 Baseline 2 (Planar Homography)

The planar baseline method used in this comparison is adapted from implementations presented in [25, 48]. Given the calibration parameters of each camera, a 3×3 homography matrix $\mathbf{H}$ can be established which maps any 2D image position to a coordinate on a plane embedded in the scene. Once the object of interest is localized within the image, the resulting coordinate x_i, y_i can be transformed via $\mathbf{H}$ to get a 3D coordinate. This process is then repeated for each camera and the results averaged. Note that this method is only suitable for objects that move on a plane; moreover the height of the object must be known relatively accurately to avoid parallax error (Figure 5.11). For example, when tracking a human-sized object the height should be known to approximately 10 *cm*.

5.5.3 Results

Figure 5.12 illustrates the error resulting from localizing a moving object when intra-camera latency is present. For a target moving at typical pedestrian speeds (about 2 m/sec), the tracking error due to latency in the baseline (triangulation) method is on the order of a few centimeters. Depending on the application requirements this level of performance may be sufficient, particularly for tracking humans. The latency compensation performed by the Predictive 3D Tracking is beneficial in applications requiring high precision, e.g. control of manipulators.

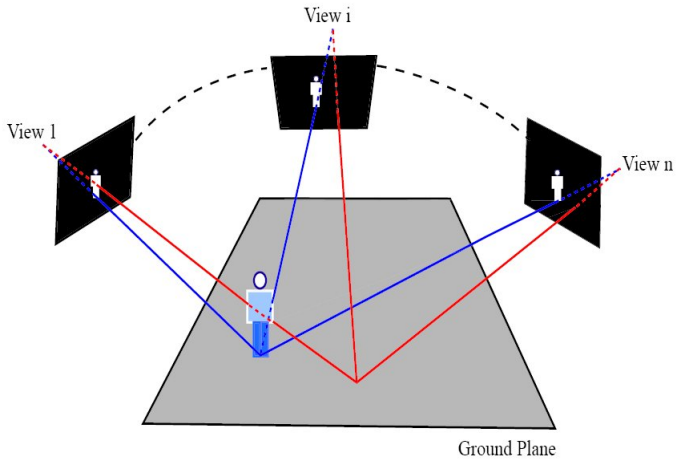


Figure 5.11: Illustration of parallax error. The blue rays tracking the figure's feet satisfy the homography constraint, as they intersect the object on the ground plane. The red ray that intersects the figure's torso does not and reprojects inconsistently to the other camera views. Image from [48].

Algorithm 5.3 Planar homography algorithm

$P_i \leftarrow$`camera matrices for camera` i
$I^t \leftarrow$`image streams for time` t
$(x,y) \leftarrow$`initial position`

`for each` $P_i \in P$ `do`

$\mathbf{H}_i \leftarrow$`HOMOGRAPHY_MATRIX`$[P_i]$

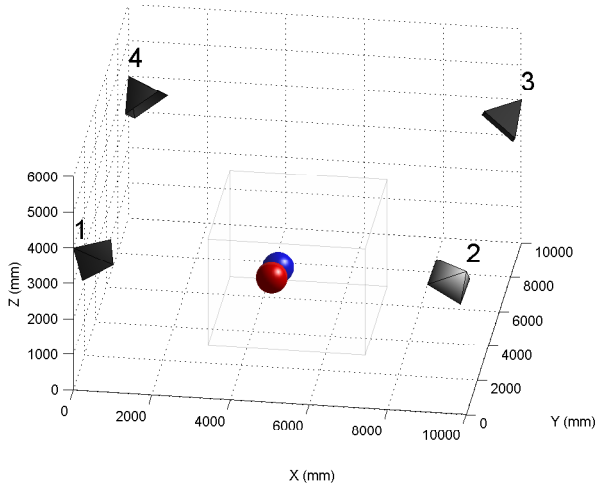
`for each` $\mathbf{H}_i \in \mathbf{H}$ `do`

$(x_i, y_i) \leftarrow$`CAMSHIFT`$[I_i^t, (x_i, y_i)]$

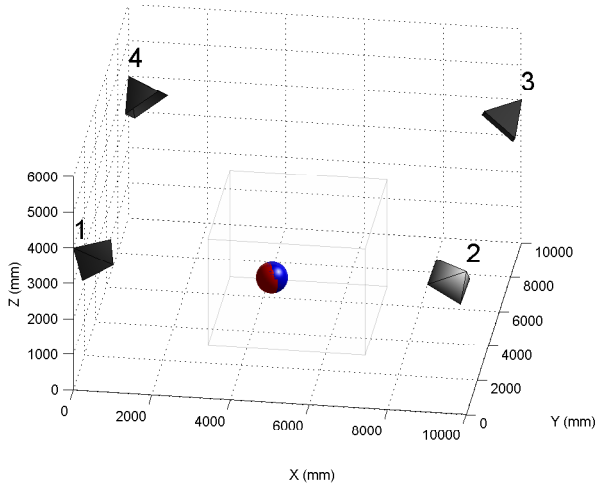
$$\begin{bmatrix} \bar{x}_i \\ \bar{y}_i \\ \bar{z}_i \end{bmatrix} \leftarrow \mathbf{H}_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$(X^t, Y^t, Z^t) \leftarrow$`mean`$(\bar{x}_i, \bar{y}_i)$

`return` (X^t, Y^t, Z^t)



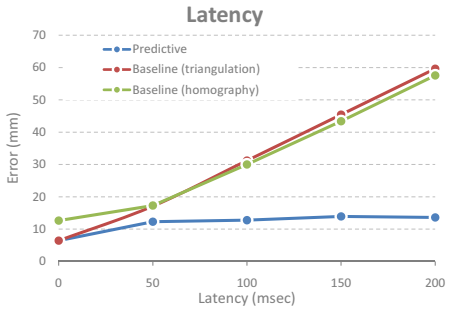
(a) Baseline method



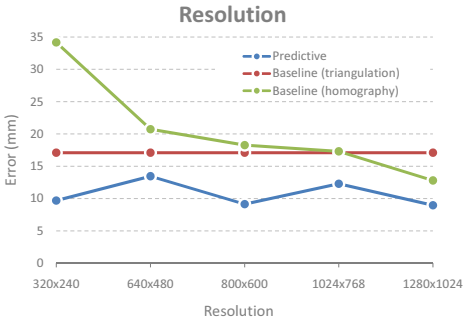
(b) Predictive 3D Tracking

Figure 5.12: Testing 3D tracking algorithms in a simulated environment. The blue target is the true position and the red target is the estimated position. Using the baseline triangulation method (a) the effects of intra-camera latency are evident, as demonstrated by the estimated position lagging the true position. With Predictive 3D Tracking (b) the effects of intra-camera latency are compensated and the targets are nearly coincident.

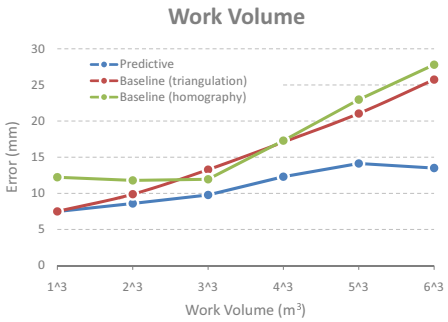
In Figure 5.13 we explore the impact of varying simulation parameters. Surprisingly, image resolution seems to have minimal influence on performance above 640×480 pixels, the minimum resolution typically encountered on commercial cameras. Figure 5.13a illustrates how the Predictive 3D Tracking method compensates for poor camera synchronization, compared to the linear increase in error associated with the baseline methods. When varying work volume dimension the algorithms behave more uniformly. In this experiment, work volume dimension was increased but camera positions were fixed. As a result, larger work volumes meant the target spent more time out of the field of views of one or more cameras.



(a)

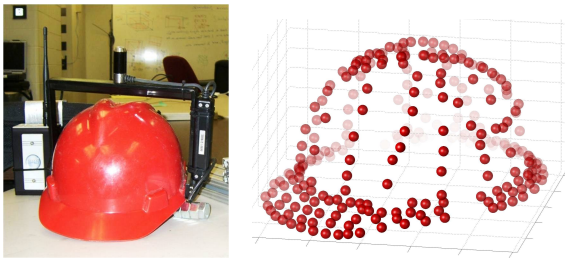


(b)



(c)

Figure 5.13: Comparison of 3D tracking methods. Error tends to increase as the (a) intra-camera latency and (c) work volume size increase. Error is relatively stable above a lower threshold of (b) resolution.



(a) (b)

Figure 5.14: (a) Close-up of tracking object. The red color of the hard hat provides a simple visual target for the cameras. Note iGPS receiver (black cylinder) mounted above the hardhat. (b) 3D point cloud model of hardhat used in the proposed method

5.6 Empirical Verification

We have also tested the Predictive 3D algorithm in a lab setting. A network of four cameras was installed at a NIST facility. The cameras were arranged at the corners of a square work volume of approximately 10 m $\times$ 10 m, and mounted height of 4 m. The task involved tracking an actor moving throughout the work volume.

The specific object being tracked was the actor's hardhat. To measure the algorithm's accuracy, we used a Metris indoor GPS (iGPS) system to establish the object's true position ("Ground Truth"). The iGPS system is a laser range finding device which is theoretically capable of 0.2 mm precision and 30 Hz update rate. Its expected precision is 2 orders of magnitude better than the camera network being tested, thus it was not expected to introduce additional error or noise.

The cameras (Point Grey Research Flea2 models) are capable of 1024 $\times$ 768 pixel resolution at a frame rate of 7.5 fps. These research-grade cameras were chosen for their particularly low noise in the low light levels present in the experimental setting. To ensure consistent data between all five devices (four cameras and the iGPS tracker), 15 calibration targets were mounted in the work volume to provide a common reference frame. To provide temporal synchronization, all devices were connected to a local NTP server. Although all the devices were synchronized they were not triggered (i.e., although all the clocks were

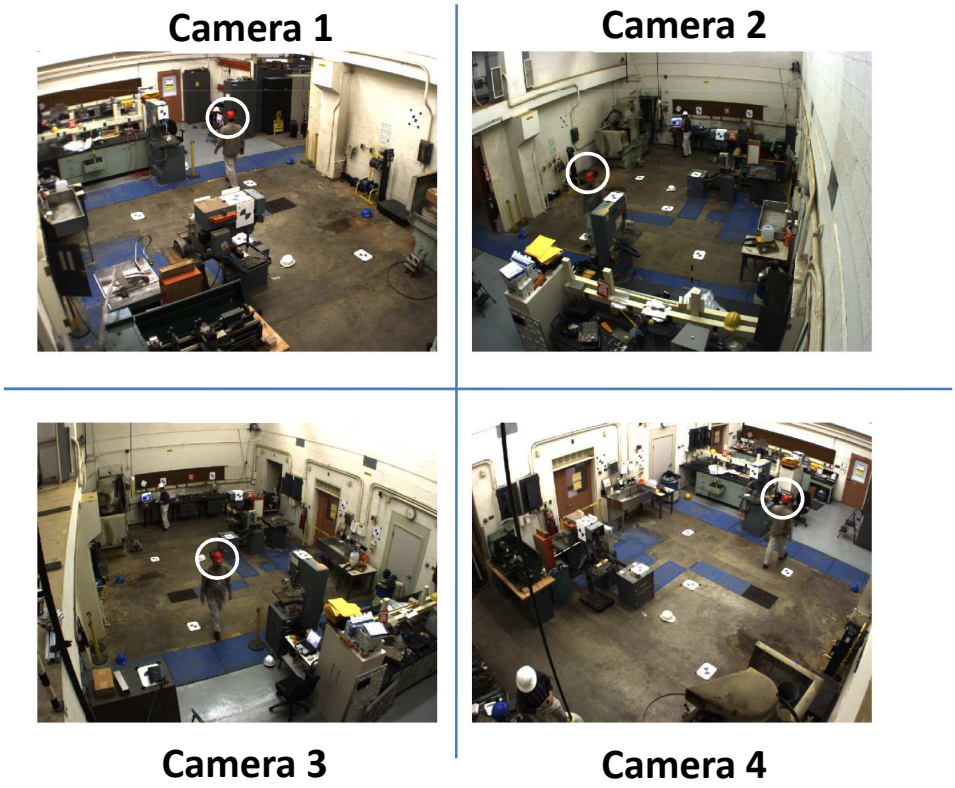


Figure 5.15: Typical camera views during a sequence. Note the hardhat being tracked (white circle)

consistent, the cameras and iGPS did not capture data at the same instant). Five video sequences[3] were recorded, demonstrating motion along the the Y-axis (North-South), X-axis (East-West), Z-axis (with a step ladder), and two with a free form motion.

The experimental results are presented in Figure 5.16. Across all sequences, the average 3D reconstruction error for the baseline method is 52.2 mm compared with the average error for the 3D Predictive Tracking method of 39.4 mm. This represents a 24% error reduction. When the experimental conditions were recreated in simulation, a similar improvement was noted. Absolute error was slightly lower in simulation however, because camera calibration uncertainty was not included in the simulation model.

[3]The term "sequence" in the computer vision context commonly refers to a short video clip used as test data.

	Empirical	Simulated
Baseline (mm)	52.2	17.1
Predictive 3D Tracking (mm)	39.4	12.3
% Improvement	**24.5%**	**28.1%**

Table 5.1: Comparison of improvement between empirical and simulated experiments.

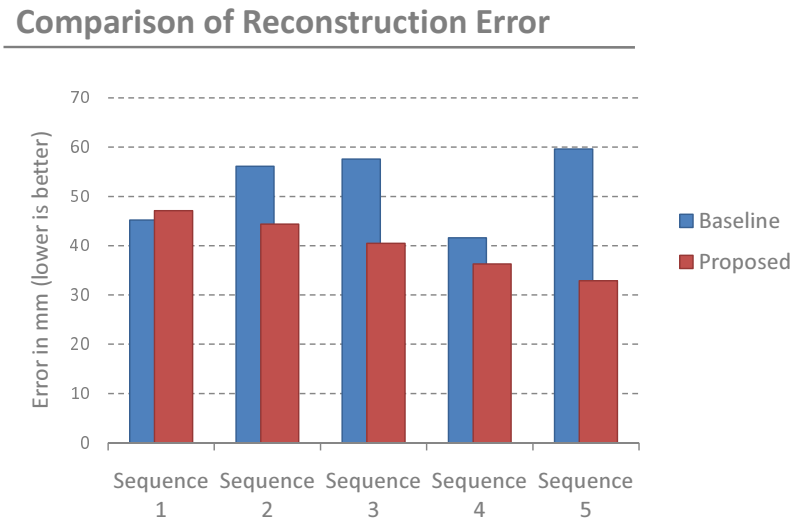


Figure 5.16: Reconstructing a trajectory from recorded sequences. The first four sequences correspond to motion along one axis. Sequence 5 is a free form motion.

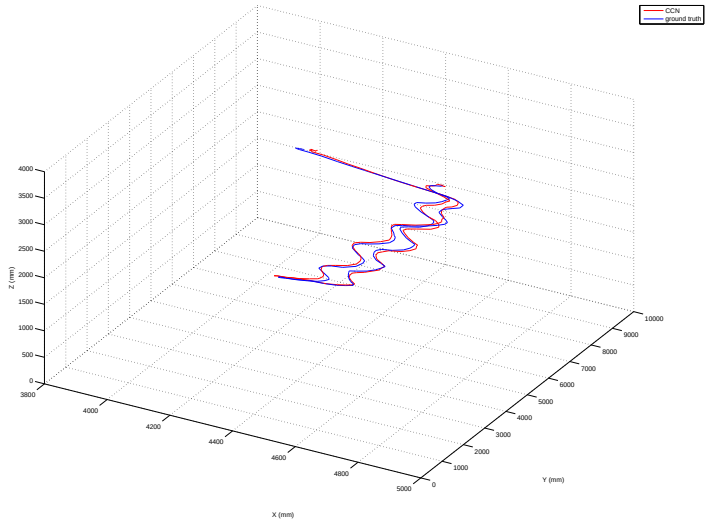


Figure 5.17: Sequence 5. The red and blue trajectories correspond to the Predictive 3D Tracking algorithm and ground truth, respectively.

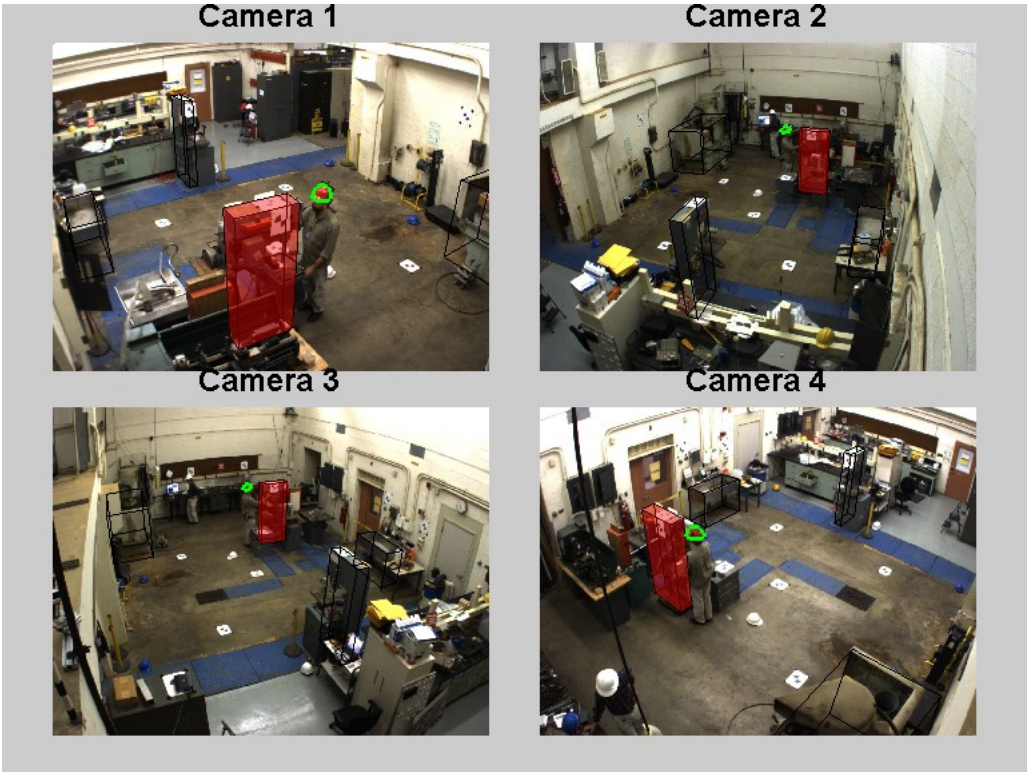


Figure 5.18: Automatically detecting proximity to a work station.

Despite the improved accuracy of the 3D Predictive Tracking approach it has several drawbacks. Because it is an iterative algorithm, it is considerably slower than the baseline method which has a fixed number of steps. On the commodity hardware used in this experiment, one frame of the proposed method took 0.48 seconds compared to the baseline's 0.15 seconds. For post-processing offline this is not a concern but applications that demand a real-time response, porting the implementation from MATLAB to a low-level language such as C/C++ should substantially reduce runtime. The other drawback is the need for a 3D point cloud of the target object. The baseline approach only requires features which are readily available from the image itself. Generating a point cloud meanwhile, requires either knowledge of modeling software, or a coordinate measuring machine (the latter was used to generate the hard hat point cloud).

5.7 Conclusion

This chapter presented an approach to 3D model-based object tracking in multiple cameras. This method described here can overcome some of the shortcomings of the standard approach, including errors due to scale and spatial inconsistencies. By estimating position along with velocity, we can additionally compensate for temporal inconsistency. The experimental results showed that performance is significantly increased over the baseline method, and the magnitude of the improvement is in agreement with simulation results, at the expense of increased computation time. Although this chapter only explored tracking object position, future work can include a relatively straightforward extension to tracking orientation.

Chapter 6

Conclusion

6.1 Summary

Camera networks are at a developmental milestone. As hardware costs go down and computer vision algorithms become more robust, camera networks are poised to emerge from laboratory installations as an extensible, multi-purpose 3D sensor. A wide variety of applications have already been suggested from various commercial sectors. However, the promise of camera networks is tempered by their complexity. Specifically, the performance of camera networks is highly dependent on the parameters the designer chooses. Understanding the implications of various design decisions, and subsequently finding optimal configurations, is a key barrier. In this dissertation we demonstrate three original contributions:

1. We present an analysis of camera network parameters and demonstrate their effect on performance in a 3D localization task. After reviewing relevant terminology and the geometry of image formation, we introduce a simulation environment which allows us to probe each parameter in turn. The resulting analysis generated a number of important insights regarding the trade-offs facing the camera network designer and recommendations were made for each. The simulation environment is shown to be a useful tool for exploring hypothetical configurations and rapidly evaluating candidate solutions.

2. We develop a novel method for automated camera network configuration based on the genetic algorithm paradigm. The sheer number of parameters, compounded by their complex interaction, make manual configuration daunting for all but the simplest cases. Existing methods (BIP and AGP) are only suitable for 2D scenes or can only handle a few cameras and thus are primarily of academic interest. For complex environments with occlusions, or for applications with highly demanding requirements, a scalable automated tool is a necessity. Towards this goal, we introduce a genetic algorithm-based method to optimize the task of camera positioning and parameter selection. By considering a large population of candidate configurations and ranking their performance, we select the best performing members and iteratively refine them until convergence. In a simulated environment this approach was shown to deliver a solution with lower error than a hand-designed camera network. Subsequent empirical verification confirmed these findings.

3. We propose a new approach to 3D tracking from multiple cameras. The reference standard algorithm for 3D tracking is an extension of 2D tracking: the target is localized in each image and then, with knowledge of camera geometry, fused to find a best-fit 3D point. Considering each camera independently, as is done in this case, has a number of limitations. In particular the timing relationship between cameras is lost—in typical installations neglecting intra-camera latency results in tracking error on the order of centimeters. In the proposed approach, Predictive 3D Tracking, targets are tracked directly in world (3D) space rather than image (2D) space. This resolves a number of ambiguities with regards to scale, location, and observation time. In simulation we demonstrate substantially reduced tracking error over the reference method. These results were mirrored in a laboratory experiment with a similar environment. This pilot study can serve as a model for the development of future multi-sensor network algorithms: camera networks have a number of capabilities that individual cameras do not. By approaching camera networks as a monolithic system we can better exploit these capabilities.

Multi-camera networks are on the cusp of revolutionizing how machines interact with the world. No other 3D sensing technology has the same combination of low cost, extensibility,

and performance potential. Although more work is needed in order to turn multi-camera networks into a commercializable reality, their unique capabilities will help ensure their future role as an indispensable sensing tool.

6.2 Future Work

The study of camera networks is a vast, interdisciplinary field which encompasses optics, computer vision, optimization, networking and other areas, only a subset of which could be addressed in this work. As an active area of research, much work remains to be done, for problems within the scope of this thesis, as well as those that have not been covered. We identify three significant extensions, corresponding to to the three contributions in this thesis.

6.2.1 Generalizing System Parameters

In parameter analysis of Chapter 3, a number of assumptions were made regarding the camera placement, homogeneity, and the types of parameters being considered. Whereas here we limited cameras to be inward facing, many existing installations have cameras distributed across rooms and corridors. Also, there is an increasing trend towards foveated imaging where different types of cameras are co-located on a node (as in the MeshEye Mote [40]). While we focused primarily on the design process for a new camera network, extending the sensitivity analysis to account for existing systems would bring the benefits of camera networks to the huge installed base of security cameras.

While extending the simulation framework is relatively straightforward, a key area for future work would be to verify the sensitivity analysis (which we only explore in simulation) against existing, field-deployed camera networks. [14] provides a laboratory-based example of this verification, but existing installations are likely to have significant complications that challenge the assumptions made here; these cameras are likely to be relatively low resolution, noisy, and poorly calibrated.

6.2.2 Model-based Optimization

In Chapter 4 we presented an automated approach to camera network configuration. While the iRoom scenario in Section 4.5 provides a proof of concept, the environment used was rectangular and occlusion-free. This is sufficient for many small baseline systems, but camera networks with wider baselines (tens of meters) will likely be subject to occlusions, both from objects in the scene as well as from the geometry of the room itself. Incorporating a geometric model of these structures will provide a more accurate prediction of performance in real-world deployments.

One way to do this to first capture a point cloud of the scene using a 3D laser scanner or photogrammetry techniques. While the former provides very high resolution geometry, the latter can be implemented with a single, uncalibrated camera and the appropriate software (such as the widely used PhotoSynth package first described in [72]). The next step would be to incorporate occlusion testing into the 3D simulation, as was done for the 2D example in Section 4.4.2. Finally, the simulation would need to be extended to import the captured scene model with the genetic algorithm optimization modified accordingly.

6.2.3 Realtime 3D Tracking

The Predictive 3D Tracking algorithm presented in Chapter 5 is promising, but may be challenging for realtime applications. Due to computational constraints (both of the computing hardware and the particular software implementation), processing of video sequences was done offline. In our experiments, optimizing a 3D point using our method was highly CPU-bound, taking about one second per frame of video. To be practical this operation must run in realtime on commodity computing hardware.

Several solutions are possible. For ease of prototyping, MATLAB was used for implementing the tracking algorithm. Multi-threaded implementations in a low-level language such as C/C++ will likely reduce the computation time significantly. Another solution is to use a hybrid approach using two tracking algorithms that have complementary precision/cost ratios. An example of this was demonstrated in [50], which used accurate SIFT features on intermittent frames, paired with fast optical flow on all other frames to achieve realtime operation.

Bibliography

[1] Digital photography review: Digital cameras timeline. http://www.dpreview.com/reviews/timeline.asp, 2010.

[2] Y. I. Abdel-Aziz and H. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range Photogrammetry*, pages 420–475, 1971.

[3] J. Albertz. A look back: 140 years of photogrammetry. *Photogrammetric Engineering & Remote Sensing*, pages 504–506, May 2007.

[4] A. Ali and J. K. Aggarwal. Segmentation and recognition of continuous human activity. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 28–35, 2001.

[5] D. Ang, Y. Shen, and P. Duraisamy. Video analytics for multi-camera traffic surveillance. In *Proceedings of the Second International Workshop on Computational Transportation Science*, pages 25–30, 2009.

[6] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[7] J. Beat. An application of axiomatic design: Uav subvehicle. Presentation, South Dakota School of Mines & Techology, 2006.

[8] J. Black and D. T. Ellis. Multi camera image tracking. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.

[9] J. Black, T. Ellis, and D. Makris. Wide area surveillance with a multi-camera network. In *Intelligent Distributed Surveillance Systems*, pages 21–25, 2003.

[10] J. Black, T. Ellis, and P. Rosin. Multi view image surveillance and tracking. In *Workshop on Motion and Video Computing*, pages 169–174, 2002.

[11] J. Y. Bouguet. Camera calibration toolbox for matlab, 2008.

[12] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Cambridge, MA, 2008.

[13] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 1998.

[14] P. Carr. Design considerations for efficient camera networks. Master's thesis, York University, 2005.

[15] T.-H. Chang and S. Gong. Tracking multiple people with a multi-camera system. In *IEEE Workshop on Multi-Object Tracking*, pages 19–26, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[16] X. Chen. *Designing Multi-Camera Tracking Systems for Scalability and Efficient Resource Allocation*. PhD thesis, Stanford University, 2002.

[17] 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, August 2002.

[18] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):484–498, 1998.

[19] R. Cucchiara, M. Piccardi, and P. Mello. Image analysis and rule-based reasoning for a traffic monitoring system. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):119–130, June 2000.

[20] T. Cuypers, C. Vanaken, Y. Francken, F. V. Reeth, and P. Bekaert. A multi-camera framework for interactive video games. In *International Conference on Computer Graphics Theory and Applications*, pages 443–449, 2008.

[21] Q. Delamarre and O. Faugeras. 3d articulated models and multi-view tracking with silhouettes. In *Proceedings of the International Conference on Computer Vision*, pages 716–721, 1999.

[22] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, 1995.

[23] D. Douxchamps. The ieee1394 digital camera list, http://damien.douxchamps.net/ieee1394/cameras/, 2010.

[24] A. O. Ercan, D. B. Yang, A. E. Gamal, and L. J. Guibas. Optimal placement and selection of camera network nodes for target localization. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems*, pages 389–404, 2006.

[25] R. Eshel and Y. Moses. Homography based multiple camera detection and tracking of people in a dense crowd. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[26] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. of the ACM*, 24:381–395, June 1981.

[27] K. P. Fishkin, M. Philipose, and A. Rea. Hands-on rfid: Wireless wearables for detecting use of objects. In *Proceedings of the International Symposium on Wearable Computers*, pages 38–43, 2005.

[28] S. Fleck, F. Busch, P. Biber, and W. Straber. 3d surveillance: A distributed network of smart cameras for real-time tracking and its visualization in 3d. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop*, 2006.

[29] S. Fleishman, D. Cohen-or, and D. Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19:2000, 1999.

[30] A. Fod, A. Howard, and M. J. Mataric. Laser-based people tracking. In *Proceedings of the IEEE International Conference on Robotics & Automation*, pages 3024–3029, 2002.

[31] C. Fraser. Photogrammetric measurement to one part in a million. *Photogrammetric Engineering & Remote Sensing*, 58:305–310, 1992.

[32] H. Furuta, K. Maeda, and E. Watanabe. Application of genetic algorithm to aesthetic design of bridge structures. *Computer-Aided Civil and Infrastructure Engineering*, 10(6):415–421, 1995.

[33] J. Gühring. Dense 3-d surface acquisition by structured light using off-the-shelf components. In *Proc. Videometrics and Optical Methods for 3D Shape Measurement*, pages 220–231, 2001.

[34] Gizmodo. Google streetview cars rocking ladybug2 spherical camera. http://gizmodo.com/gadgets/ladybug2-picnic/google-streetview-cars-rocking-ladybug2-spherical-camera-283769.php, 2010.

[35] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3:95–99, 1988.

[36] S. Gould, J. Arfvidsson, A. Kaehler, B. Sapp, M. Messner, G. R. Bradski, P. Baumstarck, S. Chung, and A. Y. Ng. Peripheral-foveal vision for real-time object recognition and tracking in video. In *International Joint Conferences on Aritifical Intelligence*, pages 2115–2121, 2007.

[37] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[38] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106 –1112, 1997.

[39] S. Hengstler, H. Aghajan, and A. Goldsmith. Application-oriented design of smart camera networks. In *Proceedings of the International Conference on Distributed Smart Cameras*, pages 12–19, 2007.

[40] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 360–369, 2007.

[41] J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[42] E. Horster and R. Lienhart. On the optimal placement of multiple visual sensors. In *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, pages 111–120, 2006.

[43] S.-C. C. Jian Zhao and T. Nguyen. Optimal camera network configurations for visual tagging. *IEEE Journal of Selected Topics in Signal Processing, vol. 2, issue 4, pp. 464-479*, 2:464–479, Aug. 2008.

[44] I. Katz, K. Saidi, and A. Lytle. The role of camera networks in construction automation. In *The 25th International Symposium on Automation and Robotics in Construction*, 2008.

[45] I. Katz, K. Saidi, and A. Lytle. The role of camera networks in construction automation. In *Proceedings of the 25th International Symposium on Automation and Robotics in Construction*, 2008.

[46] I. Katz, K. Saidi, and A. Lytle. Model-based 3d tracking in multiple cameras. In *International Symposium on Automation and Robotics in Construction*, 2010.

[47] I. Katz, N. Scott, and K. Saidi. A performance assessment of calibrated camera networks for construction site monitoring. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 244–247, 2008.

[48] S. M. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proceedings of the European Conference on Computer Vision*, 2006.

[49] D. Koller, K. Danilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[50] T. Lee and T. Hollerer. Multithreaded hybrid feature tracking for markerless augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):355 –368, 2009.

[51] J. Leonard and H. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic, 1992.

[52] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. In *Foundations and Trends in Computer Graphics and Vision*, pages 1–89, 2005.

[53] B. Liu, M. Yu, D. Maier, and R. Männer. An efficient and accurate method for 3d-point reconstruction from multiple views. *International Journal of Computer Vision*, 65:175–188, 2002.

[54] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.

[55] S. Luke. *Essentials of Metaheuristics*. 2009. available at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[56] A. Lytle, I. Katz, and K. Saidi. Performance evaluation of a high-frame rate 3d range sensor for construction applications. In *Proceedings of the International Symposium on Automation and Robotics in Construction*, 2005.

[57] T. Maatta, A. Harma, and H. Aghajan. On efficient use of multi-view data for activity recognition. In *Proceedings of the International Conference on Distributed Smart Cameras (ICDSC) 2008*, 2008.

[58] D. Maddalena and G. Snowdon. Applications of genetic algorithms to drug design. In *Expert Opinion on Therapeutic Patents*, pages 247–254, 1997.

[59] E. McCollough. Photographic topograhy. *Industry: A Monthly Magazine Devoted to Science, Engineering and Mechanic Arts*, pages 399–406, 1893.

[60] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1998.

[61] H. Nakashima, H. Aghajan, and J. C. Augusto, editors. *Handbook of Ambient Intelligence and Smart Environments*. Springer, New York, 2010.

[62] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. Landmarc: Indoor location sensing using active rfid. *Wireless Networks*, 10(6):701–710, November 2004.

[63] P. Nikitin and K. Rao. Performance limitations of passive uhf rfid systems. In *Antennas and Propagation Society International Symposium*, pages 1011–1014, 2006.

[64] G. Olague and R. Mohr. Optimal camera placement for accurate reconstruction. *Pattern Recognition*, 35(4):927 – 944, 2002.

[65] J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., New York, NY, 1987.

[66] Qualitest. Total station range surveying equipment. http://www.worldoftest.com/totalstation.htm, Nov. 2010.

[67] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:187–194, 2004.

[68] J. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. *IEEE International Conference on Computer Vision*, pages 612–617, 1995.

[69] J. Ren, J. Orwell, G. A. Jones, and M. Xu. Tracking the soccer ball using multiple fixed cameras. *Computer Vision and Image Understanding*, 113(5):633–642, May 2009.

[70] K. Römer, P. Blum, and L. Meier. *Handbook of Sensor Networks: Algorithms and Architectures*, chapter Time Synchronization and Calibration in Wireless Sensor Networks, pages 199–237. 2005.

[71] P. Schalk, E. Fauster, P. O'Leary, and M. Weiss. Framework for automatic quality control in industrial environments using distributed image processing. *Journal of Electronic Imaging*, 13(3):504–514, 2004.

[72] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, pages 835–846, 2006.

[73] C. Stauffer and K. Tieu. Automated multi-camera planar tracking correspondence modeling. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 259–266, 2003.

[74] G. Stein, R. Romano, and L. Lee. Monitoring activities from multiple video streams: Establishing a common coordinate frame. Technical report, MIT, 1999.

[75] B. Stenger, P. R. S. Mendonça, and R. Cipolla. Model based 3D tracking of an articulated hand. In *Computer Vision and Pattern Recognition*, pages 310–315, 2001.

[76] P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 432–437, 1999.

[77] N. P. Suh. *The Principles of Design*. Oxford University Press, 1990.

[78] A. Sundaresan and R. Chellappa. Multi-camera tracking of articulated human motion using motion and shape cues. In *Asian Conference on Computer Vision*, pages 131–140, 2006.

[79] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, 2000.

[80] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[81] E. P. K. Tsang and J. Li. Combining ordinal financial predictions with genetic programming. In *Proceedings of the Second International Conference on Intelligent Data Engineering and Automated Learning*, pages 532–537, 2000.

[82] Valeo. Valeo equips first vehicle with its multi-camera system. *http://www.valeo.com/en/press-releases/details.html?id=94*, 2009.

[83] C. J. Veenman, C. J. Veenman, M. J. T. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:54–72, 2001.

[84] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965 –980, Oct. 1992.

[85] B. Wolfgang, K. Werner, M. Florian, M. Stefan, M. Bernhard, N. Thomas, S. Johannes, and S. Stefan. *Application development and management of smart camera networks*, pages 259–266. Springer Berlin Heidelberg, November 2009.

[86] D. Woo and D. Capson. 3d visual tracking using a network of low-cost pan/tilt cameras. In *Proceedings of the 2000 Canadian Conference on Electrical and Computer Engineering*, pages 884–889, 2000.

[87] J. Wu, R. Sharma, and T. Huang. Analysis of uncertainty bounds due to quantization for three-dimensional position estimation using multiple cameras. *Optical Engineering*, 37:280–292, Jan. 1998.

[88] O. Wulf and B. Wagner. Fast 3d scanning methods for laser measurement systems. In *International Conference on Control Systems and Computer Science*, pages 312–317, 2003.

[89] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), 2006.

[90] Y. Yoon, A. Kosaka, J. B. Park, and A. C. Kak. A new approach to the use of edge extremities for model-based object tracking. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.*, pages 1871–1877, 2005.

[91] K. Yu, J.-p. Montillet, A. Rabbachin, P. Cheong, and I. Oppermann. Uwb location and tracking for wireless embedded networks. *Signal Process.*, 86(9):2153–2171, 2006.

[92] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 1998.

[93] Z. Zhang. Camera calibration with one-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):892 –899, 2004.