SELF-LOCALIZING SMART CAMERAS AND THEIR APPLICATIONS

Babak Shirmohammadi

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2012

Camillo J. Taylor Associate Professor of Computer and Information Science
Supervisor of Dissertation

Jianbo Shi Associate Professor of Computer and Information Science
Graduate Group Chairperson

Dissertation Committee:

Kostas Daniilidis Professor of Computer and Information Science

Jianbo Shi Associate Professor of Computer and Information Science

Mark Yim Professor of Mechanical Engineering and Applied Mechanics

Hamid Aghajan Professor of Electrical Engineering

Self-localizing Smart Cameras and their applications

Acknowledgments

I am deeply grateful and indebted to many people without their contributions and support the completion of this work would have remained an unfulfilled dream.

I wish to express my sincere gratitude to my major advisor CJ Taylor, for giving me the privilege of working with him and for his continuous guidance, support and encouragement throughout my years at UPENN specially for preparing this document. Without his support this document would have not gone very far from the acknowledgment page! I have benefited from his profound experience and knowledge in different areas of computer sciences and his positive attitude. His advice will continue to have a great impact on my professional and personal life.

I would like to express my appreciation to the members of my Advisory Committee, Dr. Kostas Daniilidis, Dr. Jianbo Shi, Dr. Mark Yim and Dr. Hamid Aghajan, for taking time out of their busy schedule to review this work and for their valuable comments and scientific suggestions which aided me to further improve this research project.

Furthermore, I would like to especially thank Dr. Mark Yim and Dr. Vijay Kumar for giving me the opportunity to work at GRASP lab as a research specialist which eventually led to my enrollment in the PhD program.

I would also like to thank Anthony Cowley for his help and support throughout my employment, WPEII exam, proposal and defense at UPENN.

Furthermore, I would like to thank Mike Felker for always being there when I needed him. Without his efficient and fast support I probably would have been

thrown out of the program, because I had forgot to sign a paper or register for a course.

I would also like to thank my former and current fellow graduate students Arvind Bhusnurmath, Roy Anati, Elena Bernardis, Mike Park, Jimmy Sastra, Alexander Toshev and Paul Vernaza for their friendship, great conversations and assistance in the GRASP lab.

Furthermore, I am eternally grateful to the Baha'i Institute for Higher Education (BIHE) in Iran Especially Mr. Foad Sanei and Mr. Kamran Mortezaei Farid. As of the date of this document, Mr. Farid has been imprisoned for over a year serving a five year sentence solely for being an educator at BIHE. The continuous commitment, courage and high spirit of the professors, staff and students of this university have been and always will be a driving force for me to thrive to excellence in every aspect of my personal and professional life.

Moreover, I would like to acknowledge my family for their unconditional love, everlasting support and encouragement throughout the years of my studies. I am forever grateful to my parents for their invaluable advice, continuous love and sacrifice, and to my sister, for her love and support.

Furthermore, I wish to express my gratitude to my beautiful wife for her genuine love, utmost affection, tremendous support and sacrifice throughout these years. Without her continuous words of encouragement I would have stopped pursuing a PhD a long time ago.

ABSTRACT

SELF-LOCALIZING SMART CAMERAS AND THEIR APPLICATIONS

Babak Shirmohammadi

Camillo J. Taylor Associate Professor of Computer and Information Science

As the prices of cameras and computing elements continue to fall, it has become increasingly attractive to consider the deployment of smart camera networks. These networks would be composed of small, networked computers equipped with inexpensive image sensors. Such networks could be employed in a wide range of applications including surveillance, robotics and 3D scene reconstruction.

One critical problem that must be addressed before such systems can be deployed effectively is the issue of localization. That is, in order to take full advantage of the images gathered from multiple vantage points it is helpful to know how the cameras in the scene are positioned and oriented with respect to each other. To address the localization problem we have proposed a novel approach to localizing networks of embedded cameras and sensors. In this scheme the cameras and the nodes are equipped with controllable light sources (either visible or infrared) which are used for signaling. Each camera node can then automatically determine the bearing to all the nodes that are visible from its vantage point. By fusing these measurements with the measurements obtained from onboard accelerometers, the camera nodes are able to determine the relative positions and orientations of other nodes in the network.

This localization technology can serve as a basic capability on which higher level applications can be built. The method could be used to automatically survey the locations of sensors of interest, to implement distributed surveillance systems or to analyze the structure of a scene based on the images obtained from multiple registered vantage points. It also provides a mechanism for integrating the imagery obtained from the cameras with the measurements obtained from distributed sensors.

We have successfully used our custom made self localizing smart camera networks to implement a novel decentralized target tracking algorithm, create an ad-hoc range

finder and localize the components of a self assembling modular robot.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As the prices of cameras and computing elements continue to fall, it has become increasingly attractive to consider the deployment of smart camera networks. Such networks would be composed of small, networked computers equipped with inexpensive image sensors. These camera networks could be used to support a wide variety of applications including environmental modeling, 3D model construction and surveillance. For example, in the near future it will be possible to deploy small, unobtrusive smart cameras in the same way that one deploys lightbulbs, providing ubiquitous coverage of extended areas. We could imagine using such a system to track passengers at an airport from the time that they arrive at curbside check in to the time that they board their flight.

A number of research efforts at a variety of institutions are currently directed toward realizing aspects of this vision. The Cyclops project at the Center for Embedded Networked Sensing (CENS) has developed small low power camera modules and has applied them to various types of environmental monitoring applications [RBI+05]. Kulkarni et al describe the SensEye system which provides a tiered architecture for multi camera applications [KGSL05]. Hengstler and Aghajan describe a smart camera mote architecture for distributed surveillance [HA06]. The Panoptes system at the Oregon Graduate Institute [FKFB05] and the IrisNet project at Intel Research

[NDK+02, NDG02, GKK+03] both seek to demonstrate applications based on networks of commercial off the shelf web cameras. Bhattacharya, Wolf and Chellapa have also investigated the design and utilization of custom smart camera modules under the aegis of the Distributed Smart Camera Project [YZC03, LLWW04]. Several research efforts on smart camera systems build upon or relate to the work being done in the field of sensor networks [ECPS02, HSW+00].

One critical problem that must be addressed before such systems can be deployed effectively is the issue of localization. That is, in order to take full advantage of the images gathered from multiple vantage points it is helpful to know how the cameras in the scene are positioned and oriented with respect to each other. In this thesis localization is viewed as a core capability which enables a range of applications; consider for instance how Global Positioning Systems (GPS) have ushered in a host of novel applications ranging from automated route planning to precision monitoring of geological motions.

This thesis makes two main contributions: firstly it describes a novel approach to the problem of localizing a distributed ensemble of smart cameras. This method offers advantages over current state of the art vision based localization schemes in terms of computational complexity, communication complexity, accuracy and scalability. Secondly, the thesis demonstrates the utility of the proposed approach by showing how such self localizing smart cameras could be applied to a range of problems including, surveillance, robotic guidance and 3D reconstruction.

More specifically, this thesis describes a novel deployment scheme where each of the smart cameras is equipped with a colocated controllable light source which it can use to signal other smart cameras in the vicinity. By analyzing the images that it acquires over time, each smart camera is able to locate and identify other nodes in the scene. This arrangement makes it possible to directly determine the epipolar geometry of the camera system from image measurements and, hence, provides a means for recovering the relative positions and orientations of the smart camera

nodes.

The scheme proposed in this thesis involves a combination of hardware and software and is, therefore, most applicable in situations where the user has some control over the design of the smart camera nodes. We argue that relatively small additions to the smart camera hardware, namely an accelerometer and a signaling LED, can be leveraged by an appropriate localization algorithm to determine the relative location of the nodes with respect to each other rapidly, reliably and accurately.

A number of considerations motivate the design of the localization scheme proposed in this thesis. These factors are listed below:

Simplicity The proposed scheme is designed to be amenable to implementation on embedded processors and other platforms where computation and communication may be limited.

Accuracy The scheme takes as input bearing angles derived from images which can typically be measured quite accurately. Further the algorithm exploits the epipolar structure of the measurement system for greater numerical stability. Experiments were carried out to investigate how the accuracy of the scheme varied as important parameters were changed.

Speed The scheme has been designed with speed in mind so that networks consisting of hundreds of cameras can be localized in a matter of seconds using modest hardware. This means that the scheme can be used to support ad-hoc deployment scenarios where constellations of sensors may be installed or redeployed on an as needed basis.

Reliability The scheme does not make assumptions about the distribution of landmarks or moving targets in the scene. So one can more readily predict how well a particular arrangement of smart cameras can be localized.

Scalability The scheme was designed to support deployments over extended environments that may involve hundreds or thousands of cameras. Furthermore

the localization computations can be distributed to support contexts where sensors may be added or removed from the system asynchronously.

The remainder of the thesis is organized as follows: the following section briefly surveys some related work on sensor localization in general and smart camera localization in particular. Chapter 2 describes the details of the localization scheme that underlies the thesis. Chapter 3 describes various realizations of the scheme in hardware culminating with the most recent implementation on custom designed smart camera nodes. Chapter 4 discusses a series of experiments that were carried out to characterize the performance of the scheme with actual hardware and in simulation. Chapter 5 provides a comparison of the proposed localization scheme with a state of the art feature based localization algorithm. This section presents a quantitative comparison of the methods through a series of simulation experiments. It also provides a discussion of the issues associated with implementing both schemes.

Chapters 6, 7 and 8 describe various applications of the proposed localization scheme. Chapter 6 describes how a network of self localizing cameras can be used to implement an ad-hoc tracking system and discusses the implementation of such a system. Chapter 7 illustrates how the scheme could be applied to various robotics problems by describing how it has been used to localize a set of modular robot components. Chapter 8 describes how the self localizing cameras were used to form an ad-hoc laser range scanner. Finally Chapter 9 discusses some of the conclusions drawn from this research effort.

1.1 Related Work

Much of the work on localization in the context of sensor networks has concentrated on the use of time of flight or signal strength measurements of radio or audio transmissions [BHET04, MLRT04, NL03]. The Cricket ranging system developed at MIT is one example of such an approach. This thesis concentrates on the problem of

recovering the relative configuration of the sensors using angular measurements derived from images rather than range measurements. It is important to note that in this framework angular measurements derived from images and range measurements derived from other sources can be treated as *complementary* sources of information. Measurements derived from the vision system can be used to determine the relative orientations of the camera systems which is important information that cannot be derived solely from range measurements. On the other hand, range measurements can be used to resolve the scale ambiguity inherent in angle only localization schemes. Similarly angular measurements can be used to disambiguate the mirror reflection ambiguities that are inherent in range only localization schemes. Ultimately it is envisioned that smart camera networks would incorporate range measurements derived from sources like the MIT Cricket system, GPS receivers or Ultra Wide Band radio transceivers. These measurements could be used to improve the results of the localization procedure and to localize nodes that may not be visible to the smart camera nodes.

There has, of course, been a tremendous amount of work in the computer vision community on the problem of recovering the position and orientation of a set of cameras based on images. Snavely, Seitz and Szeliski [SSS06] describe an impressive system for recovering the relative orientation of multiple snapshots using feature correspondences. This work builds on decades of research on feature extraction, feature matching and bundle adjustment. An excellent review of these methods can be found in [HZ03].

Antone et al [AT02] and Sinha et al [SP06] both describe schemes for calibrating collections of cameras distributed throughout a scene. Sinha et al [SP06] discuss effective approaches to recovering the intrinsic parameters of a pan tilt zoom camera while Antone et al [AT02] discuss approaches that leverage the rectilinear structure of buildings to simplify the localization procedure.

Devarajan et al [DRC06, DCR08, CDR07] describe an interesting scheme which

distributes the correspondence establishment and bundle adjustment process among the cameras. The scheme involves having the cameras communicate amongst themselves to detect regions of overlap. This approach can be very effective when sufficient correspondences are available between the frames.

Several researchers have developed algorithms to discover spatio-temporal correspondences between two unsynchronized image sequences [TG04, CSI06, WZ02, CPSK04]. Once these correspondences have been recovered, it is often possible to recover the epipolar geometry of the camera system. The idea of using correspondences between tracked objects to calibrate networks of smart cameras has also been explored by Rahimi et al [ARD04] and by Funiak et al.[FGPS06]. These approaches can be very effective when the system can discover a sufficient number of corresponding tracks.

Another interesting approach to smart camera localization has been presented by Sinha, Pollefeys and McMillan [SPM04] who describe a scheme for calibrating a set of synchronized cameras based on measurements derived from the silhouettes of figures moving in the scene.

All of the camera localization schemes that have been advanced in the literature involve the solution of two key subproblems: establishing correspondences between features seen in multiple images and recovering the relative configuration of the ensemble from these correspondences. The approach proposed to solving these problems in this thesis offers advantages over previously proposed approach in the specific context of distributed, embedded smart cameras where the computational and communication constraints on the nodes are more stringent.

In the proposed approach the critical problem of establishing correspondences between the nodes is accomplished via optical signaling. This provides a mechanism for reliably identifying nodes in the scene. Other schemes rely critically on the existence of an appropriate set of stationary or moving targets in the scene that can be matched between views. The advantage of such schemes is that they do not

require any modification of the camera hardware, however they can fail in situations where such correspondences are hard to obtain or are not appropriately spaced. Furthermore, the problem of matching objects between views is non-trivial and can require significant computational resources particularly as the number of images grows. Agarwal et al. [ASS+09] describe a state of the art, optimized, distributed scheme for finding correspondences between images. They report computation times of 5 hrs, 13 hours and 27 hours to find correspondences among 57,845, 150,000 and 250,000 images respectively. These computations were performed on a network of 62 dual quad core machines. Cheng et al. [CDR07] propose the use of feature digests to effectively reduce the amount of information that must be sent between smart cameras in a network to establish correspondences. However, this scheme still requires each camera to broadcast approximately 100 kilobytes of information to every other node in the ensemble.

The scheme proposed in this thesis provides a more direct approach that can reliably identify neighboring nodes without any prior information in a matter of seconds using embedded processors. Effectively the smart camera nodes act as their own fiducials and the risks associated with relying on an appropriate distribution of feature correspondences in the scene are reduced.

Importantly the resulting sightings directly measure the epipolar structure of the camera network and, therefore, provide more information about the relative location of the nodes than shared point correspondences. This can be seen by noting that two cameras that can see each other can determine their relative position and orientation up to a scale whereas traditional relative orientation schemes require at least 5 correspondences in a non-degenerate configuration to recover the same information. Because of this the number of measurements required to localize the network and the amount of information that must be communicated between the nodes is significantly reduced.

Another key advantage of the proposed scheme is that it leverages the sparseness

inherent in the system of sighting measurements which makes the resulting algorithms much faster than standard vision-based schemes. In recent work Agarwal et al. [ASS+09] describe a state of the art, heavily optimized vision-based localization system intended for city-scale reconstructions. They report reconstruction times of 16.5 hours, 7 hours and 16.5 hours on data sets with 11,868, 36,658 and 47,925 images respectively. Furukawa etal. [FCSS09] applied this reconstruction method to indoor environments they report reconstruction times of 13 minutes for a system with 22 cameras, 76 minutes for a system with 97 cameras, 92 minutes for a system with 148 images and 716 minutes for a system with 492 cameras on a dual quad-core 2.66 GHz computer. Devarajan et al. [DRC06] describe a distributed approach to camera localization and report reconstruction times on the order of 54 minutes for a system with 40 images.

In contrast, the method described in this thesis can be used to localize networks consisting of hundreds of cameras in a matter of seconds with modest computational effort. This is particularly relevant in the context of smart camera systems where computational effort can be directly related to power consumption and time complexity determines the responsiveness of the system. In our experiments with our ten camera implementation the nodes were typically able to detect each other, communicate their measurements and recover their relative positions within 30 seconds. The most time consuming phase being the blinker detection portion which could be accelerated with faster frame rates. The resulting system is fast enough that it can be used for ad-hoc deployments and can respond quickly when nodes are added, removed or displaced.

Recently two interesting algorithms have been proposed which address the smart camera localization problem using distributed, consensus style schemes driven by message passing. Piovan et al. [PSF+08] describe a scheme for recovering the relative orientation of a set of cameras in the plane. Their method converges over time to an estimate that is close to the global least squares estimate. Tron and Vidal

describe a scheme that recovers the position and orientation of a set of cameras in 3D. Their method relies on standard algorithms from computer vision that are employed to recover the relative position and orientation of pairs of cameras based on correspondences between the two images. Their approach uses a series of message passing steps to recover an estimate for the relative orientation of the cameras, then another set of message passing steps to recover the translation between the cameras. Lastly the pose estimates are refined by a final set of iterations which adjust both the position and orientation of the nodes. Both methods are effectively distributed forms of gradient descent which seek to optimize agreement between the predicted image measurements and the observed values at each node.

Distributed localization methods based on consensus style approaches have the advantage of only requiring communication between neighboring nodes in the network. However, the number of communication steps in these schemes depends critically on the convergence rate of the algorithm. For example Tron and Vidal [TV09] report using 1400 rounds of message passing to localize a network of 7 nodes. In contrast, in the method proposed in this thesis the nodes that perform the localization procedure collect the sighting measurements from all of the nodes they wish to localize and run a computation to determine the relative configuration of the ensemble. This procedure only requires one round of communication and only three numbers need to be transmitted for each measurement.

Early versions of the scheme proposed in this thesis were described in [Tay04, TC05] subsequent works that built on these ideas were presented in [TS06] and in [BSLS06]. The concepts were also adapted for use on small self assembling mobile robots as discussed in [STY+07]. This thesis describes a novel variant of the scheme which leverages the measurements from three axis accelerometers onboard the cameras. These measurements allow the cameras to gauge their orientation with respect to gravity and greatly simplify the problem of recovering the relative orientation of the cameras. Once the camera orientations have been estimated, the localization

problem is effectively reduced to the problem of solving a sparse system of linear equations. A subsequent, optional bundle adjustment stage can be employed to further refine the position estimates. Here again we show how one can exploit the sparse structure of the measurement system and perform this optimization efficiently even on networks involving hundreds of cameras.

Importantly, the proposed scheme allows us to develop smart camera systems that can be deployed and calibrated in an ad-hoc fashion without requiring a time consuming manual surveying operation.

Chapter 2

Technical Approach to Localization

Figure 2.1 illustrates the basic elements of our vision based localization system. In this localization scheme each of the embedded camera systems is equipped with a controllable light source, typically an infrared Light Emitting Diode (LED), a three-axis accelerometer and a wireless communication system. Each smart camera uses its signaling LED as a blinker to transmit a temporally coded sequence which serves as a unique identifier. The cameras detect other nodes in their field of view by analyzing image sequences to detect blinking pixels and, hence, are able to determine the relative bearing to other visible nodes. Figure 2.1 shows the simplest situation in which two nodes can see each other. Here we note that the accelerometer measurements provide another independent source of information about the orientation of the cameras with respect to the vertical axis. These measurements allow two smart cameras to determine their relative position and orientation up to a scale factor. When a collection of smart cameras is deployed in an environment, these visibility relationships induce a sparse graph among the cameras as shown in Figure 2.7 where the nodes correspond to the smart cameras and the edges to the bearing measurements. These measurements can be used to localize the entire network. The scheme

provides a fast, reliable method for automatically localizing large ensembles of smart camera systems that are deployed in an ad hoc manner.

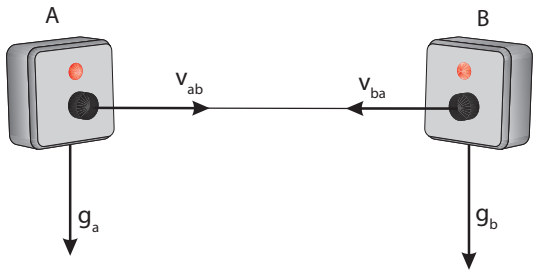


Figure 2.1: This figure shows the basic elements of the proposed localization scheme. It depicts two smart camera nodes equipped with controllable light sources and accelerometers. The camera nodes are able to detect and identify other nodes in the scene by analyzing their video imagery. They can then determine their relative position and orientation up to a scale from the available measurements. Larger networks can be localized by leveraging this relative localization capability.

One advantage of the proposed localization scheme is that it can also be used to detect and localize other smaller, cheaper sensor nodes that are simply outfitted with blinking LEDs. Figure 2.3 shows the result of localizing a constellation of 4 smart cameras and 3 blinker nodes. The ability to automatically survey the locations of a set of sensor motes distributed throughout a scene could be used to enable a variety of application. We could imagine, for example, using the smart camera system to localize a set of audio sensors in an environment. Once this has been accomplished the signals from the microphone sensors could be correlated to localize sound sources in the scene as was done by Simon et al. [SBM+04]. The locations of these sound sources could then be related to the images acquired by the cameras so that appropriate views of the sound source could be relayed to the user.

Various components of the proposed localization scheme are described in more detail in the following section.

2.1 Algorithm

2.1.1 Blinker Detection

In the first stage of the localization process, the nodes signal their presence by blinking their lights in a preset pattern. That is, each of the nodes would be assigned a unique string representing a blink pattern such as 10110101, the node would then turn its light on or off in the manner prescribed by its string. Similar temporal coding schemes are employed in laser target designators and freespace optical communication schemes. [1]

The blink patterns provide a means for each of the camera equipped nodes to locate other visible nodes in their field of view. They do this by analyzing the images to locate pixels whose intensity varies in an appropriate manner. This approach offers a number of important advantages, firstly it allows each node to localize and identify neighboring nodes since the blink patterns are individualized. Secondly, it allows the system to reliably detect nodes that subtend only a few pixels in the image which allows for further miniaturization of the camera and sensor nodes.

Figure 2.2 depicts the timing of the blink pattern and the image acquisition process. As described earlier, the blinkers continuously repeat a prescribed bit sequence at a fixed frequency. In the current implementation, this blinking function is carried out on each node by a microcontroller based subsystem which controls an LED array.

Our current optical detection scheme *does not* seek to synchronize the image acquisition process on the cameras with the blinkers. This implementation decision significantly reduces the complexity of the system and the amount of network traffic required.

In our detection scheme we assume that the exposure time of the images is small compared with the bit period of the optical signal being transmitted. For example in

[1] One could argue that freespace optical communication dates back to classical antiquity when the invading Greeks signaled to their hidden fleet using torches once they had successfully breached the gates of Troy.

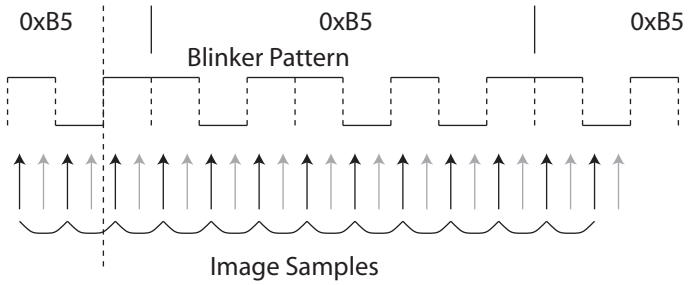


Figure 2.2: The optical intensity signal in the imager is sampled at twice the bit period which ensures that either the odd or even sample set will correctly sample the message regardless of the offset between the sampling and encoding clocks.

our current implementation the bit period is (1/6)th of a second while the exposure time of each camera is approximately 10 microsecond. This means that each pixel in the camera effectively functions as a sample and hold circuit sampling the value of the intensity signal at discrete intervals. In general, if we were to sample a binary signal with a sampling comb of the same frequency it would correctly reproduce the binary signal on almost every occasion the only exception being when the sampling comb happens to be aligned with the transitions in the binary signal. In that case since the samples are being taken while the input signal is transitioning between a high value and a low value, the resulting sample can take on any intermediate value and the decoded result will typically not correspond to a valid code. This problem can be overcome by sampling the signal at twice the bit encoding frequency. We can then divide the samples into two sets corresponding to odd and even numbered samples as shown in Figure 2.2 and can guarantee that at least one of these sets correctly samples the binary signal.

More specifically if the even set of samples happens to be aligned with the bit transitions we can be sure that the odd samples which are offset by precisely half a bit period will be safely sampled in the middle of each bit and vice versa. That

is, while one or the other of the sets of samples may be corrupted by an accidental alignment with the bit transitions at least one set of samples must sample the bit pattern cleanly.

In our implementation, as each new image is obtained it is compared with the previous odd or even frame, each pixels intensity measurement is compared with its previous value and if it has changed by more than a specified amount we push a 1 bit on a shift register associated with that pixel otherwise we push a 0 bit. By testing the change in intensity values between frames rather than the intensity values themselves we avoid setting an absolute threshold on intensity values which makes our implementation more robust to varying illumination conditions.

As an example, if the system observed the following sequence of intensity values from a given pixel { 108, 110, 113, 70, 20, 68, 98, 58, 18, 60, 105, 108, 112, 55, 12, 54, 100, 101 }, it would produce the following 8 bit values from the odd and even samples respectively 01111011 , 00000100 assuming that the sample indices start at 1 and a threshold value of 50 is used to test the changes between consecutive samples. These 8 bit patterns are then compared to the transition patterns that would result from the signal patterns that the smart camera is interested in detecting to see if a match exists. For example a blink code of 0xB5 would produce the following pattern of transitions 11011110. This decoding can be accomplished simply and efficiently by using a lookup table. Note that since the samples can start at any point in the sequence the bit transitions can correspond to any cyclic permutation of the pattern of interest. Similarly negating a given pattern produces the same sequence of transitions in the intensity measurements only inverted. These issues are easily handled by appropriately tagging all equivalent patterns of transitions in the lookup table with the same base code. In the example above, the transition pattern 01111011 would map to the code 0xB5 in the lookup table. Because of these equivalences, the number of unique codes that this recognition scheme can distinguish is on the order of $\left(\frac{2^{(n-1)}}{n}\right)$ where n denotes the number of bits in the code.

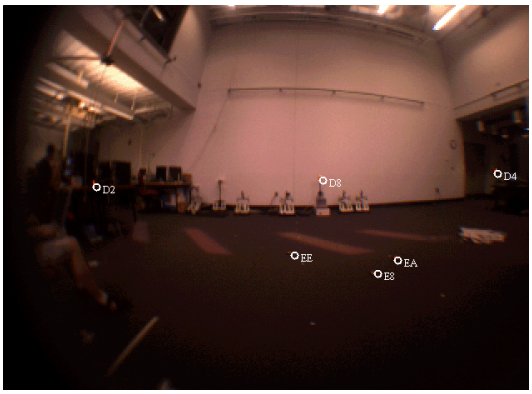
After eight even or odd samples have been acquired each additional frame adds another transition which can be used to confirm the presence of a detected code. That is, given a complete set of 8 samples one can predict what the next transition will be if the pixel is in fact exhibiting the suspected blink pattern. This means that the system can confirm the presence of a blink code by monitoring the pixel over a specified number of samples to be sure of its identity. For example if a code of 0xB5 is detected at a particular pixel based on either the odd or the even samples. The system would monitor that location for an additional 20 frames which would provide 10 more odd or even samples which should follow the proscribed pattern before the detection is confirmed. In practice this simply means that the transition patterns detected at the pixel in question should be mapped by the lookup table to the same target code over an extended sequence of frames. This is a very effective approach for removing false detections caused by spurious sampling alignments since these false detections do not recur reliably.

More sophisticated decoding schemes are certainly possible. One could, for example imagine a coding scheme which used a unique preamble to delineate the start of the bit sequence. The advantage of the scheme described here is the fact that it is amenable to real time implementation using straightforward per-pixel operations. With our current system we are able to process and decode 3 Mpix images at 12 frames per second on an embedded processor. Note that this scheme returns all of the relevant blinker patterns detected in the image so the camera can simultaneously detect multiple targets.

Figure 2.3 shows the results of the blinker detection phase on a typical image. Here the detected locations in the image are labeled with the unique codes that the system found.

Once the blinkers have been detected and localized in the images, we can derive the unit vectors, v_{ab} and v_{ba}, that relate the nodes as shown in Figure 2.1. Here we assume that the intrinsic parameters of the cameras (focal length, principal point,

distortion coefficients) have been determined previously. These parameters allow us to relate locations in the image to direction vectors relative to the camera frame.

a. 

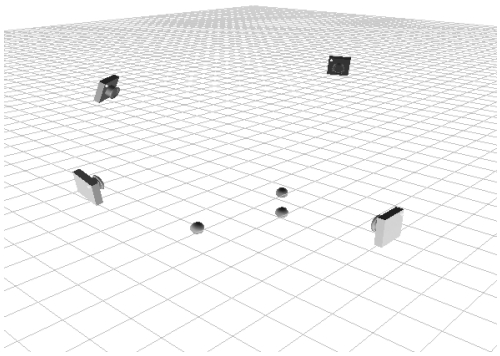
b. 

Figure 2.3: This figure shows the results of automatically localizing a constellation of 4 smart cameras and 3 blinker nodes. The image obtained from one of the smart cameras is shown in (a) while the localization results are shown in (b).

2.1.2 Recovering Orientation

Each of the smart camera nodes is equipped with an accelerometer which it can use to gauge its orientation with respect to gravity. More specifically given a unit vector $\mathbf{g}_C$ denoting the measured gravity vector in the cameras frame of reference we can construct an orthonormal rotation matrix $R_{CW} \in SO(3)$ which captures the relative orientation between the cameras frame of reference denoted by C, and a local gravity referenced frame centered at the camera denoted by W where the z axis points upwards as shown in Figure 2.4.

From the vector $\mathbf{g}_C$ we can derive a second vector $\mathbf{n}_C$ which represents a normalized version of $(\mathbf{e}_x \times \mathbf{g}_C)$ where $\mathbf{e}_x$ denotes the unit vector along the x-axis, that is $\mathbf{e}_x = (1, 0, 0)^T$. We use the vector $\mathbf{n}_C$ to define the y axis of the gravity reference frame, $\mathbf{y}_W$, in Figure 2.4. From the two perpendicular unit vectors, $\mathbf{g}_C$ and $\mathbf{n}_C$, we can construct the rotation matrix $R_{CW} \in SO(3)$ as follows:

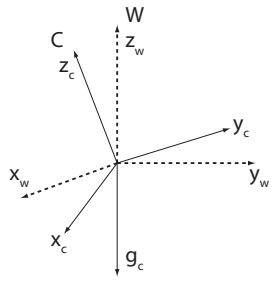


Figure 2.4: Each smart camera uses the measurements from an onboard accelerometer to gauge its orientation with respect to gravity.

$R_{CW} = \begin{bmatrix} (\mathbf{g}_C \times \mathbf{n}_C) & \mathbf{n}_C & -\mathbf{g}_C \end{bmatrix}$. Note that the columns of R_{CW} correspond to the coordinates of the x, y and z axes of the world frame in the cameras frame of reference. These equations can easily be modified in situations where the gravity vector is aligned with the x-axis.

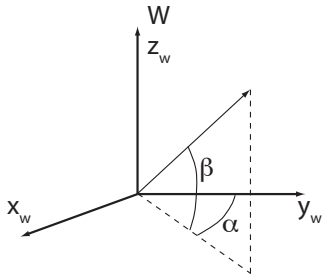


Figure 2.5: Sighting vectors in each camera can be transformed to a local, gravity referenced frame and represented in terms of azimuth, α, and elevation, β, angles.

This rotation matrix can be used to transform the sighting vectors recovered in the camera frame into the local gravity referenced world frame where they can be conveniently represented in terms of azimuth and elevation angles, α and β, as shown in Figure 2.5. More specifically, once a blinker has been detected in the image, one can use its position in the frame along with the intrinsic parameters of the camera, which are recovered in a prior calibration phase, to compute a 3D vector, $\mathbf{v}_C$, which

represents the ray from the center of projection of the camera to the blinker [2]. The rotation matrix R_{CW} can then be used to transform this vector from the cameras frame of reference to the local gravity referenced frame as follows: $\mathbf{v}_W = R_{CW}^T \mathbf{v}_C$. Equation 2.1 shows how the resulting vector, depicted in Figure 2.5, is related to the azimuth and elevation angles, α and β.

$$\mathbf{v}_W = \begin{pmatrix} \mathbf{v}_W^X \\ \mathbf{v}_W^Y \\ \mathbf{v}_W^Z \end{pmatrix} \propto \begin{pmatrix} \cos\beta\cos\alpha \\ \cos\beta\sin\alpha \\ \sin\beta \end{pmatrix} \tag{2.1}$$

These equations allow us to recover the azimuth and elevation angles from the components of the vector $\mathbf{v}_W$ as follows: $\alpha = \text{atan2}(\mathbf{v}_W^Y, \mathbf{v}_W^X)$, $\beta = \text{asin}(\mathbf{v}_W^Z)$. This change of coordinates simplifies the overall localization problem since we can use the azimuth angle measurements to localize the nodes in the horizontal plane and then recover the vertical displacements between the nodes using the elevation angles in a second phase.

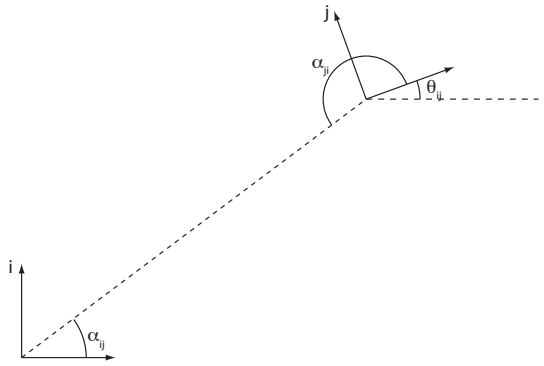


Figure 2.6: The relative yaw angle between two camera frames in the plane can easily be recovered from the measured azimuth angles if the cameras can see each other.

While we can construct a gravity referenced frame for each of the cameras from

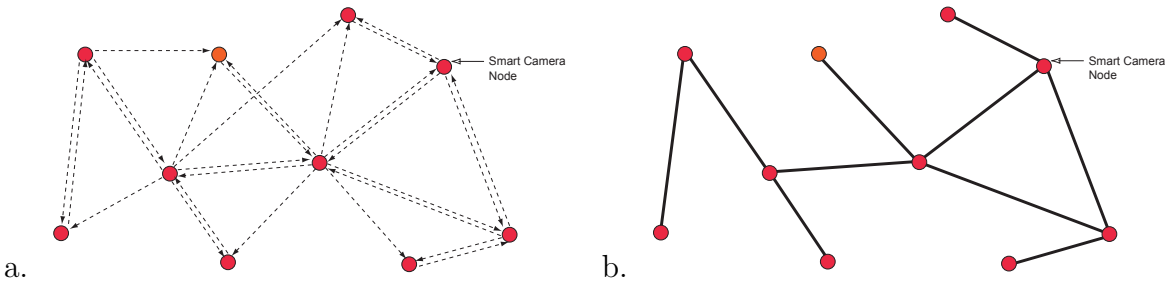
[2]This is a standard operation in many Computer Vision codes and one can find a thorough description of the procedure by consulting the Matlab Calibration Toolbox which is freely available online. See also [HS97].

the accelerometer measurements, the relative yaw between these frames is initially unknown. However, when two smart cameras can see each other as depicted in Figure 2.6 it is a simple matter to estimate their relative orientation from the available azimuthal measurements α_{ij} and α_{ji} which are related by the following equation.

$$\alpha_{ji} = \alpha_{ij} - \theta_{ij} + \pi \tag{2.2}$$

Here the parameter θ_{ij} captures the yaw angle of camera frame j with respect to camera frame i.

More generally, the visibility relationships between the smart camera nodes can be captured in terms of a directed graph where an edge between nodes i and j indicates that node i can measure the bearing to node j as shown in Figure 2.7. Any smart camera node can construct such a graph by querying its neighbors for their sighting measurements. From this directed visibility graph we can construct an undirected variant where two nodes are connected if and only if they can see one another. If there is a path between two nodes in this undirected graph, they can determine their relative orientation. This allows any smart camera node to estimate the relative orientation of its neighbors via a simple breadth first labeling.



a. b.

Figure 2.7: The visibility relationships between the nodes can be represented with a directed graph as shown on the left. If we consider only pairs of nodes that are mutually visible we end up with the undirected variant shown on the right.

Once this has been done, all of the bearing angles can be referenced to a single frame of reference, that of the root node. What remains then is to determine the

position of the nodes relative to the root. This can be accomplished by concatenating all of the available azimuthal measurements into a single homogenous linear system which can be solved using singular value decomposition (SVD).

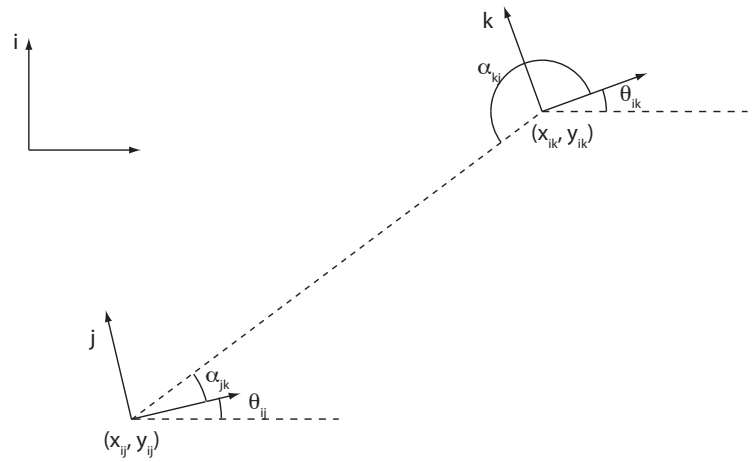


Figure 2.8: In this figure the positions and orientations of the cameras j and k are referenced to the root node, camera i. The bearing measurements α_{jk} and α_{kj} induce linear constraints on the coordinates (x_{ij}, y_{ij}) and (x_{ik}, y_{ik}) .

Consider the situation shown in Figure 2.8 where node j measures the relative bearing to node k. Since we have already recovered the relative orientation between camera frame j and the root camera node i, θ_{ij}, each bearing measurement induces a homogenous linear equation in the unknown coordinates of the following form.

$$(x_{ik} - x_{ij})\sin(\alpha_{jk} + \theta_{ij}) - (y_{ik} - y_{ij})\cos(\alpha_{jk} + \theta_{ij}) = 0 \tag{2.3}$$

Here (x_{ij}, y_{ij}) and (x_{ik}, y_{ik}) denote the coordinates of nodes j and k with respect to camera frame i. The collection of homogenous linear equations can be aggregated into a row sparse system of the form $A\mathbf{p} = \mathbf{0}$ where p is a vector with $2n$ entries formed by concatenating the coordinates of the n camera frames with respect to the root $\mathbf{p} = (\mathbf{x_{i1}}, \mathbf{y_{i1}}, \mathbf{x_{i2}}, \mathbf{y_{i2}}, ..., \mathbf{x_{in}}, \mathbf{y_{in}})^{\mathbf{T}}$. The matrix A will have one row for each bearing measurement.

Singular value decomposition can be employed to find the null space of the matrix A. More specifically, it can be used to find the vector corresponding to the minimal singular value of A or the minimum eigenvalue of $A^T A$. Because of the sparse structure, such problems can be solved efficiently using modern matrix codes even for systems involving hundreds of cameras [GL96]. This approach subsumes and improves upon earlier approaches to localizing larger collections of cameras based on repeated triangulation [TS06]. If the structure of the network cannot be completely determined from the available measurements the dimension of the null space of A will be two or more. This can be detected by considering the ratio between the smallest and second smallest singular values.

Since this linear system is homogenous we can only resolve the configuration of the nodes up to a positive scale factor. In other words, the camera systems provide us with angular measurements which allow us to perform localization via triangulation. They do not provide distance measurements directly so the overall scale of the reconstruction is undetermined. This ambiguity can be resolved with a single distance measurement, that is, knowing the distance between any two nodes in the network determines the scale of the entire constellation.

If additional position measurements are available for some of the nodes, via GPS or a prior survey, such information can easily be incorporated into the localization process. For example if (x_j^w, y_j^w) denote the easting and northing GPS coordinates of node j we can add the following two equations which relate the coordinates recovered from the homogenous system to the GPS measurements.

$$x_j^w = \lambda(x_{ij}\cos\gamma - y_{ij}\sin\gamma) + t_x \tag{2.4}$$

$$y_j^w = \lambda(x_{ij}\sin\gamma + y_{ij}\cos\gamma) + t_y \tag{2.5}$$

Where t_x, t_y and γ denote the position and orientation of the root node with respect to the geodetic frame of reference and λ denotes the overall scale parameter that relates the two frames. If we let $c = \lambda\cos\gamma$ and $s = \lambda\sin\gamma$. We end up with

two linear equations in the unknowns c, s, t_x and t_y. Given two or more such GPS measurements one can solve the resulting linear system to recover these unknown parameters and, hence, recover the geodetic locations of all of the nodes in the system.

$$x_j^w = cx_{ij} - sy_{ij} + t_x \tag{2.6}$$

$$y_j^w = sx_{ij} + cy_{ij} + t_y \tag{2.7}$$

2.1.3 Recovering Vertical Displacements

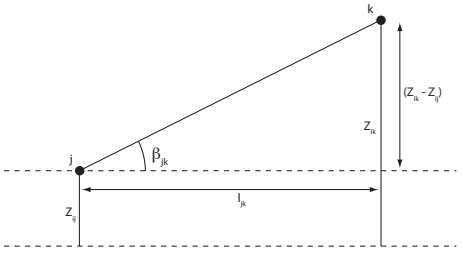


Figure 2.9: The elevation measurements induce a linear constraint on the relative heights of the nodes once the locations in the plane have been estimated.

Once the (x, y) locations of the nodes in the horizontal planes have been estimated, it is a simple matter to recover relative heights of the nodes. Figure 2.9 shows how an elevation measurement, β_{jk}, relates the heights of two nodes z_{ij} and z_{ik}. From each such elevation measurement one can construct a linear equation.

$$\frac{z_{ik} - z_{ij}}{\sqrt{(x_{ik} - x_{ij})^2 + (y_{ik} - y_{ij})^2}} = \tan \beta_{jk} \tag{2.8}$$

These constraint equations can be aggregated into a single linear system of the form $B\mathbf{z} = \mathbf{c}$ where the vector $\mathbf{z}$ represents the aggregate of all of the unknown

vertical coordinates $\mathbf{z} = (\mathbf{z_{i1}}, \mathbf{z_{i2}}, ..., \mathbf{z_{in}})^{\mathbf{T}}$. Once again the root node i defines the origin so its z coordinate is 0.

2.1.4 Refining Pose Estimates

If necessary, the estimates for node position and orientation produced by the linear process described in the preceding sections can be further refined. In this refinement step the localization process is recast as an optimization problem where the objective is to minimize the discrepancy between the observed image measurements and the measurements that would be predicted based on the estimate for the relative positions and orientations of the sensors and cameras. This process is referred to as Bundle Adjustment in the computer vision and photogrammetry literature [HZ03].

In the sequel we will let $u_{jk} \in \mathbb{R}^3$ denote the unit vector corresponding to the measurement for the bearing of sensor k with respect to camera j. This measurement is assumed to be corrupted with noise. The vector $v_{jk} \in \mathbb{R}^3$ corresponds to the predicted value for this direction vector based on the current estimates for the positions and orientations of the sensors. This vector can be calculated as follows:

$$v_{jk} = R_{ij}(T_{ik} - T_{ij}) \tag{2.9}$$

In this expression $R_{ij} \in SO(3)$ denotes the rotation matrix which relates camera frame j to the root frame i while $T_{ik}, T_{ij} \in \mathbb{R}^3$ denote the relative positions of nodes j and k.

The goal then is to select the camera rotations and sensor positions so as to minimize the discrepancy between the vectors u_{jk} and v_{jk} for every available measurement. In equation 2.10 this discrepancy is captured by the objective function $\mathcal{O}(\mathbf{x})$ where $\mathbf{x}$ denotes a vector consisting of all of the rotation and translation parameters that are being estimated.

$$\mathcal{O}(\mathbf{x}) = \sum_{\mathbf{i,j}} \left\| \mathbf{u_{ij}} - \frac{\mathbf{v_{ij}}}{\|\mathbf{v_{ij}}\|} \right\|^{\mathbf{2}} \tag{2.10}$$

Problems of this sort can be solved very effectively using variants of Newton's method. In these schemes the objective function is locally approximated by a quadratic form constructed from the Jacobian and Hessian of the objective function

$$\mathcal{O}(\mathbf{x} + \delta\mathbf{x}) \approx \mathcal{O}(\mathbf{x}) + (\nabla\mathcal{O}(\mathbf{x}))^{\mathbf{T}}\delta\mathbf{x} + \frac{\mathbf{1}}{\mathbf{2}}\delta\mathbf{x}^{\mathbf{T}}(\nabla^{\mathbf{2}}\mathcal{O}(\mathbf{x}))\delta\mathbf{x} \tag{2.11}$$

At each step of the Newton algorithm we attempt to find a step parameter space $\delta\mathbf{x}$ that will minimize the overall objective function by solving a linear equation of the form.

$$\delta\mathbf{x} = -(\nabla^{\mathbf{2}}\mathcal{O}(\mathbf{x}))(\nabla\mathcal{O}(\mathbf{x})) \tag{2.12}$$

Here we can take advantage of the fact that the linear system described in equation 2.12 is typically quite sparse More specifically, the Hessian matrix $\nabla^2\mathcal{O}$ will reflect the structure of the visibility graph of the sensor ensemble. This can be seen by noting that the variables corresponding to the positions of nodes j and k only interact in the objective function if node j observes node k or vice versa. For most practical deployments, the visibility graph is very sparse since any given camera typically sees a relatively small number of nodes as depicted in Figure 2.7. This means that the computational effort required to carry out the pose refinement step remains manageable even when we consider systems containing several hundred cameras and sensor nodes.

The optimization problem given in Equation 2.10 can be further simplified by restricting the problem to recovering the relative positions of the camera in the horizontal plane. This can be accomplished simply by projecting the bearing measurements into the plane perpendicular to the gravitational vector. In this case

Equation 2.9 would be modified, the rotation matrix R_{ij} would be an element of $SO(2)$ and the vectors T_{ik}, T_{ij} and v_{jk} would be in $\mathbb{R}^2$.

2.2 Scaling Up

The proposed linear and non-linear localization schemes which exploit the sparse structure of the relevant matrices can be used to localize hundreds of nodes at a time. However, when we consider networks of the future we may ultimately want to handle systems that cover extended areas such as the airport scenario mentioned in the introduction. Such systems may involve thousands of camera and sensor nodes which are added and removed continuously. Here it may not be feasible or desirable to have each node recover its position with respect to every other node in the ensemble. The proposed scheme can be employed to allow each smart camera node to estimate its position with respect to all of its neighbors within a specified radius. Each node then would have an estimate for the configuration for a subset of the total ensemble. This is however, sufficient to allow all of the nodes to agree on locations of salient objects via a process of coordinate transformation.

Consider a situation where camera node j wants to inform its neighbor k of the coordinates of some event. Let $R_{jk} \in SO(3)$ and $T_{jk} \in \mathbb{R}^3$ denote the estimate for the position and orientation of node k with respect to node j which is maintained by node j. Similarly let $R_{kj} \in SO(3)$ and $T_{kj} \in \mathbb{R}^3$ denote node k's estimate for the relative position of node j. Let l_{jk} denote the distance between j and k in node j's frame of reference while l_{kj} denotes the length of the same vector in k's reference frame. Notice that since the two nodes localize each other independently there is no reason that these lengths should be the same in the absence of absolute distance measurements. Given the location of a point in j's reference frame, P_j one can transform that coordinate to k's reference frame using the following expression.

$$P_k = \left(\left(\frac{l_{kj}}{l_{jk}} \right) R_{kj} P_j \right) + \mathbf{t}_{kj} \tag{2.13}$$

Here the ratio $\frac{l_{kj}}{l_{jk}}$ accounts for the change in scale factor between the two coordinate frames. Since the procedure does not require the nodes to agree on a common scale factor it can be employed even when no absolute distance measurements are available to the nodes.

These transformation can be chained so that events detected by one smart camera node can be relayed to other nodes through a sequence of transformations so that all of the events are referenced to a common frame where they can be compared and correlated.

One can imagine embedding these coordinate transforms into the communication and routing protocol so that position information is seamlessly transformed into the prevailing coordinate frame of reference as it is sent through the network.

Chapter 3

Hardware Implementation

This chapter describes three different incarnations of our implementation of a self localizing smart camera network. The first version used off-the-shelf camera components. While this approach offered a fairly rapid path towards deployment it did not offer the possibility of higher frame rates or of miniaturization. The second implementation made use of commercially available smart camera modules. This implementation offere relatively small size but could not be modified to allow for higher resolution imagers or higher performance image processing. Based on what was learned from these two earlier efforts, our third realization was based on a custom design which allowed us to acheive greater levels of flexibility, programmability and performance.

3.1 First implementation

The first implementation made use of Dragonfly firewire cameras from Point Grey Research. As shown in Figure 3.1, each camera was paired with a simple 5mm ultra bright red LED blinking with a fixed 8bit pattern at 30 HZ. Each camera could capture VGA resolution imagery (640x480) pixels at 30 frames per second. All of the cameras were connected to a laptop via a FireWire hub. With this setup we

Figure 3.1: Dragonfly Camera

were able to demonstrate self localization on a four camera network.

Early implementations of the blinker detection system made use of two thresholds to discriminate high and low intensity levels which proved to have issues in very bright or very dark environments. In this version the blinker operated at the same frequency of the cameras which led to intermittent sampling problems.

The software side of the implementation made use of a modular programming framework called ROCI [CHT04]. For each camera, a ROCI Dragonfly image capture node and a blinker detection node was instantiated. The outputs from these detection nodes were fed to a localization node, which in turn calculated the location and orientation of all the cameras up to scale.

3.2 Second implementation

This implementation made use of VCSBC50 smart camera modules manufactured by Vision Components GmbH. Each smart camer was equipped with an ADSP2185

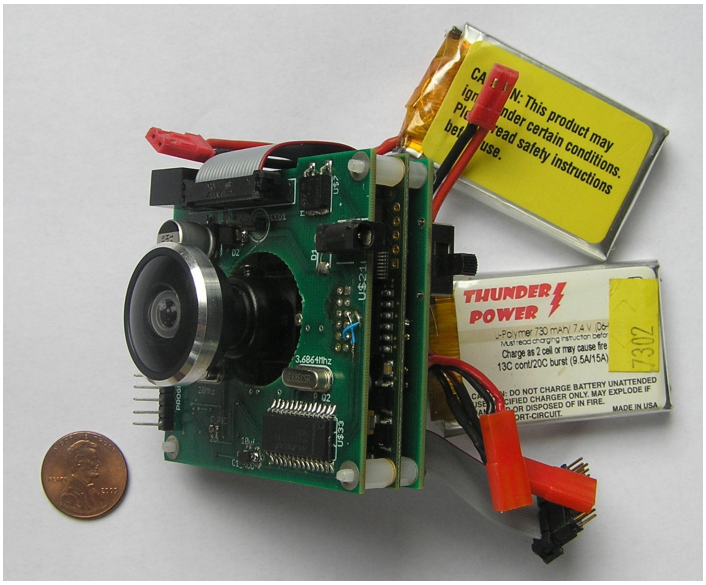


Figure 3.2: VCSBC50 Camera

Digital Signal Processor (DSP) running at 75MHz and a 1/4" CCD imager (SONY ICX098AL-6). They were also outfitted with 8Mb of DRAM and 2MB Flash-EPROM.

At startup the camera runs a simple shell that communicates with the user via a RS232 serial interface. The shell offers a number of simple commands which can be invoked from the command line. Additional routines can also be installed on the EPROM and called in a similar manner. The camera comes with a set of basic image processing libraries. While these libraries are quite extensive, they are slow and are not appropriate for real-time applications. This, to some degree, defeats the purpose of having a smart camera.

Nonetheless, with custom software we were able to succesfully implement a system that could decode 8 bit patterns at 10 frames per second on this platform. The ADSP2185 DSP has 16KB on chip data memory and a single DMA controller, therefore the image processing program spends most of the time waiting for DMA

controller. In order to communicate with camera we designed an interface board using an 8-bit PIC processor (PIC 18f2680). The interface board has an accelerometer, an RS232 to SPI bridge, a CAN transceiver, a wireless module and power management circuitry. Figure 3.2 shows a picture of VCSBC50 camera outfitted with the interface board and battery.

3.3 Third implementation

Figure 3.5 shows a picture of the current generation of the Argus Smart Camera System. Each smart camera system is powered by a dual core 600 MHz Blackfin processor from Analog Devices. This Digital Signal Processor was designed to support high performance image processing operations in low power devices such as cameras and cell phones. The smart camera board can be interfaced to a range of Aptina CMOS imagers, the configuration shown in this Figure is outfitted with a 3 megapixel imager and a fisheye lens which affords a field of view of approximately 180 degrees. The system is also outfitted with a Zigbee wireless communication module, an Ethernet controller, a three axis accelerometer and an 850 nm high intensity infrared signaling light. When properly aligned, the smart cameras can detect the infrared signaling lights at distances in excess of twenty meters.

In this realization, the center of the lens and the center of the LED array are offset by 5.5 cm. Ideally, they should be colocated. This could be accomplished by surrounding the lens with the LEDs. In practice we assume that the modeling error introduced by this offset will be negligible if the distance between the cameras is relatively large, on the order of 2 meters or more.

3.3.1 Hardware

Figure 3.3 shows the major components of our smart camera system. The BlackFin processor is connected to 64MB SDRAM and 4MB of serial flash memory. To accommodate various imagers a XC2C128 CoolRunner-II CPLD was integrated into the design. Using the CPLD, the imager port can be mapped to different IO ports on the BlackFin. The logic voltage level of the ports can also be modified. The system has an ICS307 which is a programmable clock source. This chip allows the system to generate a wide variety of clock frequency for different imagers. The PLX 2272 facilitates a full speed USB 2.0 connection to the camera. The USB connection makes it possible to use the camera like a simple USB camera. It also can be used as a debugging interface to test a BlackFin program or upload new firmware to flash memory. An 8-bit PIC microprocessor is also integrated into the camera. The PIC 18F97J60 processor has 3KB of RAM and 64KB of Flash memory. It also has an integrated 10MB Ethernet module. In this design the PIC processor is handles the communication with the onboard sensors and the wireless module. It can pass the necessary values to the BlackFin processor via an SPI connection. The main PCB is a 10 layer design with 5 mil accuracy, which was designed using the PADS CAD software package. Figure 3.4 shows an unassembled main PCB. The unit can, optionally, be equipped with a GPS receiver and/or a three axis magnetometer which would allow it to gauge its absolute position and orientation. The unit consumes less than 3 watts of power in operation and can be powered for 6 hours with a 6 ounce Lithium Ion battery pack.

3.3.2 Software

All of the programs running on the smart cameras were written in C and C++. The program running on the PIC processor was compiled using Microchip C18 compiler, and the BlackFin program was compiled using the Visual DSP system from Analog Device.

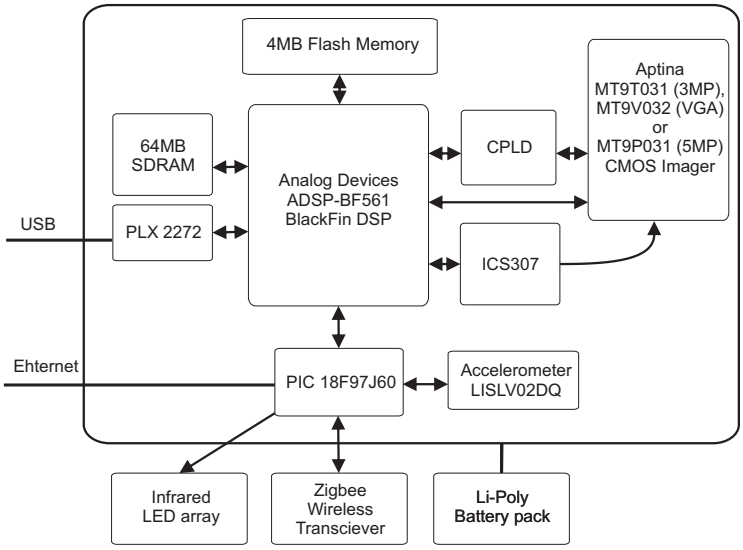


Figure 3.3: Block Diagram showing the major components of the Smart Camera node.

In order to obtain optimal performance from the BlackFin processor it was important to understand and respect the memory hierarchy on the device. Each of the two cores on the device has exclusive access to 16KB of single cycle L1 memory which is the highest level of the hierarachy. There is also 128KB of onchip memory which is shared by the two cores (L2). Finally the remaining 64MB of memory on the camera is provided by relatively slow DRAM memory (L3).

In the blinker detection algorithm, the incoming images were processed at frame rate as they were read from the image sensor into a three line round ribbon buffer in Core A L1 memory by the DMA controller. Another DMA controller copies the same line of the image from the previous frame and shift register frame within L3 into L1. Before the next line comes in, Core A must finish processing the current line and store the results in L3. Core B only deals with shift register values stored in L3. Core B runs a simple and quick connected component algorithm to detect and report the different blinker codes. All the calculation related to localization is done on Core B. The PIC processor handles the blinker, Zigbee and the accelerometer. It

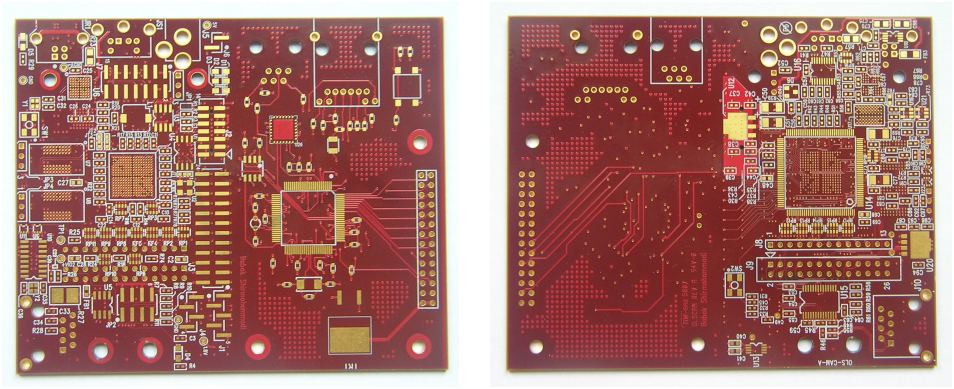


Figure 3.4: Front side of the blank PCB (left); Back side of the blank PCB

communicates with the BlackFin processor via a high speed SPI channel.

3.3.3 Calibration

Each camera is equipped with a fisheye lens. Therefore, in order to use the images, the lens distortion needs to modeled. In an ideal world the intrinsic parameters of all of the lenses would be identical and the accelerometer and image frames would be perfectly aligned. Unfortunately the vagaries of the manufacturing process dictate that each camera must be calibrated individually after it has been assembled.

Camera calibration

For our cameras, we have used an 8th order radial distortion model with no tangential distortion. The lack of tangential distortion in the model is justified by the fact that most lenses, currently available do not have imperfections in centering. The calibration parameters were calculated using a few images of a planar checkerboard. To minimize the number of images needed for this procedure we used a custom setup with a large checkerboard and placed the camera close to this target. This way, the checkerboard almost fills the entire image, which in turn cuts down the total number of images needed for calibration.

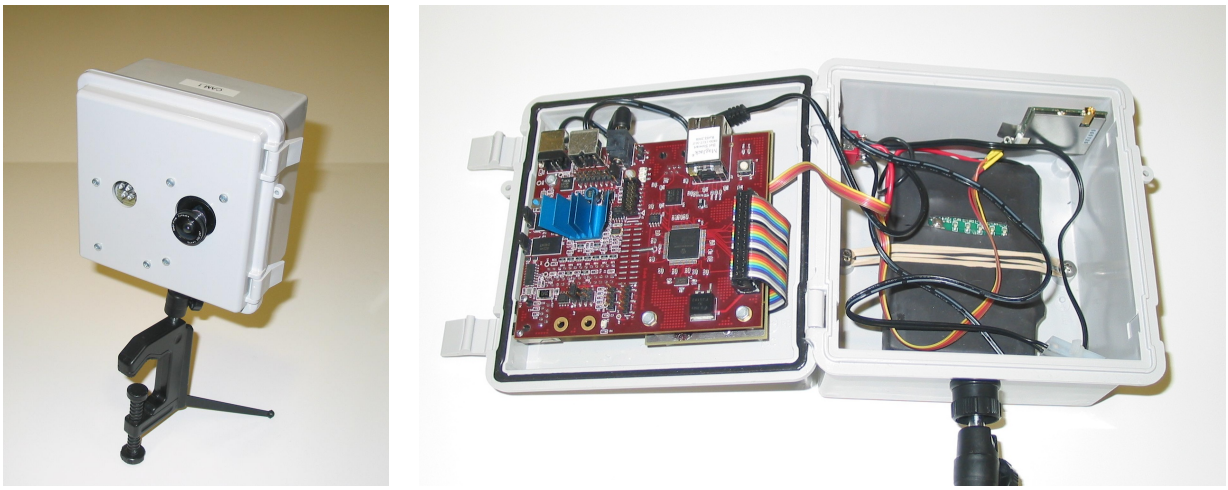


Figure 3.5: Argus Smart Camera Node used in our experiments.

Error in imager assembly process introduces a small roll angle between the image frame and the camera PCB frame. We could measure this angle by putting two blinker at same height a few meters a part and several meters away from the camera and measuring the height of each blinker in the camera frame while the camera PCB is leveled. In all of our cameras this error was insignificant compare to accelerometer accuracy.

Accelerometer calibration

Imperfections in the accelerometers and the soldering process dictate that the accelerometers need to be individually calibrated after the circuit boards have been assembled. This is accomplished by using a sensitive bubble level with an accuracy better than half a degree to level the circuit board with respect to gravity. The observedxs accelerometer measurements on the two axes nominally perpendicular to the gravitational direction are used as bias terms.

Chapter 4

Localization Experiments

In order to characterize the efficacy of the proposed localization scheme a number of experiments were carried out both in simulation and with actual hardware. Section 4.1 recounts the localization experiments that were carried out with our custom smart camera nodes. Section 4.5 describes the results of a set of simulation experiments that were designed to further characterize the behavior of the method.

4.1 Experimental Results

The smart camera nodes were deployed in an ad hoc manner in various locations in and around our laboratory facility. The linear localization and bundle adjustment process were carried out on the bearing measurements obtained from the sensors. The results of this procedure were then compared to measurements for the distances between the nodes obtained with a Leica Disto D3 handheld range finder. Because of the distances involved and the geometry of the camera nodes the errors in these ground truth distance measurements are on the order of 10 centimeters. In each experiment an appropriate scale factor was chosen to account for the scale ambiguity in the localization result.

For each of the deployment scenarios we show pictures of the environment along

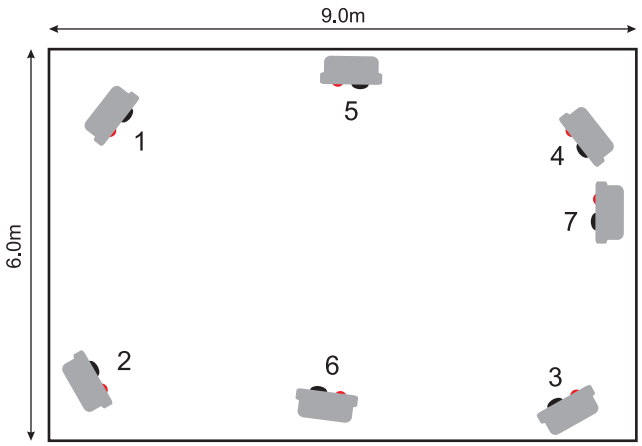


Figure 4.1: Floor Plan of the High Bay area showing the dimensions of the space and the approximate locations of the cameras

with a sketch indicating the dimensions of the space and the approximate locations of the cameras. Three dimensional renderings of the camera positions recovered by the method are presented.

4.2 High Bay

This experiment was conducted in the High Bay portion of our laboratory in an area 6.3 meters by 9 meters on side. The results obtained by the localization scheme were compared with 23 inter node distance measurements ranging from 3.62 meters to 9.98 meters. For the linear method the average absolute error in the recovered range measurements was 5.36 cm while the average relative error in the measurements was 0.91 %. After bundle adjustment the average absolute error was 6.24 cm and the average relative error was 1.05%.

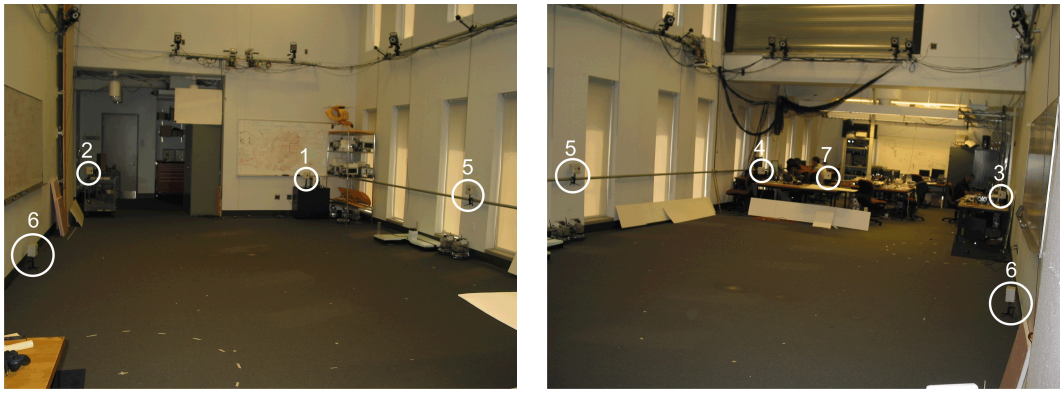


Figure 4.2: Snapshots of the High Bay area showing the deployed cameras

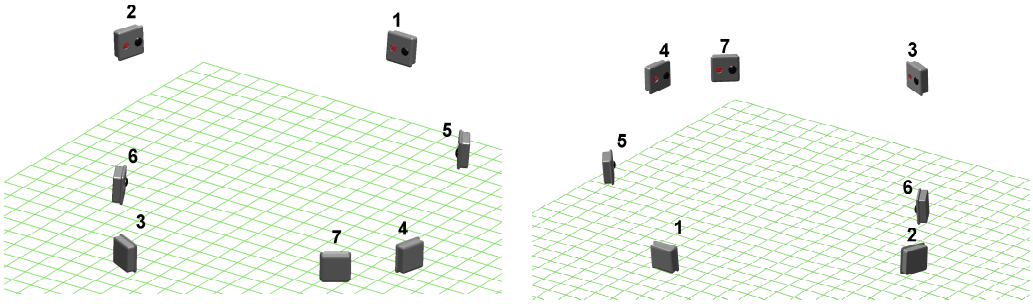


Figure 4.3: Localization results returned by the proposed localization method showing the relative positions and orientations of the nodes.

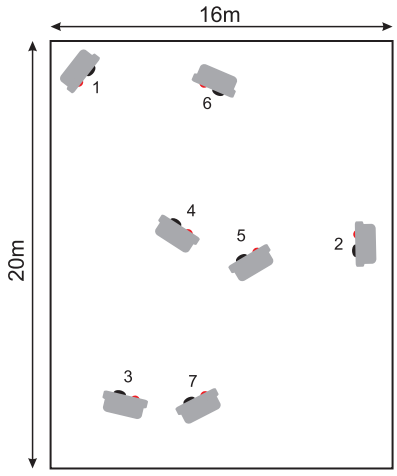


Figure 4.4: Floor Plan of the GRASP Lab area showing the dimensions of the space and the approximate locations of the cameras

4.3 GRASP Laboratory

This experiment was conducted in one of the main office areas of our laboratory in an area 20 meters by 16 meters on side. The results obtained by the localization scheme were compared with 16 inter node distance measurements ranging from 3.58 meters to 16.19 meters. For the linear method the average absolute error in the recovered range measurements was 13.17 cm while the average relative error in the measurements was 1.56 %. After bundle adjustment the average absolute error was 12.90 cm and the average relative error was 1.35%.

4.4 First Floor CS Building

In this experiment the cameras were deployed to cover the entire first floor of the Computer and Information Science building an area approximately 20 meters by 16 meters on side. The results obtained by the localization scheme were compared with

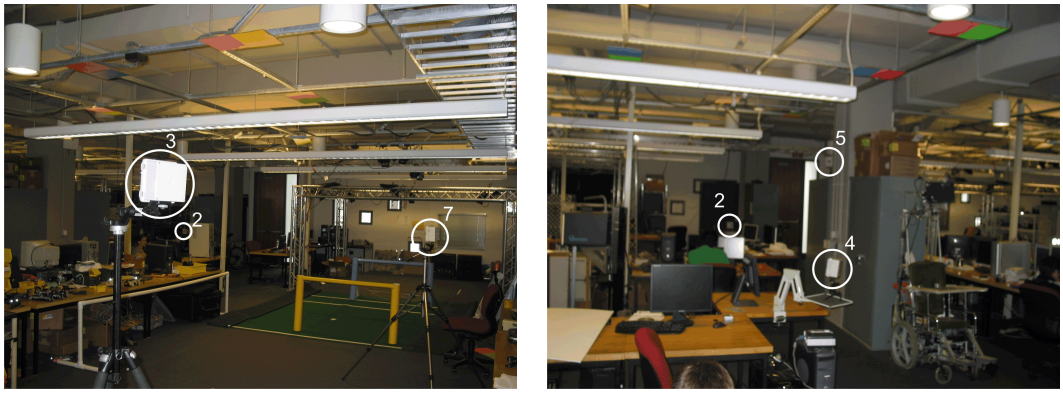


Figure 4.5: Snapshots of the GRASP Lab area showing the deployed cameras

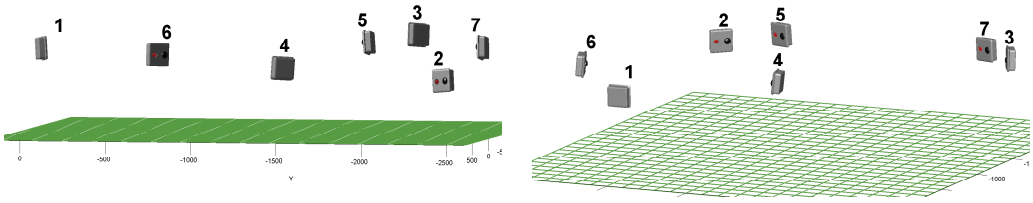


Figure 4.6: Localization results returned by the proposed localization method showing the relative positions and orientations of the nodes.

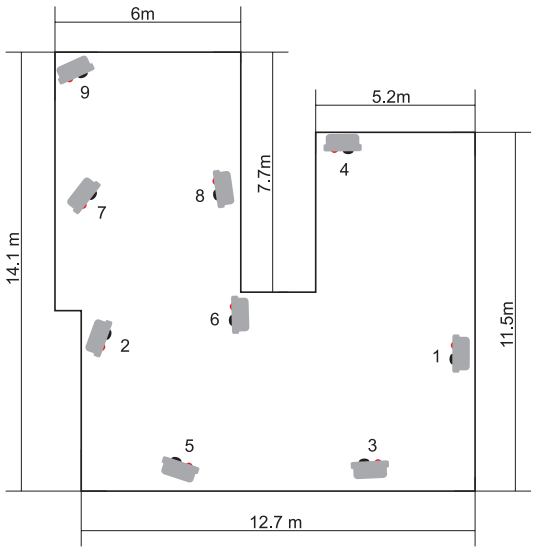


Figure 4.7: Floor Plan of the first floor area showing the dimensions of the space and the approximate locations of the cameras

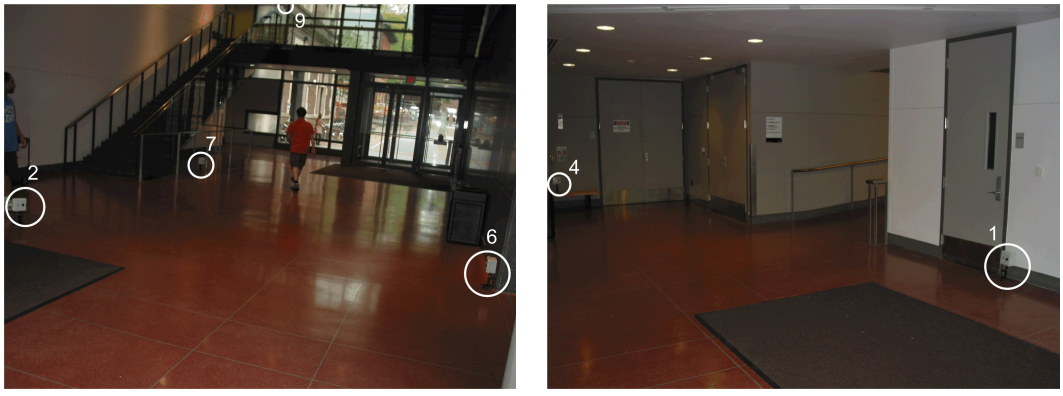


Figure 4.8: Snapshots of the first floor area showing the deployed cameras

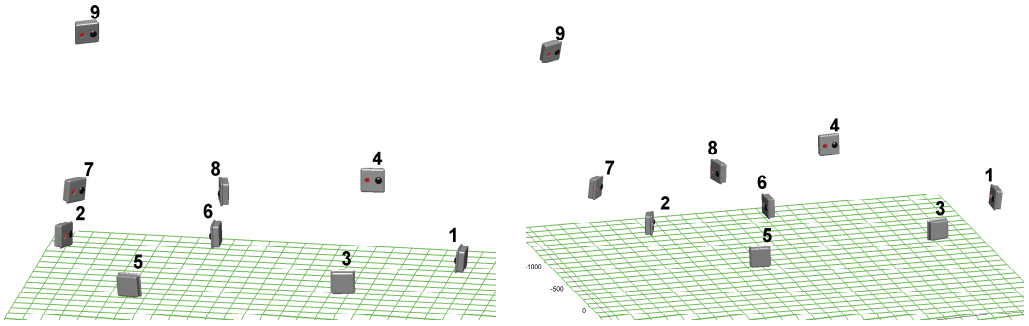


Figure 4.9: Localization results returned by the proposed localization method showing the relative positions and orientations of the nodes.

16 inter node distance measurements ranging from 5.06 meters to 17.48 meters. For the linear method the average absolute error in the recovered range measurements was 41.40 cm while the average relative error in the measurements was 4.23 %. After bundle adjustment the average absolute error was 32.75 cm and the average relative error was 3.31%. In this experiment camera 9 was actually mounted on the mezzanine overlooking the entranceway which accounts for its vertical displacement.

4.5 Simulation Experiments

A series of simulation experiments were carried out to investigate how the proposed scheme would perform on networks that were considerably larger than the ones we could construct with our available hardware. Figure 4.10 shows the basic elements of these simulation experiments. The horizontal plane was divided into a grid where the cells were unit length on side. Each grid was populated with a number of virtual smart cameras which were randomly positioned and oriented within that area. In these experiments, the number of smart cameras per cell is referred to as the camera density. Limitations on the cameras field of regard were modeled by stipulating that each camera could observe all of the cameras in its own grid cell and the adjoining

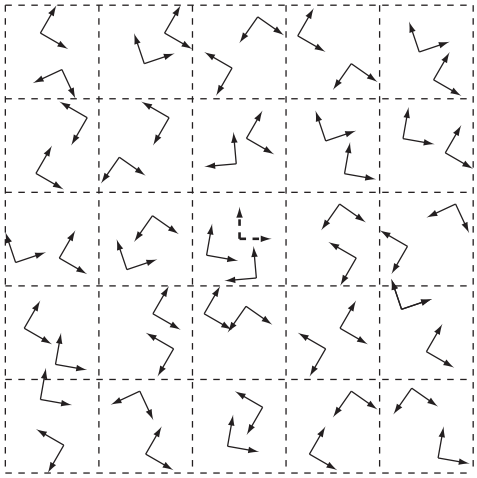


Figure 4.10: In the simulation experiments the virtual smart camera nodes were randomly placed within various grid cells in the plane. Each cell is unit length on side and the number of cameras in each cell is refered to as the camera density. One camera frame at the center defines the base frame of reference.

cells but no others. The cameras were assumed to be effectively omnidirectional so they could measure the bearing to all of the other cameras within their field of regard.

One camera was placed at the origin of the coordinate system and defines the coordinate frame of reference. The proposed localization schemes were employed to recover the positions and orientations of all of the other smart cameras with respect to this base frame. The localization was restricted to the horizontal plane since vertical displacements could easily be recovered once the horizontal locations were determined.

The bearing measurements recovered by the smart cameras were corrupted by uniformly distributed random noise. The maximum error in the bearing measurements is referred to as the bearing error, so a bearing error of 2 degrees would indicate that the measured bearing could differ from the true value by up to 2 degrees.

The first simulation experiment was designed to explore how the error in the reconstruction varied as a function of distance from the reference camera. Here

we explored various camera configurations within a 7 by 7 grid. For each trial we recorded the error in the rotational and translational error in each position estimate and segregated these errors based on distance. In Figure 4.11 the first error bar in each plot reports the mean and standard deviation of the error for cameras between 0 and 1 units from the reference camera, the second bar reports the error for cameras between 1 and 2 units from the origin and so on. The bearing error for these experiments was fixed at 2 degrees. The reconstruction procedure first recovered an estimate for the camera pose using the linear method and then refined that estimate with a bundle adjustment stage.

The graphs indicate how the rotational and translational error increase as the distance from the reference node grows. This is similar to the effect observed in robotic localization systems where small errors accumulate over time as the robot moves further from its point of origin.

These experiments were repeated for camera densities varying from 1 camera per cell up to 5 cameras per cell as shown in Figure 4.11. These plots indicate that as the camera density increases, the reconstruction error decreases. Effectively, adding more cameras to each cell increases the number of bearing measurements available and further constrains the reconstruction improving the accuracy.

The second set of experiments was designed to explore how the error in the reconstruction varied as a function of the bearing error. Several trials were carried out on a 7 by 7 grid with a camera density of 2 as the bearing error was varied from 0.5 degrees up to 3 degrees. Figure 4.12 shows how the mean rotational and translational error in the reconstruction were affected as the simulated measurement error grew.

The third set of experiments characterize the improvement afforded by the bundle adjustment phase of the reconstruction procedure. The plots on the left hand side indicate the rotational and translational error as a function of distance in the estimate provided by the linear estimation stage over several trials. The plots on the right

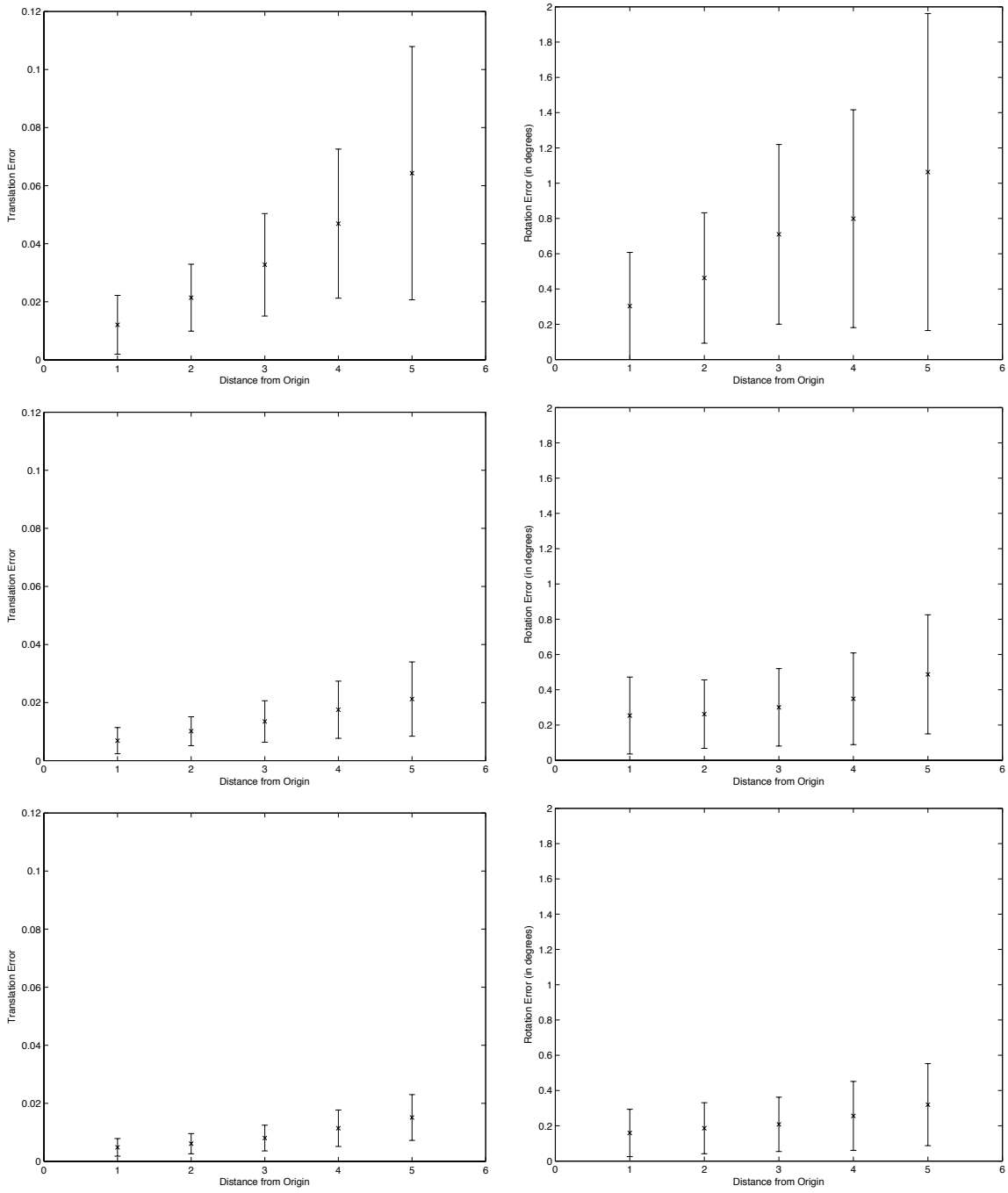


Figure 4.11: This figure shows how the position and orientation errors vary as the distance from the reference frame increases and the camera density changes. The first second and third rows of graphs correspond to camera densities of 1, 3 and 5 cameras per cell respectively. The error bars in the graph indicate the mean and standard deviation of the errors in the reconstruction.

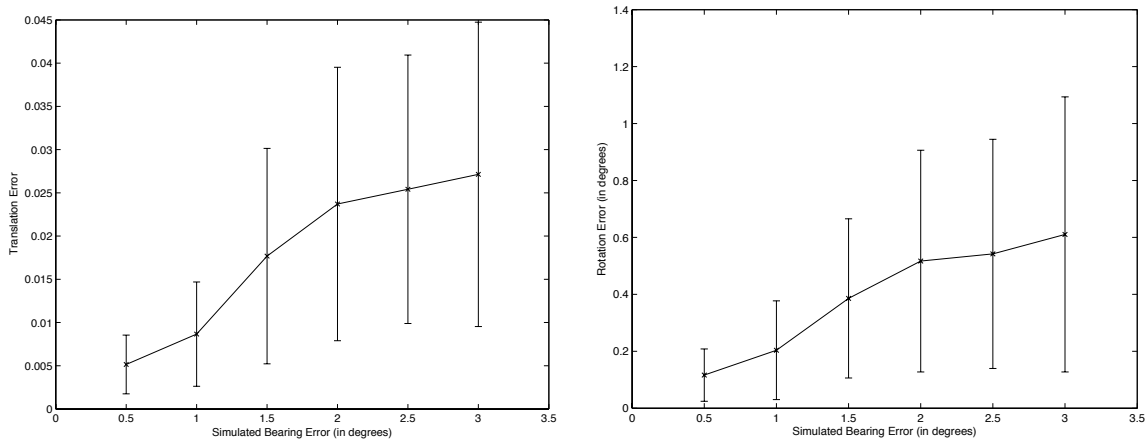


Figure 4.12: This figure shows how the position and orientation errors vary as the magnitude of the bearing error increases. The error bars in the graph indicate the mean and standard deviation of the errors in the reconstruction.

record the error after those estimates have been refined by the bundle adjustment stage. In these experiments we employed a 7 by 7 gird with a camera density of 2 and a bearing error of 2 degrees.

Figure 4.14 plots the time required to perform both the linear and bundle adjustment phases of the scheme as a function of the total number of smart cameras being localized. The procedure was implemented in Matlab and run on a MacBook Pro laptop. Note that even for 451 camera positions the time required to execute the bundle adjustment phase was under 10 seconds. The linear phase of the reconstruction is executed in under 1.5 seconds in all cases. These experiments were run with a camera density of 2 and a bearing error of 2 degrees. The number of cameras was increased by increasing the number of grid cells.

4.6 Discussion

These experimental results show that the accuracy of the proposed localization scheme compares favorably with the accuracy results reported for other distributed

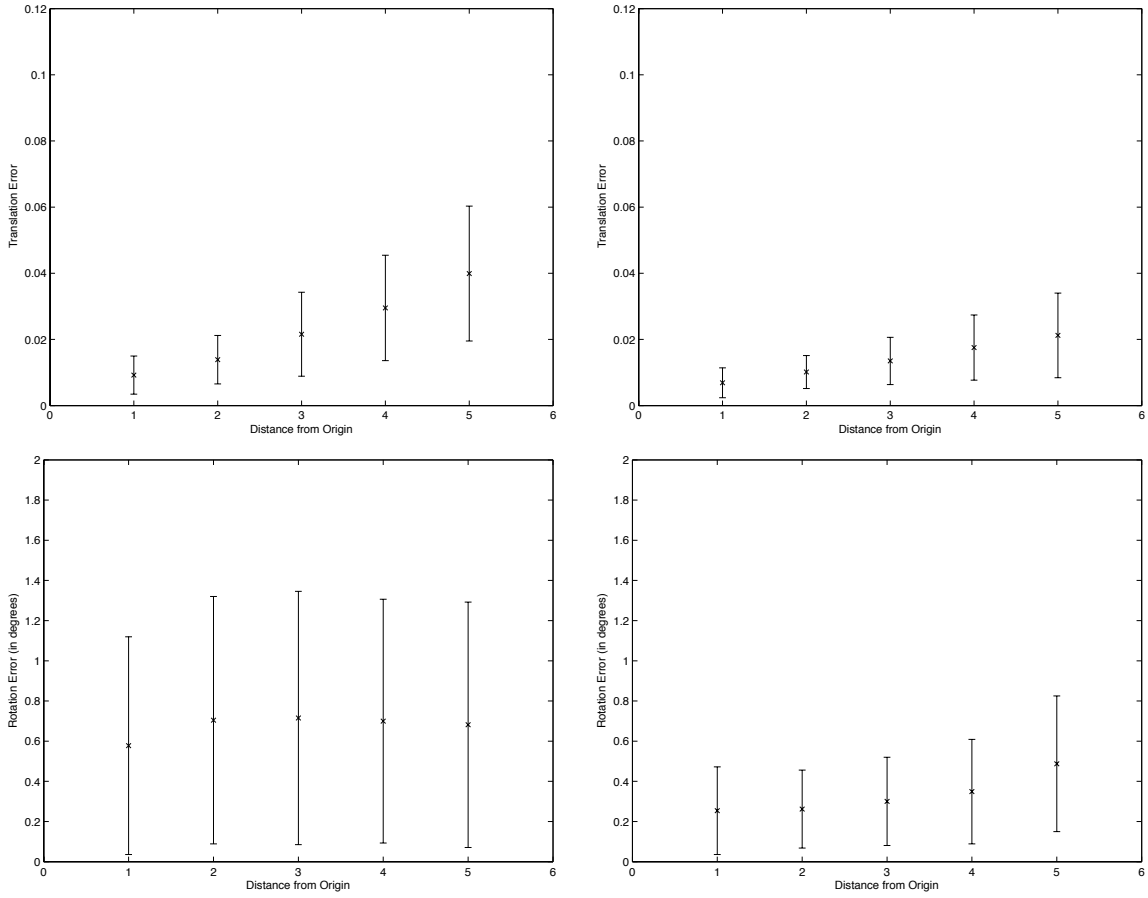


Figure 4.13: The plots on the left hand column of the figure depict the error in the pose estimates after the linear phase of the reconstruction procedure while the plots on the right depict the errors after the bundle adjustment phase.

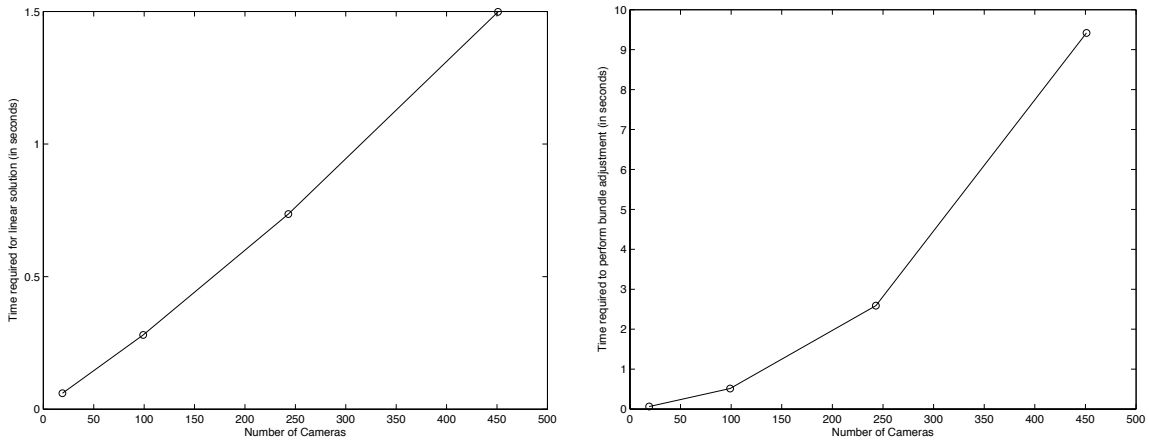


Figure 4.14: This figure shows how the time required to perform the linear and bundle adjustment phases of the localization procedure grows as the number of cameras is increased.

localization schemes on comparable problems. The simulation results provided in [DRC06] consider the problem of localizing a network of 40 cameras distributed over a circle with a radius of 110 meters. The maximum level of measurement noise considered in this work was 0.1 degrees. Under these conditions their scheme localized the cameras with a mean rotation error of 0.12 degrees and a mean translation err of 120.1 cm. In our simulation experiments the minimum noise level considered was 0.5 degrees. Even with this level of noise the proposed scheme was able to localize a network of 99 cameras distributed over a square 200 meters on side with a mean rotation error of 0.15 degrees and a mean translation error of 14.28 cm.

Funiak et al. [FGPS06] describe an experiment which involved localizing a network of 25 cameras distributed over a rectangular area of 50 square meters. Their scheme was able to localize the nodes with a root mean square error of approximately 20 cm. The scheme proposed in this thesis was used to localize a network of 6 cameras distributed over 54 square meters with an average error of 6.24 cm. The proposed scheme also improves upon the reconstruction results described in [ARD04], [TV09] and [PSF+08].

Chapter 5

Comparison to Feature-Based Localization Methods

This chapter compares the self localization scheme advanced in Chapter 2 to the state of the art feature based localization schemes commonly employed in the Computer Vision literature. The feature based approach involves extracting a set of salient features from the available images, matching these features between views to establish correspondences and then solving for the relative positions of the cameras based on these correspondences. The overall approach is described quite elegantly in the classic text by Hartley and Zisserman [HZ03].

Section 5.1 describes a series of simulation experiments which compare the accuracy and computational complexity of the feature based localization approach with the proposed self localization scheme. Based in part on these results section 5.2 provides a discussion of the issues associated with implementing the feature based method and the proposed self localization algorithm and compares these two schemes along a number of axes.

In these experiments we make use of the Sparse Bundle Adjustment (SBA) package developed by Lourakis and Argyos [LA09]. This open source package provides a state of the art implementation of a bundle adjustment procedure which exploits the

sparse structure of the Hessian of the image based objective function. More specifically the bundle adjustment procedure is optimized by partitioning the variables into two sets, one for the camera positions and the other for the feature positions, and observing that the core matrix inversion procedure in the underlying Newton optimization algorithm can then be simplified using Schur complements. The SBA package has been used as the basis for a number of successful image based reconstruction procedures including the well known Photosynth project [SSS06].

5.1 Simulation Experiments

The goal of this series of experiments was to provide a quantitative comparison of the performance of the Sparse Bundle Adjustment Package with the proposed self localization scheme on the kinds of camera configurations that would approximate those that one would encounter in extended environments. The simulated environment consists of a series of cells as shown in Figure 5.1. Each cell contains four cameras deployed along the edges of a cubic region unit length on side, the four cameras face into the cell as shown. The simulation was intended to model VGA resolution imagers with a horizontal field of view of 120 degrees and a vertical field of view of 90 degrees so the angular resolution of the simulated images was on the order of (120/640) degrees per pixel. The simulated image measurements were corrupted with random noise to simulate imaging errors.

Each cell also contains a number of simulated 3D feature points randomly distributed in the volume. The entire environment consists of an array of cells as shown in the figure. By varying the size of the array we can vary the number of cameras, the number of feature points and the extent of the simulated environment. Each camera had a simulated visibility range on the order of $\sqrt{8}$ units which meant that it could see features and cameras in some of the neighboring cells but not beyond. The intent in choosing this simulated environment was to capture the essential features

of an environment where cameras could measure information about their neighbors and where the extent of the area could grow without limit as cameras and features were added.

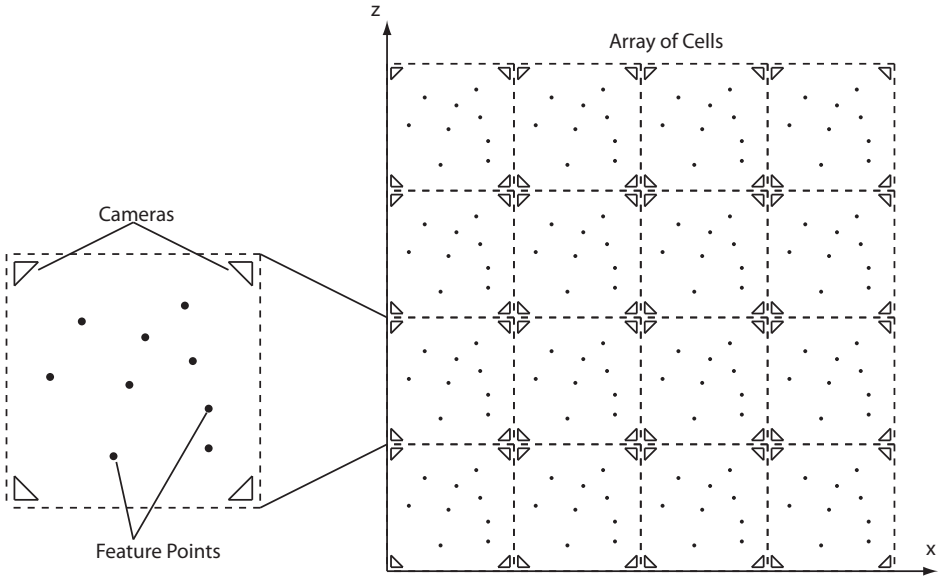


Figure 5.1: The simulated environment consists of an array of cells. Each cell contains four cameras and a number of 3D feature points. The cells are unit length on side and the radius of visibility of the cameras is set at $\sqrt{8}$ so that they can observe some of the features and cameras in adjacent cells.

The experiments compare the results obtained with the Sparse Bundle Adjustment code (SBA) with the results obtained after running the bundle adjustment phase of the proposed Self Localization scheme (SL). The intent was to compare the best results achievable with a state of the art feature based localization scheme, SBA, with the best results that could be obtained with the self localization scheme, SL.

A series of simulation experiments was carried out. The first one characterized how the accuracy and computational complexity of the localization schemes changed as the number of cameras and number of features was varied. Another experiment

investigated how the accuracy of the methods changed as the error in the image measurements was varied. The third experiment characterized how the methods fared as the error in the initial relative orientation measurements was increased.

5.1.1 Initial Estimates

Both localization schemes are phrased as non-linear optimization procedures based on Newton's method, as such both schemes require initial estimates for the camera positions. Initial estimates for the camera orientations were obtained from a procedure which simulated the action of a global optimization procedure which took as input estimates for the relative orientation of mutually visible cameras and produced estimates for the camera orientations that were in best agreement with those measurements. The camera in the lower left hand corner of the array was held fixed to define the base frame of reference for the localization procedure.

Given these initial estimates for camera orientation one can derive initial estimates for the camera positions and feature positions by solving a set of sparse linear systems derived from the image measurements.

More specifically if we let $\mathbf{u}_{ij} \in \mathbb{R}^3$ denote the sighting vector of feature j as seen from camera i. Equation 5.1 shows how we expect that vector to be related to the position and orientation of camera i, ($R_i \in SO(3)$, $\mathbf{t}_i \in \mathbb{R}^3$), and the position of feature j, $\mathbf{P}_j \in \mathbb{R}^3$.

$$\mathbf{u}_{ij} \propto R_i(\mathbf{P}_j - \mathbf{t}_i) \tag{5.1}$$

$$\Rightarrow \mathbf{u}_{ij} \times \{R_i(\mathbf{P}_j - \mathbf{t}_i)\} = \mathbf{0} \tag{5.2}$$

Note that given an initial estimate for the camera orientation, R_i, and the image measurement vector, $\mathbf{u}_{ij}$, Equation 5.2 is *linear* in the position vectors, ($\mathbf{P}_j$ and $\mathbf{t}_i$). This means that these equations can be consolidated into a single sparse, homogenous

linear system which can be solved to provide initial estimates for the camera and feature positions.

5.1.2 Experiment 1

The first simulation experiment was carried out to characterize how the accuracy and computational complexity of the SBA and SL methods varied as we changed the number of cameras in the simulation and the number of feature points. To accomplish this the number of rows and columns in the simulated arena was varied as was the number of feature points in each cell. Ten trials were carried out for each experimental condition and the results of these trials were averaged to produce the results shown below in the tables and graphs. In these simulation experiments the number of rows and columns in the simulated array of cells was always the same, that is nrows = ncols = n, so the number of cameras in any trial is given by $4n^2$. The maximum error added to the image measurements was fixed at 0.5 pixels.

Figures 5.2 and 5.3 indicate how the error in the camera rotation estimates and camera translation estimates varied as we varied the size of the array, n and the number of features in each cell. Here it is useful to recall that each cell or block in the array is unit length on side so a translation error of 0.1 would correspond to an error in the camera position that was on the order of 10% of the width of a cell.

The first four bars in each group correspond to different trials of the SBA method as the number of feature points in each cell varies from 5 to 10 to 20 to 40. Generally speaking, as the number of features is increased more image measurements are available to the optimization procedure and this tends to improve the accuracy of the SBA measurement; this is borne out in the graphs. Changing the number of features has no effect on the SL method since it only utilizes sightings of other cameras. The results are summarized in Tables 5.1 and 5.2 .

These results indicate that the SL method consistently produces estimates for camera rotation that are comparable to those produced by the SBA method and

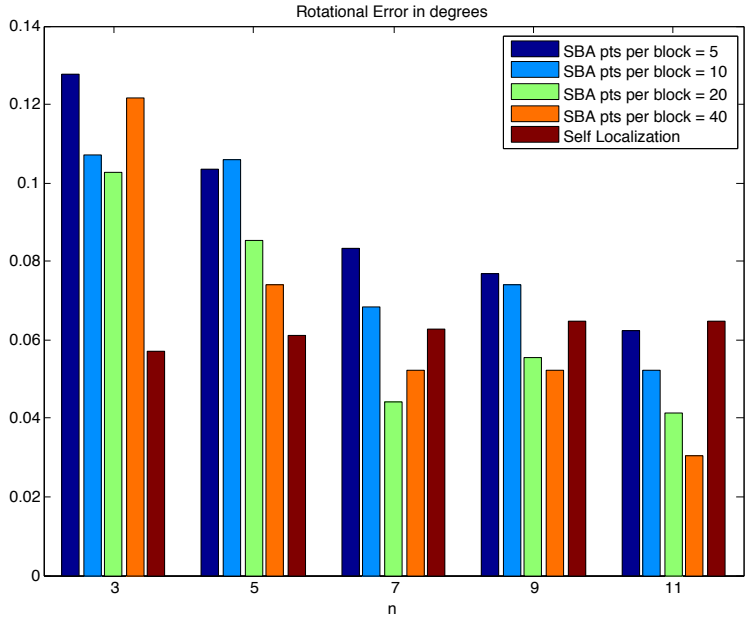


Figure 5.2: Figure showing how the rotational error in the recovered camera positions varies as the size of the environment and number of cameras is changed, n, and the number of feature points is changed.

n	SBA points in each cell				SL
	5	10	20	40	
3	0.1279	0.1070	0.1029	0.1217	0.0571
5	0.1035	0.1061	0.0854	0.0740	0.0609
7	0.0833	0.0682	0.0441	0.0522	0.0626
9	0.0770	0.0739	0.0555	0.0524	0.0646
11	0.0622	0.0521	0.0413	0.0303	0.0646

Table 5.1: Rotational Error in reconstructed camera positions in degrees

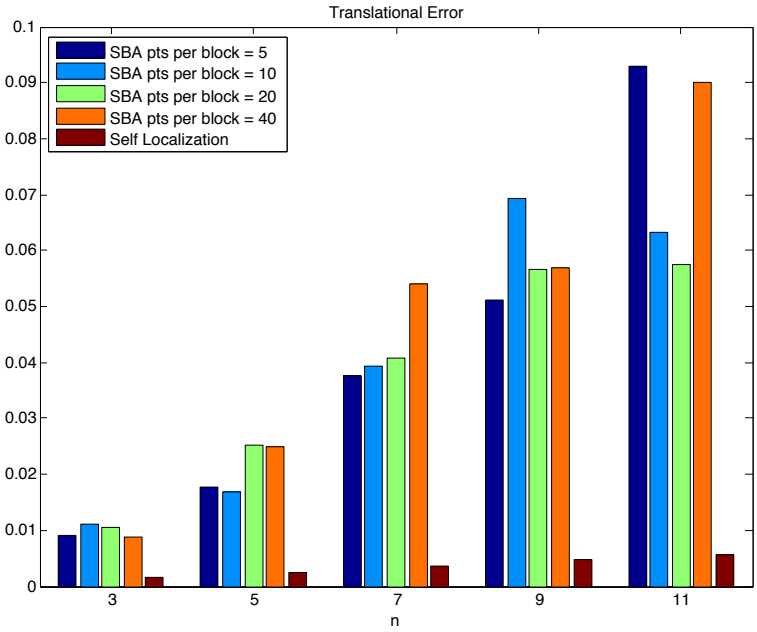


Figure 5.3: Figure showing how the translational error in the recovered camera positions varies as the size of the environment and number of cameras is changed, n, and the number of feature points is changed.

n	SBA points in each cell				SL
	5	10	20	40	
3	0.0089	0.0112	0.0106	0.0089	0.0016
5	0.0178	0.0167	0.0254	0.0250	0.0023
7	0.0375	0.0394	0.0407	0.0541	0.0036
9	0.0512	0.0694	0.0565	0.0570	0.0047
11	0.0930	0.0632	0.0576	0.0901	0.0057

Table 5.2: Translational Error in reconstructed camera positions

estimates for camera position that are 5 to 10 times more accurate than those produced by the SBA algorithm. This difference can be attributed to the fact that the measurements of the camera centers, provides a direct constraint on the camera positions while the SBA method recovers the camera locations indirectly from the feature correspondences.

The results also show that the average error in the reconstructed camera positions increases as the number of cameras and the extent of the simulated area increases. To probe this effect further Figure 5.4 and Figure 5.5 show how the rotational and translational errors increase as a function of the distance of the simulated camera from the lower left hand camera which is fixed as the origin of the reconstructed reference frame. These plots demonstrate that cameras that are further away have a larger translational error which accounts for the rise in average error as the size of the simulation is increased.

This phenomenon can be explained by considering that the image measurements that are used to deduce the camera locations provide information about the *relative* position of the features and the cameras as opposed to the *absolute* camera positions. As such the locations of each camera with respect to the base camera is effectively recovered from a chain of measurements, as the distance from the origin grows, the size of this chain increases and the errors in the estimates accumulate.

Figure 5.6 shows how the number of image measurements utilized by the localization procedure changes with the number of cameras and the number of feature points. The grouping of the bars is the same as in Figures 5.2 and 5.3. As one would predict the number of measurements used by the SBA method increases linearly with both the number of cameras and the number feature points per block while the number of measurements used by the SL method increases more slowly and only depends on the number of cameras being localized.

Figure 5.7 compares the running time of the SBA method to that of the SL method. Here it is important to note that the SBA measurement was written in

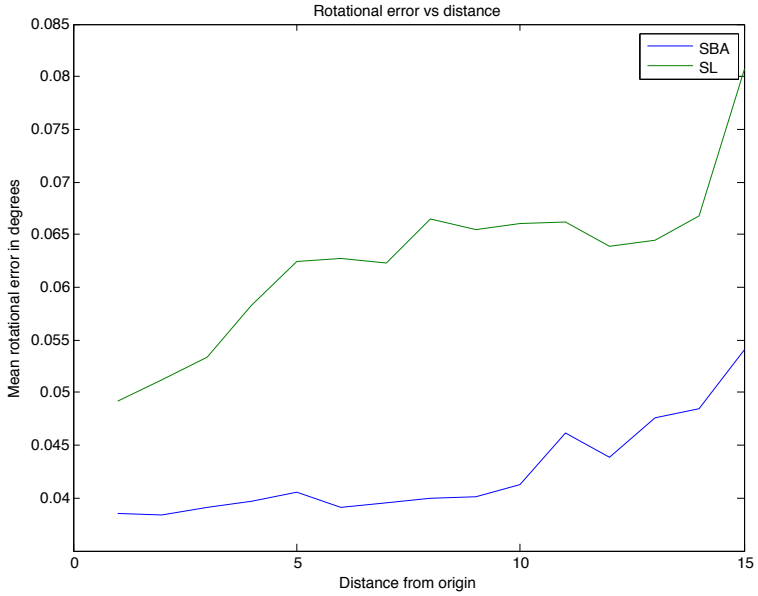


Figure 5.4: Figure showing how the rotational error in the recovered camera positions varies as the distance from the origin is increased.

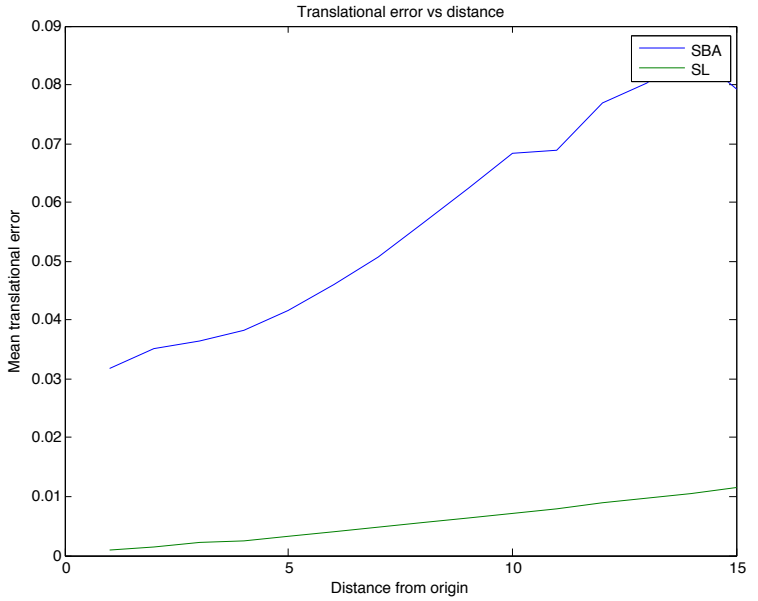


Figure 5.5: Figure showing how the translational error in the recovered camera positions varies as the distance from the origin is increased.

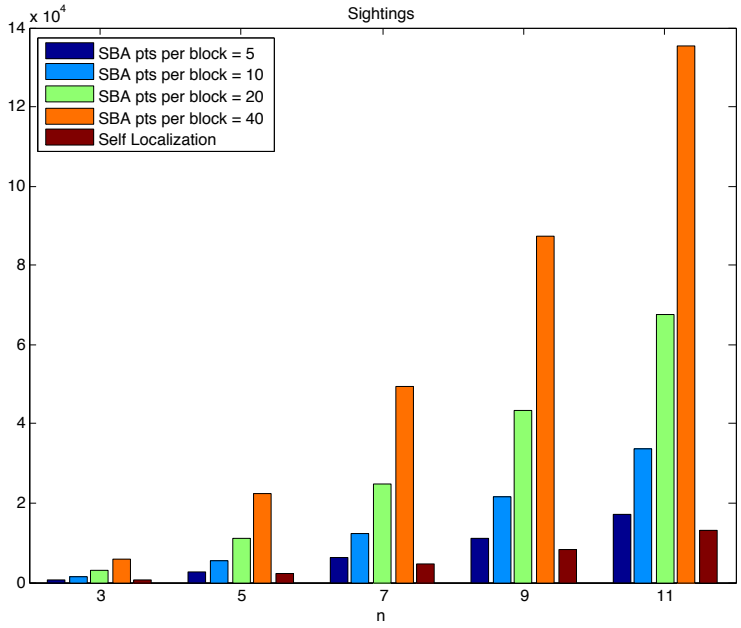


Figure 5.6: Figure showing how the total number of image measurements used by the reconstruction schemes varies as the size of the environment and number of cameras is changed, n, and the number of feature points is changed.

n	SBA points in each cell				SL
	5	10	20	40	
3	0.0470	0.0640	0.1010	0.1950	0.2995
5	0.3630	0.5340	0.8260	1.5860	0.9371
7	2.0990	2.3860	3.2180	4.7880	2.2524
9	9.7770	10.3900	12.3810	16.5780	4.5587
11	32.9720	32.1630	36.1380	42.8850	8.2339

Table 5.3: Average running time required by localization algorithms in seconds

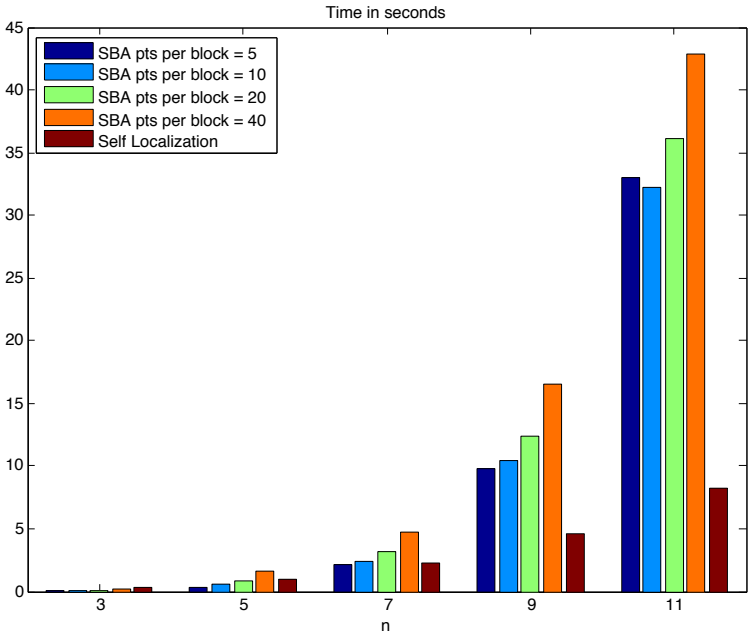


Figure 5.7: Figure showing how the total running time of the reconstruction schemes varies as the size of the environment and number of cameras is changed, n, and the number of feature points is changed. Note that the SBA scheme is implemented in optimized C while the SL method is implemented in Matlab, nonetheless, it manages to outperform the SBA scheme as the number of cameras is increased

C and then compiled with full optimization to produce a final executable while the SL method was implemented in Matlab. Nonetheless, the results show that when n is increased to 7 or more, which corresponds to the number of cameras being 196 or more, the SL method significantly outperforms the SBA method and the gap grows larger as the number of cameras is increased. Consider for example the case where $n = 11$ which simulates a situation with 484 cameras, here the SL method is 5 times faster than the SBA method. One would expect that a more heavily optimized implementation of the SL method would perform even better. The timing numbers are summarized in Table 5.3, all timings were carried out on a quad core Intel i7 laptop.

From these experimental results we can conclude that as the number of cameras is increased the SL method produces more accurate estimates for the camera configuration from fewer measurements and with less computational effort.

5.1.3 Experiment 2

The second simulation experiment was designed to investigate how the error in the camera position estimates returned by both methods varied as the amount of error in the input image measurements was increased. The angular resolution of the simulated camera was fixed at (120/640) degrees per pixel and the maximum amount of error added to the measurements was increased from 0.5 pixels to 3 pixels. The size of the cell array was fixed at 7 by 7 so there were 196 cameras in each trial. The number of points in each cell was fixed at 20. Twenty trials were performed for each experimental condition and the results were averaged to produce the following plots.

The results from these experiments are plotted in Figures 5.8 and 5.9. These plots show that the rotational estimate from the SL method is more affected by the increasing image error than the SBA method. This is probably due to the fact that the camera orientation estimates are more directly derived from those measurements. The translational error of the SL method is consistently lower than that of the SBA

method in all cases.

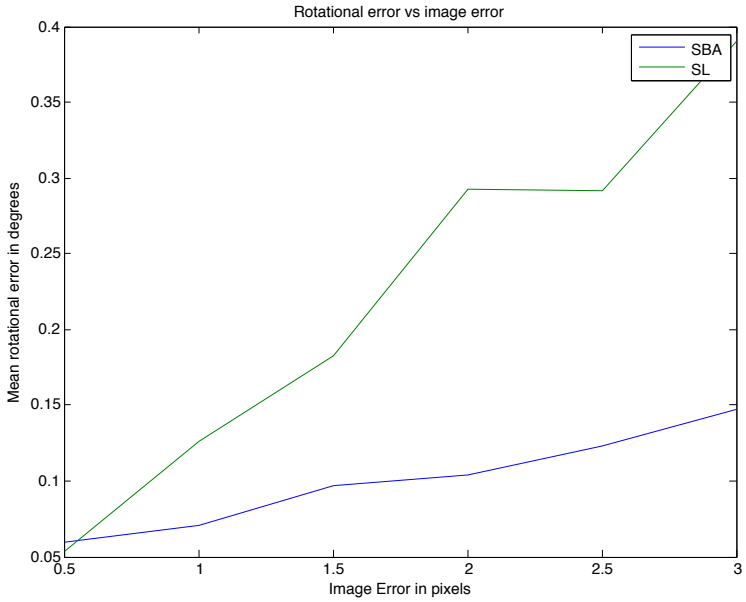


Figure 5.8: Figure showing how the rotational error in the recovered camera positions varies as the amount of error in the input image measurements is changed.

5.1.4 Experiment 3

The third simulation experiment was designed to explore how the SBA and SL methods performed as the error in the initial estimates for the camera rotations was increased. This was accomplished by varying a parameter which represented the error in the relative orientation measurements between the frames. This value was increased from 1 degree to 4 degrees. Recall that the initial orientation for each camera is obtained by simulating a procedure that recovers an optimal estimate for the camera rotations from the relative orientation measurements between pairs of cameras. Once again the size of the cell array used in the experiments was fixed at 7 by 7 and the number of points in each cell was fixed at 20. Twenty trials were performed for each experimental condition and the results were averaged to produce

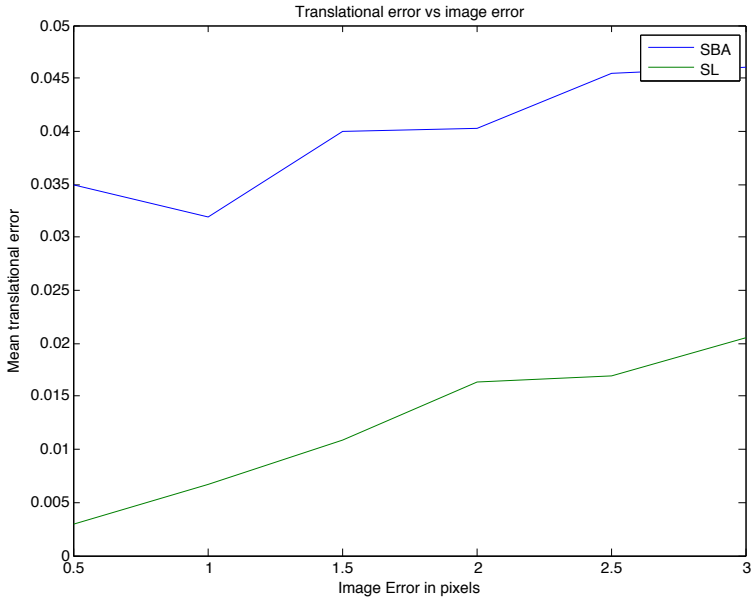


Figure 5.9: Figure showing how the translational error in the recovered camera positions varies as the amount of error in the input image measurements is changed.

image error in pixels	SBA	SL
0.5	0.0593	0.0537
1.0	0.0710	0.1258
1.5	0.0967	0.1823
2.0	0.1037	0.2925
2.5	0.1227	0.2920
3.0	0.1478	0.3899

Table 5.4: Table showing how average rotational error in the recovered cameras estimates in degrees varies as the image errors are increased.

image error in pixels	SBA	SL
0.5	0.0349	0.0030
1.0	0.0320	0.0067
1.5	0.0399	0.0109
2.0	0.0402	0.0164
2.5	0.0454	0.0170
3.0	0.0461	0.0206

Table 5.5: Table showing how average translational error in the recovered cameras estimates varies as the image errors are increased.

relative orientation error in degrees	SBA	SL
1.0	0.0621	0.0610
2.0	0.1203	0.0642
3.0	0.1664	0.0625
4.0	0.1841	0.0704

Table 5.6: Table showing how average rotational error in the recovered cameras estimates in degrees varies as the error in the relative orientation measurements is increased.

the following plots.

These results show that the SL method is relatively robust to the error in the initial rotation estimate. The errors in the final estimates increase only very slowly as the errors in the initial estimates are increased. The SBA method does not fare as well, both the rotational and translational error increase significantly with the initial rotation error. This seems to indicate the difficulty associated with converging to the correct answer based on the relative sighting measurements that are available to the method. In this case it appears that there are a number of estimates for the camera configuration that are equally plausible given the available measurements.

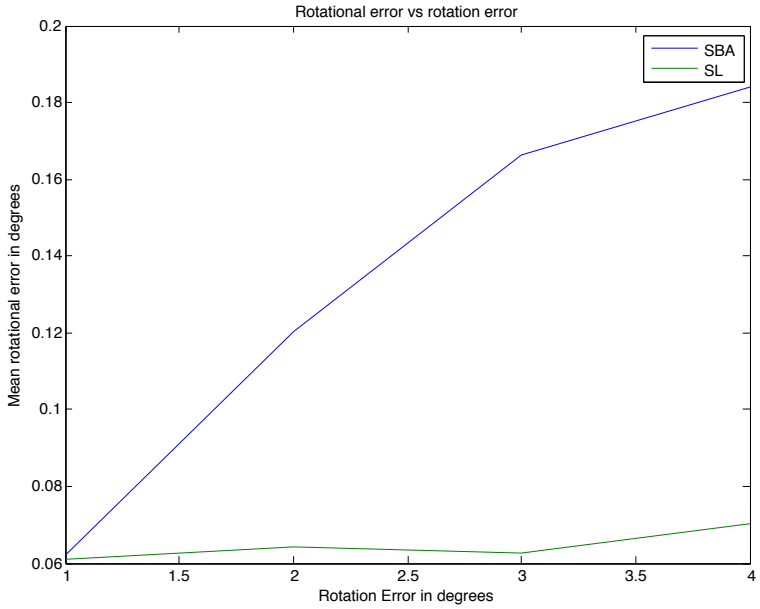


Figure 5.10: Figure showing how the rotational error in the recovered camera positions varies as the rotational error in the initial relative orientation estimates is increased.

relative orientation error in degrees	SBA	SL
1.0	0.0337	0.0034
2.0	0.0981	0.0040
3.0	0.1859	0.0035
4.0	0.2917	0.0038

Table 5.7: Table showing how average translational error in the recovered cameras estimates varies as the error in the relative orientation measurements is increased.

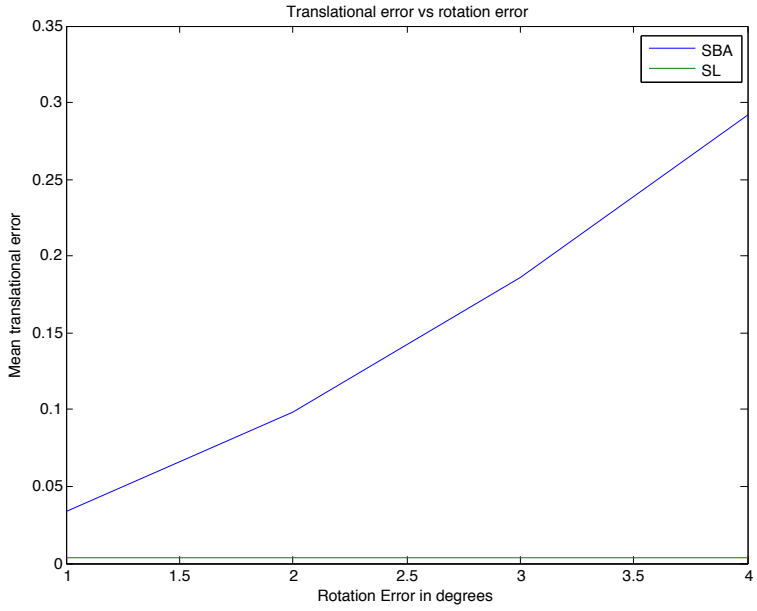


Figure 5.11: Figure showing how the translational error in the recovered camera positions varies as the rotational error in the initial relative orientation estimates is increased.

5.2 Comparison of Localization Schemes

The goal of this section is to compare and contrast the issues associated with implementing a smart camera localization scheme based on the classical feature based method using SBA and the self localization approach advocated in this thesis. Recall that the feature based method proceeds in a series of stages, first one extracts point features in each of the images then one matches these features between views to obtain correspondences, these correspondences are then supplied to a localization procedure which computes an estimate for the configuration of the cameras and the locations of the feature points. The self localization approach is similar in that it also involves extracting correspondences from the imagery and using these correspondences to recover the camera configuration but the approach taken to extracting features, the measurements used and the localization scheme are all different.

One can compare the feature based method with the self localization scheme along a number of relevant axes. One can consider the relative complexity of the feature extraction and matching stages which produce the correspondences. One can consider the implied communication requirements of the schemes, one can look at the computational complexity of the localization procedures and one can compare the accuracy of the estimates returned by the two schemes.

All feature based localization schemes begin with a feature extraction and matching stage. SIFT and SURF features are very popular in this context since they provide descriptors which can be used to reliably match features across views even when the features are rotated, translated and scaled. The price for this invariance is that the feature extraction methods are typically quite complex and the feature descriptors that they provide can be fairly large. Typical implementations such as Photosynth [SSS06] spend a large amount of computational resources on this phase in order to extract as many correspondences as possible so as to improve the final reconstruction results.

Once the features have been extracted from each image the resulting feature

descriptors are usually transmitted to a central location where they can be matched to features extracted in other views. Sophisticated data structures based on k-d trees or randomized hashing are often employed to optimize this procedure and avoid the cost of matching every feature to every other feature. Nonetheless the cost of gathering, storing and matching features from hundreds or thousands of cameras requires significant computational resources. In a distributed context this method involves having each camera transmit lists of feature vectors to a central server or to its neighbors. The cost of this communication can become quite significant as the number of cameras increases.

One of the goals in the design of the proposed self localization scheme was to develop a method that would be amenable to implementation on distributed, low-power embedded processors. To that end the feature extraction method is based on a temporal analysis of the pixels in each frame. This procedure can be readily implemented using frame differencing. Moreover the approach yields the identity of the camera being viewed which obviates the need for a matching procedure. This feature extraction process produces a much smaller list of sightings since one only reports on other smart cameras in the field of view which lowers the amount of information that each smart camera must transmit.

Experiment 1 considered the computational complexity of the SBA and SL camera localization schemes. These results show that the self localization scheme requires less computational effort and that the difference between the two schemes becomes more pronounced as the number of cameras is increased. This difference can be attributed to the fact that the SBA system must solve a larger problem involving both the camera locations and the feature positions. Although the SBA system cleverly exploits the sparsity structure of the underlying measurement system, it still requires more effort than the self localization scheme which only involves the camera location parameters and can work with a measurement system that is smaller and sparser than the one used by SBA.

Experiment 1 also demonstrates that the Self Localization scheme can be expected to produce more accurate camera position estimates than the SBA system. Here again the fact that the feature based method must recover the camera positions indirectly from feature correspondences hampers the resulting algorithm. Even though the Self Localization scheme uses fewer measurements than the feature based scheme these measurements are more useful since they directly reveal the epipolar structure of the camera system and, thus, directly constrain the camera positions.

Taken together, the experimental results indicate that the proposed self localization scheme requires less computational effort to extract correspondences, requires fewer image measurements and, thus, less bandwidth to communicate those measurements and produces more accurate results with less computational effort than the SBA method. From this we conclude that the method offers a number of advantages over current state of the art feature based methods and is particularly well suited for large scale smart camera deployments which may involve hundreds or thousands of distributed cameras where computational and communication resources must be carefully husbanded.

Chapter 6

Tracking

6.1 Introduction

It is natural to consider applying smart camera networks to the problem of tracking people, vehicle or animals as they move over extended environments. Recalling the motivating example from the introduction, we could consider the problem of developing a system to track passengers at an airport from the time they arrive at the facility until the time that they board their flight. Similarly, we could use such a system to monitor the movements elderly or infirm individuals in their homes in order to improve their quality of care.

In order to achieve our vision of a robust situational awareness percept derived from an ensemble of distributed cameras, we will need to address the problem of distributed sensing and tracking. More specifically, the challenge will be to reliably detect, localize and track targets as they move over an extended area of regard covered by multiple distributed smart cameras.

In order to field these kinds of systems we will need to develop approaches to detection and tracking which can be distributed over multiple sensors without requiring excessive amounts of communication. These systems must be scalable to allow for deployments that may involve thousands of cameras distributed over extended

regions and must be robust to failure so that the overall system responds gracefully when individual sensors are added or removed asynchronously.

Most detection and tracking systems that have been developed or proposed fuse information from multiple sensors at a central point in the network which is responsible for establishing tracks and associating measurements from different views. As the number of sensors grows, increasing demands are placed on the communication system which must route information to these processing centers. Moreover failures in these processing centers can often render the entire network useless.

In this thesis we describe a new approach to detection and tracking for smart camera networks which is fundamentally decentralized. This approach builds upon the self localization scheme described in chapter 2. We have developed novel network protocols with limited communication requirements which allow the system to distribute detection and tracking problems evenly through the network accounting for sensor handoffs in a seamless manner.

The approach also distributes knowledge about the state of tracked objects throughout the network. This information can then be harvested through distributed queries which allow network participants to subscribe to different kinds of events that they may be interested in. For example a process could request to be updated on all movements of a particular target or may want to be told about all targets that pass through a particular area of interest. These approaches can be used to develop simple but robust tracking systems that respect the constraints of a distributed deployment context.

6.2 Related Work

Distributed tracking on smart camera systems has attracted a lot of recent attention and a number of groups have developed systems for this task [QMK+08, SSRCF08, KCA+07, SFP06, FBBS06, GD06, BALaDO+06]. For example Kayumbi, Anjum

and Cavallaro [KAC08] describe an effective scheme for localizing soccer players with a network of distributed cameras. Quinn et. al. [QMK+08] propose a scheme for calibrating a set of cameras in a room and using them to track targets. Their approach splits the tracking task between a collection of smart camera nodes and a higher level process which fuses the sightings from these cameras. In contrast, the goal of this thesis is to develop protocols that can be employed on large networks covering extended areas.

More closely related to the approach described in this thesis is the work of Medeiros, Park and Kak [MPK08]. This paper describes a distributed approach to triangulating targets and distributing the tracking task over multiple nodes. This protocol involves electing a leader associated with every tracked object which is responsible for maintaining that track. Klausnet Tengg and Rinner [B.08] describe a distributed multilevel approach to fusing the measurements gleaned from a network of smart cameras. Their paper addresses the problem of automated aggregation of measurements through a hierarchy where different nodes have different capabilities and are given different responsibilities. The approach proposed in this thesis is different from these methods since the cameras are all viewed as peers. There is no need for a leader election process nor is there any hierarchy. This simplifies the deployment procedure and the resulting protocol and results in a scheme which is simpler to implement and more resilient to failure since the state of the tracker is automatically replicated and distributed throughout the network.

Our approach builds on the work of Mikic [MSJ00] and Focken [FS02] who describe schemes for tracking objects in 3D by triangulating the sightings obtained from multiple distributed vantage points. Like these works we formulate the tracking problem as one of associating measurements from different cameras and establishing correspondences over time. Our work extends these approaches by leveraging previous work on self localization and by describing how the scheme can be scaled to hundreds or thousands of camera nodes.

Arth, Leistner and Bischof [AC07] describe a scheme for object tracking where tracks are associated between cameras in the network by extracting distinctive features and matching these features over widely distributed viewpoints. In contrast the approach described in this thesis does not make use of distinctive appearance based features but instead assumes that the cameras are densely distributed in the scene so that targets can be handed off seamlessly.

Funiak Guestrin Paskin and Sukthankar [SFP06] describe an interesting algorithm inspired by work on Simultaneous Localization and Mapping wherein they tackle the tracking and camera localization problems in a single unified framework. Their system is capable of both localizing the targets and the cameras with a single convergent procedure given a sufficient number of corresponding tracks. Our system leverages our self localization scheme which allows the cameras to directly estimate their relative positions. This decomposition allows us to scale our approach more readily to larger networks since it allows us to avoid the problem of establishing correspondences between cameras in the absence of localization information and the complexities of uncertainty management in the SLAM approach.

When the images or results from multiple cameras can be processed at a central location, several sophisticated and effective algorithms have been proposed that provide state of the art results on the multicamera tracking problem. See for example recent systems proposed by Liem and Gavrila [ML09], Eshel and Moses [EM08], Mittal and Davis [MD03], Khan and Shah [KS06], Arsic [AHL$^+$08] and Fleuret [FBLF08]. In this work we consider what can be accomplished in the context where the tracking task must be carried out in real time by a distributed ensemble of embedded processors with limited communication bandwidth.

6.3 Target Tracking

Unlike other sensor modalities, the measurements obtained from camera systems are most useful when they are combined together as depicted in Figure 6.1. Here we see a typical situation where a target moving through the scene is localized in space by combining bearing measurements obtained from a variety of vantage points. Each bearing measurement is referred to as a sighting and for each sighting the camera can determine the range to the associated target by looking for confirming evidence from at least two other cameras. This notion of collaborative tracking is commonly employed in a number of vision based tracking systems [ML09, FBLF08, KS06, MD03, FS02, MSJ00] and is most easily done when the bearing measurements can be relayed to a central location for processing.

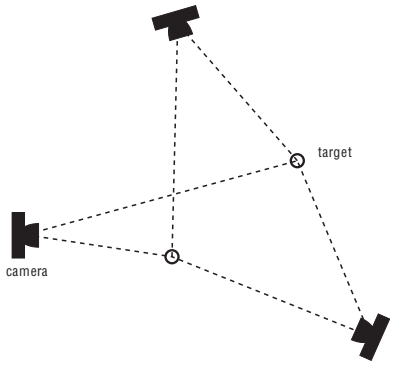


Figure 6.1: The bearing measurements obtained from two or more smart camera systems can be fused via triangulation to determine the position of the targets in three dimensions. When three or more smart camera systems are viewing the same area of space redundant measurements are available which can be used to help eliminate false tracks.

In this work we prefer a distributed approach where each camera localizes targets on its own by communicating with its neighbors. This is accomplished by having each node communicate its sightings to all of the other nodes in its immediate neighborhood. Once this has been done, each camera independently considers every pair of sighting measurements that it learns about, for each of the resulting candidate

(x, y, v_x, v_y)	State vector encoding the position and velocity of the object.
C	Covariance matrix associated with the state estimate
track id	Globally unique identifier associated with the track. The smart camera that creates this track forms this number by concatenating its own unique identifier with a local counter value to yield a globally unique identifier.
first timestamp	Time when the track was first created
last timestamp	Time when the track was last updated with a sighting

Table 6.1: Fields associated with each track data structure

points it looks for confirming sighting measurements in other views. The end result of this procedure is a set of candidate points in space which we term **targets**.

In addition to the set of targets derived from the sighting measurements, each camera also maintains a set of active **tracks** corresponding to trajectories of targets over time. Each of these track structures contains a number of fields which are described in Table 6.1.

Associated with each track is a state vector which encodes the position and velocity of the associated target along with the covariance matrices that are required to implement a Kalman filter tracker. Each track is tagged with a globally unique identifier which persists as the object moves through the entire scene. The track structure also contains timestamps indicating the first time that the tracked object was detected and the last time that the track was updated.

On each cycle every smart camera system must solve a data association problem resolving the relationship between the current tracks and the current targets. We can model the situation in terms of a bipartite graph as shown in Figure 6.2 . Here the nodes on the left depict the current tracks while the nodes on the right depict the current targets. For each track we determine the most probable target based on the Kalman filter estimate and covariance and link the nodes as shown. Note that at this stage there may be more than one track associated with each target. Each target then

chooses its best match among the list of possible tracks by considering the relative age of the tracks and choosing the oldest one. The measurements associated with this target are then used to update the Kalman filter associated with the winning track. Tracks that are not updated are propagated forward allowing for short periods of occlusion or tracker failure. Tracks that are starved of updates are eventually elided from the list. In this manner, short ephemeral tracks are removed from the system in favor of longer lasting records. In this scheme we assume that the clocks on the smart camera nodes are roughly synchronized so that timestamps can be compared without issue. This scheme is similar to the best-hypothesis tracking scheme described in [FS02].

Detected targets that are not explained by any of the existing tracks are used to create new tracks. When a smart camera creates such a track it concatenates its own unique smart camera identifier with a local counter value to form a globally unique identifier. This global identifier is then used in all subsequent communications and effectively travels with the object as it moves through the network.

Once the camera has resolved the relationship between tracks and targets and updated the list of tracks, it sends its current list of active tracks to its neighbors receiving in turn a list of all the targets that they are tracking. In this manner, information about tracks and target identities is propagated through the network allowing for seamless handoff as targets move throughout an extended scene. Note that ephemeral tracks may be introduced from time to time due to tracker failures or glitches but these are typically corrected eventually since the system is biased to prefer older labels for tracks wherever possible. We also expect occasional miss-associations in the triangulation phase. These cases typically produce outlier targets which do not win data association competitions and are thus starved for updates. As stated earlier, tracks which do not receive a sufficient number of updates per second are elided. (In the current implementation a track must receive at least 2 updates per second).

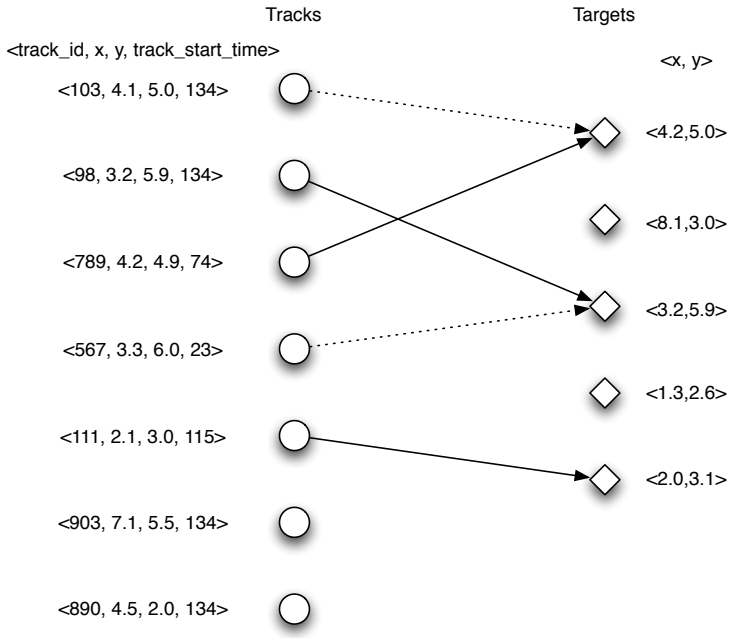


Figure 6.2: With every new image, each smart camera node must associate the detected targets with the tracks that it currently maintains. For each of the current tracks the system finds the best matching target - if any. The system then selects between multiple associations by favoring tracks with longer histories. In this figure the ultimate matches are indicated by solid lines while the dashed lines indicate possible matches that are rejected. This scheme causes the overall system to maintain the identities of the tracked objects as they move through the system. Unmatched targets become new tracks while unmatched tracks are eventually elided.

This protocol allows the network of distributed, loosely coupled smart camera systems to achieve consensus on target identities. By preserving target identifiers, the entire system is able to track targets over extended areas without requiring a central coordinating authority. Moreover, since the protocol only requires communication among near neighbors it can be scaled to networks of arbitrary size.

The entire procedure carried out by the smart camera is outlined in Algorithm 1 in pseudo-code.

Algorithm 1 Distributed Tracking Protocol

1: **loop**
2: Process current image and extract sightings
3: Gather sighting measurements from neighboring cameras
4: Triangulate sightings to obtain positions of current targets
5: Match current targets against the list of current tracks and update the tracks that are matched.
6: Targets which are not matched with any existing tracks are used to form new tracks.
7: Prune tracks that have not been updated recently.
8: Gather current tracks from neighboring cameras removing duplicates as needed
9: **end loop**

In summary, on every iteration of the tracking algorithm each camera sends to its neighbors a list of its current bearing measurements (step 3 of the algorithm). It also sends out a list of its current tracks (step 8). Each track structure contains the fields described earlier. Since these messages contain relatively little information and each camera only communicates with its near neighbors the method makes efficient use of the available communication bandwidth which is often quite limited.

6.4 Exfiltrating Information

An interesting feature of the proposed protocol is that the information about the targets in the scene is distributed among the smart cameras in the network. In fact information about the trajectory of a particular target is distributed among

the smart camera nodes that viewed the target at various portions of its trajectory. These trajectory are linked by a common target id which can be used to correlate the information across the network.

In order to harvest information from the network we propose a subscription model where a user can inject a request into the network indicating her interest in particular kinds of tracking events. This request would be broadcast periodically through the prevailing communication network to the individual camera nodes which would respond by sending back events that matched the criterion of the query.

For example a user may indicate that she is interested in all tracks passing through a particular region in the scene or all tracks that persist for more than a certain amount of time. Alternatively she may indicate interest in a particular set of target ids which could then be tracked over time with the information relayed back to the subscriber.

This scheme would decouple the service providers, the individual smart camera nodes, from the service subscribers. The subscribers would not request information from particular nodes which could fail or be removed at any time but would rather phrase their request in terms of queries which would be broadcast to all of the nodes that may be able to provide them with the desired information. This would mean that individual smart camera nodes would be able to change their mode of operation without disabling client applications. It also implies that the network could service multiple applications for multiple subscribers concurrently.

6.5 Experimental Results

6.5.1 Smart Camera Results

The custom smart camera in section 3.3 was used to test the proposed tracking scheme. In these experiments the camera was outfitted with a XVGA resolution imager (720x480) and a fisheye lens which affords a field of view of approximately

Figure 6.3: Snapshots of the first floor area showing some of the deployed cameras

180 degrees.

In our experiments a set of 8 cameras were deployed in an ad-hoc manner to cover the entire first floor of our office building, an area approximately 14 meters on side shown in Figure 6.3. The cameras were automatically localized as described in pervious chapters and then used to track targets moving through the area of regard in real time.

The first stage in the target tracking procedure is an adaptive background subtraction phase which determines which aspects of the image have changed significantly over time. A connected component phase is applied to the resulting binary image to find the most significant moving targets in the scene. The result of this analysis is a set of bearing vectors emanating from each of the cameras into the scene. Importantly, all of the real-time image processing occurs on the smart camera nodes themselves. Only the final sighting vectors associated with the targets are relayed over the Zigbee wireless network for further processing. This approach allows us to distribute the most computationally intensive aspects of the tracking procedure onto the smart camera nodes and avoid having to relay video imagery over the limited wireless bandwidth. Currently, each smart camera system extracts sighting measurements from the images at a rate of 15 frames per second.

Figures 6.5 and 6.6 shows the results of two tracking experiments. In the first

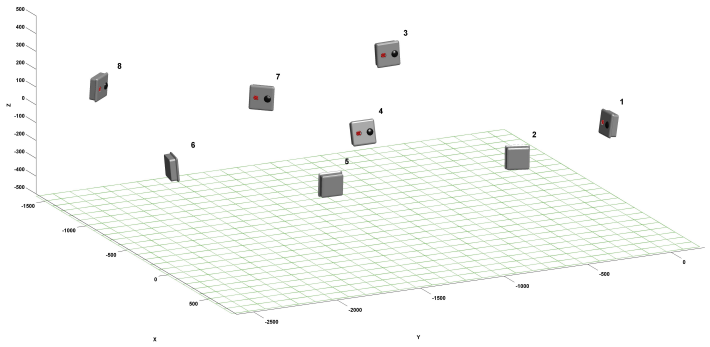


Figure 6.4: The smart cameras automatically determine their positions and orientations with respect to each other in 3D in a matter of seconds.

experiment the camera network was used to track a single person who walked into and around the first floor area and eventually exited through the same point he entered. The second experiment shows the system tracking two people who start off walking together and then split off onto two separate trajectories. In both of these experiments the targets were correctly tracked and associated throughout the sequence in real time through multiple camera handoffs since the structure of the scene ensured that no one smart camera had the targets in view throughout.

6.5.2 Simulation Results

In addition to the actual implementation on our smart camera network, a series of simulation experiments was carried out to investigate how the proposed scheme would perform on networks that were considerably larger than the ones we could construct with our available hardware. Figure 6.8 shows an example of an indoor environment reminiscent of an airport. This particular scene is monitored by a collection of 168 cameras mounted along the walls. The cameras could only see targets within a fixed distance of their positions. Using the proposed protocol, the

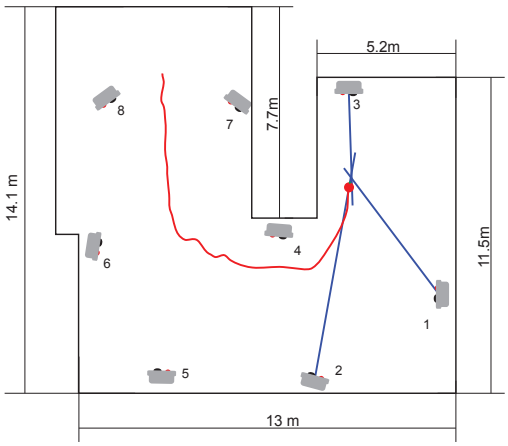


Figure 6.5: Snapshot of a real time tracking experiment showing the current position and past trajectory of a target moving through the scene. The lines emanating from the cameras represent sighting measurements which are triangulated to determine target location.

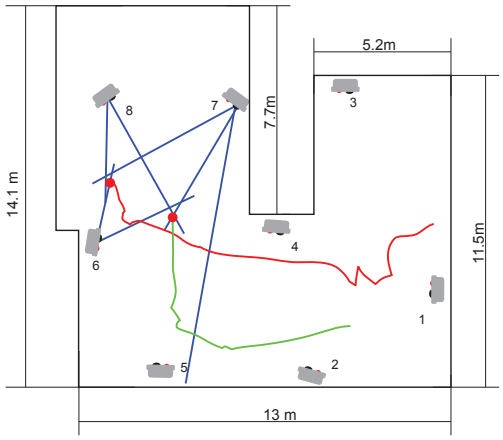


Figure 6.6: Snapshot of a real time tracking experiment showing the current position and past trajectory of two targets moving through the scene. Note the spurious sighting measurement, caused by a reflection in the window, which is not confirmed by the other cameras and, hence, discarded.

system was able to concurrently track a collection of 100 simulated targets over 100 timesteps. In order to characterize the communication requirements of the protocol the average number of messages received by each node was recorded on every timestep and the results are plotted on the graph in Figure 6.7. This plot includes both types of messages exchanged over the network, sighting measurements and track information. After an initial transient where the nodes communicate a number of measures to achieve consensus on target identifiers, the system settles into a steady state. The communication load here is evenly distributed throughout the network reflecting the distribution of the targets.

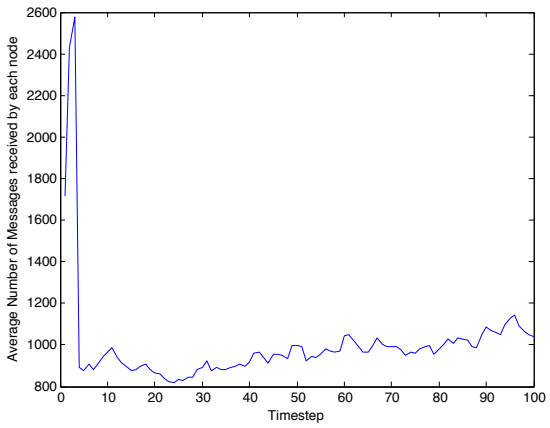


Figure 6.7: This graph indicates the average number of messages received by each of the nodes over the course of the simulation

Figure 6.8 shows the trajectories recovered for a few of the tracked targets. Note that the targets move throughout the environment between many cameras but the tracking system correctly maintains their identities.

In order to investigate how the protocol performed in the presence of failure the simulation experiment was repeated with the additional wrinkle that on each timestep each of the simulated smart cameras had a ten percent chance of failure. A camera that fails produces no measurements in the network. Even in this situation

the tracker was able to track 87 percent of the targets correctly through all 100 of the timesteps in the simulation. The remaining thirteen continue to be tracked but their identities are changed from the first to the last timestep indicating a tracker reacquisition. The resilience of the system to individual camera failure is a product of the fact that the tracker state is naturally distributed among a number of nodes so the failure of any single node is not catastrophic.

6.6 Discussion

This chapter describes an approach to using a network of distributed self-localizing smart cameras to localize and track moving obstacles in the scene. Our approach takes the view that smart cameras are currently small enough and cheap enough that one can consider deploying them fairly densely much as one installs lightbulbs. The challenge then is one of coordinating their activities so as to extract useful information from the ensemble subject to the prevailing computational and communication limitations.

In this approach each of the cameras independently analyzes its video imagery to find moving targets in its field of view, the results of this analysis are fused in the network to triangulate the location of the objects of interest in space. This approach devolves all of the low level image processing to the smart cameras and allows the nodes to use the limited wireless bandwidth more efficiently since they need only share sighting measurements and track data with their near neighbors. Using this approach we have been able to demonstrate real time tracking of targets over an extended area using a collection of embedded smart cameras, deployed in an *ad-hoc* manner and connected by a wireless communication network.

Currently the memory architecture and limited computational power of our smart camera nodes constrains what can be implemented in real time on our network. More powerful processors would allow for more sophisticated target detection schemes

such as face detection or pedestrian recognition which would cut down on false sightings. Nonetheless, the current implementation demonstrates that the proposed fusion scheme is resilient to spurious sightings from individual cameras because of the cross validation.

Importantly the proposed scheme is completely distributed, all of the nodes behave as peers and the tracking computations and tracking results are distributed throughout the network. Nonetheless, the protocol allows the networked cameras to achieve distributed consensus on the target identifiers which allows the system to seamlessly track targets as they move throughout the scene. The system is robust to isolated failures of individual nodes since multiple cameras typically cover any area and it recovers gracefully from momentary tracker failures.

The approach leverages the fact that the nodes can recover the relative position and orientation of their neighbors automatically. This makes it feasible to consider deploying large collections of smart camera nodes in an *ad-hoc* manner since one need not manually survey their relative locations. Furthermore, it allows the smart cameras to rapidly and reliably determine the nodes with which it must collaborate in the tracking application. This drastically reduces the cost and complexity of fielding multi-camera surveillance systems and allows them to be applied to a wider range of applications.

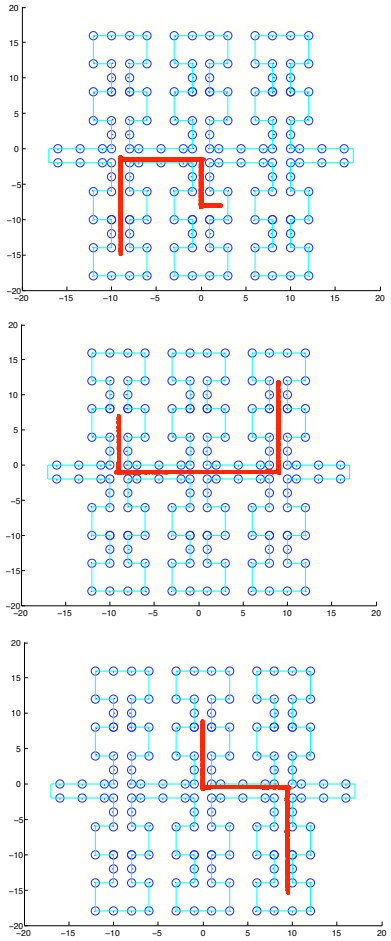


Figure 6.8: This simulation experiment modeled the layout of an airport. The system successfully tracked 100 targets using 168 smart camera nodes. The figure shows trajectories of individual targets successfully tracked throughout the environment. The small circles denote camera positions, the light lines indicate walls.

Chapter 7

Robot Self-assembly

As improvements in manufacturing continue to make sensors, actuators and processors smaller and cheaper it has become increasingly appealing to think of constructing robotic systems out of collections of modular components. The underlying theme motivating research in the field of modular robotics is the idea that complex electromechanical systems can be assembled from collections of modules in the same way that our bodies are constructed from collections of cells. Such an arrangement offers a number of potential advantages. Firstly, constructing robots from a few basic modular pieces can reduce the cost of the system since the basic units can be mass produced much like Lego blocks. Secondly modular systems can be more reliable and robust since the system could leverage the redundancy afforded by multiple active modules each having some actuation, sensing and computational capability. Finally, one of the most intriguing aspects of modular robots is the notion that the constituent modules of a robot could be reorganized or reconfigured depending on the dictates of the task at hand. Imagine, for instance, a robot that was originally configured as a humanoid that could reshape itself into a snake like form to worm through narrow passageways and then reconstitute its original form on the other side. In order to realize this vision of self reconfiguring or self assembling modular robots the individual modules in the system need to be able to gauge their position

and orientation with respect to each other. Here we describe an approach that uses our localization approach to provide this positioning information.

There are several compelling advantages to using our vision based methods for this localization task. Firstly, the required hardware, the imagers, computers and light sources are all amenable to miniaturization and are compatible with the manufacturing processes used to produce the robot modules themselves. Secondly, the proposed localization scheme requires relatively little power or communication. Thirdly the light beams used for localization are non-interfering which means that several nodes can self localize at the same time which is important in situations where we are interested in having tens or hundreds of modules operating simultaneously. Finally the imaging systems used for localization could also be used for other purposes such as sensing the environment to find obstacles or track targets. This allows us to consider applications where modular robotic systems are used to automatically deploy camera systems to advantageous locations to provide better situational awareness.

The idea of using a team of robots as mobile landmarks for localization was proposed by Kurazume et al as a more accurate and robust alternative to robot positioning via dead reckoning [KK$^+$94, Sas96, KH98] . The procedure was termed Cooperative Positioning (CP). With the proper sensor suites, the three-dimensional configuration of a team of robots could be determined by sharing relative position and orientation information. Three different types of CP methods were outlined dependent upon the sensors and number of robots available.

Type 1 CP required only a pair of robots capable of measuring relative range, azimuth and elevation angles. At any given time, only one of the two robots would move, leaving the second to act as a landmark of known position. This scheme was later revisited by Rekleitis et al [RDM02], and a planar version implemented using range and azimuth information obtained from a camera system.

Type 2 positioning required three robots capable of measuring relative azimuth and elevation angles with respect to a common axis. Based on these measurements

it is possible to recover the configuration of the team up to a scale factor. If two of the robots are held stationary the third can be allowed to move with its position continuously updated via triangulation.

Type 3 positioning required three robots where relative angle and distance measurements could be made. This method allowed up to two of the three robots to move at any given time. Kurazume et implemented and tested variants of this scheme using laser range finders. They showed far superior positioning results to those obtained from dead reckoning [Sas96, KH98]. Grabowski et al [GNsPK00] also implemented a planar version of a comparable scheme using the "Millibots" platform. In this work radio frequency and ultrasonic emissions were combined to determine relative agent position and orientation for collaborative localization. Similair ideas for modular robot localization were explored by Zhang et al in [ZAD+04]. In the commercial sector, IS Robotics (ISR) has also done similar work with their SWARM project. Using infra-red light, a team of robots is able to determine relative range and orientation information. The scheme described here can be viewed as a form of cooperative localization where the relevant inter-robot measurements are obtained using a distributed smart camera network.

7.1 Implementation

The modular reconfigurable robot used in this work is based on the CKBot (Connector Kinetic roBot) design which is described more fully in [PCTY08]. The kinematics and connector strategy used in this design is typical of many chain style reconfigurable modular robots. Each module in the system consists of:

1. A laser cut plastic (ABS) body with a hobby servo actuator to control one rotational degree of freedom.

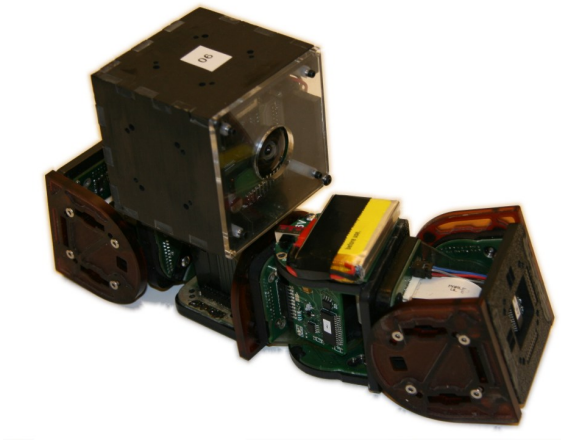


Figure 7.1: Each cluster in the modular robot is composed of four CKBot modules and a smart camera system.

2. A controller (PIC18F2680) and associated hardware for implementing a Controller Area Network (CAN) and neighbor-to-neighbor IR communications protocol.

3. Four connector faces that pass the communications bus and power bus with an option of attaching at 90 rotations.

The individual modules can be connected to each other either using screws or through a set of magnetic linkages. These magnetic linkages are implemented using an array of rare earth magnets embedded in the connector faces of the modules. Four north facing and four south facing magnets arranged such that two opposing faces will attract each other. The magnets have enough strength hold a chain of seven modules together against gravity. Using these novel magnetic linkages modules in an assembly can dynamically attach or disconnect from each other.

For the reassembly experiment described in Section 7.2 the modules were grouped into 3 clusters each of which was composed of four CKbot modules and a smart camera module. The modules within each cluster were screwed together in a chain as shown in Figure 7.1 and the clusters were connected to each other via magnetic

linkages. When the clusters are connected together they form a simple bipedal robot which can walk over flat terrain as depicted in Figure 7.3a; individually, the clusters can move about on the floor using snakelike gaits.

The smart camera modules used in this work are explained in section 3.2. Each smart camera module communicates with the other CKBot modules in the cluster via a CAN bus using the Robotics Bus protocol [GISG+04]. This CAN bus serves as the spinal cord of the cluster. The smart cameras are powered by rechargeable lithium ion batteries which are integrated into the design.

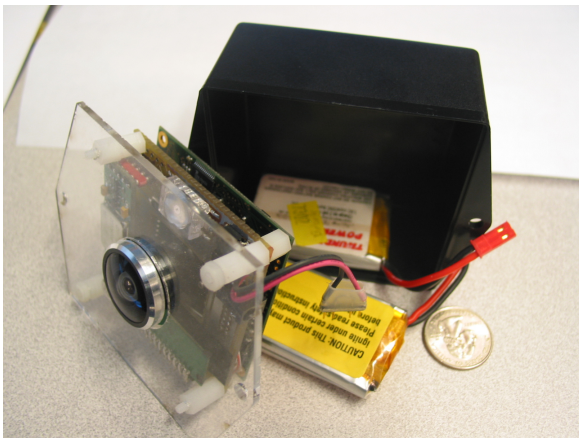


Figure 7.2: Each smart camera module is equipped with an imager outfitted with a fisheye lens, a digital signal processor, an accelerometer, a CAN bus transciever, and an LED signalling light

7.2 Reassembly experiment

The goal of the reassembly experiment was to demonstrate, for the first time, a modular robotic system that was capable of self-reassembly after an explosive disassembly event. The sequence of events is described in Figure 7.3; Figure 7.3a shows the modular robot which is initially configured as a bipedal walking system. This robot is violently disassembled with a swift kick and breaks into its three constituent

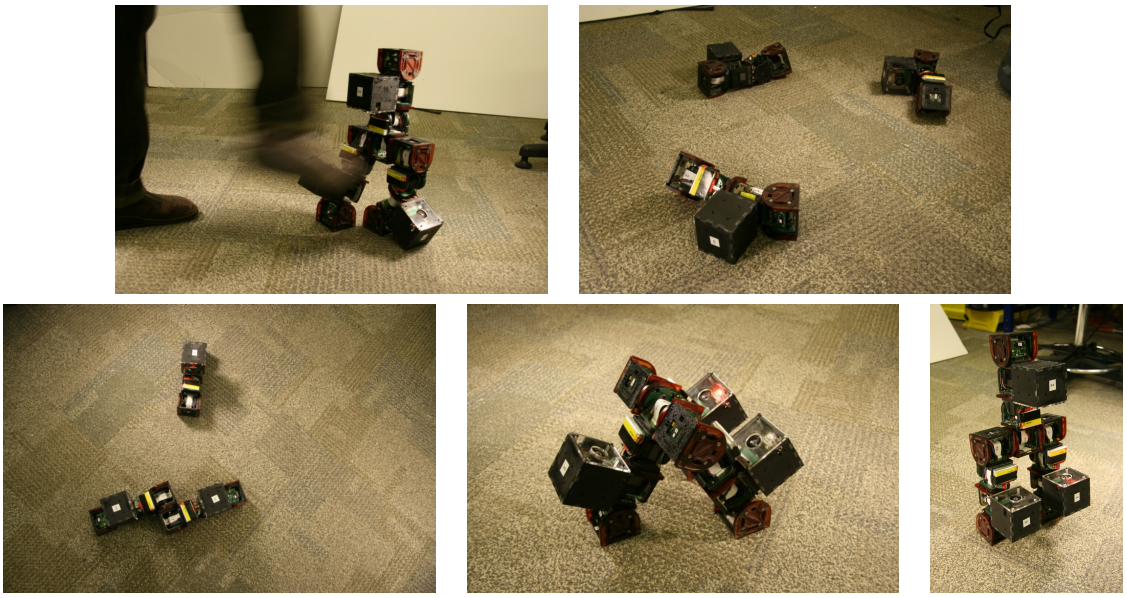


Figure 7.3: This figure shows the phases in the automated reassembly experiment.

clusters which end up randomly distributed on the floor. The modules can determine that they are no longer connected by monitoring the status of inter cluster communication links. Each cluster then automatically orients itself with respect to the gravity so that the camera modules are on top using the measurements from the 3-axis accelerometers.

Once upright, the clusters perform a search to find each other visually. In our implementation, the camera nodes signal their presence by blinking their lights in a preset pattern and use our localization schema to localize their neighbors.

The size of the blinking light in the image is a function of relative angle and distance between the two camera nodes. Although this size is not a good indicator of range when the camera nodes are far apart, our experience shows it is very effective measure at close range. This fits perfectly with docking where accurate measurements are only needed when the modules are close. Figure 7.4 shows an image acquired by one of the camera modules during the docking procedure while Figure 7.5 depicts how the size of the blinker in the image changes as a function of

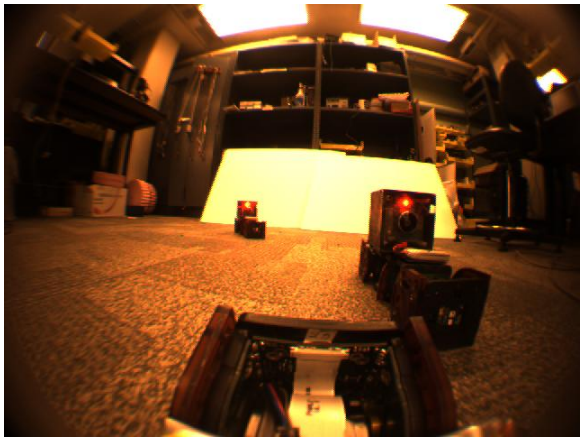


Figure 7.4: This figure shows an image taken by one of the smart camera systems mounted on one of the clusters.

distance.

In the first phase of the reassembly procedure the cluster corresponding to the torso and the cluster corresponding to the left leg rotate around in search of one another. When the left leg cluster locates the torso cluster it uses its estimate for the relative range and bearing to crawl towards that module and achieve the proper relative position for docking. Similarly, when the torso module sees the left leg it continually rotates around the vertical axis so that it maintains the correct relative orientation. In this manner the modules approach each other and orient themselves using a form of visual servoing. When the leg and torso clusters are sufficiently close the magnetic linkages reattach automatically and the communication and control links are reestablished.

Once the left leg and torso are docked the torso module changes its blinker pattern. This serves as a signal for the other leg cluster to begin its approach and docking sequence. Here, the blinker systems are used not just for localization but also for signaling and control. The right leg docking proceeds in a similar manner to the left leg with the torso module rotating to achieve the expected relative orientation to the incoming leg.

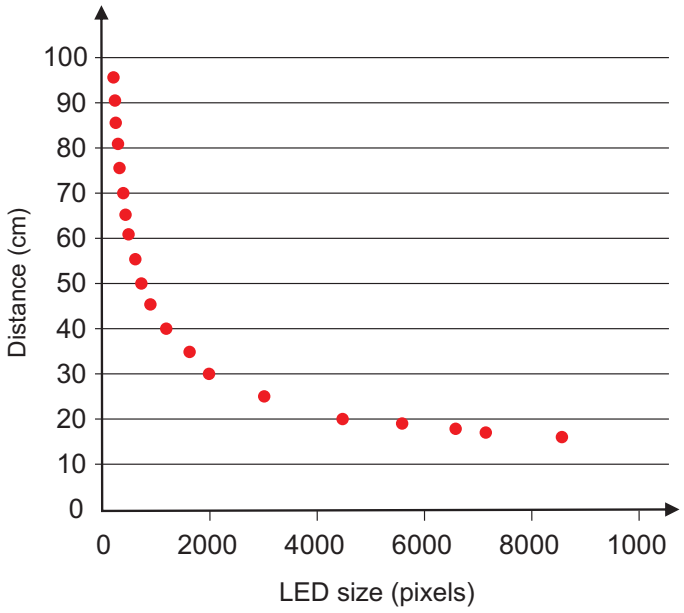


Figure 7.5: This graph shows how the area of the blinker in pixels varies as a function of distance

7.3 Future work

In this work the distributed smart cameras provide a relative localization capability that can be used in a number of ways. To date, this capability has been used to demonstrate self-reassembly of a modular system but the same technology could be used to guide more purposeful reconfiguration operations where a modular robot may deliberately break into pieces so that it can reassemble itself in another form for a different purpose. We can imagine distributing a collection of modular components onto a planetary surface and having them automatically assemble themselves into structures that perform different functions. The fact that smart camera technology is amenable to miniaturization makes it possible to consider endowing each of the modules with the means to sense and localize other members of the ensemble.

Further, we envision using the camera systems mounted on the modules to sense the environment and provide information that the system can use to plan its activities. Like Argus, the hundred eyed shepherd of Greek mythology, we envision robotic systems that would continually capture imagery from multiple distributed vantage points and fuse this information to localize each other, to find and track moving objects and to build 3D representations of the scene. This information would allow the ensemble to plan its motions, to manipulate objects and to respond intelligently to changes in its environment.

Chapter 8

3D reconstruction

8.1 Visual Hull Reconstruction

Multi camera systems are commonly used to derive information about the three dimensional structure of a scene. One approach to the reconstruction problem which is particularly well suited to the proposed self localizing smart camera network is the method of volume intersection which has been employed in various forms by a number of researchers [MBMG00]. This method can be used to detect and localize dynamic objects moving through the field of view of the smart camera network. Here a set of stationary cameras are used to observe one or more objects moving through the scene. Simple background subtraction is employed to delineate the portions of the images that correspond to the transient objects. Once this has been accomplished one can interrogate the occupancy of any point in the scene, P, by projecting it into each of the images in turn and determining whether or not it lies within the intersection of the swept regions. This process can be used to produce an approximation for the 3D structure of the transient objects by sampling points in the volume. The results of such an analysis are shown in Figure 8.1.

In this application the ability to rapidly localize a set of widely separated cameras is a distinct advantage. Other implementations of this reconstruction scheme involve

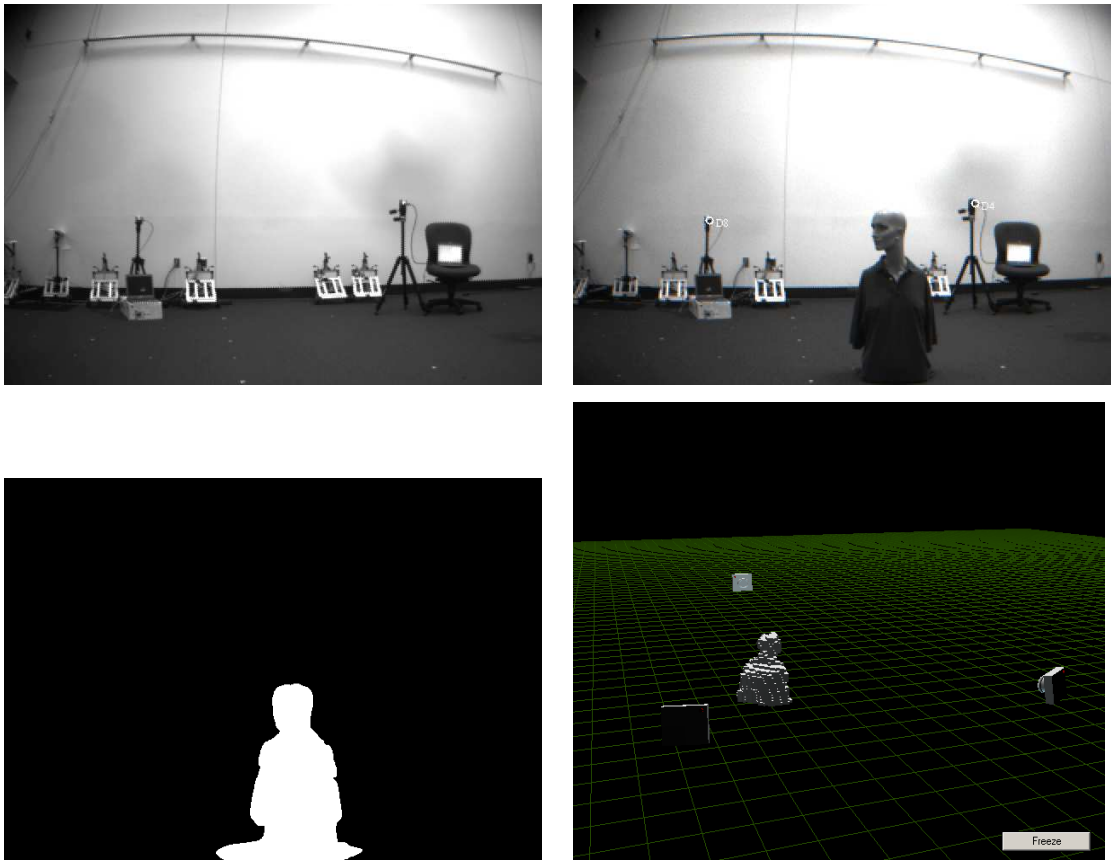


Figure 8.1: (a) Background image of a scene (b) Image with object inserted (c) Results of the background subtraction operation (d) Results of applying the volumetric reconstruction procedure to the difference images derived from the three smart camera nodes

complex, time consuming calibration operations. This implementation, in contrast, could be be quickly deployed in an ad-hoc manner and would allow a user to localize and track moving objects such as people, cars or animals as they move through the scene.

8.2 Ad Hoc Range Finder

Another approach to reconstructing the 3D geometry of the scene using the imagery from the smart camera network involves establishing stereoscopic correspondences between points viewed in two or more images. If we are able to find such corresponding points we can readily reconstruct their 3D locations through triangulation. In order to employ this scheme we need a mechanism for establishing correspondences between pixels in one image and their mates in another.

One approach to establishing these inter frame correspondences is to employ structured illumination to help disambiguate the matching problem. This idea has been employed successfully in a number of stereo reconstruction systems. One such structured illumination scheme is depicted in Figure 8.3 where a projection system sweeps a beam of light across the surface of the scene. Correspondences can then be established by simply observing when various pixels in the two images are lit by the passing beam.

Figure 8.2 shows a pair of images acquired using such a structured light correspondence scheme. Here a plane of laser light is swept across the scene and the curves corresponding to the illuminated pixels in the two images are recovered. In each image, every point on the curve corresponds to a ray in space emanating from that camera position. To find the correspondence for that point in the other image we first project that ray into the other image to construct the corresponding epipolar line and then search along that line to find the corresponding pixel that is also illuminated by the laser plane as shown in Figure 8.3.

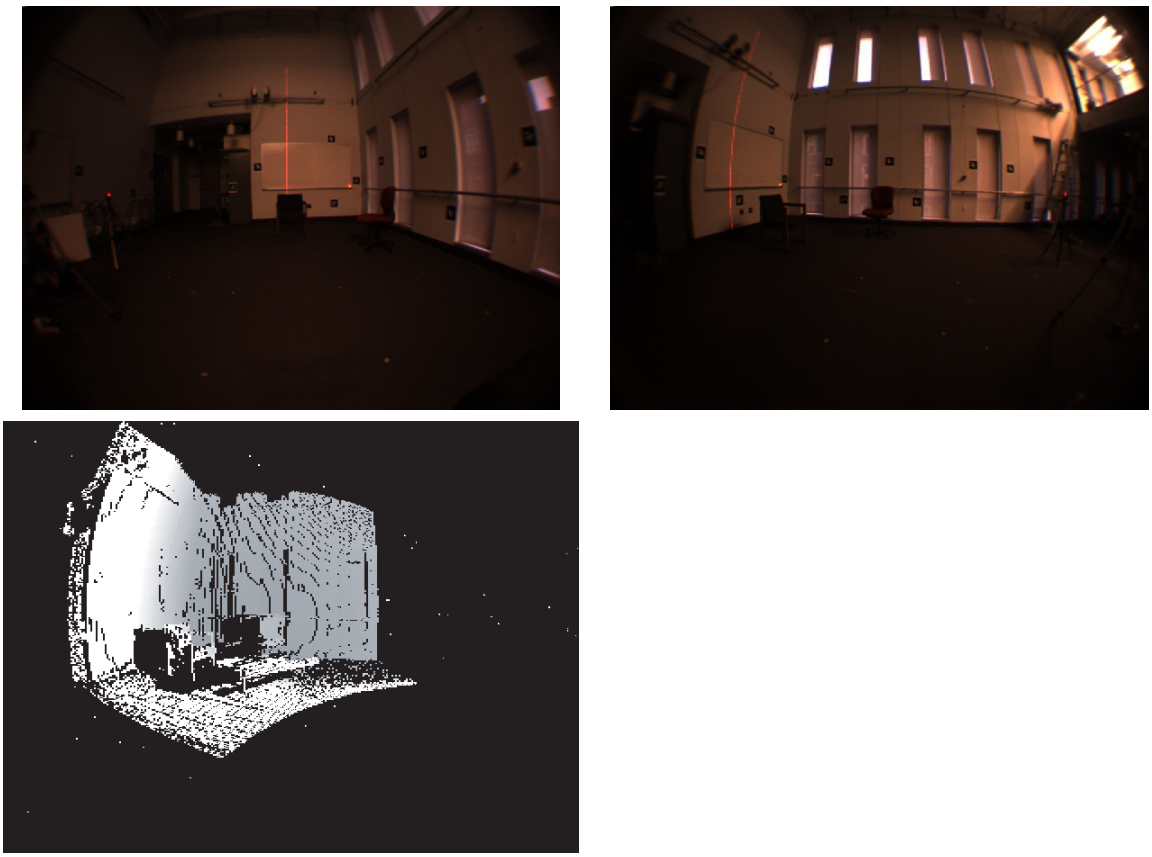


Figure 8.2: ((a) and (b) show Two images of a scene illuminated with a plane of laser light which is used to establish correspondences between the two views (c) shows the range map constructed based on the correspondences derived from a sequence of such images.

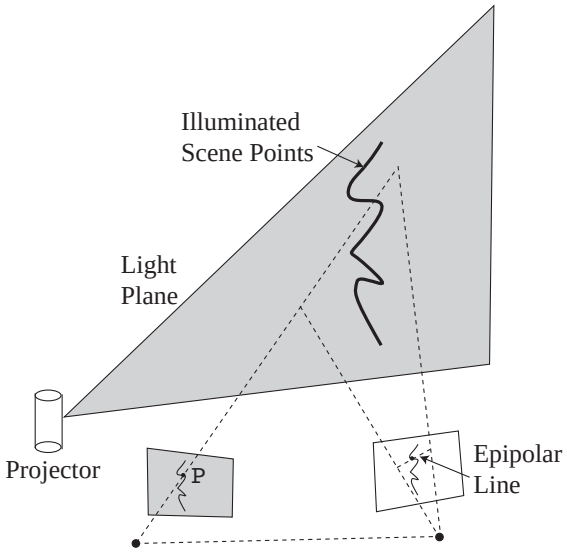


Figure 8.3: At every point in time the projector illuminates a set of scene points along a planar curve in the scene. For every point on the projected curve in one image we can locate its correspondent in the other image by searching along the epipolar line in the other image.

After sweeping the plane over the entire scene we are able to determine the range to most of the points in the scene that are visible from both camera positions even though those two camera positions are widely separated. Such a range map is shown in Figure 8.2c. This range scan was constructed by sweeping the laser plane through 180 degrees in 1 degree increments.

It is important to note here that this range map is constructed in an ad-hoc manner since the relative positions and orientations of the cameras are reconstructed automatically using the self localization algorithm and the position and orientation of the projector are not needed to recover the scene depths. The proposed reconstruction scheme is interesting because it provides a mechanism for recovering the structure of an extended scene using an ensemble of small, cheap image sensors and beam projectors which can be deployed in an ad-hoc manner. This is in contrast to the traditional approach of recovering scene structure using expensive range sensors

which must be carefully calibrated and aligned.

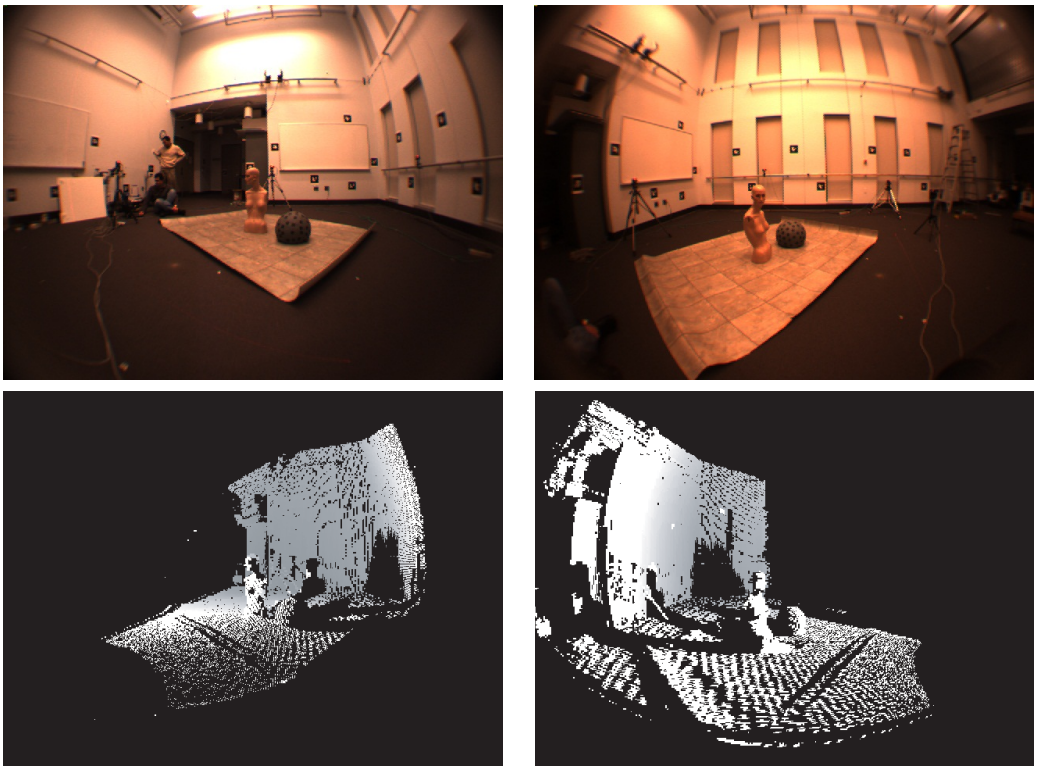


Figure 8.4: In this experiment range maps of the scene were constructed from 4 different vantage points using different configurations of cameras and projectors. Two of these scans are shown here along with the corresponding images

The scheme can be extended for use with multiple cameras and multiple beam projectors as shown in Figure 8.4. Here we are able to obtain multiple range maps of the scene taken from different vantage points using a collection of camera systems and projector positions. Importantly, since we are able to recover the relative positions of all of the cameras used here via the self localization scheme, all of the recovered range maps can be related to a single frame of reference. This provides an avenue to recovering the structure of extended environments by merging the range maps obtained from the different camera systems into a single coherent model of the scene.

Chapter 9

Conclusion

In his novel "A Deepness In the Sky" (1999) Vernor Vinge describes a spacefaring society which makes use of "localizers" small, embedded, networked sensors that could be distributed throughout an extended area such as a spaceship or a planet surface to provide a seamless web of communication, monitoring and surveillance services. This concept echoes in literature the notion of "SmartDust" advanced by Pister and his colleagues in the Sensor Network community. The goal of this thesis has been to realize, in part, this vision of small intelligent sensors which can be deployed in an ad-hoc fashion to provide intelligence about a given area.

This thesis argues for using a combination of hardware and software to address the crucial problem of localization. More specifically, by equipping the smart camera nodes with controllable light sources and accelerometers we are able to simplify the problem of reliably discovering correspondences in the scene and stabilize the problem of estimating the relative position and orientation of the nodes in the scene.

The goal in this work has been to develop an approach to localization which takes into account the constraints on computation and communication that one faces when developing for networked embedded platforms. The method that has been proposed has several virtues: the detection and localization computations are simple enough that they can be implemented efficiently on embedded processors, furthermore the

entire localization procedure can be performed quite rapidly allowing for real-time, ad-hoc deployment of sensor nodes. Further, the accuracy of the results compares favorably with the accuracy obtained using other methods that have been proposed for this problem. Using the sensor nodes as fiducials eliminates the tacit assumption that the environment will supply an adequate distribution of feature correspondences or tracked objects.

The method was designed to be scalable to support deployments that may ultimately involve thousands of sensor systems. The localization computation exploits the sparseness of the system of measurements which means that the method can be used to localize hundreds of cameras in a matter of seconds using modest computational resources. The computation can also be distributed in a natural manner where each node runs the procedure to localize its neighbors. Once this has been done these estimates can be used to translate locations from one frame of reference to another.

The proposed localization scheme has been compared with a state of the art feature-based camera localization scheme, the Sparse Bundle Adjustment package developed by developed by Lourakis and Argyos [LA09]. The methods were compared along a number of axes including the accuracy of the camera location estimates produced, the number of image measurements used, the computational complexity and the resilience to measurement and initialization error.

The simulation results show that as the number of cameras is increased the proposed self localization scheme produces more accurate results from fewer image measurements with less computational effort. More specifically, the translational error in the camera position estimates was on the order of 5 to 10 times more accurate and the procedure required on the order of 5 times less computational effort and less than one tenth the image measurements. The efficacy of the method can be attributed to the fact that while the self localization method uses fewer measurements, these measurements directly capture the epipolar structure of the camera configuration

and are, therefore, more useful than the measurements used by the feature-based methods which only provide indirect information about the camera layout.

It is important to keep in mind, however, that the method was designed for a particular deployment context and is not appropriate for all situations. For example, since the method makes use of hardware modifications, it could not be directly applied to existing surveillance networks where the camera hardware is already fixed. Further, an underlying assumption in this work is that the camera nodes can be distributed fairly densely in the scene so that one can leverage situations where two or more cameras are mutually visible. As such the method would not be as useful in situations where the fields of view of the cameras do not overlap significantly.

In this thesis localization is viewed as a basic capability which enables a host of applications. In order to demonstrate the utility of the proposed approach the thesis describes a few applications which illustrate how self localizing smart camera networks could be employed.

Tracking is one of the most natural and popular applications of smart camera networks. This thesis describes how a network of self localizing cameras can be used to provide real time tracking of objects moving through an extended area. Importantly the proposed approach allows for ad-hoc deployments where the system can be deployed and made operational in a matter of minutes. This allows us to consider scenarios where a collection of cameras can be rapidly deployed to create a perimeter or a virtual fence. Another important characteristic of the proposed approach to tracking is that the tracking computation is distributed throughout the network. All of the nodes act as peers and there is no need for leader election. This makes the resulting system robust to intermittent failures and allows us to add or delete nodes from the network while the system is running.

The thesis also describes one application of self localizing cameras in the field of robotics. Here the cameras are used to provide localization services for a collection of modules which can dynamically reconfigure themselves. In this context we argue

that small self localizing cameras can be adapted for use in a wide range of robotic applications since the underlying technology, imagers, wireless networking and computation are all amenable to miniaturization. This allows us to consider deploying collections of cameras around a robots workspace to localize the components and to provide information about the structure of the scene which could be used for planning and obstacle avoidance. The fact that the camera systems can self localize makes it easier to consider deploying such multi-eyed systems in practice.

Another application that is briefly described in Chapter 8 is 3D reconstruction. Once the relative position and orientation of the cameras has been recovered, one can view the ensemble as a wide baseline stereo system. The aim here is to establish additional correspondences between the views which can be used to reconstruct the 3D structure of the scene. The ability to recover the 3D structure of the scene opens up a number of possibilities. One could imagine using such a network of cameras to study how people interact with their surroundings.

The applications that have been explored in this thesis were intended to be representative rather than exhaustive. Clearly there is further scope for experimentation and innovation in this area. The underlying theme has been that cameras are rapidly becoming small enough and cheap enough that they could be unobtrusively embedded throughout an area. This thesis addresses the question of how we could go about coordinating and exploiting the information gathered from multiple vantage points. Based on this work one could imagine smart environments where a collection of smart cameras embedded in the walls and ceilings could be used to help localize other sensors such as temperature, pressure and air-flow sensors which could be used to provide detailed information on how a buildings heating and air conditioning system are functioning in order to reduce energy consumption. Such a network could also be used to localize hosts of small robots which could be deployed into the building to clean floors, to take additional measurements or to or provide security services.

Combining the angular measurements derived from the camera sensors with range measurements derived from ultra wide band transceivers or other sources could potentially provide a comprehensive solution to the problem of sensor localization since the range signals could be used to resolve the scale ambiguity inherent in angle based localization and could be used to localize nodes that are not visible to the camera systems. Conversely the imagery acquired by the cameras provides orientation information and contextual information that the range measurements do not. For example if a temperature sensor that has been localized by the camera system reported an anomalously high value the system could use the imagery acquired by the cameras to determine whether or not the area was on fire or not.

It would also be interesting to investigate how current work in networked embedded camera systems could be fused with emerging trends in cloud computing. Could such networked sensors be married with networked computational and storage resources which could be scaled elastically to match the demands for analytical and archival services. This may be a viable approach to fusing the local information gathered from scores of sensors with global analytical capabilities which could correlate and summarize the measurements to provide an overall picture of the environment which could be used to enhance our wealth, safety and welfare.

Bibliography

[AC07] Bischof H. Arth C., Leistner C. Object reacquisition and tracking in large-scale smart camera networks. In *Proc. First ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '07)*, pages 156–163, Vienna, Austria, Sept 2007.

[AHL+08] D. Arsic, E. Hristov, N. Lehment, B. Hornler, B. Schuller, and G. Rigoll. Applying multi layer homography for multi camera person tracking. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–9, Sept. 2008.

[ARD04] B. Dungan Ali Rahimi and Trevor Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 187–194, Washington D.C., June 2004.

[ASS+09] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *International Conference on Computer Vision*, pages 72–79, Kyoto, September 2009.

[AT02] Matthew Antone and Seth Teller. Scalable extrinsic calibration of omni-directional image networks. *Int. J. Comput. Vision*, 49(2-3):143–174, 2002.

[B.08] Klausner A. Tengg A. Rinner B. Distributed multilevel data fusion for networked embedded systems. In *Selected Topics in Signal Processing, IEEE Journal of Publication Date: Aug. 2008*, Aug 2008.

[BALaDO+06] Alexandra Branzan-Albu, Denis Laurendeau, Sylvain Comtois Patrick Hebert, Andre Zaccarin, Marc Parizeau Maldague, Richard Drouin, Stephane Jean, Helene Torresan Monnet: Monitoring pedestrians with a network of loosely-coupled cameras. In *International Conference on Pattern Recognition (ICPR 2006)*, Hong Kong, China, August 20-24 2006.

[BHET04] Nirupama Bulusu, John Heidemann, Deborah Estrin, and Tommy Tran. Self-configuring localization systems: Design and experimental evaluation. *Trans. on Embedded Computing Sys.*, 3(1):24–60, 2004.

[BSLS06] A. Barton-Sweeney, D. Lymberopoulos, and A. Sawides. Sensor localization and camera calibration in distributed camera sensor networks. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–10, San Jose, Oct. 2006.

[CDR07] Zhaolin Cheng, Dhanya Devarajan, and Richard J. Radke. Determining vision graphs for distributed camera networks using feature digests. *EURASIP J. Appl. Signal Process.*, 2007(1):220–220, 2007.

[CHT04] Anthony Cowley, Hwa-Chow Hsu, and Camillo J Taylor. Distributed sensor databases for multi-robot teams. *IEEE*, pages 691 – 696 Vol.1, 2004.

[CPSK04] Rodrigo L. Carceroni, Flavio Padua, Geraldo Santos, and Kiriakos Kutulakos. Linear sequence-to-sequence alignment. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 746–753, Washington D.C., June 2004.

[CSI06] Yaron Caspi, Denis Simakov, and Michal Irani. Feature-based sequence-to-sequence matching. *Int. J. Comput. Vision*, 68(1):53–64, 2006.

[DCR08] D. Devarajan, Zhaolin Cheng, and R.J. Radke. Calibrating distributed camera networks. *Proceedings of the IEEE*, 96(10):1625–1639, Oct. 2008.

[DRC06] Dhanya Devarajan, Richard J. Radke, and Haeyong Chung. Distributed metric calibration of ad hoc camera networks. *ACM Transactions on Sensor Networks*, 2(3):380–403, 2006.

[ECPS02] Deborah Estrin, David Culler, Kris Pister, and Gaurav Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, 2002.

[EM08] R. Eshel and Y. Moses. Homography based multiple camera detection and tracking of people in a dense crowd. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[FBBS06] S. Fleck, F. Busch, P. Biber, and W. Strasser. 3d surveillance a distributed network of smart cameras for real-time tracking and its visualization in 3d. In *Embdded Computer Vision 06*, page 118, 2006.

[FBLF08] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2):267–282, 2008.

[FCSS09] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *International Conference on Computer Vision*, pages 80–87, Kyoto, October 2009.

[FGPS06] Stanislav Funiak, Carlos Guestrin, Mark Paskin, and Rahul Sukthankar. Distributed localization of networked cameras. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 34–42, New York, NY, USA, July 2006. ACM.

[FKFB05] Wu-Chi Feng, Ed Kaiser, Wu Chang Feng, and Mikael Le Baillif. Panoptes: scalable low-power video sensor networking technologies. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(2):151–167, 2005.

[FS02] Dirk Focken and R. Stiefelhagen. Towards vision-based 3-d people tracking in a smart room. In *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces (ICMI 02)*, 2002.

[GD06] Nicolas Gehrig and Pier Luigi Dragotti. Distributed sampling and compression of scenes with finite rate of innovation in camera sensor networks. In *in Proceedings of Data Communication Conference (DCC), Snowbird,Utah,USA*, pages 83–92, 2006.

[GISG+04] Daniel Gomez-Ibanez, Ethan Stump, Ben Grocholsky, Vijay Kumar, and Camillo Taylor. The robotics bus: A local communications bus for robots. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, 2004.

[GKK+03] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. Irisnet: An architecture for a world-wide sensor web. Technical Report IRP-TR-04-12, Intel Research, October 2003.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins, 1996.

[GNsPK00] Robert Grabowski, Luis E. Navarro-serment, Christiaan J.J. Paredis, and Pradeep K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8:293–308, 2000.

[HA06] Stephan Hengstler and Hamid Aghajan. A smart camera mote architecture for distributed intelligent surveillance. In *ACM Sensys Workshop on Distributed Smart Cameras (DSC 06)*, pages 23–31, Boulder, October 2006.

[HS97] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106 –1112, June 1997.

[HSW+00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104. ACM Press, 2000.

[HZ03] Richard E. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[KAC08] G. Kayumbi, N. Anjum, and A. Cavallaro. Global trajectory reconstruction from distributed visual sensors. In *International Conference on Distributed Smart Cameras 08*, pages 1–8, 2008.

[KCA+07] T. Ko, Z. M. Charbiwala, S. Ahmadian, M. Rahimi, M. B. Srivastava, S. Soatto, and D. Estrin. Exploring tradeoffs in accuracy, energy and latency of scale invariant feature transform in wireless camera networks. In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, pages 313–320, 2007.

[KGSL05] Purushottam Kulkarni, Deepak Ganesan, Prashant Shenoy, and Qifeng Lu. Senseye: a multi-tier camera sensor network. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 229–238, New York, NY, USA, 2005. ACM.

[KH98] Ryo Kurazume and Shigeo Hirose. Study on cooperative positioning system: Optimum moving strategies for cps-iii. In *ICRA*, pages 2896–2903, 1998.

[KK+94] Ryo Kurazume, , Ryo Kurazume, Shigemi Nagata, and Shigeo Hirose. Cooperative positioning with multiple robots. In *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, pages 1250–1257, 1994.

[KS06] Saad M. Khan and Mubarak Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *In European Conference on Computer Vision*, 2006.

[LA09] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[LLWW04] C. H. Lin, T. Lv, and I. B. Ozer W. Wolf. A peer-to-peer architecture for distributed real-time gesture recognition. In *International*

Conference on Multimedia and Exposition, volume 1, pages 57–60, Baltimore, June 2004.

[MBMG00] Wojciech Matusik, Christopher Buehler, Ramesh Raskarand Leonard McMillan, and Steven J. Gortler. Image-based visual hulls. In *SIGGRAPH*, 2000.

[MD03] Anurag Mittal and Larry S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *Int. J. Comput. Vision*, 51(3):189–203, 2003.

[ML09] Dariu M. Gavrila Martijn Liem. Multi-person tracking with overlapping cameras in complex, dynamic environments. In *British Machine Vision Conference*, 2009.

[MLRT04] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of the Second International Conference on Embedded and Networked Sensor Systems, SenSys 04*, pages 39–50, November 2004.

[MPK08] H. Medeiros, J. Park, and A.C. Kak. Distributed object tracking using a cluster-based kalman filter in wireless camera networks. *IEEE Journal of Selected Topics in Signal Processing*, 2(4):448–463, August 2008.

[MSJ00] Ivana Mikic, Simone Santini, and Ramesh Jain. Tracking objects in 3d using multiple camera views. In *Asian Conference on Computer Vision (ACCV)*, January 2000.

[NDG02] S. Nath, A. Deshpande, and P.G. Gibbons. Mining a world of smart sensors. Technical Report IRP-TR-02-05, Intel Research, August 2002.

[NDK+02] S.K. Nath, A. Deshpande, Y Ke, P.G. Gibbons, B. Karp, and S. Seshan. Irisnet: An architecture for compute-intensive wide-area sensor network services. Technical Report IRP-TR-02-10, Intel Research, December 2002.

[NL03] P. Newman and J. Leonard. Pure range-only subsea slam. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1921–1926, Taiwan, September 2003.

[PCTY08] Michael Park, Sachin Chitta, Alex Teichman, and Mark Yim. Automatic configuration recognition methods in modular robots. *International Journal of Robotics Research*, 27, 03/2008 2008.

[PSF+08] Giulia Piovan, Iman Shames, Baris Fidan, Francesco Bullo, and Brian D. O. Anderson. On frame and orientation localization for relative sensing networks. In *IEEE Conference on Decision and Control*, pages 2326–2331, Cancun, July 2008. IEEE.

[QMK+08] M. Quinn, R. Mudumbai, T. Kuo, Z. Ni, C. De Leo, and B. S. Manjunath. Visnet: A distributed vision testbed. In *ACM/IEEE International Conference on Distributed Smart Cameras, 2008*, pages 364–371, Sep 2008.

[RBI+05] Mohammad Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. Cyclops: in situ image sensing and interpretation in wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 192–204, New York, NY, USA, 2005. ACM.

[RDM02] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. Multi-robot cooperative localization: A study of trade-offs between efficiency and accuracy. In *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2690–2695, 2002.

[Sas96] R Kurazume S Hirose S Nagata N Sashida. Study on cooperative positioning system (basic principle and measurement experiment). In *IEEE International Conference on Robotics and Automation*, pages pp. 1421–1426, 1996.

[SBM+04] G. Simon, G. Balogh, M. Maroti, B. Kusy, J. Sallai, A. Ledeczi, A. Nadas, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the Second International Conference on Embedded and Networked Sensor Systems, SenSys 04*, pages 1–13, Baltimore, November 2004.

[SFP06] Guestrin Rahul Sukthankar Stanislav Funiak, Carlos Ernesto and Mark Paskin. Distributed localization of networked cameras. In *Fifth International Conference on Information Processing in Sensor Networks (IPSN'06)*, pages 34 – 42, April 2006.

[SP06] Sudipta N. Sinha and Marc Pollefeys. Pan-tilt-zoom camera calibration and high-resolution mosaic generation. *Comput. Vis. Image Underst.*, 103(3):170–183, 2006.

[SPM04] Sudipta Sinha, Marc Pollefeys, and Leonard McMillan. Camera network calibration from dynamic silhouettes. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 195–201, 2004.

[SSRCF08] B. Song, C. Soto, A.K. Roy Chowdhury, and J.A. Farrell. Decentralized camera network control using game theory. In *International Conference on Distributed Smart Cameras*, pages 1–8, 2008.

[SSS06] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006.

[STY+07] Babak Shirmohammadi, Camillo Jose Taylor, Mark Yim, Jimmy Sastra, Michael Park, and Mike Dugan. Using smart cameras to localize self-assembling modular robots. In *First ACM/IEEE International Conference on Distributed Smart Camera*, pages 76–80, Vienna, September 2007.

[Tay04] Camillo Jose Taylor. A scheme for calibrating smart camera networks using active lights. In *ACM SENSYS 04 - Demonstration Session*, Baltimore, November 2004.

[TC05] Camillo J. Taylor and Ryan Cekander. Self localizing smart camera networks. In *IEEE Conference on Computer Vision and Pattern Recognition (Demonstration Session)*, San Diego, June 2005.

[TG04] Tinne Tuytelaars and Luc Van Gool. Synchronizing video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 762–768, 2004.

[TS06] Camillo J. Taylor and Babak Shirmohammadi. Self localizing smart camera networks and their applications to 3d modeling. In *ACM SenSys/First Workshop on Distributed Smart Cameras (DSC 06)*, Boulder, October 2006.

[TV09] R. Tron and Rene Vidal. Distributed image-based 3-d localization of camera sensor networks. In *IEEE Conference on Decision and Control*, pages 901–908, Shanghai, December 2009.

[WZ02] L. Wolf and A. Zomet. Sequence-to-sequence self calibration. In *European Conference on Computer Vision*, pages 370–382, Copenhagen, May 2002.

[YZC03] Z. Yue, L. Zhao, and R. Chellappa. View synthesis of articulating humans using visual hull. In *Proceedings of the IEEE International Conference on Multimedia and Exposition*, volume 1, pages 489–492, Baltimore, July 2003.

[ZAD+04] Ying Zhang, Lee Ackerson, David Duff, Craig Eldershaw, and Mark Yim. Stam: A system of tracking and mapping in real environments. *IEEE Journal on Wireless Communication*, 2004.