FUSION OF STEREO DEPTH MAPS AND SENSOR LOCALIZATION IN WIRELESS SENSOR NETWORKS

BY

DAVID J. SCHERBA

B.S., University of Illinois at Urbana-Champaign, 2003

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

Abstract

This thesis addresses the following problem: Given multiple images of a scene, and sensor localization data, can we improve our knowledge of scene geometry and sensor locations? A novel approach to improving three dimensional scene geometry and sensor locations by fusing localization data from wireless sensor networks (WSN) with depth maps obtained through stereopsis is presented along with a software prototype. In experiments, "smart" wireless sensors, and a digital camera are used. Sensor locations are determined via an acoustic time-of-flight ranging technique, and the uncalibrated depth map is computed using a binocular stereopsis technique. Depth map calibration is performed (a) by fitting a three dimensional surface to a set of a priori known co-planar sensor locations, and (b) by computing the depth map calibration model parameters through minimizing the squared distance between the sensor-defined plane and the corresponding depth map measurements. Fusion is performed by analyzing the expected uncertainties of the outputs from computational stereopsis and wireless sensor network localization techniques, and then by minimizing the uncertainty over a wide range of depth values. Algorithms for computational stereopsis, sensor localization, and depth map and sensor location fusion are presented, followed by multiple experiments and obtained simulation and experimental fusion results. The contribution of the presented work is in building a first prototype for improving our knowledge of scene geometry and sensor locations based on camera and WSN data fusion using TinyOS and Image to Knowledge (I2K) software tools. A summary of challenges with respect to automation, computational requirements, and obtained accuracy of depth estimation is included.

To my parents, who showed me the world,
to Rebecca, who makes it worthwhile,
and to Homer, who tries his best to make me laugh.

Acknowledgments

This thesis would not have been possible without the help of a number of people. First and foremost, my adviser Peter Bajcsy, Ph.D. of the National Center for Supercomputing Applications (NCSA) provided assistance, motivation, ideas, and funding throughout the duration of my research. Himanshu Khurana, Ph.D. also of NCSA provided additional funding during the later stages of my research. Rob Kooper, of NCSA, helped with some Image to Knowledge issues, asked some very good TinyOS-related questions, and always had an available lab key. Marquita Miller of NCSA took care of all the hard paperwork, helped replace faulty sensors, and kept the fridge full. The other graduate students in the Automated Learning Group provided a productive office environment. Last, but not least, my wife Rebecca Swartz helped me to stay focused when I most needed it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALG [NCSA] Automated Learning Group

AWGN Additive White Gaussian Noise

GPS Global Positioning System

I2K Image to Knowledge

MEMS Micro-Electo Mechanical Systems.

NCSA The National Center for Supercomputing Applications

ToF Time-of-Flight

WSN Wireless Sensor Network.

XML Extensible Markup Language

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) have captured the imagination of a large number of people. Visionaries imagine a future where "smart dust" can be deployed to gather interesting information quickly and cheaply. Today, the technology for mass-produced, standardized, inexpensive sensors is still young. Most research on wireless sensor networks is focused on the network infrastructure; problems like communication protocols, security, and localization.

This thesis presents work on a possible application of this infrastructure: increasing the accuracy of determining a scene's geometry. In addition to improving the quality of this geometric data, this work also shows one way of combining the output of wireless sensor networks with traditional sensors, namely cameras. This fusion process, creating a unified model of the scene from multiple kinds of sensors, can potentially contribute to a higher-level view of sensing and hopefully drive future applications.

1.1 Problem Statement

This thesis addresses the following problem: Given multiple images of a scene, and sensor localization data, can we improve our knowledge of scene geometry and sensor locations? The answer is yes, as this thesis shows. We discuss the relevant components of information acquisition: computational stereopsis and WSN localization. The error arising in each 3-D information recovery approach is derived as a function of scene depth. We utilize this error to come up with a data fusion rule. Finally, we quantify the improvement of our scene knowledge in a laboratory environment after applying the fusion rule. A flow chart of the fusion process can be seen in Figure 1.1.

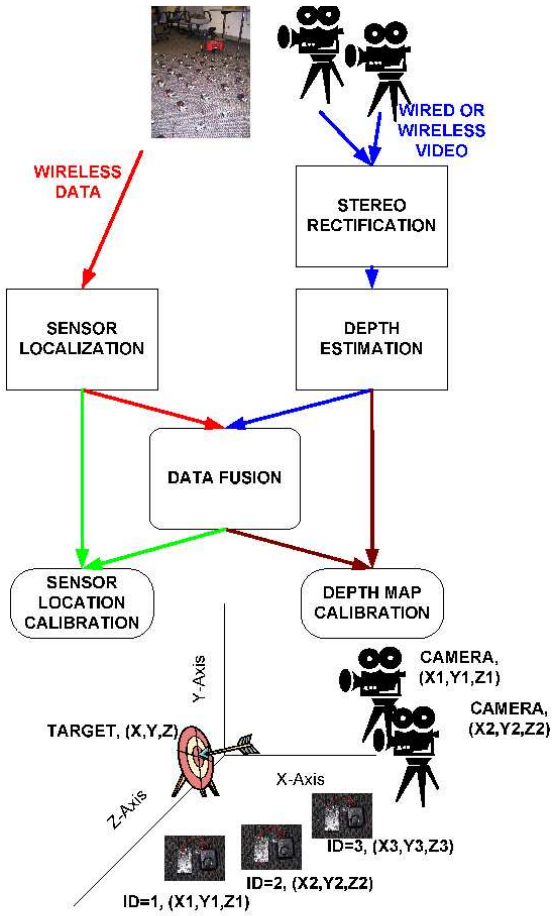

Figure 1.1: Flow Chart of Stereopsis and Localization Fusion

1.2 Computational Stereopsis

Computational stereopsis is the problem of recovering the depth of a scene from multiple 2-D images using a computer. If we consider the case of two images, the images are referred to as a *stereo pair*. A *depth map* is a third image with each pixel containing the distance from the corresponding scene point to the camera's image plane.

Chapter 2 reviews techniques that extract and calibrate a depth map from a stereo pair. These stereopsis techniques allow us to use a camera to acquire additional, possibly redundant, 3-D information from a scene instrumented with a WSN.

1.3 Wireless Sensor Networks

1.3.1 Definition of Terms

For the purposes of this thesis, a wireless sensor network is a collection of sensors with the following properties:

1. Each sensor can operate under its own power
2. Each sensor has the capability to communicate with other sensors via a radio link
3. Each sensor has computational resources, i.e. a CPU
4. Each sensor has small data storage capabilities

These sensors are commonly called *motes* in the literature and this convention is retained. Wireless sensor networks are often mentioned in conjunction with micro-electro mechanical systems (MEMS) as the recent surge in novel MEMS sensors is well-aligned with the goals of wireless sensor networks: small size, the ability to be integrated in a single device, and low power usage. As the problem this thesis addresses is independent of the sensors used to collect data, no knowledge of MEMS is needed.

An example of a mote is the Crossbow MPR400 [14], commonly known as the MICA2. The MPR400 uses add-on boards to perform the sensing such as the Crossbow MTS420 [15]. The MPR400 and the MTS420 are shown in Figure 1.2.

Figure 1.2: Crossbow MPR400 MICA2 Sensor and Crossbow MTS420 Sensor Board

1.3.2 Applications of Wireless Sensor Networks

The obvious application of wireless sensor networks is large-scale data collection, especially in regions where wired infrastructure does not or can not

exist. This lack of wired infrastructure may be due to the temporary nature of the site (e.g. a battlefield), or cost. For example, the authors of [3] note that the costs of wiring are going to dominate the cost of sensing in the long-term. Software such as TinyDB [44] has been developed which makes this mode of operation a reality today and companies such as Dust Networks [18] and Crossbow Technology, Inc. [13] are working to commercialize and capitalize on this technology. One prominent example of large-scale data collection is environmental monitoring, as described in [11] and [27]. The WSN used in these applications is the same as in traditional environmental sensing, only without wires.

A variant of the large-scale data collection applications utilizes the processing capability of the motes to aggregate data and potentially make data-driven proactive decisions. This proactive aggregation of data can potentially reduce the amount of communication in the network, thereby reducing the power requirements and increasing the network lifetime.

More recent and novel applications of WSNs seek to utilize the autonomous character and small size of the motes. An example is [6] which uses a WSN to calibrate thermal infrared (IR) images with *in situ* measurements. Another emerging field is the development of "hazard-aware spaces": spaces that use WSNs to monitor and react to hazards (e.g. a fire) [7].

As applications move farther away from the basic "monitor and report" paradigm, the sensor infrastructure becomes increasingly important. Infrastructural software, similar to TinyDB [44], needs to be developed that robustly solves other problems such as localization. Estrin et al. [19] gives a nice overview of many areas of WSN research largely made possible by the advent of TinyOS [45] which standardized the sensor research platform. This thesis presents a prototype which may form the basis of such a system seeking to integrate WSNs with traditional sensors such as cameras.

1.3.3 Limitations of Wireless Sensor Networks

Wireless sensor networks are still very young and inherently at the state-of-the-art in many disciplines. There are many open problems related to deploying WSNs [19], [1]. The mote power constraint, in particular, proves to be challenging since it determines how long the motes and the WSN can operate. Other factors that constrain WSN operation are: limited on-board memory, a simple CPU, and a limited broadcast range.

On the WSN level, additional limitations appear: localization, the problem of determining the physical location of motes; time synchronization

between motes; reliable multi-hop message routing; and efficient communication protocols.

1.3.4 Localization in Wireless Sensor Networks

We are interested in the sensor localization problem, i.e. automatically mapping sensors to the world at large. As an example, consider the wireless sensor network deployment described in [26]. Sensors with known identifiers (IDs) are affixed to large machines and vibration data is read from the network. If a given sensor is registering an unexpected vibration profile, the monitoring process/person knows which machine should be inspected. In the more well-known Great Duck Island experiment [27], one sensor per nest of birds was used in the monitoring. In both examples, the mapping between sensor identifier and the item of interest is known. This thesis reports on work that does not make the strong assumption of a known mapping between sensors and locations.

1.4 Thesis Contribution

The contribution is in building a first prototype for improving our knowledge of scene geometry and sensor locations based on camera and WSN data fusion. This prototype is built on top of TinyOS [45] and Image to Knowledge [2] and was used to generate all of the data in the remainder of this thesis. The appendices describe the operation of this prototype. As is discussed and quantified in chapter 4, this prototype system is able to increase our knowledge of scene depth.

1.5 Thesis Outline

First, chapter 2 discusses computational stereopsis. A discussion of localization in wireless sensor networks follows in chapter 3. The process of fusing data from computational stereopsis with localization data is presented in chapter 4 along with quantitative fusion results. Chapter 5 presents concluding thoughts and areas for future research. Appendices discussing the TinyOS [45] and Image to Knowledge [2] implementations of this thesis in enough detail to replicate the results follow.

Chapter 2

Computational Stereopsis

2.1 Introduction to Computational Stereopsis

Computational stereopsis is the problem of recovering the depth of a scene from multiple two dimensional images using a computer. Various aspects of this problem have been addressed in the past by many researchers in the computer vision, machine vision and signal/image processing communities [47], [16], [32]. The motivation for obtaining 3-D information often comes from applications that require object identification, recognition and modeling. There is an abundance of research and industrial use of 3-D information for (1) designing autonomous vehicle movement (collision avoidance and path planning), (2) performing teleoperation of vehicles (industrial robots, space rowers, aircrafts, and cars), (3) determining medical diagnosis with non-invasive methods (MRI, CT, X-Ray, ultrasound), (4) modeling urban sites for military or communication purposes, and (5) developing augmented reality for training and telepresence.

2.2 Problem Statement of Computational Stereopsis

Stereopsis is the construction of three-dimensional geometry given multiple views of a scene as in [39], [40], [17]. Computational stereopsis is the science of using computers to perform stereopsis. Figure 2.1, from [39], shows a generalized stereopsis configuration. The cone on the checkerboard pattern represents a scene. Points $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$ represent optical centers of three [pinhole] cameras. I_1, I_2, and I_3 represent the image planes of these cameras: the inputs to a stereopsis algorithm. In the general case, the stereopsis problem can be posed as the reconstruction of the scene geometry given the two-dimensional data (images) $I_1, I_2, I_3, \ldots, I_n$.

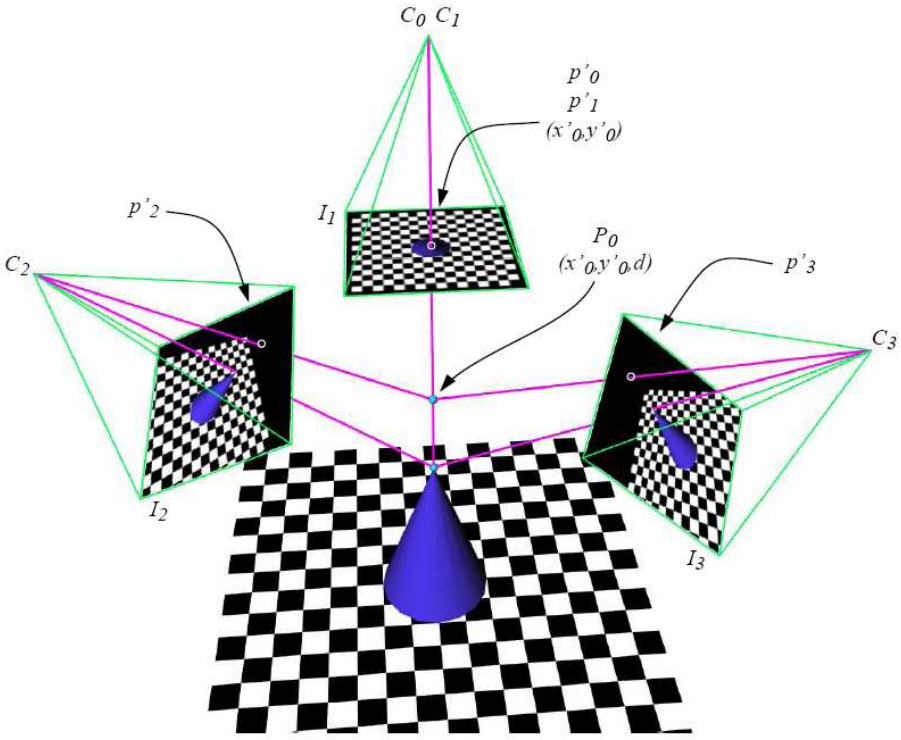

Figure 2.1: Generalized Stereopsis Configuration from [39]

A simplification of the general stereopsis configuration is the case with two images at a time. As it is readily apparent from Figure 2.1, the 3-D reconstruction of a scene point is straight forward given matching points on the images. The scene point can be calculated as the intersection of the two lines passing through the matched points and the optical centers. With known camera parameters, this setup reduces computational stereopsis to a problem of image matching. Some stereo image matching techniques are described in more detail in section 2.5.

A natural question to ask is: What can be determined if camera parameters are unknown? The reference to camera parameters includes both intrinsic (e.g., lens distortions) and extrinsic (e.g., camera position) parameters. The intrinsic parameters are usually estimated from specification sheets provided by camera manufactures while the extrinsic parameters are controlled during the image acquisition, for example, by using stereo-rigs [40]. We do not consider the case of unknown intrinsic parameters and we deal with the case of unknown extrinsic parameters only. Unknown extrinsic parameters naturally occur when using images taken from unknown scene positions. Not having to rely on stereo rigs or precisely placed cameras is

important in the "real world" as existing cameras are not likely to be of this type or need to be mobile (e.g. security cameras). It is well known that without extrinsic parameters, stereopsis can still extract scene depths, albeit not to scale [17].

2.3 Related Work on Computational Stereopsis

The stereopsis component of our fusion system is a "pair" of visible spectrum cameras created by precisely moving a single camera to image a stationary scene. Contrary to wireless sensor networks (WSNs), cameras are viewed as traditional sensors and have proven to be reliable, relatively inexpensive, and suitable for collecting a dense set of measurements (a raster image) from their environment. Many techniques have been developed in the past two decades that can extract shape information from images and video [47]. For example, Pankati and Jain in [35] cover extracting shape from multiple cues. Many applications of computational stereopsis exist including object recognition, room geometry determination for robot path planning, extraction of land elevation from aerial photographs, and investigations into the human visual system brain [32]. In our work, we focus on stereopsis using two images to derive a depth map.

2.4 Stereo Rectification

In this section, we focus on the special case of stereopsis without knowledge of extrinsic camera parameters. In this case, it is useful to perform "stereo rectification" on the images prior to attempting image matching. Stereo rectification is a process which aligns one of the images (taken to be the right image of a stereo pair in this thesis) such that matching points in the resulting images are on the same "scanline" (image row or y-coordinate). The resulting images form a "rectified stereo pair" that corresponds to a configuration with cameras displaced purely horizontally from each other (see Figure 2.2).

Stereo rectification serves two main purposes. First, it simplifies the geometry of the stereopsis problem tremendously. In the rectified images, everything can be expressed in terms of *disparity*, namely the distance between pixels in one image and the matching pixels in the other image (Figure 2.3). In general, each image point may have a unique disparity associated with it which is inversely proportional to the depth of that image point in the scene. The second, and perhaps more important, purpose of stereo rectifica-

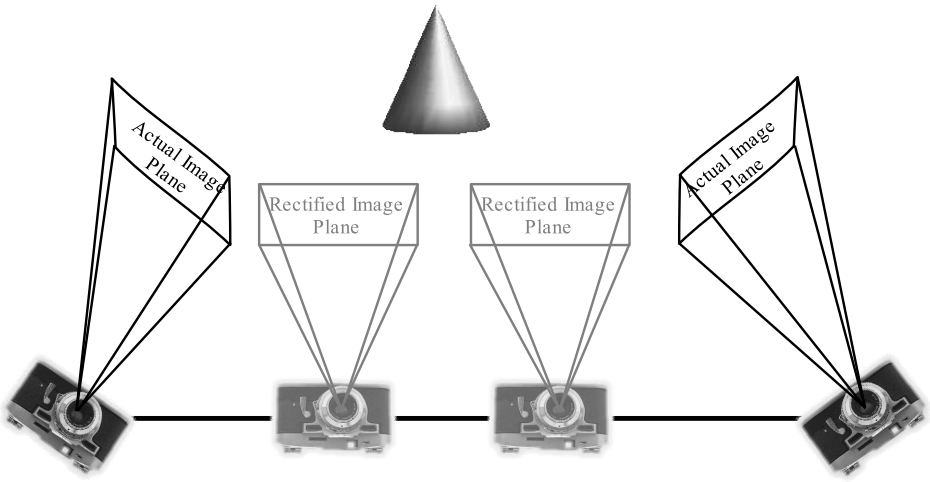

Figure 2.2: Stereo Rectification

tion is that the image matching task is simplified by using rectified images. If the rectification is successful, then the correct match for any pixel in an image would be found along the same scanline in the other image, and hence reduce a two-dimensional search per pixel to a one-dimensional search.

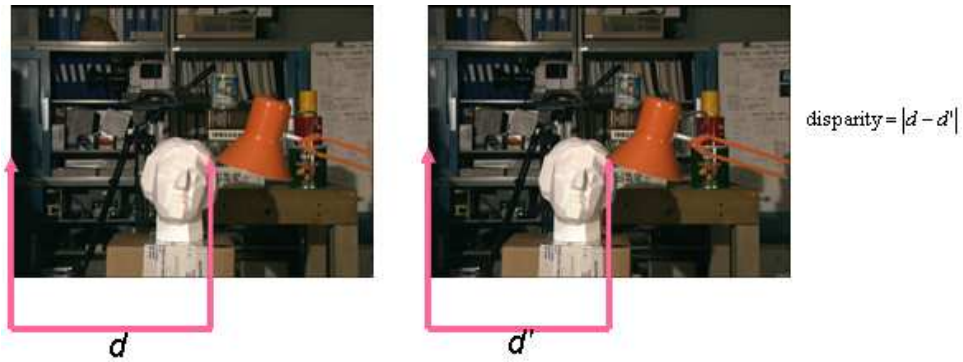

Figure 2.3: Definition of Disparity using Tsukuba Stereo Pair [41]

We decided to follow the approach proposed by Hartley in [22] and implement the algorithm as one of the Image To Knowledge (I2K) software tools [2]. Hartley's technique allows us to find a "matched pair" of rectifying homographies (perspective projections), such that the epipole of the right image is mapped to infinity and epipolar lines in both images are equal. Isgro and Trucco describe an implementation of Hartley's approach in [28]. Instead of finding and mapping the epipole manually (as Hartley suggests), Isgro and Trucco recognize that the fundamental matrix from a rectified pair of images is:

$$F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.1}$$

They then propose using pairs of matched image points to numerically compute the homographies using the Levenberg-Marquardt [36], [31] and standard linear least-squares algorithms. The former is used to minimize the following cost function:

$$F(H_1, H_2) = \sum_{i=1}^{N} [(H_2 \mathbf{p}_{2_i})^T F H_1 \mathbf{p}_{1_i}]^2 \tag{2.2}$$

The latter is used to minimize:

$$\sum_{i=1}^{N} [(H_1 \mathbf{p}_{1i})_x - (H_2 \mathbf{p}_{2i})_x]^2 \tag{2.3}$$

Minimizing this equation ensures that the rectified x-coordinates of the images are not displaced "too much" (i.e. that the image will not be ripped apart by the first constraint). Any pair of homographies satisfying these equations is sufficient, but we choose a pair of homographies so that H_2 is constrained to be a rigid transformation about a point in space.

To rectify an image pair, we just reproject the second ("right") image using:

$$\mathbf{p}'_2 = H_1^{-1} H_2 \mathbf{p}_2 \tag{2.4}$$

We have implemented this approach in the Image to Knowledge StereoRectify tool accessible through the Stereo Tool from the main menu (see the documentation on Stereo in [2]). Matching image points are currently specified by hand, as automatic image point/feature matching is another area of research beyond the scope of this thesis.

Figure 2.4 demonstrates the functionality of the stereo rectification tool implemented in I2K. The top set of images is an unrectified image pair. The second set of images has the rectified "right" image. The colored lines have been overlaid on the same scanlines in both sets of images. As an example, one can look at the top line across the images. This line has been selected to intersect the sprinkler head in the left image of both pairs. By comparing the distances between the sprinkler head and the magenta line in the right images, one can obtain better understanding of what the stereo rectification algorithm accomplishes.

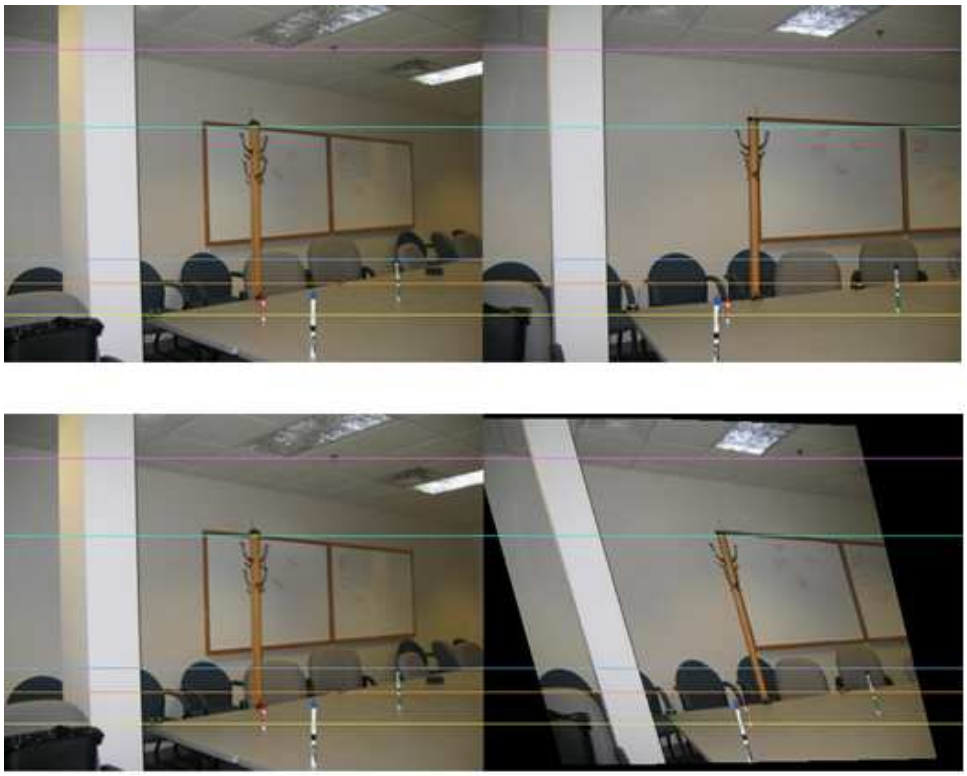

Figure 2.4: Stereo Rectification Example

A practical problem in the stereo rectification algorithm implementation is its sensitivity to the trivial solution of zero. The Levenberg-Marquardt numerical technique that is used is akin to a sophisticated gradient descent. A local minimum is desired, but the global minimum of zero is easily reached and has strong influence in practice. The rectified image displayed above, although improved in some ways, exhibits some behavior which can be attributed to this problem. Visibly, the column on the left side of the image is skewed in the rectified image more than one would expect for a simple, horizontal camera translation (the model that the rectified image pair should mirror). This, in turn, can mislead the image matching techniques discussed in section 2.5.1. Robust, automatic stereo rectification, although it would be useful, still appears to be a hard problem worthy of additional research. In the absence of a rectified image pair, one can design the system to produce pre-rectified image pairs. Using a single camera mounted on a tripod, this is accomplished by moving the camera such that the stereo pair image planes are coplanar. The experimental results presented in this thesis use this approach.

2.5 Stereo Image Matching

Once a stereo image pair has been rectified, as in section 2.4, the problem of computational stereopsis becomes one of image matching. We want an algorithm that takes a pixel in one image and efficiently finds the pixel in the other image which corresponds to the same scene point. As defined in 2.4, this difference is the *disparity* and is inversely proportional to the depth of the scene point:

$$z_{scene} = \frac{\alpha}{\text{disparity}} \tag{2.5}$$

There are cases where the disparity is undefined, e.g. a point that is occluded in one of the stereo images. These issues are discussed in section 2.5.1.

There are two predominant approaches to the stereo matching problem: correlation, and graph cuts. Correlation matching is simple in theory and can be described as follows. For each pixel in the left image, find a matching pixel along the same scanline in the right image by comparing two windows centered on the selected pixels using a correlation metric [10]. This scheme can be modified by (a) using adaptive windows, further limiting the search space along the scanline, or (b) doing the search in a multiscale fashion [24]. These modifications generally aim to increase the speed or robustness of the match. See section 2.6 for details on the modifications implemented to the correlation algorithm used in I2K. The main disadvantage of correlation matching is that every pixel fends for itself. This generally leads to good matches along each scanline, but can lead to inter-scanline discrepancies and errors (e.g. jagged edges of objects).

Graph cut matching is similar to correlation matching in that a similar distance metric is used to decide what a good match is. Unlike correlation matching, graph cut techniques look to minimize global "costs" and can therefore penalize inter-scanline discrepancies. Briefly, graph cut methods formulate the stereo matching problem as a graph theoretic maximum flow problem. Solving the flow problem (using a known algorithm such as Edmunds-Karp [12]) also gives a depth map solution that minimizes a global cost given a penalty weight. The advantage to graph cut techniques is that they have produced one of the best computational stereopsis results to date [41]. The disadvantage of graph cut algorithms is their computational complexity: they are time-consuming relative to correlation (e.g. an execution can take many minutes for a single stereo pair).

2.5.1 Limitations

Regardless of the image matching technique used, the main obstacles to successful stereo matching are scene occlusions and mismatches.

The problem of occlusions arises from regions of the stereo pair that are absent from either image. The algorithm for stereo matching cannot a priori determine occluding regions, so matching errors are highly likely in these regions. There has been considerable research towards identifying and handling occluding regions [10], but these discussions are beyond the scope of this thesis. A simple strategy to detecting any occluding regions is to run stereo matching twice, the first time searching for matches to the left image and the second time searching for matches to the right image. If the two runs agree on disparities, the result is kept. If the two runs disagree on disparities, the result is thrown out and considered to be in an occluding region.

Stereo mismatches can also occur when the image matching is not a one-to-one problem. This is very likely to happen if the scene does not contain any visually salient features (it could be called a visually "boring" scene), namely scenes which have wide areas with absolutely no details to match against (e.g. a white wall with even lighting), or scenes with man-made objects exhibiting regular patterns (e.g. textured areas, or similar features like the windows on a skyscraper in 2.5). In all of the preceding cases, a small distance between [image matching] windows can mistakenly be classified as an incorrect match. A large number of incorrect matches will generally produce useless output. Using techniques discussed above to reduce the search space can help eliminate the effects of "boring" scenes. Similarly, global techniques, such as stereo using graph cuts, will avoid these problems. Global techniques would converge on a global minimum to promote the correct matches if given enough context. Other than these techniques, there is little that can be done algorithmically. One technique that does work in practice is to introduce texture into a scene (e.g. with large amounts of newspaper, or similar non-regular textures). We have used this method in some of our experiments with favorable results.

2.6 Implementation in Image to Knowledge

In Image to Knowledge [2], we implemented a multi-scale, correlation-based stereo image matching technique. It is implemented in the Stereo class and is accessible through the Stereo Tool interface. The correlation technique we

Figure 2.5: A hard-to-match object due to repetitive similar features (windows)

use was proposed by Hirschmuller in [24] and differs from straight-forward correlation in its use of an adaptive window shown in figure 2.6. The adaptive window is really composed of five windows. The "main" window is the red window in the center of the figure. It is surrounded by four offset "secondary" windows. The final correlation is computed by summing the correlation value from the "main" window with the correlation values from the two best "secondary" windows. This approach is used with the left-right consistency check described in section 2.5.1 to identify occluded regions. All together, this algorithm corresponds with steps 1 through 4 of Hirschmuller's algorithm in [24]. As shown in [24] and [41], this algorithm performs fairly well when compared with other stereo algorithms, especially those based on correlation, on reference stereo pairs. This algorithm is also termed as "real-time" in [24] and [41], although it is not in our implementation. The graph-cut algorithms, while producing higher quality results, are much harder to implement and have long running times. We did not use them in this thesis because of these reasons.

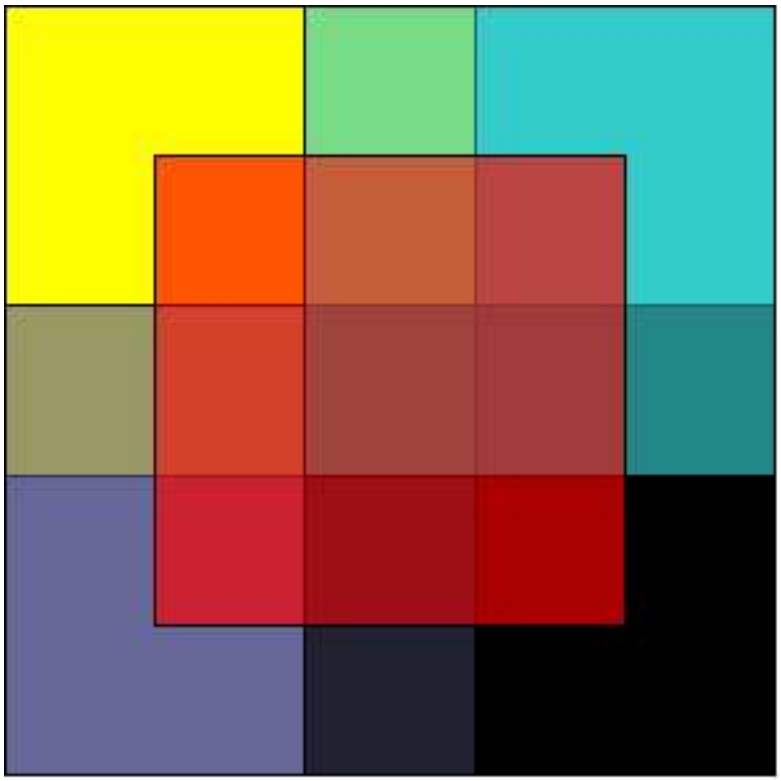

Figure 2.6: Adaptive Correlation Window

Results from our stereo image matching follow in Figures 2.7 through Figure 2.10. The first two images in each figure are the stereo pair input to the algorithm (some of which are rectified, some of which are not). The third image shown in each Figure is a depth map, the inverse of the disparity map, either in grayscale, or pseudo-color (if hard to see otherwise). Note that black (grayscale) or dark blue (pseudo-color) represents those pixels which are of unknown depth (i.e. those that failed the left-right consistency check). The fourth image, if present, represents the "ground truth" data. We will come back to depth map calibration and the validity of the results in section 2.7.

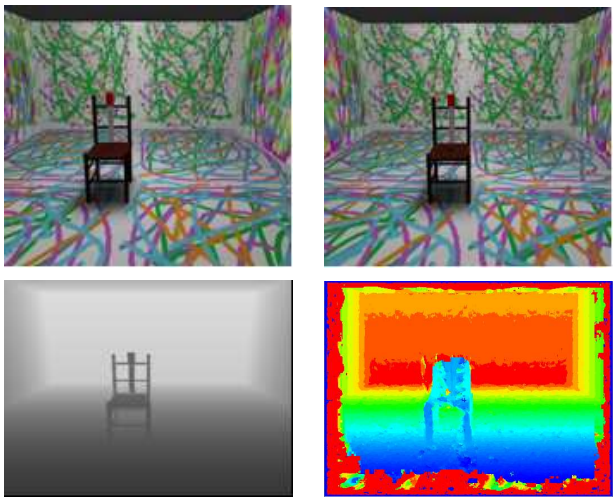

Figure 2.7: Synthetic Images. Top Left and Top Right: The input [rectified] stereo pair generated with POV-Ray, Bottom Left: Ground truth depth map generated with POV-Ray (white: far from viewer, dark: close to viewer), Bottom Right: Pseudo-color computed depth map (black: invalid, cool: close to viewer, warm: far from viewer)

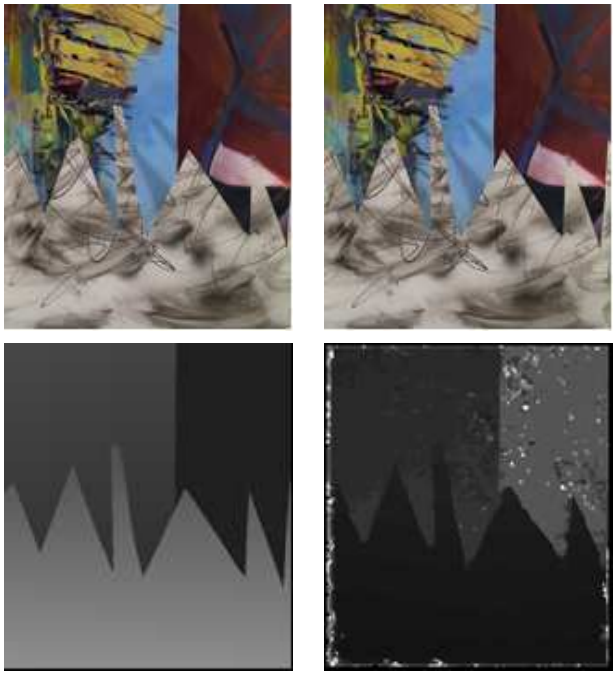

Figure 2.8: Measured Stereo Images ("Sawtooth") from [41]. Top Left and Top Right: The input [rectified] stereo pair, Bottom Left: Ground truth depth map obtained from [41] (white: close to viewer, dark: far from viewer), Bottom Right: Computed depth map (white: far from viewer, dark: close to viewer)

Figure 2.9: Measured Laboratory Images (Room without Added Texture). Left and Middle: The input [rectified] stereo images, Right: Computed depth map (black: invalid, cool: close to viewer, warm: far from viewer)

Figure 2.10: Measured Laboratory Images (Room with Added Texture). Left and Middle: The input [rectified] stereo images, Right: Computed depth map (black: invalid, cool: close to viewer, warm: far from viewer)

2.7 Depth Map Calibration

The image matching step, discussed in section 2.5, leaves us with a disparity map. Equation 2.5 shows us that if we invert each element of the disparity map, we will obtain a scaled depth map. This section solves the problem of finding the correct scaling. In order to be more general and allow for systematic errors in the determination of the disparity map, we also allow for an offset in the fit, namely:

$$z_{scene} = \alpha z_{depth\ map} + \beta = \frac{\alpha}{\text{disparity}} + \beta \tag{2.6}$$

We next assume that we have a set of known, coplanar points in the image corresponding to known points in the scene. The points in the scene are trivially coplanar as well. In general, one would desire to identify a large plane, both in terms of points on it, and in percentage of image covered. Techniques for finding a planar subset in a collection of 3-D points exist in the literature, for instance [34]. In this thesis, we manually select points

from the stereo pair that lie on a plane in a world coordinate system defined by the left camera.

We continue by finding the parameters of the plane in 3-D that fits the "world" coordinate locations using a linear least squares approach and minimizing the squared error in z (the axis perpendicular to the image plane). We fit the plane $ax + by - z + d = 0$ to the real-world coordinate locations. Using the computed plane parameters (a, b, d) we then make the substitution for z_{scene} using Equation 2.6 and solve (in a linear least squares fashion) the resulting problem over all known points to obtain α and β. At this point the depth map can be calibrated and the results compared against what is known.

The above technique, using manually specified points, has been implemented in the Image to Knowledge Stereo Tool.

2.7.1 Depth Map Calibration Results

Section 2.7 covered the implementation of the fusion algorithm used in Image to Knowledge. Section 2.7.1 details our evaluation methodology for this algorithm. The quantitative results from our experiments are reported in section 2.7.1.

Methodology for accuracy evaluations

Ultimately, we measure our fusion performance by the accuracy of the resulting depth map relative to a ground truth depth map. In practice, the ground truth depth map is generally not available, or is very difficult to obtain. The situation is a bit different in our experimental setup as we have the ability to acquire ground truth measurements. We conducted experiments with theoretical/synthesized stereo pairs and actual/measured stereo pairs. In the synthetic image case (e.g. Figure 2.7), we can generate a dense, theoretically correct depth map. In the real-world cases, we do not have the luxury of a dense depth map and must resort to a relatively small set of points at hand-verified distances.

Another issue is the error metric for comparing ground truth depth maps with estimated depth maps. We consider two error metrics: (a) the average absolute distance error for each pixel/hand-verified point, and (b) the average absolute distance error as a percentage of maximum measured range in the image. Both values decrease with more accurate calibration (fusion) and are asymptotically optimal (zero). The accuracy evaluation methodology for each set of input images is outlined next.

Methodology for Synthetic Images

1. Create a synthetic stereo pair
2. Compute a theoretical depth map based on the geometry of the scene
3. Compute an uncalibrated depth map $z_{depth\,map}$ from a stereo pair of images using the I2K Stereo Tool
4. Using four points from step 2 that fall in a plane, calibrate the depth map from step 3 using the I2K Stereo Tool to obtain z_{scene} based on Equation 2.6
5. Compute the average absolute distance error using all image points (perhaps excluding a border of a given width)
6. Compute the average absolute distance percentage of maximum measured range. The maximum distance of the points from step 2 is "Max Range" and we use: Error % Max Range $= \frac{\text{Avg. Abs. Distance Error}}{\text{Max Range}} * 100\%$

Methodology for Real Images

1. Take a stereo pair of a real scene
2. Record manually distance measurements to N points ($N > 4$) in the scene. Four of these points will be used for calibration and should lie in a plane. One point should represent the maximum range of the image (used in step 6)
3. Compute an uncalibrated depth map $z_{depth\,map}$ from a stereo pair of images using the I2K Stereo Tool
4. Using the four points from step 2, calibrate the depth map to obtain z_{scene} based on Equation 2.6
5. Compute the average absolute distance error using all points from step 2: Avg. Abs. Dist. Error $= \frac{1}{N} \sum_{i \in \text{points}} |z_{scene} - z_{ground\,truth}|$
6. Compute the average absolute distance percentage of maximum measured range. The maximum distance of the points from step 2 is "Max Range" and we use: Error % Max Range $= \frac{\text{Avg. Abs. Distance Error}}{\text{Max Range}} * 100\%$

Table 2.1: Results obtained for a synthetic stereo pair consisting of a single plane perpendicular to the camera

	Number of Points	**Scale** α	**Offset** β	**Avg. Absolute Dist. Error**	**Maximum Image Range**	**Error % of Max Range**
Scale Only	57408	0.616	N/A	0.512	12	4.26%
Scale and Offset	57408	-0.702	19.925	0.054	12	0.45%

Computational Stereopsis Results

We performed a number of different experiments in order to evaluate quantitatively the accuracy of results as a function of scene texture and calibration model complexity. Specifically, we conducted experiments that change the amount of texture in the scene (which affects the quality of the stereo output). We also include results which have been calibrated using only "scaled" depth maps (assuming). We observed that the calibration results under the assumption of led to smaller error in some cases, seemingly due to the sensitivity of calibration to the quality of the plane fit.

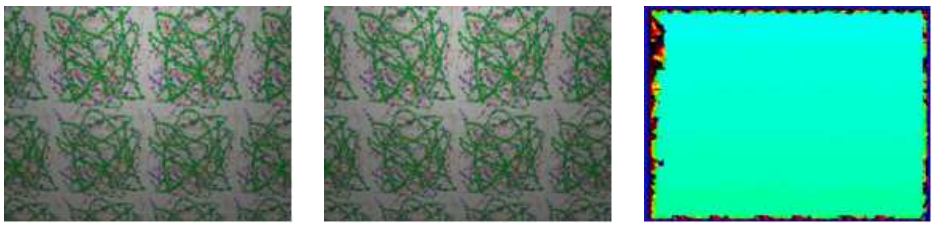

Figure 2.11: Synthetic Plane. Left and Middle: The input [rectified] stereo pair generated with POV-Ray, Top Right: Computed depth map (pseudo-color)

As an algorithm test, we generated a synthetic stereo pair (Figure 2.11) consisting of a single plane perpendicular to the camera (e.g. a normal pointing along the z-axis). The generated stereo images are of size 512 by 384. We fitted a plane to four image points, as before, and verified that the fitted plane had a normal along the z-axis regardless of the specific implementation of a stereo method. The scale and offset factors differed, however, leading to differing errors in the calibrated depth map. Our results (excluding a border of width 100 pixels, 70% of the image area) are summarized in Table 2.1.

Table 2.2: Calibration Evaluations

Stereo Pair	**Number of Points**	**Avg. Absolute Dist. Error** $\alpha \neq 0, \beta \neq 0$	**Avg. Absolute Dist. Error ("scaled only" or** $\beta = 0$**)**	**Maximum Image Range**	**Error % of Max Range (Scale and Offset/Scale)**
Synthetic (Figure 2.7)	17808	7.664	2.731	12	63.8% 22.76%
Untextured Room (Figure 2.9)	15	1.771m	2.546m	7.7m	23% 33.1%
Textured Room (Figure 2.10)	15	1.269m	1.412m	7.7m	16.5% 11.8%

During testing with real stereo pairs, the phenomenon of near-zero scaling factors occurred numerous times, creating very large calibration errors. We believe this arises due to overfitting of our calibration points. The errors in the chosen points allow for a local minimum "fit" that is just their average (i.e. purely an offset), rather than a purely scaled, or mixed solution that approaches the global minimum error. Theory predicts that scaling is the only operation needed to achieve the global minimum, so incorporating this a priori knowledge into the fitting step is the "correct" thing to do. In real and synthetic stereo pairs, scaling does not always produce better results. The cases where this occurs are the same cases where stereopsis fails, specifically with regions that lack texture. Fortunately, both "scaling" and "scaling and offsetting" result in empirically similar numbers when considering feature points in real, textured scenes (e.g. Figure 2.10): our target application. We present both figures in our calibration results shown in Table 2.2.

Interestingly, we find that performance with measured images is better than performance with synthetic images in terms of distance accuracy as a percentage of maximum image range, a useful measure of calibration accuracy. We attribute this to the simplicity of our synthetic scene compared with the complexity of the actual room, and to the large depth discontinuities within the chair object (e.g. chair rungs) which significantly contribute to error. In the actual room, especially in the textured case, there are more unique "textures" that the stereo algorithm can match against resulting in a better depth map. Performance is better in the textured room than in the untextured room for the same reason. Another reason for the difference

between the synthetic and measured performance is due to the density of ground truth points. In the actual room, we measured only certain, "easily identifiable" points, while in the synthetic scene we knew 3-D locations of all image points. Stereo image matching does very well with distinctive points thereby biasing our comparison in favor of the real scenes.

We do not have data on how these range estimates compare with range estimates obtained with calibrated cameras (i.e. when the intrinsic and extrinsic parameters of both cameras are known). We leave this comparison as an area for future research.

2.8 Derivation of Depth Uncertainty

In order to make an informed data fusion decision, a model of the depth uncertainty from computational stereopsis must be formed.

The central relation in computational stereopsis relates the depth of points in the scene to the disparity (the output of the image matching step) between the stereo images. Specifically, for our setup this relation is:

$$z_{scene} = \frac{\alpha}{\text{disparity}} + \beta \tag{2.7}$$

We can consider the perfect disparity to be a real number $disp_{ideal}$, however our image matching algorithm only determines matches at pixel granularity. Additionally, the image matcher is not perfect and has an average error of $disp_{m.e.}$. This means that the disparity we actually use for computation falls in the range:

$$(\lfloor disp_{ideal} \rfloor - disp_{m.e.}, \lceil disp_{ideal} \rceil + disp_{m.e.}) \tag{2.8}$$

If we assume that the disparities we use in calculations fall uniformly across this range, we find that the average distance error is:

$$\text{error}_{dist} = \frac{\frac{\alpha}{\lfloor disp_{ideal} - disp_{m.e.} \rfloor} - \frac{\alpha}{\lceil disp_{ideal} + disp_{m.e.} \rceil}}{2} \tag{2.9}$$

Since we obviously never know $disp_{ideal}$ in practice we make the approximation that $disp_{ideal} = disp_{depth\ map}$. The latter is determined using Equation 2.7. The important thing to note about Equation 2.9 is that it is an increasing function of depth. This makes sense since in the limit, as z_{scene} approaches ∞, $disp$ approaches 0. Since we rely on the disparity to discern depth, we expect to see low error where we can measure the most disparity and vice-versa. Hence, stereopsis is theoretically more accurate

closer to the camera, than farther away.

This relationship is explored in detail in [25] and is shown graphically in Figure 2.12. Stereopsis can be viewed as a mapping from each point in a stereo pair to one of the regions shown in Figure 2.12. The region close to the camera, marked by A, has much less depth uncertainty than the region farther from the camera, marked by B. This relationship will be compared with the localization error function in section 3.6.

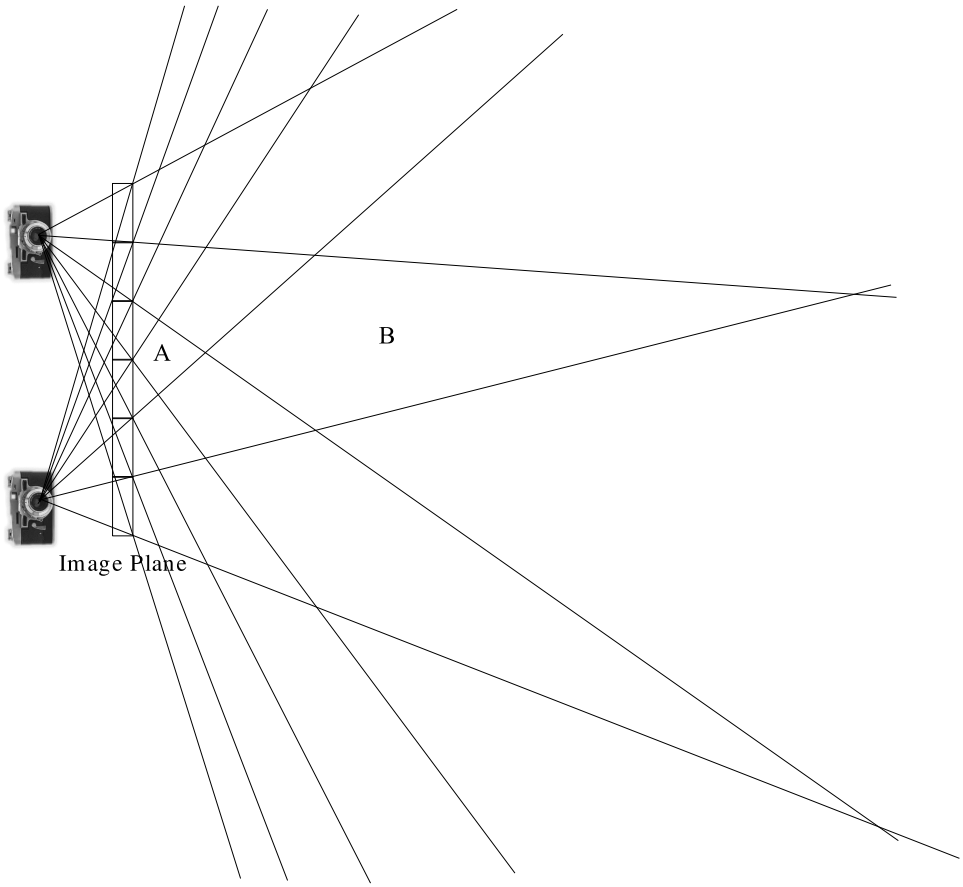

Figure 2.12: Stereo Uncertainty versus Distance from Camera

Chapter 3

Sensor Network Localization

3.1 Introduction to Sensor Network Localization

Sensor network localization is a technique central to sensor data fusion as it gives us a way to get sensor locations from within the sensor network. Section 3.2 describes the problem of sensor network localization. Major approaches to sensor network localization are considered in section 3.3. Ranging techniques, a common way to perform distance measurements inside a sensor network, are discussed in subsection 3.3.1. Section 3.4 details the implementation of localization within TinyOS and Image to Knowledge. Steps to calibrate the point-to-point localization results are given in section 3.5. Section 3.6 shows the empirical relationship between point-to-point ranging error and localization error.

3.2 Problem Statement of Sensor Network Localization

Sensor network localization is the problem of finding out the locations of all sensors in a wireless sensor network. Depending on the application, the localization problem can take many forms. For instance, in MIT's Cricket application [33] localization is used to identify the room (or part of a room) that some sensor is in. This can be used for (a) service discovery, (b) enabling functionality (e.g. environmental control), or (c) providing functionality based on the high-level location of the device.

For the applications considered in this thesis, we are interested in knowing the coordinates of the sensors in space relative to a coordinate system defined by the position of one of our stereo cameras. In our implementation, the global coordinate system is centered on the left stereo camera.

3.3 Related Work to Sensor Network Localization

There are two major approaches to sensor network localization. The first approach relies on existing localization infrastructure, such as GPS. In this scenario, each mote in the network carries a GPS receiver [15]. The mote's GPS receiver receives satellite broadcasts in order to find its global coordinates (i.e. latitude and longitude). The motes then convert these global coordinates into local coordinates relative to a chosen coordinate system. The major advantage of this localization system is that it is based on very solid technology. Current (2003) consumer GPS receivers, such as the Leadtek GPS-9543 [30], provide an accurate measurement to within 5m using differential GPS techniques. The cons of this approach are cost, power consumption, fine-grained accuracy and areas not covered by GPS. The cost primary derives from the GPS receiver and antenna. These costs will presumably drop over time, but not relative to the cost of MEMS sensors. The power consumption comes from the radio (and currently an additional processor) that is needed to receive the GPS signal. Since a GPS receiver needs to be active (powered) for a bit while locating and tracking satellites, this is not a battery-friendly operation. There are many areas that do not receive a GPS signal, notably almost any indoor location. Finally, 5m accuracy may be fine on a global-level, but is not very useful on a laboratory-scale. Other localization techniques that rely on infrastructure exist (e.g. triangulation from cell phone towers, beacons similar to those used in MIT's Cricket, etc...) and have similar tradeoffs.

The second approach to sensor network localization does not rely on external infrastructure. Instead, motes in the network attempt to locate themselves relative to neighboring motes. This "relative localization" can be done in a number of ways. One straight-forward way is ranging, namely using a technique to find the Euclidean distance between two motes. In this thesis, we focus on the relative localization approach.

3.3.1 Ranging

There are a number of ways to perform ranging as summarized in [23]. The most common [active] techniques are direct measurement, phase-based techniques, and techniques based on time of flight (ToF). Direct measurements are useful in static sensor deployments with few motes. Nonetheless, they become unpractical very quickly if not impossible due to time/location constraints. Phase-based techniques utilize multiple signals with differing

wavelengths and infer the distance based on phase differences between the received signals. Time of flight ranging is based on measuring the time of signal propagation and using knowledge of propagation speed to compute the distance to the object. An overview of these ranging techniques in terms of accuracy and system implementation is provided in Table 3.1.

The Crossbow MICA platform with the MTS300 sensor board, has limited ranging capability. Specifically, the only ranging capable hardware contained is a sounder, microphone, and tone detection circuit. The sounder and tone detection circuit are both tuned to 4 kHz which limits the practicality of all but the time of flight approach. Some research groups have experimented with custom sensor boards outfitted with ultrasonic transducers as ultrasound is a more traditional/refined market for ranging hardware. Good results using ultrasound ranging have been reported in [5] and [48]. We may investigate this technology in the future when it enters the commercial sensor board market.

Table 3.1: Ranging Techniques

Technology	**Example Implementing System**	**Reported Accuracy**
GPS	Leadtek GPS-9543 [30]	5m
Laser Phase-based Rangefinder	Acuity AccuRange 4000 [43]	0.003m
Ultrasound Time-of-Flight	ActiveBats [5]	0.03m
Acoustic Time-of-Flight	Calamari [48]	No Longer Available

Ranging Based on Acoustic Time-of-Flight (ToF)

As mentioned in section 3.3.1, acoustic time-of-flight ranging is both an accepted and easily implemented ranging technique. This section details the strategy of acoustic time-of-flight ranging we implemented (see Figure 3.1 for a diagram). The first step is to send a message to a ranging endpoint node. The endpoint node, after receiving the message, simultaneously broadcasts a radio ranging message with a 4 kHz chirp. Every node in the network is configured to listen for the radio ranging messages and starts a timer which stops when the audible chirp is heard. A broadcast message announcing the distance between the endpoint and receiving nodes is then sent for all who are interested.

Ranging is possible in this setup due to the differential in radio transmission speed (governed by the speed of light, the radio stack, and system-level issues) and the speed of sound in the sensing environment (we use 346.65

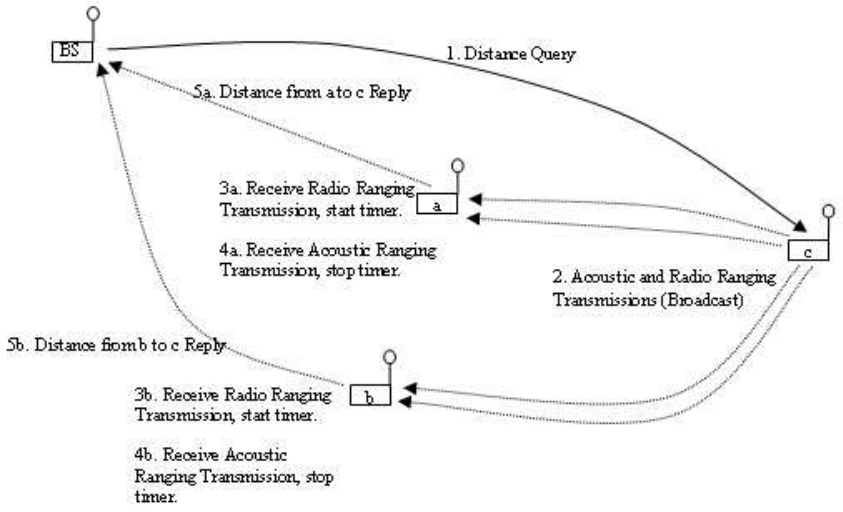

Figure 3.1: Acoustic Time-of-Flight Ranging

m/s for our experiments which corresponds to the speed of sound in air at 25 degrees Celsius). The granularity of the timer on the receiving nodes primarily dictates the uncertainty in the ranging estimates.

As mentioned in section 3.2, the goal of sensor network localization is to find the spatial coordinates of all the sensors in the network, not the set of ranges between sensor pairs. Theoretical work in converting ranging data into a set of locations was done in [37] and is briefly discussed in 3.3.1.

One Dimensional Localization

As the main contribution of this thesis is on the fusion, not the localization, simplifying assumptions have been made. We call our most restrictive set of assumptions for localizing n motes labeled $0..n-1$ the "rigid string approximations" as it assumes the distances between consecutive motes are known exactly:

1. [Rigid String in 1-D Assumption] We know precisely all [Euclidean] distances from a given mote i to a mote $i-1$.

2. We have ranging estimates, possibly with error, of Euclidean distances between mote i and motes $i-2$ and $i-3$ for all i.

3. We know the location of mote 0 and mote 1 (we assume mote 1 is just horizontally displaced from mote 0 by the distance known from assumption 1).

4. We assume that mote 2 is not below the horizontal line given by motes 0 and 1 (this is needed since we don't know the distance between mote 2 and mote -1: mote -1 does not exist!).

The following derivation of localization under these assumptions is shown since, (a) unlike the techniques in 3.3.1 it is deterministic and guaranteed to converge, and (b) it gives some insight into the challenges faced in converting ranges to locations.

There are two cases that we see. Note that in the following figures, the motes are labeled as 0, 1, 2, and 3. These numbers represent adjacent motes and should be considered as motes $i-3$, $i-2$, $i-1$, and i respectively (keeping the i notation makes the derivation messy). We always know the locations of motes 0, 1, and 2 and seek the location of mote 3.

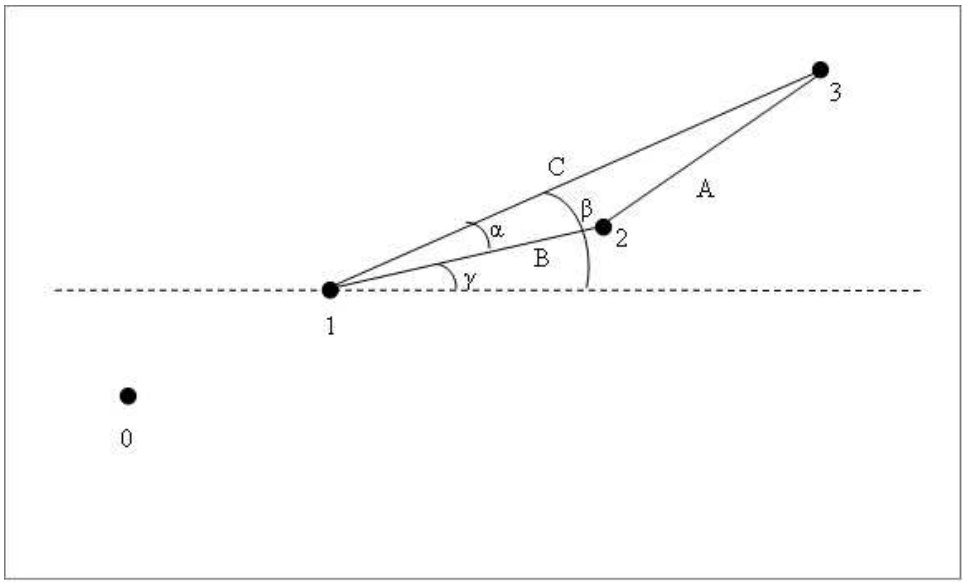

Figure 3.2: One Dimensional Localization: Case 1

Figure 3.2 and Figure 3.3 show the two cases we see with valid localization data. From the law of cosines, we get:

$$\alpha = \arccos\left(\frac{B^2 + C^2 - A^2}{2BC}\right) \tag{3.1}$$

From basic right triangle trigonometry we get:

$$\gamma = \arctan\left(\frac{dy_{21}}{dx_{21}}\right) \tag{3.2}$$

Now we have two choices for β: $\beta = \alpha + \gamma$ and $\beta = \alpha - \gamma$ for case 1 and case 2 respectively. To decide, we use d_{03} to see which choice gives us a

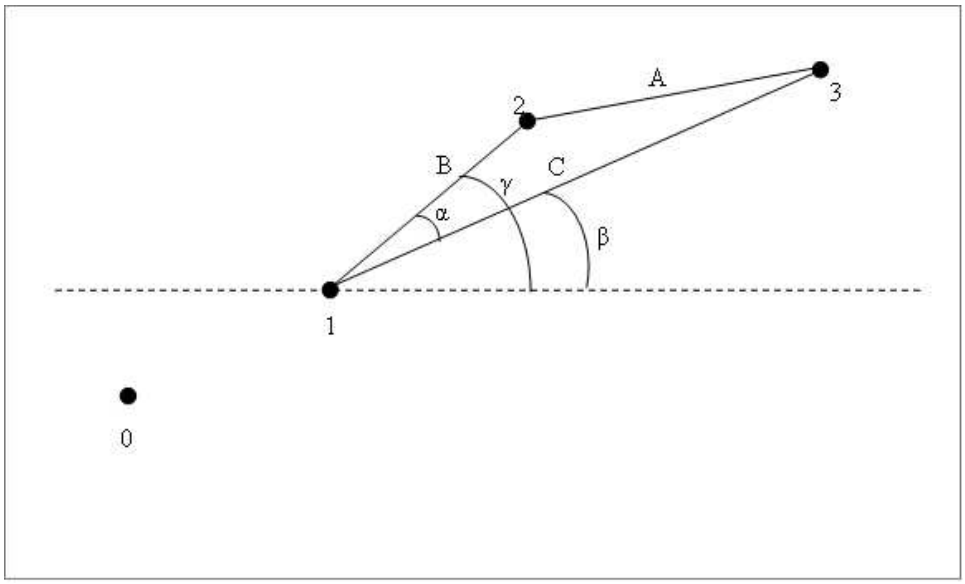

Figure 3.3: One Dimensional Localization: Case 2

smaller localization error (i.e. we compute our resulting d_{03} and compare it to the "known value" which will, in general, have error). In the case where the errors are equal, the case 2 value of β is preferred (this can be partially justified by "gravity" arguments as it is a state with lower potential energy for mote 3). Finally, we compute the location of mote 3:

$$\mathbf{P}_3 = (\mathbf{P}_{2x} + d_{13}\cos(\beta), \mathbf{P}_{2y} + d_{13}\sin(\beta)) \tag{3.3}$$

There are cases, due to errors in localization, where the triangle "collapses" and can not be solved. In this case, we pick:

$$\mathbf{P}_3 = (\mathbf{P}_{1x} + d_{13} + d_{23}, \mathbf{P}_{1y}) \tag{3.4}$$

Multidimensional Localization

The multidimensional problem is not always solvable, but the pathologic cases tend not to appear in practice [37]. The authors of [37] frame the problem in a graph theoretic manner. Let each mote be a node, and let the average ranging distance between motes be the undirected edge lengths. If a three dimensional embedding of the above graph exists and is globally rigid, we have a valid solution to the localization problem. The approach suggested in [37] to find such a solution has two steps: first the graph is "unfolded," then an iterative, "mass and spring" graph relaxation technique is used. The "resting lengths" of the springs are the ranging distances, so the solution

will even handle errors in ranging by converging to a "minimum energy" configuration.

We use a modified version of this algorithm that does not include an "unfolding" step. Specifically, we fix the location of the first mote at $(0, 0, 0)$ and assign all of the other motes a random location distributed randomly across the cube with corners $(0, 0, 0)$ and $(1, 1, 1)$.

At the start of each iteration we compute the "force" felt by each mote due to the influence of the other motes that have exchanged ranging data:

$$\mathbf{F}_i = \sum_{j \in \text{connected motes}} (d_{actual}(\mathbf{P}_j, \mathbf{P}_i) - d_{ranging}(\mathbf{P}_j, \mathbf{P}_i)) \frac{\mathbf{P}_j - \mathbf{P}_i}{||\mathbf{P}_j - \mathbf{P}_i||^2} \tag{3.5}$$

At the end of each iteration, the position of each mote is updated:

$$\mathbf{P}_i = \mathbf{P}_i + \frac{\mathbf{F}_i}{\text{Number of Neighboring Motes}} \tag{3.6}$$

The algorithm is terminated when the total amount of motion in the network falls below a threshold.

Similar to the results we will show in section 3.6, we empirically derive the relationship between point-to-point ranging errors and localization errors in Figure 3.4.

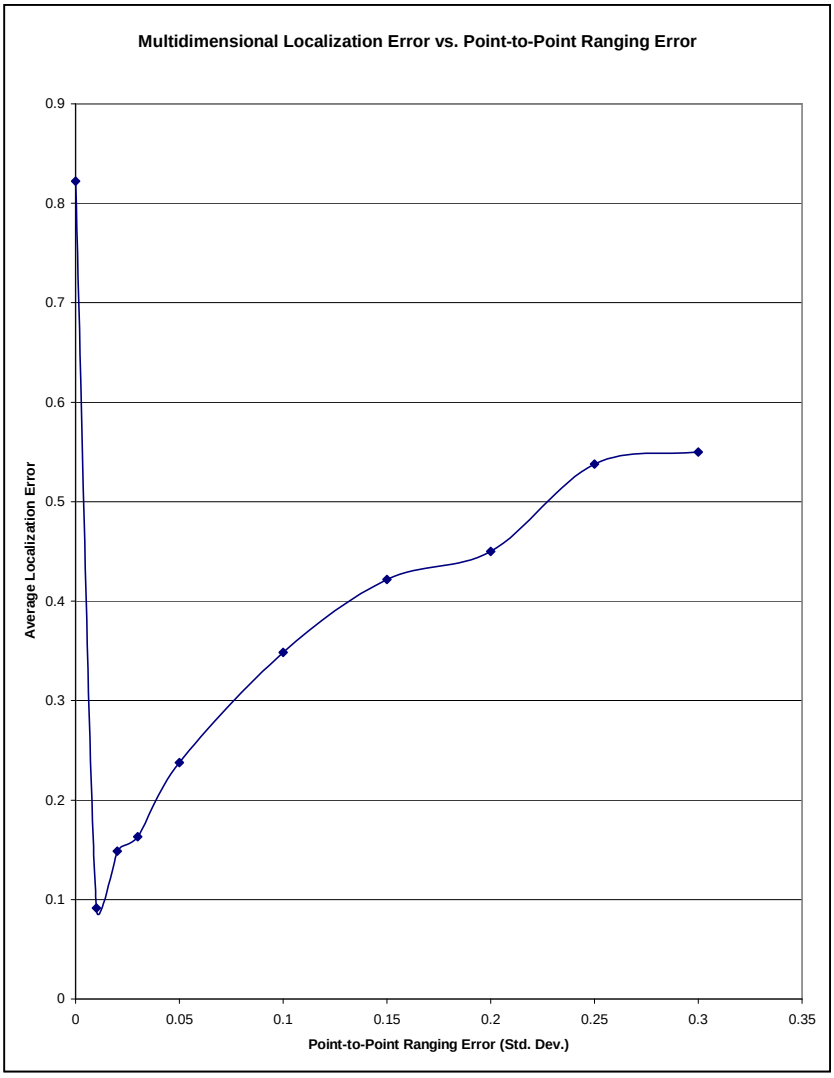

Figure 3.4: Multidimensional WSN Localization Error versus Point-to-Point Ranging Error

It turns out that the one dimensional localization gives adequate results assuming reasonable point-to-point localization data (i.e. an average error of 0.03m achievable using ActiveBats [5]). There are certainly gains to be made from further research in the general multidimensional sensor localization case, but the fundamental process of data fusion remains the same.

3.4 Implementation of Localization in TinyOS and I2K

Sections 3.3.1 and 3.3.1 introduce the high-level algorithms which we implemented in TinyOS. This section presents details of the implementation and presents formulas to convert timer "ticks" to distance (in meters) and to calculate the uncertainty in the measurements.

Our implementation of ranging based on acoustic time-of-flight in TinyOS is a modification of the code from the Calamari project [48]. The released Calamari code was designed for ultrasonic sensors using the MICA platform, not the MICA2. We modified the code to use the piezo-electric buzzer, microphone, and tone-detection circuit present on the MTS300 (basicsb) sensor board. We kept the timer granularity at the default 1/1024s (approximately 1 millisecond). This granularity would imply point-to-point distance granularity of $\frac{1}{1024}\text{s} \cdot 346.65\frac{\text{m}}{\text{s}} = 0.34\text{m}$: too high for indoor data fusion experiments. The MICA2 timer is capable of running at close to 30000 Hz which corresponds to a point-to-point distance granularity of 0.01m. Unfortunately, changing the clock frequency higher than 1024 Hz is problematic for TinyOS without major source code modifications.

Without modifying the clock frequency, the relationship between "ticks" and distance (in meters) is just:

$$\text{distance} = \frac{346.65 \text{ ticks}}{1024} \tag{3.7}$$

The ranging code returns a large number of results for each ordered mote pair. Since our setup was static, the reported data has been averaged resulting in the "Averaged Ticks" column in the following table. We compute the "Absolute Error" as follows:

$$\text{Absolute Error} = |\text{Ranging Distance} - \text{Actual Distance}| \tag{3.8}$$

When using this ranging procedure in a real room with real sensors, our ranging results are summarized in Table 3.2 and graphed in Figure 3.5.

Table 3.2: WSN Ranging Results in an Office Environment

Mote Pair	**Averaged Ticks**	**Ranging Distance (m)**	**Actual Distance (m)**	**Absolute Error (m)**
0 to 5	179.8	60.8668652	3.03	57.83686523
1 to 0	27	9.14018555	2.63	6.510185547
1 to 3	11.25	3.80841064	2.07	1.738410645
1 to 5	7.26315789	2.45876336	0.725	1.733763363
2 to 1	26	8.80166016	1.45	7.351660156
2 to 4	21.3333333	7.221875	2.07	5.151875
2 to 5	13.8974359	4.70463492	0.725	3.979634916
2 to 6	107	36.2222168	1.27	34.9522168
4 to 0	208	70.4132813	1.55	68.86328125
4 to 1	7.25	2.45430908	1.48	0.974309082
4 to 3	9	3.04672852	1.45	1.596728516
4 to 5	14.7027027	4.97723818	1.65	3.327238176
4 to 6	45.3333333	15.3464844	1.95	13.39648438
5 to 0	15.75	5.3317749	3.03	2.301774902
6 to 1	1614	546.37998	1.27	545.1099805
6 to 3	79.5	26.9127686	1.95	24.96276855
6 to 5	7.63157895	2.58348324	1.04	1.543483244
			Avg Abs Error (m)	45.96062707

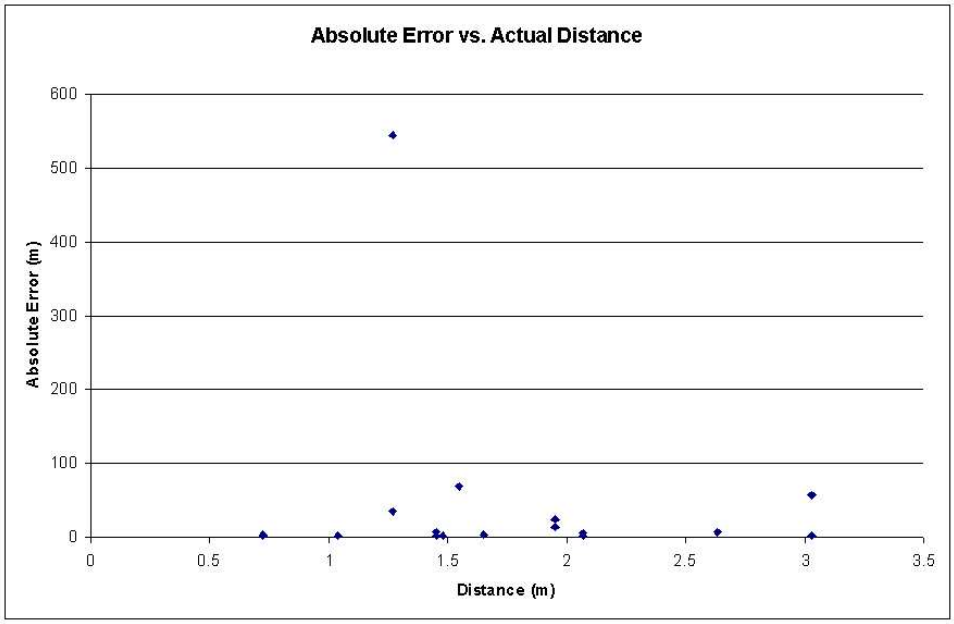

Figure 3.5: WSN Ranging Results versus Distance in an Office Environment

We were not pleased with this result and conducted some further research. We searched for explanations related to (1) inaccuracy of measured

clock ticks and (2) range dependency and variation with respect to oriented mote-to-mote communication. First, judging by the large range in the measured clock ticks (e.g. from around 10 to over 200 ticks) for a fairly small range of distances, it appears that there are some system-level behaviors that we have not accounted for. Specifically, the motes have a single processor which is shared between all scheduled TinyOS tasks. Scheduled tasks regularly utilize the processor and delay the time-sensitive ranging task. Scheduling delays, in turn, increase the number of reported ticks. This hypothesis is consistent with some of the large jumps we see in the data and is also consistent with the "positive" error we always see. This variation in ticks directly translates into variation in distance (by quite a lot as sound travels quickly through air) which in turn translates into ranging errors.

Second, we noticed the irregularities in the data with respect to oriented mote-to-mote communication. As an extreme case, consider the difference between the computed distance from mote 0 to 5 and mote 5 to 0 (the same physical distance). Further investigation revealed that the speakers and microphones that we used for this experiment were not very consistent. Some speakers produced very different tones that some microphone and tone-detection circuitry could not detect. Some microphones were not as sensitive to tones as others. Noticeable air conditioning noise was a likely source of data complication in addition to hardware inconsistencies.

We had the good fortune of borrowing a collection of MICA2 motes from the University of Illinois Computer Science department that had been modified to conduct ranging experiments. Specifically, the speakers on the sensor boards had been bypassed with another more powerful speaker shown in Figure 3.6.

Figure 3.6: MICA2 Mote with Loud Speaker

We ran a more methodical ranging experiment where we incremented

Table 3.3: WSN Loud Speaker Ranging Results versus Distance in an Office Environment

Mote Pair	**Ranging Distance (m)**	**Actual Distance (m)**	**Absolute Error (m)**
0 to 1	1.354101563	0	1.354101563
1 to 0	1.861889648	0	1.861889648
0 to 1	5.501037598	0.5	5.001037598
1 to 0	2.285046387	0.5	1.785046387
0 to 1	2.482519531	1	1.482519531
1 to 0	4.739355469	1	3.739355469
0 to 1	3.949462891	1.5	2.449462891
1 to 0	4.513671875	1.5	3.013671875
0 to 1	4.062304688	2	2.062304688
1 to 0	7.109033203	2	5.109033203
0 to 1	7.334716797	2.5	4.834716797
1 to 0	7.532189941	2.5	5.032189941
0 to 1	6.996191406	3	3.996191406
1 to 0	6.601245117	3	3.601245117
0 to 1	8.350292969	3.5	4.850292969
1 to 0	8.463134766	3.5	4.963134766
0 to 1	10.24039307	4	6.240393066
0 to 1	7.786083984	4.5	3.286083984
1 to 0	8.124609375	4.5	3.624609375
		Avg Abs Error (m)	3.594

the distance between two motes by 0.5 meters and collected data as before. Our results are summarized in Table 3.3 and graphed in Figure 3.7.

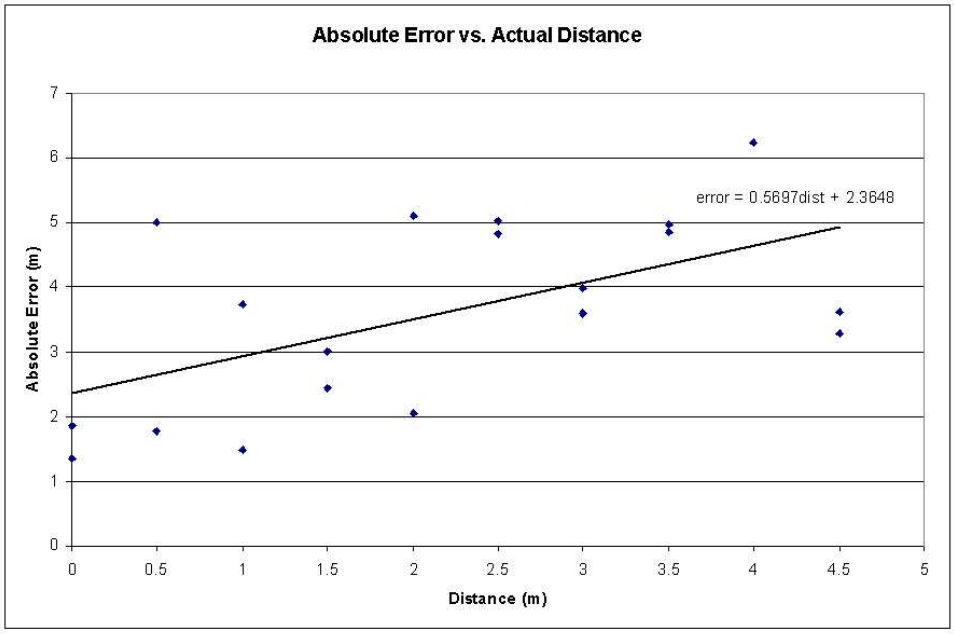

Figure 3.7: WSN Loud Speaker Ranging Results versus Distance in an Office Environment

3.5 Point-to-Point Ranging Calibration

Looking at Figure 3.7, it is obvious that there are ranging errors. Interestingly, we have a significant error at a distance of 0 m which then increases as a function of distance. This sort of error suggests a systematic error in our procedure which can be modeled with a linear function:

$$\text{error} = m * \text{distance} + b \tag{3.9}$$

From our experimental data shown in Table 3.3, we found $m = 0.5697$ and $b = 2.3648$. Removing this trend from the localization data, we get a new "tick" to distance equation:

$$\text{distance} = \frac{\frac{346.65 \text{ ticks}}{1024} - b}{m + 1} \tag{3.10}$$

The resulting [calibrated] graph of ranging error versus distance (Figure 3.8) improves dramatically as well. The new average absolute error is 0.66m. Note that each pair of sensors has its own calibration curve that needs to be derived before gathering ranging data.

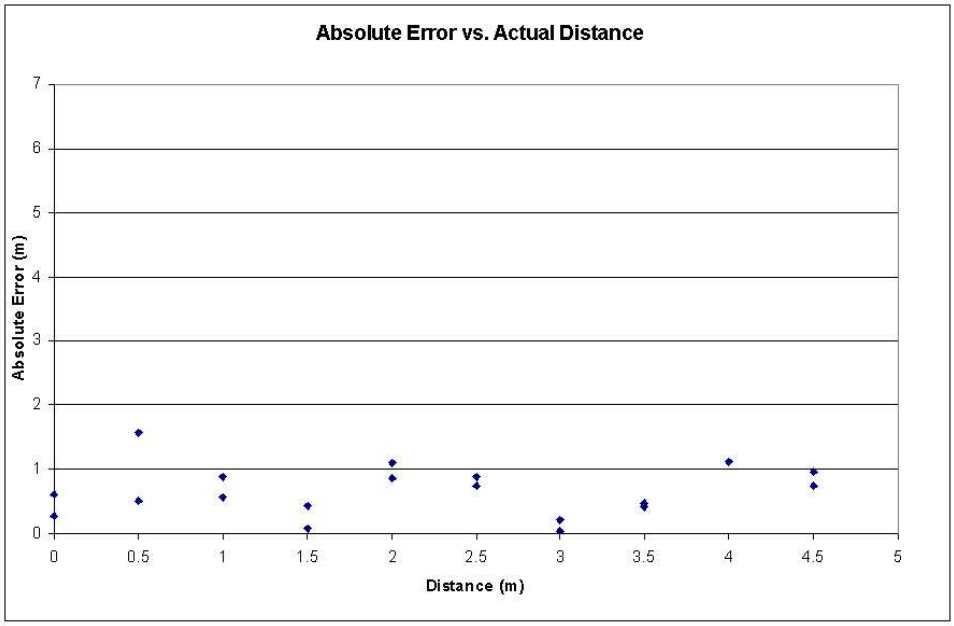

Figure 3.8: WSN Loud Speaker Ranging Results versus Distance in an Office Environment with Calibration

The above results are much better than our first ranging results. We believe that much of the error reduction is due to the improved speakers as they were qualitatively much louder than the background noise. Average point-to-point errors of 0.66m after calibration are still not good enough for our applications, however. We conclude that further research in ranging needs to be completed before automatically collected ranging data is useful for localization. We assume that such a method exists and can produce accurate localization data for the remainder of the thesis. An average point-to-point error of 0.3m, theoretically achievable using the techniques compared in Table 3.1, is used for the remainder of this thesis. This error is synthetically generated by adding additive white Gaussian noise (AWGN), $N(0, 0.3)$, to ground truth measurements of sensor ranges. Figure 3.9 shows how localization using actual ranging and synthetic ranging compare.

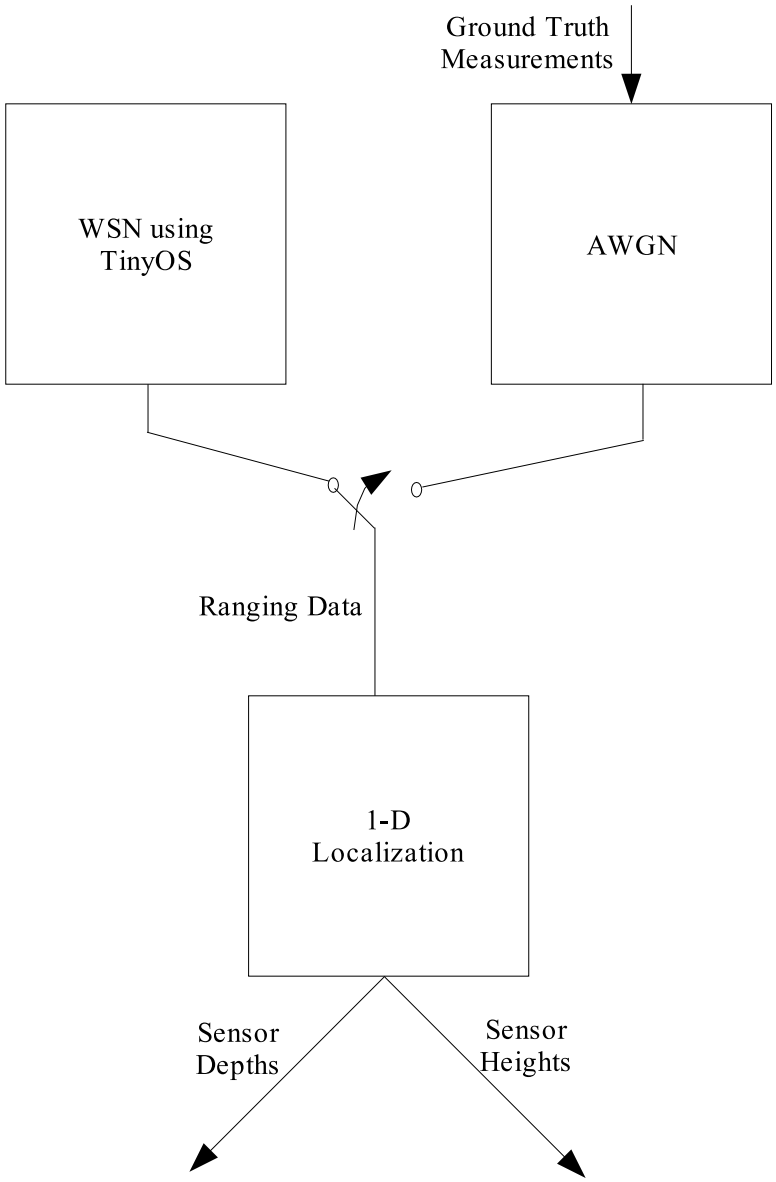

Figure 3.9: Schematic of WSN Localization using Actual and Synthetic Ranging. AWGN represents the addition of additive white Gaussian noise

3.6 Empirical Localization Error

The last unanswered question is how the sensor network localization error depends on the point-to-point ranging error. This relationship is akin to the relationship between distance and stereopsis error presented in section 2.8 as it also impacts our fusion result.

Unlike the stereopsis distance error, the localization distance error is in-

Table 3.4: One-Dimensional WSN Localization Error versus Point-to-Point Ranging Error

Ranging Error σ	**Average Localization Error (m)**
0	4.13E-004
0.01	0.22734939
0.02	0.379998377
0.03	0.436389979
0.05	0.487409176
0.1	0.560905693
0.15	0.622361083
0.2	0.665518586
0.25	0.702981488
0.3	0.731265889
0.35	0.754249712
0.4	0.772064259
0.45	0.78817713
0.5	0.798318631

dependent of distance within a reasonable range. This range is primarily determined by the range of acoustic reception, but is also dependent on the environment and the network topology. For example, a multi-hop WSN will increase the time between radio transmission and reception with each additional routing hop, thereby increasing error. For this thesis, we restrict our WSN to be a single-hop network in a room with good acoustical properties, primarily no echos.

The relationship between point-to-point ranging error and localization error proves non-trivial to analytically derive, but a very good fit can be made empirically. A set of example data is given in Table 3.4. Nine motes were used in the generation of this data. The ranging distances were synthetically generated by perturbing the ground truth distance by instances of $N(0, \sigma)$. This ranging data was localized using the procedure in section 3.3.1. The average localization error was computed by running the experiment 1000 times and taking the mean of the absolute localization error given by Equation 3.8.

A plot of the data in Table 3.4 is given in Figure 3.10. This data is fit extremely well ($R^2 = 0.99$, where R is the correlation coefficient) by the Equation 3.11.

$$\text{error}_{\text{localization}} = 0.138 \ln(\text{error}_{\text{ranging}}) + 0.896 \tag{3.11}$$

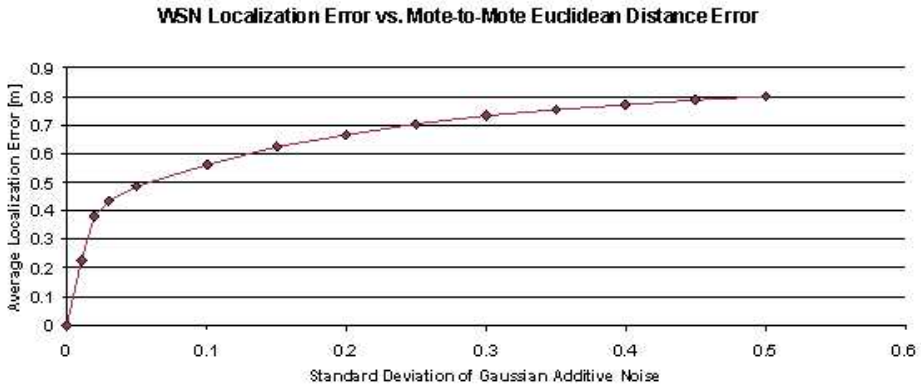

Figure 3.10: One-Dimensional WSN Localization Error versus Point-to-Point Ranging Error

Chapter 4

Fusion of Stereopsis and Localization

4.1 Introduction to Stereopsis and Localization Fusion

This chapter seeks to bring the results of chapter 2 and chapter 3 together and solve the problem posed in section 1.1. We assume that the scene, relative to the camera, remains static throughout the data collection and fusion process.

4.2 Stereopsis and Localization Fusion Error

We seek an error metric to quantify our results. In the computational stereopsis sensing mode, our result is an image called a *depth map*. A depth map is simply an image of a scene where each pixel value represents the depth of the corresponding scene point from the camera/observer. In the WSN localization sensing mode, the result is simply a set of sensor coordinates relative to an anchor mote taken to be at the same location as the camera/observer in the computational stereopsis case. Assuming we know the ground truth locations of N motes represented both as a depth map and as localization data, we develop the following error metric, e_{scene}:

$$e_{depthmap} = \sum_{i=1}^{N} \left\| \text{depthmap}_{computed} - \text{depthmap}_{ground\,truth} \right\|_2 \tag{4.1}$$

$$e_{localization} = \sum_{i=1}^{N} \|\text{localization}_{computed} - \text{localization}_{ground\,truth}\|_2 \qquad (4.2)$$

$$e_{scene} = e_{depthmap} + e_{localization} \qquad (4.3)$$

In general, we are interested in minimizing e_{scene} for any choice of N.

4.3 Related Work to Stereopsis and Localization Fusion

To our knowledge, there is no other work that addresses the specific problem of depth map and sensor localization fusion. The problem of three dimensional registration in the context of multiple sensors, the first step towards multi-sensor data fusion, has been investigated in the image processing community [9], [4]. The larger problem of improving knowledge about the depth of the scene is simplified multisensor data fusion [21]. The process is simplified due to empirical and theoretical models detailing the accuracy of our sensors, thereby eliminating the need for the full theoretical framework.

4.4 Registration of Depth Map and Localization

This section addresses the problem of finding a map between sensor identifiers and real-world sensor locations, a crucial step in practical systems. The only approach in use with sensor networks is to specify this mapping manually, as in [27].

A better approach would be to avoid this manual mapping: for large sensor networks, or even unusual deployment scenarios, manual mapping may not be an option. This mapping can be automatically found with a large number of sensors and the assumption that the sensors are "on top of," rather than embedded in, the depth map as it appears from the vantage point of the stereo cameras.

We assume that we have a calibrated depth map, that we know the relationship between one mote (conventionally the mote with identifier 1) and the camera position, and that the sensor network localization has completed successfully. The mapping between the sensor identifiers and the real-world sensor locations is just a three dimensional image registration problem. We seek the three angles that will rotate the set of localized sensor points so

that they the minimize the cost function:

$$\sum_{i \in \text{sensors}} (R\mathbf{P}_{i_z} - \text{DepthMap}(R\mathbf{P}_{i_x}, R\mathbf{P}_{i_y}))^2 \tag{4.4}$$

Minimizing Equation 4.4 is just a non-linear least squares optimization problem. Although the solutions presented in [9], and [4] are more elegant, this problem can also be solved using standard numerical techniques such as the downhill simplex search [36]. The latter was easier to implement and is used in [2]. After registration, the mapping between the sensor identifiers and the real-world coordinates (given by the location of the sensor on the depth-map) is obvious.

To simulate this process using [2], we generated a series of "depth maps" consisting of many randomly placed and valued rectangles as in Figure 4.1. We then simulated this three dimensional registration by taking a number of random points on the depth map as sensors. The sensor points were, as a group, randomly rotated in all possible directions. We ran our optimization routine to minimize Equation 4.4 and evaluated the success of our algorithm by evaluating Equation 4.4 at algorithm termination and dividing by the number of sensors: a perfect result is 0. Our simulated results for a varying number of points taken from the depth map shown in Figure 4.1 are given in Figure 4.2. Note that the error figures represent one run of the algorithm and have not been averaged over multiple trials.

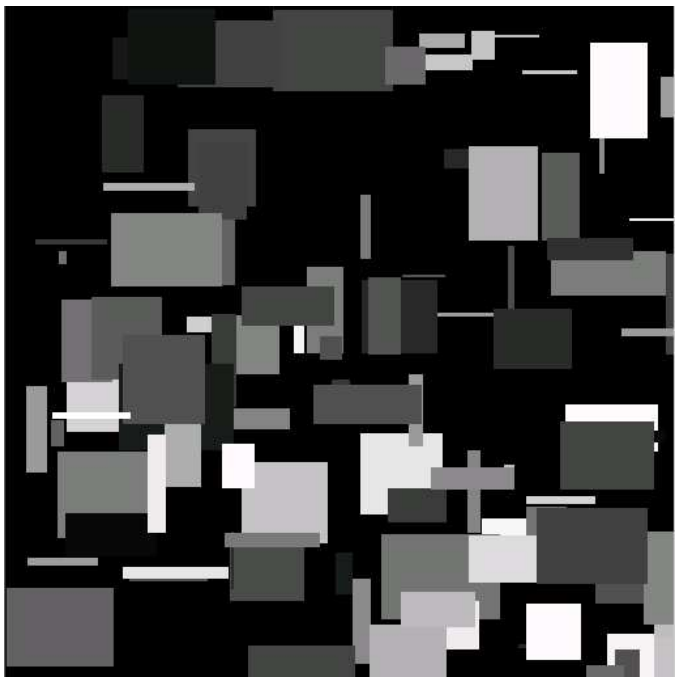

Figure 4.1: Randomly Generated Test Image for Fusion Registration

As shown in Figure 4.2, this optimization does not perform well with few sensors. Empirical results have been omitted since our eight sensors are much too few to say anything. In the remainder of this thesis, the depth map and localization registration is performed manually.

4.5 Implementation of Stereopsis and Localization Fusion in I2K

We needed to make some assumptions during the implementation of the fusion algorithm in Image to Knowledge:

1. Point-to-point sensor ranging is accurate up to an additive Gaussian noise with standard deviation 0.03m, corresponding to the state-of-the-art acoustic ToF ranging results [5].

2. A mapping between sensor ids and sensor locations is given. We can not utilize the techniques in section 4.4 due to lack of sensors.

The implementation of fusion in Image to Knowledge follows directly. The Image to Knowledge Stereopsis and Localization Fusion Tool, shown in Figure 4.3, prompts the user for the necessary parameters (e.g. the stereo matching uncertainty and the point-to-point ranging uncertainty).

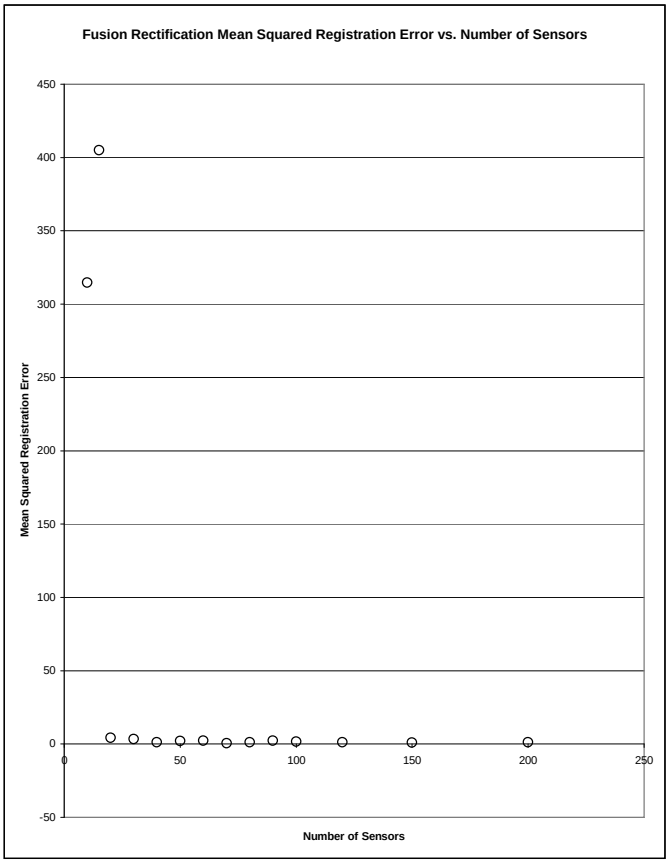

Figure 4.2: Graph Relating Fusion Registration Mean Squared Error to Number of Sensors

The stereo distance uncertainty and the localization distance uncertainty are calculated as in section 2.8 and section 3.6 respectively. A decision point is then found using Equation 4.5 giving a decision rule as shown in Figure 4.4.

$$\text{depth}_{thresh} = x : \text{error}_{dist}(x) = \text{error}_{\text{localization}} \tag{4.5}$$

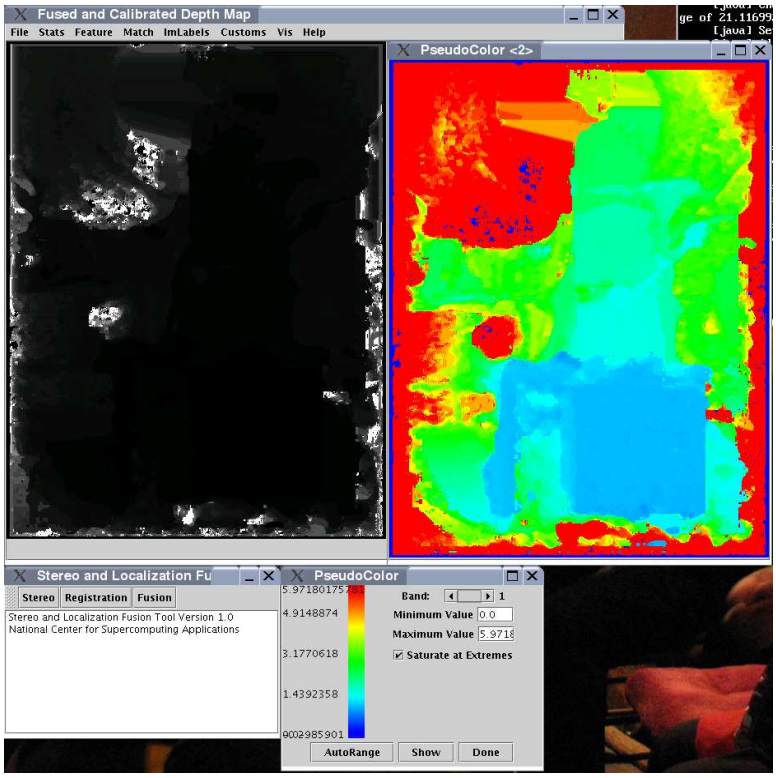

Figure 4.3: Resulting Fusion of Stereopsis and Localization in I2K

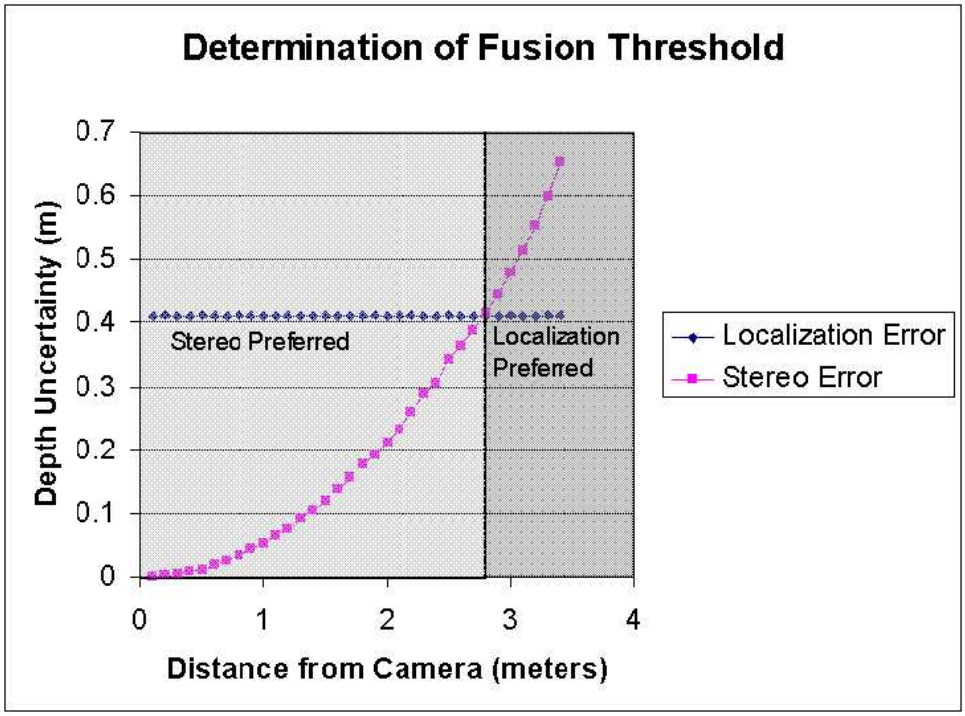

Figure 4.4: Fusion Decision Rule for an Image Matching Error of 4 Pixels and a Point-to-Point Ranging Error of $\sigma = 0.03m$

The fusion algorithm is to simply apply Equation 2.7 to each known point of the scene. If $z_{scene} < \text{depth}_{thresh}$, the stereo error is decided as smallest and the localization depth is set to be the average of the stereo depth in a neighborhood of the known point. Otherwise, the neighborhood of the known point in the depth map is set to the depth value from the localization. Obviously, for best results, a large number of points covering the area of interest is wanted.

Figure 4.5 shows how Figure 4.4 changes with variation of image matching uncertainty. As expected, the depth uncertainty due to stereo increases with additional image matching uncertainty. Since the depth from stereopsis is more accurate than the depth from localization near the camera, additional image matching uncertainty pushes the threshold depth closer to the camera.

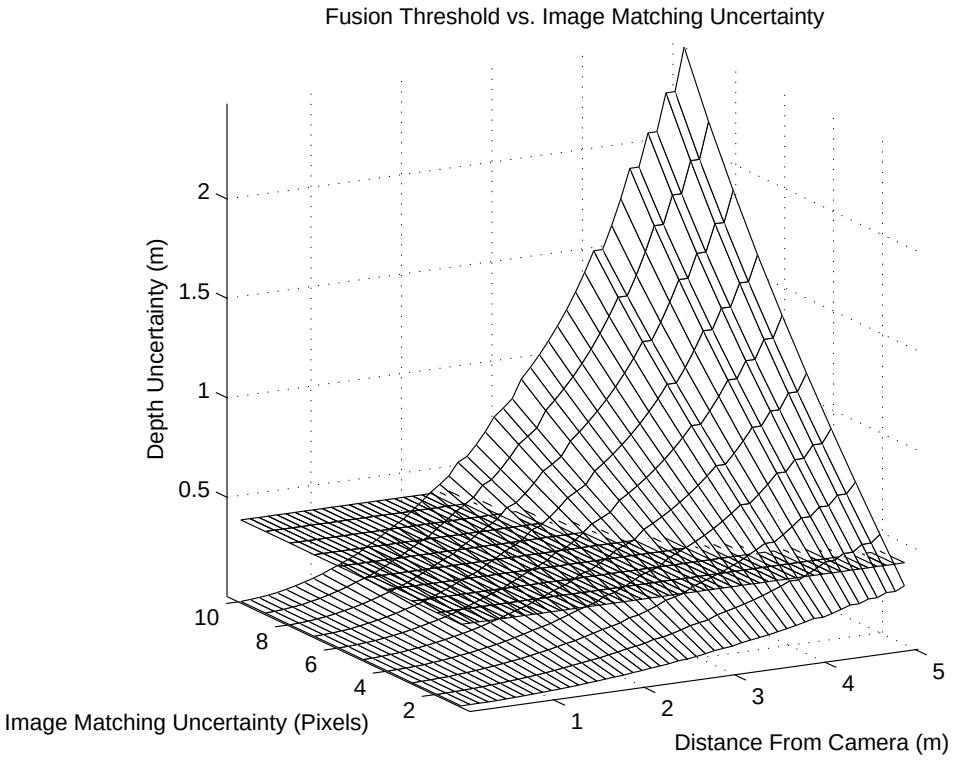

Figure 4.5: Fusion Threshold vs. Image Matching Uncertainty with a Point-to-Point Ranging Error of $\sigma = 0.03m$

Figure 4.6 shows how Figure 4.4 changes with variation of point-to-point ranging errors. As before, the depth uncertainty increases with additional point-to-point image errors. Since the depth from localization tends to be less accurate than the depth from stereopsis near the camera, additional

point-to-point ranging error pushes the threshold depth away from the camera.

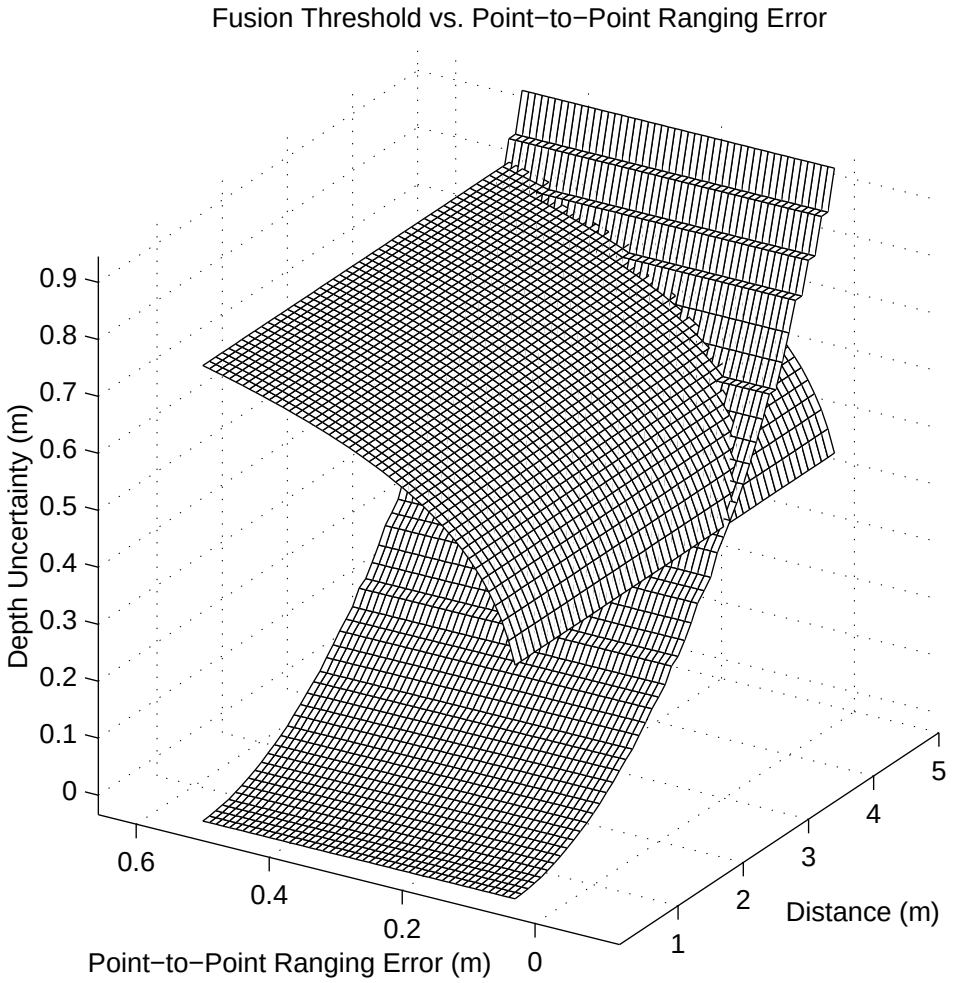

Figure 4.6: Fusion Threshold vs. Point-to-Point Ranging Error with an Image Matching Error of 4 Pixels

Comparing Figure 4.5 and Figure 4.6, it appears that the threshold depth is more sensitive to errors in image matching during stereopsis than point-to-point ranging errors encountered during localization. Figure 4.7 confirms this observation over the scale of error encountered in practice.

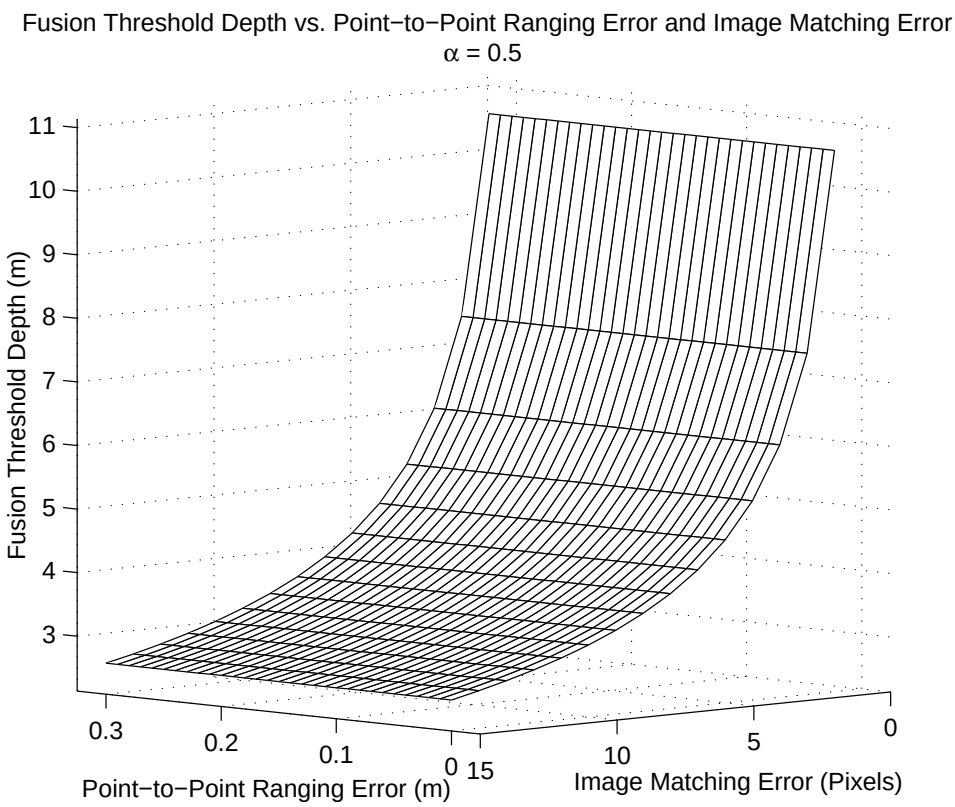

Figure 4.7: Fusion Threshold Distance (Shown as the Intersection of Two Curves in Figure 4.4) as a Function of Image Matching Uncertainty and Point-to-Point Ranging Error

4.6 Results of Stereopsis and Localization Fusion

4.6.1 Evaluation Criteria

We use the error metric presented in section 1.1 to evaluate the fusion result.

4.6.2 Lab Example

As an example of how the fusion process works, we ran an experiment in an indoor laboratory performing all of the steps except for actual ranging due to the issues discussed in section 3.3.1 and section 3.5. A picture of the setup is given in Figure 4.8 and an overhead diagram in Figure 4.9. The scene represents a "staircase" of depth in front of the camera and newspaper is used in order to give some texture to the scene and provide some planar surfaces for easy identification and measurement of points.

Figure 4.8: Picture of Stereopsis and Localization Experiment

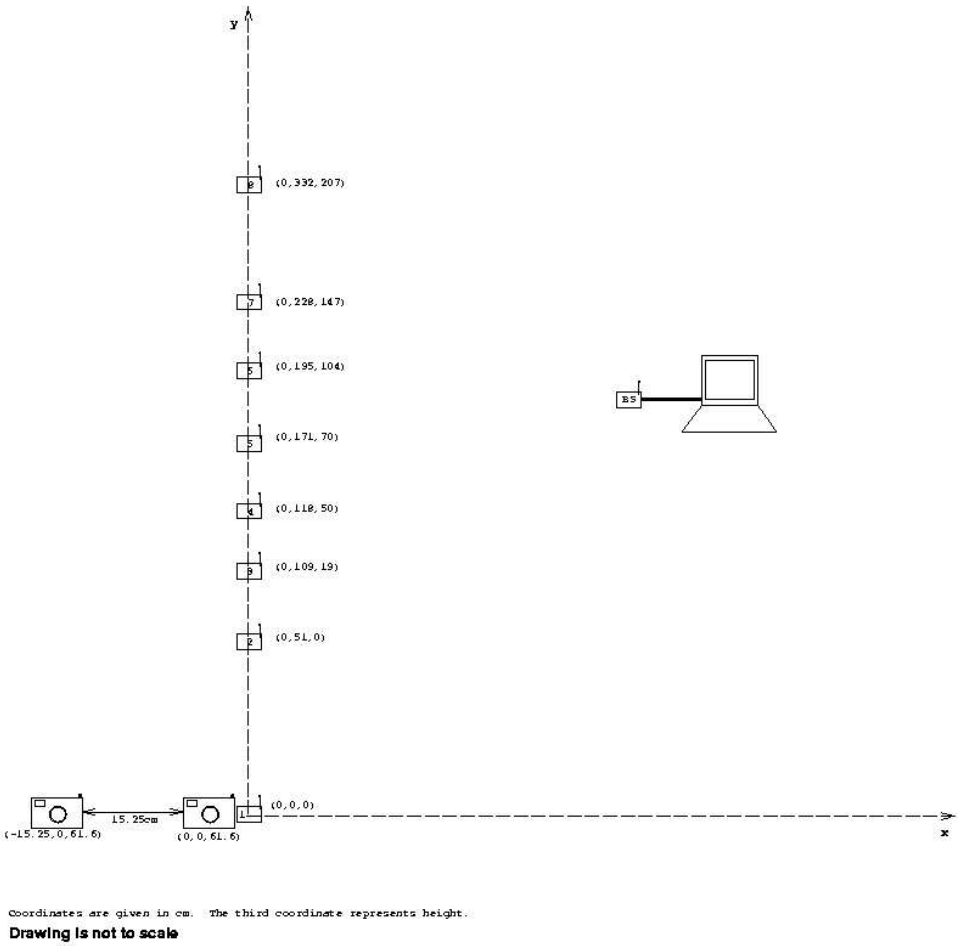

Figure 4.9: Diagram of Stereopsis and Localization Experiment

The Image to Knowledge Stereo Tool was used to obtain and calibrate a depth map which is shown in Figure 4.10.

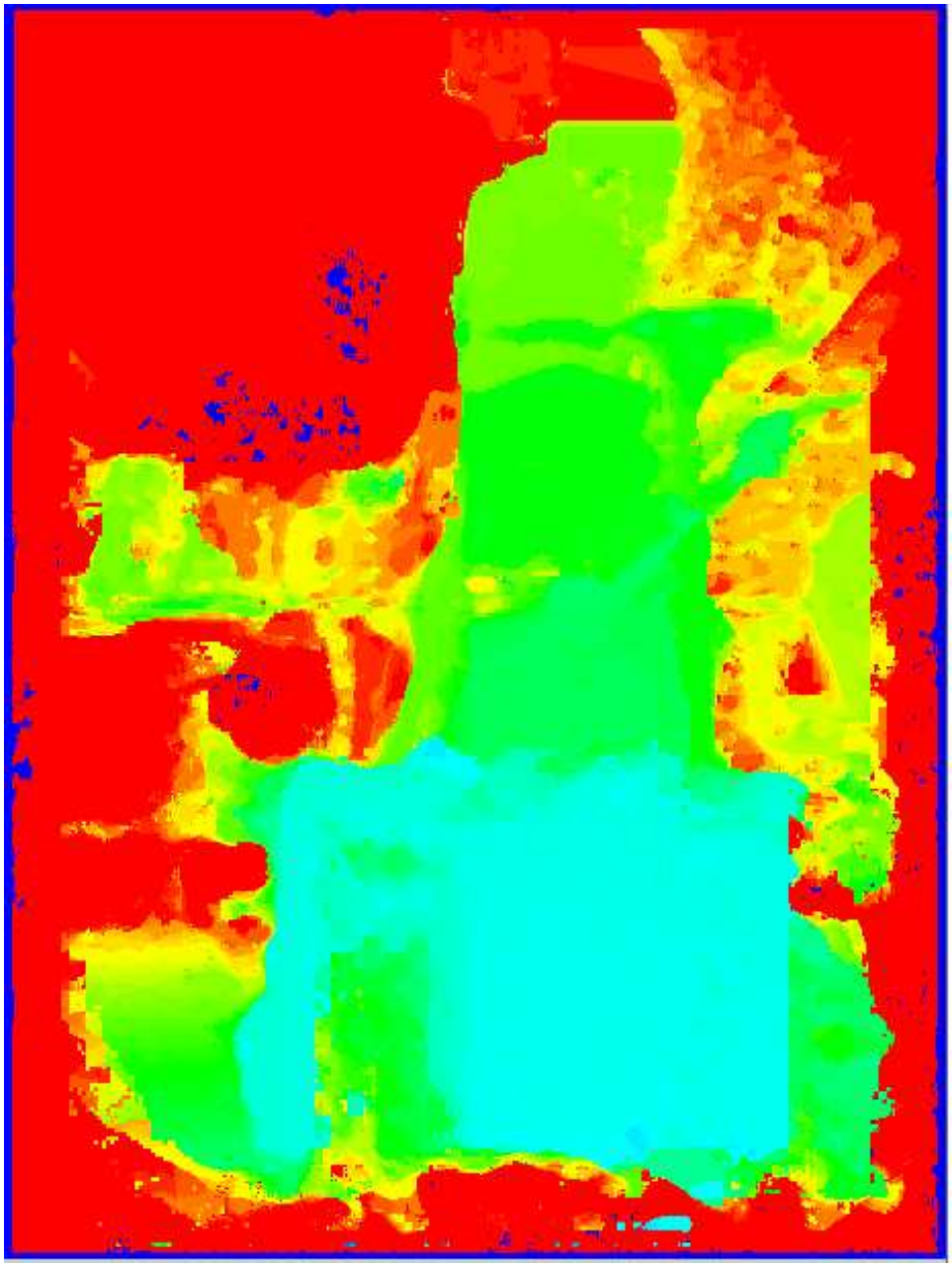

Figure 4.10: Calibrated Stereo Depth Map

Although we did not run a successful ranging step, we did add noise with $\sigma = 0.03$m and utilize the one dimensional localization procedure given in 3.3.1. The results from that procedure are given in Table 4.1.

The manual registration of image points to sensor locations using Image to Knowledge is shown in Figure 4.11.

Table 4.1: Localization Output from Stereopsis and Localization Experiment

Mote ID	**Distance (m)**	**Height (m)**
1	0.0	0.0
2	0.51	0.0
3	1.12	0.07
4	1.30	0.33
5	1.84	0.48
6	2.11	0.80
7	2.64	0.94
8	3.73	1.44

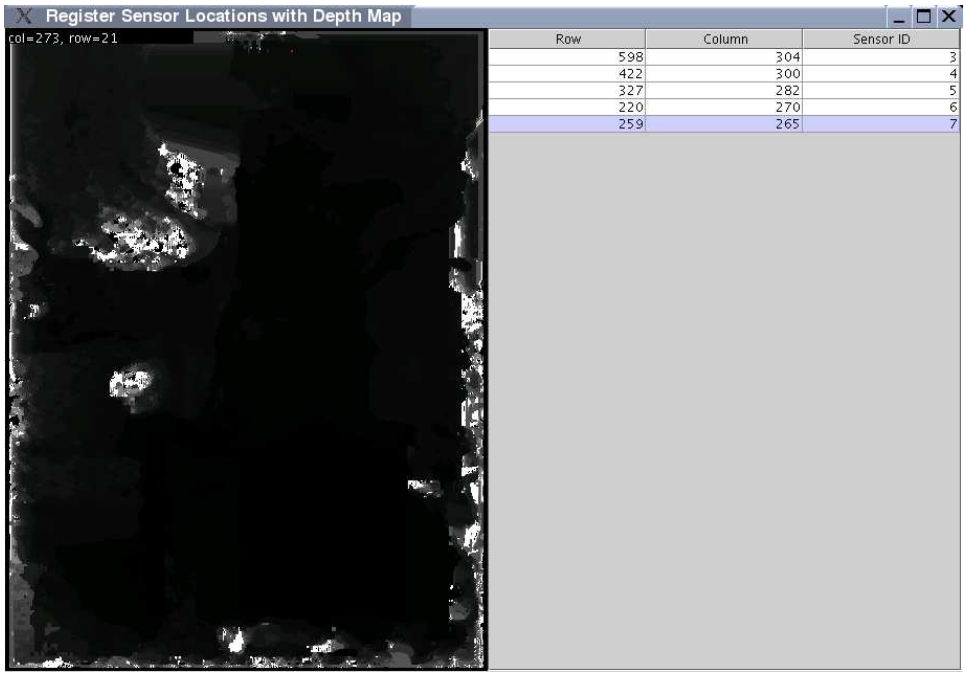

Figure 4.11: Manual Registration of Image Points to Sensors using I2K

The stereopsis and localization fusion is conducted using the Image to Knowledge Stereopsis and Localization Fusion Tool shown in Figure 4.3. Finally, we obtain the fused depth map in Figure 4.12 and the fused localization result given in Table 4.2.

Table 4.2: Fused Localization Output from Stereopsis and Localization Experiment

Mote ID	**Distance (m)**	**Height (m)**
1	0.0	0.0
2	0.51	0.0
3	1.12	0.07
4	1.11	0.33
5	1.67	0.48
6	1.96	0.80
7	1.96	0.94
8	3.73	1.44

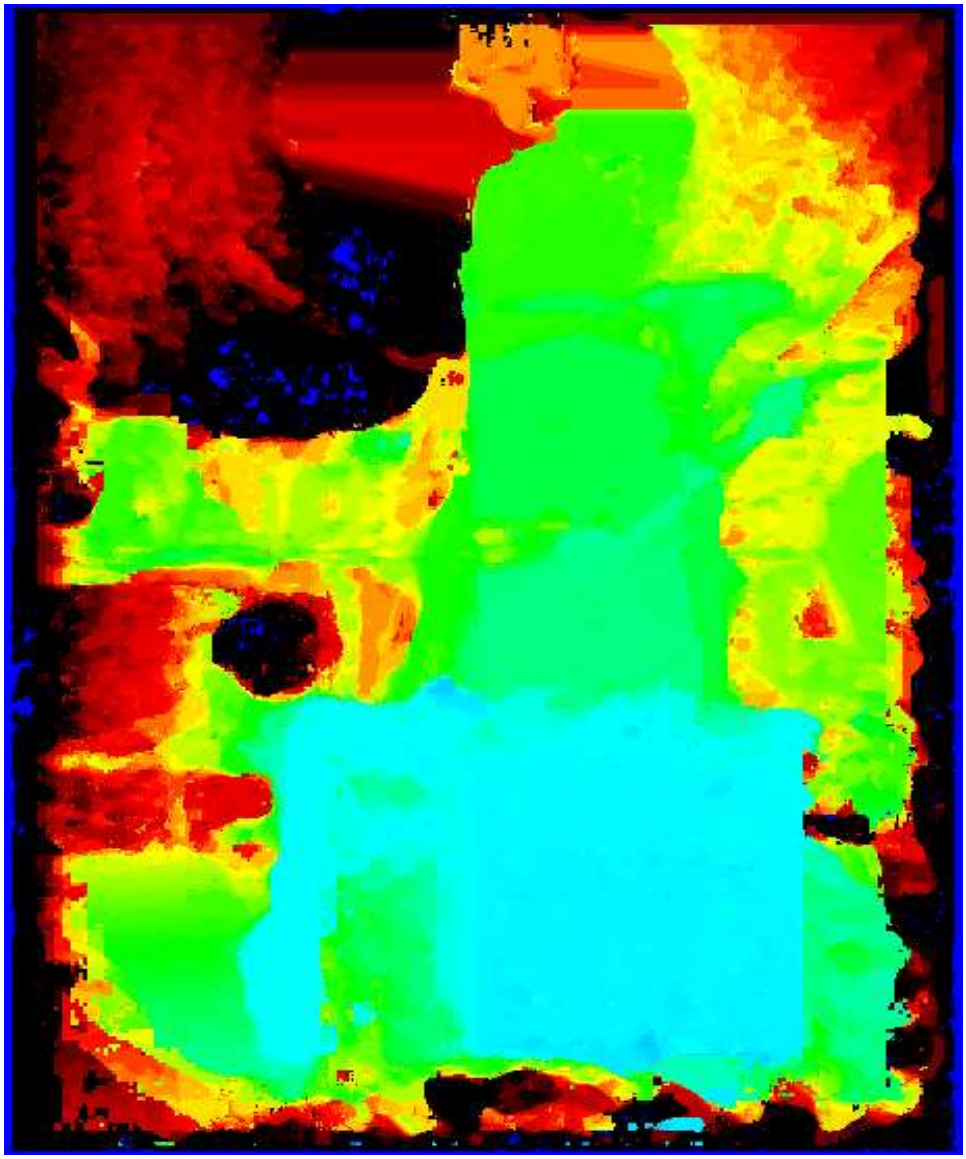

Figure 4.12: Fused and Calibrated Stereo Depth Map

Table 4.3: Fusion Error Results from Stereopsis and Localization Experiment

Data Errors	**Error (m)**	**% Improvement**
Unfused $e_{localization}$ (Equation 4.2)	1.19	27%
Fused $e_{localization}$ (Equation 4.2)	0.87	
Unfused $e_{depthmap}$ (Equation 4.1)	31.06	98%
Fused $e_{depthmap}$ (Equation 4.1)	0.72	
Unfused e_{scene} (Equation 4.3)	32.25	95%
Fused e_{scene} (Equation 4.3)	1.59	

In this case, we see fairly drastic improvement in our error metric. Table 4.3 shows that very real gains can be made using the methodology detailed in this thesis. Amazingly, the improvements are able to correct areas of the stereo depth map with errant stereo results. This happens because errant stereo results (mismatches) tend to be very large distances. The distances exceed the fusion threshold and the more correct localization data overrides the depth map. Even without this effect, we see that improvements to our knowledge of the scene are made.

Chapter 5

Conclusion

5.1 Conclusion

The problem of improving knowledge of scene geometry and sensor locations by fusing data from computational stereopsis with WSN localization data was presented in chapter 1. A detailed discussion of performing computational stereopsis along with limitations of stereopsis was found in chapter 2. Chapter 3 discussed WSN localization using acoustic time-of-flight ranging. The data fusion of data from the techniques presented in chapter 2 and chapter 3 is detailed in chapter 4 along with our experimental results.

The main contribution of this work is the prototype software incorporated into TinyOS [45] and Image to Knowledge [2]. This software forms a fusion system with the following capabilities:

1. Rectification of stereo image pairs
2. Computation of scene depth from stereo image pairs (stereopsis)
3. WSN Localization through acoustic time-of-flight ranging
4. Supervised and unsupervised fusion of stereo depth maps and WSN localization

Details of prototype operation are given in the appendices to this thesis.

5.2 Areas of Future Research

There are still open problems related to this work, however. Perhaps most noticeable is the lack of good acoustic ToF ranging. The other open problem is that of automation. The current steps in the process from start-to-finish that require human intervention or major assumptions are:

1. Picking matching image points for stereo rectification

2. Matching sensor identifiers to image coordinates for stereo calibration

3. Registering localization result with depth map

All three of these problems are classic problems in image processing that do not have easy solutions.

The problem of finding matching image points between images without any additional knowledge (i.e. that the images have been rectified) has been extensively studied in the computer vision community as "point matching." A good review of the point matching problem in the context of stereo systems is given in [46]. The generally accepted procedure is to first extract salient image features (e.g. corners) from both stereo images. These features are often matched with each other using a variant of correlation. Points that can not be matched are often thrown out of the algorithm as outliers. As always, this whole issue can be side-stepped if a calibrated stereo rig, or equivalent, is used to capture the stereo pair.

Matching sensor identifiers to image coordinates is a two-phase problem: first identify where sensors lay in the image, and then determine which sensor belongs to which identifier. The former is a basic pattern recognition problem where the target shape is known, but the location, orientation, and scale are unknown. One method of finding such sensors is "template matching," another well-researched area in the computer vision literature akin to the well-known matched filter [20], [29]. The latter problem can likely be solved by an optimization extending the one used in section 4.4. The main difference from the procedure in section 4.4 is the presence of another parameter: the scale factor of the depth map. Given enough data, these problems should be solvable, although we have not had the time to investigate them.

The problem of registering localization results with a depth map of the same scene was treated in section 4.4. As discussed, this approach is not attractive for small numbers of sensors, but we do not consider this to be a problem as large numbers of sensors are likely to be present in real-world scenarios.

Appendix A

WSN Localization using TinyOS

A.1 Introduction

This appendix covers the procedure for performing acoustic time-of-flight (ToF) ranging with a network of Crossbow MICA2 sensors [14]. The mechanism of acoustic time-of-flight ranging is described in section 3.3.1 with the details of the TinyOS implementation given in section 3.4. As stated in section 3.4, the code is a modification of code released by the Calamari project [48].

The process of programming the sensor network is discussed in Section A.2. The process of performing the localization is given in section A.3. The output from the ranging code is not in an appropriate format for the fusion process described in chapter 4; section A.4 details the steps required for data conversion.

A.2 Programming Sensors to Perform Ranging

To program the sensors, it is assumed that a working installation of TinyOS [45] is present. In the development of this thesis, TinyOS version 1.1.0 was used exclusively. Additionally, the modified `.../contrib/calamari` and `.../tools/java/net/tinyos/acoustic_ranging` code from the NCSA ALG CVS repository should be installed.

One sensor should be programmed as the base station and will not be part of the ranging process. To program the mote, change the current directory to `.../apps/TOSBase` and issue the following command:

```
make mica2 install.0
```

The additional sensor are programmed with incremental identifiers starting with 1. The identifier in the below programming command has been replaced with i. Before programming the motes, change the current directory to `.../contrib/calamari/micaRangingApp`. To program each mote, issue the following command:

```
make mica2 install.i
```

A.3 Performing Ranging using TinyOS

After all the motes have been programmed, connect sensor boards with microphone and speaker to all motes excluding the base station. These equipped motes should be deployed and powered-on. The base station mote should be attached to the programmer/interface board, powered-on, and should be connected to a computer via a serial port (assumed here to be COM1).

The `SerialForwarder` Java program is started as follows:

```
java net.tinyos.sf.SerialForwarder -comm serial@COM1:mica2
```

Finally, the `acoustic_ranging` Java program is started:

```
java net.tinyos.acoustic_ranging.acoustic_ranging
```

The ranging process will take a long time, especially with a large network of motes. Upon completion, a `moteMap.out` output file will be written in the current working directory.

A.4 Converting TinyOS Ranging Output to the Localization XML Format

The `net.tinyos.acoustic_ranging.acoustic_ranging` Java program writes the ranging results into a file named `moteMap.out` in the current working directory. This file is actually a serialized Java class and is not in a suitable format for use in other applications.

The `net.tinyos.acoustic_ranging.MoteMapToXML` Java program converts the serialized Java class into an XML application. The program usage is as follows:

```
java net.tinyos.acoustic_ranging.MoteMapToXML moteMap.out moteMap.xml
```

The `moteMap.xml` file generated by the above command can be renamed and moved out of the TinyOS file hierarchy: it has no dependencies on TinyOS code. The XML application has the following format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<localization>

   <rangingPair recvId="2" sendId="1">
      <range>0.51</range>
   </rangingPair>

   <rangingPair recvId="3" sendId="1">
      <range>1.1064357188738982</range>
   </rangingPair>

   ...

</localization>
```

Appendix B

Performing Computational Stereopsis using I2K

B.1 Introduction

This appendix describes the operation of the Image to Knowledge Computational Stereopsis Tool. Computational stereopsis is discussed in chapter 2. A description of the algorithm implemented in Image to Knowledge (I2K) is given in section 2.6.

B.2 Starting and Opening Images

This appendix assumes a working installation of I2K [2]. To start the Computational Stereopsis Tool, run I2K and select the "Stereo" option from the "Customs" menu. A new window, shown in Figure B.1, will open with the Computational Stereopsis Tool.

The Computational Stereopsis Tool can load multiple images for image selection and comparison. Clicking the "LoadSet" or "AppendSet" buttons launches a new file selection window titled "Open." In the "Open" window, a user may load multiple images by holding the "Shift" or "Ctrl" key. Using the "LoadSet" button loads multiple images and discards all existing images in a thumbnail viewer. A user can preserve all existing image in the thumbnail viewer and append new images by clicking the "AppendSet" button. Figure B.2 shows an image "Open" window.

After loading images, the thumbnail window will appear for image browsing and selection. Figure B.3 shows the thumbnail viewer. By moving the scroll bar on the bottom of the viewer window, a user can browse the images and check the filenames of each image. The "Select" check box above each image is for selecting particular images for image processing tasks (e.g.

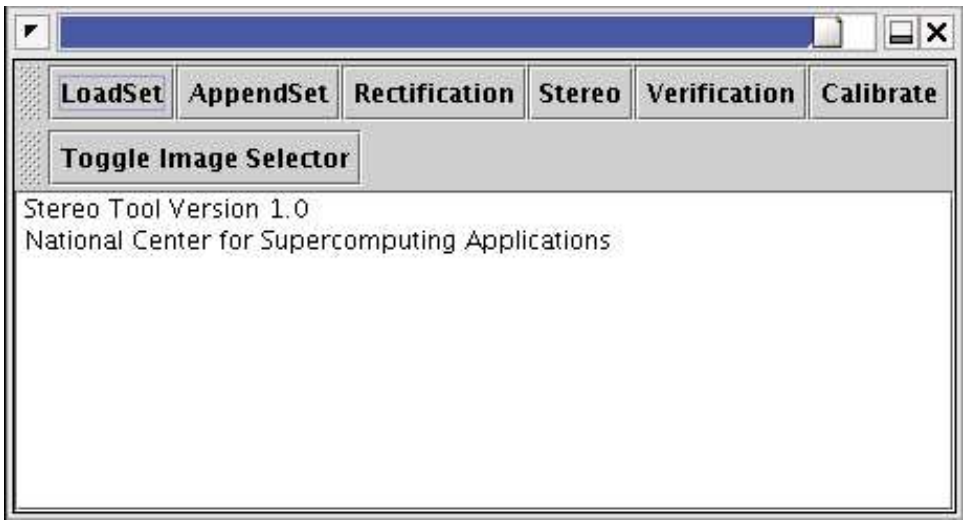

Figure B.1: Main window of I2K Computational Stereopsis Tool

rectification, stereo, and comparison).

B.3 Rectifying a Stereo Pair

By clicking the "Rectification" button in the main window, one can launch the Stereo Rectification Tool shown in Figure B.4. The Stereo Rectification Tool window consists of two panels: "Stereo Rectification" and "Rectification Parameters." The "Stereo Rectification" panel contains buttons for computing, loading and saving rectification parameters, and image transformation using the computed (or loaded) parameters. The "Rectification Parameters" window shows the values of computed or loaded perspective transformation parameters. The "Send to Main" button will load the selected image into the main image window of Image to Knowledge.

In the Rectification Parameters panel, nine values define the perspective transformation model in row-major order. The numbers at the end of the parameter name are the subscripts in the [3x3] transformation matrix.

B.3.1 Computing Coordinate Transformation Parameters

By clicking the "Compute Parameters" button in the Stereo Rectification Tool window, one can launch the "Feature Selector" window for selecting transformation control points. Figure B.5 shows the GUI of the "Feature Selector" window. The "Feature Selector" window displays two images before their rectification, for instance, two images of a scene from slightly differing views. A user should select approximately 15 matching pairs of control

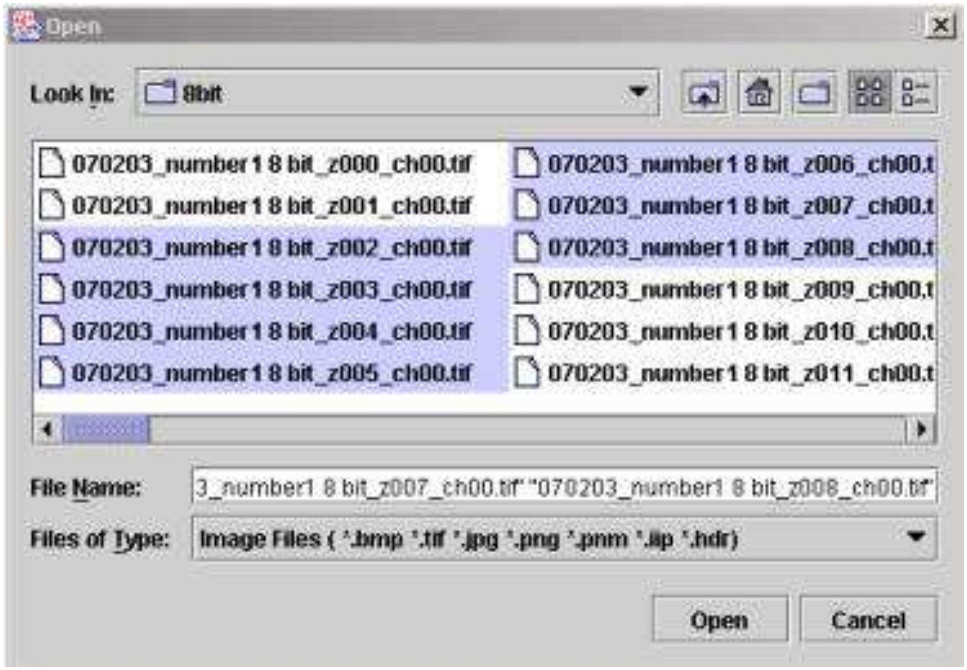

Figure B.2: Image file selection in I2K Computational Stereopsis Tool

Figure B.3: Thumbnail viewer in I2K Computational Stereopsis Tool

points in both images to compute the perspective transformation parameters successfully. Each selected point is displayed as a colored circle with the index of the pair. Clicking "Transform" will use the selected control points to calculate the parameters used for stereo rectification.

Once the perspective transformation parameters are determined, a set of images can be transformed and saved by selecting an image and clicking the "Image Transform & Save" button. A user can specify the suffix of the transformed saved images. Note that the Computational Stereopsis Tool current image set may need to be appended with the newly rectified image.

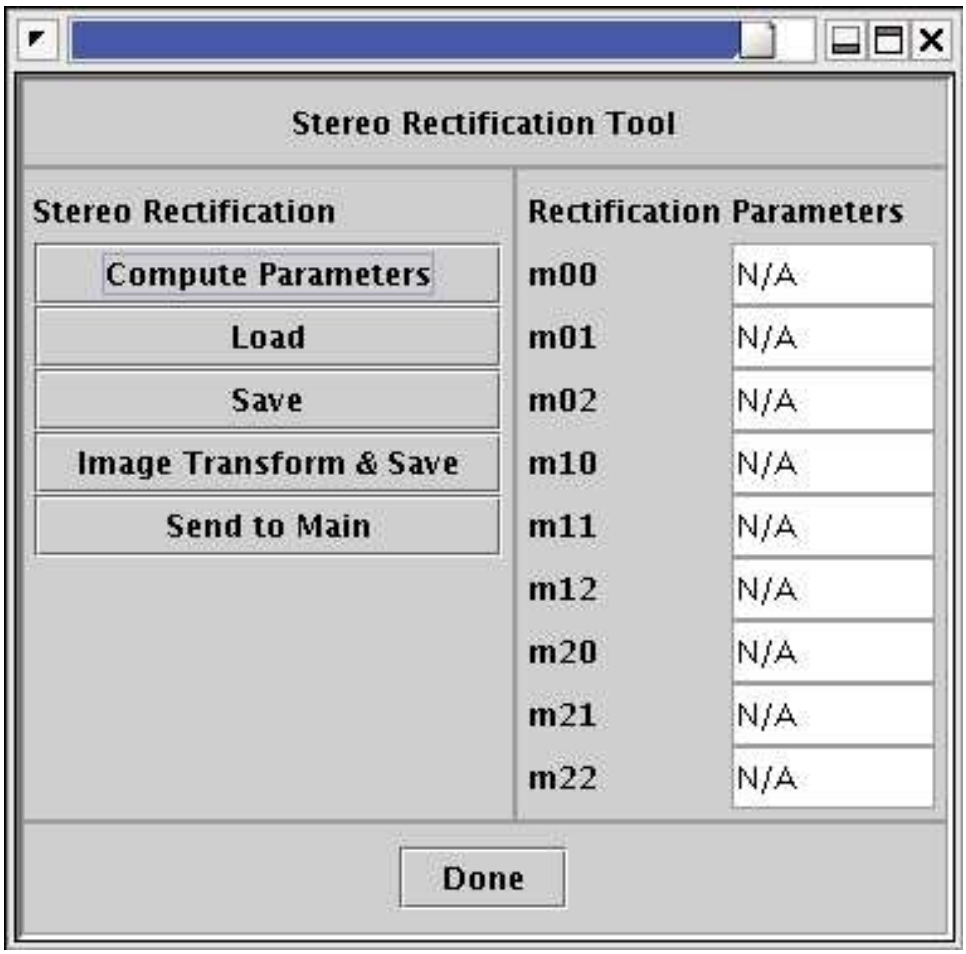

Figure B.4: Stereo Rectification Tool

B.4 Acquiring a Depth Map

To perform computational stereopsis, simply pick two [rectified] images and select the "Stereo" button in the Computational Stereopsis Tool window (Figure B.1). The stereopsis calculation can take a long time. After the computation has completed, the [uncalibrated] depth map will appear in the main I2K window.

B.5 Verifying a Depth Map

By selecting two images in the thumbnail window followed by clicking the "Verification" button in the main window, one can compare two images. Figure B.6 shows the "Image Composition" window comparing a [saved] depth map to the "left" image of a stereo pair.

Figure B.5: Feature Selector Window

B.6 Calibrating a Depth Map

To calibrate a depth map, ensure that the depth map is loaded in the main "Image to Knowledge" window (the default location of a computed depth map). Click the "Calibrate" button in the Computational Stereopsis Tool window to bring up the window in Figure B.7. To calibrate the image, click a pixel corresponding to a point of known scene depth: you will be prompted for the depth in meters. You can repeat this process for multiple pixels in an attempt to improve the calibration. When done, click the "Calibrate" button at the bottom of the window shown in Figure B.7: the depth map will be calibrated in-place.

In many cases, the points of known depth may be difficult to "pick out" from the depth map. Another image corresponding to the "left stereo" image (or the "left stereo" image itself) can be used instead by selecting the image from the "Thumbnails" image selection window before clicking the "Calibrate" button in the ComputationalStereopsis Tool window. The rest of the calibration procedure is as before.

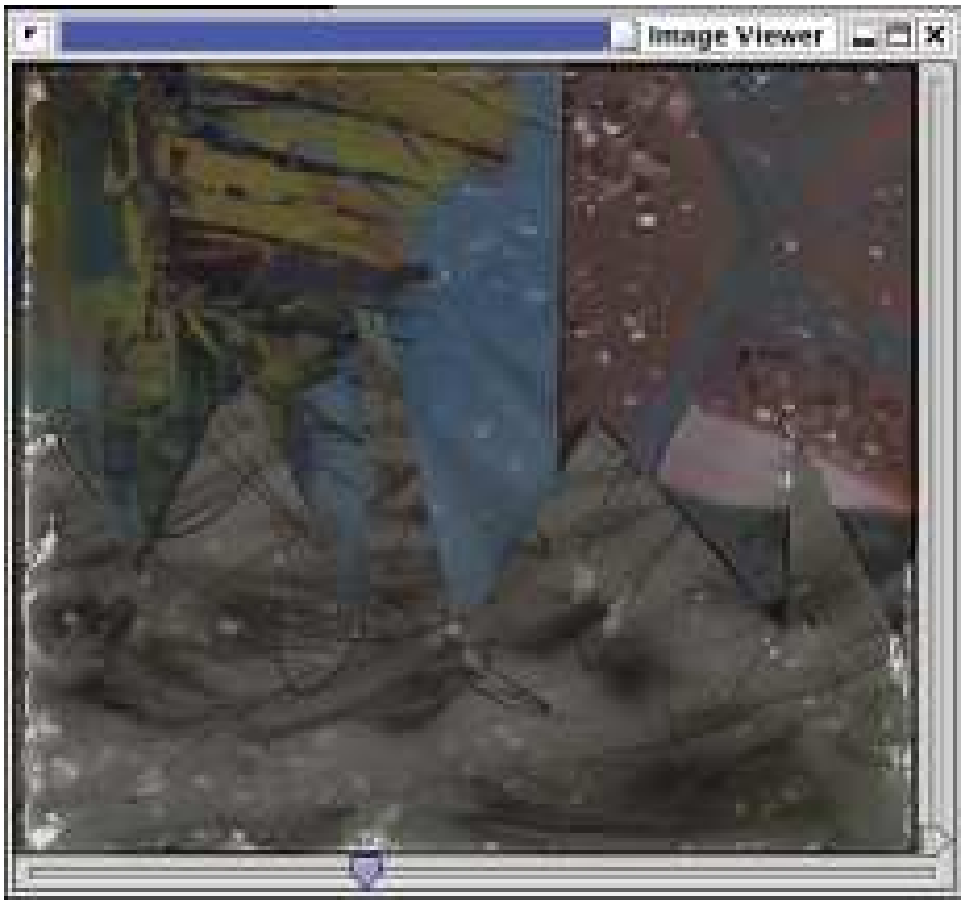

Figure B.6: Image Composition/Verification Window

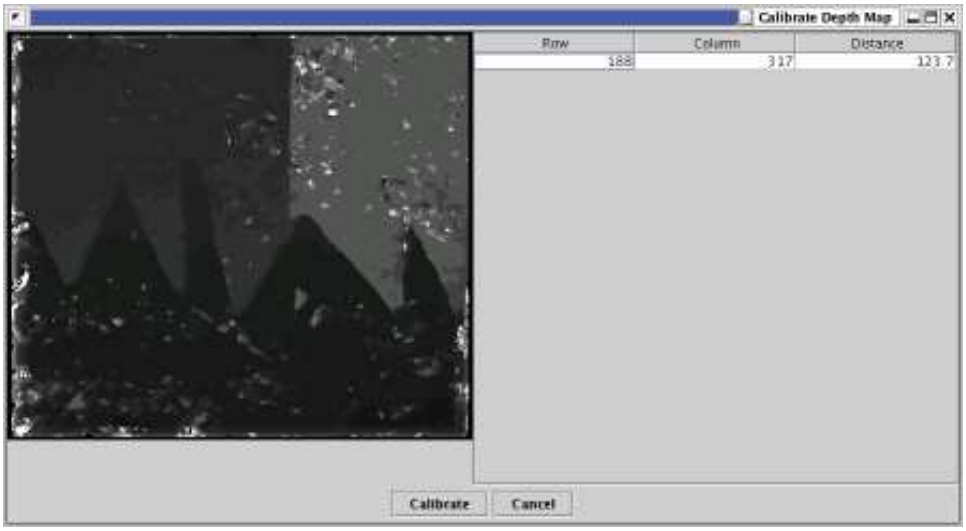

Figure B.7: Depth Map Calibration

Appendix C

Fusing Stereo and Localization Using I2K

C.1 Introduction

This appendix describes the operation of the Image to Knowledge Stereopsis and Localization Fusion Tool. The fusion process is described in chapter 4. The Image to Knowledge implementation is detailed in section 4.5.

The Stereopsis and Localization Fusion Tool provides a tool for fusing a calibrated depth map with sensor network ranging data. The output of the tool is a calibrated and fused depth map that conveys a more accurate picture of the imaged scene along with corrected sensor localization data. This tool provides an interface to:

1. Use the Computational Stereopsis Tool to extract a depth map from a stereo pair.

2. Manually register sensor locations with depth map points.

3. Perform a data fusion process with user-specified stereo and localization uncertainties in order to extract a more accurate depth map and localization result. Note: at this time, the sensor network localization is limited to a two-dimensional case with at least first-order sensor "intradistances" known.

C.2 Prerequisites

To successfully run the Stereopsis and Localization Fusion Tool, you must have an uncalibrated and calibrated depth map of the scene in addition to an XML application describing ranging data. Appendix B and appendix A describe how both are obtained respectively.

The calibrated depth map should be loaded as the main I2K image prior to starting the Stereopsis and Localization Fusion Tool. This can be done using the "Open" command from the main I2K "File" menu.

C.3 Starting

The Stereopsis and Localization Fusion Tool can be started by using the "Depth Map and Localization Fusion" option of the I2K "Customs" menu. When you select this option, a new window will open on the screen (Figure C.1).

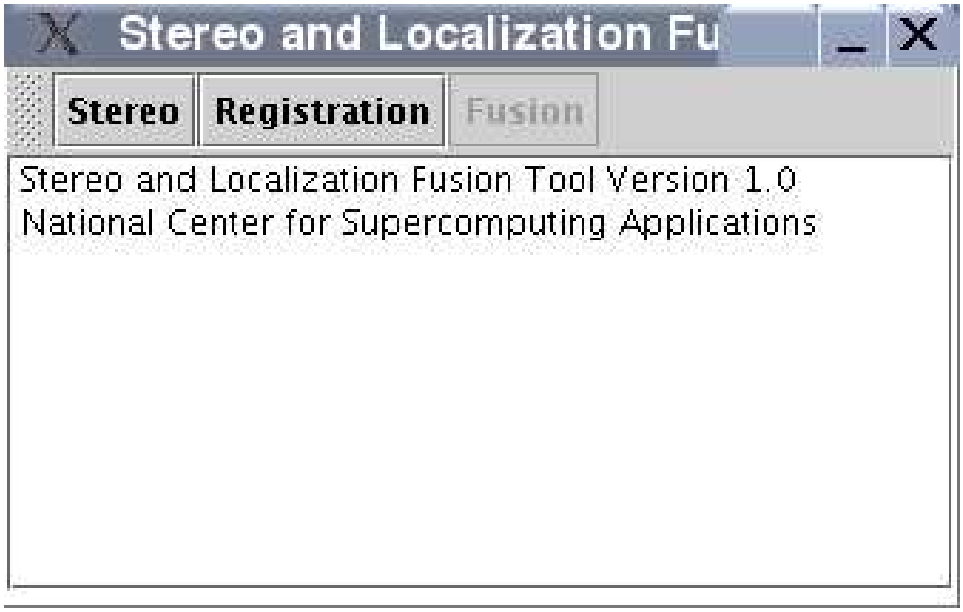

Figure C.1: Main window of I2K Stereopsis and Localization Fusion Tool

C.4 Manual Registration of Sensors with a Depth Map

To fuse the two sets of depth information, the depth map and the localization must be registered/aligned with each other. Although there may be automated techniques for handling this problem in the future, the current method is to specify this mapping manually. To start specifying this mapping, use the "Registration" button from the Stereopsis and Localization Fusion window. A window like Figure C.2 should open.

You should specify every sensor that appears in the depth map. To specify the location of a sensor, click in the left frame of the "Register

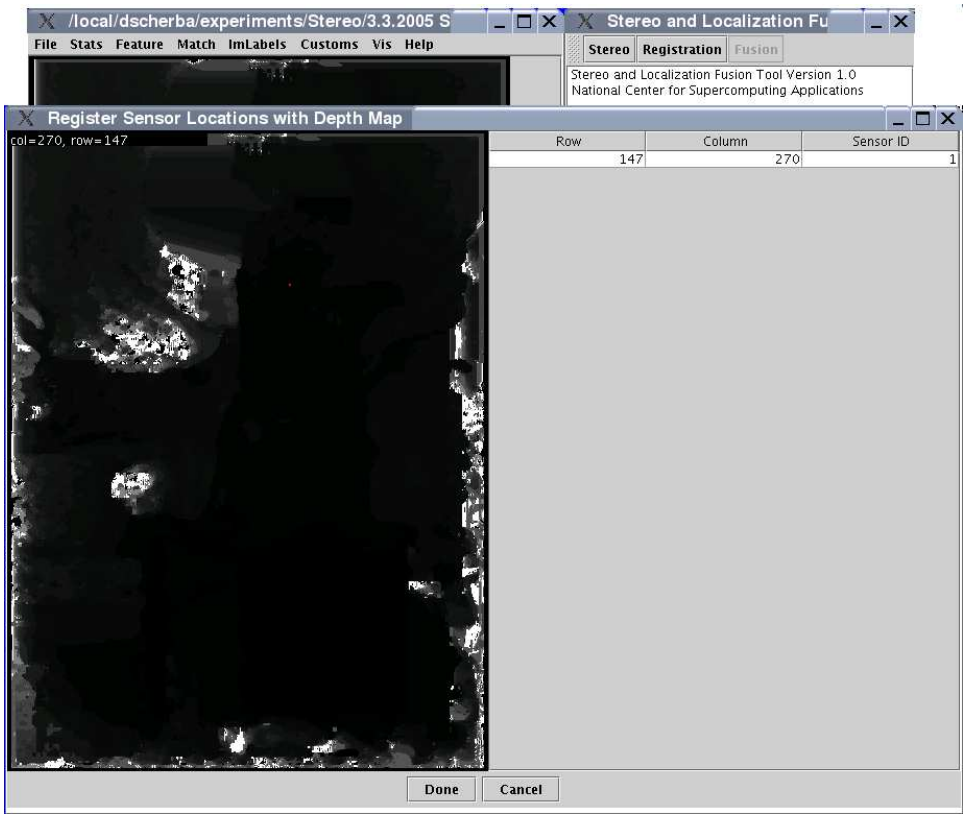

Figure C.2: Manual Registration Window

Sensor Locations with Depth Map" window. A dialog will appear asking you to input the sensor ID corresponding to that point. A list of specified points will appear in the right pane and can be edited by double clicking in the cells similar to a spreadsheet.

When you have completed registration, click on the "Done" button.

C.5 Fusion

After registering sensor locations with a depth map, the "Fusion" button in the Stereopsis and Localization Fusion window will become available. Clicking this button will bring up the dialog box shown in Figure C.3.

Filling in the form should be largely self-explanatory. The reason that both calibrated and uncalibrated data are used is that the scaling in stereo calibration must be derived. It is assumed that you know or can conservatively estimate the stereo matching uncertainty in pixels. This error depends on a number of factors for the correlation-based matching technique used in I2K. Matching error will tend to rise for scenes that do not exhibit good "texture" for the correlation window to "lock on" to. This value can also

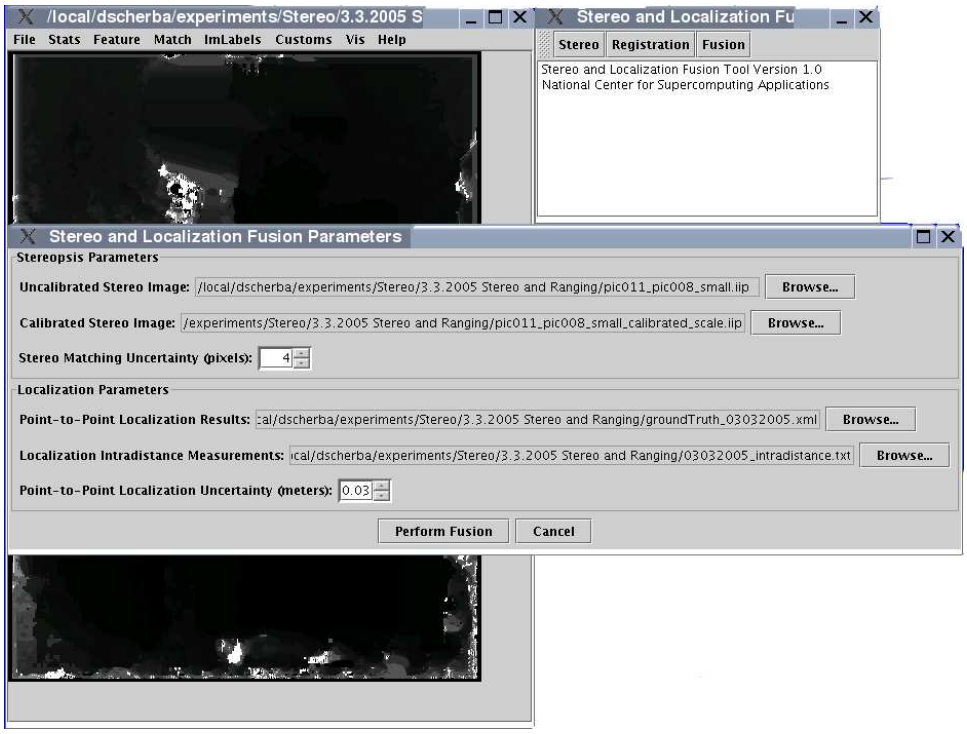

Figure C.3: Fusion Dialog Box

change depending on the amount of downsampling used in the input stereo images (e.g. downsampling the images by 2 will reduce the error by a factor of 2). We have found an error of 4 pixels to be a reasonable amount in our testing.

The "Point-to-Point Localization Results" need to be submitted as an XML file. This file format is defined in section A.4 and should not contain multiple <rangingPair> elements per mote pair, or multiple <range> elements per <rangingPair>.

The "Localization Intradistance Measurements" file is a simple text file with the following form ([]'s delineate fields and are not to be included. Lines 1 and 2 are required, the rest are optional.):

```
[n sensors]
[distance between sensors 1 and 2] [distance between sensors 2 and 3] ... [distance between sensors n-1 and n]
[distance between sensors 1 and 3] [distance between sensors 2 and 4] ... [distance between sensors n-2 and n]
[distance between sensors 1 and 4] [distance between sensors 2 and 5] ... [distance between sensors n-3 and n]
```

Similarly, you should know the point-to-point ranging error of the specified data in meters. This is the average error that you see in the ranging result (e.g. by acoustic time-of-flight ranging) between two motes in the distance regime of interest. The state-of-the-art acoustic time-of-flight ranging systems report uncertainty around 3cm at the office-sized scale.

Clicking the "Perform Fusion" button of this dialog box will start the sensor network localization and fusion process.

C.6 Obtaining Results

The output from the tool is given in two places. First, the I2K image in the main window will change to represent the fused depth map. Note that this image is not saved automatically, and doing so is suggested. Second, the fused [and unfused] sensor network localization data is dumped to the window with the I2K console output. This information, while not very useful for looking at the scene, is potentially useful in other sensor network applications.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, 2002, pp. 393-422.

[2] ALG, NCSA, UIUC, "I2K: Image to Knowledge," Aug. 2004; http://alg.ncsa.uiuc.edu/do/tools/i2k.

[3] G.O. Allgood, W.W. Manges, and S.F. Smith, "It's Time for Sensors to Go Wireless," *Sensors Magazine*, May 1999; http://www.sensorsmag.com/articles/0599/0599_p70/main.shtml.

[4] K.S. Arun, T.S. Huang, and S.D. Blostein, "Least Square Fitting of Two 3-D Point Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, 1987, pp. 698-700.

[5] AT&T Laboratories Cambridge, "The Bat Ultrasonic Location System," Aug. 2004; http://www.uk.research.att.com/bat/.

[6] P. Bajcsy and S. Saha, "A New Thermal Infrared Camera Calibration Approach Using Wireless MEMS Sensors," *Proceedings Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, CA, Jan. 2004.

[7] P. Bajcsy, "Toward Hazard Aware Spaces: Knowing Where, When and What Hazards Occur," Technology Presentation, NCSA Private Sector Program, May 2005; http://www.ncsa.uiuc.edu/Conferences/2005Meeting/agenda.html.

[8] M. Bergamasco, P. Dario, A. Bicchi, and G. Buttazzo, "Multi Sensor Integration for Fine Manipulation," *Highly Redundant Sensing in Robotic Systems*, J.T. Tou, ed., J.G. Balchen, ed., NATO-ASI Series F: Computer and Systems Sciences, Berlin: Springer Verlag, Vol. 58, 1990, pp. 55-66.

[9] P.J. Besl and N.D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, 1992, pp. 239-256.

[10] M.Z. Brown, D. Burschka, and G.D. Hager, "Advances in Computational Stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, no. 8, Aug. 2003, pp. 993-1008.

[11] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology," *Proceedings ACM SIGCOMM Workshop of Data Communications in Latin America and the Caribbean*, April 2001.

[12] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[13] Crossbow Technology, Inc., "Crossbow Technology," Apr. 2005; http://www.xbow.com/.

[14] Crossbow Technologies, Inc., "MPR400 MICA2 Datasheet," Aug. 2004; http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-05_A_MICA2.pdf.

[15] Crossbow Technologies, Inc., "MTS400/MTS420 DataSheet," Aug. 2004; http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MTS400-420_Datasheet.pdf.

[16] P. Dario, M. Bergamasco, A. Bicchi, and A. Fiorillo, "Multiple Sensing for Dexterous End Effectors," *Robots with Redundancy: Design, Sensing and Control*, A.K. Bejczy, ed., Berlin: Springer Verlag, 1992.

[17] U. Dhond and J.K. Aggarwal, "Structure from Stereo–A Review," *IEEE Transactions of Systems, Man, and Cybernetics*, vol. 19, no. 6, Nov 1989, pp. 1489-1510.

[18] Dust Networks, "Dust Networks," Apr. 2005; http://www.dust-inc.com/.

[19] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks," *Proceedings Acoustics, Speech, and Signal Processing*, May 2001, vol. 4, pp. 2033-2036.

[20] D.A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Upper Saddle River, New Jersey: Prentice Hall, 2003.

[21] D.L. Hall and J. Llinas, "An Introduction to Multisensor Data Fusion," *Proceedings of the IEEE*, vol. 85, no. 1, 1997, pp. 6-23.

[22] R.I. Hartley, "Theory and Practice of Projective Rectification," *International Journal of Computer Vision*, vol. 35, no. 2, 1999, pp. 115-127.

[23] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *IEEE Computer*, vol. 34, no. 8, Aug. 2001, pp. 57-66.

[24] H. Hirschnuller, "Improvements in Real-Time Correlation-Based Stereo Vision," *IEEE Workshop on Stereo and Multi-Baseline Vision*, Kauai, Hawaii, Dec. 2001, pp. 141-148.

[25] L.F. Hodges and E.T. Davis, *Geometric Considerations for Stereoscopic Virtual Environments*, tech. report GIT-GVU-93-02, Graphics, Visualization & Usability Center, Georgia Institute of Technology, 1993.

[26] Intel Corporation, "Intel Research: Vibration Monitoring," Apr. 2005; http://www.intel.com/research/exploratory/wireless_sensors.htm#vibration.

[27] Intel Research Laboratory at Berkeley, "Habitat Monitoring on Great Duck Island," Apr. 2005; http://www.greatduckisland.net/.

[28] F. Isgro and E. Trucco, "Projective Rectification without Epipolar Geometry," *Computer Vision and Pattern Recognition (CVPR)*, vol. 1, June 1999, pp. 94-99.

[29] A.K. Jain, R.P.W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, 2000, pp. 4-37.

[30] Leadtek Research, Inc., "GPS-9543 Datasheet," Aug. 2004; http://www.leadtek.com/datasheet/gps-oem-module.pdf.

[31] J.P. Lewis, "LM.java (Levenberg-Marquardt)," Aug. 2004; http://www.idiom.com/~zilla/Computer/Javanumeric/LM.java.

[32] D. Marr, *Vision*, San Francisco: W.H. Freeman, 1982.

[33] Network and Mobile Systems Group, CSAIL, MIT, "The Cricket Indoor Location System," Aug. 2004; http://nms.lcs.mit.edu/projects/cricket/.

[34] K. Okada, S. Kagami, M. Inaba, and H. Inoue, "Plane Segment Finder: Algorithm, Implementation and Applications," *Proceedings IEEE International Conference on Robotics & Automation*, Seoul, Korea, May. 2001, pp. 2120-2125.

[35] S. Pankanti and A.K. Jain, "Integrating Vision Modules: Stereo, Shading, Grouping, and Line Labeling," *PAMI*, vol. 17, no. 9, Sep. 1995, pp. 831-842.

[36] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.

[37] N.B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-Free Distributed Localization in Sensor Networks," Tech Report 892, Laboratory for Computer Science, MIT, 2003.

[38] W. Roush, A.M. Goho, E. Scigliano, D. Talbot, M.M. Waldrop, G.T. Huang, P. Fairley, E. Jonietz, and H. Brody, "10 Emerging Technologies," *Technology Review*, vol. 106, no. 1, 2003, pp. 33-46.

[39] S. Roy, "Stereo Without Epipolar Lines: A Maximum-Flow Formulation," *International Journal of Computer Vision*, vol 34, no. 2/3, 1999, pp. 147-161.

[40] R. Sara, "Accurate Natural Surface Reconstruction from Polynocular Stereo," *Confluence of Computer Vision and Computer Graphics*, Ed. A. Leonardis, et al, Boston: Kluwer Academic Publishers, 1999.

[41] D. Scharstein and R. Szeliski, "Middlebury Stereo Vision Page," Aug. 2004; http://cat.middlebury.edu/stereo/.

[42] D.J. Scherba and P. Bajcsy, "Depth Map Calibration by Stereo and Wireless Sensor Network Fusion," to be published in *Proceedings Fusion 2005*, Philadelphia, July 2005.

[43] Schmitt Measurement Systems, Inc., "Acuity AR4000 Datasheet," Aug. 2004; http://www.acuityresearch.com/pdf/ar4000-data-sheet.pdf.

[44] UC Berkeley, "TinyDB: A Declarative Database for Sensor Networks," Apr. 2005; http://telegraph.cs.berkeley.edu/tinydb/.

[45] UC Berkeley, "TinyOS Community Forum," Aug. 2004; http://www.tinyos.net/.

[46] E. Vincent and R. Laganiere, "Matching Feature Points in Stereo Pairs: A Comparative Study of Some Matching Strategies," *Machine Graphics & Vision*, vol. 10, no. 3, 2001, pp. 237-259.

[47] H. Wechsler, *Computational Vision*, Boston: Academic Press, 1990.

[48] K. Whitehouse and X. Jiang, "Calamari: A Localization System for Sensor Networks," Aug. 2004; http://www.cs.berkeley.edu/~kamin/calamari/.