Sensor Data Fusion Using Kalman Filters on an Evidence Grid Map

By

An Sheng, B.Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario
March 16, 2005

Abstract

Multi-sensor fusion is a common technique for dealing with uncertainty in mobile robot mapping and navigation. This introduces the problem of integrating and diverse sensor information into a common representation of the environment.

This thesis describes a framework for the fusion of sonar, infrared and visual data into a two-dimensional environmental model intended for navigation and exploration in robotics. The general data fusion framework adopts the Kalman filter theory, in order to compensate for the fact that sensor observations are uncertain, incomplete, and/or imprecise. In this thesis, a more robust data fusion strategy which integrates sensor readings from three heterogeneous sensors is studied under a realistic environment and many experiments are conducted. Experimental results are provided using real data to support our expectation that the fusion process provides clear benefits showing that it is possible for multiple sensor systems to complement and correct one another.

Keywords: mapping, ultrasonic, stereovision, IR proximity sensor, mobile robot, evidence grid map, sensor fusion, Kalman filter.

Acknowledgements

I would like to gratefully acknowledge the enthusiastic supervision of Dr. Mark Lanthier and Dr. Doron Nussbaum in finishing this thesis. I thank Yanxuan Zhang for the technical discussions on the Kalman filter and sensor model, and Shaoying Zou for the help with robotic experiments and building.

I am forever indebted to my parents for their support and encouragement when it was most required.

Finally, this thesis is for my most lovely wife, Wenhao Gao. Without her, I am nothing.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Autonomous robots are often required to perform tasks in unknown environments. The mandatory requirement to finish this task successfully is that of acquiring knowledge of the environment as well as its location. This knowledge allows an autonomous robot to plan and navigate its way through objects. To accomplish this task, autonomous robots rely on sensors (e.g., camera, ultrasonic sensor etc).

Sensor data fusion is the process of combining data from different sensors in order to generate more complete information of the surrounding environment. Sensor data fusion is mainly used in applications that require the detection of a target and/or to map an environment. More generally, different inputs may originate from a single sensor at different times (fusion in time) or from multiple sensors at the same time (multi-sensor data fusion) [44].

Data fusion enhances the accuracy-related performance of a large number of tasks such as obstacle detection, object recognition, identification and object tracking. These tasks and others like these may be encountered in many application domains such as Defense, Exploration, Space and Navigation.

In principle, the fusion of multi-sensor data provides significant advantages over the use of single source data. In addition to the statistical advantage gained by combining same source data [18] (e.g., obtaining an improved estimate of a physical phenomena via redundant observations), the use of multiple types of sensors may increase the accuracy

because of the diverse characteristics of the individual sensors. For example, a wide beam radar sensor is able to determine the distance to an obstacle accurately, but has a limited ability to determine the angular direction of the obstacle. By contrast, an infrared proximity sensor can accurately determine the angular direction to the obstacle, but is less accurate for range measurements. If these two observations are correctly associated, then the estimate of the location of the obstacle based on the combined information will be more accurate than that of either of the two independent sensors. Another benefit of multi-sensor data fusion is that a more complete sensor reading coverage may be obtainable. For example, car radars/sonars can be found on new luxurious cars. Drivers can detect most obstacles through the rear view mirror but blind areas still exist. Thus, a car radar/sonar improves the driver's object detection ability when blind areas are considered.

Sensor data fusion is often categorized as low, intermediate or high level fusion, depending on the processing stage at which data fusion takes place.

- Low level fusion (also called *simple data fusion*) combines several sources of raw data to produce new (e.g., averaged) raw data that is expected to be more informative and synthetic than the original input data.
- Intermediate level fusion (also called *feature level fusion*) combines various features, such as edges, corners, lines and texture parameters into a feature map that may then be used by high-level processing. Those features may come from several raw data sources (e.g., multiple sensors, a single sensor at different moments, etc.) or from the same raw data. In the latter case, the objective is to

find a limited number of relevant features among those available from several feature extraction methods.

- High-level data fusion (also called *decision fusion*) combines decisions coming from several experts. By extension, it is called decision fusion even if the experts return a confidence (score) but not a decision. Methods of decision fusion include voting methods, statistical methods and fuzzy logic-based methods.

Note that the above categorization is not exhaustive, as input and output of the fusion process may represent different levels of processing. Typically, fused raw data is used as input for feature fusion and then features are used as input for decision fusion. Various researchers define the level of fusion differently; some by the output level, others by the input level [21]. In practical problems, the applied fusion procedure is often a combination of the previously mentioned three levels.

In this thesis, we focus on low and intermediate level sensor data fusion. We research heterogeneous sensor data fusion with respect to time and/or location. Our research focused on complementary multi-sensor data fusion by exploring the advantages and disadvantages of various sensor types. We conducted carefully designed experiments in order to verify the design of our sensor combination strategy.

1.1 Contributions

The contributions of this thesis include both software and hardware aspects. Our hardware contribution focused on the construction of an autonomous robot. Here we researched the properties of various sensors and determined the types and number of

sensors to be installed on the robot. Note that the body of the robot already included a sonar sensor ring.

Once we obtained the sensors, the software framework was developed to allow the robot to manipulate them. Experiments were also conducted to test the performance of these sensors.

Software contributions include an expandable framework developed for sensor data fusion, which are used in our research and experiments.

We studied sensor data fusion based on three different kinds of heterogeneous sensors instead of two, which has been more broadly researched in literature. We investigated the pitfalls of each sensor type on our robot and several experiments were designed to test different approaches and improve the performance. We published our preliminary research in IASTED 2004 [35].

A number of algorithms were studied and we implemented our software framework and a modified algorithm which best fit our need. Finally, we conducted several very complicated experiments in a realistic environment with realistic obstacles to verify our data fusion algorithm and made a detailed experimental analyses based on our experimental results.

The structure of this thesis is as follows. In Chapter 2, we review commonly used sensor types, data fusion strategies and mapping techniques, examining their relative advantages and disadvantages. In Chapter 3, we introduce different approaches that we applied in order to improve sensor readings for each kind of sensors as well as our sensor data fusion approach, motivation and methodology. We also present our implementation:

the software framework. In Chapter 4, we introduce all of our experiments and the analysis of their results. High-level map building based on a grid map is also included in this chapter. We summaries the thesis and discuss future works in Chapter 6.

Chapter 2 Background

2.1 Sensors Review

In order for a mobile robot to perform its assigned tasks, it often requires a representation of its environment as a navigator. These problems have been characterized by the three fundamental questions of mobile robotics: "*Where am I?*", "*Where am I going?*" and "*How can I get there?*". Sensors are used to assist in answering these questions. For example, a robot may answer the question "*Where am I?*", which normally refers to a localization problem, by using odometers and/or accelerometers to measure its movement.

No single sensor can get all information about an environment and each sensor type has its strengths and weaknesses, there is no perfect [1] sensor. Combining readings of several different sensor types improves the capability of measuring the environment. For autonomous robots, range sensors, which can obtain the distance information between a robot and obstacles, are most useful to resolve the localization problem. In this section, we introduce the most commonly used range sensors on robots and the characteristic of these sensors.

---

[1] A perfect sensor is one that can get correct sensor readings under all circumstances.

2.1.1 Radar - Radio Detection and Ranging

Radar is a range or distance measuring device. A radar sensor consists of a transmitter, a receiver, an antenna, and an electronic system to process and record the data. The transmitter generates successive short bursts or pulses of microwave radiation at regular intervals, which are focused by the antenna into a beam. The radar beam emitted by the antenna forms an angle, which is termed the "emitting angle" and a radar can detect obstacles within the emitting angle only. The antenna receives a portion of the transmitted energy that is reflected from various objects within the beam. By measuring the time delay between the transmission of a pulse and the reception of the echo from different targets, their distance to the radar, which relates to their location, can be determined. Radar is widely used in remote sensing due to its high accuracy, but the large size collecting dish and waveguide make it difficult to apply to smaller robots.

2.1.2 Laser Range Finder

The principles of a laser range finder are the same as those of a radar sensor. A laser range finder transmits light instead microwave. The transmitted light is reflected by targets and some of this light is scattered back to the instrument where it is analyzed. The change in the properties of the light enables some attribute of the target to be determined. The time taken by the light to travel from the laser range finder to the target and back is used to compute the distance to the target. A laser range finder can obtain distance information at a very high accuracy.

The main drawbacks of the laser range finder relate to the nature of the target surfaces. In the presence of surfaces that reflect light poorly, such as dark colors and soft materials (fabrics, furs, carpets), a reflected laser beam is returned with a feeble intensity due to absorption by the materials. However, more serious problems with lasers occur when the target has high reflectance surfaces, such as mirrors and windows. A mirror changes the laser beam direction, producing an enlarged duplicate of the environment behind its surface, which cause erroneous sensor readings. Glass windows appear to the laser beam as transparent, partial mirrors, or as perfect mirrors, depending on the glass type, thickness and angle of incidence, which also cause noisy and/or wrong sensor readings.

2.1.3 Electrostatic Ultrasonic Sensor

Ultrasonic ranging and detecting devices use high-frequency sound waves to detect the presence of an object and to determine its range. Such systems either measure the echo reflection of sound from objects or detect the interruption of a sound wave as objects pass between the transmitter and the receiver.

An ultrasonic transducer emits and receives an ultrasonic signal (see Figure 1). Within a detectable distance, the incoming echo is checked, the distance that the signal should travel in the environment of the sensor in that amount of time is calculated, and the estimated distance is generated accordingly. If the object is too close to the sonar transducer, the echo arrives before the ultrasonic transducer has reached steady state and is ready to receive; thus, the object will not be detected.

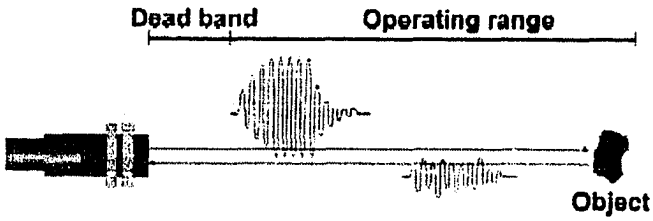


Figure 1: The principle of ultrasonic sensor.

2.1.4 Infrared Proximity Sensor

An infrared (IR) proximity sensor contains an infrared transmitting diode, a position sensitive photo detector, some optics and a signal-processing unit. Generally the principle of infrared proximity sensors is to send out a beam of infrared light, and then compute the distance to any nearby objects by analyzing the characteristics of the returned signal. There are several ways to compute the distance, each with its own advantages and disadvantages:

Time of Flight (TOF): The sensor sends out a beam of light and calculates the time it takes for the beam to bounce back. An IR sensor using TOF for distance measuring is expensive but very accurate; however, they are not very accurate for determining short distances due to the high speed of light.

Returned Signal Strength: The sensor sends out a beam of infrared light and detects the amount of reflection. A photodiode and an IR LED are sometime used to create a signal strength infrared detector. There are, however, some potential problems. First, ambient infrared light would interfere with the sensor readings. Second, two targets at the same distance with different reflectivity would give different readings.

Pattern: The sensor sends out a beam of infrared light in a specific wave pattern, and a photodiode is configured to only detect the IR having this pattern. This strategy has potential problems similar to those of detection by returned signal strength.

Triangulation: This is the most common type of infrared sensing technique used. The sensor sends out a beam of light and measures the angle of return, using trigonometry to calculate changes in the position of the target (see Figure 2).

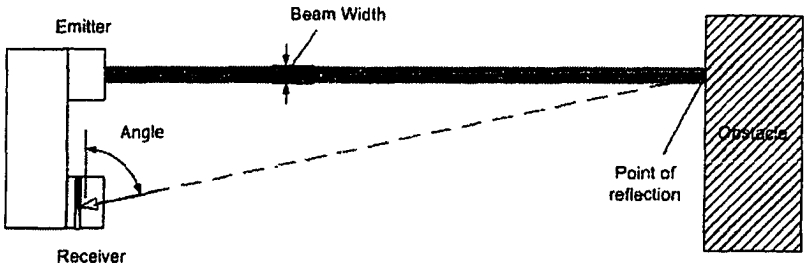


Figure 2: The IR proximity sensor based on triangulation.

The advantage of the IR proximity sensor is its narrow beam width, which allows it to obtain feature information of obstacles more accurately. The main problem of IR proximity sensors is their limited detecting range (normally less than 2 meters) and their sensitivity to surface reflectivity, which is discussed in detail in Chapter 3.1.1.

2.1.5 Stereo Vision Sensor

Binocular stereo vision is a biologically motivated approach that uses two slightly different views of a scene to extract information about its three-dimensional properties. Two eyes or cameras looking at the same scene from different perspectives provide a means for determining three-dimensional shape and position. The same object will have a slightly different position in the left image and the right image; by calculating the

discrepancy in position of the same point on the two images, depth information can be determined by triangulation.

The essential principle in stereo vision is to find *correspondence* points between two stereo images. Each point in real world space is a 3D point within an *image space*. A system with two cameras is shown in Figure 3. The focal points of the cameras are F_l and F_r, and their image planes are I_l and I_r. A point P in the 3D image space is projected onto P_l in the left image and onto P_r in the right image. The projections of a single 3D point in the different image spaces are termed *correspondence points*. The difference in the position of corresponding points in their respective images is called the *disparity* of that point (see Figure 3 b). Disparity is a function of both the position of the 3D point and of the position, the orientation, and physical characteristics of the stereo devices (e.g. cameras, lens).

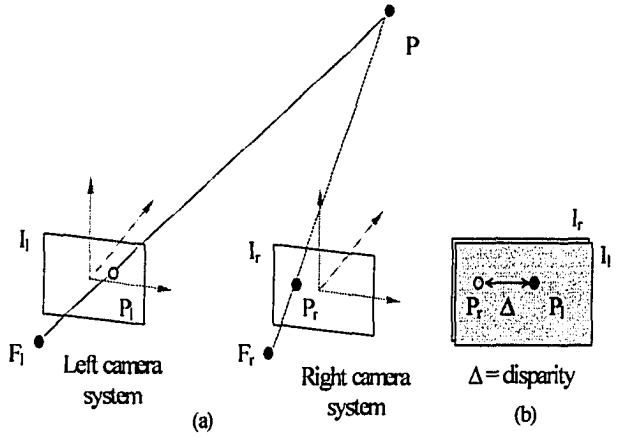


Figure 3: (a) Projection of a point *P* onto image plane of cameras, (b) cyclopean view: the disparity *Δ* is the difference between P_l and P_r.

In addition to providing the function that maps pairs of corresponding image points onto 3D points in image space, the lens layout of the stereo camera defines the search

dimension for corresponding image points. Any point in the 3D image space, together with the lens centers of the two cameras, defines an *epipolar plane*. The intersection of the plane with the image plane of each camera is an *epipolar line* for the camera (see Figure 4). Every point of an epipolar line in a stereo image must correspond to a single point on the corresponding epipolar line on the other stereo image. The search for a matching point in the first image may therefore only be along a line in the other image plane; thus, there is no need to search the whole image for the correspondence point.

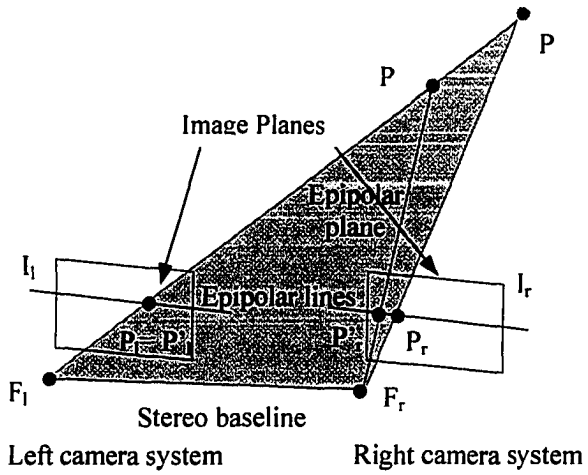


Figure 4: The epipolar lines and the epipolar planes.

Ideally, one would like to find correspondences for each individual pixel in both stereo images. However, there are generally many pixels with the same color (intensity), making it difficult to find a match for only one pixel. In practice, continuous areas of

image intensity, which form basic units, are matched. This approach (termed area matching) usually involves some form of cross-correlation[2] to establish correspondences.

The main problem in matching is to find a best match correlation. Correlation scores are computed by comparing a fixed window in the first image against a shifting window in the second. The second window moves in the second image by integer increments along the corresponding epipolar line, and a correlation score curve is generated for integer disparity values. The reported disparity will be that which provides the largest peak in the correlation score (Figure 5).

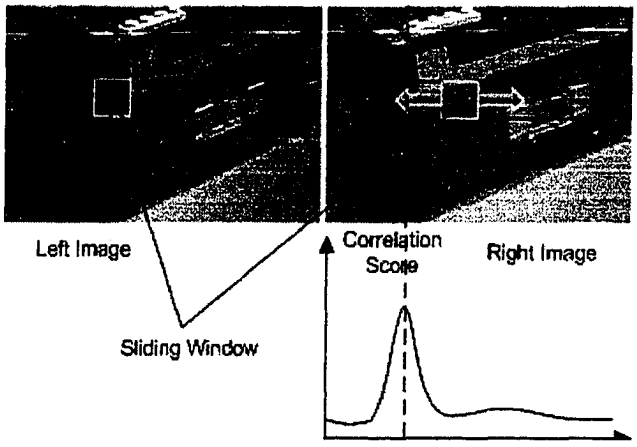


Figure 5: Correlation result of a pair of images.

There are many applied criteria, that can quantify the similarity between two correlation windows, we must choose among them to produce reliable results in minimal computation time. Denote by $I_1(x, y)$ and $I_2(x, y)$ the intensity values at pixel (x, y) for sliding windows I_1 and I_2 respectively (shown in Figure 5). Let the width of the image be

[2] A measure of the similarity of two patterns.

2n + 1 and the height of the image be *2m + 1*, such that the origin is in the center of the image. The columns run from –n to n and the rows from –m to m. Therefore, the indexes that appear in the formula below vary between -n and n for the i-index and between -m and m for the j-index.

The normalized sum of squared differences (SSD) correlation algorithm [57] is used by our stereo camera sensor. The normalized SSD equation that we used is as stated in Equation 1 as:

$$S(x,y,\Delta) = \frac{\sum_{i,j}[I_1(x+i,y+j) - I_2(x+\Delta+i,y+j)]^2}{\sqrt{\sum_{i,j} I_1^2(x+i,y+j)} \times \sqrt{\sum_{i,j} I_2^2(x+\Delta+i,y+j)}} \quad (1)$$

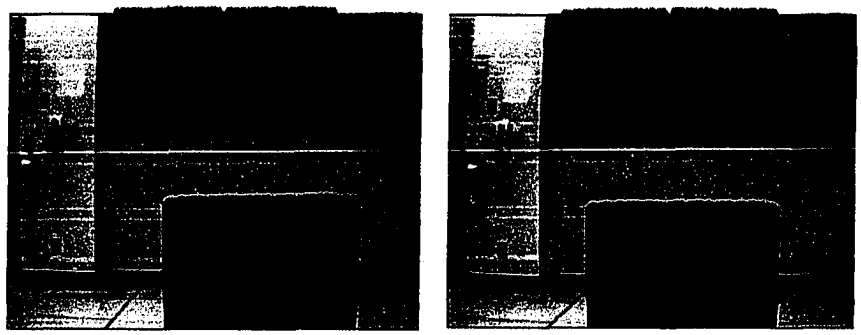
Where $S(x,y,\Delta)$ is the normalized sum of squared differences of the pixel (x,y) whose disparity is to be calculated using a n × m window. d is the offset in the right image with respect to the pixel (x,y) in the left image. The computed disparity is the one that has a selected window with a suitable size which minimizes Equation 1.

Figure 6 and Figure 7 show stereo images for two vision sensor experiments. The points in each image are the 2D projections of a single 3D scene. Before we examine how to combine these stereo images, we examine the correspondence process in a little more detail[3].

---

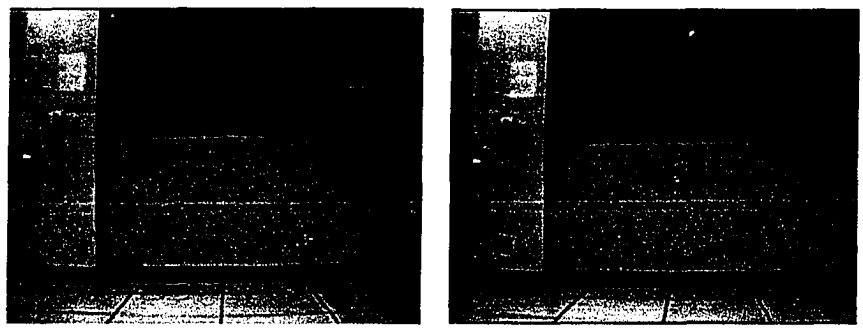
[3] Most of the approaches that determine correspondence are either intensity-based or feature-based. The differences between intensity-based and feature-based correspondence are discussed by Ullman [59] and also by Haralick and Shapiro [28].

Stereo algorithms differ in not only the way they encode correspondence between pixels, but also how much of the image data is used. When designing a stereo algorithm, one soon discovers that the textured regions in a scene are relatively easy to match (except in the case of repeated textures), while regions lacking patterns are hard to match accurately. Stereo algorithms can be roughly broken down into two approaches, depending on whether or not they match textureless regions.



(a) (b)

Figure 6: The stereo image pair for experiment one. (a) left image. (b) right image.



(a) (b)

Figure 7: The stereo image pair for experiment two. (a) Left image. (b) Right image.

In the feature-based approach (see [24][26][40][3][47]), only the "feature" pixels are detected and matched. These are the physically significant image pixels, such as intensity edges or corners. Textureless regions are left unmatched. The motivation is that

the pixels in textureless regions cannot be matched reliably anyway. The advantage of the feature-based methods is that they produce accurate results; the results are rather sparse, though. Many applications require dense measurements, and measurement interpolation is a difficult problem in itself. Feature-based methods were especially popular in the early days of computer vision because image quality was generally poor and so comparing the raw image intensities of pixels was not a reliable measure for the likelihood of their correspondence.

The second approach, intensity-based correspondence, is to match all (or almost all) image pixels to produce dense disparity estimates. The intuition is that by propagating disparity estimates from the highly textured areas, the disparity in less textured areas can be inferred. The requirement of dense disparity estimates in many applications and the improvements in image quality account for the greater popularity of the dense stereo algorithms in recent years [50][54].

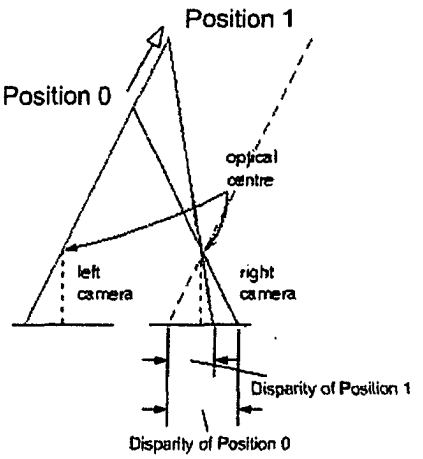
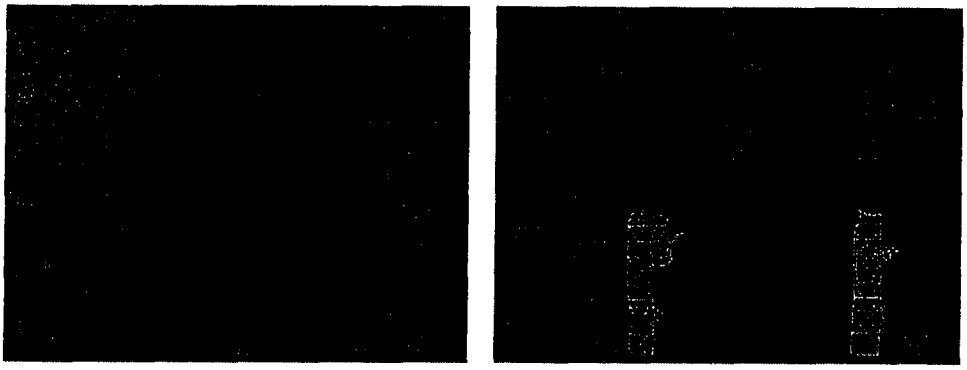


Figure 8: The relationship between disparity and distance.

Since disparity results generated by feature-based approaches are sparse and prone to noise, it is not suitable for robot navigation or map building. All of our approaches use intensity-based correspondence. Such approaches are based purely on the levels of gray (or color) in the images. Each point in one image is matched to a point in the other that has a similar or, ideally, identical intensity. The reasoning behind such methods is that the color of an object is usually constant and so the color, or intensity, of its projected pixels should remain constant. This assumption is not always valid because of varying lighting conditions. For example, a ball, which is partially covered by shadow, will show different color although the color is same. Another major problem with intensity-based approaches is that there are often many points in an image that have the same, or very similar, intensities. This causes the number of possible matches to be very large and so makes the task of identifying the correct one more difficult. One advantage of this approach, however, is that every point in an image has an intensity value associated with it and theoretically, we can find at least one correspondence for every point in the image.

Calculating distance information from observed disparity is straightforward, as shown in Figure 8. The distance is in inverse proportion to disparity; the bigger the disparity, the closer the obstacle is. If the obstacle is infinitely far away, the disparity will approach zero (see Figure 8).



(a) Disparity map for experiment one. **(b) Disparity map for experiment two.**

Figure 9: The disparity map for experiments one and two.[4]

The result of the stereo matching algorithm is disparity values and the disparity values of pixels in a scene form a disparity map. Since there is a linear relationship between the disparity value of a pixel and its distance value, disparity maps are normally interchangeable with distance maps. Figure 9 shows a representation of the disparity in a scene; the darker areas indicate more distant obstacles and the lighter areas are closer obstacles.

2.2 Classic Problems in Sensor Measurements

We introduce the common problems for different sensors in this section. We classify sensors into two main categories: active sensors and passive sensors. An active sensor is a detection device that requires input energy from a source other than that which is being sensed. A passive sensor is a detection device that obtains information by reception of energy from that which is being sensed.

---

[4] Snapshots have been lightened for print purposes.

2.2.1 Classic Problems for Active Sensors

All problems that we discuss in this section apply to common active sensors such as sonars, radars and laser range finders etc.. We use an ultrasonic sensor as an example to illustrate these problems.

Blind Spot

Before further discussion, it is necessary to introduce some definitions which are visually depicted in Figure 10.

1. Minimum sensing distance: Lower limit of the specified sensing range.
2. Maximum sensing distance: Upper limit of the specified sensing range.
3. Blind zone: Zone between the sensing face of the sensor and the minimum sensing distance in which no object can be reliably detected. The presence of an object in this blind zone during the sensor operation could lead to instability of the output states.
4. Overall beam angle: Angle of emitted ultrasonic wave in regards to the reference axis.

Blind spots occur when same transducer is used for both sending and receiving an ultrasonic signal. After sending the pulse train (See Figure 11 the pulse train contains 16 pulses), the transducer foil must stop "ringing" before it can receive the returning signal. In order to stop "ringing", a blink pulse is sent to make the surface of the transducer still (see Figure 11, since the Polaroid® 6500 sonar has capability to detect multiple obstacles during single excitation, there are two blink pulses) and not turning on the receiving circuitry until a delay interval has passed. This means that the sensor cannot detect an

object whose distance to the transducer is less than half the distance that sound travels during the delay interval. Ultrasonic sensors using only one transducer for both sending and receiving can not avoid this problem. For the Polaroid® 6500 transducer, the "blind spot" is 0.6ms and then the blink distance is 10.5cm (see Figure 11).

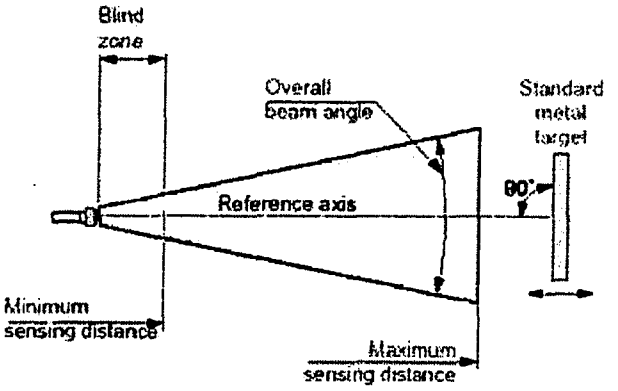


Figure 10: Terms used to describe ultrasonic sensor operation characteristics.

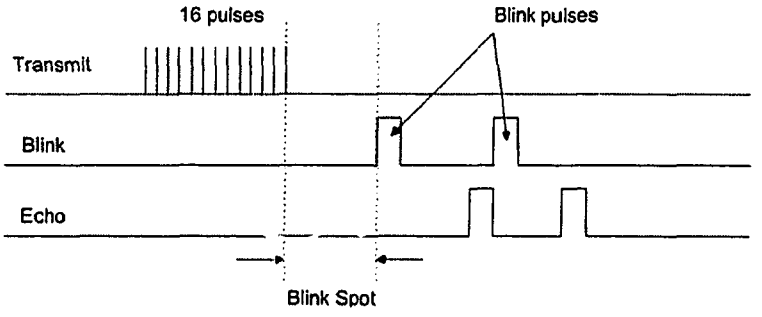


Figure 11: The blink spot due to blink pluses.

Specular Reflection

When the sound wave of the sonar (echo) strikes a smooth flat object at an angle sufficiently distant from the sensor's perpendicular, it is often reflected away from the sonar transducer, resulting in an invalid range reading. Figure 12 shows a typical example of such case, where almost the entire echo is reflected away from the sonar's transducer due to the smooth surface of the obstacle. A mapping algorithm incorporating the sonar range information will then either interpret all the space in the sonar beam as empty space, if no range reading was returned, or place an obstacle further from the sensor than is correct, if the sonar wave struck multiple obstacles before returning to the sensor.

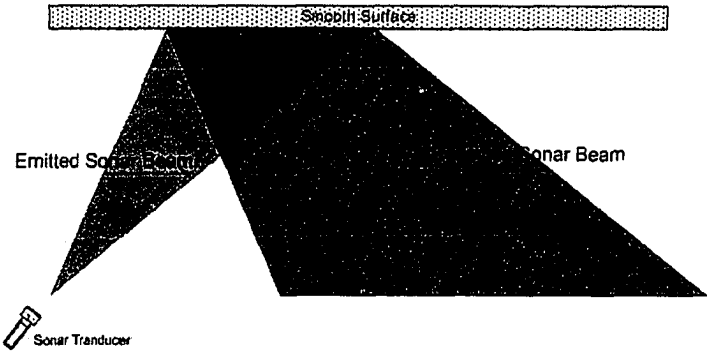


Figure 12: The specular reflection of sensor sonar.

Crosstalk

Crosstalk is an undesirable phenomenon in which one sensor receives ultrasound waves emitted by other sensors. When multiple sonar sensors fire simultaneously, crosstalk may occur between adjacent sonar transducers. Figure 13 shows this scenario. In the presence of a band-shaped obstacle having a width close to that of the beam, it is

very easy for single reflection crosstalk to occur. As the name suggests, there is only one reflection. The reason for single reflection crosstalk is the wide beam width of sonar and the distance between adjacent transducers. If the beam were narrow enough or sonar transducers are separated far enough from each other, single reflection crosstalk would not be an issue.

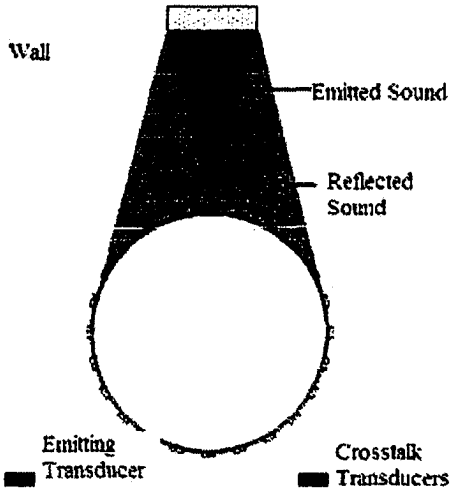


Figure 13: Principal of the single reflection crosstalk.

Multi-reflection crosstalk occurs when the sound wave rebounds off various surfaces in the environment several times before it comes back to a transducer. In Figure 14, the robot is close to a corner, the echo emitted by one transducer is reflected by walls two to three times and returns to other transducers instead of the one which emits the echo. The transducers in brown are those which may generate erroneous sensor readings due to crosstalk.

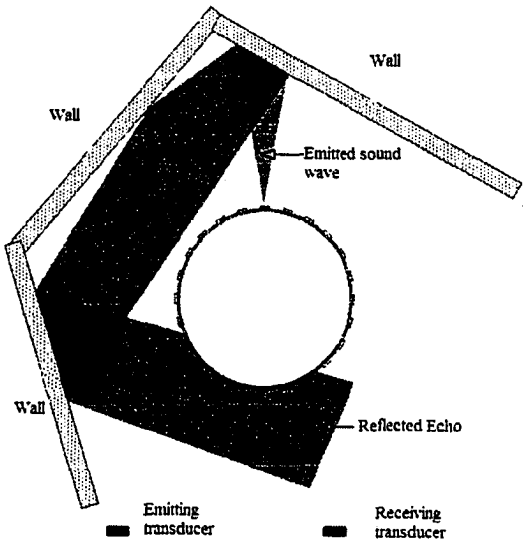


Figure 14: Principle of multi-reflection crosstalk.

Multi-Reflection

Multi-Reflection is similar to crosstalk, but multi-reflection only affects the sensor that emitted the ultrasound signal. When multi-reflection happens, the sensor that sends an ultrasonic pulse will not receive the echo directly reflected by obstacle, but the echo reflected several times by the surfaces of different obstacles (see Figure 15). The sensor will get a reading of a much longer distance compared with the actual distance from the transducer to the obstacle. In real environments, multi-reflection seldom happens, because it requires the surfaces of obstacles be smooth enough to have specular reflection and form a path that reflects the signal back to the sensor.

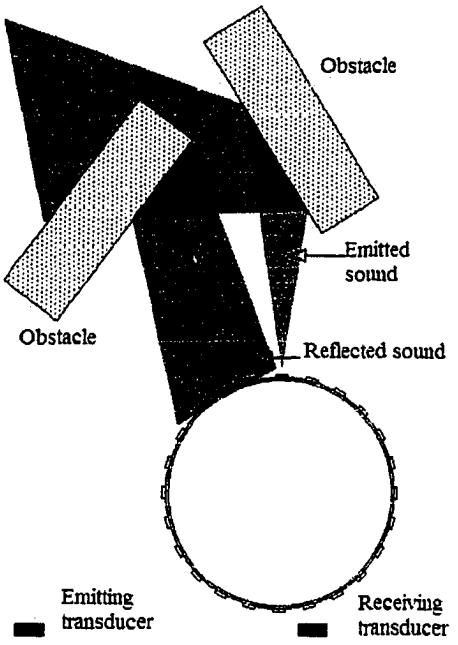


Figure 15: Sensor noise due to multi-reflection.

External Noise

An external sound that is sufficiently loud and that contains transducer sensitive frequencies may cause the receiver circuit to be falsely triggered and lead to noisy data. We may have observed this with our ventilation system which can produce high pitched whistling. These external noise sources can be classified as active or passive and as broadband (i.e., white noise or random frequency) or narrow-band (fixed frequency).

If possible, an experiment site should be free of extraneous ambient background noises. A noise survey can aid in the decision for site selection. The survey should at least cover daily and weekly patterns. A qualitative survey should be conducted to identify all active noise sources.

Internal Noise

In addition to external noise that causes false detection, active sensors can exhibit internal sensor noise which can cause problems as well. Since the common place for all active sensors is that they emit energy and detect the returned energy. The commonly encountered internal noise for active sensors is electronic noise. In the case of sonar sensors, the sonar transducer's exciting circuit could be a possible source of internal noise.

In order to avoid the excited transducer being detected as returned signal, the transducer control circuit will send several blink pulses to stop the excited transducer in order to detect echoes (see Chapter 2.2.1). If electronic noise arrives after the blink pulses (e.g., from power supply), the transducer may be triggered by these noises and generate false detection.

To avoid these internal sensor noises, three different approaches are normally applied. Firstly, large capacitors, which can filter out the DC voltage vibrations, are used in the power supply circuit of all sonar transducer control boards. Secondly, a good shield is especially important for our case since the electronic noise generated by the robot's DC motors could couple into the transducer control circuit. Lastly, we avoided the use of plugs and switches on the circuit board and soldered all wire connections. When the robot moves and turns, unsoldered wire connections and plugs may generate electronic noises.

2.2.2 Classic Problems for Passive Sensors

All problems we discuss in this section also apply to common passive sensors, but mostly to cameras. We use a stereo vision sensor, which is a commonly used passive distance sensor, as an example to describe these problems.

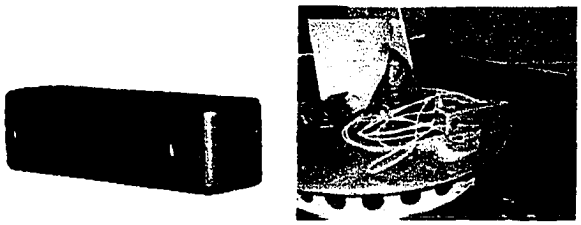


Figure 16: The Bumblebee stereo camera vision sensor.

The K2A is equipped with a "Bumblebee" stereo vision sensor from Point Grey™ (as shown in Figure 16), which is a two-lens stereovision camera system that is pre-calibrated to account for lens distortions and camera misalignments. It does not require in-field calibration and is guaranteed to stay calibrated. The specification of the Bumblebee camera's stereo images are 320 × 240 pixels. We used a 320 × 240 two-dimensional array to store the distance information.

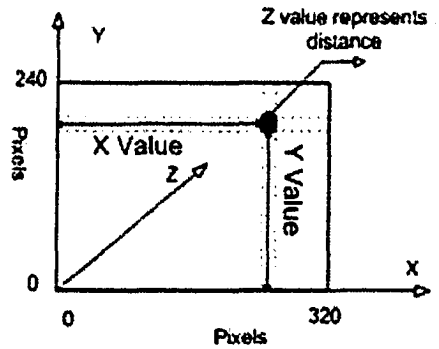


Figure 17: The 3D coordinate system for our stereo images.

We define the column id of the array as X, the row id of the array as Y, and the values stored in the array as Z, which are the distances to obstacles (See Figure 17).

A 2D cross section depth map defines a row (Y is constant) in the two dimensional array (see Figure 18).

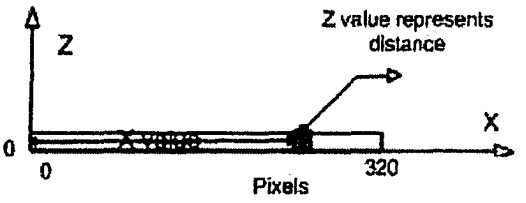
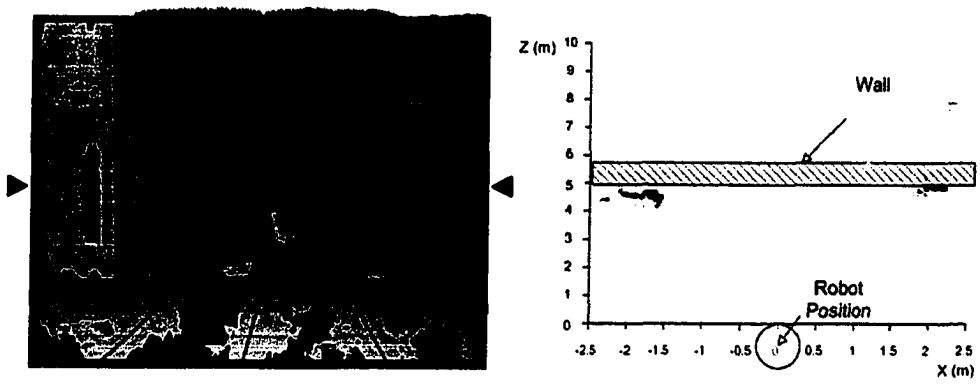


Figure 18: The 2D coordinate of scan line.

The Miscorrelation Problem

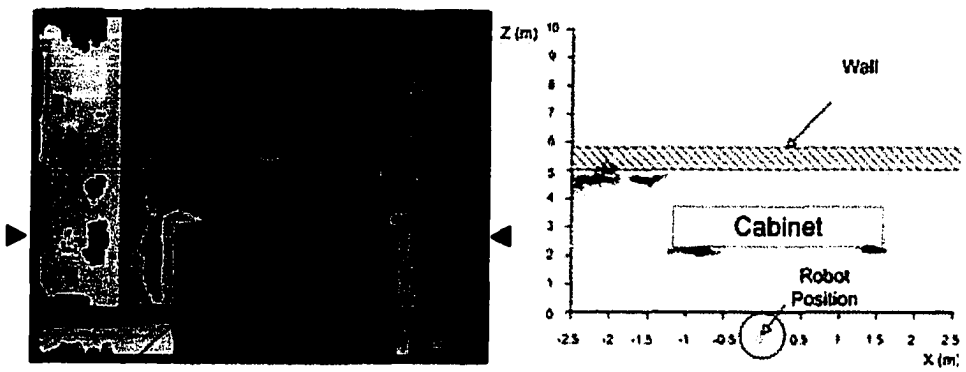
In areas where the surfaces of the obstacles do not exhibit much texture, the matching algorithm is not able to find reliable distance values for the obstacles, which is called the miscorrelation problem.

In our experiments, the resolution of the stereo image pair is 320 × 240 pixels and there are 76,800 pixels in total in each image. In practice, it is computationally expensive to match every pixel in the right image with a pixel on the left. When the scene contains too many similarly colored regions or large areas that have little texture, this will usually lead to a correspondence problem (also known as miscorrelation), as there is too much ambiguity for the matching process. This occurs, for example, with scenes containing blank walls or featureless surfaces such as those of a filing cabinet, the matching algorithm will not be able to find a peak on the correlation score curve. In a disparity map, these points are shown in black (see Figure 19(a) and Figure 20(a)).



(a) (b)

Figure 19: (a) The disparity map with miscorrelation region for experiment one. (b) The depth map for Y = 0 m. The origin of X-axis is the focus of the right lens.



(a) (b)

Figure 20: (a) The disparity map with miscorrelation region for experiment two. (b) The depth map for Y = -1m. The origin of X-axis is the focus of the right lens.

As shown in Figure 19 (a) and Figure 20 (a), non-black pixels represent valid depth information. Black pixels form regions, which are the regions of miscorrelation, where depth information are invalid due to miscorrelation. These regions of miscorrelation are shown as gaps in depth maps (see Figure 19 (b) and Figure 20 (b)).

It is easy to see that miscorrelation is a serious problem for stereo camera sensors. If there are large textureless regions, a high percentage of depth values will be corrupt or unreliable.

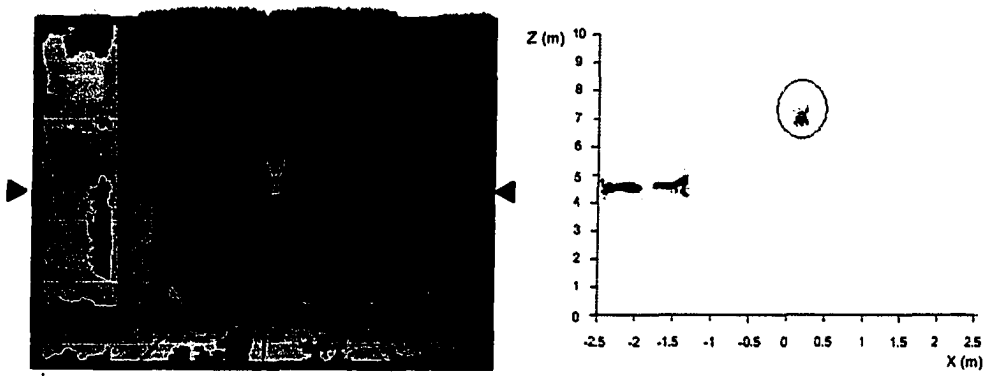
The Error Correlation Problem

Error correlation is another problem which is caused by textureless scenes, blank regions or repetitive patterns. It is the result of incorrect matches during the correlation matching where pixels in the left and right images are incorrectly matched together.

The matching algorithm also may make an incorrect correspondence due to repetitive background color or pattern such as striped patterns in blinds. In Figure 21 (b) and Figure 22 (b), the circled distance values are erroneous due to error correlation. The error correlation in Figure 21 (b) is caused by the blank wall, and the error correlation in Figure 22 (b) is due to the textureless region of the side panel of the cabinet. Errors due to error correlation may appear in a random pattern making this data similar to that of noisy sensor readings. When detecting a textureless region, there may have no obvious peak on the correlation curve. At this case, image noises will affect the result significantly and a slight sensor noise may change the peak position.

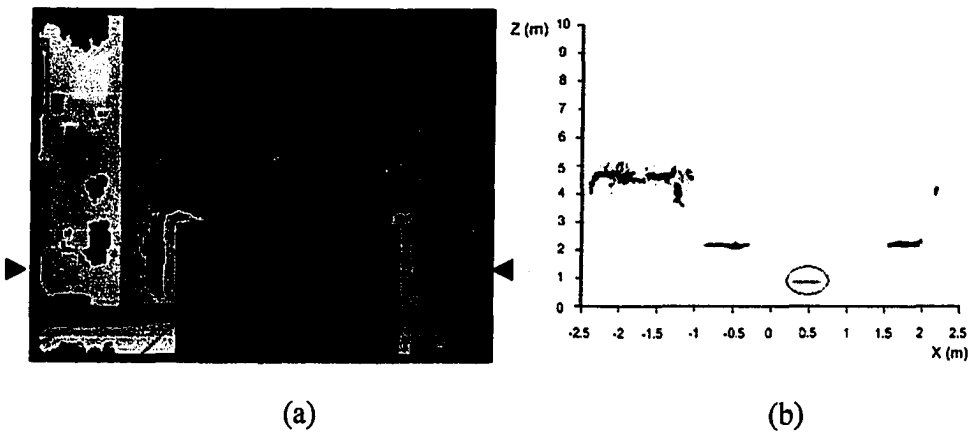
The distribution of depth values (i.e., range) for correctly matched pixels is of sufficiently low deviation. However, noise due to stereo mismatches is a serious problem. Indoor environments often contain blank surfaces, repetitive patterns, and time-varying light sources that can cause these types of errors. Increasing the number of cameras can

reduce error correlation. A three lens stereo camera can improve the correlation result by making comparisons on both the X axis and the Y axis.



(a) (b)

Figure 21: (a) The disparity map with error correlation spots. (b) The depth map with error (circled part) for experiment one.



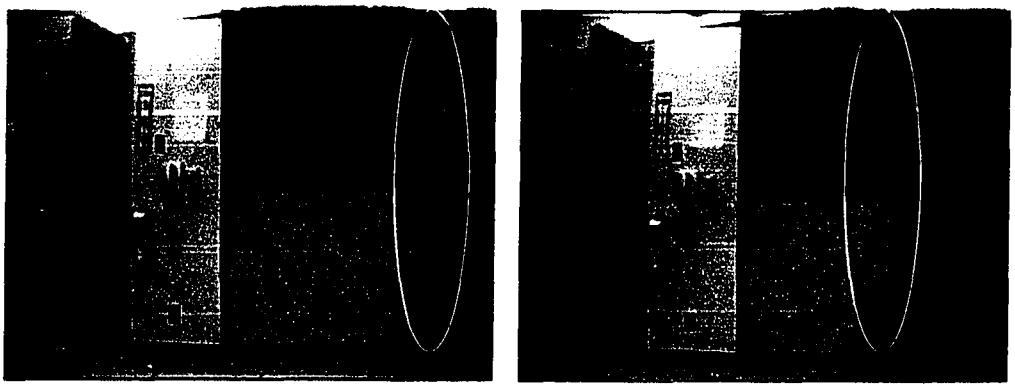
(a) (b)

Figure 22: (a) The disparity map with error correlation spots. (b) The depth map with error (circled part) for experiment two.

The Binocular Parallax Problem

Binocular parallax [58] is the difference in the angles formed by the lines of sight to two objects situated at different distances from the eyes; it is a factor in the visual perception of depth, and the reason that stereo cameras can sense depth.

Because of binocular parallax, some parts of a scene in the right image may not appear in the left one and vice versa. As can be seen in Figure 23, the left side of the box on the right edge appears in the left image but does not show in the right. This difference makes it impossible for the matching algorithm to make a correspondence on the box.



(a) Left image. *(b) Right image.*

Figure 23: The stereo image pair with a binocular parallax problem.

On the color disparity map (see Figure 24), there is a miscorrelation column region due to a binocular parallax error at this spot in the image.

The closer an obstacle is to the stereo camera, the higher the probability of a binocular parallax problem occurring. When an obstacle is extremely close to the stereo camera, for example, when the obstacle is almost between the two lenses, the left and right images of the stereo pair would be completely different. Therefore, as the robot

moves close to the obstacle, the possibility of having binocular parallax errors increase dramatically.

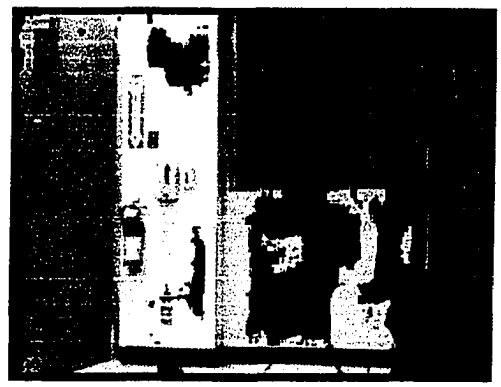


Figure 24: The colored disparity map.

The Noise from CCD Image Sensors

For a digital camera, the CCD (charge-coupled device) is a core component. It determines the noise performance, instead of the system electronics which also generate noise but the quality of CCD defines the noise level of the camera system. Some papers have made detailed analyses of CCD camera noise sources [31][56]. There are two kinds of noise sources: temporal and spatial. Temporal noise can be reduced by frame averaging, whereas spatial noise cannot. However, some spatial noise can be removed by frame subtraction or gain/offset correction techniques. An example of temporal noise is *shot noise*; an example of a spatial noise is *dark current*.

Shot noise is time-dependent fluctuations in electrical current caused by the discreteness of the electron charge, is well known to occur in solid-state devices, such as tunnel junctions, Schottky barrier diodes and p-n junctions.

Dark current is the result of imperfections or impurities in the silicon dioxide interface. When a CCD has been in operation for some time, the heat generated by the silicon dioxide interface causes electrons to escape and generate "dark current", which causes noise. This kind of noise is not temporally random, but spatially random.

2.3 Sensor Fusion Strategies Review

There are many multi-sensor applications ranging from military systems to processing plants and much research has been done related to sensor data fusion. Naturally, the problem of interpreting sensor measurements cannot be described using a common representation, for use in a general multi-sensor system.

The approach generally taken has been to pool the information using "weighted averaging" techniques, of varying degrees of complexity. The Independent Opinion Pool and the Independent Likelihood Pool are two such methods, described by Berger in [7], and an initial discussion of probabilistic data fusion can be found in [41][27][55]. Another common approach is to develop probabilistic models for the sensors, one example of this is the work by Cou'e et al. on the application of Bayes networks to the act of driving a car [18]. As shown in [2] and [16], probabilistic descriptions are extremely useful in building physical models by providing a way of objectively evaluating the sensors readings.

Such methods as Bayesian estimation [20], least squares estimation and especially Kalman filtering [1][60] have been widely described. Full surveys of the area are provided, for example, in [33] and, more recently in [10] and [2]. In the following

subsections, we first introduce the concept of a sensor model which is very important for data fusion approaches based on probability theory. Then we discuss the Bayesian approach, the Kalman filter approach, the Dempster-Shafter theory approach, the fuzzy logic-based approach and the voting-based approach along with their references.

2.3.1 Sensor Models

The *sensor model* is an essential element for data fusion algorithms based on probability theory. It models the interaction of the physical sensor with the environment by mapping the sensor's range measurement to a probability distribution of the presence of objects. The sensor model typically accounts for the uncertainty of the sensor in both range and angle. It considers sensor orientation and beam geometry, which may be wide (in the case of sonar) or narrow (for the laser and stereo vision camera) when generating the certainty value. The probability distribution also accounts for environment-sensor interaction hypotheses such as the independence of subsequent readings, the existence of only one object per observation and specular or diffuse reflections. Abundant research exists on sensor models for sonar, laser and stereo vision with various levels of complexity [11][36][6][43][62]. Based on this research, we applied some new approaches to simplify the computation of the sensor data.

A sensor model works as a converter from device-dependent sensor readings to device-independent sensor readings for the purpose of sensor data fusion. A robot is normally equipped with several kinds of homogeneous or heterogeneous sensors; in order to fuse sensor readings from heterogeneous sensors, these sensor readings have to be

converted to a universal representation. Thus, a sensor model works as a translator (See Figure 25).

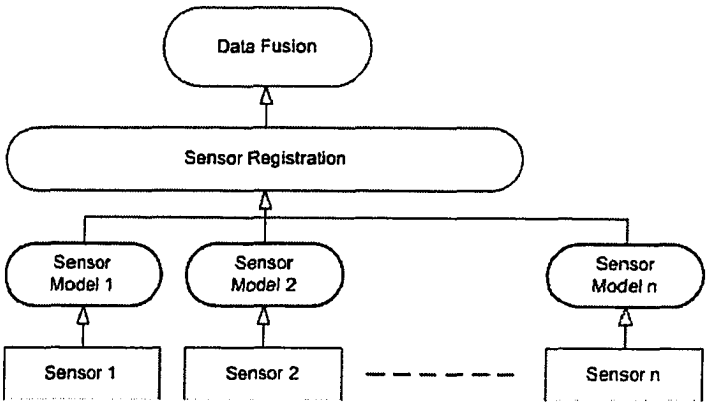


Figure 25: A sensor model works as translator between the physical sensors and the fusion algorithm.

In our sensor data fusion framework, there are different sensor models for different sensors. It is necessary to build a new sensor model and register it through a sensor model registration module for each new sensor added. The sensor data fusion module gathers converted sensor readings from all registered sensor models for the data fusion procedure. The sensor model registration module makes adding new sensor models without affecting whole system.

Virtually all state-of-the-art sensor models for robot mapping in the literature have one common feature: they are all probabilistic [34]. They all employ probabilistic models of the robot and its environment, and they all rely on probabilistic inference for turning sensor measurements into maps. Some authors make their probabilistic thinking very explicit, e.g. by providing mathematical derivations of their algorithms from probabilistic first principles [61]. Others use techniques that on the surface do not look specifically

probabilistic, but that can in fact be interpreted as probabilistic inferences under appropriate assumptions [45].

The popularity of probabilistic techniques in sensor fusion stems from the fact that robot sensor readings are characterized by uncertainty and sensor noise. Probabilistic algorithms approach the sensor noise problem by explicitly modeling different sources of noise and their effects on the measurements. In the evolution of sensor models, probabilistic models have emerged as the sole winner.

For probability-based sensor models, "Belief" and its standard deviation are the outputs of a sensor model, which is a general representation for the high-level data fusion. "Belief" is a probability value, ($b \in [0,1]$), which represents the probability that a specified position in an evidence grid map is occupied. The higher the value, the more certain it is that there is an obstacle at that position. The standard deviation σ of the "belief" b is also very important ($\sigma = D(b)$). The σ stands for the confidence of these readings, which has very close relationship with the characteristic of different sensors. Normally, the more accurate the sensor is, the smaller σ will be and vice-versa.

2.3.2 Bayesian Approach

Any model of a real phenomenon is inherently incomplete. Real-world phenomena are very complex and modeling all parameters involved has proven to be beyond the capabilities of contemporary computers and research [19]. The result is that a model and the phenomenon it represents changes over time. Furthermore, perception and control are inherently uncertain. This uncertainty arises from sensor limitations or from noise.

Rational reasoning with incomplete and uncertain information is a significant challenge. Bayesian Programming addresses this challenge by relying upon a well-established formal theory, probability theory [19].

The interpretation of Bayesian probability theory is very close to everyday language. Probability expresses how strongly someone believes in something. Belief is always subjective and depends on the background knowledge. Notation $P(A \mid B)$ means: the probability that A occurs if B occurs.

The Bayesian probability theory is based on a few simple rules. It is evident that a proposition and its negation are related. According to the sum rule:

$$P(A \mid B) + P(\overline{A} \mid B) = P(B) \qquad \textbf{(2)}$$

If one wishes to verify the truth of AB, one can first verify A and then verify B assuming A. Hence $P(AB)$ is evidently a function of $P(A)$ and $P(B \mid A)$. The product rule states that:

$$P(AB) = P(A)\, P(B \mid A). \qquad \textbf{(3)}$$

Bayes' rule (the basis of the Bayesian data fusion approach) can be derived from the product rule. It tells how the probabilities of explanations change, when A is observed.

$$P(B_i \mid A) = P(B_i)\, P(A \mid B_i) / P(A) \qquad \textbf{(4)}$$

$P(B_i)$ is the probability before the knowledge about A and it is called the prior probability of B_i. Correspondingly, $P(B_i \mid A)$ is called the posterior probability of B_i. One can see from Bayes' rule that the posterior probabilities of explanations B_i, which explain A better, are higher than the prior probabilities and vice versa.

Consider an example to illustrate the use of Bayes' rule. Let A represent the probability that sensor reading is 1 meter, B_1 represent the probability that there is an obstacle 1 meter away and B_2 represent the probability that there is no obstacle 1 meter away which is $\overline{B}_1$. Assuming that we have known the probabilities $P(A|B_1)$ (the probabilities of having a sensor reading of 1 meter when there is obstacle 1 meter away), $P(A|B_2)$ (the probabilities of having a sensor reading of 1 meter when there is no obstacle 1 meter away) and $P(B_1)$ (the probabilities of there being obstacle 1 meter away). We assign them the numerical values $P(A|B_1) = 0.95$, (i.e., when a sensor reading is 1 meter, the probability that the obstacle is at 1 meter is 95%) $P(A|B_2) = 0.05$ (i.e., when the sensor reading is 1 meter, the probability that the obstacle is not at 1 meter is 5% = 100% - 95%) and $P(B_1) = 0.1$ (i.e., assuming that the detection range of the sensor is 10 meters and one obstacle can only lay on 1,2,3 ... 10 meter grid points, then the possibility that the obstacle is 1 meter point away is 0.1 = 10%). The probability of having a sensor reading of 1 meter is $P(A) = P(A \mid B_1)\, P(B_1) + P(A \mid B_2)\, P(B_2) = 0.95 * 0.1 + 0.05 * 0.9 = 0.095 + 0.045 = 0.14$. The probability that the obstacle is at one meter is originally fairly small, only one in ten. If we have a one-meter sensor reading, the probability that the obstacle is at one meter increases. $P(B_1 \mid A) = P(B_1)\, P(A \mid B_1) / P(A) = 0.1 * 0.95 / 0.14 = 0.68.$

Bayes' rule tells how the beliefs in a hypothesis change when observations are made and how the beliefs in hypotheses are taken into account when making predictions based on them.

In the Bayesian sensor data fusion approach, the sensor assigns probabilities $P(Z)$ to the estimates of the parameters (e.g., the probability that the grid is occupied). These

probabilities are the certainty values and given the measurement R_A of the sensor A, this measurement is converted to the common representation $P(Z)$ by using the conditional probability density function $P(Z \mid R_A)$ which is normally called a *sensor model.* The sensor model represents the sensor properties $P(R \mid Z)$ through the Bayes's rule: $P(Z \mid R) = P(R \mid Z)\, P(Z) / P(R)$. In Figure 26, a sensor model used in Bayesian approach is shown as $P(Z_{d,\alpha} \mid R_A)$ (Assuming this sensor is a sonar and d represents for distance and α represents for angle position of a obstacle).

The sensor models shown in Figure 26, Figure 27 and Figure 28 are two-dimensional probability distributions. The M axis represents the distance distribution, where M represents the measured distance and numbers on the M axis represents the percentage values of the measured distance (e.g., 1.05 means 105% measured distance). The Degree axis represents the angle distribution, where probability distribution spreads over all emitting angles.

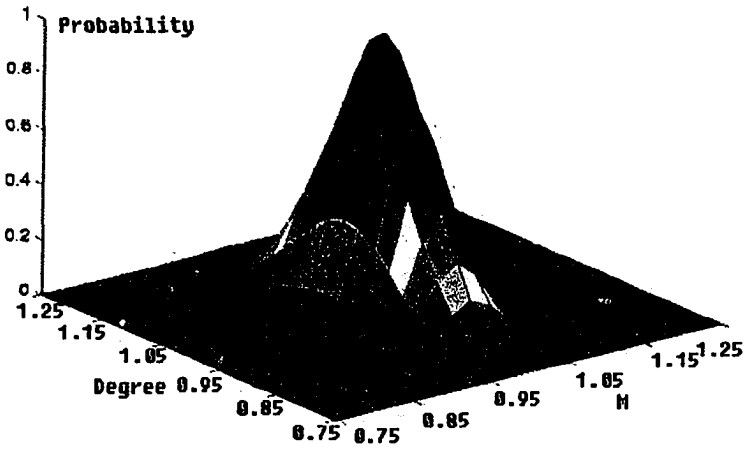


Figure 26: The sensor model for sensor A in the Bayesian approach.

Now suppose another sensor B measures the same obstacle and converts its measurements as illustrated in Figure 27. This sensor has converted its measurements using another sensor model $P'(Z_{d,\alpha} \mid R_B)$ The difference in these sensor models reflects the fact that these sensors have different error characteristics. In this example, the first sensor would typically be an acoustic sensor while the other sensor would typically be an infrared proximity sensor.

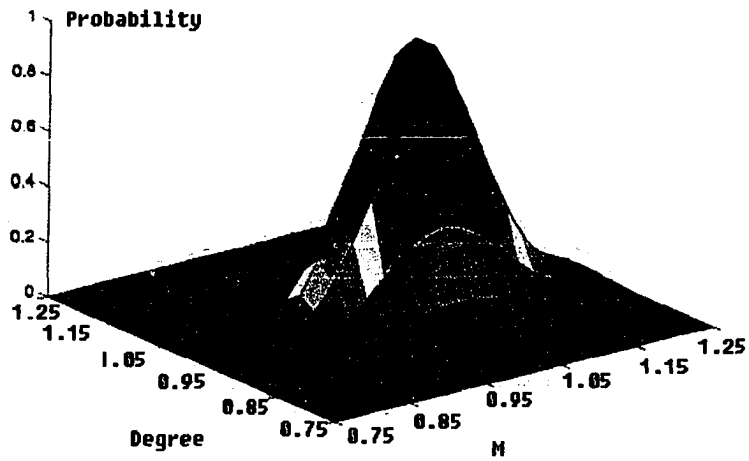


Figure 27: The sensor model for sensor B in the Bayesian approach.

Fusion of the converted measurements of both sensors can now be performed by taking the product of the two probability density functions. This way we obtain a joint probability density function

$$P_{joint}(Z \mid R_A R_B) = P(Z \mid R_A)\, P(Z \mid R_B) / P(Z) \qquad (5)$$

This can be derived from

$$P(Z, R_A \mid R_B) = P(Z \mid R_A, R_B)\, P(R_A \mid R_B) = P(R_A \mid Z, R_B) P(Z, R_B) \qquad (6)$$

$$P(Z \mid R_A, R_B) = P(R_A \mid Z, R_B)\, P(Z \mid R_B) / P(R_A \mid R_B) \qquad (7)$$

Assuming that the measurements of both sensors R_A and R_B are *independent*[5], which gives: $P(R_A|\ Z,\ R_B) = P(R_A|\ Z)$ and $P(R_A|\ R_B) = P(R_A)$. So $P(Z \mid R_A,\ R_B) = P(R_A \mid Z)\ P(Z \mid R_B) / P(R_A)$. Applying Bayes' rule again: $P(R_A \mid Z) = P(\ Z|\ R_A)P(R_A)$ results in the $P_{joint}(Z \mid R_A R_B)$ (shown in Figure 28).

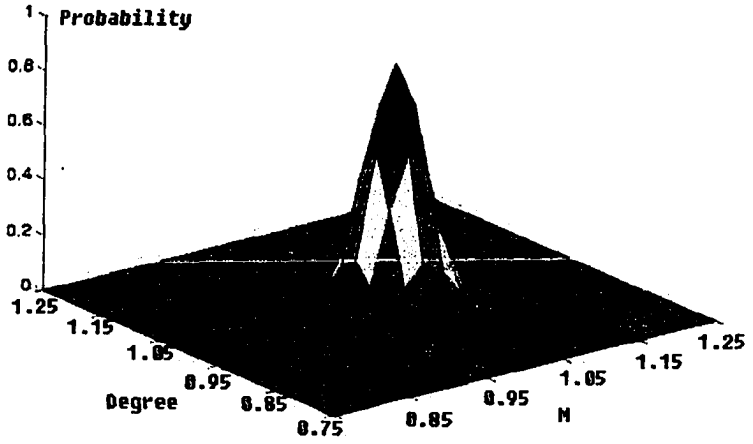


Figure 28: The sensor model for the fusion result.

2.3.3 Kalman Filter Approach

The Kalman filter, explained in detail in [14], is an optimal linear estimator[6] based on a recursive process. Various applications use Kalman filters, and it is particularly effective when applied to sensor fusion. It recursively evaluates an optimal estimate of the state of a linear system. In our case, the state of the system is the position and location

---

[5] The occurring of event A does not affect the probability of event B happening.

the state of a linear system. In our case, the state of the system is the position and location of the robot. During each recursive stage of the filter, a new estimate of the state (i.e., pose) is evaluated, using the new information (i.e., sensor data) available to the filter. The Kalman filter process consists of two sub-processes, "prediction" (also termed "time update") and "correction" (also termed "measurement update"), which repeat recursively at a time step of d_t. (In order to avoid confusion, we use "prediction" and "correction" instead of "time update" and "measurement update" for the reminder of this thesis.) One of the most important advantages of the Kalman filter data fusion approach is that it can predict a future state. After making a prediction, the Kalman filter will attempt to make the correction based on the latest reading (if a sensor measurement is available). If there is no new sensor readings, the Kalman filter ignores the correction procedure. The prediction and correction procedure can minimize the influence of erroneous sensor readings.

We first introduce the state change model and the measurement model. Equation 8 defines the state change mode and we can see that system state X in time t are related with system state X at time *t-1* and system control input U at time *t-1* where A and B are parameter matrices. For example, the speed of a car at time t depends on the its speed at time *t-1* and whether you press the gas paddle. w_t is noise. Equation 9 defines the measurement model, the sensor reading Z at time t is related to system state X at time t where H is parameter matrix and noise v.

$$X(t) = AX(t-1) + BU(t-1) + w_t \quad \textbf{(8)}$$

$$Z(t) = HX(t) + v_t \quad \textbf{(9)}$$

One easy way to explain the Kalman filter is to assume that it incorporates just two sets of sensor measurements. There are four important variables in the Kalman filter approach. The first is the prior state of the system $X(t-1)$, it represents the value of fused data at time $t-1$. The second is post state of the system $X(t)$, it represents the value of the fused data at time t. The third is prior state estimate $X_{prior}(t)$, it is the system state prediction for time t at time $t-1$. The last one is new measurement $Z(t)$ at time t.

The concept on the Kalman filter is that at time t, first we make a prediction of the fusion value at time t based on its old value on time $t-1$. We make a new sensor reading $Z(t)$ at time t and rectify our prediction using $Z(t)$ to generate the fusion value at time t. This procedure will be iterated.

In the prediction process, a prior state estimate $X_{prior}(t)$ is computed based on the previous state estimate $X(t-1)$ (e.g., the probability of a grid is occupied at time t based on its value at time $t-1$). Normally they are independent, but when there are moving obstacles, the state t+1 could have relationship with state t. Then, in the correction process, this prior estimate is blended with the probability value converted by direct measurements through a sensor model, thus obtaining the new updated state estimate $X(t)$. To understand how the Kalman filter fuses sensor readings from different sensors, we give here a detailed explanation of these two sub processes.

Prediction

As described by Equations 10 and 11, the optimal estimate from the previous iteration, noted $X(t-1)$, is projected in time through the state transition matrix A. Matrix A

defines the relationship between system state at time t and time $t+1$ (For grid map building, matrix A defines the probability that a grid is occupied at time $t+1$ will change, providing the probability of that grid at time t). The control input $U(t)$ (from actuators) are fed to the system through matrix B, which defines the relationship between system state and control input.

$$X_{prior}(t) = A \cdot X(t-1) + B \cdot U(t) \quad \textbf{(10)}$$

$$P_{prior}(t) = A \cdot P(t-1) \cdot A^T + Q \quad \textbf{(11)}$$

In Equation 10, the state X is the linear position of the robot and U is the input from the driving motor. Matrix A reflects the relationship between the current and previous states of the system, and matrix B reflects the relationship between system states and the system's control inputs.

Correction

This process is formalized by equations 12 to 14, where direct noisy state measurements $Z(t)$ coming from sensors are compared with the prior state estimate $X_{prior}(t)$. This comparison yields a correction that can be applied to the prior estimate in order to obtain the new estimate $X(t)$. Matrix H relates the measurements to the state.

$$K = P_{prior}(t) \cdot H^T \cdot (H \cdot P_{prior}(t) \cdot H^T + R)^{-1} \quad \textbf{(12)}$$

$$X(t) = X_{prior}(t) + K^T \cdot (Z(t) - H \cdot X_{prior}(t)) \quad \textbf{(13)}$$

$$P(t) = (I - K \cdot H) \cdot P_{prior}(t) \quad \textbf{(14)}$$

The importance of each estimate (the prior estimate $X_{prior}(t)$ and the measurement estimate $Z(t)$) is determined by the Kalman gain K. This gain is in turn determined by matrices Q and R, which respectively represent the process noise variance/covariance (from indirect measurements) and the measurement noise variance/covariance (from direct measurements). The Kalman Gain K takes a value between 0 and 1: 0 represents the use of the prediction only, while 1 represents the use of direct measurements only. The error covariance matrix P, which is predicted in Equation 11, is corrected in Equation 14 to reflect the correction process.

Continuing with the previous example, $Z(t)$ in Equation 13 represents a measurement coming from a range sensor, and H is equal to 1 (the identity matrix) if there is no direct correspondence between Z and X.

The Kalman filter is popular in sensor fusion applications because its formulation makes it easily adaptable to sensor fusion. In order to help develop a better understanding for the operation and the capability of the Kalman filter, we present here a simple example for its usage.

Let us attempt to estimate a random constant: the probability that a grid cell is occupied, for example. The sensor readings from different sensors are distances and we need sensor models, which are similar to those used in Bayesian approach, to convert those sensor readings into a general format: the probability value. In this example, our system is modeled as a linear system.

Matrix A defines the system state relationship between time $t+1$ and t. System states could change themselves (e.g., if an object is moving at a constant speed). If the

system states that we are interested in are $x = \begin{bmatrix} Pos \\ Vel \end{bmatrix}$, then $x_{t+1} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} x_t + noise$, assuming a sample rate = 10Hz. Every time the system state is updated, $Pos(t+1)=Pos(t)+0.1Vel$, where the unit of Vel is m/s. If the object is not moving, then $x_{t+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_t + noise$, since the system state at time $t+1$ is the same as time t. In our case, the prediction mode is deterministic. The system state is the probability value in a grid cell and it is not changing in time if there is no new incoming sensor data. Therefore, we set matrix A to I (the identity matrix).

Matrix B defines the relationship between the system state and control inputs (external force which may change the system state, such as forces or energy). If there are control inputs, the system state at time $t+1$ may be related to it, (e.g, the moving object has a propeller and it is speeding up). The system state model may change to $x_{t+1} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 4 \\ 2 \end{bmatrix} u_t + noise$. Let us assume $Vel = 2u$ and Pos = 2^2u. The relationship between velocity and force is proportional and the relationship between displacement and force is exponential. Since there is no control input, matrix B of Equation 10 is set to 0. The sensor readings represent the state directly, so H is set to 1 (the identity matrix). (Notice that we dropped the subscript t in several places because these respective parameters remain constant in our sample model).

Our prediction equations become:

$$\begin{aligned} x_{prior}(t) &= x(t) \\ P_{prior}(t) &= P(t-1) + Q \end{aligned} \qquad \textbf{(15)}$$

and our correction equations become:

$$K(t) = \frac{P_{prior}(t)}{P_{prior}(t) + R}$$
$$x(t) = x_{prior}(t) + K_t(z_t - x_{prior}(t)) \quad \textbf{(16)}$$

Presuming a very small process variance, let Q = 0.0001. We could certainly let Q = 0, but assuming a small but non-zero value gives us more flexibility in "tuning" the filter as we will demonstrate below. Let us assume that from experience we know that the true value of the random constant has a standard normal probability distribution, so we will "seed" our filter with the guess that the constant is 0.5. In other words, before starting we let $x_{prior}(t\text{-}1) = 0.5$ (a probability of 0.5 means that the grid could be either occupied or empty).

Similarly, we need to choose an initial value for P_{t-1}, which is P_0. If we are absolutely certain that the initial state estimate $X_0 = 0$ is correct, we would set P_0 to any non-zero number, since 0 will cause the filter to always believe the next state is same as previous state. However the initial P is not critical, since eventually filter will converge. We start our filter with $P_0 = 1$.

Figure 29 shows a typical Kalman filter data fusion process, where a Kalman filter runs 50 iterations. There are 50 sensor readings which are *Z(t)* (see cross signs in Figure 29). The straight solid line is the true value of the system state, whose exact value we can never get, and the zigzag line is the system state value of the Kalman filter *X(t)*, which approaches the true value over time. Although the sensor readings from different sensors are very noisy, the fusion result is stable and converges. When doing heterogeneous

multi-sensor data fusion, although sensor readings from low cost sensors are noisier, they still provide additional information about obstacles and improve the fusion result.

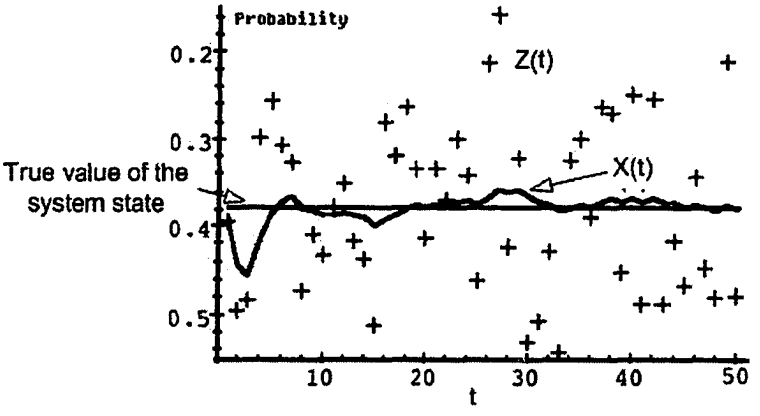


Figure 29: An example of the Kalman filter sensor data fusion.

2.3.4 Dempster-Shafer Theory Approach

In a Dempster-Shafer reasoning system, all the possible context facts (or events) that are of the same kind but are mutually exclusive are counted as part of "the frame of discernment Θ" [62].

The Dempster-Shafer decision theory is a generalized Bayesian theory, which is a canonical method to deal with statistical problems. It allows the distribution of support for a proposition (e.g., this grid is occupied), and also for the union of propositions that include it (e.g., this is likely either empty or occupied, which is unknown). Compared with Bayesian theory, the capability of the Dempster-Shafer theory to assign uncertainty or ignorance to propositions is a very powerful tool to deal with a large range of problems that otherwise would be difficult to handle. For example, it makes it very easy to use the processed information from the sensors. Once processed, the sensor data will convey

information like, "it is highly probable (e.g. with a confidence of 0.9) that this is not landmark A, although we do not know which this landmark is".

For example, suppose we try to build a grid map for the robot's surrounding environment (this experiment is introduced by Zou etc in detail in [66]). For each grid cell, there are three possible states: Occupied, Empty and Unknown, which is represented as θ.

$$\theta = \{O, E, U\} \tag{17}$$

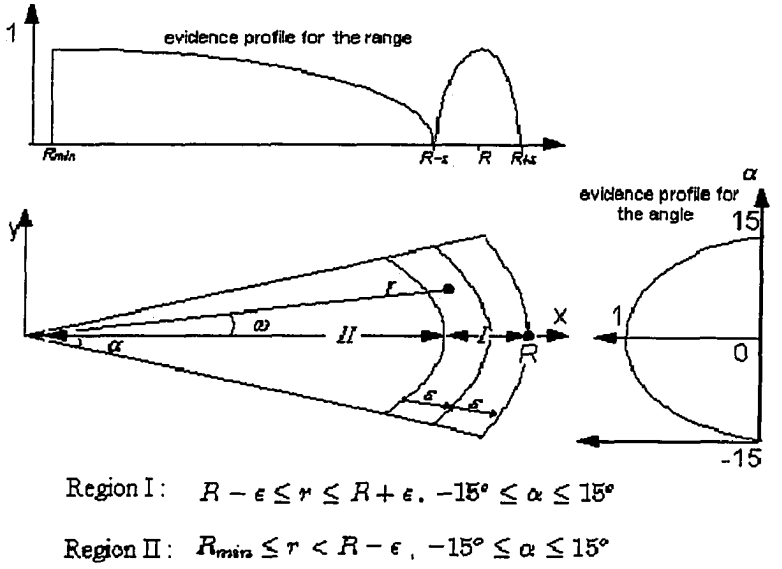


Figure 30: The sensor model for sonar in the Dempster-Shafer theory approach (ε is distance error rate α is emitting angle = 30°).

Each sensor S_i will contribute its observations by assigning values to elements of θ that reflect its beliefs. This assignment function is called the "probability mass function" of the sensor S_i (which is very similar to the sensor model used in the Bayesian approach and the Kalman filter approach) denoted by M_i. For example, according to observations

by sensor S_i (i.e., sonar), the probability that the grid is occupied is indicated by a confidence value, as shown in Figure 30 and Equations 18 to 23.

The emitting region of the sonar is divided into two regions. Region I is a sector band around the distance reading. The width of the band is $2 \times L \times \varepsilon$, where L is the distance reading and ε is the distance error rate. Region II is the sector region within the inner border of region I and the sonar transducer. It can be treat as a kind of sensor model, it converts sensor readings to probability values and put into grid cells depending on the region the grid cell to be. Figure 30 shows the definition of two regions and the distance to probability conversion functions of them.

Region I, where $R - \varepsilon < r < R + \varepsilon$:

$$m(\{Occupied\}) = \frac{(\frac{\alpha - \omega}{\alpha})^2 + (\frac{\varepsilon - |R - r|}{\varepsilon})^2}{2} \quad \textbf{(18)}$$

$$m(\{Empty\}) = 0 \quad \textbf{(19)}$$

$$m(\{Unknown\}) = 1.00 - m(\{Occupied\}) \quad \textbf{(20)}$$

Region II, where $R_{min} < r < R - \varepsilon$:

$$m(\{Occupied\}) = 0 \quad \textbf{(21)}$$

$$m(\{Empty\}) = \frac{(\frac{\alpha - \omega}{\alpha})^2 + (\frac{R - \varepsilon - r}{R - \varepsilon})^2}{2} \quad \textbf{(22)}$$

$$m(\{Unknown\}) = 1.00 - m(\{Empty\}) \quad \textbf{(23)}$$

Here, R is the range reading from the sonar. (r, ω) is the coordinate of a point inside the sonar cone. ε is the range error and it distributes the evidence in Region I. α is the half open beam angle of the sonar cone, which is normally 15°.

After having sensor models, Dempster's rule of combination is very straightforward.

$$m_n(t)=\frac{1}{1-\mathrm{P}}\sum_{t=v\cap t-1;t\neq\phi} m_S(v)m_O(t-1) \quad (24)$$

$$\mathrm{P}=\sum_{v\cap t-1=\phi} m_S(v)m_O(t-1) \quad (25)$$

Here, the new evidence $m_n(t)$ is updated by the two evidence sources m_s from sensors and m_o from the old existing evidence. The term P in Equation 24 measures the extent of conflict between the different sources of evidence. P is defined in Equation 25. The $m_n(t)$ will become $m_o(t-1)$ at the next iteration.

2.3.5 Fuzzy Logic Approach

Fuzzy logic accommodates imprecise states or variables. It provides tools to deal with context information that is not easily separated into discrete segments and that is difficult to model with conventional mathematical or rule-based paradigms [51][37]. One example of such information is the status of a grid cell in a grid map, which is commonly referred to using descriptions like, "Occupied" or "Empty". There are no rigid boundaries between "Occupied" and "Empty" with which to make decisions regarding the occupancy of a grid cell (what if an obstacle occupied half the grid cell?).

There are three primary elements in a fuzzy logic system: (a) fuzzy sets, (b) membership functions and (c) production rules.

Fuzzy sets consist of the imprecisely labeled groups of input and output variables that characterize the fuzzy system; "Occupied", "Empty" and "Unknown" in the above grid map example constitute a fuzzy set corresponding to the status of a grid.

Each fuzzy set has an associated membership function to provide a representation of its boundaries. An element in a fuzzy set is assigned a membership value between 0 and 1, with 0 indicating the variable is not in that status and 1 indicating it is completely in that status. An intermediate membership means a variable may belong to several elements in the fuzzy set, for example, a probability of 0.7 may be assigned as 0.15 "Empty", 0.25 "Unknown" and 0.6 "Occupied". So we can also employ the sensor models used in previously introduced approaches.

Production rules specify logic inference processes in the form of IF-THEN statements, which are often referred to as *fuzzy associative memory*. The output fuzzy set is usually defuzzified to convert the fuzzy values, represented by the logical products and consequent membership functions, into a fixed and discrete output that can be used by target applications.

For example in a multi-sensor data fusion system, we can define a fuzzy rule as follows for each grid A:

IF (the sensor reading from a Laser range finder: $P_{Laser} > 0.7$ **AND** the reading from a sonar $P_{sonar} > 0.2$) **THEN** set "Occupied" = 0.8, "Empty" = 0.05 and "Unknown" = 0.1 (This rule shows that we rely more on the most precise sensors).

IF (the sensor reading from a Laser range finder: $P_{Laser} < 0.2$ **AND** the reading from a sonar $P_{sonar} > 0.8$) **THEN** set "Occupied" = 0.6, "Empty" = 0.1 and "Unknown" = 0.3 (This rule shows that we fuse readings from imprecise sensors to improve the result).

We can define comprehensive rules for different scenarios for a robot and the boundaries between sets of values are not sharply defined when the events occur only partially, or the specific mathematical equations that govern a process are not known. With the capability to deal with this kind of information, and with its cheap computation to solve very complicated problems, the fuzzy logic method is expected to develop extensively in various context-aware computing applications.

2.3.6 Voting Approach

Voting methods combine detection and classification information from multiple sensors by treating the input from each sensor as a vote; majority, plurality, or decision tree rules are then used to determine a consensus among the votes from the sensors. The generally used voting architecture is a boolean combination of outputs from multiple sensors, although additional discrimination can be introduced by weighting specified votes from each sensor [30].

The principle underlying voting fusion is the combination of logical values representing sensor confidence levels, which are based on predetermined detection probabilities for an object. For a proposition H_k representing the event that grid cell k is occupied, the inputs to voting fusion are the detection probabilities $P_i(H_k)$ and $P_j(H_k)$ of sensor S_i and S_j, respectively, and their false alarm probability $P_{fai}(H_k)$ and $P_{faj}(H_k)$,

respectively; the output is the detection probability $P(H_k)$ and the false alarm probability $P_{fa}(H_k)$. $P(H_k)$ and $P_{fa}(H_k)$ in each iteration is calculated using Equations 26 and 27 accordingly.

$$P(H_k) = P_i(H_k) + P_j(H_k) - P_{i \cap j}(H_k) \quad \textbf{(26)}$$

$$P_{fa}(H_k) = P_{fai}(H_k) + P_{faj}(H_k) - P_{fai \cap j}(H_k) \quad \textbf{(27)}$$

Assuming that a robot is equipped with a laser range finder, a sonar and a stereo camera and the robot is making a grid map. Based on the accuracy of different sensors, we set different weights for them. For example, we set $W_{laser} = 6$ for the laser range finder, $W_{stereo} = 4$ for the stereo camera and $W_{sonar} = 1$ for the sonar. The weight of the fusion result is the sum of the weights of all sensors which give positive sensor readings (e.g., if all three sensors agree that a grid cell is occupied, the weight of the result will be $11 = 6 + 4 + 1$. If the laser range finder and the sonar confirm that the grid cell is occupied, but the stereo camera does not agree then the weight of the fusion result will be $3 = 6 + 1 - 4$). The data fusion procedure is very straightforward which is defined in Equations 28 to 29.

$$P(k+1) = (W_{laser}R_{laser} + W_{stereo}R_{stereo} + W_{sonar}R_{sonar} + W_{P(k)}P(k)) / (W_{laser} + W_{stereo} + W_{sonar} + WP(k)) \quad \textbf{(28)}$$

$$W(k+1) = W_{laser} + W_{stereo} + W_{sonar} + W_{P(k)} \quad \textbf{(29)}$$

Voting methods greatly simplify the sensor fusion process, and can provide a prediction of object detection probability as well as a false alarm probability. However, voting fusion is more suitable for dealing with "yes/no" problems like the classical inference method, because the granularity of its reasoning is usually not sufficient for the

purposes of multiple state context discrimination. For example, in the case of grid map construction, the status of each grid cell can only be either occupied or empty, which is useful for robot navigation, due to its simplicity and ease of processing. However, if knowledge regarding sensors, tasks or the environment is available, then model-based methods perform better.

2.4 Storage Strategies Review

Maps are extremely important in the navigation and exploration process since robots "require the ability to recover robust spatial models of the surrounding world from sensory information" [23]. Based on the information used to represent the environment, maps are normally classified as either vector-based or grid-based. The different types of maps vary in terms of data structures and in their resolutions. Some maps are more suitable than others are for the path planning approaches. Generally, however, most of them are interchangeable.

Maps may be either local maps or global maps. Local maps define a finite area around the robot. The boundary of a local map may be walls of a room or a given radius around the robot. Global maps normally cover a larger region relative to the region to be explored by the robot. For example, the extent of a global map could consist of several floors of an office building. Global maps are used to determine a general path to a final goal or destination. Local maps include detailed information, pertaining to any detected nearby obstacles which are then used to continually update the global map [17][39].

Reactive navigation can rely solely on local maps. Planned navigation requires global mapping, at least to the extent of memorized local maps.

The major difference among map types is the method used to handle sensor noise and uncertainty. The detected position of an obstacle may also differ from the actual position due to sensor resolution limitations. This is especially significant when using low-resolution sensors, such as ultrasonic sensors. Internal robot sensors, such as encoders, are also prone to error. Mapping often depends on sensor data that is relative to the location of the robot. The location of the robot is, in turn, relative to the coordinate system of its global map. Other maps rely on multiple-sensor data sets that must align with the relative robot movements over a period of time. Errors in robot odometry can quickly cause misalignment during coordinate transformation, and should be considered carefully during map selection. Maps such as certainty grids a form of grid-based mapping inherently store average sensor data, and thus are robust against erroneous data. On the other hand, obstacle maps using vector representations rely upon accurate sensor data.

2.4.1 Vector-Based Maps

There are three major types of vector-based maps: path maps, free space maps and object-based maps. The vector-based maps use vectors to represent the obstacles. In the following subsections, we introduce these three types of maps.

Path Map

Path maps contain lists of paths or movements based on human programming or teaching methods. Path maps are created offline and often contain room layouts and

landmark locations (see Figure 31). Teaching a robot involves physically moving it and periodically recording motions and stop points. These maps are stored internally in the robot and are used to guide the robot with the aid of external beacons or guide wires. Path maps are used extensively in industrial applications where there is little variation in the robot's paths or in the application that it is expected to perform [39]. Path maps are typically of little use to fully autonomous robots.

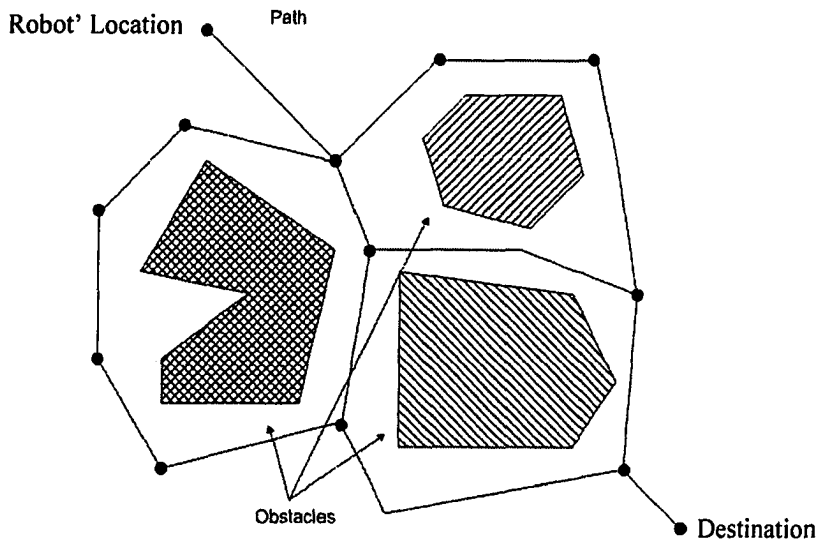


Figure 31: An example of a path map.

Free Space Map

Free space maps are concerned with representing the navigable space between obstacles rather than the obstacles themselves. As the robot proceeds through an unknown environment, data concerning the extent of free space is collected. The locations of occupied areas, or obstacles, are recorded in a spatial graph [39]. Once sensor data has been collected, the free space may be represented in several ways. Some research presents a free space method using generalized cones [12]. A generalized cone is

one formed by sweeping a specified cross-section along a spline. In this case, the splines are straight lines, equidistant between two given objects. The swept cross-section is a line perpendicular to the spline and whose end points are defined by the outer boundaries of the objects (Figure 32). Each of the generalized cones forms a "freeway" such that the center spline represents the safest path in each freeway. Overlapping cones form possible robot routes and their intersections.

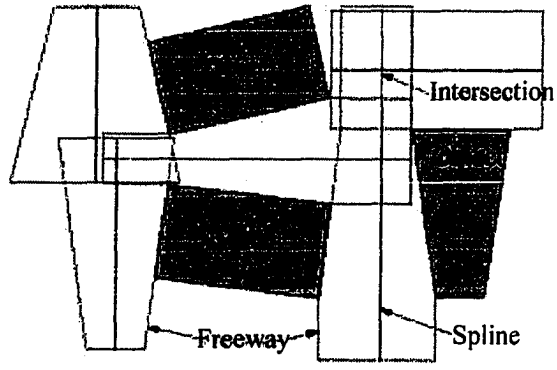


Figure 32: An example of the generalized cone free space map.

Object Map

Object-based maps are concerned with identifying and storing the location of obstacles in the environment. The free space is determined by implication (e.g., areas without obstacles). Objects are typically recorded as sets of vertices relative to a global reference frame. Alternatively, the shape, position and orientation of each object can be specified in a linked list of x, y, and θ coordinates [13][39]. Both approaches are shown in Figure 33. Object-based maps have the advantage of modeling a given environment with a small data set, relative to other map types. However, the ability to produce accurate object models is highly dependent upon precise, detailed sensor data, thus

limiting the practical applications of this map type [63]. Object maps are most effective in situations where the environment is well known.

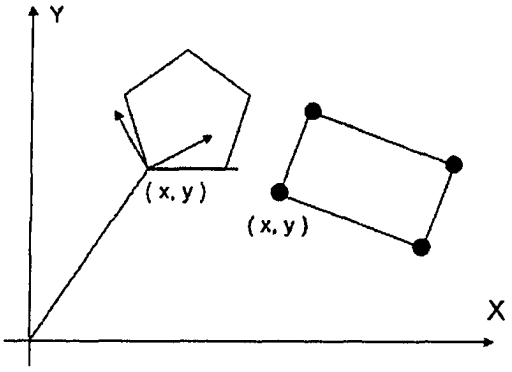


Figure 33: An example of the object map.

2.4.2 Grid-Based Maps

None of the previous mapping approaches completely represents the environment. Free space maps ignore objects; object-based maps are only concerned with objects, and the free space is implicit. Grid maps include data concerning both objects and free space.

Normal Grid Map

A grid map is one in which "the configuration of a rigid object is a set of independent parameters that characterize the position of every point of an object" [13]. The set of parameters are typically (x, y) coordinates that relate to the equations of motion for the robot. Thus, the configuration space specifies the boundary between free space and object space during robot movement.

A common approach for grid-based mapping is the *evidence grid map* (also called the occupancy grid map). Evidence grid maps superimpose a geometric structure over the environment. Grids may be tessellated into finite areas or points. Each individual grid

"cell" or point represents a finite amount of space within the environment [63][39]. Grid cells may contain data concerning their occupancy, danger, reachability, observability and reflectance.

Evidence grids combine sensor data for each finite grid cell. A threshold is then applied to each grid cell based on heuristic data. The cell is then determined to be empty, occupied, or to be unknown (see Figure 34 (b)). This trinary approach locates both free space and objects. The disadvantage of the evidence grid map is that it can only represent a finite area of the world (due to the computer's storage and process capability). An obstacle may only partially occupy a cell, but the cell will be listed as fully occupied [23]. This problem can be overcome by using certainty, which is a probability of the grid cell being occupied, or by a quadtree grid map.

Quadtree Grid Map

A quadtree grid map [48][49] is based on the successive subdivision of region into four equally sized quadrants. A region is recursively subdivided until either a sub-region free of obstacles is found or the smallest grid cell is reached. Quadtrees allow efficient partitioning of the environment since single cells can be used to encode large empty regions. Figure 34 (c) shows an example of a quadtree grid map.

Grid-based mapping has several advantages. Grids are good for unknown environments since several sensor types can be easily combined for input to a grid-based map since they do not need assumptions to do the feature extraction or the high level map abstraction, which are normally required for vector-based maps. Probabilistic models are

often used to compensate for variations in accuracy and noise among sensors. Grids can be continually updated to build a more robust world model. Certainty and quadtree grids specifically have the advantage of averaging or blurring together multiple data sets during integration. This helps to eliminate errors caused by a robot's dead reckoning error. The quadtree data structure also lends to the easier integration of local maps into larger global map [63].

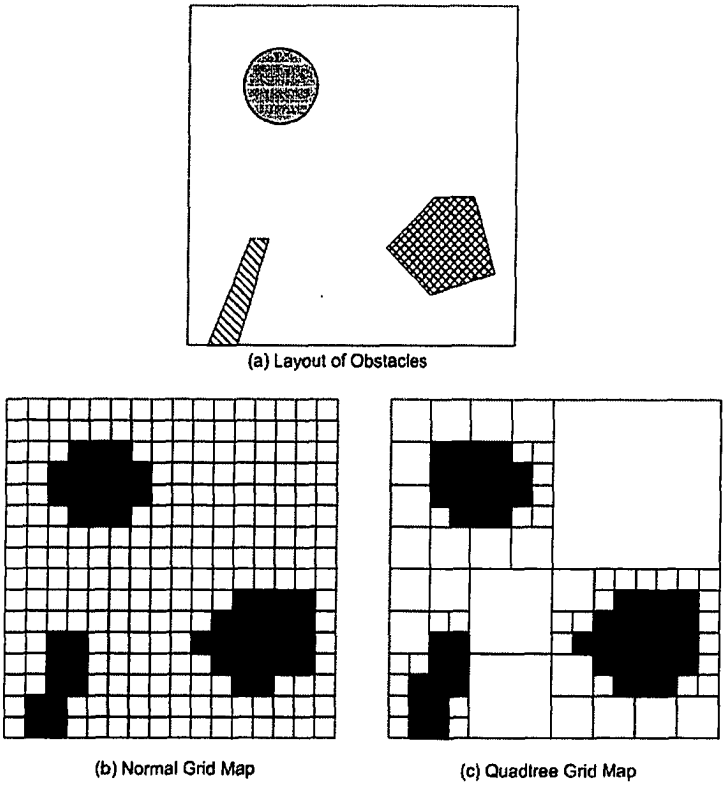


Figure 34: Examples of the normal grid map and the quadtree grid map.

Grids maps do have several disadvantages. As mention previously, they generate a finite representation of the world. The representation may also be discrete, the smallest region is a grid, which can only be occupied or empty, in the case of evidence grids. Thus, round objects may be represented as squares or occupy more grid space than they

do in reality. This can be partially overcome by using quadtree grids or different grid cell geometries, such as hexagons or triangles. Building a grid-based map is also typically computationally intensive. The data structure is highly dependent upon grid resolution and field of view; again, this may be partially overcome by using a quadtree approach, especially for sparse environment [23].

Chapter 3 Kalman Filter-based Sensor Data Fusion

Our research with the K2A robot started in September 2002 with the arrival of a Cybermotion® K2A robot, borrowed from NRC-CNRC[7] (See Appendix 1). In this chapter, we describe all approaches that we applied to different sensors and our data fusion strategy. At the end of this chapter, we introduce our implementation: the software framework for the data fusion system.

3.1 The Methodology for Obtaining Sensor Measurement

We used three different kinds of sensors for indoor mapping and exploration, which each provides different area coverage and has different features. Before we introduce our multi-sensor data fusion strategy, we first introduce different approaches, which we applied to each type of sensor to improve their reading quality.

3.1.1 Methodology for IR Proximity Sensor Array Problems

As indicated in the IR sensor manual (also verified by our experiments), we found that there exists a valid reading range and two noisy or invalid reading ranges (see Figure 35). In the valid reading range, the IR sensors give reliable sensor readings with relatively

[7] National Research Council of Canada.

low error. In the noisy reading ranges, the sensor reading becomes far less accurate. As can be seen in the figure, the sensor's valid range is between 10cm and 150cm.

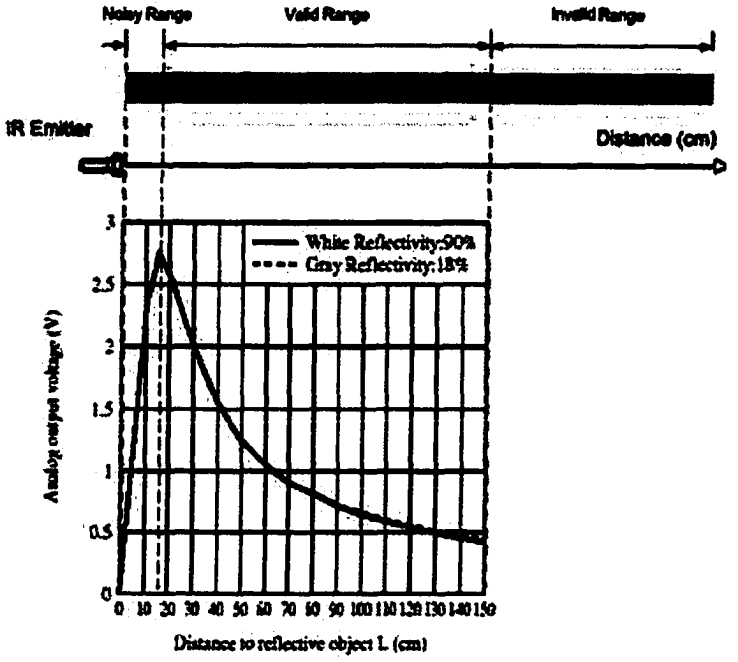


Figure 35: The valid reading ranges and noisy reading ranges for the Sharp IR proximity sensor (values shown in figure are typical expected values, see [52]).

From the distance to voltage transformation graph in Figure 35, The reason for the two noisy ranges can be easily seen. In the first noisy range, the shape of transformation graph is very sharp, which means that relatively small changes in distance will cause relatively big changes in voltage. This means that small disturbances will cause noisy readings. In this region beyond 150cm, the shape of transformation graph is very flat, which means relatively big changes in distance will only generate relatively small changes in voltage. In this range, electronic noise will present noisy readings.

The Non-linear Voltage Output Problem

One drawback of the Sharp® sensors is their non-linear response (see Figure 36). In other words, a linear change in output voltage does not always indicate an equally linear change in distance. To obtain distance values corresponding to a particular output voltage, a user must find the appropriate conversion function. In general, the response curve is different for different sensors. Therefore, in order to get an accurate response curve we tested each IR proximity sensor individually and computed its response curve.

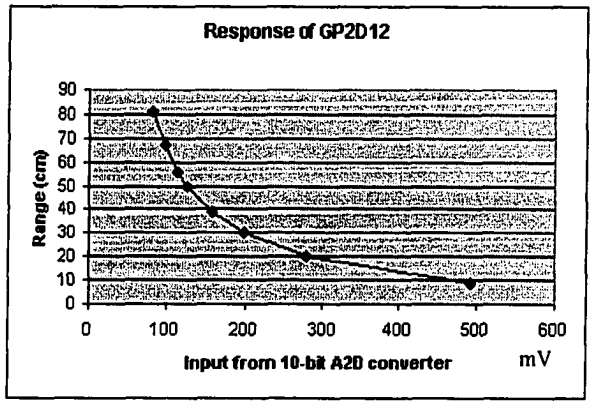


Figure 36: The distance to voltage conversion curve for the valid range (actual conversion curve from experimentation.

In order to obtain accurate sensor data, we applied hardware and software solutions to overcome the non-linear voltage output problem of Sharp IR sensor. Since there are two noisy reading ranges and one valid reading range, the robot, if only using the IR sensor, cannot tell whether an obstacle is in the noisy range. However, with additional information provided by other sensors, the robot can determine the quality of the IR sensor reading. When other sensors detect an obstacle that is within the valid range of the

IR sensor (i.e., 15cm to 150cm), the IR sensor readings are enabled (or considered valid) for sensor fusion. Otherwise, the IR sensor readings are presumed to be invalid.

The K2A is equipped with bumper sensors that do not allow obstacles to get closer than 20cm to the IR sensors. Sensor readings from the robot's stereo camera and sonar sensors can be used to determine whether the obstacle is within 150cm of the robot.

Using sophisticated mathematics programs to generate a curve fit, we can solve the problem of linearizing the non-linear distance to voltage conversion. Such generated functions are quite accurate but usually require floating point math and a good math library in order to implement them. This is computationally expensive when the robot need to obtain sensor readings in real-time.

Another approach is to use a piecewise-linear approximation to convert the output voltage to a range value. This involves breaking up the response function into small straight lines and doing a separate approximation for each line. Straight-line approximations are simple to compute and can be implemented with high accuracy by using integer math. The disadvantage is that they require more storage space and more time to calculate than the simple approximation.

Ideally, it would be nice to have a single approximation function that works well with integer math. Fortunately, there are some simple calculations that can "linearize" the response of the Sharp sensors.

According to page 10 of the Sharp Device Specification for Distance Measuring Sensor [52], a plot of the following relation: V vs. 1 / (R + 0.42) (where V is voltage and R is range), is a very straight line (See Figure 37). The division operation acts as a

linearizing function that turns the nonlinear curve into a linear plot. This observation is the key to finding a simple approximation function.

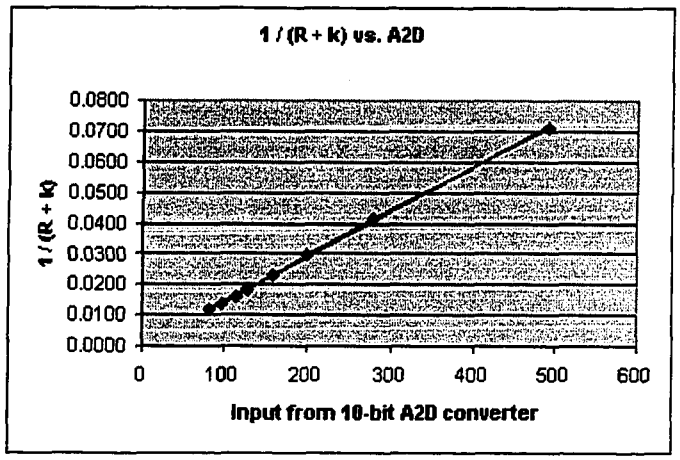


Figure 37: The plot of V and R + 0.42. Y-axis has no unit.

The value of the constant (e.g., 0.42) used in the linearization function, which is the slope of the function, depends on the features each IR sensor and calibration data. The value of 0.42 (the slope of the function shown in Figure 37) is example data given by the user manual, which is not suitable for our Sharp IR sensors. We needed to find the proper parameters for our IR sensors in the array ourselves, to be used for the linearization functions for our sensor array. The most important step in obtaining a good voltage-to-range function is to find a constant k that linearizes the data.

We found a straight-line approximation that relates the voltage to the linearizing function. This involves finding suitable m and b constants for the line equation:

$$y = mx + b \tag{30}$$

In this case, y is the linearized distance. Substituting the linearizing function for y and substituting V for x yields:

$$1/(R+k) = mV + 1 \quad (31)$$

Some shuffling of the terms results in an equation with range as a function of voltage:

$$R = (1/(mV+b)) - k \quad (32)$$

It can be rearranged further to get:

$$R = (m'/(V+b')) - k \quad (33)$$

Where m' = 1/m and b' = b/m. This extra step produces an equation that works nicely with integer math.

Deciding on a value to use for the constants of Equation 33 takes some work. The first step is to obtain some calibration data experimentally by putting an obstacle in front of the IR sensor at different distances (e.g., 10cm, 20cm ... 80cm, etc.) and measuring the voltage readings of the sensor.

Bits A2D	**10**
Supply V	**5.00**
Constant k	**4.00**

Calibration Data		Raw Linearizing Data	
Voltage	**R (cm)**	**A2D**	**1 / (R + k)**
2.40	**10**	491.52	0.0714
1.36	**20**	278.53	0.0417
0.97	**30**	198.66	0.0294
0.77	**40**	157.70	0.0227
0.62	**50**	126.98	0.0185
0.56	**60**	114.69	0.0156
0.48	**70**	98.30	0.0135
0.40	**80**	81.92	0.0119

Figure 38: Calibration data for the IR proximity sensor.

The calibration data is shown in Figure 38, where A2D is the raw input the computer gets from the A/D converter (i.e., $A2D = \frac{Voltage}{SupplyV} \times 2^{BitsA2D}$). The parameter

k is most important in finding a good linearization function. We choose the best value of k by trial and error until the outputs of the conversion function fit the real voltages.

After the calibration experiment, we select m' = 6787, b' = -3 and k = 4. The final approximation function is:

$$R = 6787/(V-3)-4 \qquad (34)$$

This approximation works well for one IR sensor in the array and we have to repeat the process above eight times. A successful implementation depends on the quality of the calibration data and a good choice of the constant k to straighten out the curve.

3.1.2 Methodology for Ultrasonic Sensor Ring Problems

The K2A is equipped with a ring of ultrasonic sensors which are a kind of economical sonar ranging module that works with Polaroid® 6500 series electrostatic transducers. This module is able to measure distances from 15 cm to 11 meters accurately. The typical relative accuracy is ±1% of the reading over the entire range.

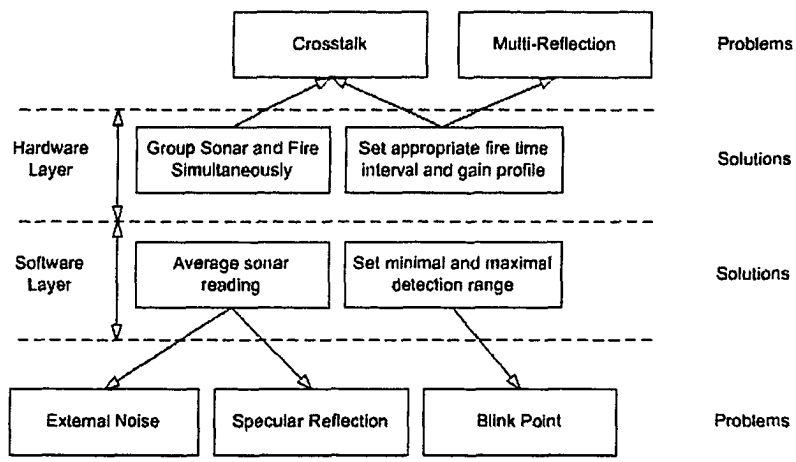


Figure 39: Different approaches used to solve the common problems.

Due to the potential problems of ultrasonic sensors (see Section 2.2.1), we applied four different approaches in an attempt to avoid these problems or to minimize their effects. These approaches include hardware optimizations and software improvements and are implemented at different layers. Figure 39 shows the four approaches along with the common problems that they address.

Group Sonar Transducers and Fire Simultaneously

A limitation of ultrasonic sensors is the time required to gather the sonar data (at least 5.8 ms per meter of desired depth). If we fire the sonar ring sequentially, then it will take a long time to get all of the sonar readings (approximately 2 to 3 seconds). Such a low update rate does not meet the requirements for real-time map building or collision avoidance. In order to increase the data gathering speed, several transducers may be fired simultaneously. The more transducers that are fired simultaneously, the quicker data are gathered. However, this increases the probability of crosstalk [65]. In order to increase the data gathering speed and still keep the possibility of crosstalk low, transducers can be divided into groups and fired at same time.

Theoretically, if the emitting angles of two transducers are not overlapping, they can be fired at same time. For our case, the 24 transducers are 15 degrees apart and the emitting angles of these transducers are 25 degrees. Theoretically, even numbered transducers and odd numbered transducers can be fired simultaneously since their emitting angles are not overlapping. However, due to the complexity of surface condition of obstacles and multi-reflection, in practice two transducers need more "angle cushion"

to avoid crosstalk. In our experiments, we divided the 24 ultrasonic sensors into three groups. At any time, three transducers were fired simultaneously and the angle between any two firing transducers reached 120 degrees (see Figure 40).

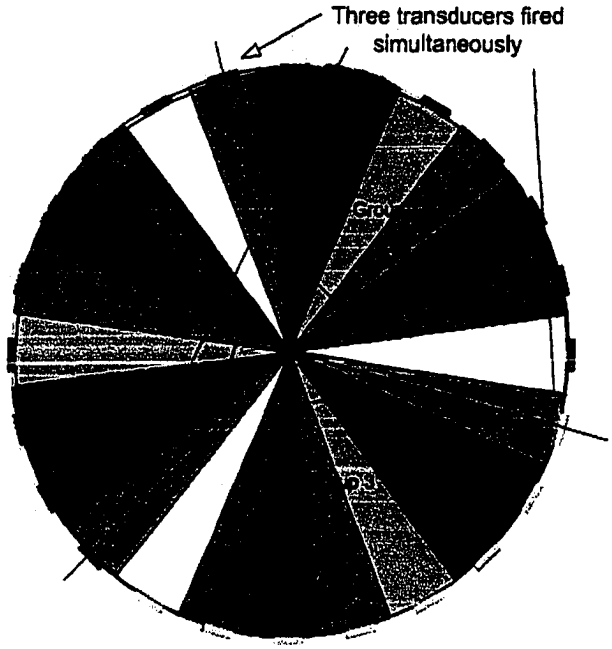


Figure 40: The fire sequence of the sonar ring.

Calibrate Distance Conversion Function and Angle Distribution

On the K2A, there are three transducer control modules (*TCMs*), and each controls the excitation pulse operations of eight transducers. The width of the digital output pulse is the time between the excitation of the transducer and the time of arrival of the first ultrasonic echo that is larger than a preset threshold. These transducer control modules are connected with a digital signal processor (*SBC*) and the computer obtains sensor readings from the SBC through the serial link (refer to Appendix 1). These readings are not distance information but have a linear relationship with it, so we needed to do calibration experiment to find out the suitable conversion function.

The emitting angle is not listed in the manual and 30 degrees is typically used for Polaroid serial sonars in the literature. The emitting angle is very important for the building of sensor models (as introduced in Chapter 2.3.1), so we had to find out the actual emitting angle through experimentation.

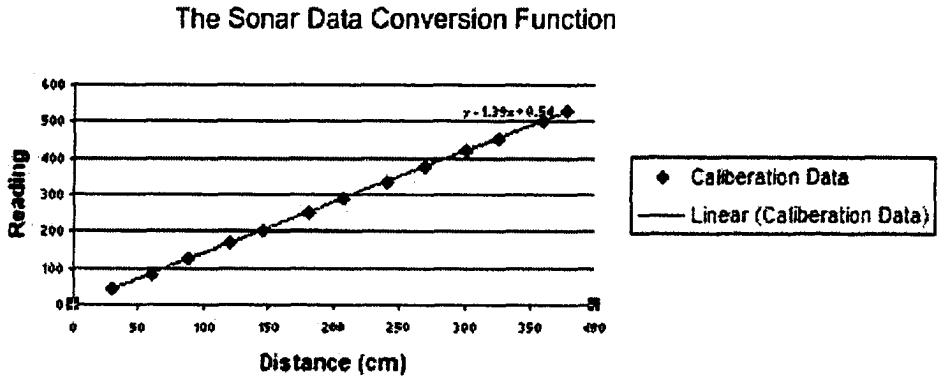


Figure 41: Calibration data and the distance conversion function.

Such calibration experiments were done by moving a small obstacle in front of a sonar transducer and collecting sensor readings accordingly. The results show an excellent linear representation (see Figure 41), indicating that this sonar has high accuracy on distance measurements.

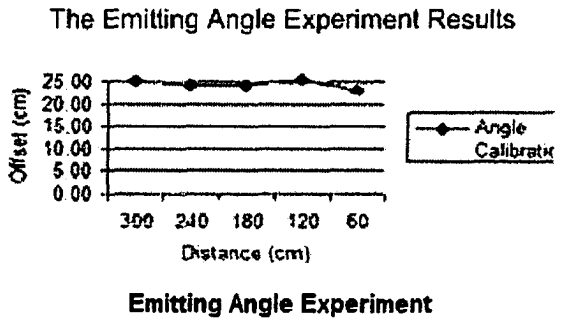


Figure 42: Experimental result to determine actual emitting angle.

We also examined the actual emitting angle of these sonar transducers. By moving a small obstacle at different distance in front of the transducer from one side to the other, we tested the boundary of the reliable detection region. The experiment results are shown in Figure 42. The experiments show that the actual emitting angle is about 25° and this number is used in the setup of the sensor model for these sonars.

Set Minimum and Maximum Detection Range

As we just discussed, after transducer excitation, there will be a short time period to reset the transducer for receiving the return echo; this period is called "blink spot" (*see* Figure 11). A sonar sensor has a minimum detection distance due to this blanking time; during the transducer reset, it cannot detect the return echo. The blanking time can be found in the specifications of the transducer control circuit. The default blanking time for the Polaroid 6500 module is 2.38ms. This blanking time dictates the shortest time allowed the echo to be received. We can calculate the minimum detection distance by using the following equation:

$$Dis_{\min} = V_{sound} \times T_{blank} / 2 = 360 \times 0.00238 / 2 = 0.43m \qquad \textbf{(35)}$$

Based on the calculation above, we set the minimum detection range as *0.4 m*.

By setting the maximum detection range properly, we can partially avoid the problems of multi-reflection and crosstalk since multi-reflection and crosstalk tend to enlarge the value of sensor readings. Theoretically, by increasing the gain, we can detect a return echo from more distant obstacles. From a practical point of view, however, it is undesirable to process an excessive amount of environmental information in a single

shot, as that will be computationally expensive. In practice, the maximum detection range is set to be ten times the diameter of the robot. In our case, the diameter of the robot is 1 meter, so we set the maximum detection range as 10 meters. Thus, our effective detection range is from 0.4m to 10m. By ignoring any sensor readings larger than 10 m, we can filter out some of those erroneous sensor readings due to crosstalk and multi-reflection which often produce range data larger than 10 meters.

Expected Shape of Free Space Region (ESFSR)

We apply the concept of ESFSR in order to improve the quality of sonar sensor readings. An ESFSR contains a set of consecutive sonar readings which differ in range from their consecutive readings by less than a tolerance [4]. ESFSR is obtained by taking multiple sensor readings and finding a region in which most sensor readings agree. In Figure 43, the region in darker bold line segments forms an ESFSR. To get the ESFSR, the robot should be in "still" mode and take several readings.

Figure 43 shows an ESFSR generated from 17 sonar ring readings (see Table 1). In this experiment, the robot remained in the center of the lab and took 17 readings using the sonar ring. 96.67% of the readings maintain very low variances, which form the ESFSR, and 3.33% of them show high variances which indicates corrupted data (see the two data points with bold circles in Figure 43). By calculating the mean and deviations of sensor readings, we can get data that belong to the ESFSR and ignore those that do not.

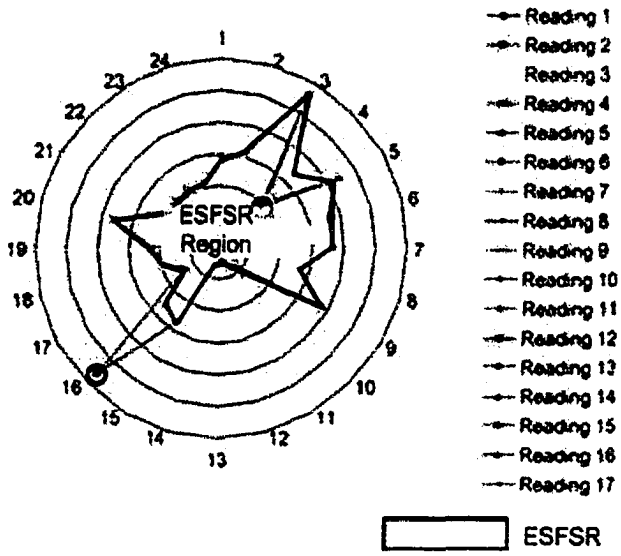


Figure 43: The definition of the RACD in polar coordinates.

Reading 2	2.9	3.1	5.8	3.4	4.2	3.8	3.7	2.8	4.0	2.0	1.2	0.6
Reading 5	2.9	3.1	5.7	3.4	4.2	3.8	3.7	2.9	4.0	2.0	1.2	0.7
Reading 9	2.9	3.1	5.8	2.0	4.2	3.8	3.7	2.9	3.9	2.0	1.1	0.7
Reading 2	0.5	2.0	2.5	5.8	1.2	2.0	2.5	3.3	2.0	2.1	1.9	2.2
Reading 5	0.5	2.0	2.5	2.6	1.2	2.0	2.5	3.3	2.0	2.0	1.9	2.1
Reading 9	0.6	1.9	2.5	2.6	1.2	2.0	2.5	3.3	2.0	2.0	2.0	2.2

Table 1: Sample sonar readings form an ESFSR.

Theoretically, the more readings a robot obtains the more reliable an ESFSR will be found. However, it takes time to get sensor readings from the sonar ring. Be aware that taking too many readings limits the robot's moving speed. In order to get a reliable

ESFSR while keeping the robot running at relatively fast speeds, we programmed the robot to take 2 to 3 readings and then used the average value to find the ESFSR.

3.1.3 Methodology for Stereo Vision Sensor Problems

Since there are many potential problems with a stereo camera sensor, we could not use stereo camera sensor data directly to generate an accurate map. We needed find ways to process the data in order to reduce invalid distance information and minimize the effect of incorrect distance information. This section presents four approaches to help minimize the potential stereo vision problems discussed in Section 2.2.2. Figure 44 shows how these approaches address each of the issues.

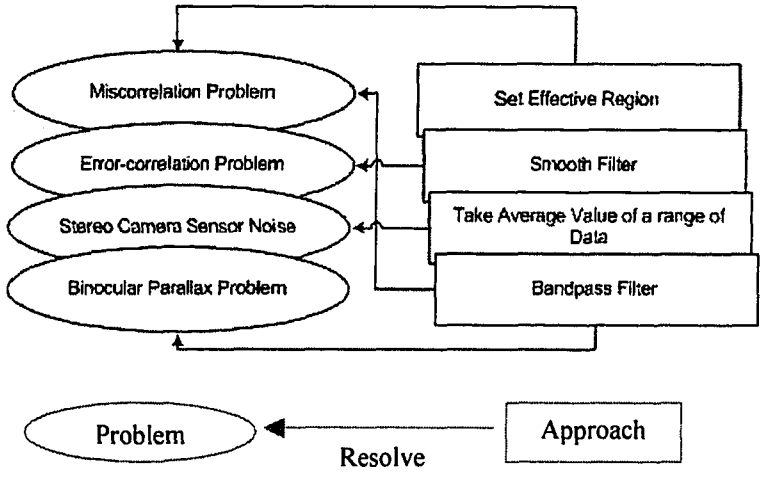


Figure 44: Approaches to resolve stereo camera problem.

Setting the Effective Region

Recall that because of the position offset of two lenses, there is a potential problem of binocular parallax. Since the two lenses of the stereo camera sensor are identical, the sizes of the vision fields are the same. The position difference of the two lenses, which is

known as the *baseline* (see Figure 4), is what makes a scene appear different in the left and right lenses. For our stereo camera sensor, the layout of the two lenses is horizontal, so the upper and lower boundary of each lens are almost the same, but the left and right boundary of each lens is different (see Figure 45).

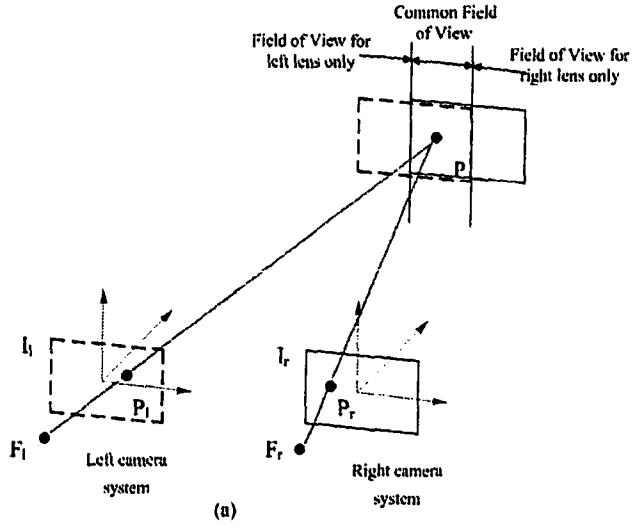
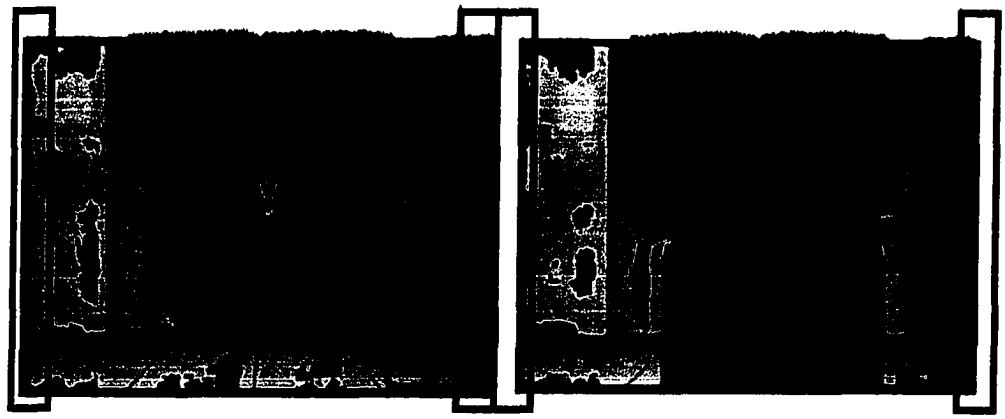


Figure 45: The vision field difference due to binocular parallax.

The stereo matching algorithm figures out distance information through correlation. It can only get distance information from those scenes that can be seen by both lenses. The left edge of the left lens and the right edge of the right lens are scenes that can be seen by only one lens. As the result, they are regions of miscorrelation (see Figure 46).

To reduce the effect of the vision field difference caused by binocular parallax, we set a smaller effective region within the vision field. Only the pixels within the effective region are considered valid (see Figure 47).



(a) Disparity map for experiment one (b) Disparity map for experiment two

Figure 46: The disparity map with binocular parallax problem.

Setting an effective region cannot resolve the binocular parallax problem, but it can help to reduce the need to deal with the problem. This approach does reduce, slightly, the area of the scene that can be processed. A similar approach has been applied by Martinez Edgar and Ohya Akihisa [22].

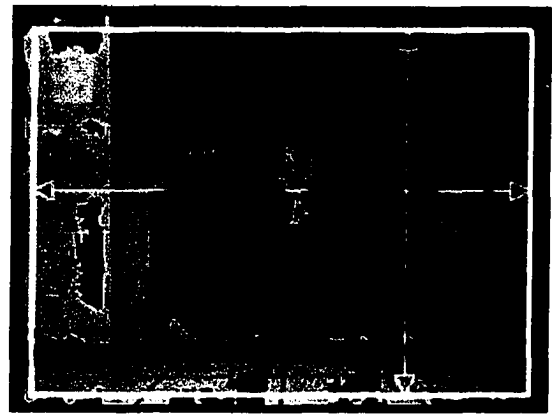


Figure 47: The effective region of a stereo image.

The width of the vertical border is fixed and the purpose is to get rid of unstable correlation on the vertical border. The horizontal border is related to the length of the baseline and the distance between obstacles and lens of the stereo camera. Since the

baseline of our stereo camera is fixed, the width of the border is only related to the distance between lenses and obstacles. As illustrated in Figure 48, when the baseline length increases, the border width increases and when the distance between the lens and obstacles increases, the border width decreases.

In order to find the most suitable border width dynamically, our solution is to probe the distance to obstacles by randomly sampling several points in the scene and estimate the distance between lens and obstacles. The border width is set based on the estimation result.

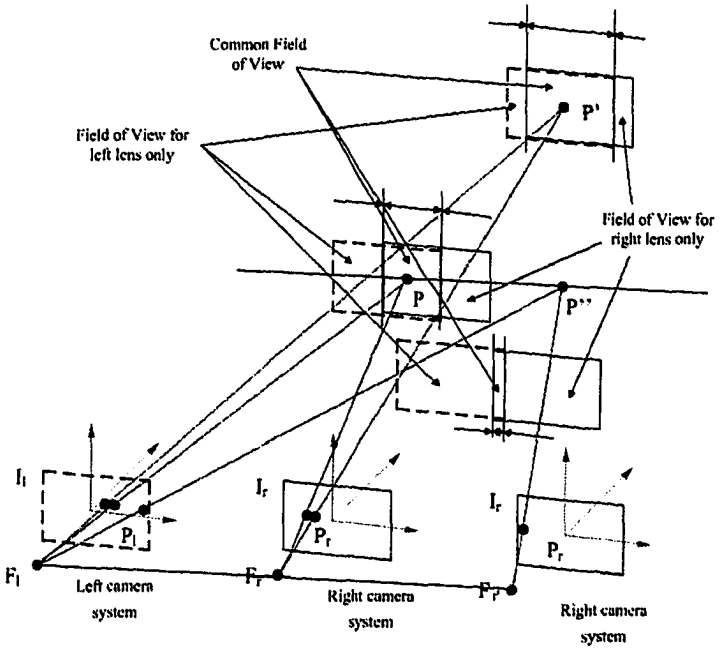


Figure 48: Relationship among effective region, length of baseline and distance to obstacles.

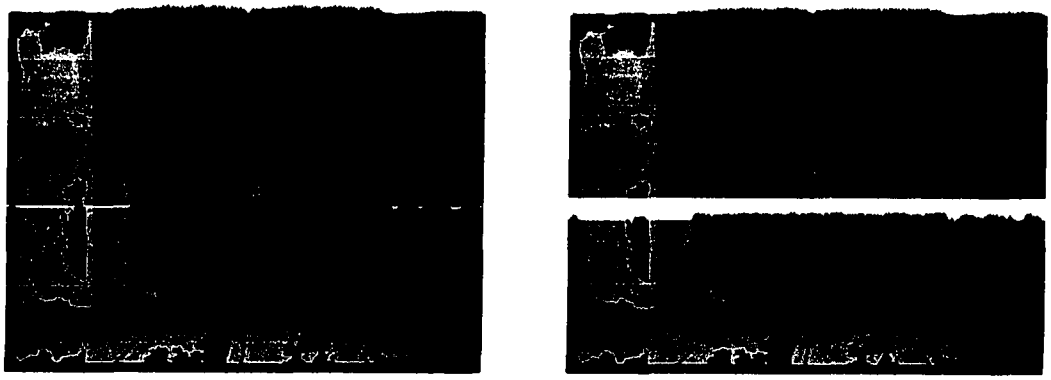
Take the Average Value of a Range of Data

In our experiments, we define the correlation quality of a stereo vision sensor reading as the percentage of pixels that are correlated correctly: The higher the

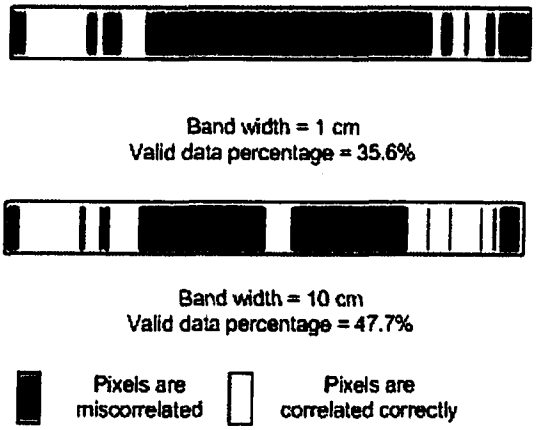
percentage, the better the quality. When there are correlation errors due to miscorrelation, however, the reading could be noisy. When miscorrelation occurs, error correlation usually also occurs. On the border of the miscorrelation region, the correlation results are not stable over time. They are sometime correlated properly sometime and erroneous at other times. Thus, by calculating the average distance values of several continues frames, error correlation errors can be reduced.

The stereo camera sensor on the robot can acquire and process images at 12Hz. Instead of using the correlation result from one frame as the sensor reading for data fusion, we take correlation results from several consecutive frames and use the average value of these results as our final distance value. If a given pixel can be correlated correctly, the discrepancy of these results will fall in the normal error rate. If the pixel cannot be correlated, its distance value will be set to infinity. After taking the average, the distance value of this pixel will still be infinite.

As also explained, a two-dimensional array is used to store the distance information of a scene. Instead of obtaining the distance information from a single row, we average the distance information from several rows adjacent to it. The valid distance values in each column of these rows are averaged and the result is used as final distance value. We call these chosen rows a *"band"*. Under the assumption that there is no abrupt depth change with in the band, this approach can generate a union of correctly correlated pixels and improve the estimate of the distance to the obstacles. By limiting the width of the band, we assume that the assumption is met.



(a) Band width h = 1 cm. (b) Band width h = 10 cm.



(c) Distance information from scan bands having widths 1 cm and 10 cm.

Figure 49: Advantages of averaging a band of data.

As shown in Figure 49, we obtained the distance information from two bands at the same height but with different widths. We made the width of the band of the first experiment equal to 1 cm (Figure 49 (a)), and increased the band width to 10 cm in the second experiment (Figure 49 (b)).

The distance results are shown in Figure 49 (c). The white strips in the band indicate valid distance information, while the black strips indicate invalid distance information. The width of the bands is same as the width of the stereo images. Due to

textureless regions and miscorrelation, the percentage of valid distance information is 35.6% in the first experiment; with the application of our approach (e.g., experiment 2), the valid data rate increases to 47.7%.

Multi-Thresh Filter

After finding the average value of a range of data (both temporally and spatially), a multi-thresh filter was applied to remove some invalid distance information. A multi-thresh filter allows data between two specific thresholds to pass, but discriminates against data at other values. The two thresholds used in the multi-thresh filter are the minimal and maximal detection ranges of the stereo vision sensor. It can eliminate extremely large or small distance values due to error correlation, but cannot eliminate invalid data caused by error correlation. A similar approach is described in [42].

Fusing Data from Other Sensors

None of the approaches mentioned can resolve the miscorrelation problem nor the error correlation problem. They can only reduce the problems by minimizing the their negative effects. Fusion of multi-sensor data provides the solution to reduce the miscorrelation and error correlation problems more successfully. There are three kinds of sensors on our robot, a stereo camera, an IR proximity sensor array and a sonar ring. Integrating the distance information from the sonar and the IR proximity sensor array, miscorrelation regions are often filled by sensor readings from other sensors and erroneous sensor readings due to error correlation are rectified.

3.2 Heterogeneous Sensor Data Fusion Strategy

As introduced in the last section, in order to get more reliable sensor readings, modern robots are normally equipped with different kinds of sensors. Since each kind of sensor has its particular limitations, the sensor combination becomes very important. A proper sensor combination strategy enables a multi-sensor system to select the most appropriate sensor configuration.

Sensor characteristics need to be studied in order to get a good sensor combination. We performed a variety of experiments to research the performance of each kind of sensor, and our sensor combination strategy was designed on basis of our experimental results.

Our sensor combination strategy maintains a balance between performance and cost. The ultrasonic sensor has a long detection range from 10.5cm to 10m which is accurate on the distance measurement (less then ±0.5%) but relatively less accurate on the direction (the emitting angle around 25°) based on our experimental results. Theoretically, the detection range of the stereo camera is from around 1m to infinity. Within 1m, the binocular parallax causes serious problems. On the other hand, the detection accuracy drops on large distance due to the short baseline. Within a reasonable detection range (e.g., 1m to 10m), the stereo camera shows high distance accuracy and directional accuracy. The IR proximity sensor is the most economical sensor of all three sensors with a relatively short range (within around 1.1m). The IR proximity sensor has relatively high accuracy on the direction measurement (around 5°) but relatively low

accuracy on the distance measurement (around ±5%). Instead of relying on a few high-accuracy sensors such as an expensive laser range finder, we used multiple less accurate sensors. A similar approach was taken by Balchen and Dessen [5]. They have investigated the performance versus cost tradeoffs in using a large number of less accurate and lower cost sensors as opposed to a few highly accurate but more expensive sensors.

Through sensor data fusion, the fused data could be considered as the sensor readings of a virtual sensor which is the combination of three different sensors. It is typically called a "combined sensor" which obtains a larger detection range, better accuracy, and lower cost when compared to each individual sensor (see Table 2).

Sensor Type	Min Range	Max Range	Distance Accuracy	Angle Accuracy	Cost
Ultrasonic Sensor	>10.5cm	1000cm	High	Low	Low
Stereo Vision Sensor	>100cm	1000cm	High	High	Medium
IR Proximity Sensor	>10cm	110cm	Medium	High	Low
Combined Sensor	>10cm	1000cm	High	High	Medium

Table 2: Sensor comparison for the K2A robot.

3.2.1 Sensor Models Applied

There are three heterogeneous sensors on our robot. Since each sensor has different characteristics including range, distance accuracy and angular accuracy, we needed to build a sensor model for each of them in order to fuse their sensor readings together. Before we describe these sensor models, we first introduce an important concept from probability theory: the Gaussian distribution, which is the basis of our sensor models.

Gaussian Distribution

When many independent random factors act in an additive manner to create variability, the data will follow a bell-shaped distribution called the Gaussian distribution (also called Normal distribution)[25].

The Gaussian distribution plays a central role in statistics because of a mathematical relationship known as the *Central Limit Theorem* [8]. To understand this theorem, consider this imaginary experiment.

1. Create a population with a known distribution (which does not have to be Gaussian). For example, let the population be the distance from the sensor to an obstacle.
2. Randomly pick many samples from that population. Tabulate the means of these samples.
3. Draw a histogram of the frequency distribution of the means.

The central limit theorem says that if your samples are large enough, the distribution of means will follow a Gaussian distribution even if the population itself does not. Since most statistical tests (such as sensor readings) are concerned only with differences between means, the Central Limit Theorem guarantees to be correct even when the populations are not Gaussian. The catch is that the population has to be reasonably large. The necessary size depends on how much the population distribution differs from a Gaussian distribution. For sensor readings, instead of taking several readings, we have to take several hundred readings.

Gaussian Noise Model

If the spectrum of sensor noise does not contain a special frequency[8], then this noise is "white noise", which has a mean value of zero. Without interference, the sensor reading error of sensors belongs to "white noise" [29]. For example, if the error rate of a sensor is ±1% on its detection range and it gets a reading of 10 meters, then the obstacle can be at any position between 9.9 meters and 10.1 meters, inclusive. The probability distribution of its position in this range follows a Gaussian distribution (See Figure 50).

Near the center of the distribution, the probability will be higher; the value will be lower near the boundary. This distribution model becomes the distance distribution model of a sensor.

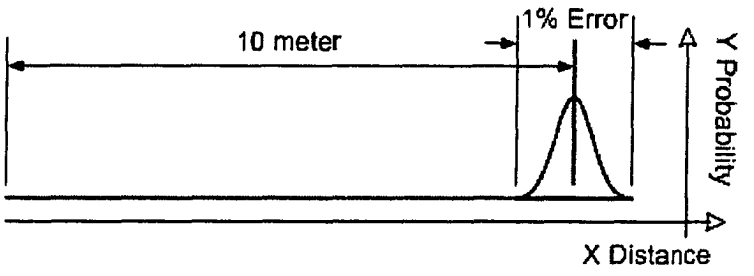


Figure 50: Gaussian distribution of sensor reading.

In addition to the distance distribution model, there is another distribution model for sensors, the angle distribution model. Each of the three distance sensors has a "beam width" (e.g., the sonar and the IR proximity sensor emit an active beam and detect the "echo" (see Figure 50)). It is a little different for the stereo camera, since there is no

[8] Sensor noise due to special reason is called special frequency. For example, the sonar sensor can be affected by sound with similar frequency.

active beam emitted. A stereo camera is a passive sensor, which does not emit anything, but rather receives a light signal from the environment. However, we can define the concept of a "virtual light beam" for it. For any camera, there is a concept of the "view angle", which is defined as the angle introduced by a photographic lens. For digital cameras, the resolution of the image defines how much information the captured image would contain. For example, our stereo camera can obtain an image of 320 × 240 pixels at 30Hz, which means our vision field consists of 320 × 240 pixels. We can imagine that there are 320 single-beam laser range finders working together (See Figure 51c), and that the distance value of each pixel represents the distance reading of each such laser range finder.

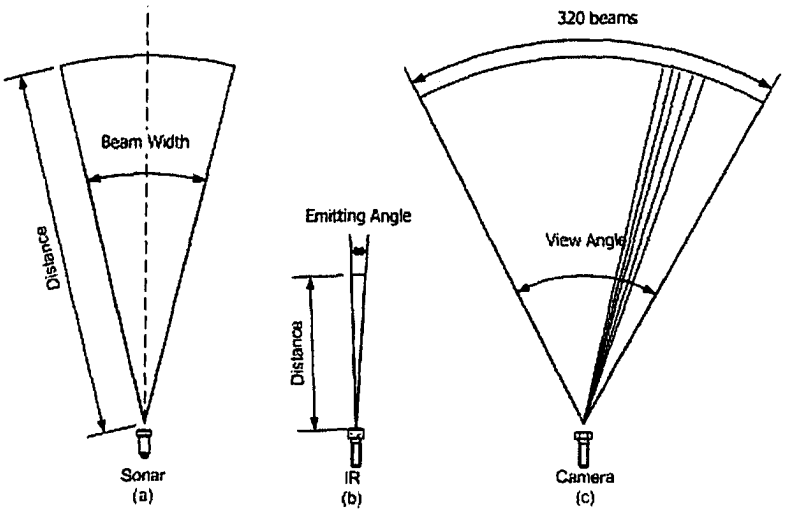


Figure 51: The beam widths of difference sensors.

The angular distribution is relevant because when a sensor gets a reading, the obstacle can be in any position on an arc. There are different approaches to model the

angular distribution of sensors, such as Gaussian distribution [15], classic distribution [9] and self-defined distribution [66].

Several considerations motivate us to apply the Gaussian distribution-based sensor model. Firstly, Kalman filter requires the system model and noise model to be Gaussian [38]. Secondly, Gaussian distributions have many convenient properties, so random varieties with unknown distributions are often assumed Gaussian, especially in physics and astronomy. Thirdly, since the result of adding two Gaussian distributions together is still Gaussian, this enables the recursive procedure of Kalman filter to work properly.

In our sensor models, the angular distribution is also modeled using the Gaussian distribution to be unified with distance distribution. Under the Gaussian angular model, when a sensor obtains readings, the obstacle is more likely to be around the bisector of the emitting angle than at the edges (See Figure 52).

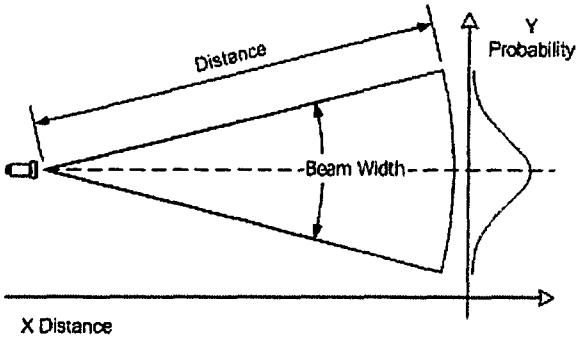


Figure 52: The angular distribution of a sensor.

The final sensor model is the combined distribution of both distance distribution and angle distribution.

Mathematical Description of a Gaussian Sensor Model

In [8]. It is shown that the Gaussian distance distribution as follows:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma_d^2}} e^{-\frac{(x-x_d)^2}{2\sigma_d^2}} \quad (36)$$

x_d is the mean, σ_d is the standard deviation. For the distance distribution, σ_d is the error rate of each sensor on distance, and x_d is the detected distance. For example, if the sonar sensor gets a sensor reading of 10m then $x_d = 10$. If the error rate of the sonar on distance is ±1% then $\sigma_d = 10 \times 2\% = 0.2$m. It means the obstacle can be at any position between 9.9m to 10.1m and its probability distribution follows the Gaussian distribution.

The angular distribution also follows a Gaussian distribution; this is shown in Equation 37, where x_a is the emitting direction and σ_a is half of the emitting angle.

$$f(y) = \frac{1}{\sqrt{2\pi\sigma_a^2}} e^{-\frac{(y-y_a)^2}{2\sigma_a^2}} \quad (37)$$

The final sensor model is the combination of Equation 36 and Equation 37 (see Equation 38). The $\sigma_{combined}$ is a covariance matrix for two random variables (see Equation 39). Since we assume that these two random variables are independent, the covariance σ_{ad} and σ_{da} is 0. The $\sigma_{combined}$ becomes $\sigma_d\sigma_a$.

$$f(x,y) = \frac{1}{2\pi\sqrt{\sigma_d^2\sigma_a^2}} e^{-\left(\frac{(x-x_d)^2}{2\sigma_d^2} + \frac{(y-y_a)^2}{2\sigma_a^2}\right)} \quad (38)$$

$$\sigma^2_{combined} = \begin{pmatrix} \sigma^2_d & \sigma^2_{da} \\ \sigma^2_{ad} & \sigma^2_a \end{pmatrix} \quad (39)$$

The random variables x_d and y_a in Equation 40 and Equation 41 are the expected values $E(x)$ and $E(y)$ in a Gaussian distribution, and that σ_a^2 and σ_d^2 are the mean squared errors in the distribution. To calculate σ_a^2 and σ_d^2 accurately, we need to calculate the continuous integral between [−∞, +∞], which is computationally expensive. In order to simplify the computation and still keep a relatively high accuracy, we use the three-sigma rule[9] to calculate σ [8]. Based on the Gaussian distribution, the probability values within the six-sigma region constitutes more than 99.72% of the probability, so We use the six-sigma range to represent the whole space from $-\infty$ to $+\infty$ (see Figure 53).

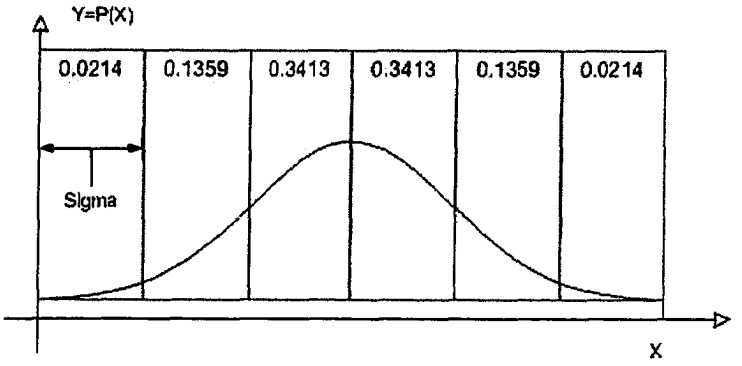


Figure 53: Six-sigma rule, the total probability from -∞ to +∞ is 1. The probability value within the range from -3σ to +3σ reaches 0.0214 + 0.1359 + 0.3423 + 0.3413 + 0.1359 + 0.0214 > 99.72% ≈ 1.

[9] Three-sigma rule: Approximately 99.7% of the data values will lie within three standard deviations from the mean.

Recall our example about the sonar sensor. Considering the distance error rate is ±1% and the sonar gets a reading of 10m, the six sigma range is from 9.9m to 10.1m, the absolute error is 0.2m, and the relative error is 2%. Since the 0.2m covers the six sigma region, then the σ_d is set to 0.2 / 6 = 0.033. If the sensor reading was 1m, the σ_d becomes 0.0033. The σ represents the accuracy of sensor readings, which is a characteristic of sensor readings, independent to sensor types. The sonar can also generate highly accurate sensor readings when close to obstacles. We applied this approach for all three sensor models.

The sensor models used in our fusion algorithm are based on Equation 38. For each sensor, we apply a different approach based on its characteristic, which are introduced in Sections 3.2.1.1, 3.2.1.2 and 3.2.1.3. In order to improve the real-time computation efficiency, we further simplified the sensor model for the stereo camera.

3.2.1.1 Sensor Model for Sonar

The sonar sensor has very high distance accuracy but a wide beam, so its angular resolution is low. The beam width of each sonars is 25°. The combined distribution is shown in Equation 38, but σ_d is much smaller than σ_a. The diagram of the final distribution is shown in Figure 54.

P is the output of the sensor model, a probability value, D is the measured distance, R is distance and θ is angle. Δ = D × error rate%. $\alpha = 2\pi D \times \theta / (360 \times 2)$. From Figure 54, we can see that if an obstacle is close to the transducer then the certainty about the location of the obstacle is higher than farther away. The reason is that if the obstacle is

closer while the emitting angle keeps constant, the length of the arc will be shorter. The position of the obstacle will be distributed on a shorter arc, which is more likely to fill in one grid. On the other hand, if the distance is longer then we are less certain about the position of the obstacle, because the probability function will be distributed along a longer arc that will cover more grid cells.

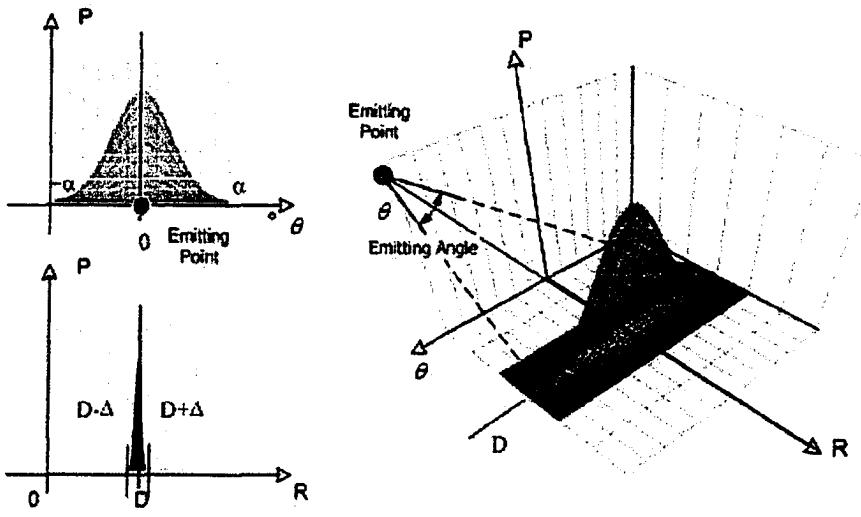


Figure 54: The combined sensor model for Polaroid 6500 sonar sensor.

3.2.1.2 Sensor Model for IR Proximity Sensor

As we introduced in Section 2.1.4, the IR proximity sensor has a very narrow beam (< 5°) and due to its short detection range, its beam width does not change by much [53]. Since these IR proximity sensors measure the angle of reflected infrared light, they could be affected by the surface conditions of obstacles.

P is the output of the sensor model, a probability value, D is the measured distance, R is distance and θ is angle. Δ=D × error rate%. $\alpha=2\pi D\times\theta/(360\times2)$. Based on our

calibration experiments, when an obstacle is close to the maximum detection range, the readings became noisy. The average full detection range error rate is set to ±5%. The combined sensor model for an IR proximity sensor is shown in Figure 55.

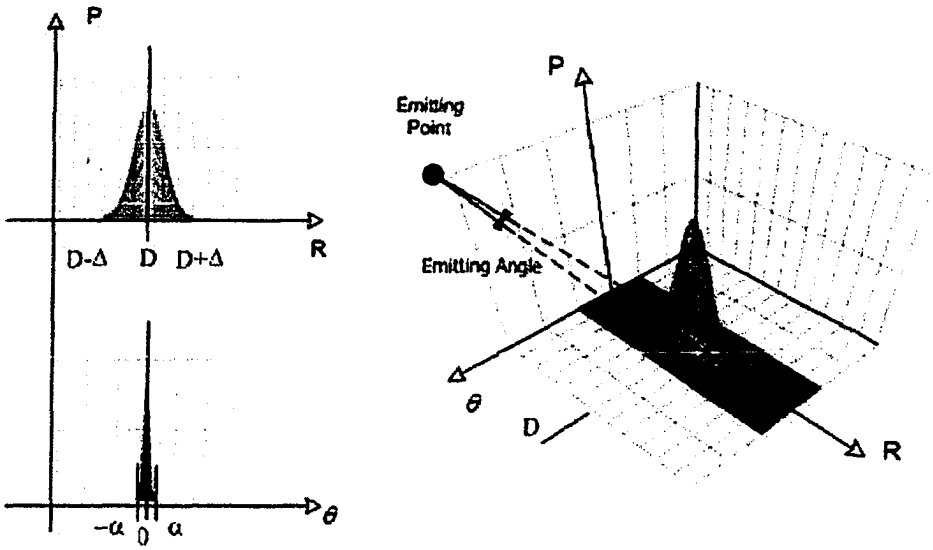


Figure 55: The combined sensor model for IR proximity sensor.

3.2.1.3 Sensor Model for Stereo Vision Sensor

The sensor model for the stereo camera is a different case. We treat the stereo camera sensor as a cluster of virtual single-beam range finders, where each pixel in the image is an independent virtual single-beam range finder. The resolution of the camera is 320 × 240 pixels, and therefore each scan line requires processing 320 virtual sensor readings.

The effective detection range of our stereo camera sensor is 10m. Beyond 10m, the accuracy decreases, since the disparity between a stereo image pair becomes lower. This

well-calibrated stereo camera sensor has high distance accuracy within the effective range. P is the output of the sensor model, a probability value, D is the measured distance, R is distance and θ is angle. $\Delta=D \times$ error rate%.$\alpha=2\pi D\times\theta/(360\times2)$.

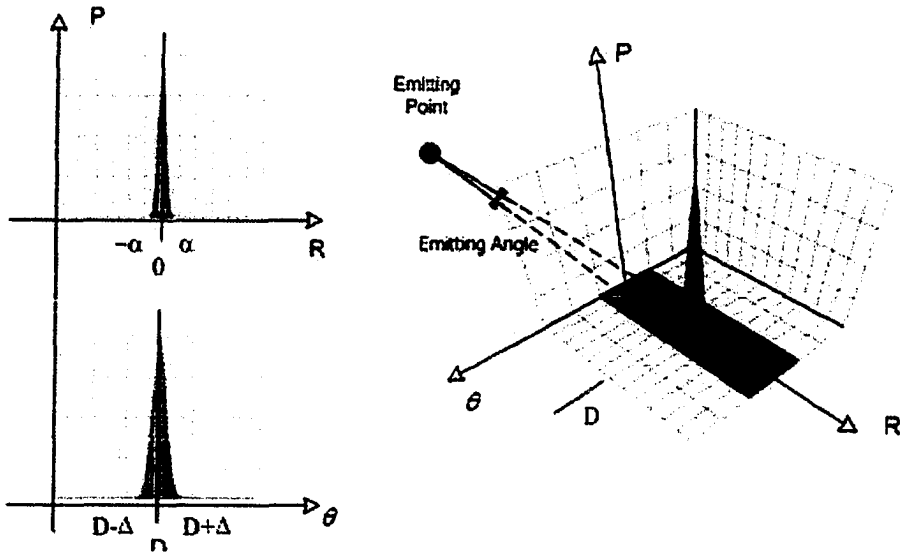


Figure 56: The combined sensor model for stereo vision sensor.

Based on the technical specification of the stereo camera [46], the error rate of its distance measurements is 1%, and its view angle is 50°. Therefore, the angular resolution is 0.156° (i.e., 50/320). Therefore, the distance resolution and the angular resolution for the stereo camera are both very high.

The combined sensor model for a stereo vision sensor is shown in Figure 56. We can see that when the stereo camera detects an obstacle, we can be very confident about the obstacle's location, since all of the probability values for the location fall within a narrow range.

3.2.2 Obtaining Obstacle Measurements

From Equations 36, 37 and 38, we know that σ_d and σ_a are critical parameters, which define the characteristic of the sensor model. σ_d and σ_a can be preset, but they are also involved in the calculation of the value of each grid element in real-time. We use the "six σ rule" (Section 3.2.1) to simplify the calculation of σ_d and σ_a. Now we discuss some practical approaches to further simplify the calculation without a big degradation in the accuracy.

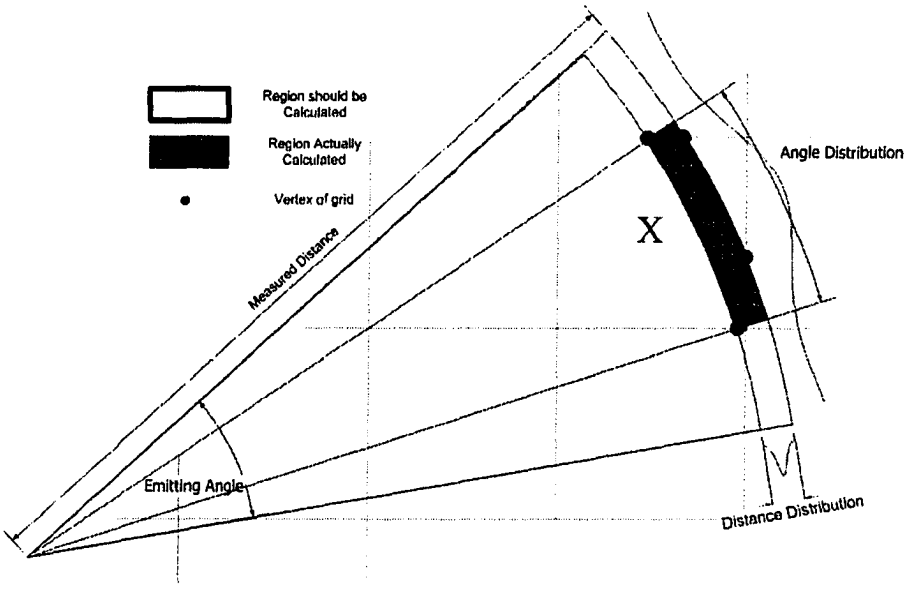


Figure 57: Computing the probability value of a grid, using four vertices of the grid to simplify the computation.

All of the discussions in this section are based on an ultrasonic sensor but they are applicable to the other three sensors. Compared with sonar sensor's low angular resolution (25°), the ±1% distance measurement error rate is insignificant. But when the sensor readings approach the maximum detectable range of 10m, the absolute distance

error can reach 20cm, which may cover more than one grid element (in our case, the grid size is 5cm × 5cm). Ideally, we should calculate the integral of the two-dimensional probability distribution within the area of each grid according to combined sensor model, but we have found that that is computationally expensive (See Figure 57).

In order to compute the accurate probability value of a grid cell, we need to compute a second-order integral on distance R and angle θ which is the area of the region within the bold line. Since the edges of the grid are not parallel to the axes R and θ, it is difficult to compute the double integral within such a region.

To simplify the problem and to keep the algorithm's update rate high and real-time, we compute a bigger region instead (shaded region in Figure 57) whose edges are parallel to the R and θ axes. This approximation introduces calculation error, but we can prove that this error is bounded and limited. In Figure 58, we show the best and worst case in the simplified calculation and from the analysis. The square represents a grid cell and when it is scanned by a sensor. In the best case, all probability fall in this grid cell which is 1 and the region we calculated is exact same as what should be calculated. In this case Vcomp = Vaccu (see Figure 58 (a)). In worst case, the grid cell is a part of the sensor scan region. The square is the region we should integrate, but the sector region is what we actually integrated and the limit of the sector area is two times the area of the square. We can predict that the computed value will be larger than the accurate value. ($Vaccu \leq Vcomp < 2Vaccu$). This error is bounded and has a limited effect on map generation (see Section 3.2.3), because it can be compensated for by the data fusion algorithm.

For the IR sensor array, from its technical specifications [53], the beam width of the Sharp GP2Y0A02YK IR proximity sensor is 1 cm and is almost constant over the entire detection range. The grid size of our evidence grid map is 5 cm and the beam width is much smaller than the grid size, which means there is a very high probability that the whole angular distribution falls within one grid element. To simplify the computation, for an IR sensor we only consider the distance distribution and ignore the angular distribution.

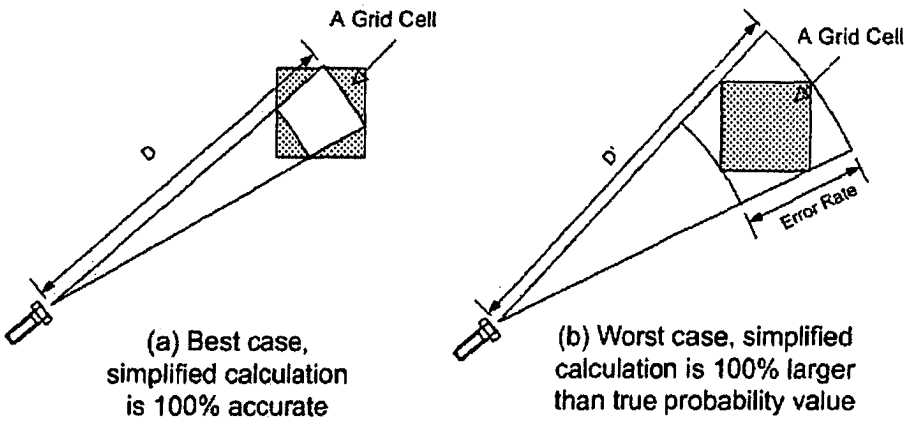


Figure 58: Simplified calculation error analysis.

Compared with the sensor data processing procedure for the sonar and the IR sensor array, the data processing procedure for the stereo vision sensor is more computationally expensive. For example, for one complete sensor reading set, we need to process one reading from the IR sensor array (8 readings from the 8 IR proximity sensors), one reading from the sonar ring (i.e., 24 individual transducer readings) and one reading from the stereo camera (320 readings from each virtual sensors). There are many more sensor readings from the stereo camera than from the other two sensors. If we can

simplify the processing procedure of each reading from the stereo camera, we can dramatically speed up the update rate of the data fusion and mapping procedures.

We have shown that the stereo camera has both a high angular resolution and a high distance resolution based on technical specifications and on our sensor model. For a view angle of 50° and an image resolution of 320 × 240 pixels, at maximum range (10 m), the angular spread is only 2.728 cm, which is smaller than the grid size (5 cm). The distance error rate is ±1%, so at maximum range the absolute error could be as large as 20 cm. Based on this analysis, we can predict that most readings from the stereo camera fall in at most four grid cells. To simplify the computation, we use an evenly distributed uniform distribution to approximate the Gaussian distribution.

3.2.3 Applying the Kalman Filter

Our sensor data fusion framework is based on a linear Kalman filter [14] which is one of the most widely used methods for tracking and estimating due to its simplicity, optimality, tractability and robustness. The Kalman filter is a recursive processing technique for noisy measurement data. We apply a distributed multi-Kalman filter approach in our sensor data fusion framework with the architecture shown in Figure 59.

As shown in Figure 54 to Figure 56, we have developed sensor models for each type of sensor and applied Kalman filters to each of them. The "pluggability" of the framework makes it extremely easy to expand the framework to add a new kind of sensor without needing to modify the existing system. We implemented the Kalman filter as an

independent module, which can be used for both robot localization and map refinement. This allows for a certain degree of reusability in the framework.

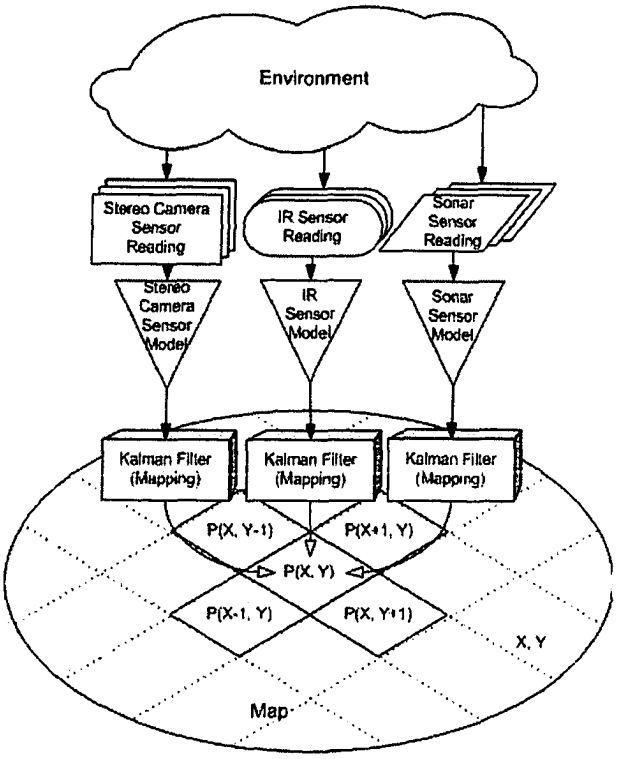


Figure 59 Distributed multi-Kalman filter sensor fusion framework

In Figure 59, different kinds of sensor readings are converted into one format through sensor models. Each sensor type has its own sensor model and its own Kalman filter, which fuses a new sensor reading ($t+1$) with the current probability value in the grid to generate the probability value at time $t + 1$.

In practice, the performance of the Kalman filter is closely related to its parameters. In this section, we introduce the parameters of the Kalman filter used in our experiments.

Since we use an evidence grid map as the representation of the data fusion result, a grid element will be the basic unit for sensor data fusion. The first step is to build the

practical system model (see Equation 10). The system that we are modeling is a grid, and there is no control input for the grid, so we set the matrix B (in Equation 10) to 0 and matrix A (in Equation 10) to 1. Equation 12 and 13 become Equations 40 and 41, respectively:

$$X_{prior}(t) = X(t-1) \quad \textbf{(40)}$$

$$P_{prior}(t) = P(t-1)' + Q \quad \textbf{(41)}$$

Equation 40 indicates that the predicted system state at time t will be same as the system state at time t-1, since there is no control input. Equation 41 indicates that the predicted covariance matrix is the history covariance matrix plus white noise (Q). Next, we calculate the Kalman gain (see Equation 12).

P is a covariance matrix in the Kalman filter. Since the system state is one-dimensional in our approach (i.e., the probability value of a grid element), the covariance matrix becomes a variance matrix[10], which is the square of the standard deviation (σ). Therefore, Equation 12 becomes the equation below:

$$K = \frac{\sigma^2_{prior}(t)}{\sigma^2_{prior}(t)+\sigma^2} \quad \textbf{(42)}$$

R (in Equation 12) is the standard deviation of measurements. The correction procedure of the Kalman filter (see Equation 13 and Equation 14) becomes the equation below:

---

[10] The elements on the diagonal of a covariance matrix $Cov(A,B)$ are the variance of A.

$$X(t) = X_{prior}(t) + \frac{\sigma^2_{prior}(t)}{\sigma^2_{prior}(t)+\sigma^2} \bullet (Z(t) - X_{prior}(t)) \quad \textbf{(43)}$$

$$\sigma^2(t) = (1 - \frac{\sigma^2_{prior}(t)}{\sigma^2_{prior}(t)+\sigma^2}) \bullet \sigma^2_{prior}(t) \quad \textbf{(44)}$$

From Equation 43 and Equation 44, we can see that the state value at time *t* depends on the measurement at time *t* and the state value of time *t-1*. The standard deviation of sensor readings decides if the final state value will tend towards the predictions or towards the measurements. When the measurements are very accurate (i.e., the standard deviation of measurement is small), we put more faith in the measurements than on predictions; when they are inaccurate (i.e., the standard deviation is large), we do the opposite.

After the parameter set-up, the Kalman filter depends only on the accuracy of the sensors; the accuracy of each kind of sensor is represented by their standard deviation, which is provided by sensor model (see Section 3.2.1).

In order to describe the Kalman filter more clearly, we present here an example on an 8×8 grid map which includes three iterations of the Kalman filter computation and fusing two sensor readings into a grid map. The 8×8 grid map is empty at the beginning (all cells are set to 0). The first sensor reading is obtained by the sonar sensor at a distance 5m, the second sensor reading is by the stereo camera at 6m and the third reading is from the IR sensor array.

There are two important values associated with each grid cell: the probability value and the σ which is the standard deviation of the probability value. When the grid map is empty the probability value is set to 0 and the σ is set to +∞ for each cell.

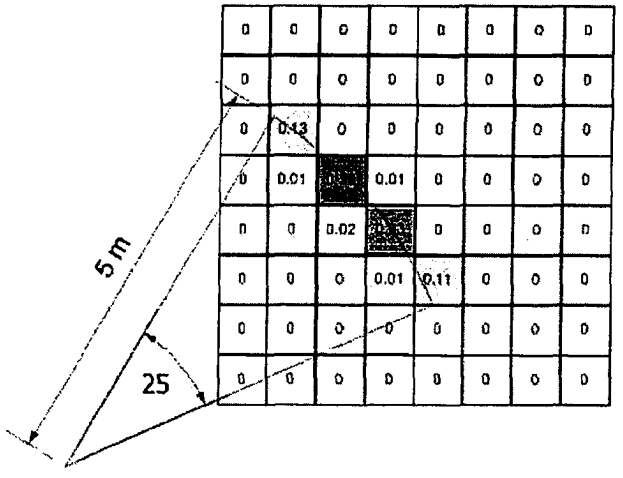


Figure 60: The grid map after initialization by the first sensor reading.

In the first iteration, we initialize the grid cells with the first reading (see Figure 60). The probability value for each cell is calculated using the sonar sensor model as introduced in Section 3.2.1.1 and the σ is calculated based on "three-sigma rule". The first sensor reading comes from the sonar and the measured distance is 5 m. The emitting angle of the sonar is 25° based on our experiments, so the length of the arc is 2.18m ($2\pi \times 5 \times 25/360$). Since six sigma region covers the whole arc, then $6\sigma_{angle} = 2.18$, and so $\sigma_{angle} = 0.36$. If the distance measurement error rate is ±0.5%, then $6\sigma_{distance} = 0.05$ (5×1%), and so $\sigma_{distance} = 0.008$. As introduced in Equation 38, the combined probability $\sigma_{combined}$ is defined as $\sigma_{distance} \times \sigma_{angle}$ in our combined sensor model (see Equation 39), it represents

the accuracy of the sensor. The smaller the number the more accurate the sensor is. The $\sigma_{combined} = 0.36 \times 0.008 = 0.00288$ and this value is also set to each cell.

In the second iteration, we have a new sensor reading from the stereo camera. The probability is calculated using the sensor model of the stereo camera (see Section 3.2.1.3). The measured distance is 6m, the view angle of the camera is 50° and the resolution of the image is 320 × 240 pixels. The degree per pixel is 50 / 320 = 0.156° and then at a distance of 6 m the length of the arc is 0.016 m, so $\sigma_{angle} = 0.016$. The distance measurement of the stereo camera is also 1%, and $\sigma_{distance} = 6 \times 0.01 / 6 = 0.01$. Thus $\sigma_{combined} = 0.016 \times 0.01 = 0.00016$. We can see that the sensor reading from the stereo camera is more accurate than that from the sonar.

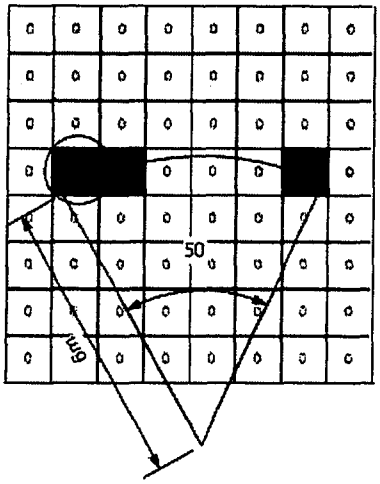


Figure 61: A sensor reading from the stereo camera after being converted by the sensor model.

We use the grid cell on row 4 column 3 as an example to describe the data fusion procedure. First we need to make a prediction, based on Equation 40 and Equation 41 $X_{prior}(2) = X(1)$ and $\sigma_{prior}(2) = \sigma(1)$. When the new stereo camera sensor reading comes in, we make the correction:

$X(2) = X_{prior}(2) + \sigma_{prior}^2(2) / (\sigma_{prior}^2(2) + \sigma^2)(z(2) - X_{prior}(2)) = 0.38 + 0.00288^2/($
$0.00288^2 + 0.00016^2)(1 - 0.38) = 0.998$

$\sigma^2(2) = (1 - \sigma_{prior}^2(2)/(\sigma_{prior}^2(2) + \sigma^2))\sigma_{prior}^2(2) = (1 - 0.00288^2/(0.00288^2 +$
$0.00016^2))\ 0.00288^2$

Here, $\sigma(2) = 0.00015$ which is smaller than either σ_{stereo} or σ_{sonar}, so the data fusion increases the accuracy of the fusion result. We calculate all grid cells according to Equation 40 and 41 and the grid map at the end of iteration 2 is shown in Figure 62.

After iteration 2, the sensor reading from the sonar is fused with the sensor reading from the stereo camera. From the fusion result, the sensor reading from the stereo camera reinforces the sensor reading of the sonar by identifying the actual obstacle position in grid cell row 4 column 2. From the sensor model, the stereo camera has higher accuracy than the sonar and our sensor fusion algorithm tends to trust the high accurate sensors, so the reinforce effect is very strong.

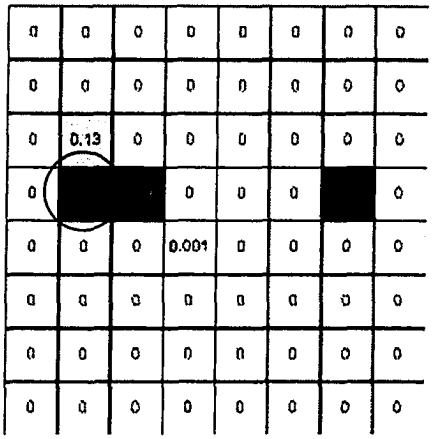


Figure 62: The grid map after fusing the sonar reading with the stereo camera reading.

In the third iteration, we get a sensor reading from the IR sensor array from a very close range of 0.8m (see Figure 63).

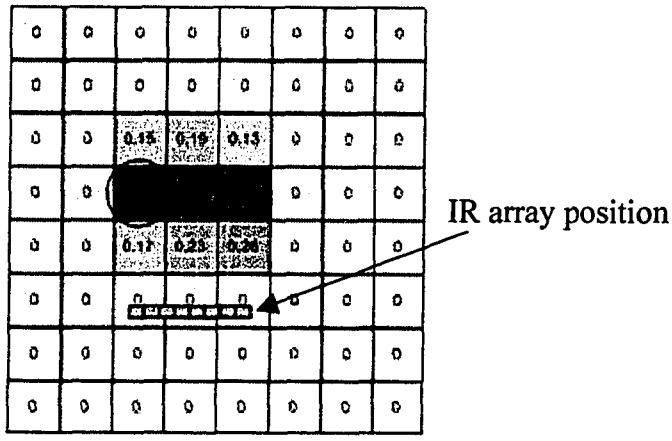


Figure 63: The sensor reading from the IR sensor array converted by its sensor model.

We use the grid cell on row 4 column 3 as an example again to introduce the data fusion procedure. After iteration 2, X(2) = 0.998 and σ(2) = 0.00015 and with the new sensor reading, z(3) = 0.8. Since the measurement error rate is ±5%, the $\sigma_{distance}$ = 0.8 × 10% / 6 = 0.013. Based on the manual of the IR proximity sensor, the beam width of the IR sensor is around 0.01m and the its emitting angle is 5°. At a distance of 0.8m, the beam width reaches 0.08m, so the σ_{angle} = 0.08 / 6 = 0.013 and the $\sigma_{combined}$ = $\sigma_{distance}$ × σ_{angle} = 0.013 × 0.013 = 0.00017. We can see that the accuracy of IR proximity sensor at distance of 0.8m is close to the stereo camera at 6 m ($\sigma_{combined}$ = 0.00016), which means the closer the distance to the obstacle, the more accurate the sensor is.

The data fusion procedure starts with the prediction stage,

$$X_{prior}(3) = X(2) = 0.998$$

$$\sigma_{prior}(3) = \sigma(2) = 0.00015$$

When we get a new sensor reading, the correction stage starts.

$$X(3) = X_{prior}(3) + \sigma_{prior}^2(3) / (\sigma_{prior}^2(3) + \sigma^2)(z(3) - X_{prior}(3)) = 0.998 + 0.00015^2/(0.00015^2 + 0.00017^2)(0.68 - 0.998) = 0.859$$

$$\sigma^2(3) = (1 - \sigma_{prior}^2(3)/(\sigma_{prior}^2(3) + \sigma^2))\sigma_{prior}^2(3) = (1 - 0.00015^2/(0.00015^2 + 0.00017^2))\,0.00015^2$$

Here, $\sigma(3) = 0.00011$ and after the third iteration, the fusion result is shown in Figure 64.

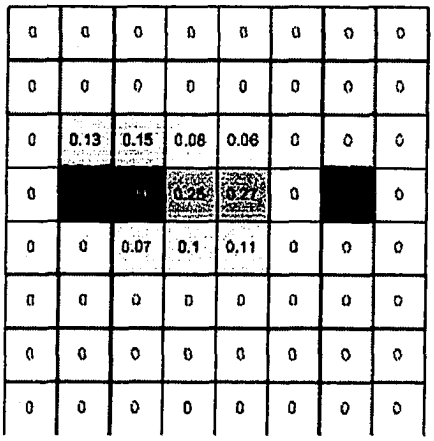


Figure 64: The fusion result of our data fusion algorithm in iteration 3.

From the fusion results, the gap in the sensor reading of the stereo camera which may due to a small textureless region, is partially recovered by the sensor reading from the IR sensor array. As can be seen, after three iterations, the final result gives a better representation of the environment than with only one sensor.

3.3 Software Framework of Data Fusion System

We have designed a software framework for sensor data-fusion and mapping. The design of the framework focuses on expandability and real-time performance. All

experimental data is generated using this framework; the software was developed using the Microsoft Windows® operating system and is written in C++.

There are in total five layers in the software framework: (a) the exploration decision layer, (b) the path planning layer, (c) the map building layer, (d) the sensor data fusion layer, and (e) the sensor actuator layer (see Figure 65). We implemented the bottom layers related to sensor data fusion only. The other layers are application layers and the implementation is not in scope of this thesis. The framework is designed using the unified modeling language (UML). UML is a simple and generic notation made of model elements that can be used to model requirements for design of the software system. Good models are essential for a software application framework and UML helps us speed up the development process, improve code quality, support adaptation to requirement changes, and to develop larger software systems (see Appendix 4).

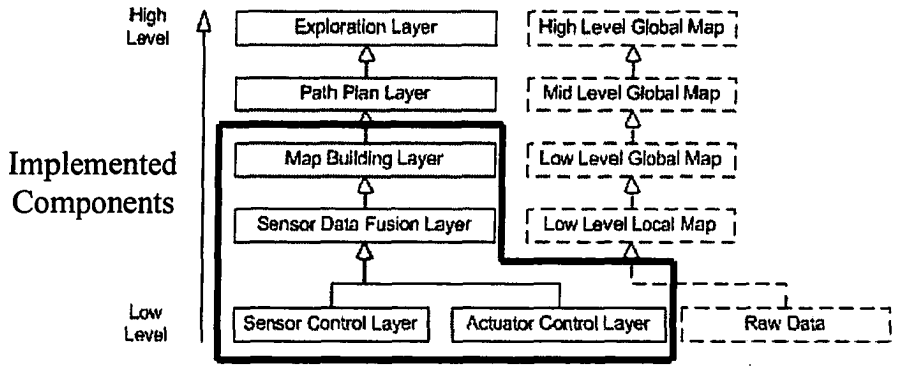


Figure 65: The software framework of the sensor fusion system.

As shown in Figure 65, the whole system can be viewed as two sets of hierarchies. The first is the hierarchy of software components and the second is the hierarchy of maps. The software component hierarchy consists of five layers. The sensor control layer and

the actuator control layer comprise the lowest layer since they talk to the hardware directly and the sensor readings from this layer is raw data. The raw data is fed into the Sensor Data Fusion Layer. Sensor readings from different sensors are fused and those results are used to build a local map. Once the local map has been produced, the map-building layer on the upper level merges the local maps together to produce a global map. With the global map, the robot can make navigation decisions and generate an exploration strategy. The robot maintains the map hierarchy in real time and different maps are used for different purposes. For example, a grid map (the low level map) is used as the final representation of the environment and a path map (the high level map) generated from the grid map, which is more abstract, is used for the high-level decision making.

Chapter 4 Experiments and Results

4.1 Experiments and Result Analysis

In this section, we describe our multi-sensor data fusion experiments and results. These experiments are divided in two parts: single-scene sensor data fusion experiments and multi-scene sensor data fusion experiments. In the single-scene experiments, the robot is moved from location to location and sensors obtained a limited number of sensor readings for the current location independently. Each scene in the single scene experiment is specifically designed to research the performance of each sensor. For example, in order to research the performance of the stereo camera, a low lighting experiment and a transparent obstacle experiment were conducted. During some of single-scene experiments, improvements gained by the sensor data fusion were also investigated. In contrast, when multi-scene experiments were conducted, the robot was moved to a number of locations, sensors obtained sensor readings from different directions and all data was transferred to the data fusion algorithm. For example, in the full lab experiment, the robot moved more than 20 meters and took hundreds of sensor readings. The objective of these multi-scene sensor data fusion experiments was to verify our sensor data fusion algorithm under a realistic scenario of mobile robot map building.

4.1.1 The Experimental Setup

All experiments were conducted in a structured indoor environment: a typical laboratory with laboratory equipment and office furniture. The robot used was the fully autonomous robot K2A manufactured by Cybermotion®.

A central control computer communicated with the robot computer via Bluetooth® devices. The robot's computer was responsible for sensor data fusion, map building, collision avoidance and robot localization (see Figure 66).

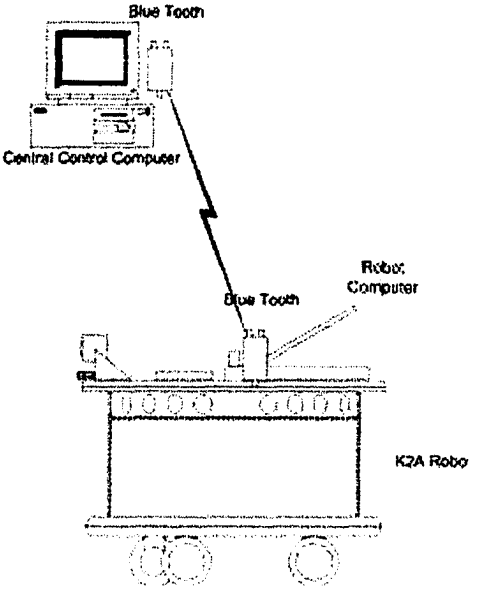


Figure 66: The system diagram of the experimental system.

There are three kinds of sensors on the K2A robot, which were introduced in Section 2.1.3, 2.1.4 and 2.1.5. In this section, we focus on connections and interaction with the robot computer.

The K2A mobile robot was built as an industrial robot with high power and high accuracy. It is designed to provide maneuverability to autonomous and tele-operated

systems in an industrial environments. The synchro-drive locks all three wheels together for both steering and driving. When a turn is executed, all three wheels turn in unison and trace parallel paths to one another. The result is that the platform itself does not rotate as a turn is executed.

The robot consists of two parts, the base and the turret. All of its mechanical parts and the sonar ring are in the base part, while the stereo camera, IR proximity sensor array and on-boaad computer are in the turret part (see Figure 67).

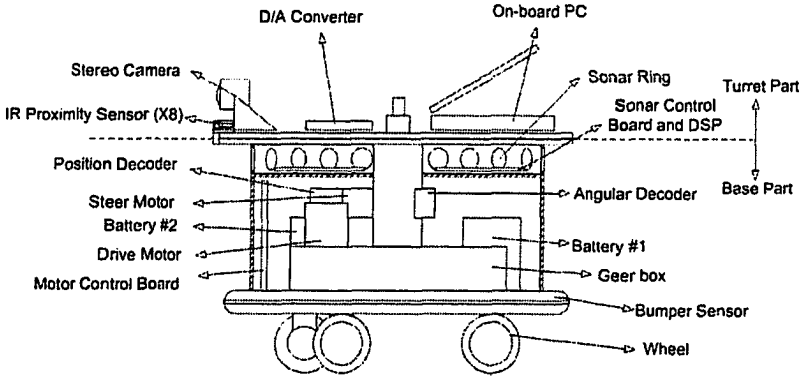


Figure 67: The structure of the K2A.

When the robot turns, the stereo camera and IR proximity sensor array turn at same amount and speed. Thus, they always point towards the direction in which the robot is moving. On the other hand, the base does not turn at all; so the sonar ring mounted on the top of the base always keeps its own orientation regardless of the robot's moving direction.

The main assumption of our sensor data fusion framework and algorithms is that the robot is operating in a structured indoor environment. Thus, all of the experiments

were conducted in a laboratory and in the corridors around the laboratory. In this section, we briefly introduce the experimental environment.

Seven experiments were held in the laboratory, and two experiments were held in the corridor. Figure 68 depicts a top down view of our lab and the surrounding corridors showing the locations that the experiments (#1 to #7) were conducted.

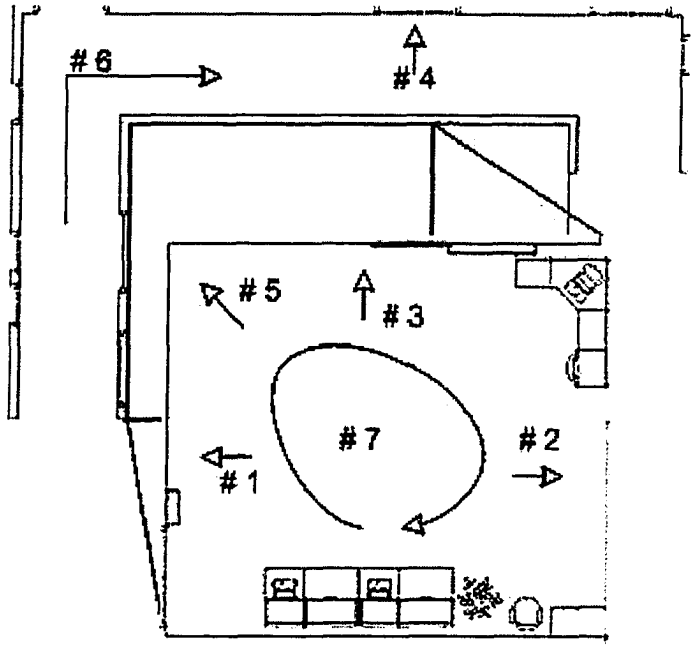


Figure 68: Top-down view of our lab and corridors showing the setting of experiments from #1 to #7.

Experiments 1 through 4 were designed to investigate the deficiencies of the stereo camera sensor under various conditions and the performance of other sensors under such conditions. The tests were done as single-scene sensor data fusion experiments where a handful of readings were taken from a particular sensor from various positions. The

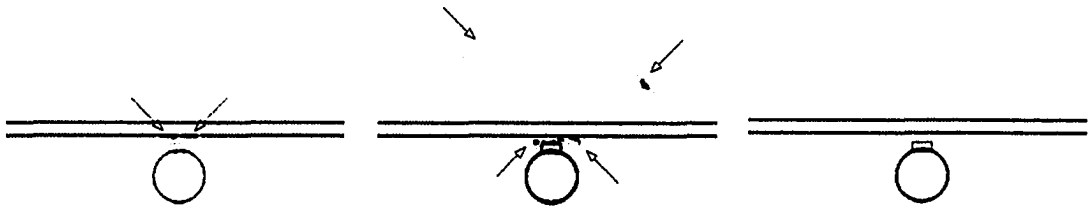
experiments show how the sonar and IR proximity sensor readings compliment the stereo camera sensor readings.

Experiment 1 (Section 4.1.2) was designed to test the miscorrelation problem of the stereo camera, and Experiment 2 (Section 4.1.3) was focused towards investigating the effects on the stereo camera when similar colors were observed in both the foreground and the background. Experiment 3 (Section 4.1.4) tested the stereo camera under different lighting conditions, and Experiment 4 (Section 4.1.5) tested whether different sensors could correctly detect a transparent obstacle.

We conducted three additional experiments to show the advantages of using our sensor data fusion algorithm. The experiments were multi-scene sensor data fusion experiments and aimed at producing an accurate map of those scenes. In Experiment 5 (Section 4.1.6), we fused sensor readings from two types of sensors, namely the stereo camera and the ultrasonic sensor when measuring distances to a corner with several obstacles clustered together. In Experiment 6 (Section 4.1.7), we also used the stereo camera and the ultrasonic sensor, in which the robot moved through a corridor containing doors, walls and windows. Experiment 7 (Section 4.1.8) is the most complicated. Here, the robot moved through the lab while taking multiple sensor readings from many different locations. The environment contains many obstacles clustered closely together in the center.

4.1.2 Experiment 1: Uniformly Colored Regions

In this experiment, the robot was placed at a distance of 45 cm from a uniformly colored (i.e., textureless) wall. The stereo camera took one sensor reading, the sonar ring took one scan, and the IR proximity sensor array took one scan. The results of these readings are shown in Figure 69 (a), (b) and (c) respectively. The circle in the figures represents the robot position.



(a) (b) (c)

Figure 69: Sensor readings from (a) the IR proximity sensor array, (b) the sonar sensor and (c) the stereo camera (arrows show range data).

Due to a lack of texture on the wall, the stereo camera generated no sensor readings (see Figure 69(c)). The IR proximity sensor array was able to generate correct distance readings (Figure 69 (a)). The sonar sensor also obtained distance readings, but due to the close vicinity to the wall, some of the distance readings were invalid (Figure 69 (b)). This experiment revealed the deficiencies of the stereo camera and demonstrated that it is possible to overcome these deficiencies by integrating sensor readings from other heterogeneous sensors.

4.1.3 Experiment 2: Low Contrast Scene

Textures of obstacle surfaces are clues for the stereo vision correlation algorithm. They directly affect the matching results, from which distance information is generated. Theoretically, when the foreground color of an obstacle is very close to the background color, it may make it difficult for the stereo matching algorithm to make a correct correlation.

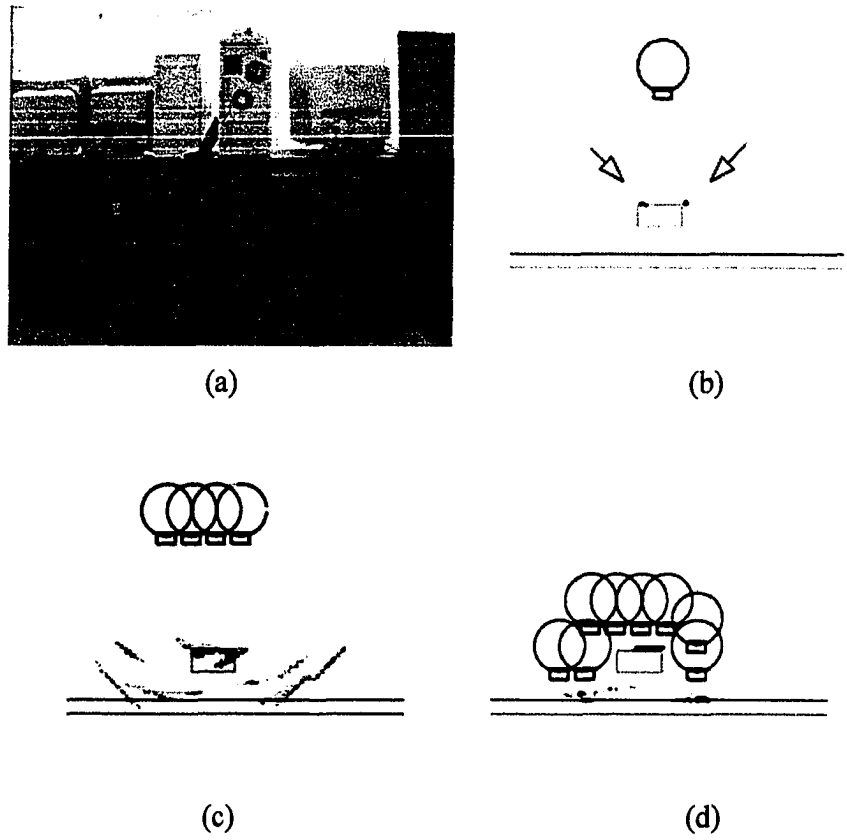


Figure 70: Snapshot (a) of a cabinet in front of a desk and readings obtained from (b) the stereo camera, (c) the sonar sensor and (d) the IR proximity sensor array.

In this experiment, we simulated that scenario by placing a small cabinet in front of a panel that was of the same color, such that it was hard to see the cabinet without

looking carefully (see Figure 70 (a)). The experimental results are shown in Figure 70 (b), (c) and (d).

The sensor readings from the stereo vision sensor are shown in Figure 70 (b); the result is better than we had expected.

It is interesting to see that although the foreground color is very similar to the background color, the stereo camera sensor was able to generate partially correct readings where the left and right edges of the cabinet are. All invalid (i.e., missing) sensor readings from the stereo camera are due to miscorrelation (i.e., the textureless region between the two edges and the textureless regions on the background desk). The sonar and the IR proximity sensor produced much more information about the environment. The fusion results of four sonar ring readings (Figure 70 (c)) shows that at close distance, sonar sensor readings can detect the location of obstacles accurately (i.e., the front panel of the cabinet is correctly detected). It provides information that is missing from the readings of the stereo camera. As we introduced previously, the drawback of a sonar[11] is its low angular resolution, and this can be seen in the results here. Even with a small number of sonar readings, the fusion results are still full of uncertainty.

The IR proximity sensor has much better angular resolution, but also due to its narrow beam width, many more sensor readings were required in order to cover the same area as a single reading from the sonar. Figure 70 (d) shows the fusion results for eight

---

[11] The sonar sensors used in the context of this thesis are general purpose ultrasonic sensors that have not been improved by special design techniques, such as sidescan or SAS.

scans by the IR sensor array and we can see that IR proximity sensor readings show high accuracy. In addition, we found that the textureless region has no negative effect on the sensor readings of the IR proximity sensor. After fusing the readings from all three sensors together, we obtain a much better estimate of the scene (see Figure 71 (a)), where the miscorrelation region is filled.

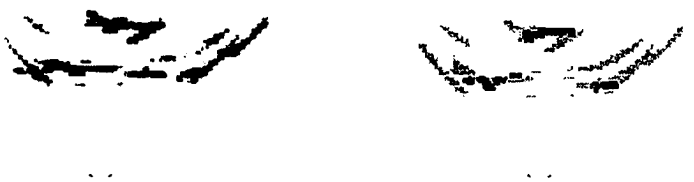


Figure 71: (a) Fusion result of Experiment 2 (b) Sensor reading classification.

The image in Figure 71 (b) shows a classification of the sensor readings where the black cells indicate agreement in readings between two or more sensors. The light grey cells indicate lower probability values, which may be due to sensor noise. Note that the sonar and IR sensor data improve the accuracy of distance measurement to the cabinet and desk.

4.1.4 Experiment 3: Low Lighting Conditions

Stereo cameras are sensitive to lighting conditions, and without sufficient light, they can produce noisy or indistinguishable images of the environment and thus generate

erroneous sensor readings. In order to verify this, another experiment was performed with obstacles arranged as shown in Figure 72 (a). This snapshot shows a typical fully lighted scene. We reduced the amount of light for this scene (see Figure 73 (a) and Figure 74 (a)). In order to investigate the effect it would have on the stereo camera measurements. The images in Figure 72 (b), Figure 73 (b) and Figure 74 (b) show the range results for this experiment under bright, dim and dark lighting conditions, respectively.

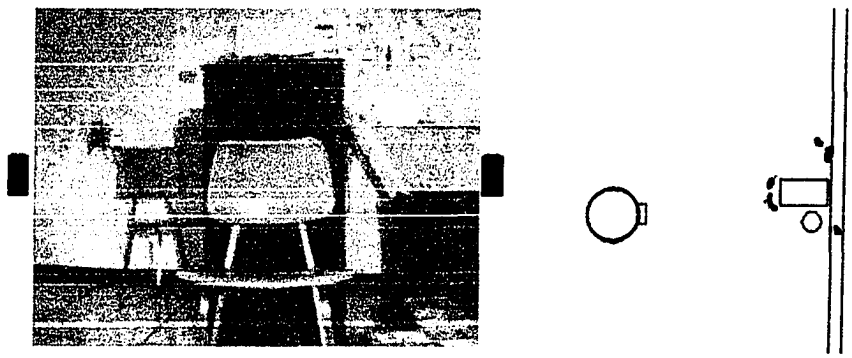
Notice that the stereo camera had difficulty detecting the garbage can, cabinet, and walls in the darker scenes, while the higher contrast chairs were still partially detectable under dim lighting conditions. Under the lower lighting conditions, the camera obtained virtually no sensor readings. From these experimental results, we verified that the stereo camera is indeed sensitive to the lighting conditions.

(a) (b)

Figure 72: (a) Snapshot under normal lighting condition (■ indicates the band position). (b) The sensor readings from the stereo camera.

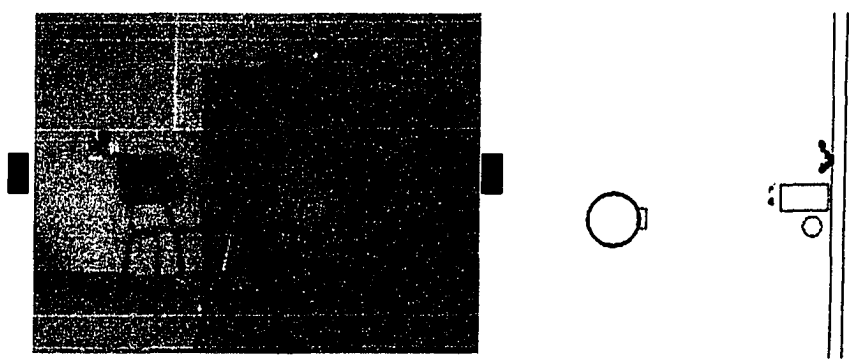
Figure 75 (a) shows the result of fusion four sonar readings in the dark lighting conditions. Notice that the walls and the cabinet are detected from the sensor readings, although are coarse due to large angle distribution. The garbage can cannot be detected at

all since it is too close to the cabinet. Figure 75 (b) shows the result of the sonar with the stereo camera. Comparing with the fusion result of the sonar sensor only, the noise generated by the sonar sensor is partially filtered out (i.e., the arcs generated by the sonar sensor are lighter in (a) than (b)). Notice that the correct sensor readings are strengthened due to the fusion process.



(a) (b)

Figure 73: (a) Snapshot under dim lighting condition (■ indicates the band position). (b) The sensor readings from the stereo camera.



(a) (b)

Figure 74: (a) Snapshot under dark lighting condition (■ indicates the band position). (b) The sensor readings from the stereo camera.

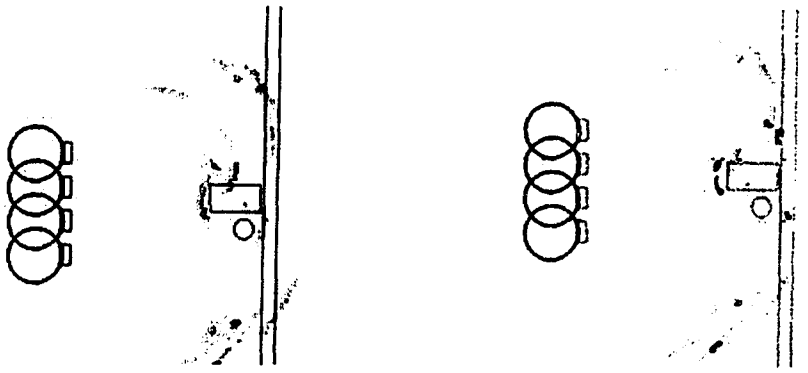


Figure 75: (a) shows the fusion result from four sonar readings and (b) is the fusion result of stereo camera with sonar.

4.1.5 Experiment 4: Transparent Regions

Since the stereo camera gets depth information visually, transparent obstacles can be hazardous to robots that rely solely on stereo camera sensors. This experiment investigated the ability of each kind of sensor to detect transparent obstacles.

The image in Figure 76 (a) shows the test scene which contains a large window overlooking a stairwell and an open door into that stairwell. A cabinet was placed in the stairwell to provide additional complexity to the test scenario. The robot was moved to various locations parallel to the scene and at each location, eight sensor readings were taken from each of the three sensors.

The sonar readings (Figure 76 (c)) show that the window was detected, although the angular variation inherent to the sonar's ring arrangement was clearly evident. The IR sensor data (Figure 76 (d)) was also affected by the transparent window, as shown by the incorrectly detected position of the railing and the cabinet.

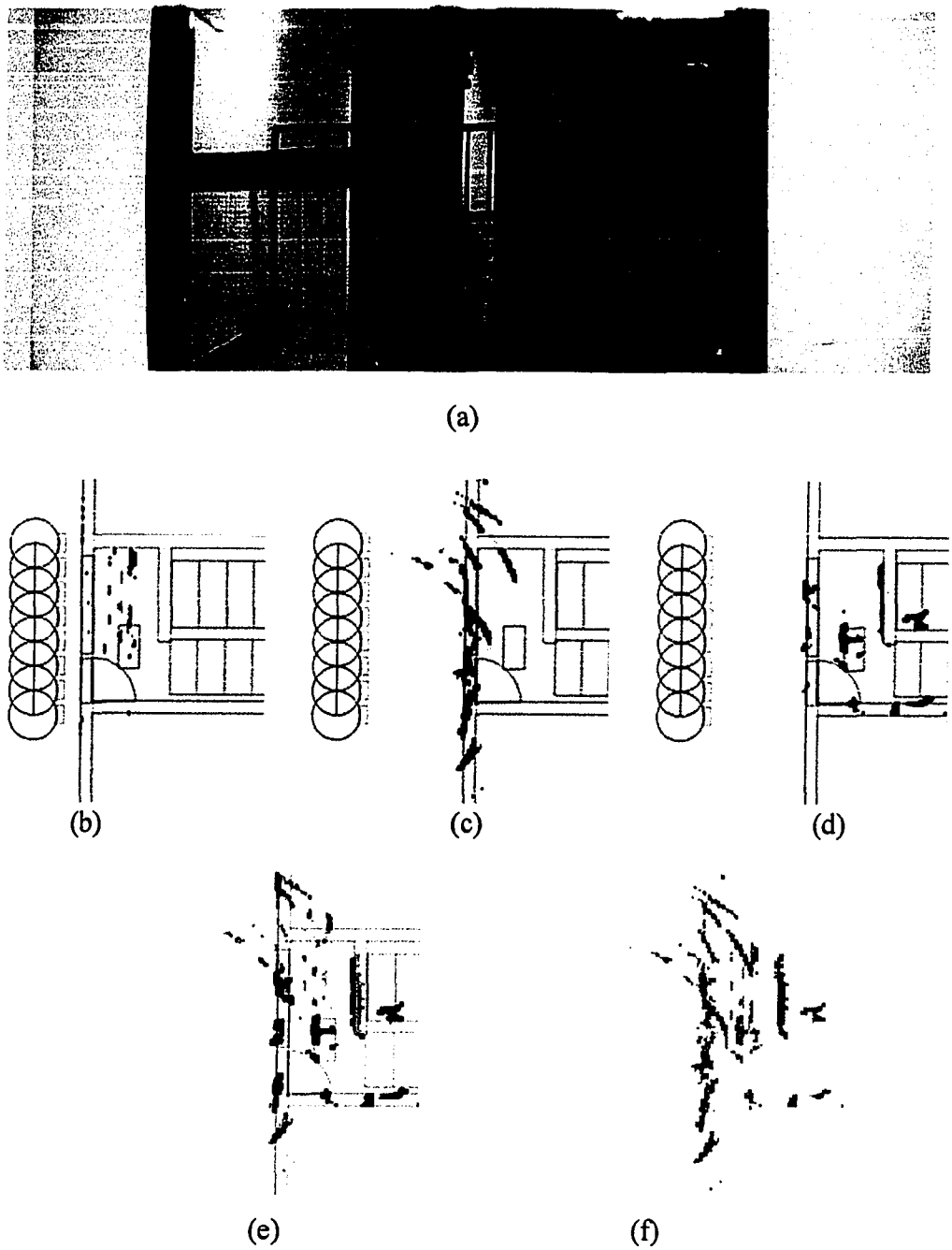


Figure 76: (a) The Panoramic view of experiment 4. (b) Sensor readings from the IR proximity sensor array. (c) The sensor readings of sonar sensor. (d) The sensor readings of stereo camera. (e) Sensor data fusion result. (f) The result fusion result classification.

Although the IR sensor detected these features, the range data was inaccurate; this is likely due to some of the IR light having been reflected back while some passed through. By fusing the sensor readings from the stereo camera with those from the sonar, the fusion result better represents the environment (Figure 76 (e)).

In the classification image of Figure 76 (f), we can see that very few cells show agreement between two or more sensors, although most of the cells that are in agreement are along the window and are due to the availability of sonar data. It is clear from this test that without the use of sonar, the window would be virtually undetectable.

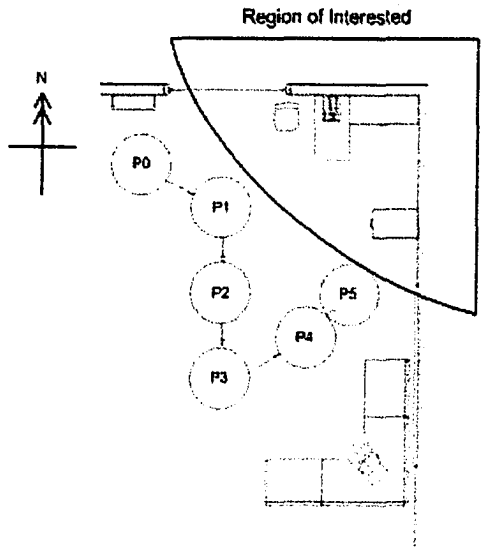
In this case, both the stereo camera and the IR proximity sensor array provide misleading sensor readings, only the sonar ring provides correct distance readings. Due to the low angular resolution feature of sonar sensors, it is difficult to generate accurate grid occupancy values using only sonar sensor data. Therefore, more sonar sensor readings are required to increase the certainty of obstacle detection. However, since the IR and the camera can see through the transparent objects, it is difficult to decide when to rely solely on sonar readings. We did not address this issue in this thesis.

4.1.6 Experiment 5: Corner

The objective of the corner experiment was to investigate whether or not our data fusion approach would generate more detail about the environment. Sometime objects may be occluded depending on the position of the robot and the direction of the sensor, so a small number of sensor readings normally can only produce partial environmental

information. By fusing a large number of sensor readings, the robot "knows" more about its surroundings.

(a)



(b)

Figure 77: (a) Panorama view of the corner in the lab, (b) Diagram of lab showing positions of the robot.

In typical indoor environments however, the layout of obstacles in a scene could be messy, and the boundaries of an obstacle may be indistinguishable. In this experiment, we researched the improvement in level of detail resulting from our sensor data fusion

approach. The objective was to show that a corner of an environment is better estimated when multiple readings are obtained from different locations.

The experiment scene is shown in Figure 77 (a). There were two filing cabinets, a box and a chair close together in the corner. The sensor could only obtain partial information about the scene from almost any given direction since the cabinet and the box enclosed a small empty space.

Sensor readings were obtained from six different positions labeled 0 to 5 in Figure 77 (b). At each position, the stereo camera took several snapshots. Since the experimental scene is the upper right corner, other sections of the lab were ignored.

The number of sensor readings taken by each sensor at each position is shown in Table 3 along with the directions used for the stereo camera (these directions are shown in degrees, where 0 degrees points north).

Position	# of readings from sonar ring	# of readings from stereo camera	Direction of readings (Degree)
0	24	6	0, 21.8, 39.375, 53.79, 71.02, 95.625
1	24	6	58, 67.85, 80.5, 89.3, 111.797, 124.453
2	24	6	26.37, 45.7, 59.06, 74.531, 87.54, 104.06
3	24	6	-29.883, 2.109, 18.98, 40.08, 70.312, 143.43
4	24	3	13, 22.5, 81.6
5	24	4	91.75, 112.85, 135.7, 149.06

Table 3: Number of sensor readings and directions.

The experimental results are shown in Figure 78. They are divided into three groups: sensor readings from the sonar sensor (i.e., (a)(b)(c)), sensor readings from the stereo camera (i.e., (d)(e)(f)), and the results of fusing the data from both sensors (i.e., (g)(h)(i)). The experiments verified the relationship between the quantity of sensor readings and the quality of the result by taking readings from:

- Only positions #0 and #3 (see Figure 78 (a)(d)(g)).
- Only positions #0, #2, #4 and #5 (see Figure 78 (b)(e)(h)).
- All 6 positions (see Figure 78 (c)(f)(i)).

The results show that there is a gap in the sensor readings from the stereo camera which is caused by the wall below the windows, which contains a big textureless region (See Figure 77(a)). This problem cannot be rectified through additional readings from the stereo camera. Figure 78 (d), (e) and (f) verifies that there is no improvement at all. On the other hand, sensor data fusion does improve the detail level of the map.

With more sensor readings, we obtained more accurate boundary features for the detectable obstacles. In Figure 78 (d), the detected outline of the small cabinet is wrong (see arrows). After fusing more sensor readings, the shape of the cabinet is improved (see arrows) and the outlines of the small cabinet, the box and the file cabinet are likewise more accurate (see arrows). We also noticed that the sensor readings from the sonar are relatively coarse (see Figure 78(a)(b)(c)), due to the low angular resolution of ultrasonic sensor. Although the resolution of sonar readings is relatively low, they played an important role in complementing the data from the stereo camera.

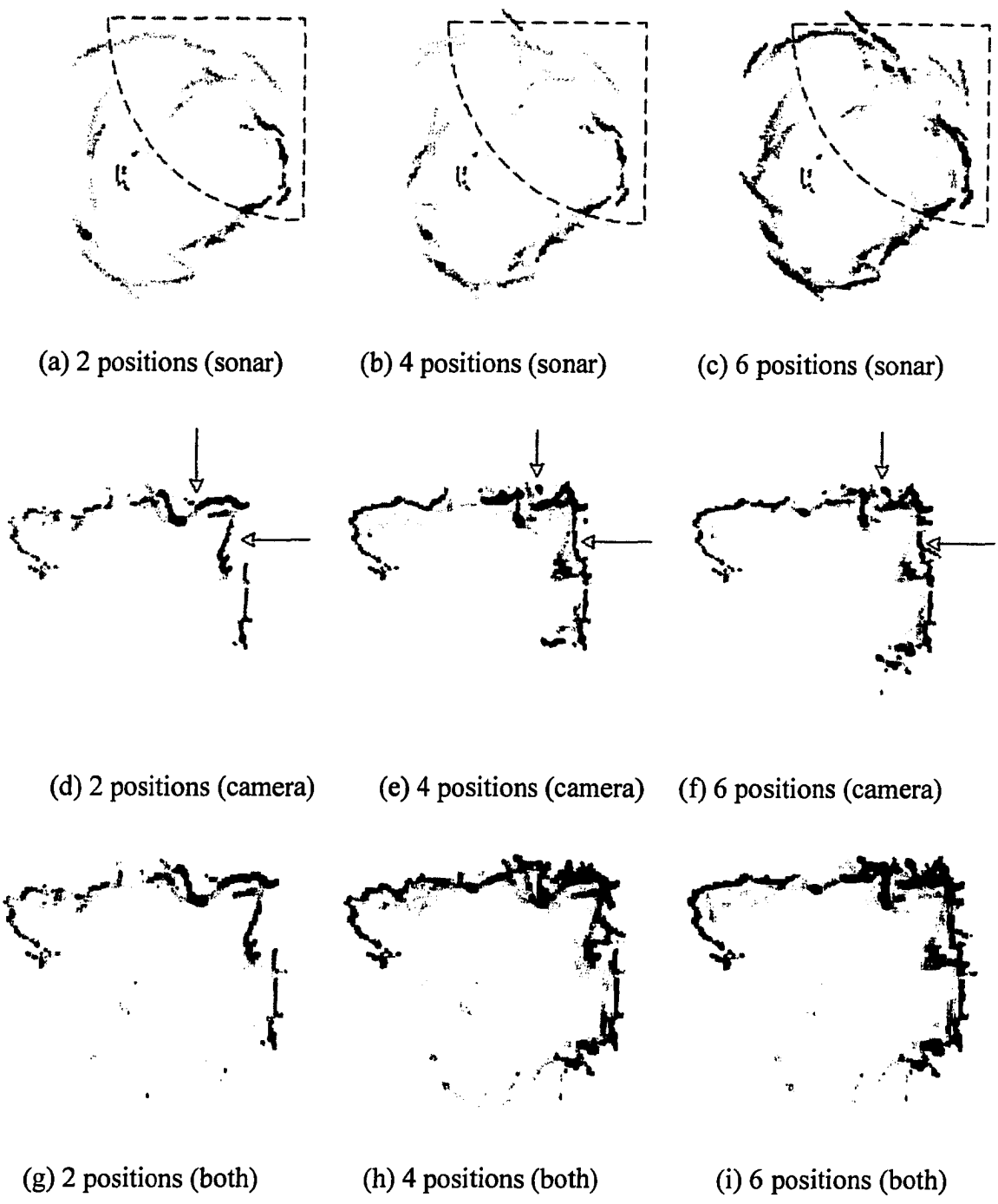


Figure 78: (a-f) Fusion results for each sensor from different positions and (g-i) the fusion results of both sensors on different positions.

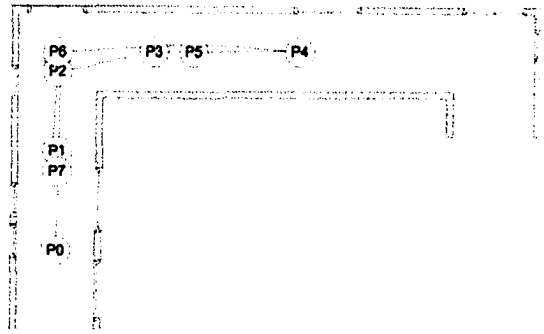
The fusion results of the two kinds of sensors are shown in Figure 78 (g), (h) and (i). The most obvious improvement is that the gaps in the results of stereo camera are filled by sensor readings from the sonar. It is also evident that multiple readings from different locations are essential for accurate mapping.

4.1.7 Experiment 6: Corridor Map

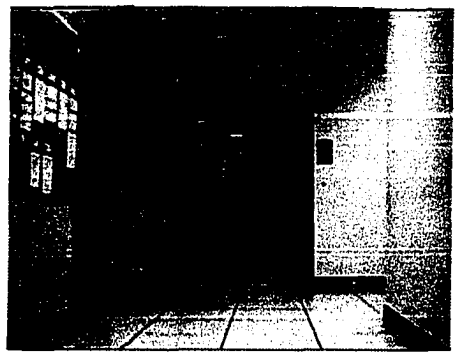
In the previous experiment, we observed that in order to generate an accurate map, the robot must collect data from different positions. As the positions, from which the robot collects data increase, the quality of the output map increases. However, the movement of the robot from one position to another position leads to another problem: how can the robot know its new position and orientation accurately? Recall that, dead-reckoning error can accumulate and this leads to uncertainties with respect to the robot position. As the number of the robot positions increase, the robot's position certainty goes down. In this experiment, the robot moved through a long "L" shaped corridor. The robot moved more than twenty meters and obtained sensor readings from eight positions. The objective was to determine the impact that robot movement has on the sensor data fusion procedure.

The scene of this experiment is shown in Figure 79 (a). Although there were not many obstacles in the corridor, the textureless walls became a big challenge for the stereo camera, and there was a glass door in the middle of the corridor, which the stereo camera

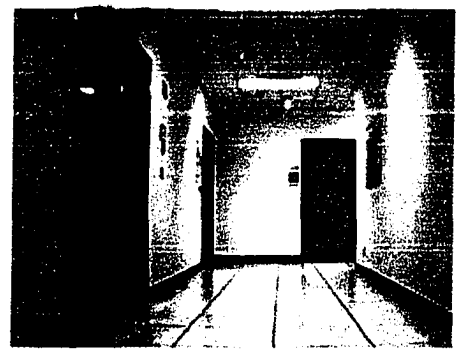
couldn't "see" (see Figure 79 (b), (c)). In the experiment, the robot followed a planned path and returned to the starting position (see Figure 79 (a)).



(a)



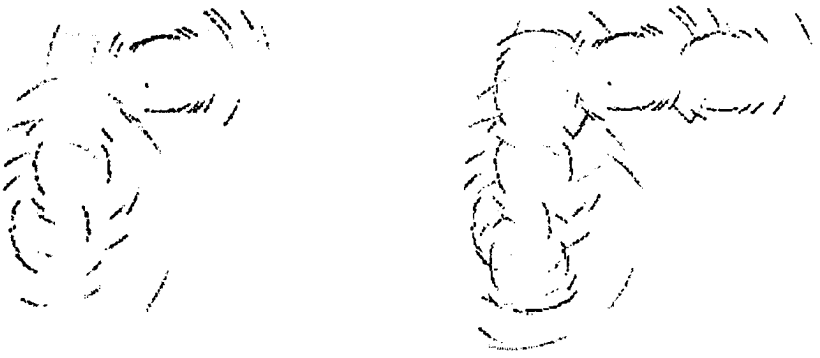
(b)



(c)

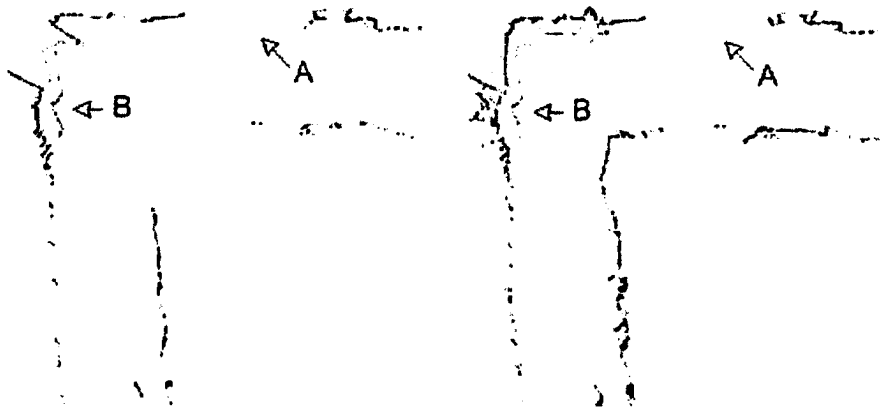
Figure 79: Experiment setup of experiment 6. (a) The moving path of the robot. (b) The left wing image of the corridor. (c) The right wing image of the corridor.

As evident in Figure 80, Figure 81 and Figure 82, the vertical portion of the corridor seems skewed. This error was due to dead reckoning errors from wheel rotation sensing within the robot. Recall that our robot does localization depending solely on its internal sensors (encoders). Since there was no external localization mechanism, the dead reckoning error accumulated. The problem of robot localization was not within the scope of this thesis. Here we only focus on its effects on our sensor data fusion approach.



(a) 4 positions (sonar) (b) 8 positions (sonar)

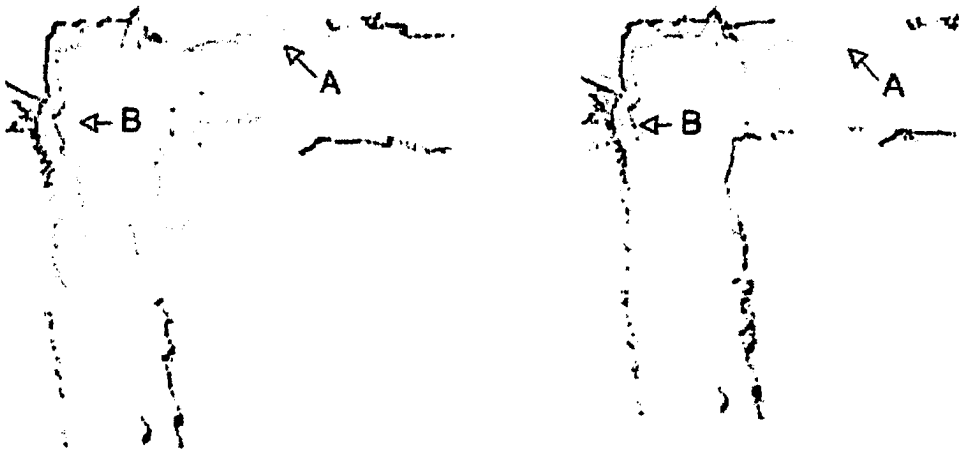
Figure 80: Fusion results of the sonar sensor on 4 positions and 8 positions.



(a) 4 positions (camera) (b) 8 positions (camera)

Figure 81: Fusion results of the stereo camera on 4 positions and 8 positions.

Figure 81 shows the experimental results for the stereo camera, in which the miscorrelation problem, due to the textureless walls, makes the sensor readings of the stereo camera discontinuous. There were some portions of the walls, which did not have accurate sensor readings, although the result improved with more fused sensor readings.



(a) 4 positions (both) (b) 8 positions (both)

Figure 82: Fusion results of all sensors on 4 positions and on 8 positions.

Figure 80 shows that most of the sonar readings are correct and accurate but the results are coarse. The fusion results from both sensors are shown in Figure 82. The gaps in the sensor readings from the stereo camera have been filled. In this experiment, the robot only takes sensor readings from eight positions, which is not enough for the sonar to generate a high belief value in the grid. If the sonar had taken more sensor readings, those gaps would have been filled more thoroughly.

Let us pay special attention to the spot A and B in Figure 81 and Figure 82. There is a stairway with a big glass window at spot A and a doorway at spot B. Notices that spot A is not well detected in all fusion results. Due to the sonar's low angular resolution, at long distance, the probability of each grid cell is low and we only took a limited number of readings from the sonar ring, so the final fusion result is not improved significantly. The sensor readings from the stereo camera and the fusion results are noisy at spot B. The

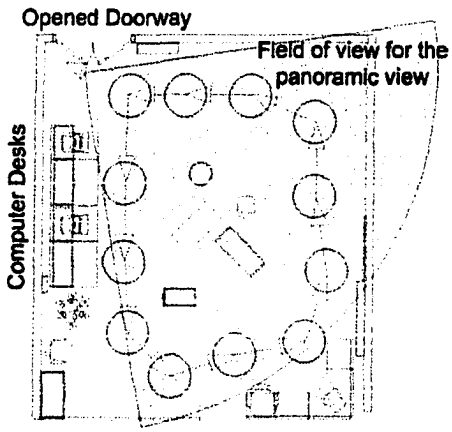
reason is that the door at spot B was sometimes opened and closed at other times during the experiment. The stereo camera did detect obstacle behind the door when the door is opened. Since once the robot detected a closed door, it did not update the grid behind the door and the robot did not move through the doorway, there is no opportunity to update those grid cells behind the doorway.

4.1.8 Experiment 7: Full Lab Map

In this experiment, we used all three kinds of sensors and fused their sensor readings. Due to the limited space of the lab, the robot sometimes moved very close to the obstacles. At that distance, the stereo camera could not get very reliable sensor readings due to binocular parallax. The IR sensor array was added to provide additional information about the environment. When the robot was too close to the obstacle, the stereo camera was disabled and the system started fusing sensor readings from the sonar and the IR sensor. The objective of this experiment was to test the sensor selection strategy and the improvement on the detail level of the fusion process. In this experiment, obstacles were placed close to one another, so that if the detail level of the fusion result had not been high enough, the robot would not have been able to detect the space enclosed by those obstacles.

We placed five obstacles within a four square meter space in the middle of the lab (see Figure 81 (b)), and the robot moved around these obstacles to get sensor readings. The distance between these obstacles was approximately between 10 and 30 cm. The layout of the experiment setup is shown in Figure 81 (a). The robot moved along a fixed

path and took sensor readings at twelve positions. The total distance traveled by the robot was 22.7 meters. The sonar ring took 12 readings, the stereo camera took 70 readings and IR sensor took 55 readings.



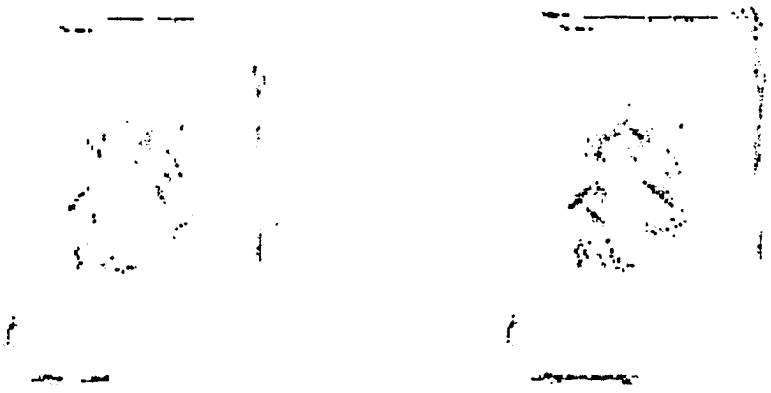
(a)

(b)

Figure 83: (a) shows the robot path and the panoramic view of experiment #7 is shown in (b).

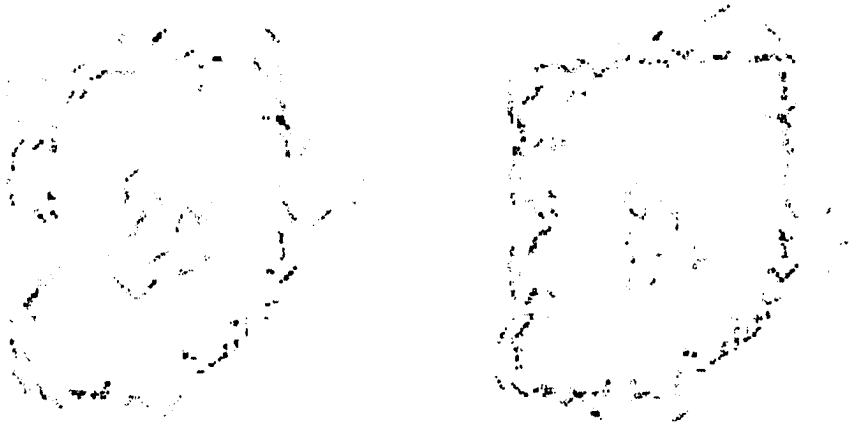
The fusion results for the IR sensor are shown in Figure 84. We first notice that the IR sensor does not detect the leftmost portion of the lab, which is due to the short detection range of the sensor. There are two computer desks at the left, and the height of

the IR sensors array is lower than the desktop; thus, they cannot detect the desktop, nor can they detect the wall behind the desk, as the wall is beyond the detection range of the IR sensor. The IR data shows a very precise reading of the walls, with some missing data on the left and bottom right due to desks that prevented the robot from getting close enough to the walls to take readings.



(a) 6 positions (IR) (b) 12 positions

Figure 84: Fusion results of the IR sensor array from 6 positions and 12 positions.



(c) 6 positions (d) 12 positions

Figure 85: Fusion results of the sonar sensor array from 6 positions and 12 positions.

The sonar data (Figure 85) provided a reasonable outline of the environment and performed better than the camera in the uniform regions. It did however produce more spurious readings and had some difficulty with the finer details of the objects in the center of the lab.

In the stereo camera data (Figure 87), the gaps in the top and right regions are caused by the large regions of uniform color on the walls in the lab. The top left gap represents an open doorway, and the "splatter" effect is caused by range information to objects in the corridor outside the lab. The bottom of the grid has a splatter effect as well, since there are additional obstacles beyond the measured portion of our lab. The reason of "splatter" effect is not due to obstacles beyond themselves, but due to the position and the view angle when detecting them. For example, the "splatters" on the top are sensor readings of obstacles on the corridor since the door was opened during the experiment. Notices that the robot was only able to detect obstacle using the stereo camera on position 9 and 10 (see Figure 86). It did not take sensor readings on that direction due to its moving direction or blocked by other obstacles. That is the reason why that "splatter" trends to the left side. The major "splatter" on the bottom happened due to similar reason. There is a big gap between those computer desks and the robot could detect scene behind the gap only on position 3, 4 and 5 (see Figure 86). On position 6, we did not take a sensor reading on that direction due to the close distance. As the result, the "splatter" on the bottom trends to the right side. Other "splatters" are due to the miscorrelation or the error correlation. We also notice that there is no "splatter" effect in the center of the scene

since the robot moved around these obstacles and the data fusion algorithm eliminated those "splatters".

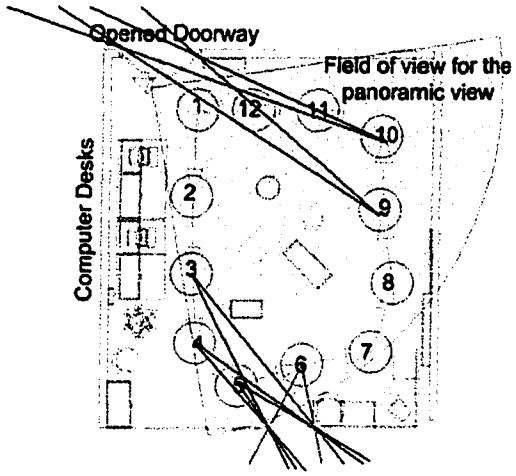
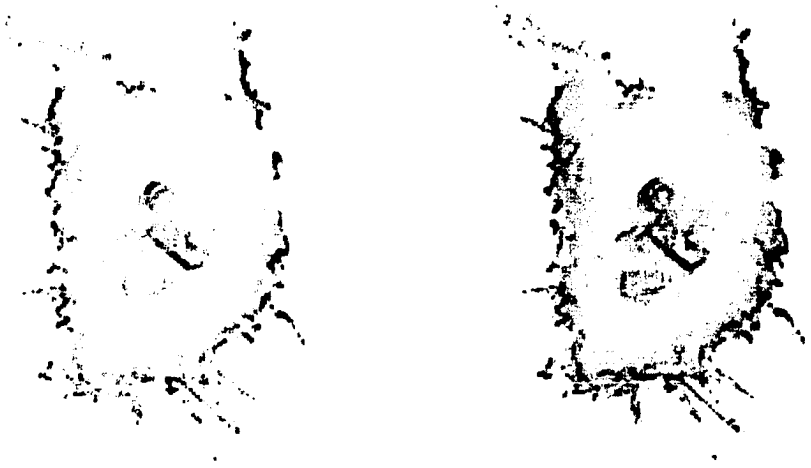


Figure 86: The reason for "splatter" effect.

The evidence grid obtained through the data fusion as shown in the combined image (Figure 88), clearly indicates the advantage of the sonar and IR data sets. Notice how much more precise the map is along the top and right borders where the IR data helped to refine the missing stereo camera data. Also, notice how the three sensors combined provide a more accurate shape estimate for the inner obstacles.

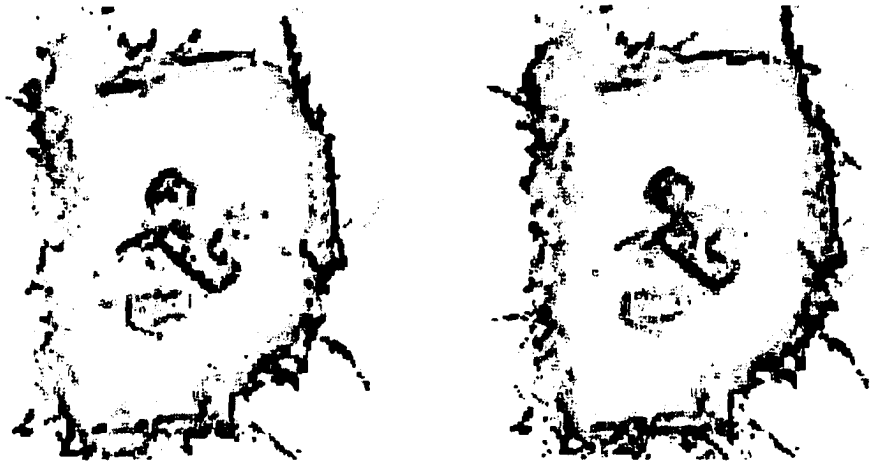
A more thorough analysis of the sensor data helps us to see the benefits of having more than two sensors. One simple measure of the benefit is to examine the evidence grid cells for which each sensor produced range readings. Figure 89(a) shows an image that classifies grid cells for the full lab test according to the number of sensors that gave readings for that cell. The light grey cells represent 59.7% of the total readings in which only one sensor produced a range value. There were 34.2% of the readings confirmed by two sensors, shown in medium grey. All three sensors confirmed only 6.2% of the

readings; these readings are shown in black. This classification indicates that sensors have different characteristics, and shows that multiple heterogeneous sensors can produce complimentary data that will commonly detect environmental features.



(e) 6 positions (f) 12 positions

Figure 87: Fusion results of the stereo camera sensor from 6 positions and 12 positions.



(g) 6 positions (h) 12 positions

Figure 88: Fusion results of all sensors from 6 positions 12 positions.

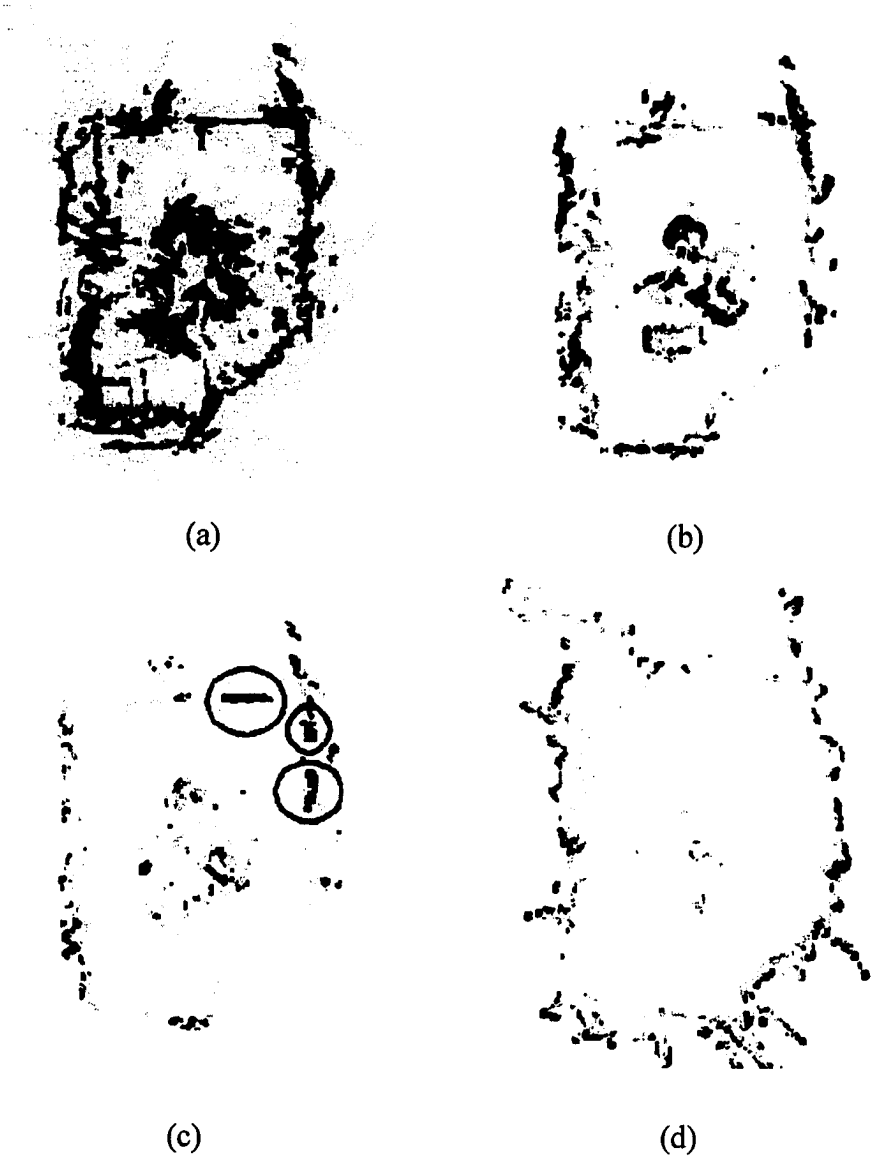


Figure 89: (a) Grid cell weight classification, (b) Combined data with single sensor readings removed, (c) Vision data combined with Sonar/IR data showing where gaps were filled in, (d) Noise that was eliminated from the vision data.

Figure 89(b) depicts the map represented by the combined data fusion of Figure 88 (h), where all single sensor classification data readings have been removed. Thus, about

60.4% of the data is discarded as untrustworthy. Examining the data from the stereo camera, Figure 89(c) shows (as circled) where the sonar and IR range data were able to fill in the gaps that were unavailable from the vision data due to regions of uniform color. These regions represented a significant 12% of the lab's border, which would have otherwise been void of range data if only the stereo camera data was used. Lastly, Figure 89(d) shows the "noise" (a significant 48.8%) that was eliminated from the stereo camera data through the data fusion process.

4.1.9 Order of Data Fusion

In our work, there are three different sensor sources and our data fusion strategy fuses them in following order: Sonar, Camera and IR.

Theoretically, using a different data fusion order during Kalman filter-based sensor data fusion results in the same occupancy grid values. To see this, let Z_1 and Z_2 be two independent sensor readings and their mean square errors are σ_1^2 and σ_2^2.

The fusion result of Z_1+Z_2 is:

$$X(t+1) = Z_1(t) + \frac{\sigma_1^2(t)}{\sigma_1^2(t)+\sigma_2^2} \bullet (Z_2(t+1) - Z_1(t)) \tag{45}$$

$$\begin{aligned}(\sigma_2^2(t)+\sigma_1^2)X(t+1) &= \sigma_1^2(t)Z_1(t) + \sigma_2^2 Z_1(t) + \sigma_1^2(t)Z_2(t+1) - \sigma_1^2(t)Z_1(t) \\ &= \sigma_2^2 Z_1(t) + \sigma_1^2(t)Z_2(t+1)\end{aligned} \tag{46}$$

$$\sigma^2(t+1) = (1 - \frac{\sigma_1^2(t)}{\sigma_1^2(t)+\sigma_2^2}) \bullet \sigma_1^2(t) = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2(t)+\sigma_2^2} \tag{47}$$

The fusion result of Z_2+Z_1 is:

$$X(t+1) = Z_2(t) + \frac{\sigma_2^2(t)}{\sigma_2^2(t)+\sigma_1^2} \bullet (Z_1(t+1) - Z_2(t)) \quad \textbf{(48)}$$

$$(\sigma_2^2(t)+\sigma_1^2)X(t+1) = \sigma_2^2(t)Z_2(t) + \sigma_1^2 Z_2(t) + \sigma_2^2(t)Z_1(t+1) - \sigma_2^2(t)Z_2(t)$$
$$= \sigma_1^2 Z_2(t) + \sigma_2^2(t)Z_1(t+1) \quad \textbf{(49)}$$

$$\sigma^2(t+1) = (1 - \frac{\sigma_2^2(t)}{\sigma_2^2(t)+\sigma_1^2}) \bullet \sigma_2^2(t) = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2(t)+\sigma_2^2} \quad \textbf{(50)}$$

Since the value of $Z_2(t)$ is equal to $Z_2(t+1)$ and $Z_1(t)$ is equal to $Z_1(t+1)$, the fusion result of Z_1 and Z_2 is equal to Z_2 and Z_1 (see Equation 46, 47, 49 and 50).

Chapter 5 High-level Map Building

An important objective of sensor data fusion and mapping is to assist in robot navigation and exploration. An evidence grid map maybe too detailed for robot navigation and path planning. In addition, many navigation and path planning algorithms are based on vector-based geometry where objects and obstacles are defined as geometric entities (e.g., polygons, and polyhedrons). As we introduced in Section 2.4, the evidence grid map is a to low-level map representation, which contains more information, but takes more time to process and more storage space. High-level applications such as a navigation or exploration module, normally uses a simpler map representation which contains information for making navigation or exploration decisions. Thus, generating a high-level map from a low-level map such as an evidence grid is often necessary.

Our high-level map building algorithm is inspired by Kovesi's line segmentation algorithm [32] which is based on the canny detection algorithm and its workflow is shown in Figure 91 (a). The input of his line segmentation algorithm is a matrix (an evidence grid map) and the output is a set of simplified line segments. The algorithm first applies Canny's edge detection algorithm to detect all edges in the input data (see Figure 90 (a) and (b)) and then applies a connected component label algorithm to segment these edges (see Figure 90 (c), different line segments are in different colors). For each edge, a line simplification algorithm is applied to reduce the number of line segments based on

the length of the line segment and the angle formed by two linking line segments (see Figure 90 (d)).

Figure 90: (a) The source image (b) the result of Canny's edge detection algorithm (c) the line segmentation result (d) the line simplification result.

Kovesi's algorithm works well in image processing, but it cannot be used as a high-level map building algorithm directly. Canny's edge detection algorithm detects edges based on the variance among adjacent grid values, the larger the difference the stronger the edge. The algorithm does not take into account the meaning of those grid values which result in detecting too many edges. After applying a line simplification algorithm, the outputs of the algorithm usually consists of either too many line segments (i.e., small obstacles are properly represented but too many line segments for large obstacles) or too few line segments (i.e., large obstacles are properly represented but small obstacles are ignored). In order to address this problem, we have to consider the meaning of the grid value.

After sensor data fusion, the occupancy probability value in each grid differs from grid cell to grid cell. As a result, small value variances are also considered as edges in the edge detection algorithm. Figure 92 (a) shows a small portion of a global map (in order to

show detail, this portion is zoomed 1700%). The values in those grids are various (in different grey levels) but most of them are various in small ranges. If we input this map portion for edge detection directly, many unwanted edges are detected (Figure 92 (b)).

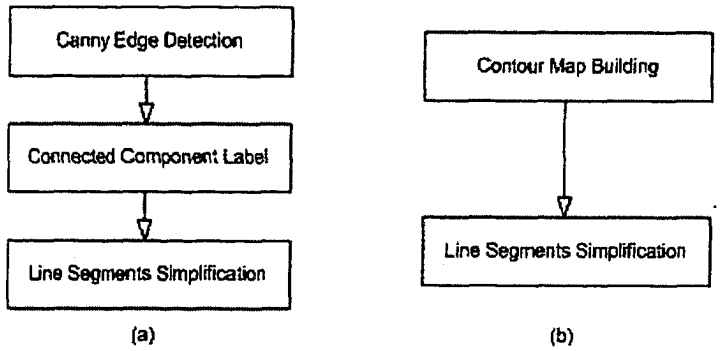
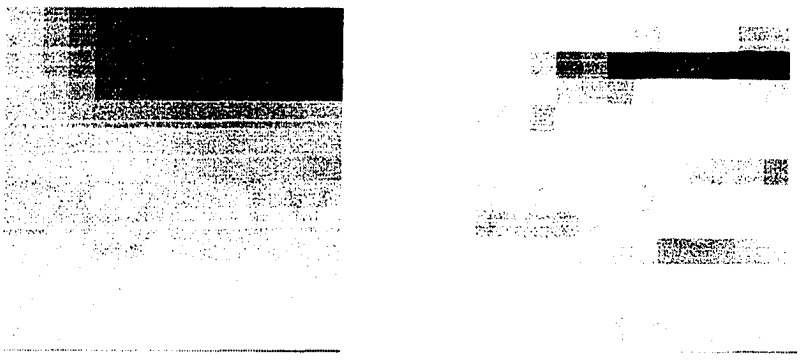


Figure 91: (a) Kovesi's line segmentation algorithm (b) our high-level map build algorithm approach hierarchy.

We modified Kovesi's original algorithm and developed a new high-level map building algorithm. The major modification is to replace Canny's edge detection algorithm and the connected component label algorithm with a contour map building algorithm. Contour building can also be considered a type of edge detection algorithm which detects the edge of constant value. In our high-level map building, it is a probability value (e.g., $P_{occupied} = 0.7$). In comparing to the edge detection-based line segmentation algorithm, contour-based line segmentation provides attractive features for high-level map building from an evidence grid map. We can select to detect those edges that we are interested with by setting levels and thresholds of contour map building. In addition, a contour map has actual meaning in high-level map building. For example, if we consider grids with value $P_{occupied} > 0.5$ as occupied and grids with value $P_{occupied} > 0.8$

as feature points for robot navigation, we can build contours for grids with value $P_{occupied}$ = 0.5 and value $P_{occupied}$ = 0.8 respectively.



(a) (b)

Figure 92: (a) A portion of the global map and (b) the edge detection result of the portion.

(a) (b)

Figure 93: (a) A portion of the global map after contour algorithm and rounding and (b) the edge detection result of the portion after contour.

The contour map building works as a multiple pass threshold function which separates the global map into several segments. By selecting proper contour values, we can use these contours to approximate the boundary of obstacles. Figure 92 (b) shows the

contour map result of Figure 92 (a) and Figure 84 (b) shows the contours obtained from Figure 84 (a). Comparing Figure 92(b) and Figure 84 (b), the noises due to small variance in probability values are eliminated and the contours better represent the shapes of the obstacles.

The workflow of our high-level map building algorithm is simpler (Figure 91 (b)) and there are two steps: contour map building and the line segment simplification. We applied the contour map building algorithm available in Matlab®: C = contourf(Z,v). The "contourf" calculates a filled contour map from matrix Z and fills the areas between the contours using a constant value. It generates a contour map of matrix Z with contour levels at the values specified in vector v and all grids on the contours are stored in matrix C (see Figure 94 (d)).

The line segment simplification algorithm takes the result of the contourf (matrix C) as input and generates simplified line segments. It considers the length of the line segments and angle formed by two line segments. The algorithm has three parameters:

- *tol*: Maximum deviation from straight line before a segment is broken in two (measured in pixels).
- *Angtol*: Angle tolerance used when attempting to merge line segments (radians).

Linkrad: Maximum distance between ends of line segments for segments to be eligible for linking (cells).

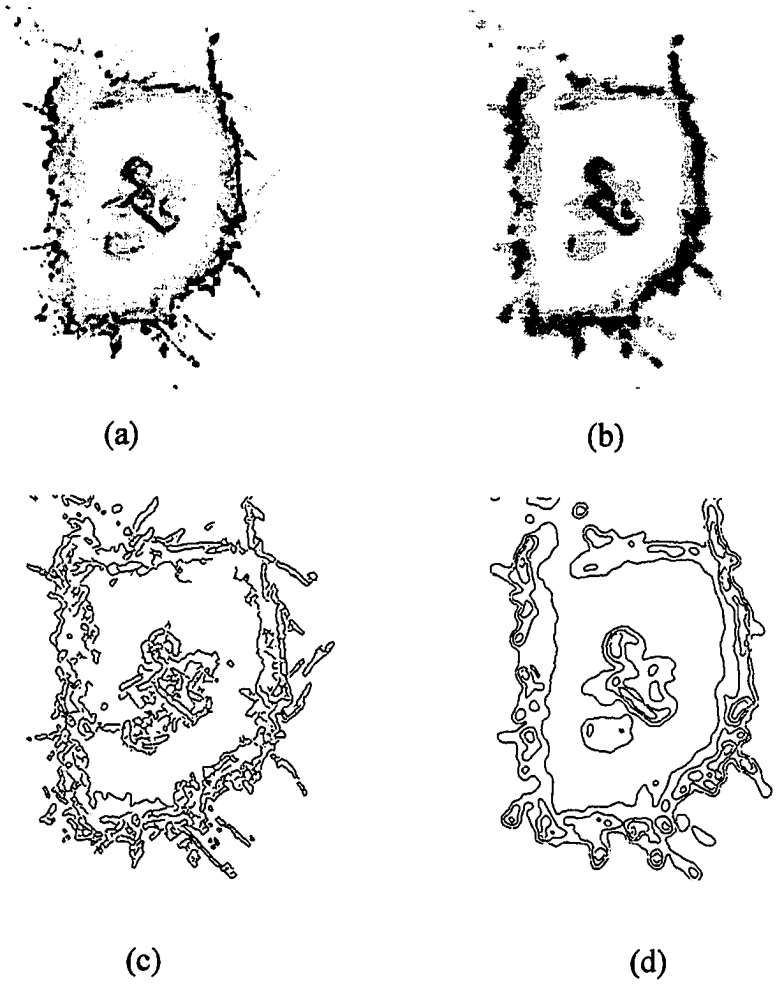


Figure 94: (a) The original evidence map and (b) the contour mapping result (c) result of applying Canny's edge detection algorithm (d) our modified algorithm using contour map building.

The line segmentation algorithm in our approach is based on least squares fitting (LSF)[12]. The algorithm evaluates several conditions when doing line simplification. Firstly, the maximum deviation from the original edges. Secondly, the length of the line

[12] A mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the offsets ("the residuals") of the points from the curve.

segments (i.e., discard all line segments less then 10 pixels). Finally, the angular difference of two adjacent line segments (i.e., if the angular difference of two line segments less then 0.05 radians, then merge them).

The pseudo code for the line simplification is shown in Figure 95.

```
Function [linelist] = lineseg(edgelist, tol, angtol, linkrad)
FOR (i ← 1 TO # contours)
    x ← contour[i].x[]; // x is array of x coordinate in the contour
    y ← contour[i].y[]; // y is array of y coordinate in the contour
    first ← 1; / first point in the edge
    last ← length(x); // last point in the edge.
    WHILE (first< last)
     maxdev,posmaxdev ← maxlinedev(x, y); //Find value & position of max dev.
     WHILE (maxdev > tol)  // While deviation is > tol
          last ← posmaxdev + first - 1;  //Shorten line to point of max dev
          [m,d] ← maxlinedev(x, y);
     END WHILE
     Nline ← Nline+1; // Record line segment.
     linelist(Nline) ← [x(first), y(first), x(last), y(last)];
     first ← last+1;   // Reset first and last.
     last ← length(x);
    END WHILE
END FOR
FOR (i ← 1 TO # of contours)
    FOR (j ← 1 TO # of line segments in contour[i] – 1)
      deltaAng ← angle between line[i][j], line[i][j+1];
      merged ← tested for end point proximity;
```

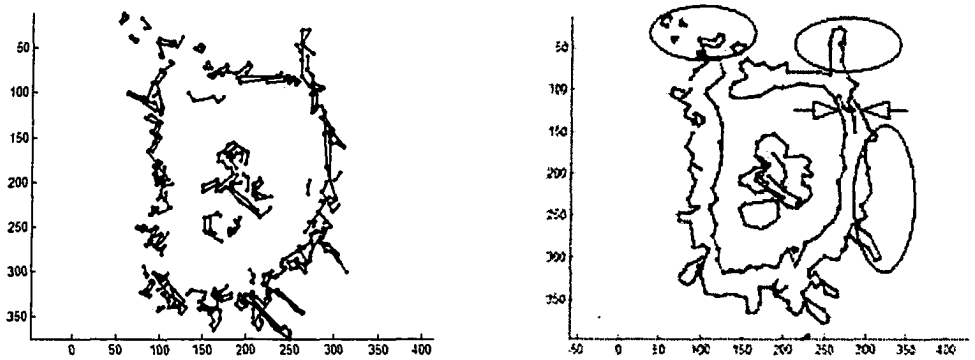
```
        IF (within linkrad) THEN
            merged = true;
        ELSE
            merge false;
        END IF
        IF (merged and deltaAng < angtol) THEN
            Merge segment j and j + 1;
        END IF
    END FOR
END FOR
```

Figure 95: Pseudo code of our line simplification algorithm.

The performance of the algorithm depends on the result of the contour map building and the setup of parameters for the function. Improper setup of the function parameters can result in too many line segments or too few line segments. Experiments need to be conducted to find out the proper parameters for the line segmentation simplification function in an indoor structured environment.

We applied Kovesi's original algorithm and our modified algorithm on the global map generated in experiment #6 (see Figure 94 (a)). Figure 96 (a) shows the result of Kovesi's original algorithm and Figure 96 (b) shows the result of the modified algorithm. After applying the contour map building technique, the line segment simplification algorithm only deals with those edges which we are interested in and the unwanted line segments decreases dramatically. The result of our high-level map building algorithm represents the environment better (see Figure 96 (b)).

Notice that the vector-based map built in Figure 96 (b) is not the actual outline of the obstacles but a border of all grid cells with probability larger then 0.5 which are the "dangerous zone" for the robot navigation. The shapes of obstacles in the center of the scene are not represented exactly correct but with some safety cushion. We also notice that the shapes of the surrounding walls are much "thicker" (arrows in Figure 96 (b)). Since our robot only detected obstacles inside the lab, the outer border can not be rectified by sensor readings. A more complete mapping route can minimize the "thicker" effect. For example, if we merge this map with the map generated in experiment 6, the "thick" wall on the top can be rectified (circles in Figure 96 (b)).



(a) (b)

Figure 96: (a) Line simplification result of Kovosi's original algorithm, which based on edge detection and (b) our line simplification algorithm on the contour map.

Chapter 6 Conclusion and Future Work

The objective of the thesis was to build a heterogeneous sensor system for a robot and use sensor fusion techniques to improve the reliability and accuracy of sensor readings. In order to accomplish that a robot needed to be constructed, which was an important part of this thesis. Knowing the limits and step-wise-refinement of the robot will make long-term contributions to further research work in regards to map creation and robot correlation.

The robot was equipped with three sensors: an IR proximity sensor array for short range, a sonar ring for moderate range and a stereo vision sensor for long range. With the exception of the sonar ring, which was ready for usage, two other sensors were chosen in order to research the complementary properties of sensor data fusion. Our research focused on properties of sensor readings (i.e., range, accuracy and robustness) and obstacle properties (i.e., texture, color, transparency level).

The methodology of our research included research and experiments as follows:

(i) A comprehensive study of properties of each sensor and an analysis of the advantages and disadvantages of each.

(ii) Design, experimentation and analysis of the quality of our data fusion algorithm.

(iii) Design and analysis of our high-level map creation algorithm.

In harmony with our methodology, we conducted a large number of experiments to study the properties of each sensor which was followed by a set of experiments for data fusion.

Our contributions can be summarized as follows:

(i) Construction of a robot with three sensors for sensor data fusion.

(ii) Demonstration of the benefits of three sensors over two sensors.

(iii) Improvement in the quality of high-level map creation (vector map) from a low-level occupancy grid map.

When compared with traditional sensor data fusion approaches that use only two sensors, sensor data fusion approaches that use three sensors have obvious advantages. For example, three sensors provide more information than two sensors, which provide additional information about the environment when voting for the majority. Three-sensor approaches provide a more complete range coverage (i.e., we have sensors for short, moderate and long ranges). In addition, three-sensor approaches provide more stable sensor readings when detecting obstacles with especial textures. For example, when detecting textureless regions, a two-sensor system (i.e., a stereo camera and a sonar) can provide one type of sensor readings only since the stereo camera can not get valid sensor readings at all. Under the same scenario, a three-sensor system (i.e., a stereo camera, a sonar and a IR proximity sensor) still can provide two types of sensor readings for the sensor data fusion. A robot is normally equipped with a high accuracy sensor as its main sensor and lower accuracy sensors as secondary sensors. When main sensors can not

generate valid sensor readings due to their pitfalls, a three-sensor system provides two additional sensor reading sources and a two-sensor system only provide one.

The data fusion results are used to generate high-level vector-based maps using our high-level map creation algorithm. Since a vector-based map is much more abstract than a grid-based map, it is very important to keep the useful information and filter out noise. Our high-level map creation algorithm (based on contour map building) is suitable for generating vector-based maps from occupancy grid maps since the robot can generate the vector-based maps based on different levels of probability of a grid cell being occupied.

We designed and applied one approach to improve the sensor reading quality of the IR proximity sensor, four different approaches for the sonar ring and four approaches for the stereo vision sensor to improve the sensor reading quality (12.1% in our "averaging a band of data" approach).

We investigated advantages of multiple sensor data fusion for a robot mapping unknown environments. We focused on the capability of the sensors to detect the range to obstacles which are uniform in appearance, objects of different materials (such as transparent objects) and objects under different conditions (e.g., low lighting). Through our experiments presented here, we have clearly shown the limited abilities of stereo camera vision systems in certain scenarios. The experiments indicate that low cost sonar and infrared proximity sensors can be used in combination to accurately "fill in" the missing vision sensor information (up to 12% of the border in our full lab test) and also to fine-tune existing vision sensor range data by discarding noise (48.8% in our full lab test).

There are many interesting extensions to the current work. We did not yet have an opportunity to test the performance of the stereo camera on reflective surfaces such as mirrors or stainless steel surfaces. Currently all experiments were conducted under a static environment. Theoretically, our sensor data fusion algorithm can adapt to a dynamic environment by adjusting the Kalman gain. More experiments can be done under the dynamic environment setting.

We believe that the hardware and software framework of the K2A Robot could serve as a solid basis for further studies.

Bibliography

[1] K. H. Afkhamie, Z-Q. Luo, and K.M. Wong, "Interior Point Least Squares Estimation: Exploiting Transient Convergence In MMSE Decision-Feedback Equalization", IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, pp 5-8, 2001.

[2] D. Avots, E. Lim, R. Thibaux, and S. Thrun, "A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments", IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 335–363, 2002.

[3] N. Ayache and B. Faverjon, "Efficient registration of stereo images by matching graph descriptions of edge segments", International Journal of Computer Vision, vol. 1, pp. 107-131, 1987.

[4] A. Bennett and J. J. Leonard. "A behavior-based approach to adaptive feature mapping with autonomous underwater vehicles". IEEE Journal of Oceanic Engineering, Vol. 25, No. 2, pages 213-226, April, 2000.

[5] J.G. Balchen, F. Dessen, "Traditional and Non-Traditional Robotic Sensors", Ed. T.C. Henderson, Springer-Verlag, Berlin Heidelberg, 1990.

[6] H. Baltzakis, A. Argyros and P. Trahanias, "Fusion of laser and visual data for robot motion planning and collision avoidance", Machine Vision and Applications, vol. 15. pp. 92–100, 2003.

[7] J.O. Berger, "Statistical Decision Theory and Bayesian Analysis", 2nd ed. New York: Springer-Verlag,1985.

[8] D. P. Bertsekas and J. N. Tsitsiklis, "Introduction to Probability", Athena Scientific, 2002.

[9] T. Bilgiç and I.B. Turksen, "Model-Based Localization for an Autonomous Mobile Robot", in proceedings of the IEEE Conference on Systems, Man and Cybernetics, pp. 22-25, Canada, 1995.

[10] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun, "Towards object mapping in dynamic environments with mobile robots", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1014-1019, 2002.

[11] J. Borenstein and Y. Koren, "Obstacle Avoidance With Ultrasonic Sensors", IEEE Journal of Robotics and Automation, Vol. RA-4, No. 2, pp. 213-218, 1988.

[12] R.A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space", IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-13, No. 13, pp. 190-197, 1983.

[13] R.A. Brooks and T. Lozano-Perez, "A Subdivision Algorithm in Configuration Space for Findpath with Rotation", IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, No. 2, pp. 224-233, 1985.

[14] R. G. Brown and P. Y. C. Hwang, "Introduction to Random Signals and Applied Kalman Filtering", Second Edition, John Wiley & Sons, Inc, 1992.

[15] A. F. Bunting, "Radar Sensor Model for Three-Dimensional Map Building", in processing SPIE, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, SPIE, Vol. 4195, 2000.

[16] J. Castellanos and J. Tard'os, "Mobile Robot Localization and Map Building: A Multisensor Fusion Approach", Kluwer Academic Publishers, Boston, MA, 2000.

[17] H. Chung, J.G. Lee and Y.S. Kim, "Planning Optimal Paths in a Partially Unknown Environment", in processing IFAC Intelligent Autonomous Vehicle, pp. 59-64, 1995.

[18] C. Cou , T. Fraichard, P. Bessire, and E. Mayer, "Multi-Sensor Data Fusion Using Bayesian Programming: an Automotive Application", Intelligent Vehicle Symposium, IEEE, 2002.

[19] C. Cou, T. Fraichard, P. Bessire and E. Mayer, "Using Bayesian Programming for Multi-Sensor Multi-Target Tracking in Automotive Applications", Int. Conference. on Robotics and Automation, Taipai, 2003.

[20] I. J. Cox and J. J. Leonard, "Modeling a dynamic environment using a Bayesian multiple hypothesis approach", Artificial Intelligence, vol. 66(2), pp. 311-344, April 1994.

[21] B. Dasarathy, "Decision Fusion", IEEE Computer Society Press, 1994.

[22] M. Edgar and O. Akihisa, "A multi-mobile robot system for steering groups of people: A Cooperative framework for crowdance and guidance", 52nd Yamabico Symposium, pp. 197-204, Gunma Prefecture, Japan, 2003.

[23] A. Elfes, "Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception", Autonomous Mobile Robots, IEEE, Vol. 1, pp 60-70, 1991.

[24] A. Fusiello and V. Roberto, "Efficient stereo with multiple windowing", in processing IEEE Conference on Computer Vision and Pattern Recognition, pp. 858–863, 1997.

[25] E. P. George, "Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building", Wiley-Inter science press, 1978.

[26] W.E.L. Grimson, "A computer implementation of a theory of human stereo vision", Royal Press, B-292, pp. 217–253, 1981.

[27] D. Hall and J. Llina, "Handbook of Multisensor Data Fusion", CRC Press 2001.

[28] R. M. Haralick and L. G. Shapiro, "Computer and Robot Vision", Addison-Wesley, 1992.

[29] M. Kam, X. Zhu, and P. Kalata, "Sensor Fusion for Mobile Robot Navigation", Proceeding of the IEEE vol. 85, No. 1, pp. 10-18, 1997.

[30] L.A. Klein, "A Boolean algebra approach to multiple sensor voting fusion", IEEE Trans. Aerospace Electron. Sys. vol. 29(2), pp. 317–327, 1993.

[31] Kodak, "CCD Image Sensor Noise Sources", technique report, Eastman Kodak Company, Aug 2001.

[32] P. Kovesi's line segmentation algorithm http://www.csse.uwa.edu.au/~pk/ Research /MatlabFns/ index.html

[33] B. Kuipers and Y-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations", Journal of Robotics and Autonomous Systems, vol. 8, pp. 47–63, 1991.

[34] G. Lakemeyer and B. Nebel, "Exploring Artificial Intelligence in the New Millennium", Morgan Kaufmann Chapter 1, pp. 1-36, 2002.

[35] M. Lanthier, D. Nussbaum, A. Sheng, "Improving Vision-Based Maps By Using Sonar and Infrared Data", Robotics and Applications, IASTED 2004, Hawaii, USA, pp. 118-123, 2004.

[36] J. J. Leonard and H. F. Durrant-Whyte, "Directed Sonar Sensing for Mobile Robot Navigation", Boston: Kluwer Academic Publishers, 1992.

[37] W. Li, "Fuzzy logic based robot navigation in uncertain environments by multi-sensor integration", in processing. IEEE Conference. on Multi-sensor Fusion and Integration for Intell. Syst., 1994, pp. 259–265.

[38] P. S. Maybeck, "Stochastic Models, Estimation and Control", Vol 1, Academic Press, 1979, pp 3-6.

[39] P.J. McKerrow, "Introduction to Robotics", Addison-Wesley Publishing Co., Sydney, 1991.

[40] G.G. Medioni and R. Nevatia, "Segment-based stereo matching". CVGIP, vol. 31(1) pp. 2-18, 1985.

[41] M. Miller and D. Snyder, "The role of likelihood and entropy in incomplete-data problems: Applications to estimating point-process intensities and toeplitzconstrained covariances", proceedings of the IEEE, vol. 75, pp. 892–906, July 1987.

[42] D. Murray and J. Little, "Using Real-Time Stereo Vision for Mobile Robot Navigation", Workshop on Perception for Mobile Agents at CVPR'98, Santa Barbara, CA, pp. 161–171, 1998.

[43] K. Nickels and C. Cianci, "Fusion of Lidar and Stereo Range for Mobile Robots", The 11th International Conference on Advanced Robotics (IRAC), 2003.

[44] K. M. Passino, "Design and Organization of Autonomous Systems", IEEE Spectum, The institute of electrical and electronics engineers inc., Volume, Number 6, 1995.

[45] P. Pirjanian, H. I. Christensen and J. A. Fayman, "Experimental Investigation of Voting Schemes for Fusion of Redundant Purposive Modules", in processing 5th Symposium for Intelligent Robotics Systems, Stockholm, Sweden, pp. 131-138, 1997.

[46] Ptgrey, Technical specification for Bumblebee stereo camera http://www.ptgrey.com/ products/bumblebee/index.html

[47] L. Robert and O. Faugeras, "Curve-based stereo: Figural continuity and curvature", in processing IEEE Conference on Computer Vision and Pattern Recognition, pp. 57–62, 1991.

[48] H. Samet, "Neighbor Finding Techniques for Images Represented by Quadtrees", Computer Graphics and Image Processing, vol. 18, pp. 37-57, 1982.

[49] H. Samet, "An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures", NATO ASI Series, Vol. F40, 1988.

[50] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo methods", Workshop on Stereo and Multi-Baseline Vision, Kauai, vol. 47, pp. 7-42, 2001.

[51] H. Schneider and P. M. Frank, "Fuzzy Logic-Based Threshold Adaptation for Fault Detection in Robots", Proceedings of IEEE Conference on Control Application, NJ, Vol. 2, pp. 1127-1132, 1994.

[52] Sharp, Technical specification for Sharp proximity sensor http://info.Hobbyengineering .com / specs/SHARP-GP2Y0D02YK.pdf

[53] Sharp, Technical specification for Sharp IR proximity sensor http://www.acroname.com/ robotics/parts/ R48-IR12.html

[54] R. Szeliski and R. Zabih. "An experimental comparison of stereo algorithms", in processing IEEE Workshop on Vision Algorithms, pp. 1-19, 1999.

[55] S. Thrun, "Probabilistic Algorithms in Robotics", AI Magazine, vol. 21, No.4, pp. 92-109, 2000.

[56] H. Tian, B. Fowler, and A. E. Gamal, "Analysis of Temporal Noise in CMOS Photodiode Active Pixel Sensor", IEEE Journal of solid-state circuits, vol. 36, No. 1, pp. 92-100, 2001.

[57] E. Trucco, A. Verri, "Introductory Techniques for 3-D Computer Vision", Prentice Hall, 1998.

[58] R. J. Trump, "Binocular vision and the stereoscopic sense", Trans. Opt. Soc. 25, pp. 261-272, 1924.

[59] S. Ullman, "The Interpretation of Visual Motion", MIT Press, 1979.

[60] J. Vaganay, M. J. Aldon and A. Fourinier, "Mobile Robot Attitude Estimation by Fusion of Inertial Data", IEEE Int. Conf. on Robotics and Automation, Atlanta, GA, pp. 277-282, 1993.

[61] W.A. Wright, "Bayesian approach to Neural-Network modeling with input uncertainty", IEEE Trans. Neural Networks, pp. 1261-1270,1999.

[62] H. Wu, M. Siege, R. Stiefelhagen and J. Yang, "Sensor Fusion Using Dempster-Shafer Theory", Proceedings of IEEE Instrumentation and Measurement Technology Conference, Anchorage, AK, USA, May 21-23, 2002, May, 2002.

[63] A. Zelinsky, "A Mobile Robot Exploration Algorithm", IEEE Transactions on Robotics and Automation, Vol. 8, No. 6, pp. 707-717, 1992.

[64] K. Zhang, I. Ginzburg, B. L. McNaughton, and T. J. Sejnowski, "Interpreting Neuronal Population Activity by Reconstruction: Unified Framework With Application to Hippocampal Place Cells", Journal of Neurophysiology. vol. 79, pp. 1017-1044, 1998.

[65] S. Zhou, G.L. Wojcik and J.A. Hossack, "An approach for reducing adjacent element crosstalk in ultrasound arrays", Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions, vol. 50, Issue: 12, pp. 234-241, 2003.

[66] Y. Zou, K. Yeong, S C. Chin and X. Zhou ,"Multi-ultrasonic sensor fusion for autonomous mobile robots", Sensor Fusion: Architectures, Algorithms, and Applications IV, Belur V. Dasarathy, Editor, pp. 314-321, 2000.

Appendix 1: Hardware Architecture of the K2A Platform

K2A is a sophisticated and accurate industrial robot and it is designed for the exploration of structured indoor environments. A sonar ring consisting of twenty-four sonar instruments was the only sensor on the robot at that time. After careful consideration, two new types of sensors were integrated into the existing sensor system. Software components including control, communication, and testing for both drive and sensor systems were built. Under the new framework, the old sensors and the new sensors are integrated seamlessly.

The following description refers to the cut-away view of K2A platform in Figure 97. The K2A has two motors (steering and drive) and their associated power trains. The steering motor (A) drives the vertical steering shaft (C) through a spiral gear reducer (B), which gives a reduction of 106:1. The vertical steering shaft is coupled to the turret-mounted flange at the top.

An optical angular pulse encoder (J) is located on the vertical steering shaft above the spiral reducer in the upper column housing (M). The connector for the steering encoder is mounted on a plate in the side of the housing just below the round connector for the slip ring. The slip ring is mounted above the encoder, with its rotating connector (N) in the center of the top of the steering shaft.

The lower end of the vertical steering shaft terminates in a miter gear, which engages three like gears on the three leg steering shafts (E). These shafts are hollow, and a drive shaft is suspended in the center of each of them. On the outside end, each leg steering shaft has an identical miter gear that engages a like gear on the foot housing (F), thus affecting the steering of the wheel.

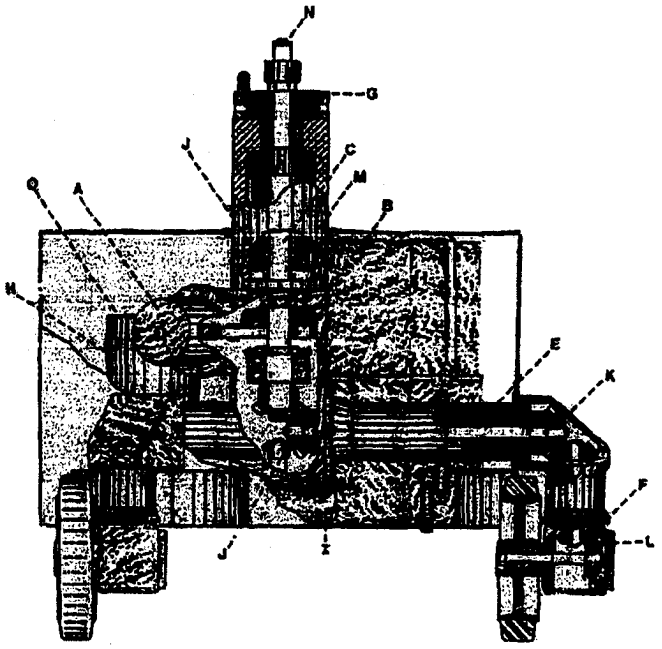


Figure 97: The mechanical architecture of the K2A robot.

The drive motor (H) has an optical pulse encoder (O) mounted on top of it. The motor drives an oil-filled gear box (I) providing a 24:1 reduction through two levels of spur gears. The output of the drive gear box (J) has a miter gear, which is smaller than those in the steering chain. This gear drives three like gears attached to the three horizontal leg drive shafts. These gears are located in the hollow centers of the three

steering gears. Each of these shafts is terminated in its outer end by an identical miter gear (K), which drives the vertical drive shaft attached to the respective foot.

The vertical drive shaft for each foot powers its respective wheel through a bevel gear set (L). This gear set has a reduction ratio that exactly matches the ratio of the wheel diameter and its steering circle (as traced on the floor). Thus, if the steering is actuated while the drive motor is held fixed, the foot rolls around the steering circle by reflex action. This action reduces floor damage and makes for a smooth, efficient turn even while the vehicle is stationary.

The K2A contains two electronic parts, the computer and the motor power amplifier. The computer is based on a Z-80 CPU and is implemented in CMOS to conserve power. The computer card is powered by a special designed power supply for efficiency and to isolate it from transient currents produced by the motors. All control signals from the computer to the motor power amplifier are optically isolated to prevent glitches and to serve as a protective barrier in case of a catastrophic failure of the motor power amplifier.

Appendix 2: Sharp® IR Proximity Sensor

Our robot uses eight Sharp GP2Y0A02YK (See Figure 98) infrared proximity sensors, which calculate distance by triangulation.

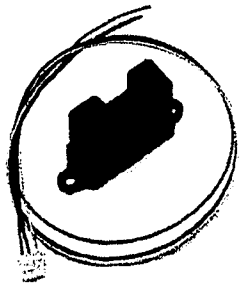


Figure 98: The Sharp GP2Y0A02YK IR proximity sensor.

Each Sharp GP2Y0A02YK IR proximity sensor consists of one infrared LED and one corresponding photo-diode. The sensor generates distance information by measuring the angle of the infrared light from the LED reflected by an obstacle. Therefore, the signal returned from the sensor is independent of the energy emitted from the LED but dependent on the detectable range of the photo-diode. These facts cause problems at the upper and lower end of the sensor range. When the sensor is positioned too close to an obstacle, (e.g. 5cm), a small change in the surface angle of the obstacle can cause a big change on the reflection angle, which makes the reading unstable. In contrast, readings taken at distances above 150cm become indistinguishable due to the small angular changes which cannot be detected by the photo-diode. Within the 5cm to 150cm range, these sensors perform well, but provide readings that will have a nonlinear relationship with respect to distance.

The Sharp GP2Y0A02YK proximity sensor is an analog device and it converts the angle of reflected infrared into voltage data. In order for the robot's computer to process such data, an *A/D* converter (i.e. analog to digital) is required to change the analog signals into digital signals. The *A/D* converter used on the K2A is a Labjack® U12, which provides eight *A/D* channels. The IR beam generated by theses IR proximity sensors is roughly football shaped with the widest portion in the middle, at about 6cm wide. It is a reasonably narrow beam pattern, when compared to other sensors such as sonar. Since the beam width of these IR sensors is narrow, it would take a long time to measure the environment with one sensor. Therefore, we built an IR sensor array using eight of these IR proximity sensors (See Figure 99) to increase the data acquisition speed.

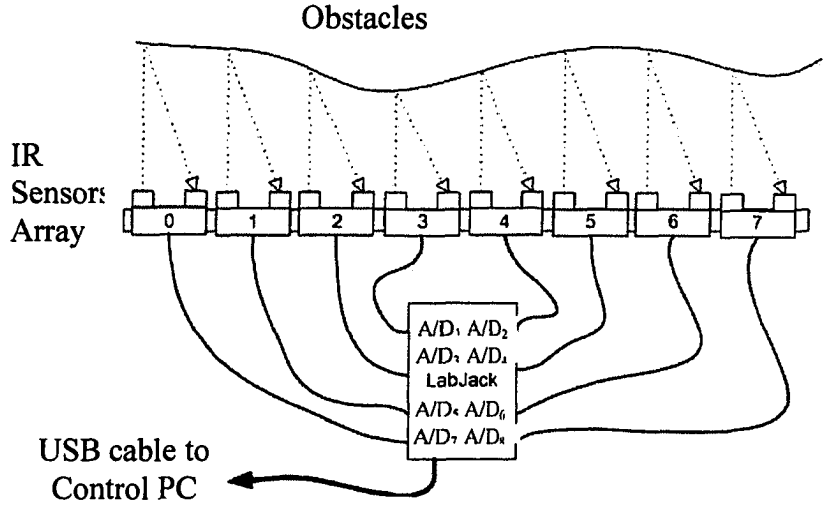


Figure 99: IR sensor array consists of eight IR sensors.

The use of the IR sensor array allows us to take full advantage of the good angular resolution of each sensor (due to the narrow beam width) while still covering a relatively wide area of the environment (i.e. eight times that of a single sensor).

Appendix 3: The Sonar Ring

The sonar ring on the K2A includes twenty-four of these Polaroid 6500 ultrasonic transducers. These transducers have a flat frequency response from 20kHz to 100kHz. There are three transducer control modules (TCMs[13]), each controlling the operation of eight transducers. These modules control the transducer excitation pulse (50kHz burst) and the depth gain compensation and produce a digital output pulse for each transducer. The width of this digital output pulse is the time between the excitation of the transducer and the time of arrival of the first ultrasonic echo that is larger than a present threshold. Figure 100 shows the interconnection of the transducer control modules and a sensor signal processor (SBC). These sensors are positioned pointing outward around a ring which is 60cm in diameter. Great care is taken to ensure that each transducer points radially outwards from the center of the ring, with the principal axis of each transducer in a common plane, 15° from each of its neighbors. A micro-controller provides an interface to a controlling computer through a RS232 interface. The SBC controls the three TCMs.

---

[13] Purchased from Denning Robotics® and modified electrically

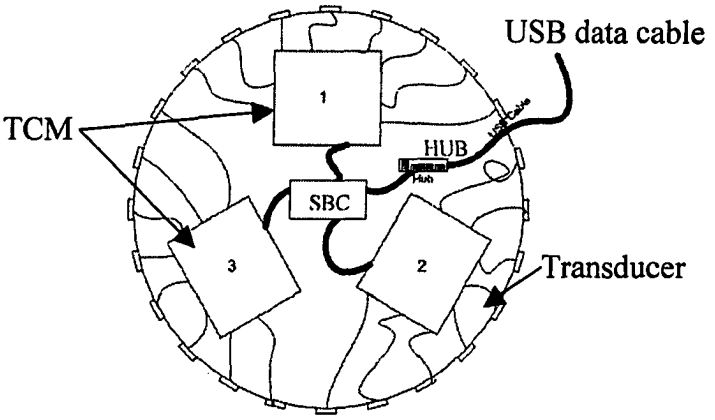


Figure 100: The sonar ring.

This module has an external blanking input that allows selective echo exclusion for operation on a multiple-echo mode. The module is able to differentiate echoes from objects that are only three inches apart. The digitally controlled-gain, variable-bandwidth amplifier minimizes noise and side-lobe detection in sonar applications.

The module has an accurate ceramic-resonator-controlled 420-kHz time-base generator. An output based on the 420-kHz time base is provided for external use. The sonar transmit output is 16 cycles at a frequency of 49.4 kHz.

Appendix 4: UML Framework design

The software framework was designed to be an autonomous robot mapping system, so several components, that were designed, were not actually implemented. There are two major hierarchies in the software framework: the functionality hierarchy and the algorithm hierarchy.

A4.1 Functionality Hierarchy

Based on the functionality hierarchy, the whole system consists four different components: the local mapping component, the global mapping component, the navigation component (not implemented) and the localization component (not implemented) (see Figure 101).

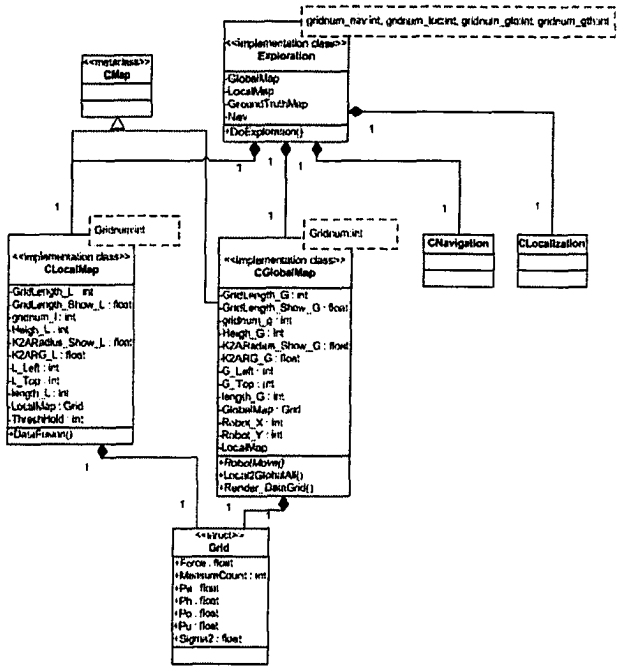


Figure 101: The functionality hierarchy.

1. Local Mapping Component (ClocalMap Class)

The local mapping component consists a local map which covers the region that the robot can detect at the current position and time. The local map is where the sensor data fusion process actually happens. The size of the local map is set to 10m×10m since the maximal sensor detection range is 10m. The sensor models for different sensors and the Kalman filter are included in this component.

2. Global Mapping Component (CGlobalMap Class)

The global mapping component consists of a global map which covers a much larger region. It acts as a superset of all local maps and as a local-map swap area. The size of the global map is adjustable, but the grid cell size in the global map is the same as the local maps. The size of the global map in our framework was set to 20m×20m.

3. Navigation Component (CNavigation Class)

Reserve for future usage.

4. Localization Component (Clocalization Class)

Reserve for future usage.

A4.2 Algorithm Hierarchy

The second hierarchy represents different algorithms including all sensor models as well as the Kalman filter algorithm (see Figure 102).

1. Sensor Model (CSensorModel Class)

The sensor model component was designed to be interchangeable and we designed and implemented sensor models for the sonar ring, the stereo vision sensor

and the IR proximity sensor array. All concrete sensor models are inherited from a base class, which make them switchable during the runtime.

2. Kalman Filter (CKalmanFilter Class)

The Kalman filter class is also inherited from a base class: Calgorithm. In the current framework, we only implemented the Kalman filter algorithm. It is easy to append new data fusion algorithms by creating another class which inherits from the base class.

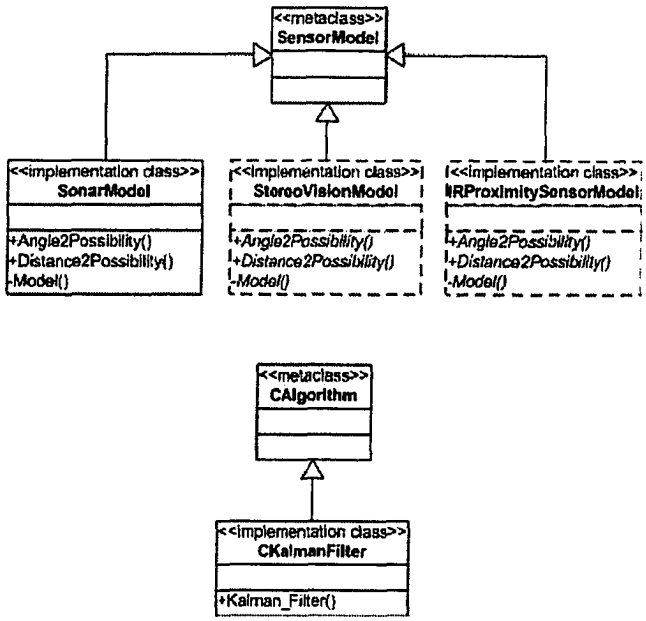


Figure 102: The algorithm hierarchy.