1-1-2006

Mobile robot localization using a Kalman filter and relative bearing measurements to known landmarks

John Ryan Burnett
Iowa State University

Mobile robot localization using a Kalman filter and relative bearing measurements to known landmarks

by

John Ryan Burnett

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Greg Luecke, Major Professor
Yan-Bin Jia
James Bernard

Iowa State University

Ames, Iowa

2006

Graduate College
Iowa State University

This is to certify that the master's thesis of

John Ryan Burnett

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to dedicate this work to my mother, Nellie Burnett, who worked hard specifically to save money to put me into college and to my father who always believed in me. I thank the rest of my family for their support and understanding throughout my years at university. Thank you also to my major professor, Greg Luecke, for taking me on as a graduate student. Fellow graduate students Jake Ingman, Adam Bogenrief, Jesse Lane, Kevin Godby, and Ethan Slattery helped keep me sane in the lab and I loved every minute. I also appreciate being able to vent my frustrations to Brenna Griffin and anyone else who asked, "So, what are you working on?"

ABSTRACT

This paper discusses mobile robot localization using a single, fixed camera that is capable of detecting predefined landmarks in the environment. For each visible landmark, the camera provides a relative bearing but not a relative range.

This research represents work toward an inexpensive sensor that could be added to a mobile robot in order to provide more accurate estimates of the robot's location. It uses the Kalman filter as a framework, which is a proven method for incorporating sensor data into navigation problems.

In the simulations presented later, it is assumed that the filter can perform accurate feature recognition. In the experimental setup, however, a webcam and an open source library are used to recognize and track bearing to a set of unique markers.

Although this research requires that the landmark locations be known, in contrast to research in simultaneous localization and mapping, the results are still useful in an industrial setting where placing known landmarks would be acceptable.

CHAPTER 1. INTRODUCTION

Leonard and Durrant-Whyte [1992] describe the problem of navigation in terms of three questions:

- "Where am I?",
- "Where am I going?", and
- "How do I get there?"

The second question requires reasoning about a high-level goal or desired destination. For example, the robot may need to travel to a certain location to fill in an ad hoc wireless network, investigate a late-night disturbance, or simply deliver carry passengers. The third question refers to the field of path planning. The best route between the current location and the goal may not be a straight line if the terrain varies or obstacles are present, so a path-planning algorithm must decide which route to actually take. The research presented in this paper answers the first question, which refers to localization or finding the position of the robot relative to some global framework.

This paper will focus on mobile robot localization using a single, fixed camera that is capable of detecting predefined landmarks in the environment. For each visible landmark, the camera provides a relative bearing or angle to the landmark but not a range or distance. Cameras are very data-rich sensors available at low cost. Most computer vision software uses only a small portion of the data present in a sequence of images.

This research works toward a sensor that could be inexpensively added to a mobile robot to provide more accurate estimates of its location. The framework that is used here is the Kalman filter, which works well for incorporating sensor data and has been used historically for navigation problems [Brown, Hwang 1997].

In the simulations presented later, it is assumed that the filter can perform accurate *feature recognition*. In the real world, in order for the camera to find the bearing to a landmark, it must first be able to identify the pixels in the image as a landmark. To perform feature recognition in the experimental section, the open source library ARToolkit is used to track a set of markers [Kato, 1999].

This research requires that distinct landmarks be present in the environment. This is a drawback for the general usefulness of these results, since it is desirable to alter the environment as little as possible. However, having known landmark locations would still be useful in an industrial setting, where engineering solutions might have more weight over aesthetic concerns. The problem of localization using landmarks with unknown locations

requires that the landmarks be mapped as the robot moves. The interested reader should refer to Dissanayake et al. [2001], Deans and Hebert [Deans, Hebert 2000 (ISER), 2000 (ICRA)], Fitzgibbons and Nebot [2001, 2006] and others in the field of *simultaneous localization and mapping (SLAM)*.

In order to understand the general field of robot localization, a description of some sensors and techniques used is given in Chapter 2. Particle filters and the SLAM framework are briefly addressed. Chapter 3 introduces some of the concepts behind the Kalman filter for readers who are not familiar with the subject. In Chapter 4, the specific components needed for the Kalman filter used in this research are developed. The simulation using Matlab of that developed filter is discussed in Chapter 5. The second part of the chapter discusses how the layout of the landmarks can affect the accuracy of the state estimate. Two experimental test runs using bearing measurements to real-world landmarks are introduced in Chapter 6, followed in Chapter 7 by a summary and conclusion as well as a few ideas for future work.

CHAPTER 2. PREVIOUS WORK

2.1 The problem: The importance of localization

A mobile robot is distinguished by its ability to move around, in fact, the remainder of this work will use the words "vehicle" and "robot" interchangeably to a signify a mobile robot. Knowing where exactly a mobile robot moves is understandably one of its most important, basic functionalities. A robot with poor localization might smash into a wall or nosedive over a drop. Even if it can avoid these obstacles and others in the local area around the robot, *global localization* or finding position relative to some non-robot reference frame is required for any strategy that requires movement to a specific location.

A distinction is made between global localization and tracking. Global position estimation is "the ability to determine the robot's position in an *a priori* or previously learned map, given no other information than that the robot is somewhere on the map. Once a robot has been localized in the map, local tracking is the problem of keeping track of that position over time" [Dalleart 1999 (ICRA)].

Although a robot that is ignorant of its location can complete certain tasks, localization almost always expands the robot's usefulness. For example, iRobot's robotic vacuum cleaner *Roomba* is a useful robot without localization, but its "bump, turn, go forward" strategy would clean floors probably clean faster if it could determine its current location and whether or not it had already cleaned there.

In a 2-dimensional plane, the problem of localization is finding the position and orientation of the robot (Figure 1). This data is generally bundled into the *state vector*, **x**.

$$\mathbf{x} = \begin{bmatrix} \text{x position} \\ \text{y position} \\ \text{orientation} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}$$

The values for x, y, and ψ are measured in some *global coordinate frame*. The location of the origin of the global coordinate frame is not usually important.

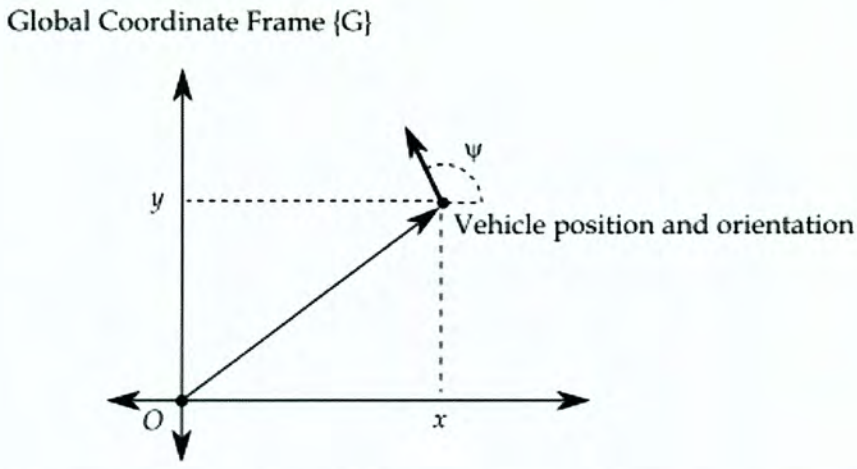

Figure 1. The state vector contains *x*-position, *y*-position, and orientation; this is called the pose of the robot.

2.2 Sensors

Consider the ways that humans localize themselves in the world. Some methods use only "on-board" information, like counting paces to measure the distance from some point, or like using the inner ear or the flow of visual motion to detect small changes in the movement of the head. Other methods require outside information from the environment, like using the stars or seeing landmarks. Measurements from the inside are called *proprioceptive* or robot-centric [Thrun 2002] in mobile robots, and measurements from the outside are *exteroceptive* or world-centric.

One distinction in localization techniques is whether or not the technique requires modification of the environment; for example, whether you can recognize the big tree on the corner or whether a sign needs to be nailed to it. Some engineering applications may require the extra precision gained from having an artificial beacon or sign, but localization without any special accommodation is preferred.

2.2.1 Odometry

Odometry refers to the estimation of a path based on proprioceptive inputs. For example, based on the number of wheel rotations, the odometer on a car displays how many miles the car has gone. Variables like wheel rotation and the steering angle of the wheels can be input to a computational model of the vehicle to estimate its path. The differences between this model and the actual vehicle response, however, will cause errors that affect all future estimates of position. The model can be made more accurate by increasing its complexity--including

variables and coefficients for friction, suspension, damping, aerodynamic drag--but no model will ever be entirely accurate, if only because measurements of the variables involved must have some finite error [Smith, Self, Cheeseman 1990].

Inertial measurement is a technique used for odometry that records the acceleration of the vehicle. This change in velocity is accumulated over time to give the velocity, which is in turn numerically integrated to give position. So, noisy or biased initial measurements acceleration produce magnified errors in the final position estimate.

2.2.2 Range and bearing

Like drawing a picture with your eyes closed, odometry-only estimates can only get worse over time, since each error affects all subsequent estimates. Environmental feedback from sonar, radar, and lasers provides external measurements of range to a feature or landmark. Since these measurement devices provide directional input, the direction that they are pointed when the range measurement is made also provides bearing or angle information. If the sensor is mounted on a mobile robot, then the bearing is *relative* to the orientation of the robot, hence the term *relative bearing.*

The *correlation problem* [Thrun 2002] is the problem of identifying that a particular measurement from the robot is actually measuring a particular feature in the environment. With four range and bearing measurements correlated with three separate map features on a two-dimensional map, a mobile robot can find its location and pose by triangulation.

2.2.3 Computer vision

Although humans can get along fine without sonar, radar or lasers, we do have binocular vision to give us range and bearing information. For bearing information from vision, we can internally sense where our eyes are pointed and also map a position in our current field of view to a position in the world. Range information is determined from the discrepancy between the left and right images in our eyes based on parallax. With monocular vision, as with a single digital camera, only bearing information is available.

2.3 Techniques

The following techniques use the measurements from the sensors above.

2.3.1 Odometry-only

The Kalman filter method described later in Section 3 uses a *state dynamics model* that incorporates odometry data [Zarchan, Musoff 2005]. This model uses the state vector from the current time step and any measured user input to the system in order to predict the next state of the vehicle. Using odometry for localization is also called *dead reckoning*. Here we will see how *kinematic equations* or equations of motion for a particular mobile robot are derived.

The known inputs for the state transition model used here are the speed and steering angle. The speed is measured from encoders counting the number of rotations of the wheels. It is common to refer to the distance traveled along a curved path as s, and so speed, the distance traveled over time, is referred to as $\dot{s}$. The speed is related to the state vector $\mathbf{x} = [x; y; \theta]$ through the velocity vector. Any vector has both magnitude and direction; the velocity vector has magnitude equal to the speed of the vehicle and direction equal to the orientation of the vehicle.

The velocities in the x and y directions are simply components of the velocity vector (Figure 2).

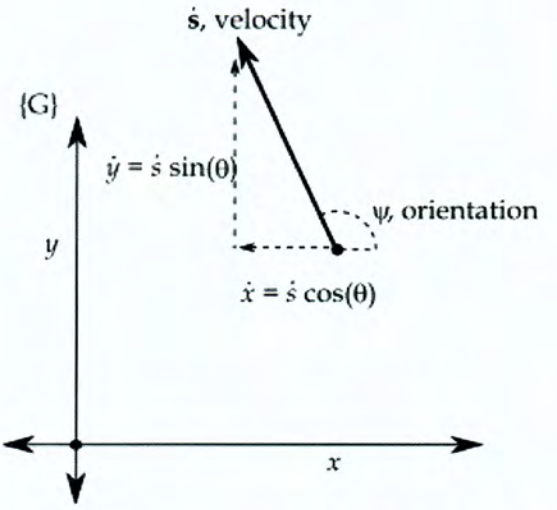

Figure 2. The kinematic model contains the *x*-velocity and *y*-velocity.

The vehicle model assumes that the wheels do not slip and always move in the direction that they are facing. This could be thought of as having perfect traction, allowing the model of the vehicle to take the tightest turns at any speed without sliding sideways. This is a common simplification to reduce mathematical complexity, but a more complex model could be used for more accurate results.

The steering angle is relates to the state vector by altering the orientation, θ. A model that approximates how steering angle affects orientation of a front-steer, four wheeled vehicle is the *bicycle model*, so named because it simplifies the four wheels into two by collapsing the front and back wheels into a single wheel for each (Figure 3).

This simplification is made by assuming that the two front wheels both follow a path along concentric circles having a common center point. Steering systems on most vehicles are designed to do this; the arrangement is called Ackerman steering. Now, the two front wheels can be replaced by a single wheel halfway between them whose path lies along a circle with the same center as the first two. The two back wheels are also replaced by a single wheel between them. This two-wheeled bicycle model has motion that is equivalent to the four-wheeled Ackerman steering model.

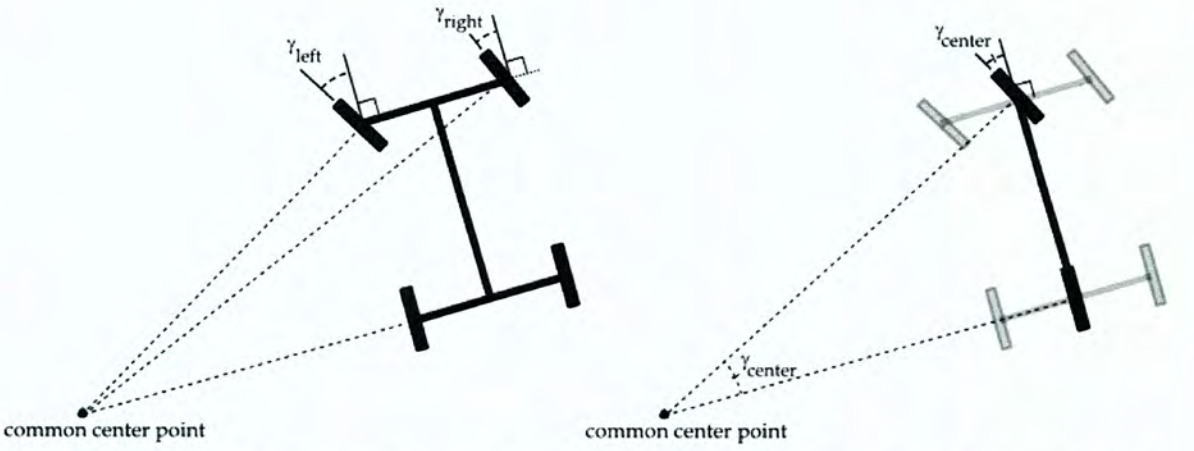

Figure 3. The Ackerman steering model can be reduced to a two-wheeled bicycle model, using the common center point provided by turning radius, *r*.

The yaw rate $\dot{\psi}$ can be found from considering the turning radius, *r*, over an instantaneous time interval *dt* as in Figure 4.

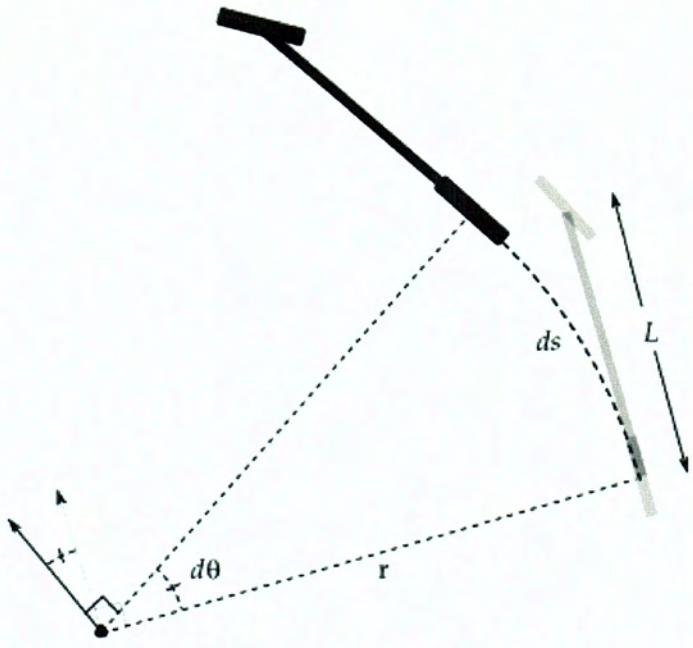

Figure 4. Using this simplified model, the yaw rate is found from the motion about the center point found from the turning radius.

$$r * d = ds$$

$$\frac{d}{dt}(r * d\psi) = \frac{d}{dt} ds$$

$$r \frac{d\psi}{dt} = \frac{ds}{dt}$$

$$r\dot{\psi} = \dot{s}$$

The turning radius, r, is difficult to measure directly, but we can find it from the steering angle, γ .

$$\tan\gamma = \frac{L}{r}$$

So, we substitute for r and solve for the yaw rate $\dot{\psi}$.

$$\dot{\psi} = \frac{u * \tan\gamma}{L}$$

The equations for $\dot{x}$, $\dot{y}$, and $\dot{\psi}$ make up the kinematic model of motion for the vehicle. These equations can be used in a *state dynamics model* to predict the motion of the vehicle based on the previous states and the known user inputs to the system.

2.3.2 Particle filters

The particle filter is a relatively new technique [Dellaert, et al. 1999]. The "particles" are really multiple hypotheses about the state of the robot. A single particle fills a state vectors with its own values.

The algorithm consists of the following steps [Thrun, et al. 2005]:

1. *Initial state distribution* – The initial states of the particles are sampled from the initial state probability density function, if one is provided. If no initial information is available, then the states are uniformly distributed.
2. *State update* - The transition model is applied to each of the particles with some variation. The variation is based on the assumed accuracy of the transition model.
3. *Compute weights* - The actual measurement is taken. Each particle uses a measurement model to predict the probability of making that measurement at its current state and is weighted accordingly.
4. *Resample* - The particles are redistributed according to these weights.

The particle filter is better than the Kalman filter at dealing with the "kidnapping" problem. Kidnapping refers to removing the robot and placing it in a new location. The particle filter can handle this if some particles are dispersed around the map during the state update. These will generally be pruned out during resampling, because the simulated measurement in the dispersed location will not agree with the actual measurement.

The CONDENSATION algorithm was used successfully by Dallaert, Burgard, Fox and Thrun to localize a robot used as a tour guide for a museum [Dellaert, et al. 1999]. The robot had a camera pointed upward to make measurements of the lighting directly above it. The robot also had a transition model to estimate the movement of the robot from odometry measurements. The lighting was measured as the robot moved and compared to a predefined lighting map of the ceiling based on many composite camera images. As the robot moved, the weights were decreased for the particles whose simulated measurements did not match the actual measurements from the robot's camera. Weights were increased for particles whose simulated measurements

matched the actual measurements. During resampling, the particles were redistributed with a probability that depended on the weight. In this way, the erroneous particles were moved to congregate around the correct position whose measurements matched the map (Figure 5).

Figure 5. Example of particle filter using sonar as measurements. (a) Initial distribution, (b) After 10 sonar measurements, (c) After 65 sonar measurements. Slides from http://www.probabilistic-robotics.org/

An important element of this approach is the choice for the number of particles. Too few particles may get stuck in a local minimum and do not converge to a single correct value. Too many particles will take longer to converge to a single location.

Since this process involves multiple particles, how do we know which is correct? Any "shotgun" approach is not correct simply because it hits the target with some particle. A single output must be chosen in order to evaluate the accuracy of the method. For particle filters, it makes sense to choose the particle with the highest weight during the *Compute weight* step.

2.3.3 Kalman filters

Kalman filters have been used for navigation, tracking and many other applications since it was introduced in 1960 [Kalman 1960]. It offers improved performance at a computational cost over other filtering techniques that require less processing power to implement [Zarchan, Musoff 2005].

The Kalman filter also uses an internal transition model of a system to estimate the future state. The filter also simulates a measurement at that future state. It uses the difference between that predicted measurement and the actual measurement to update the state.

The primary difference between the Kalman filter and the particle filter is the way that the state estimate probability or confidence is stored. The Kalman filter models the probability as Gaussian, with only the mean

and variance. The particle filter stores the probability using the weights and positions of the particles. This allows it to store non-Gaussian probabilities that have more than one peak, instead of only the bell-shaped Gaussian curve. The filter will be introduced in more detail next in Section 3. Section 4 will show how the filter is specifically applied to the problem of using relative bearing measurements for robot localization.

2.3.4 SLAM

Kalman filters are also used in the Simultaneous Localization and Mapping framework [Dissanayake, et al. 2001]. This framework allows the robot to create a map of stationary landmarks as it drives and to localize itself inside of that map. This system has the advantage of not requiring a modification to the environment. A robot with radar can evaluate and find stationary landmarks and store their estimated locations in the state vector along with position and orientation. Then, using those landmarks, it can find its own position relative to them.

Before the landmark estimates can be used for localization, "the initial feature estimate must be 'near enough' to the true feature location" [Bailey 2003]. So, trying to localize based on a measurement that is a total guess does not work; the feature must be *initialized*. In one experimental setup, "the vehicle starts at the origin, remaining stationary for approximately 30s" [Dissanayake, et al. 2001]. This stationary time allows the radar to initialize enough landmarks. In this sense, the SLAM framework is not quite simultaneous—it does a little mapping before it starts localization.

The system using radar is able to initialize landmarks while not moving because radar provides both range and bearing, which is enough information to locate the landmark relative to the vehicle. Using only bearing information, a stationary robot only has a measurement of the direction in which a landmark lies. The landmark might lie any distance in that direction.

Two bearing measurements are required to pinpoint the location of a landmark. This means that the robot must move before it can make an initial guess at the locations of the landmarks. "Constrained initialization" used by Bailey [2003] involves deferring measurements and the poses in which they were taken until the landmark can be initialized by a well-conditioned pair of measurements. Then, the accumulated deferred measurements are

applied to the new landmark. A discussion of this delay in initializing the landmark map can be found in [Ortega, et al. 2005].

2.3.5 Present research

The research presented here addresses the same problem as Bailey but under less strict constraints. Landmark positions are assumed to be known, removing the requirement to do any online mapping. The problem is reduced to localization using bearing measurements from a camera with a known map. This is a more straightforward Kalman filter application.

This avenue of research still provides a useful result. The SLAM approach tackles the considerable problem of mapping without any *a priori* information. This is useful and necessary in novel situations like underwater navigation or piloting planetary rovers. But for many engineering applications, obtaining information about a robot's workspace is not a problem. In addition, modifying that workspace by adding visual landmarks is acceptable if it increases the precision of the robot.

In this domain, localization using relative bearing measurements to known landmarks is a useful tool. Using known landmarks increases the accuracy of the state estimate, since the filter does not have to map the landmarks from noisy measurements.

2.4 Summary

Localization is a fundamental part of mobile robotics. In order for a robot to locate itself in the environment, it must have either proprioceptive internal sensors or exteroceptive external sensors. Examples of proprioceptive sensors are those used for odometry, such as encoders for counting wheel revolutions or potentiometers for measuring steering angle. Exteroceptive sensors include radar, lidar, sonar and CCD cameras. These sensors all provide range data, except for the camera which can only provide bearing information. The data from these sensors can be utilized by several techniques in order to localize a robot. The simplest technique is dead reckoning from odometry data. A newer technique is the particle filter, which uses multiple hypotheses and probability to find the most likely location of the robot. Finally, the technique used in this research is the Kalman filter, which will be introduced in more detail next in Chapter 3. Readers who are familiar with the

Kalman filter may want to skip ahead to Chapter 4, where its application to this research's specific problem is addressed.

CHAPTER 3. KALMAN FILTER INTRODUCTION

Relative bearing information from cameras can be used to triangulate the position of a mobile robot. This position will have errors due to uncertainty in the bearing measurement, or the measurements may not even be available due to occlusion of the available landmarks. The objective is to use a Kalman filter to fuse the results of an odometry estimate of position with a relative bearing estimate of position.

3.1 The Kalman filter method

3.1.1 Introduction

The Kalman filter is useful for multi-input, multi-output, time-varying systems. Its matrix form makes it easy to add to the filter's input with more measurements or add to its output by tracking more states [Brown, Hwang 1997].

Since it is a *recursive* algorithm, the filter is light-weight in terms of storage. To illustrate the difference between *batch* and *recursive* algorithms, consider taking the average of a column of values. If a new value is added to the column, the batch algorithm needs to retrieve all the values from data storage, add them up, and divide by the column length. A recursive algorithm needs only the previous average, the column length, and the value to be added to the column. Similarly, the Kalman filter keeps only the previous states and a matrix that describes the error present in those states, the *state error covariance matrix*.

The recursive method also has a downside: "If a new observation becomes available which indicates that the old state estimate was wrong, there is no way to go back and fix it... For linear problems and Gaussian noise, the Kalman filter is optimal, and there is no never a need to go back and fix errors in the estimate" [Deans 1999]. The non-linear extended Kalman filter is suboptimal, however, and errors can affect the state estimate. The extended Kalman filter must be used for the mobile robotics problem in this research..

3.1.2 An "Observer"

Controllers use *feedback* to control the response of a system. A user-supplied desired value is subtracted from the current output by the system [Nise 2003].

$$\text{control input} = \text{gain} * (\text{output}_{actual} - \text{output}_{desired})$$

This output difference is input back into the system to form a *control loop*. It is multiplied by a factor called the *gain* to control how fast or how slow the system reacts. If the actual output is not high enough, the control input will be positive. A positive control input generally increases the actual output so that it reaches the desired value. In this way, a controller uses the difference between a desired output and the system output states to control the output of the system.

The Kalman filter is a specific type of controller called an *observer*. An observer is used when direct measurement of an internal system value or *state* is not available. Suppose we have a real world system whose states are not available for measurement. An observer is a mathematical model of the system that runs in parallel with that system (Figure 6).

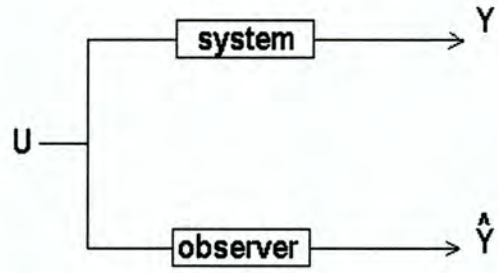

Figure 6. Real world system and observer

The observer receives the same inputs U as the real world system and also produces its own output $\hat{Y}$. The observer uses the difference between the real-world system output and the observer model's output to correct the state estimates of the observer (Figure 7). Contrast this with a controller, which uses the difference between the actual state and the desired state to correct the output. In effect, the observer is using feedback from the *output* to control the *states*, instead of using feedback from the *states* to control the *output*.

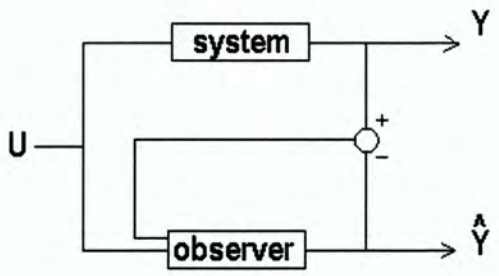

Figure 7. The output difference is used to alter the observer's state estimates.

An observer is a recursive algorithm, and the states X are fed back into the system. Similar to the controller, an observer has a gain to control how fast or slow the observer converges to the correct states (Figure 8).

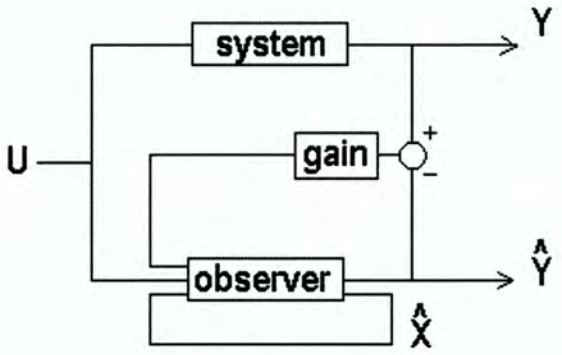

Figure 8. The observer also uses a gain.

3.1.3 Observability

With a controller, we control a system by changing the input to make the output values do what we want. However, if the input has no effect on the output values, it doesn't matter what we do; the system is not *controllable*. A similar concept is used for observers, except that we are using the output to determine the state estimates. Changes in the state usually change the output. If a state variable has no effect on the output, then the system is not *observable*. Since the state variable is disconnected from the output, we cannot evaluate this state variable by observing the output.

One lesson to learn here is that there are no magic solutions, only algorithms that push a designer's responsibility elsewhere. A controller requires knowledge of the states. If you can't access the states, you can

use an observer to estimate the states. An observer requires a mathematical system model and also gains that help the observer quickly settle to the correct states. What if a designer doesn't want to find the correct gains? That responsibility can be pushed elsewhere again by using a Kalman filter. A Kalman filter provides the correct observer gains, but now the designer is required to provide *statistical* information about the process and the measurements of the output.

3.1.4 Kalman gains

A fine introduction to the Kalman filter is given by Maybeck [1979]. He defines the Kalman filter as an *optimal recursive data processing algorithm.* Although there are many ways to define *optimal*, the Kalman filter is "optimal with respect to virtually any criterion that makes sense" as seen in a moment. *Data processing algorithm* is more precise than "filter"; meaning that it is a computer program designed to take discrete measurements and turn them into optimal estimates [Brown, Hwang 1997].

The Kalman filter uses three pieces of information:

1. knowledge of both the system and measurement dynamics;
2. statistical descriptions of the system noise, the measurement noise and the model uncertainty;
3. information about the initial conditions of the system.

One of the most useful features of the filter is its ability to use vector modeling to incorporate as much data as is available. For vehicle tracking, information could be incorporated from odometry, inertial systems, GPS, compass heading, radar measurements. The measurements may have differing levels of accuracy, but they are incorporated in such a way that they give the minimum statistical error in the state estimate. This means that compared to any other algorithm run for the same application, the *average* results of the Kalman filter would be better.

The Kalman filter tries to maintain information about the *conditional probability density* of the states.

Conditional probability, $P(x \mid z)$ is the probability of one event, x, given some previous knowledge, z. For

example, given a positive review of a restaurant, you might be more likely to eat there than if you had not read the review.

$$P(\text{choose that restaurant} \mid \text{read the review}) > P(\text{choose that restaurant})$$

Probability density is slightly different than probability. The shape of a *probability density* curve is a way of illustrating the spread of likelihood; that nothing is certain but some things are more certain than others. A tall peak shows an area of high probability, whereas a wide flat area shows choices that are all equally likely to each other. The area under the density curve on a certain interval is equal to the probability that the actual value lies on that interval. Since the actual value must lie on the graph somewhere, the area under the entire density curve must equal one.

Here is an example from Maybeck. Suppose that an experienced sailor and I are out at sea. If I measure the angle between the horizon and the north star using a sextant, the conditional probability of the measurement might look like Figure 9 below, where z_1 is my particular measurement of the angle x. The height of the curve at a certain point represents the amount of confidence that I have that the true value will lie at that point. I am most confident that the true value will lie directly at my measurement z_1, but the shape of the curve suggests that I think it might lie to either side with diminishing confidence.

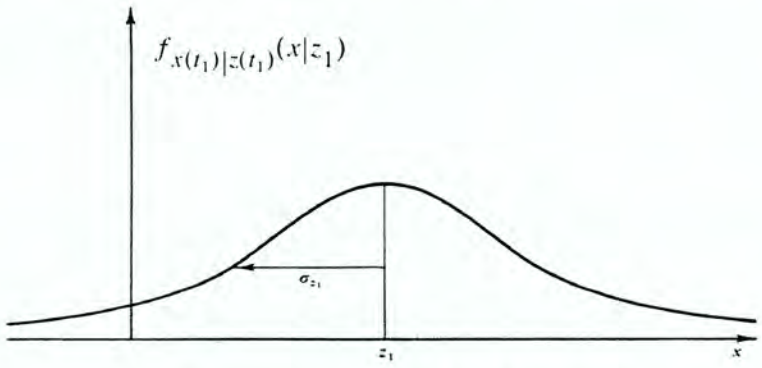

Figure 9. "Land-lubber" angle measurement is broad and flat. This reflects the likelihood that I screwed up; that the actual value might be some other value of x than z_1.

The sea captain's estimate, z_2, will have more confidence associated with it (Figure 10). This measurement has a smaller standard deviation, σ, which means that values close to the measurement's average value or *mean* have a high probability.

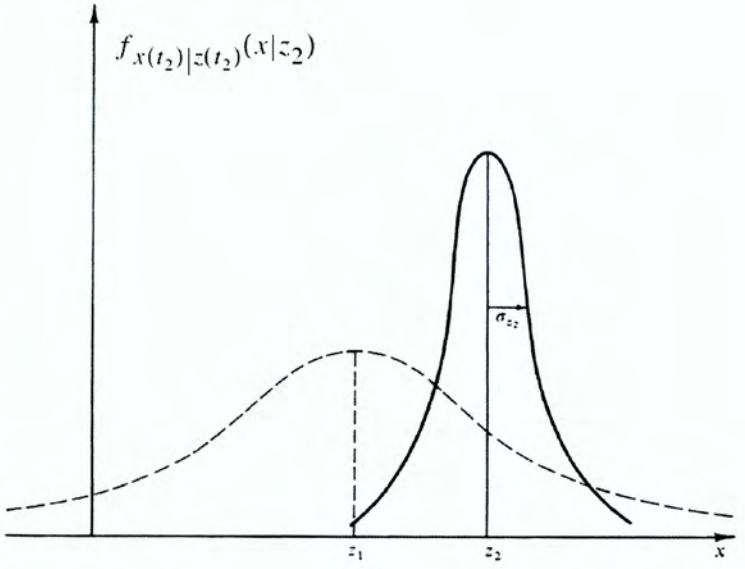

Figure 10. A second, more accurate measurement has a narrower conditional probability density

Neither measurement z_1 nor z_2 is 100% correct. The best guess would lean more toward the captain's measurement, but the first measurement should not be completely ignored. Figure 11 shows the combination of the two measurements; this is the probability density conditioned on both the z_1 and z_2 measurements, $f(x \mid z_1, z_2)$.

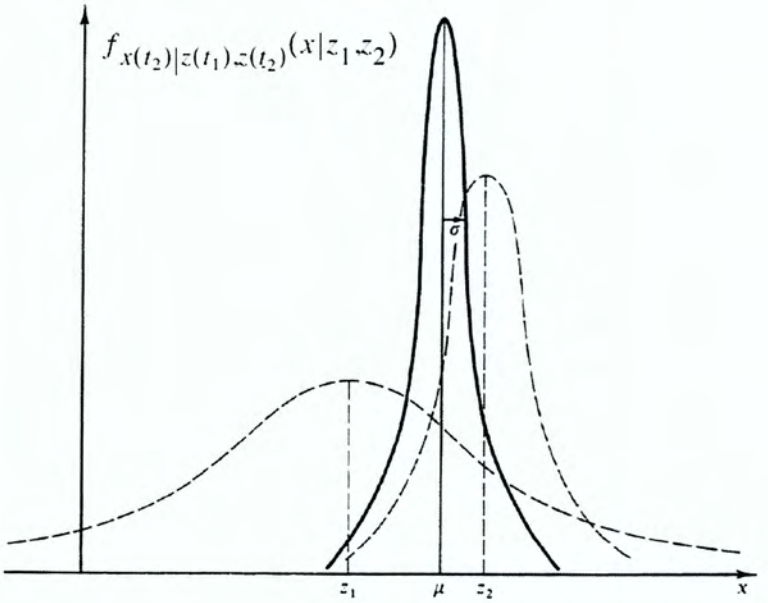

Figure 11. Two estimates can be combined to produce a more accurate third measurement.

If we had a conditional probability density of some arbitrary shape, the definition of what estimate is *optimal* becomes important. Is it best to choose the point at the peak of the probability density, the point at the center of the area, or the point that divides the total density in half? Maybeck points out that under the Kalman filter assumptions, these choices coincide, giving the optimal answer.

The three Kalman filter assumptions are that the system can be described with a *linear* model and that the model and measurement noises are *white* and *Gaussian*. These Kalman filter assumptions "are almost universally violated" in practical application [Deans 1999].

First, although many systems of interest in engineering are non-linear, a common approach to non-linearity is to linearize around some initial point. The extended Kalman filter used in this research actually violates this assumption.

Second, noise or random error is white if it is uncorrelated in time. This means that knowing the present noise value will not help you at all in determining what the next value will be. So, the white noise at a certain time does not depend on values from any other time. White noise is also defined as having the same finite power for all frequencies. Since there are infinite frequencies, this implies that white noise has infinite power, and no such noise can actually exist. However, any system can only respond to a certain range of frequencies; frequencies that are too high or too low do not contribute to the system output. White noise can be approximated by noise that has the same power over the practical range of the system. The white noise assumption is made to simplify the mathematical derivation of the Kalman filter.

For the final assumption, the Gaussian noise concerns the amplitude of the noise, rather than the frequency. A Gaussian probability density has a bell-curve shape. The assumption is also made for mathematical simplicity. This is the assumption that clearly shows which choice is *optimal*: the choice at the center of the bell-curve. The assumption can be justified in two ways: complex sources of noise resemble a Gaussian distribution, according to the central limit theorem, which states that the sum of a large number of random variables will closely resemble a Gaussian distribution, regardless of the distribution of the individual random variables. Second, a Gaussian distribution is completely described by only two variables, mean and variance. This makes

it easier for the Kalman filter designer, who is responsible for the description of the process and measurement noise.

3.2 Summary

The Kalman filter is a particular kind of controller, called an observer, that is capable of estimating the internal states of a process. The Kalman filter uses differences between predicted measurements and actual measurements to adjust its estimates of state. This difference is multiplied by a gain that is calculated by the Kalman filter. Based on the statistics of the problem, the gains give more weight to measurements that are more accurate. Next in Chapter 4, we will look at how the Kalman filter is applied to the specific problem of localization using relative bearing measurements.

CHAPTER 4. KALMAN FILTER DEVELOPMENT

In the preface to their "Practical Approach," Paul Zarchan and Howard Musoff point out that setting up the problem is often the hardest part of implementing a Kalman filter; and in fact, just getting something to compile and work *incorrectly* is a challenge [Zarchan, Musoff 2005]. My own research fully corroborates this fact. Zarchan and Musoff also point out that certain steps are consistently followed for most applications. The following development of the extended Kalman filter for bearings-only tracking follows the same outline. Begin by forming mathematical functions of the process and measurements using differential equations. Based on the problem statement, choose which variables the filter should estimate and put them in the *state vector*. Linearize the process and measurement functions to determine the *system dynamics matrix* and *measurement matrix*. Discretize the system dynamics matrix using a Taylor-series approximation to form the fundamental matrix. Determine the continuous *process noise matrix* both from knowledge of uncertainty in the real-world system and from acknowledgement of uncertainty in the accuracy of the system dynamics model. Now, use the fundamental matrix to find the discrete process noise. This defines all the elements for the matrix Riccati equations, which are used to determine the Kalman gains. The final step requires choosing a numerical integration method to propagate the current states into the next sampling time.

4.1 Problem setup and state vector

The problem is to find the location of a 4-wheeled mobile robot using measurements of bearing from the robot to known landmarks relative to the robot's own orientation. Since it is the location of the robot that we are interested in, the state vector should include x- and y- coordinates. The location of the robot could also be specified by polar coordinates, but it is not clear that this gives any particular advantage. The orientation, ψ, should also be included since it is required for the relative bearing measurements. So, the state vector should at least include x, y, and ψ.

A second or third order filter could be used, which would include first and second derivatives of x and y in the state vector. These higher order states would estimate the velocity and acceleration of the vehicle in the Cartesian frame. If the dynamics of the system are unknown, this would be a good option to in order to

estimate some characteristics of the system in the filter. Adding more states, however, increases the complexity of the filter and increases the time required for convergence [Zarchan, Musoff 2005]. Also, in our case we have a kinematic model of the system, derived in Chapter 2. So, the state vector remains,

$$\mathbf{x} = \begin{bmatrix} x, y, \psi \end{bmatrix}^T$$

This is called the *pose* of the vehicle in the 2-dimensional plane.

We have already derived the equations of motion in Chapter 2.

$$\dot{x} = \dot{s} * \cos(\psi)$$

$$\dot{y} = \dot{s} * \sin(\psi)$$

$$\dot{\psi} = \frac{\dot{s}}{L} \tan(\gamma)$$

Multiple bearing measurements are made relative to the direction of the vehicle from the vehicle's location P_v to the landmark P_i. The measurements are made from the center of the rear axle for computational simplicity, but the camera could be placed anywhere on the vehicle with an additional translation [Dissanayake, et al. 2001]. The global frame that is used to localize the vehicle is usually defined with its origin at the vehicle's starting position, but this need not be the case.

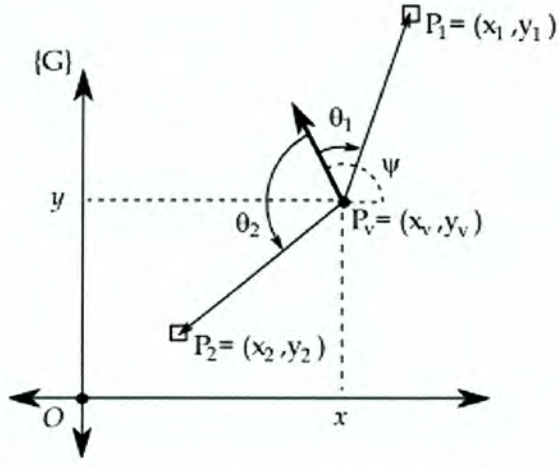

Figure 12. Bearing measurement angles from vehicle to landmarks P_1 and P_2

The bearing measurement θ_i (Figure 12) can be found with the tangent function

$$\theta_i = \tan^{-1}\left(\frac{x_v - x_i}{y_v - y_i}\right) - \psi + w_\theta$$

where w_θ represents noise that is associated with the measurements.

4.2 System dynamics matrix

Since the equations of motion are non-linear, they must be linearized before they can be used in the Kalman filter as the system dynamics matrix, **F**.

Linearizing a function of two variables refers to picking a point on the function and drawing a *line* tangent to the curve. From this linear approximation, the "rise" is equal to the slope times the "run", or $\Delta y = {dy}/{dx}\,\Delta x$.

For functions of more than one variable, the partial derivative is used in a more complicated Jacobian matrix, but the idea of linear approximation remains the same.

$$\begin{bmatrix} \Delta\dot{x} \\ \Delta\dot{y} \\ \Delta\dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{\partial\dot{x}}{\partial x} & \frac{\partial\dot{x}}{\partial y} & \frac{\partial\dot{x}}{\partial\psi} \\ \frac{\partial\dot{y}}{\partial x} & \frac{\partial\dot{y}}{\partial y} & \frac{\partial\dot{y}}{\partial\psi} \\ \frac{\partial\dot{\psi}}{\partial x} & \frac{\partial\dot{\psi}}{\partial y} & \frac{\partial\dot{\psi}}{\partial\psi} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\psi \end{bmatrix}$$

This is written as

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x}$$

The entire state dynamics equation is generally written as $\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u}$. The term **u** is the vector of inputs velocity and steering angle, $\mathbf{u} = \begin{bmatrix} \dot{s}, \gamma \end{bmatrix}$. In the linear filter, this equation would be used to propagate the states from one time-step to the next. Since we are using the extended Kalman filter, the equations of motion are

numerically integrated to propagate the states. The **G** matrix is not used in the extended Kalman filter and so it is not included here.

The system dynamics matrix **F** is evaluated at some particular point x_0.

$$\mathbf{F} = \left[\begin{array}{ccc} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} & \frac{\partial \dot{x}}{\partial \psi} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial \psi} \\ \frac{\partial \dot{\psi}}{\partial x} & \frac{\partial \dot{\psi}}{\partial y} & \frac{\partial \dot{\psi}}{\partial \psi} \end{array}\right]_{x=x_0}$$

Now, the equations of motion are used to find these nine partial derivatives.

$\frac{\partial \dot{x}}{\partial x} = 0$	$\frac{\partial \dot{x}}{\partial y} = 0$	$\frac{\partial \dot{x}}{\partial \psi} = -\dot{s}\sin(\psi)$
$\frac{\partial \dot{y}}{\partial x} = 0$	$\frac{\partial \dot{y}}{\partial y} = 0$	$\frac{\partial \dot{y}}{\partial \psi} = \dot{s}\cos(\psi)$
$\frac{\partial \dot{\psi}}{\partial x} = 0$	$\frac{\partial \dot{\psi}}{\partial y} = 0$	$\frac{\partial \dot{\psi}}{\partial \psi} = 0$

So, the system dynamics matrix is

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & -\dot{s}_0 \sin(\psi_0) \\ 0 & 0 & \dot{s}_0 \cos(\psi_0) \\ 0 & 0 & 0 \end{bmatrix}$$

4.3 Discrete fundamental matrix

Next, we need to use this systems dynamics matrix to create a discrete matrix. When implemented on a computer, the filter will not be continuous, but will perform an operation on a regular cycle of time *t*, say every *dt* = 0.1 seconds.

A Taylor-series approximation will allow us to use the system dynamics matrix to approximate the continuous equations of motion explicitly as a function of time. The resulting matrix is called the fundamental matrix, Φ.

In order to use the Taylor-series, we need to make the assumption that the matrix won't change very much between sampling instances. Since the time between samples will only be a fraction of a second, this assumption seems reasonable.

The infinite term Taylor-series approximation is

$$\ddot{\mathbf{O}}(t) = \mathbf{I} + \mathbf{F}t + \frac{\mathbf{F}^2 t^2}{2!} + \frac{\mathbf{F}^3 t^3}{3!} + \frac{\mathbf{F}^4 t^4}{4!} + \ldots$$

Notice that the terms of the series are divided by increasingly larger numbers. (For example, recall that $3! = 3*2*1$ and $4! = 4*3*2*1$.) A Taylor-series approximation with infinite terms actually matches the original continuous function perfectly, which is not really an approximation at all. However, Zarchan shows that having only the first two terms is all that is necessary in applications where the assumption that the matrix remains approximately constant between samples is true.

So, we truncate the series after two terms

$$\ddot{\mathbf{O}}(t) = \mathbf{I} + \mathbf{F}t$$

Now, we just substitute **I** and **F** into this equation. The **I** term refers to the identity matrix, filled with zeros except for ones on the diagonal. And for space and clarity, the terms of **F** are replaced with variables

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & f_{13} \\ 0 & 0 & f_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

where the subscripts for f_{13} indicate *row* 1 and *column* 3. These two variables hold the place of the original contents of the matrix cells

$$f_{13} = -\dot{s}\sin(\psi_0)$$

$$f_{23} = \dot{s}\cos(\psi_0)$$

Substitute **I** and **F** into the first two terms of the Taylor-series approximation.

$$\ddot{\mathbf{O}}(t) = \mathbf{I} + \mathbf{F}t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & f_{13} \\ 0 & 0 & f_{23} \\ 0 & 0 & 0 \end{bmatrix} t$$

which leaves

$$\ddot{\mathbf{O}}(t) = \begin{bmatrix} 1 & 0 & f_{13}t \\ 0 & 1 & f_{23}t \\ 0 & 0 & 1 \end{bmatrix}$$

Since this continuous fundamental matrix is explicitly a function of time, to obtain the discrete fundamental matrix we simply set t equal to the sampling time T_s. The discrete fundamental matrix is

$$\ddot{\mathbf{O}}_k = \begin{bmatrix} 1 & 0 & f_{13}T_s \\ 0 & 1 & f_{23}T_s \\ 0 & 0 & 1 \end{bmatrix}$$

The subscript k refers to a certain time-step. If we sample every 0.1 seconds, then after 1.2 seconds the discrete fundamental matrix will be on its 12th sample: $\ddot{\mathbf{O}}(t = 1.2) = \ddot{\mathbf{O}}_{12}$.

The fundamental matrix describes how the states change from time-step to time-step, but in the extended Kalman filter, it is not used to propagate the states. Instead, the original non-linear equations of motion are numerically integrated to predict the next state. The fundamental matrix is only used in the Riccati equations to decide how much confidence the filter should place in the state predictions.

4.4 Measurement matrix

The same discretization process must be followed for the measurement equation. Recall from Figure 12 the measurement equation

$$\theta_i = \tan^{-1}\left(\frac{x_v - x_i}{y_v - y_i}\right) - \psi + w_\theta$$

This equation needs to be converted into a matrix form, in the same way that we have converted the equations of motion into the fundamental matrix. The form of the measurement equation is

$$\mathbf{z} = \mathbf{H}\mathbf{x}$$

The measurement matrix **H** relates the measurements **z** to the state vector **x**. Our **z** vector is simply a list of bearing measurements. In order to relate each measurement to each state, if we have *i* measurements and *j* states, **H** must be a $j \times i$ matrix.

$$\begin{bmatrix} \Delta\theta_1 \\ \vdots \\ \Delta\theta_i \\ \vdots \\ \Delta\theta_m \end{bmatrix} = \begin{bmatrix} \frac{\partial\theta_1}{\partial x} & \frac{\partial\theta_1}{\partial y} & \frac{\partial\theta_1}{\partial\psi} \\ \vdots & \vdots & \vdots \\ \frac{\partial\theta_i}{\partial x} & \frac{\partial\theta_i}{\partial y} & \frac{\partial\theta_i}{\partial\psi} \\ \vdots & \vdots & \vdots \\ \frac{\partial\theta_m}{\partial x} & \frac{\partial\theta_m}{\partial x} & \frac{\partial\theta_m}{\partial x} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\psi \end{bmatrix}$$

Finding the three partial derivatives for the measurement θ_i will generalize to all the measurements.

$$\frac{\partial\theta_i}{\partial x} = \frac{-(y_0 - y_i)}{(x_0 - x_i)^2 + (y_0 - y_i)^2}$$

$$\frac{\partial\theta_i}{\partial y} = \frac{x_0 - x_1}{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

$$\frac{\partial\theta_i}{\partial\psi} = -1$$

Now, substitute these values into the measurement matrix.

$$\begin{bmatrix} \Delta\theta_1 \\ \vdots \\ \Delta\theta_i \\ \vdots \\ \Delta\theta_n \end{bmatrix} = \begin{bmatrix} \frac{-(y_0 - y_1)}{(x_0 - x_1)^2 + (y_0 - y_1)^2} & \frac{x_0 - x_1}{(x_0 - x_1)^2 + (y_0 - y_1)^2} & -1 \\ \vdots & \vdots & \vdots \\ \frac{-(y_0 - y_i)}{(x_0 - x_i)^2 + (y_0 - y_i)^2} & \frac{x_0 - x_i}{(x_0 - x_i)^2 + (y_0 - y_i)^2} & -1 \\ \vdots & \vdots & \vdots \\ \frac{-(y_0 - y_n)}{(x_0 - x_n)^2 + (y_0 - y_n)^2} & \frac{x_0 - x_n}{(x_0 - x_n)^2 + (y_0 - y_n)^2} & -1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\varphi \end{bmatrix}$$

4.5 Discrete process noise matrix, Q_k

The equations of motion will never exactly describe what happens in the real world. Error may come from a number of places. The equations of motion assumed no tire slip, and did not include any dynamics that might change the output. The measurements of the input steering and velocity might be noisy.

For all these reasons, the state estimate might be a little off. The filter performs better if this uncertainty about the state dynamics is included. The uncertainty is added in the *process noise matrix*, Q. "Process" is another name for the system, so process noise in this case refers to the noise in the dynamics of the vehicle.

The process noise is added to each of the equations of motion, to indicate that they might not entirely match the real world state dynamics.

$$\dot{x} = \dot{s} * \cos(\psi) + \nu_x$$

$$\dot{y} = \dot{s} * \sin(\psi) + \nu_y$$

$$\dot{\psi} = \frac{\dot{s}}{L}\tan(\gamma) + \nu_\psi$$

This extra noise term is easily added to the end of the state dynamics equation as a vector, **w**.

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \mathbf{w}$$

$$\mathbf{w} = \begin{bmatrix} \nu_x \\ \nu_y \\ \nu_\psi \end{bmatrix}$$

Since this noise simply functions as a 'fudge-factor', the same noise was chosen to be added to all the channels, $\nu_x = \nu_y = \nu_\psi$.

The random variable ν is a zero-mean white sequence process noise that is uncorrelated in time. Although these process noise errors are uncorrelated from one time step to another, they may be correlated at one particular time step, *k*. The covariance matrix associated with ν_x is called the discrete process-noise covariance matrix $Q_v(k)$.

The covariance matrix can be found from

$$E\left[\mathbf{w}(k)\mathbf{w}(i)^T\right] = \begin{cases} Q(k), & i = k \\ 0, & i \neq k \end{cases}$$

where the piece-wise definition shows that the noise is only correlated at the current time-step, *k* [Brown, Hwang 1997].

The resulting continuous process-noise matrix, **Q** is

$$\mathbf{Q}(t) = \begin{bmatrix} \Phi_s^2 & \Phi_s^2 & \Phi_s^2 \\ \Phi_s^2 & \Phi_s^2 & \Phi_s^2 \\ \Phi_s^2 & \Phi_s^2 & \Phi_s^2 \end{bmatrix}$$

where Φ_s is the spectral density of the white noise sources that are being added to the states. This process noise is added to account for any noise that may be a part of the system. Unfortunately for nomenclature, this reuses the capital phi character that also refers to the fundamental matrix, since I have retained the naming conventions from Zarchan. Be careful to note the subscript *s* to tell the difference between the spectral density of the process noise and the fundamental matrix.

The discrete process-noise matrix can be derived [Zarchan, Musoff 2001] from the continuous process-noise matrix with

$$\mathbf{Q}_k = \int_0^{T_s} \ddot{\mathbf{O}}(\tau)\mathbf{Q}\,\ddot{\mathbf{O}}^T(\tau)dt$$

The continuous fundamental matrix $\ddot{\mathbf{O}}(t)$ is included in this equation to show how the noise from one variable can affect the others over time. In the continuous case the time considered is instantaneous. In the discrete case, noise has time to propagate from one state to another. For example consider how error in the orientation at $t = 0$ might affect both the x and y estimates at $t = 10$.

The discrete process noise matrix can either be solved analytically or $\mathbf{Q}_k$ can be found from numerical integration at every time-step. The amount of noise to add is generally "determined from simulation" [Zarchan], which amounts to picking a reasonable order of magnitude based on the units being used, and then adjusting it until the results desired are achieved.

4.6 Riccati equations

The Riccati equations calculate the Kalman gains, K,

$$M_k = \Phi_k P_{k-1} \Phi_k^T + Q_k$$

$$K_k = M_k H^T \left(HM_k H^T + R_k\right)^{-1}$$

and propagates the state covariance matrix,

$$P_k = \left(I - K_k H\right)M_k$$

The development of these equations is addressed in [Brown, Hwang 1997] and [Zarchan, Musoff 2005].

4.7 Initial state and state error covariance matrix

The user must specify the initial state values and also the initial state error covariance matrix **P**. The state error covariance matrix reflects the confidence that the filter has in the accuracy of the state estimate. It is recommended that the initial state estimate error should not be more than 2.5 times the standard deviation for that state, found from the state covariance matrix. [BarShalom, Fortman 1988].

The reason is that "when the first update is made, it is important that the covariance associated with the initial estimate reflect its accuracy realistically. A large error in the initial estimate, if the latter is deemed highly

accurate, will persist for a long time, because it leads to a low filter gain and thus the new information from the measurements receives too low a weighting" [Bar-Shalom, Fortman 1988].

However, Zarchan simply sets the initial covariance to infinity (or at least very high) and lets the filter narrow it down based on the state errors. This causes slower convergence than if the initial state error covariance exactly matched the error in the initial state, but it may be a better option than setting the initial state error covariance low.

4.8 State propagation using numerical integration

In the extended Kalman filter, the states are propagated using the non-linear equations of motion rather than using the linearized state dynamics matrix. Either Euler integration or any more accurate method like Runge-Kutta can be used. Zarchan shows that the integration error from Euler integration can ruin a filter's estimates, so this research uses second-order Runge-Kutta integration.

4.9 Summary

In the extended Kalman filter, the state dynamics matrix uses the linear approximation of the nonlinear equations of motion to describe how states change over time. The measurement matrix describes how the measurements affect the state vector. The process noise matrix is the designer's acknowledgement of differences between the state dynamics matrix and the real-world process. Since the filter is recursive, it needs to start with some initial value for each state and also the error covariance for each state. The filter outlined in this chapter is simulated using Matlab in Chapter 5 and run with experimental data in Chapter 6.

CHAPTER 5. SIMULATION RESULTS

The Kalman filter described in Section 4 was implemented using Matlab 7.

5.1 Results

The simulated vehicle has a wheel base of L=30cm. The camera is positioned at 90 degrees to the vehicle's left, with a field of view of 45 degrees as shown in Figure 13. The width of the vehicle is irrelevant because of the 2-wheeled bicycle model used, but it is shown as 0.75*L.

The filter receives two noisy input measurements: *velocity* and *steering angle*. The actual velocity and steering are given by the functions

$$\text{Velocity, } \dot{s} = 0.2$$

$$\text{Steering angle, } \gamma = -5 * \frac{\pi}{180} \sin(0.5t)$$

The velocity measurement was contaminated with zero mean Gaussian noise with standard deviation of 0.05 m/s. The steering angle had a constant bias in addition to Gaussian noise.

Velocity	$N(\mu = 0, \sigma = 0.05\text{m/s})$ + no bias
Steering angle	$N(\mu = 0, \sigma = 2.0\text{deg})$ + 0.5 deg bias

Table 1. Input noise and bias

Notice that the bias is barely noticeable from the measurements in Figure 14, but it leads to large errors in the odometry-only estimate of position after 10 meters of travel, shown in Figure 15, an overhead view of the experimental run. The landmarks are plotted as boxes. The actual velocity and steering angle are plotted as "Actual position". The noisy measurements of velocity and steering angle were input into a vehicle model whose states were not corrected by the Kalman filter. The path of this vehicle is plotted as "Odometry-only position". The position estimated by the Kalman filter is plotted as "Estimated position". The brighter red filter estimate crosses indicate estimates taken with more landmarks in view. The darkest crosses show estimates when no landmarks were in view.

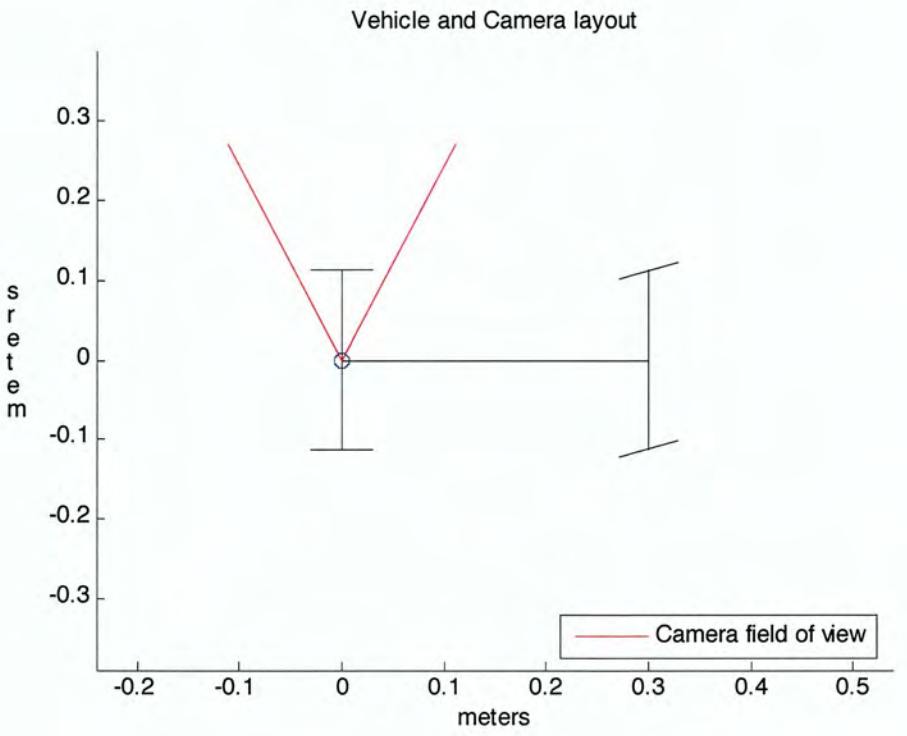

Figure 13. The camera orientation and field of view on the simulated vehicle.

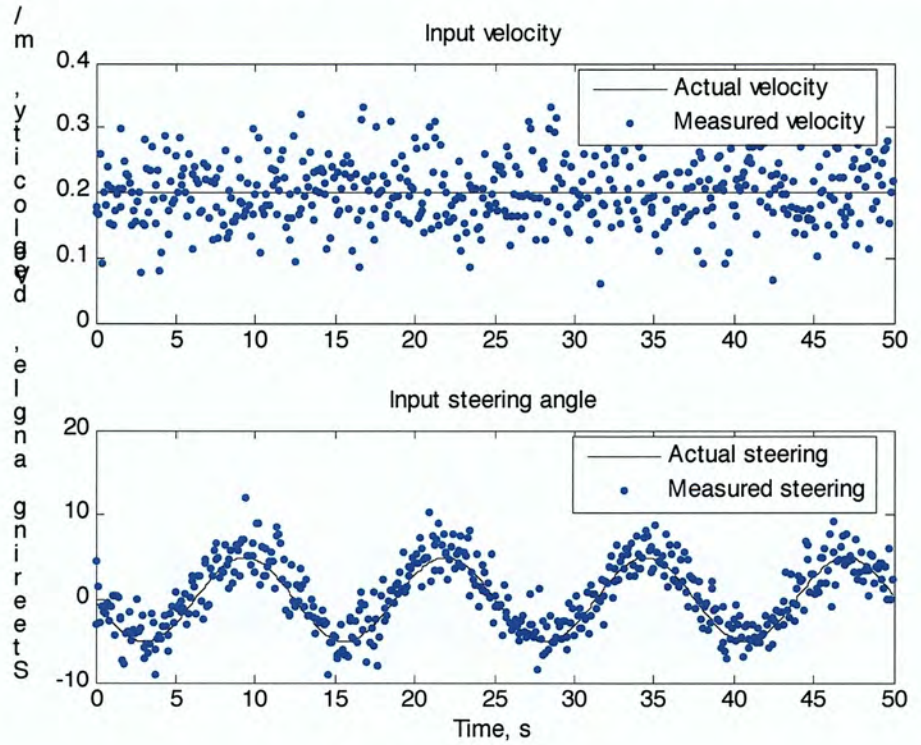

Figure 14. Inputs to filter

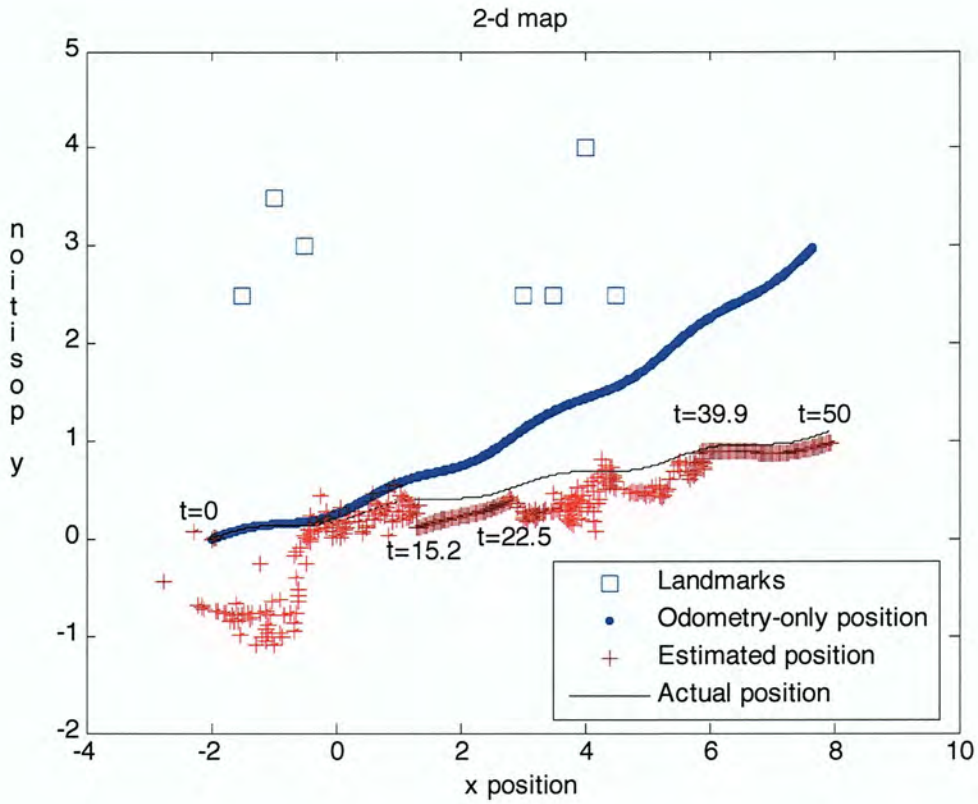

Figure 15. 2-D plot of actual path, filter estimate, and odometry-only estimate

The simulated vehicle started moving as soon as the simulation started. The estimate converges at about the time that it reaches the origin. The relatively straight estimated positions from t = (15.2, 22.5) and from t = (39.9, 50) coincide with periods when no landmarks are visible, shown in Figure 16. During these times, the filter receives no update from new measurements, and "coasts" on the equations of motion. Without a state estimate correction from new measurements, the filter behaves exactly like the odometry-only estimate.

Let's consider each of the three estimated states individually. Figure 17 shows the x-position and its error. The filter tracks the x-position of the vehicle well; the worst x-position estimate is 0.5 meters (Figure 5). However, the odometry-only estimate also gave good results here. The Gaussian noise contamination of the input velocity did not have much effect on the vehicle estimates.

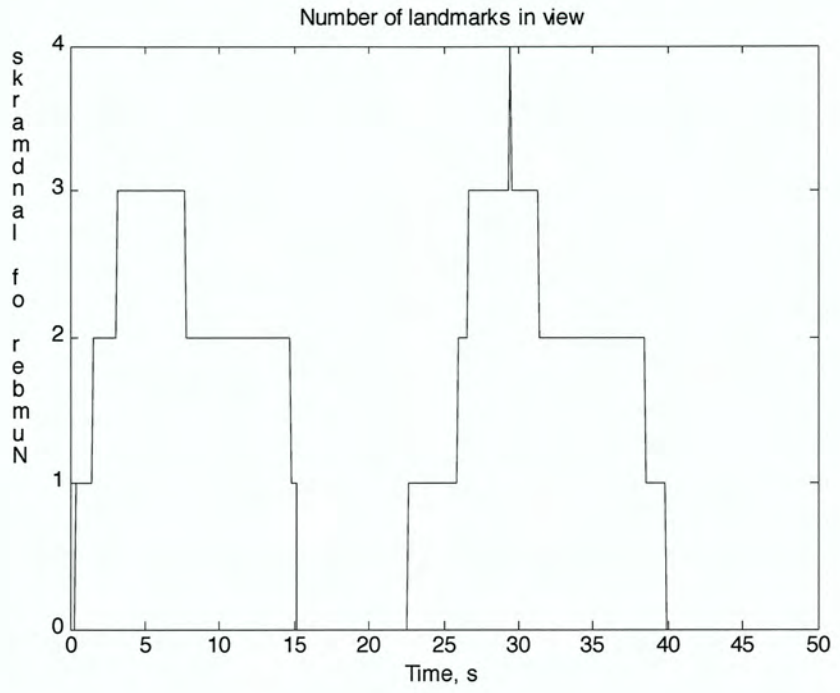

Figure 16. Number of landmarks that appear within the camera's field of view

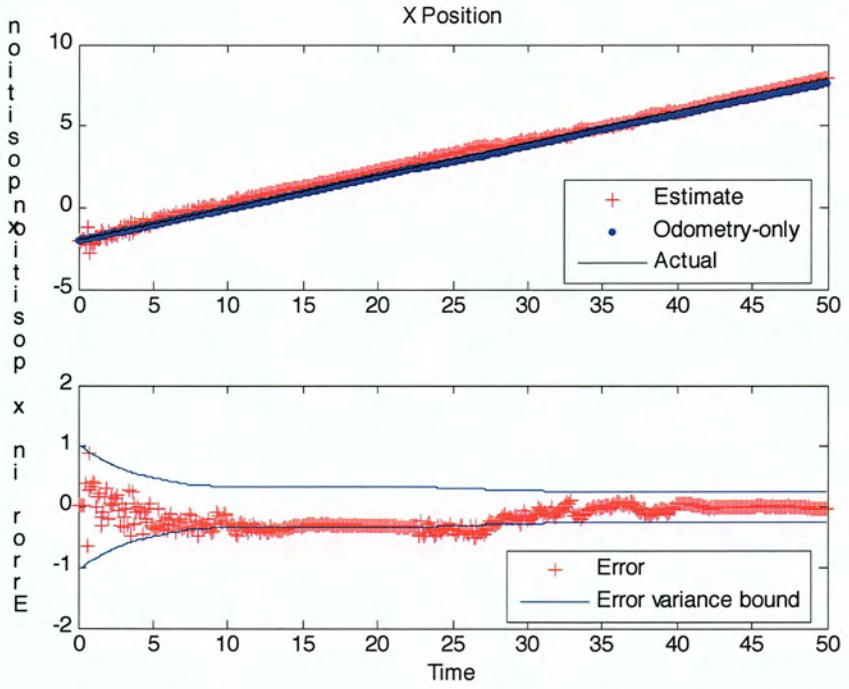

Figure 17. X-position estimates and error in X-position

Figure 18 shows the y-position and error. Although the y-position is not tracked as well, the error still falls inside of the state error covariance 1-sigma (63%) boundaries. This shows that the filter is working correctly: The error that the filter 'thinks' is occurring matches the actual error. The odometry-only estimate veers in the positive y-direction due to the bias in the steering angle measurement.

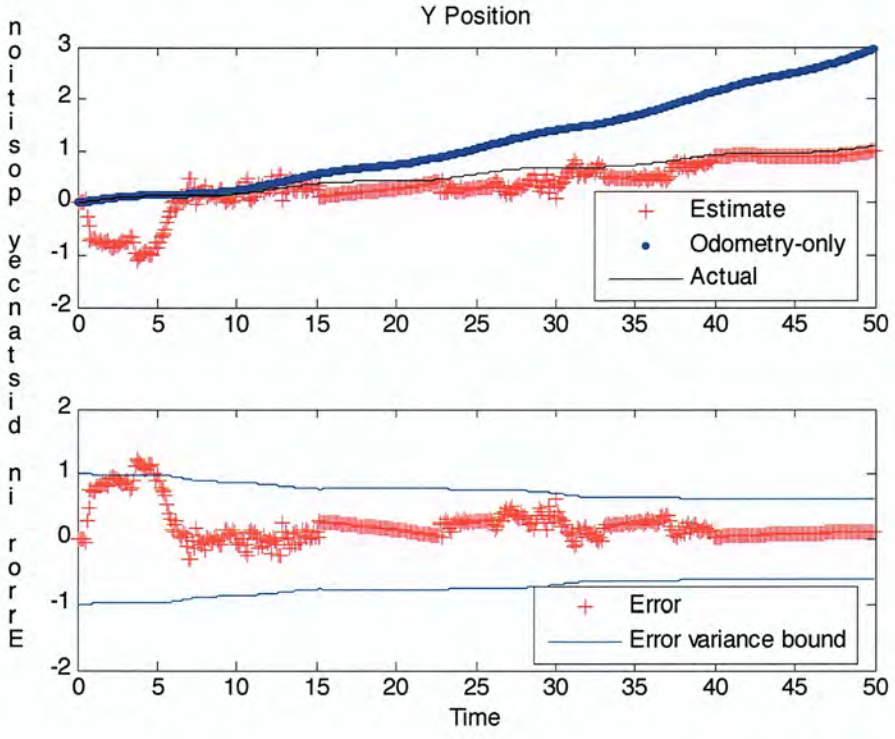

Figure 18. Y-position estimates and error in Y-position

The orientation is shown in Figure 19. For the first 25 seconds, orientation is not tracked well. At $t = 25$, the orientation is 10 degrees off. Recall that no landmarks were in view from $t = (15.2, 22.5)$. For this interval, the estimate simply drives off in the direction that it was facing, closely following the odometry-only estimate. In particular, the orientation error drifts when fewer than 3 landmarks are in view. With 3 landmarks in view, measurements from three pairs of landmarks combine to constrain the estimated position and orientation. This effect is discussed in more detail later in the chapter.

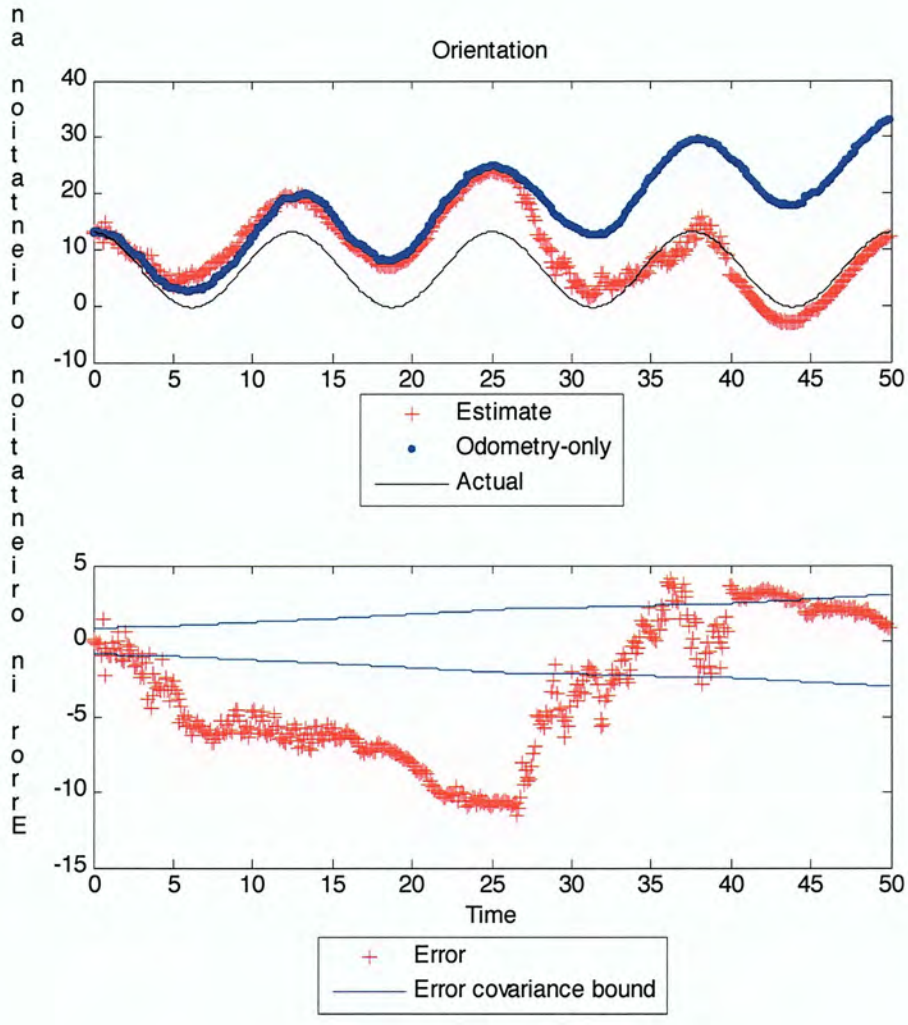

Figure 19. Orientation estimates and error

Figure 20 shows the bearing measurements for Landmark 1 that the Kalman filter uses to adjust its state predictions. The simulation uses simulated camera's field of view to only allow visible landmarks to provide measurements. Landmark 1 goes out of view at about t = 8 seconds.

The filter measurement predictions have been left to show that the filter is still capable of making these estimates even if the landmark is not visible.

Figure 21 is zoomed in on the measurements to landmark 1 made in Figure 20. The difference between the measurement and the predicted measurement is drawn with a dashed line.

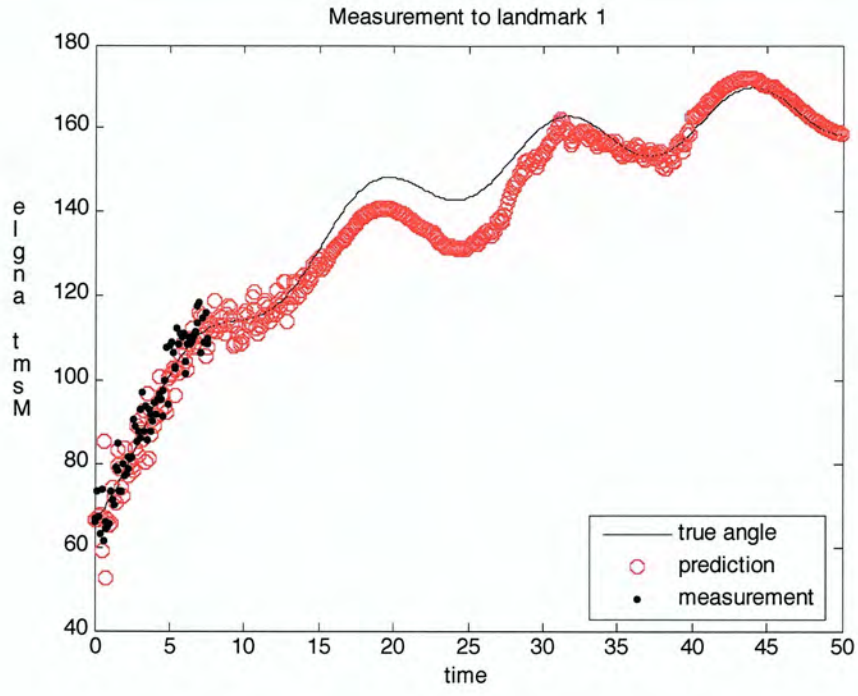

Figure 20. Measurement and measurement prediction to Landmark 1

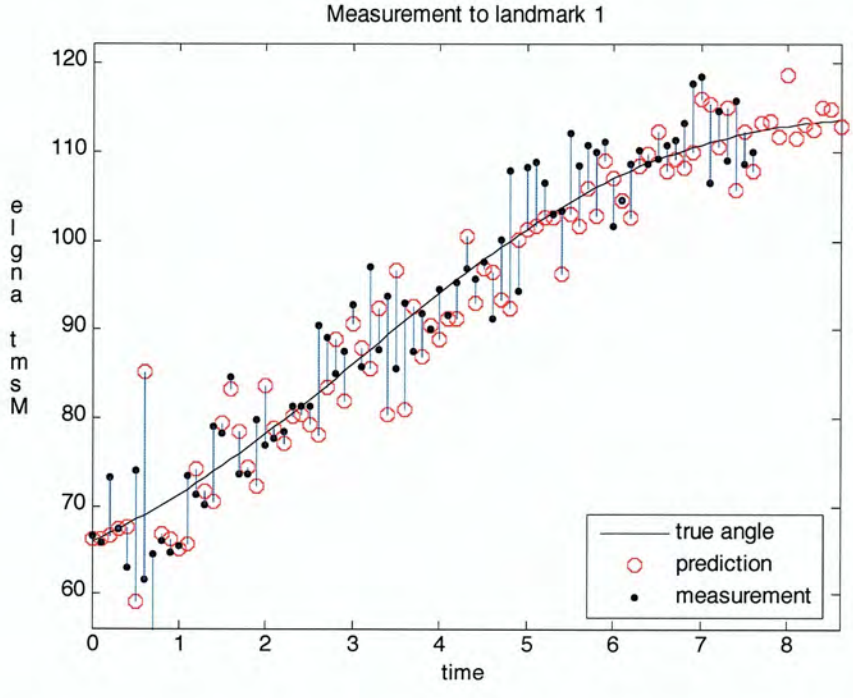

Figure 21. The difference between actual measurements and measurement predictions is called the residual.

The difference between the measured bearing and the predicted bearing is called the *residual*, plotted in Figure 22. A healthy residual should have a mean at zero. If the residual moves away from zero, the filter has a persistent error between the measurement and the prediction of the measurement.

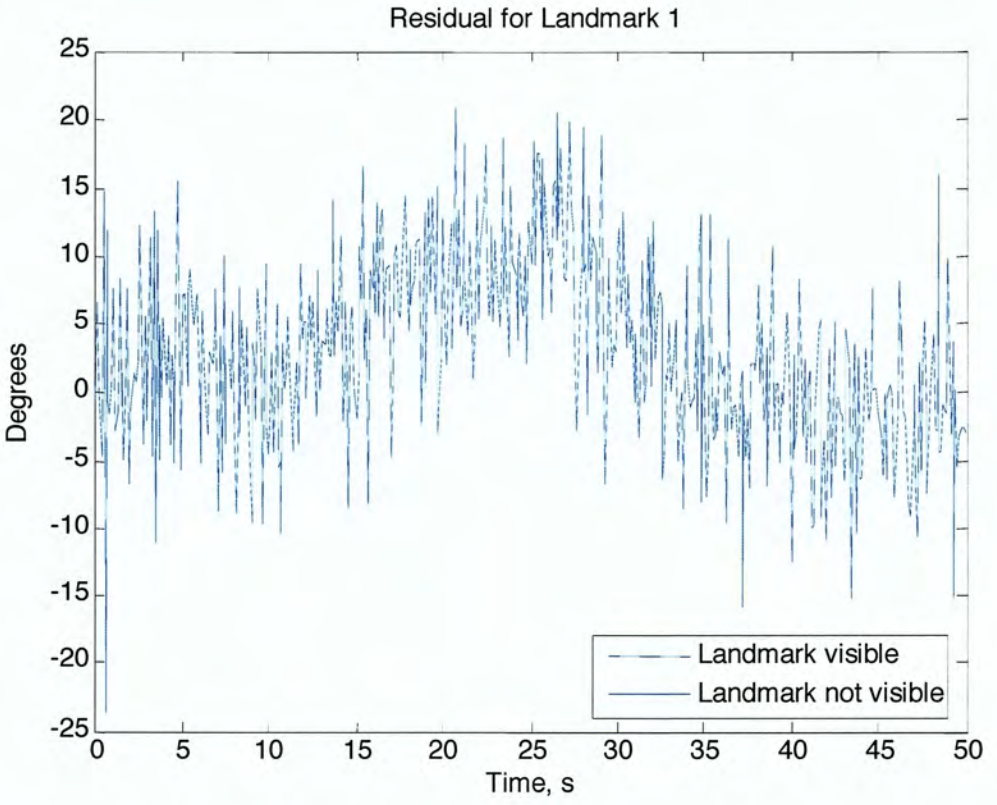

Figure 22. Residual for Landmark 1

To correct these errors, the residual is multiplied by the Kalman gain, which is computed from the state and measurement covariances as well as the process noise matrix.

A Kalman gain is computed to allow each measurement to affect each state.

$$\mathbf{x}(k+1) = \hat{\mathbf{x}}(k) + \mathbf{K}\left(\mathbf{z}^*(k) - \hat{\mathbf{z}}(k)\right)$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_n \end{bmatrix} + \begin{bmatrix} K_{11} & \cdots & \cdots & K_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ K_{n1} & \cdots & \cdots & K_{nm} \end{bmatrix} \begin{bmatrix} z_1^* - \hat{z}_1 \\ \vdots \\ \vdots \\ z_m^* - \hat{z}_1 \end{bmatrix}$$

Figure 23 shows the gains associated with one particular landmark's measurements, P_1. These gains are from the first column of **K** and control how the residuals from landmark 1 affect the state estimate.

Notice in Figure 23 that the gains associated with landmark 1 are reduced to zero at around 8 seconds. This is a trick used in Zarchan to ignore unwanted or unavailable measurements. When the landmark is not visible, the measurement covariance is chosen to be a very high value. This means that the uncertainty in that measurement is very high, which forces the Kalman gain to almost zero, so that this "very uncertain" measurement does not affect the estimate.

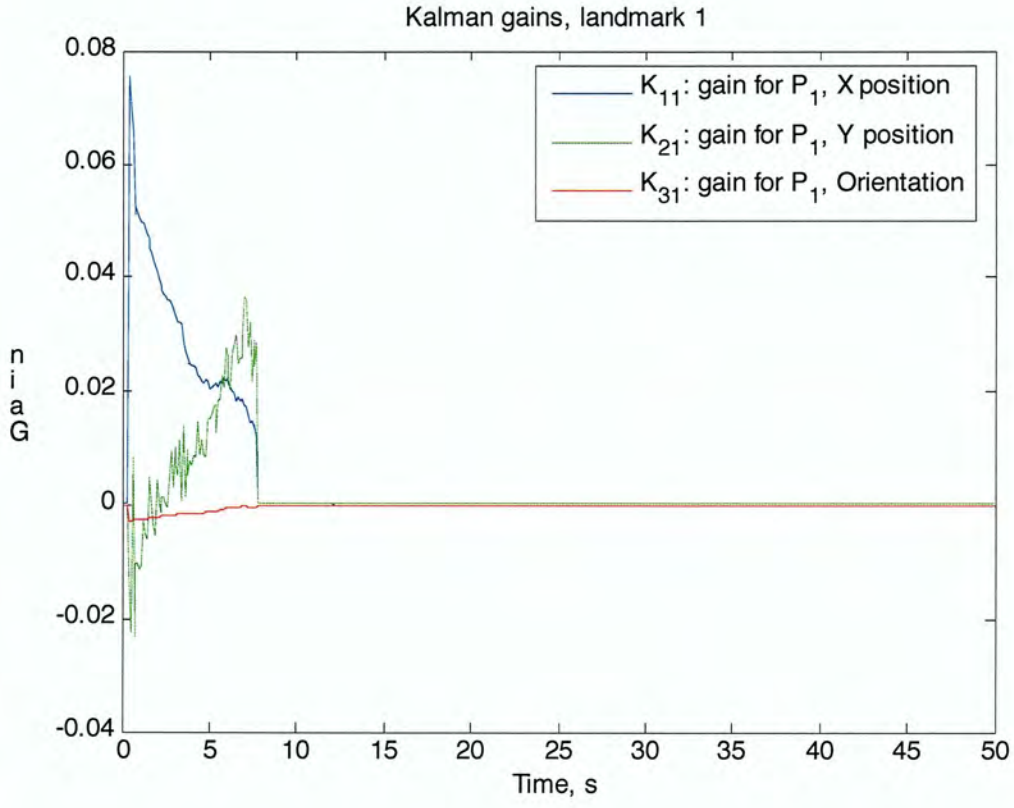

Figure 23. Kalman gains associated with Landmark 1

The next section discusses how the position of the landmarks can affect the quality of the position estimate.

5.2 Relative bearing geometry

Localization works better under some conditions than under others due to the geometry of this particular problem. The following section answers the questions: "How many landmarks are required to find position and

bearing?", "What effect does bearing measurement error have on localization?", and "Given a set of landmarks, what parts of the map provide the best localization?"

5.2.1 Relative bearing and two landmarks

Consider two stationary point landmarks and a mobile vehicle. The vehicle can measure the bearing to each of the landmarks relative to the vehicle's own orientation. That is, the vehicle can determine that one landmark is directly right of it and the other landmark lies more toward the front, but it cannot determine the bearing in any global frame (Figure 24). For example, it cannot determine that one landmark is east of it, and the other is northeast.

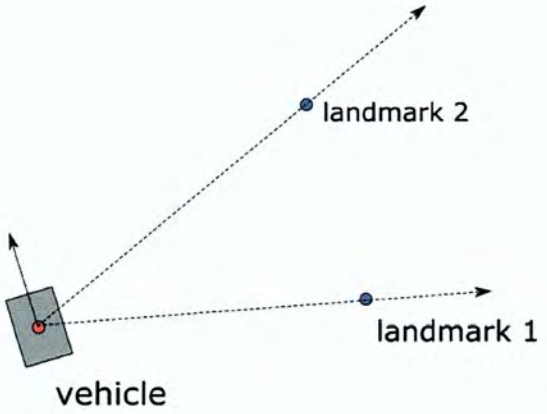

Figure 24. A vehicle measures relative bearing to two landmarks.

An important question in any algorithm used for localization is, "Does this measurement determine the vehicle's location uniquely?" Or are there any other locations where this vehicle could measure the same relative bearing between the two landmarks?

Sketching the system with a few sample orientations indicates that the vehicle could lie anywhere on a specific circle while still measuring the same difference in bearing to the landmarks (Figure 25).

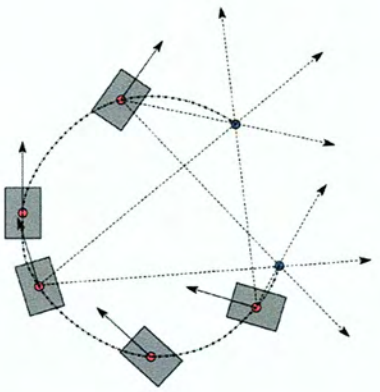

Figure 25. The vehicle measures the same relative bearing measurement from a smooth curve of positions.

5.2.2 Vector loop solution

What is the equation for this circle?

To answer this question, a mechanical engineer might imagine this system as a mechanism with a fixed-angle crank and two pivoting sliders (Figure 26). The sliders represent the landmarks, and the elbow of the crank represents the vehicle.

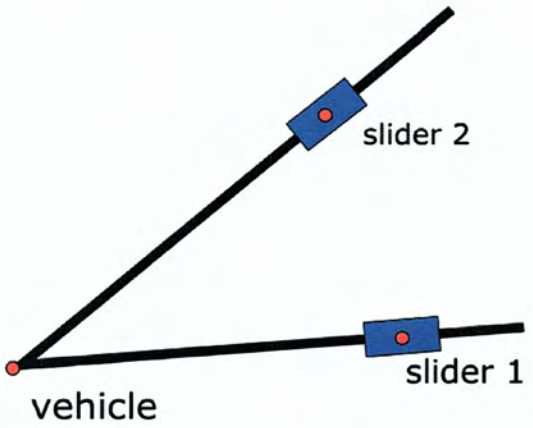

Figure 26. A vehicle with a fixed bearing measurement to two landmarks can be thought of as a mechanism with two rotating sliders.

We can use vector loop equations to describe the location of the vehicle (Figure 27).

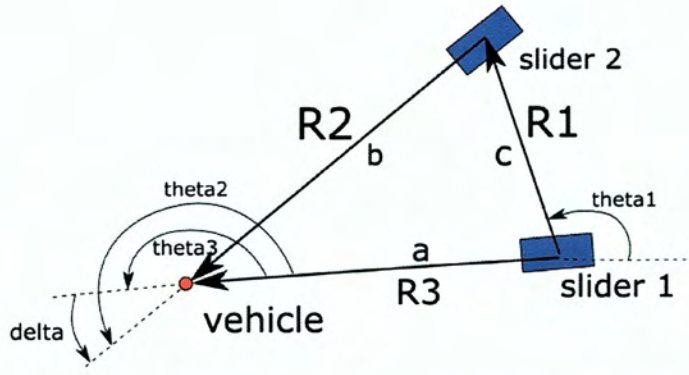

Figure 27. A vector loop equation can be used to solve for the position of the vehicle.

The vector loop equation begins with the equality,

$$\overline{R}_3 - \overline{R}_2 - \overline{R}_1 = 0$$

From this, we can solve for a.

$$a = \frac{c\sin\theta_1 - c\cos\theta_1 \tan(\delta + \theta_3)}{\sin\theta_3 - \cos\theta_3 \tan(\delta + \theta_3)}$$

Since θ_1, c, and δ are constant, we can use θ_3 as an input and plot the possible locations.

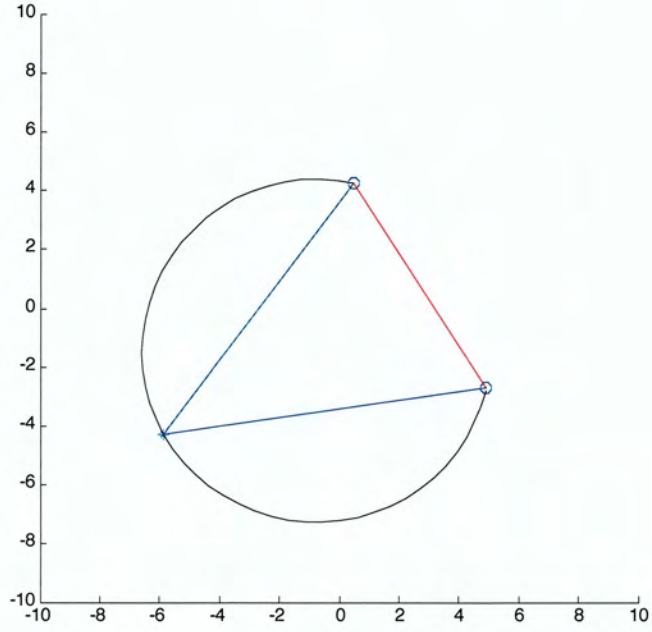

Figure 28. Vector loop solution plotted with one position and bearing measurements as an example.

The equation obtained from the vector loop equation does indeed plot what appears to be a circle (Figure 28).

5.2.3 Solution from geometry

We can verify that this is a circle using a theorem from geometry:

> *The Inscribed Angle Property*: The measure of an angle inscribed in a circle ($\angle ABC$) is half the measure of the arc it intercepts ($\angle AOC$). It follows that all inscribed angles that intercept a given arc have equal measure.

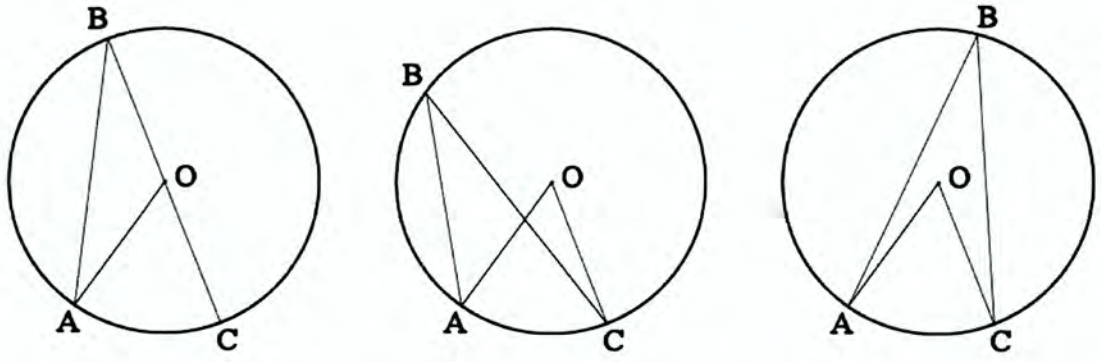

Figure 29. from http://mcraefamily.com/MathHelp/GeometryTriangleInscribedAngleCircle.htm

Our question is actually the inverse of this theorem. We wondered what *shape* would result from a fixed-angle intersecting two points. Some angle intersecting two points could be thought of as intersecting a certain arc, like $\angle AOC$ above. The theorem tells us that "all inscribed angles that intercept a given arc have equal measure." So, from that theorem, our fixed-angle intersecting two points must be *inscribed in a circle.*

The next question is, "Given (1) the location of the two landmarks and (2) the measured angle between them, what is the equation for the circle that describes the vehicle's possible location?"

To find the center of the circle, we must characterize "the arc it intercepts," $\overrightarrow{AOC}$ in the theorem. The center of the arc must lie on a line perpendicular to the center of the line connecting the two landmarks (Figure 30). The arc's center lies in the middle of the circle. Since the legs of the arc $\overrightarrow{OC}$ and $\overrightarrow{OA}$ are both radii, they both

have the same length. This creates an isosceles triangle and means that the center of the circle must lie on a line perpendicular to the center of the line connecting the two landmarks.

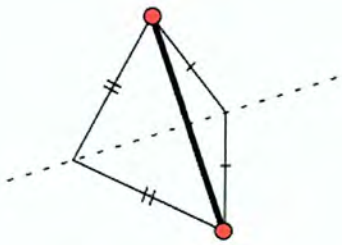

Figure 30. The radii of the arc form an isoceles triangle, so the center of the arc must lie on a line perpendicular to the center of the line connecting the two landmarks.

We can narrow our choices of points on the line by selecting the point that makes an angle of $2^* \Delta\theta$, where $\Delta\theta$ is the measured difference in bearing. This follows from the *Inscribed Angle Property*, "The measure of an angle inscribed in a circle is half the measure of the arc it intercepts." Or the reverse, the arc it intercepts must be twice the measure of the inscribed angle. Keep in mind that the inscribed angle is the bearing measurement. Two points fit this description, one on either side of the line (Figure 31).

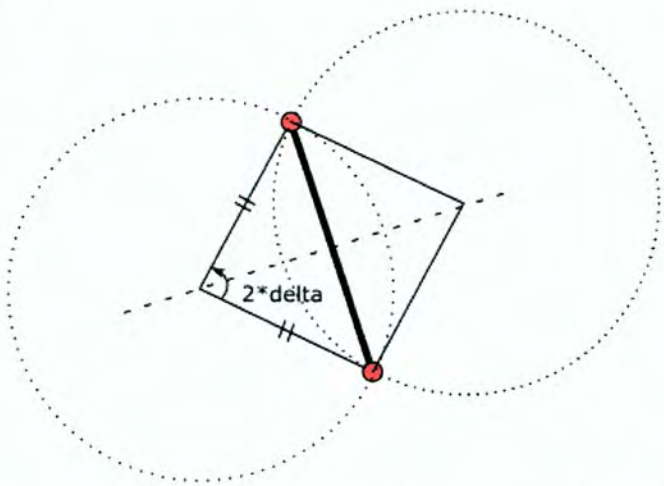

Figure 31. The arc angle must be twice the measured angle.

If landmark 1 can be distinguished from landmark 2, we can choose either the left or right center point. In this example, let's say that the vehicle encounters landmark 1, then landmark 2 while sweeping counter-clockwise. Then the vehicle must lie to the left of the landmarks rather than the right. If it were on the right side, the vehicle would have encountered landmark 2 then landmark 1.

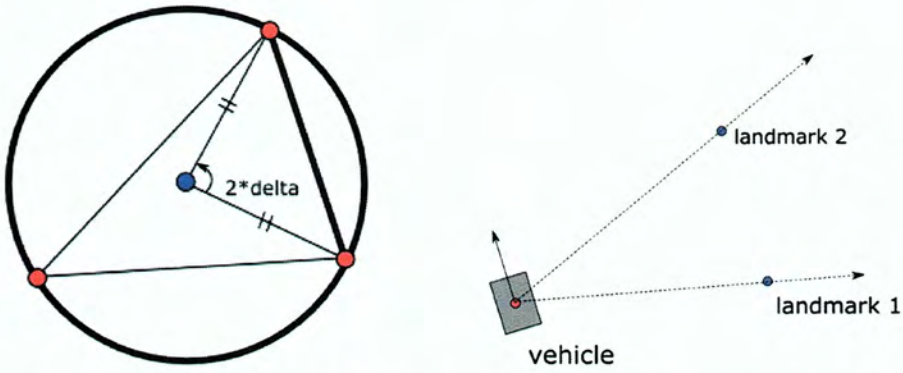

Figure 32. The vehicle location at time of measurement and the arc of possible locations based on that measurement.

5.2.4 Three landmarks

So, given *three* landmarks, three arcs could be drawn. An arc of possible location is drawn for:

- landmark 1 and landmark 2,
- landmark 2 and landmark 3,
- and, landmark 3 and landmark 1.

The point where the three arcs intersect gives the location of the vehicle (Figure 33).

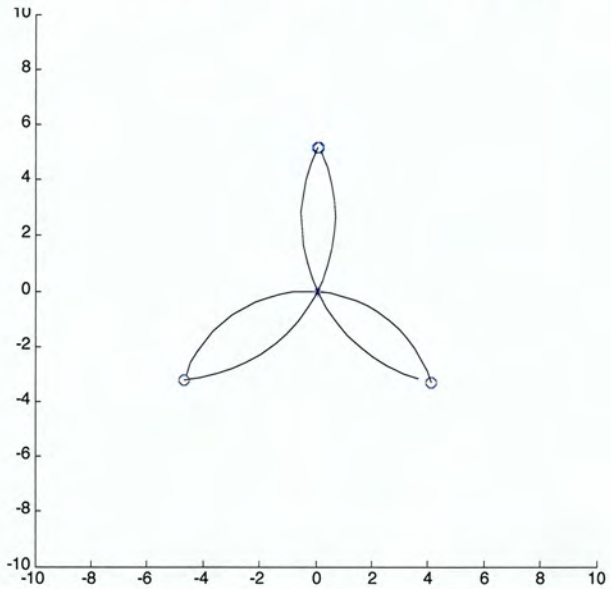

Figure 33. Three arcs of possible location intersect at the location of the vehicle. The third landmark removes ambiguity from the position estimate.

One requirement is that the third landmark is not added to the circle generated by the first two. In this case, the extra landmark does not narrow down the vehicle's location. The vehicle could still lie anywhere on the arc

formed by the first two landmarks, because the measurements are *exactly the same regardless of the vehicle's position on the arc.*

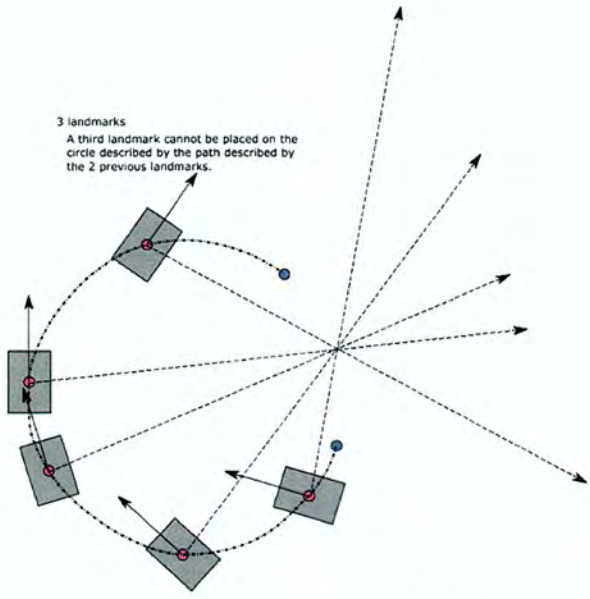

Figure 34. On the circle containing all three landmarks, the same measurement will be made from any position.

This is because the arcs of possible location created by the addition of another landmark are concentric with the original arc. The next figure shows three landmarks and a vehicle taking a relative bearing measurement from a position close to the circle containing all three landmarks. The three arcs of possible location for the vehicle are nearly concentric.

The circle that contains all three landmarks will provide poor localization because the vehicle could lie anywhere on that circle. Even if more than three landmarks are added, but lie on the same circle, the additional landmarks will not provide more information about the vehicle's location.

However, if the landmarks can be uniquely identified, then the vehicle would be able to tell on the circle which landmarks it was between. There will be a gap of at least 180 degrees among the measurements to landmarks on the circle. This gap represents the "outside" part of the circle. The landmarks on the edges of this gap are neighbors to the vehicle. The position of the vehicle would then be narrowed down to somewhere on the arc between those two neighbors.

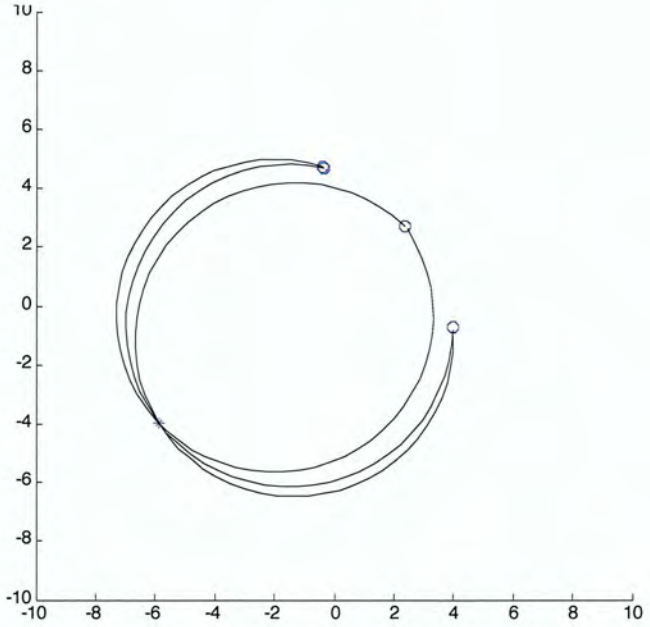

Figure 35. Measurements from the circle that contains all three landmarks. If the the vehicle lies on this circle, it produces arcs of possible location that overlap.

Now, we consider what effect measurement error has on localization.

5.2.5. With measurement error

The sensors on the robot may not measure the relative bearing completely correctly. In computer vision, the pattern recognition process may introduce some error, the vehicle might shudder and shake the camera, or at the very least some tiny error is introduced from choosing a discrete pixel to represent the location of the landmark in the image.

Some error tolerance can be added to the robot measurement of bearing, by adding and subtracting some amount from the measurement. This new upper and lower bound produce two separate arcs of possible location for each pair of landmarks.

With only 2 landmarks, the vehicle could lie anywhere on a broad crescent-shaped arc--what Garulli and Vicino call a 'thickened ring' [2001]. Shown below is a thickened ring arising from a 5 degree error.

With three landmarks, three thickened rings can be drawn, one for each pair of landmarks. The intersection of these rings shows the area where the vehicle could be located (Figure 37).

If the vehicle moves toward the circle that contains all three of the landmarks, the intersected area becomes larger (Figure 38).

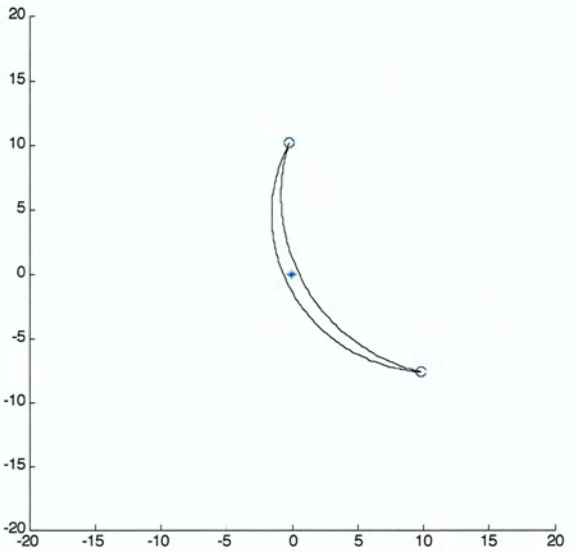

Figure 36. With measurement error, the vehicle could lie anywhere on a "thickened ring." Without considering error, the vehicle would simply lie on a line.

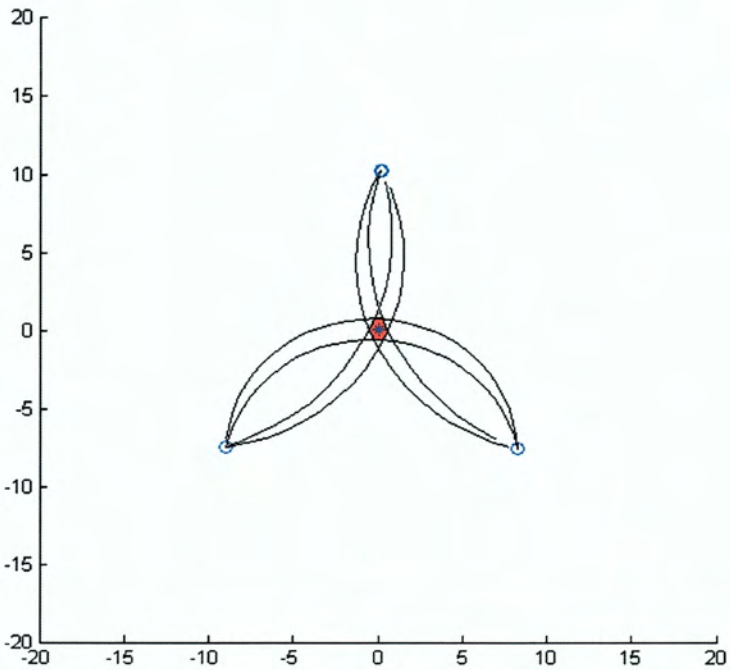

Figure 37. With three thickened rings, the possible vehicle location is limited to the intersection area. When error is considered, three landmarks no longer completely specify the vehicle's location.

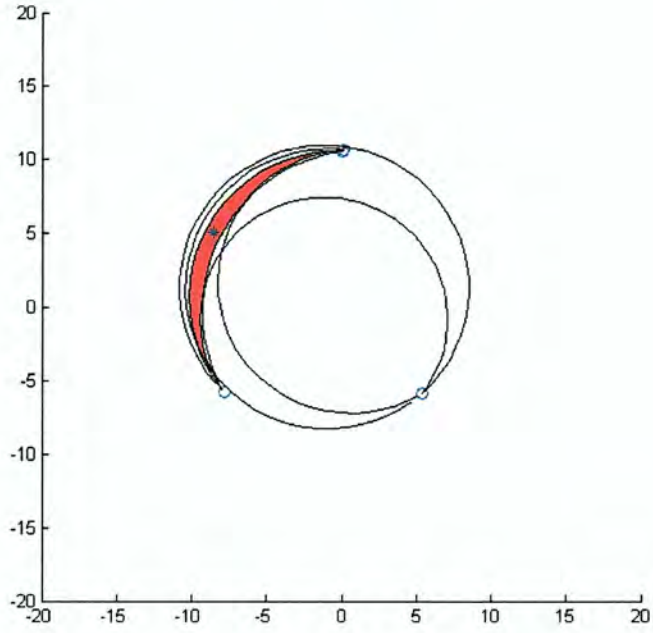

Figure 38. The size of the patch of possible location depends on the vehicle location. If the vehicle lies on the circle that contains three landmarks, the area of the patch will be large.

5.2.6 Geometric considerations during simulation

During the simulation, a colored patch shows the region where the vehicle could be located, based on the measurements to the landmarks in view.

At $t = 3$ seconds, only two landmarks are in view and the thickened ring shows a large area of uncertainty regarding the vehicle's location. At $t = 4$ seconds, three landmarks are in view and the intersection of the three thickened rings gives a smaller area of uncertainty (Figure 39). The large patch of possible vehicle locations at $t = 3$ gives some insight into why the filter's y-position estimate had a 1m error at the beginning of the simulation. When the vehicle acquired a third landmark in its view, the estimate is improved to within 0.1m (Figure 18). Even if all the landmarks are visible, some parts of the map will provide less accurate localization than others. The area of the error intersection patch at a certain position gives an estimation of how accurate the filter will be able to track the robot. For a grid of positions across the map, the area was plotted to show the parts of the map where the filter will be less accurate (Figure 40). Parts of the map near a landmark have a small error. As the vehicle moves away from the landmarks, the error increases. This is because the measurements to the cluster of

landmarks start looking more like measurements to a single point to the filter. Since the cluster of landmarks looks like a single point from far away, they provide less information for localization.

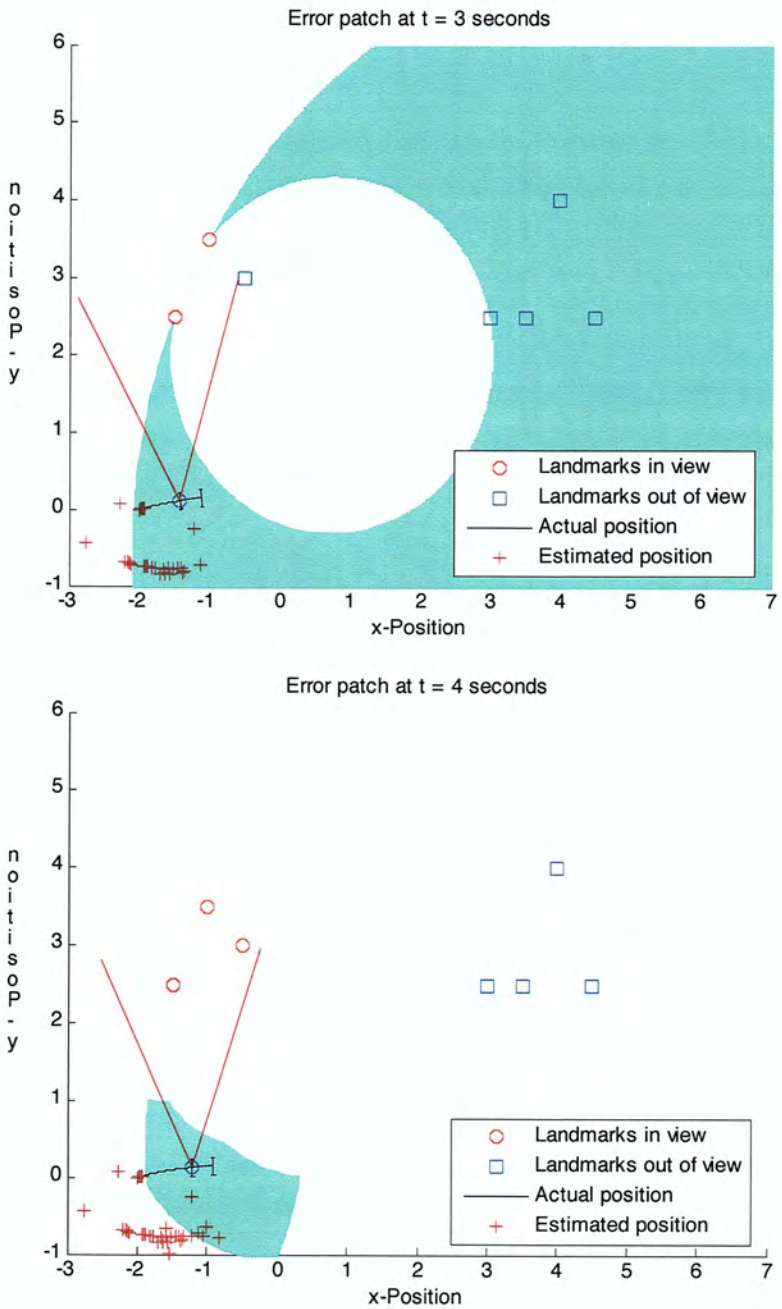

Figure 39. The patch of possible vehicle locations shows areas that may cause problems with the state estimate.

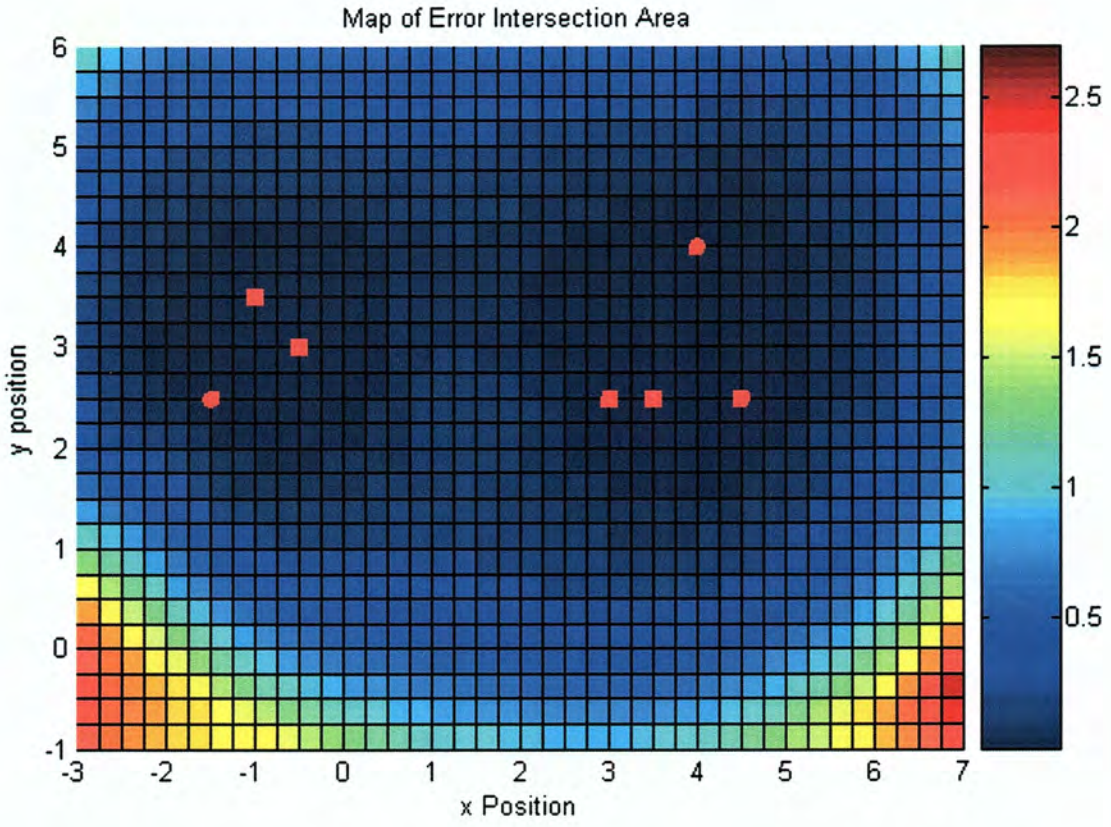

Figure 40. The area of the error patch can be plotted over the workspace of the robot, if it is assumed that all the landmarks are visible. Even with all the landmarks visible, some parts of the map provide less certainty about the vehicle location than others.

5.3 Summary

The Kalman filter described in Section 4 was implemented in simulation. Noise was added to the input steering angle and velocity, in addition to a bias added to the steering angle. This biased steering angle represents the inaccuracies of a vehicle model compared to a real-world vehicle. The odometry-only estimate was influenced by this bias and veered off course. The Kalman filter estimate provided a way to localize the vehicle, but only when landmarks were in view. When no landmarks are in view, the filter behaved the same as the odometry-only estimate. The residual is the difference between the predicted measurement and the actual measurement. When the filter is working correctly, the residual should have a mean of zero, which implies that no persistent error exists.

To better understand the limitations that the layout of the landmarks place on the accuracy of the state estimate, the geometry of the problem was considered. Having two landmarks in view only limits the vehicle to a position somewhere on a circular arc of possible locations. Three landmarks therefore produce three arcs of possible location between each pairing of landmarks. These three arcs overlap and isolate a single vehicle position. However, when measurement error is introduced, three landmarks no longer uniquely locate the vehicle but produce only a bounded area of possible location. The shape of this bounded area provides an indication of how the geometry of the landmark layout may be causing the filter to track poorly.

CHAPTER 6. EXPERIMENTAL SETUP

The filter was also run using measurement data to visual landmarks taken from a CCD camera on a moving vehicle in the real world.

6.1 Landmark recognition

In the computer simulation, it was assumed that the robot could recognize landmarks. This sort of feature recognition is actually a difficult problem in computer vision. This research used an open source software library called ARToolkit in order to recognize the landmarks. The software library was developed for use with *augmented reality* applications, which overlay three dimensional models on flat, patterned fiduciary markers like the one shown below.

Three of the ARToolkit library functions were used:

1) Recognize possible patterns using the square black border
2) Find the marker position and orientation
3) Match the marker to templates in memory

The ARToolkit markers are similar to the landmarks modeled in the computer simulation. Both the simulated landmarks and the real-world ARToolkit markers can be used to represent a point location. The center of the marker was used as the specific landmark point in these experiments. One difference from the simulation, however, is that the ARToolkit markers cannot be identified from all angles. The physical markers are simply printed on a piece of paper and glued to a stiff piece of cardboard. The same pattern could be glued to both

sides, but the pattern recognition function cannot identify the pattern from angles that are too far to the side, starting at about 75-80 degrees from the pattern's normal. A second difference is that the ARToolkit functions are able to extract the pose of the pattern. Although this capability was not used in this research, the pose of the pattern in the camera's image would convey some information about the absolute pose of the robot and could increase the accuracy of the robot position estimation.

6.2 Extracting bearing from an image coordinate

In order to convert the screen location of a marker into the bearing of the marker relative to the camera, some parameters of the camera need to be characterized. This was done using calibration software bundled with ARToolkit. The software prompts the user to align a series of vertical and horizontal parallel lines on a test pattern, repeated at specific distances. The result is a camera projection matrix **P** that converts screen coordinates into points in 3-dimensional space [Hartley, Zisserman 2004].

$$\mathbf{x} = \mathbf{PX}$$

$$\text{screen coordinate} = \left[\text{camera projection matrix}\right] \text{real-world point}$$

The camera projection matrix is a 3×4 matrix that can be written as $\mathbf{P} = \left[M \mid \mathbf{p}_4\right]$. The actual values returned by the camera calibration for a 640x480 image are shown below.

$$P = \begin{bmatrix} 562.3 & -1.7 & 282.1 & 0.0 \\ 0.0 & 559.7 & 241.8 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}$$

Hartley presents the following method for back-projection of points to rays: Find the camera center in world coordinates to be the start of the ray, and then back-project a point on the screen to "the plane at infinity", a simplifying concept used in projective geometry. In any case, this back-projected point provides a second point for the ray.

The camera center is

$$\tilde{C} = -M^{-1} p_4$$

The second point is

$$D = \begin{bmatrix} M^{-1}\mathbf{x} \\ 0 \end{bmatrix}$$

These two points give a vector V_1 by subtraction.

$$V_1 = D - \tilde{C}$$

Two such vectors emanating from the focal point of the camera are used to determine the bearing measurement. This first vector is made from the image center of the camera and the second from the marker location on the screen. Since we are only interested in the vehicle's plane, only the x-coordinates are used in a dot product that returns the angle between them.

$$\theta = \cos^{-1}\left(\frac{V_1^T V_2}{|V_1||V_2|}\right)$$

If the camera is mounted at an angle so that the screen's x-coordinates do not match the vehicle plane, the points could be rotated by that mounting angle before finding the bearing.

6.3 Results

The techniques above were used to extract bearing data in an experimental setup. A camera was mounted to the mobile platform and oriented 90 degrees to the vehicle's right.

The following steps were performed:

1. Landmarks were set up in a known position. Vehicle began in a known position, orientation and steering angle.
2. The vehicle was moved forward for 2m while recording video from camera.
3. ARToolkit was used to find screen coordinates of markers.
4. The bearing was extracted from the screen coordinates.
5. The filter was run with the landmark locations and the bearing data.

Two considerations should be noted here. First, the filter was not run in real-time—the test was run, the image data was preprocessed and then entered into the filter. Based on the running time of the programs involved, it should be possible to run all the processes in real time. ARToolkit is designed for real-time work, and the

Kalman filter is relatively simple, with only a series of matrix multiplications and a matrix inversion. These experiments are offered as a demonstration of using computer vision to identify and measure bearing from landmarks in the real world.

Second, the odometry data for this test was estimated. At the time of the test, the data from the wheel encoder was not available. Instead, the total distance was simply divided by the total time to give the average speed, for example, 2m /16 seconds = 0.125 m/s. This actually makes it harder for the filter, since the odometry data does not quite match the bearing measurement data. The estimation of velocity could be thought of as having noisy measurements that differ from the true value.

6.3.1 Experiment 1: Without added steering bias

The wheels of the vehicle were aligned so that the robot traveled straight forward for 2m and then stopped. The landmarks were positioned as shown in Figure 41.

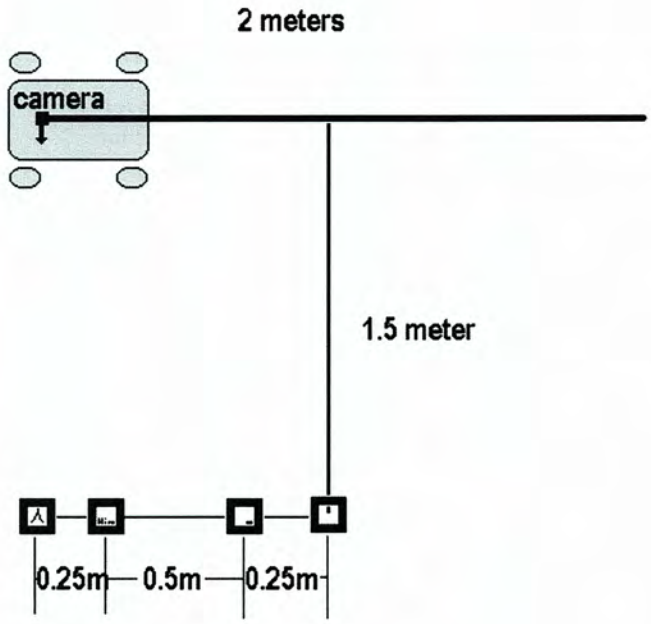

Figure 41. Layout of first test, with accurate odometry.

The actual position was available only for the start and the end of the test, where the position was measured by hand.

The visibility of the landmarks is shown in Figure 42. Two landmarks are initially visible, followed by a short period where all four are visible. The angles that were extracted from the screen coordinates of the markers are plotted in Figure 43. These measurements were then used in the Kalman filter.

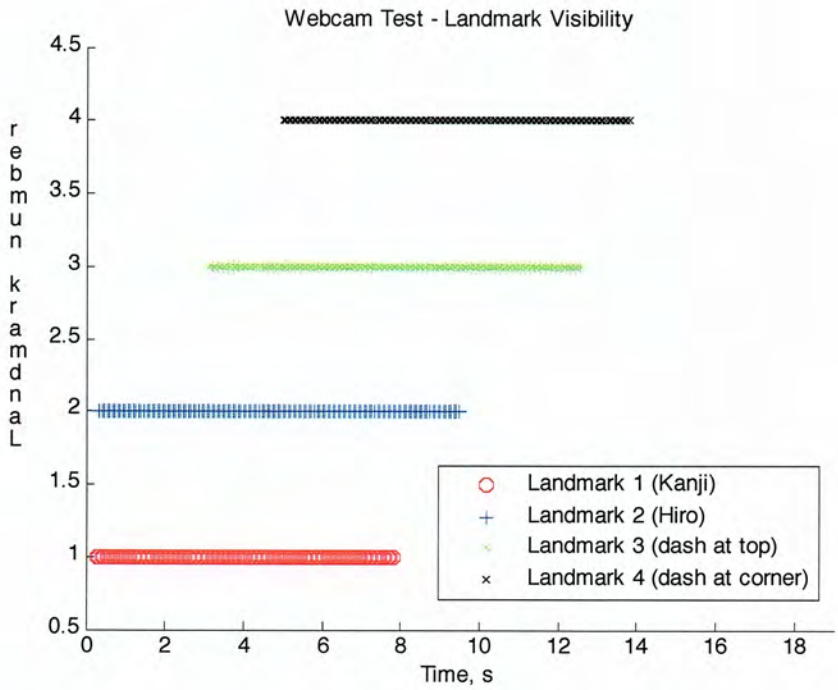

Figure 42. Visibility of landmarks. Landmarks 1 and 2 are initially visible, followed by landmarks 3 and 4 coming into view.

The plot below shows the results of the test. The odometry-only estimate was correct for this experiment. The Kalman filter estimate ended with 0.2m error. Note that after the robot got to 1.3 m, the camera had less than two landmarks in view. This test shows that "coasting" on the filter's estimate from odometry when no landmarks are visible still provides reasonable estimates. But, without the correction from bearing measurements these estimates become less reasonable over time, as odometry-only estimates of position tend to do.

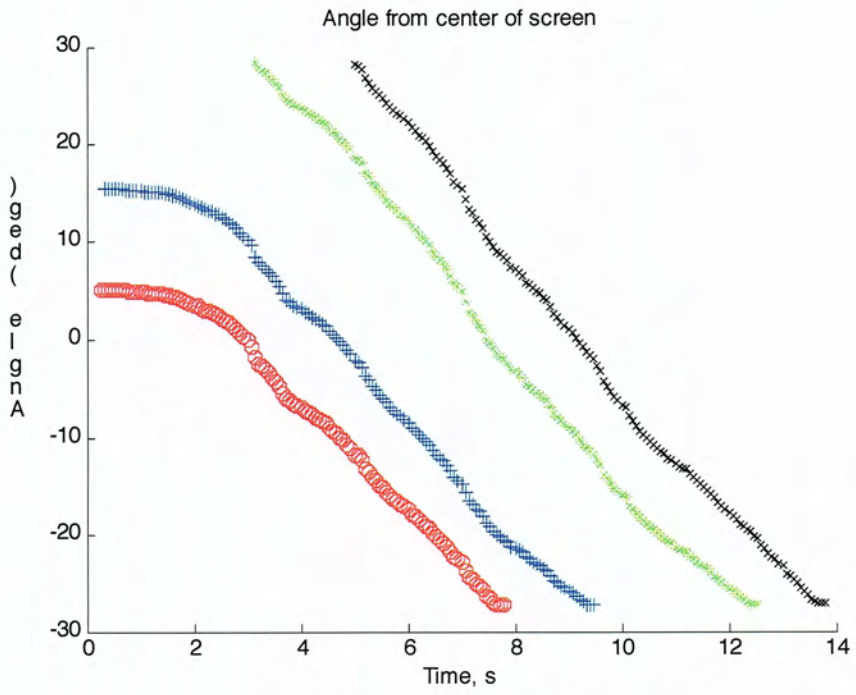

Figure 43. Angles computed from screen coordinates

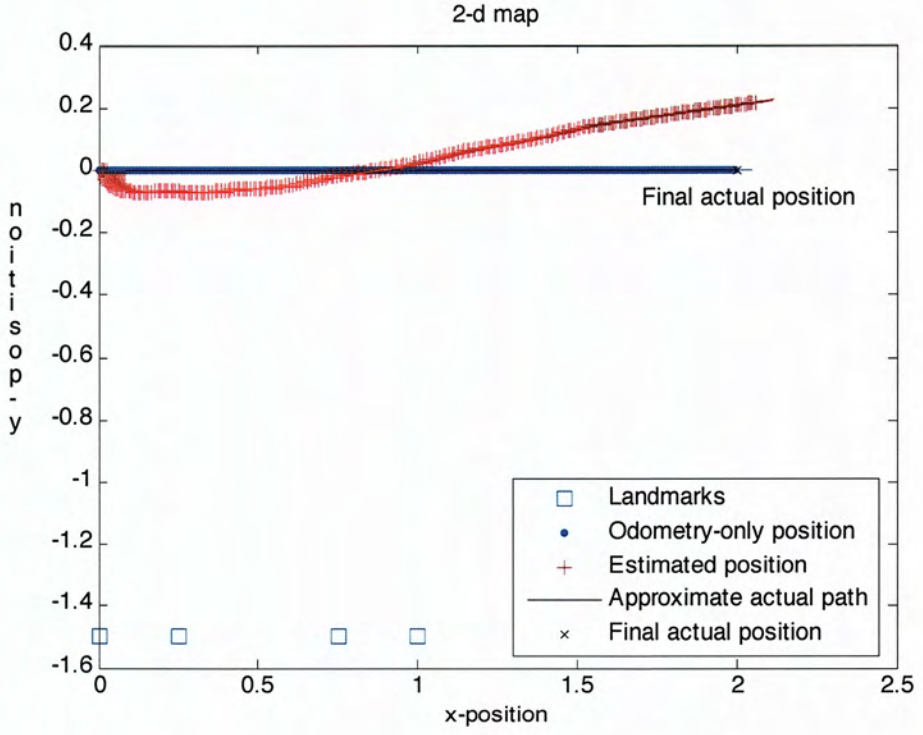

Figure 44. A map of the filter's output for Experiment 1. The filter tracks the position, even with estimated velocity to within 0.1m, then veers off when the landmarks are no longer visible.

6.3.2 Experiment 2: With steering bias

A second test was run in which the wheels were shifted slightly to the right. The filter's input data still indicated that the vehicle was traveling straight, with a steering angle of zero. Also, in this experiment the movement of the vehicle caused the ARToolkit markers to be momentarily lost or misreported. The results will show how the filter reacts to this extra noise in the measurements.

The landmark visibility is shown Figure 45. Note that at around six seconds, visibility of each of the landmarks is lost.

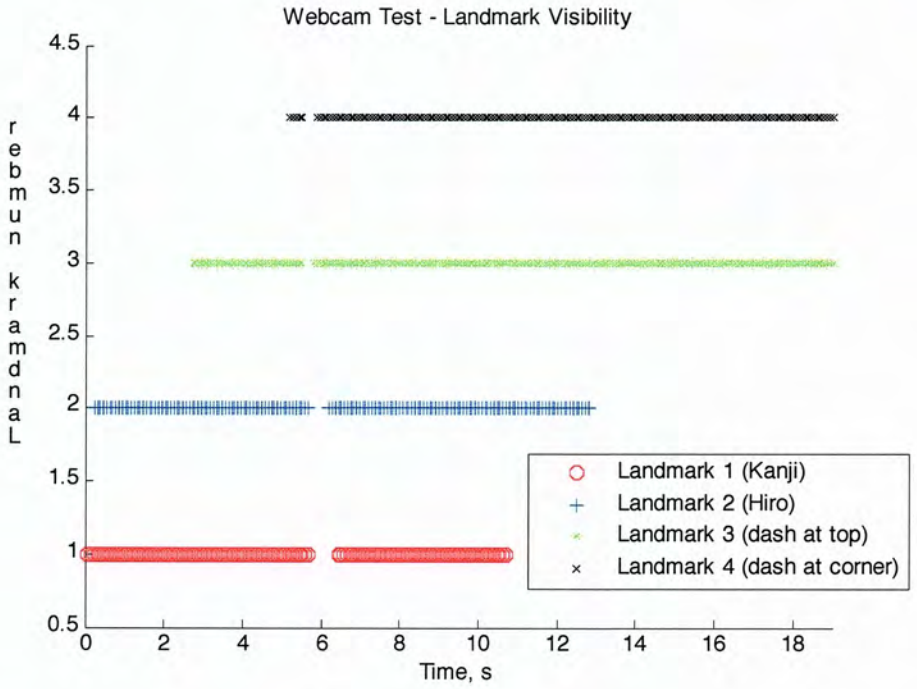

Figure 45. Visibility of landmarks. The landmarks are not visible for a short period around $t = 6$ due to shaking of the camera.

The angles that were extracted from the screen coordinates are plotted in Figure 46. During the loss of visibility Landmark 1 is mistaken once for Landmark 3 and then a moment later for Landmark 4.

The results of this second test are shown below in Figure 47. First, at about 0.4m, the filter's position estimate moves rapidly. This is due to the misreported landmarks at $t = 6$. If landmark 3 and then landmark 4 were actually where they were reported to be, the location of the vehicle would be very different. The filter made a prediction about what the bearing measurement would be based on a predicted location of the robot. The

predicted measurement was subtracted from the actual (and in this case, incorrect) measurement. This difference forms the residual, which is multiplied by the Kalman gain and added to the predicted position as a correction. For the misreported landmark, the residual was large and skewed the position estimate.

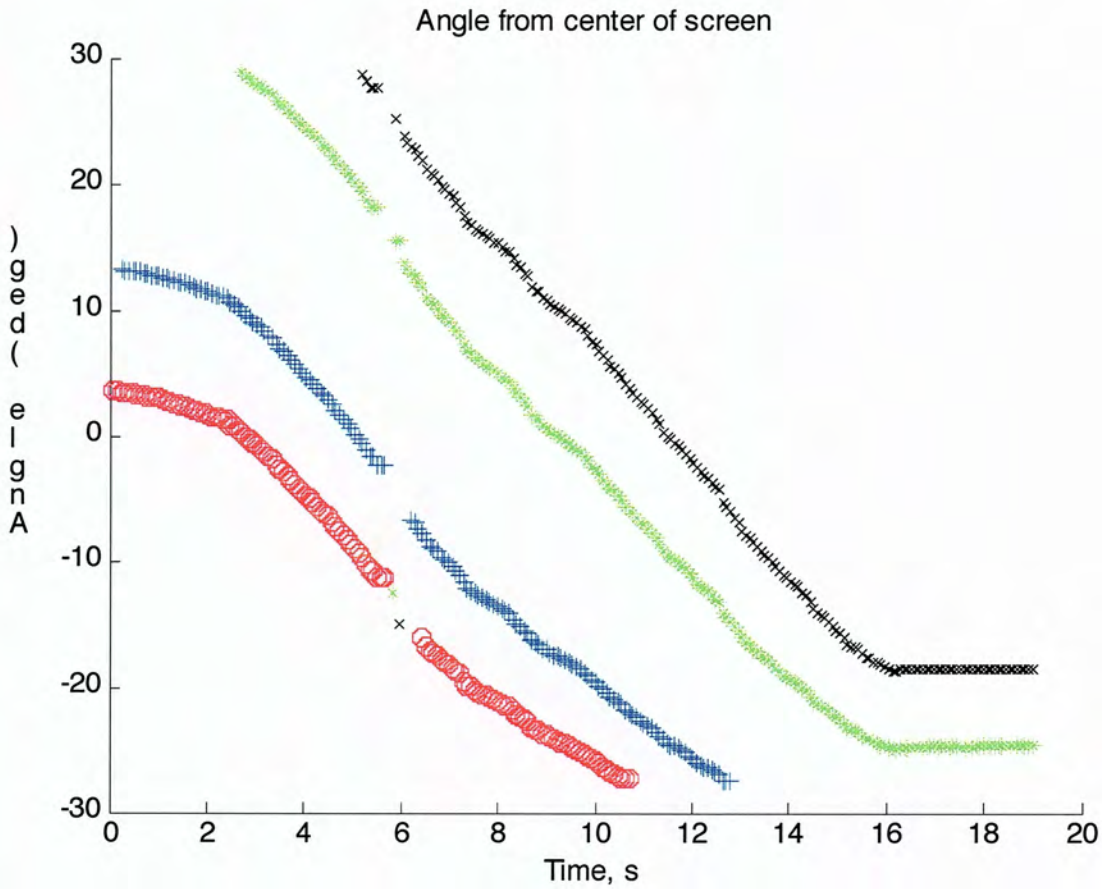

Figure 46. Bearing angle measurements calculated from screen coordinates. Landmark 1 is misidentified as landmark 3 and then landmark 4.

What effect did the biased steering angle measurement have on the estimate? The vehicle ended up steering to the right, but the odometry-only estimate remained straight. The measurements from the landmarks moved the filter estimate toward robot's final location with a final error of about 0.3m.

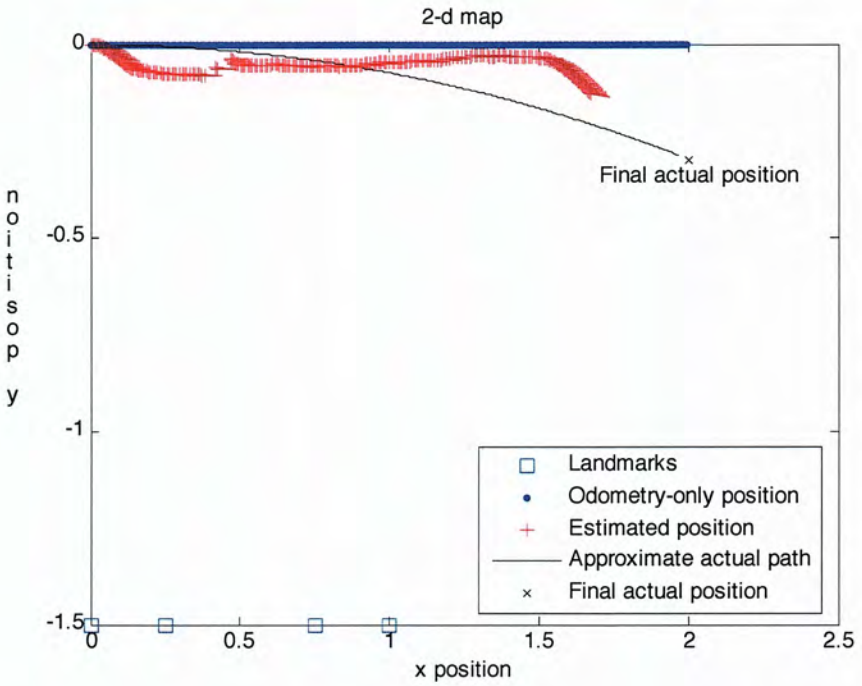

Figure 47. A map of the filter's output for Experiment 2. The filter recovers from the misidentified landmark, but stops short of the final position. The filter estimate's final heading orientation is very close to the actual final heading orientation.

6.4 Summary

The actual application of this filter to track from bearing measurements to known landmarks requires a computer vision system to recognize those landmarks. The first step is pattern recognition in order to identify the landmarks, and then extracting the bearing information using the characterized camera's projection matrix. Two experiments were run. The first experiment involved no odometry errors or measurement errors. The filter ended within 0.2m of the final position after a 2m run. In the second experiment, the robot had a steering angle measurement bias as well as a momentary loss of visibility due to motion of the camera. The outlying measurements affected the position estimate, but the filter was able to recover. The filter estimate moved toward the final actual location of the robot, but stopped 0.3m short.

CHAPTER 7. CONCLUSION

7.1 Summary

Localization is a fundamental part of mobile robotics. This research presents a filter that uses a map of known landmarks to localize a robot. In order for a robot to locate itself in the environment sensors are needed, as discussed in Chapter 2. Proprioceptive, internal sensors are those used for odometry, and exteroceptive, external sensors include radar, lidar, sonar and CCD cameras. Cameras are inexpensive and data-rich, but a single camera only provides bearing information. Several techniques can be used to localize a robot with data from these sensors. Dead reckoning from odometry data is a simple technique, but it provides no way to globally localize the robot. The particle filter is a promising new technique that uses multiple hypotheses and probability to find the most likely location of the robot. However, the technique evaluated in this research is the Kalman filter.

Chapter 3 introduced the Kalman filter as an observer from automatic controls that is capable of estimating the internal states of a process. The Kalman filter uses differences between measurements that are predicted by an internal model and actual measurements taken by sensors. This difference between predicted and actual measurements is multiplied by a gain that is calculated by the Kalman filter and then added to the predicted state vector. The Kalman gains use statistics about the process and measurement of the problem, as well as stored statistics about the uncertainty level between states.

In Chapter 4, the specific components needed for the Kalman filter were developed. The state dynamics matrix, the measurement matrix, the process noise matrix, and the initial state estimation were each addressed. The simulation of the developed filter was discussed in Chapter 5. From the simulation, it was clear that the input noise and bias severely affected accuracy of the odometry-only estimate. The Kalman filter estimate used the bearing measurements to localize the vehicle, but only when landmarks were in view. When no landmarks are in view, the filter behaved the same as the odometry-only estimate.

The second part of Chapter 5 discussed how the layout of the landmarks can affect the accuracy of the state estimate. With two landmarks in view the vehicle is only constrained to a position somewhere on a circular arc of possible locations. Three landmarks therefore produce three of these arcs of possible location, one arc

between each pairing of landmarks. These three arcs overlap and isolate the vehicle at a single position. However, when measurement error is introduced, three or more landmarks no longer uniquely locate the vehicle but produce only a bounded area of possible location. The shape of this bounded area provides an indication of how the geometry of the landmark layout affects the filter's ability to track.

Chapter 6 showed two test runs of the filter using bearing measurements to real-world landmarks. A computer vision system was required to recognize the landmarks. The vision system first used pattern recognition to identify the landmarks, and then extracted the bearing information using the characterized camera's projection matrix. The first experiment was run with estimated velocity, and accurate steering angle. The filter ended within 0.2m of the final position after a 2m run. In the second experiment, the robot had a steering angle bias as well as a momentary loss of landmark visibility due to motion of the camera. The outlying measurements affected the position estimate, but the filter was able to recover. The filter estimate moved toward the final actual location of the robot, but stopped 0.3m short.

The contribution of this research is its development of application of the Kalman filter to an interesting engineering problem. The use of a camera as a sensor presents a few challenges for the problem of localization. It was shown that three landmarks are enough to theoretically localize a vehicle, as long as the vehicle does not lie on the circle containing the three landmarks. A fourth landmark, if it does not lie on the circle containing the first three, provides enough information to uniquely identify the vehicle's position. However, when the measurements are considered to be imperfect, the location is no longer uniquely defined by any number of landmarks but is represented by a bounded area. The calculation of this area can be applied to a given landmark map to show parts of the map that provide less accurate localization. Even with the same number of landmarks in view, the geometric layout can cause the level of uncertainty about position to differ.

The characterization of the uncertainty of the vehicle's position as a bounded area raises an interesting point. The Kalman filter stores uncertainty about the states in the state error covariance matrix, **P**. This matrix describes the error in the states using only the statistic of variance. This is because one of the three Kalman filter assumptions is the assumption that the error can be modeled by a Gaussian curve. However, the crescent-shaped error that arises from having two landmarks in view cannot be characterized as Gaussian. It is true that the time-tested usefulness of the Kalman filter is its ability to perform well outside its own assumptions, for

example: the non-linearity assumption is violated by the extended Kalman filter, white noise processes do not even actually exist, and most error is only approximately Gaussian. This research has shown that the Kalman filter can be effectively applied to this problem, even with stretching the rules. Despite the fact the Kalman filter works, the bounded area error of this problem might be more appropriate for the particle filter. One of the particle filter's key features is the ability to handle non-Gaussian error. The particle filter could handle the crescent-shaped area by simply coalescing particles in the shape of a crescent based on the agreement of the predicted measurement with the actual measurement.

Another challenge from using a camera is its limited field of view. It is difficult to keep three markers on screen at once. One improvement would be to use an actuated camera to maximize the number of landmarks in view. Another option would be to use a 360 degree camera or a series of cameras with their views "stitched" together. The alternative is to use many more landmarks, so that a robot with a limited view always has plenty to see. The idea of having so many markers disqualifies this method of localization from use in the public sphere, where humans may not like having distinctive markers all over. The most appropriate application of this system of localization would be in an industrial setting; for example, localizing Automated Guided Vehicles (AGV) in an automobile factory. Factors such as problems with landmark occlusion might cause problems even there, though.

7.2 Future work

Some immediate future work would include incorporating more than one camera or a spherical mirror lens that can view 360 degrees. This would allow more accurate tracking by keeping more landmarks in view.

The next step in this line of research would be to remove the requirement for known landmarks. Some research has already been done in this area [Bailey 2003], [Fitzgibbons, Nebot 2002], [Deans, Hebert 2000]. Most of this work uses the SLAM framework, which is based on the Kalman filter. The results of this work indicate that the landmark initialization problem presents the biggest challenge. From the lack of a clear and complete solution to the bearings-only localization and mapping problem in the literature, it might be necessary to explore a method different than the SLAM framework.

Another step that could be taken with the present research would be to explore navigation strategies that work well in a bearings-only setting that could be used whether the landmarks were known or not, expanding the work of Sim [2005].

WORKS CITED

T. Bailey. Constrained Initialisation for Bearing-Only SLAM.In IEEE International Conference on Robotics and Automation, Taiwan, 2003..

Y. Bar-Shalom, T.E. Fortman, Tracking and Data Association, New York: Academic, 1988.

K. Briechle, U.D. Hanebeck. Localization of a Mobile Robot Using Relative Bearing Measurements. IEEE Transactions on Robotics and Automation, 2004.

R. G. Brown, P. Y .C. Hwang. Introduction to Random Signals and Applied Kalman Filtering with MATLAB Exercises and Solutions, Third Edition. John Wiley and Sons, New York. 1997.

M. Deans, M. Hebert. Experimental comparison of techniques for localization and mapping using a bearing-only sensor. Seventh International Symposium on Experimental Robotics (ISER), 2000.

M. Deans. Robust and Efficient Simultaneous Localization and Mapping from Bearings-Only Sensing. Thesis Proposal, Carnegie Mellon University. 1999.

M. Deans, M. Hebert. Invariant Filtering for Simultaneous Localization and Mapping, Proceedings from the IEEE International Conference of Robotics and Automation (ICRA), 2000.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo Localization for Mobile Robots. IEEE International Conference on Robotics and Automation (ICRA), 1999.

F. Dellaert, W. Burgard, D. Fox, S. Thrun. Using the CONDENSATION Algorithm for Robust, Vision-based Mobile Robot Localization. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99) - Volume 2 p. 2588, 1999.

F. Dellaert, A. Stroupe. Linear 2D Localization and Mapping for Single and Multiple Robot Scenarios. Proceedings of the IEEE International Conference on Robotics and Automation. 2002.

M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. In Robotics and Automation, IEEE Transactions, Volume 17, Issue 3, pages 229-241. Jun 2001.

T. Fitzgibbons, E. Nebot. Application of Vision in Simultaneous Localization and Mapping, Proc. 2001 Australian Conference on Robotics and Automation 2001.

T. Fitzgibbons, E. Nebot. Bearing-Only SLAM using Colour-based Feature Tracking. Proceedings of the Australasian Conference on Robotics and Automation. 2002.

E.W. Frew, S. Rock. Exploratory Motion Generation for Monocular Vision-Based Target Localization. Proceedings of the IEEE Aerospace Conference. 2002.

A. Garulli, A. Vicino. Set Membership Localization of Mobile Robots via Angle Measurements. IEEE Transactions on Robotics and Automation, Volume 17, Number 4, August 2001.

R. Hartley, A. Zisserman. Multiple View Geometry in Computer Vision, Second Edition. Cambridge University Press, 2004.

H. Kato, M. Billinghurst. Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System. In Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99). October, San Francisco, USA. 1999.

R.E. Kalman. A new approach to linear filtering and prediction problems. Journal of Basic Engineering, ASME Vol. 82, pp. 35-45, 1960.

J.P. Le Cadre, . Jauffret. On the Convergence of Iterative Methods for Bearings-Only Tracking. IEEE Transactions on Aerospace and Electronic Systems, 1999.

J. J. Leonard and H. F. Durrant-Whyte, Directed Sonar Sensing for Mobile Robot Navigation. Boston, MA, USA. Kluwer Academic. 1992.

P.S. Maybeck. Stochastic Models, Estimation, and Control. Mathematics in Science and Engineering, Volume 141, 1979.

N. Nise. Control Systems Engineering. Wiley, 2003.

J.S. Ortega, T. Lemaire, M. Devy, . Lacroix, A. Monin. Delayed vs. Undelayed Landmark Initialization of Bearing Only SLAM. Workshop on Simulateous Localisation and Mapping International Conference on Robotics and Automation, Barcelona, Spain, 2005.

Y. Oshman, P. Davidson. Optimization of Observer Trajectories for Bearings-Only Target Localization. IEEE Transactions on Aerospace and Electronic Systems, 1999.

R. Sim. Stable Exploration for Bearings-only SLAM. IEEE International Conference of Robotics and Automation (ICRA), 2005.

R. Smith, M. Self, P. Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. Autonomous Robot Vehicles, pages 167-193, 1990.

S. Thrun. Robotic Mapping: A Survey. CMU-CS-02-111, 2002.

S. Thrun, W. Burgard, D. Fox. Probabilistic Robotics. MIT Press, 2005.

P. Zarchan, H. Musoff. Fundamentals of Kalman Filtering: A Practical Approach, Second Edition, Revised. From the Progress in Astronautics and Aeronautics Series, 208. American Institute of Aeronautics and Astronautics, 2005.