Bare Minimum SLAM: A Simultaneous Localization and Mapping Tutorial for Undergraduate Engineering Students

Adam A. Wright, Nathan Swaim, Orko Momin
Advisor: Dr. David Ahlgren
Trinity College

May 10, 2010

Contents

1 Introduction

The purpose of this tutorial is to make SLAM more accessible to undergraduate engineering students. SLAM requires certain knowledge in the areas of probability, linear algebra, and stochastic control systems that were not addressed in our own undergraduate classes. It was also found that the tutorials available left out a lot of intermediate steps required to make the project work. We hope that by publishing this tutorial, we will sufficiently explain the lower level steps required to make a SLAM robot. The tutorial was developed as part of our capstone senior design project at Trinity College.

Section 2 gives some background required to understand the concepts involved in SLAM. The section covers concepts in probability, statistics, linear algebra, and stochastic control systems. If you are well versed in these areas, skimming section 2 should suffice. Section 3 describes the basic hardware and software setup used in our SLAM robot. Sections 4 and 5 describe the idea behind Kalman filter SLAM and explain what the Kalman filter is and the stochastic control system used in Kalman filter SLAM. Section 6 explains how to implement two of the main experiments completed in our senior project. The first is an experiment involving only one landmark, and the second involves three landmarks.

2 Mathematics Preliminaries

2.1 Linear Algebra

In this section it is assumed that readers have minimal experience with matrix concepts, such as matrix multiplication, matrix transpose, and matrix inverse.

Definition 2.1. Identity Matrix
The identity matrix of size n, denoted I_n is defined to be the $n \times n$ matrix

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

2.2 Probability and Statistics

A random variable is a function describes the outcome of some experiment. Mathematically, a random variable assigns a value to each element in some sample space. A sample space is a collection of possible events. If we were talking about flipping a coin two times, a suitable sample space would be

$$\Omega = \{HH, HT, TH, TT\},$$

where T denotes tails and H denotes heads. An example of a random variable on the sample space Ω would be the random variable X, where X is the number of heads after two coin flips. For this random variable, $X(HH) = 2$, $X(HT) = X(TH) = 1$, and $X(TT) = 0$.

A type of random variable that is very important to this tutorial is one where the sample space is $\mathbb{R}$. Such a random variable could be used to describe the act of taking a sensor measurement. Let's use the example of taking a distance measurement using a LIDAR (LIght Detection And Ranging) sensor. The sample space $[0, \infty)$ could describe the actual distance of the object being measured. We will define the random variable Y as the distance measurement of the LIDAR sensor.

Another important concept is that of a Probability Density Function, commonly referred to as PDF, or density. The density of a random variable corresponds to the probability that the random variable will take on a certain value. As such the input is a possible value that the random variable might take on, and the output is a number between 0 and 1 that is the probability of the random variable being that value. One possible density for the random variable Y, called $f_Y(y)$ is the one shown in figure 1. The most important property of a density function is that the area under the curve from $-\infty$ to ∞ is 1.

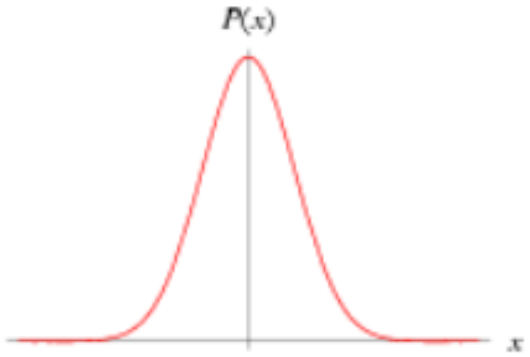

Figure 1: Possible density for Y: the Gaussian distribution.

The following are mathematical definitions of the mean and variance of a random variable.

Definition 2.2. Expectation
The expectation of a continuous random variable V is the mean of V, or

$$E[V] = \int_{-\infty}^{\infty} v f_V(v) dv.$$

The mean of a random variable is often denoted μ_V. The expectation is important for SLAM since it is used in the definition of the covariance matrix. Intuitively, this can be thought of as the value you would expect the random variable V to take on.

Definition 2.3. Variance
The variance of a random variable V is

$$\text{var}(V) = E\left[(V - \mu_V)^2\right].$$

We note that the variance is the standard deviation, σ, squared and as such is frequently denoted σ^2. The variance is an indication of how spread out over the real line the area under the density function of a random variable is.

There are several named probability distributions that have a specific density function associated with them. One example of a distribution that is central to this tutorial is the Gaussian (also called normal) distribution. A random variable X is said to be a Gaussian random variable if it has density

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}},$$

where μ and σ^2 are the mean and variance of X, respectively [11]. Gaussian density functions look like the one shown in Figure 1. This shape is called the bell curve. Notice that if the variance is increased, the width of the "bell" becomes wider, and if the variance is decreased, the bell becomes narrower.

Definition 2.4. Covariance
The covariance between two random variables V and W is

$$\operatorname{cov}(V, W) = E\left[(V - \mu_V)(W - \mu_W)\right].$$

Note that the covariance of a random variable with itself is just the variance of that random variable. Covariance describes how related two random variables are.

Definition 2.5. Random Vector
A random vector $\vec{v}$ is a function from some sample space into $\mathbb{R}^n$. of random variables $v_1, \ldots, v_n$ is

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}.$$

The density of V is denoted $f_V(v_1, \ldots, v_n)$, and outputs the probability that the random vector will take on the value $(v_1, v_2, \ldots, v_n)$. The mean of a random vector is just the mean of the components of the vector. The next definition is for the covariance matrix, which is the vector analog to the variance of a random variable.

Definition 2.6. Covariance matrix
The covariance between the $1 \times n$ random vector $\vec{v} = \begin{bmatrix} v_1, \ldots, v_n \end{bmatrix}^T$ and the $1 \times m$ random vector $\vec{w} = \begin{bmatrix} w_1, \ldots, w_m \end{bmatrix}^T$ is the $n \times m$ matrix

$$\operatorname{cov}(\vec{v}, \vec{w}) = \begin{bmatrix} \operatorname{cov}(v_1, w_1) & \operatorname{cov}(v_1, w_2) & \cdots & \operatorname{cov}(v_1, w_m) \\ \operatorname{cov}(v_2, w_1) & \operatorname{cov}(v_2, w_2) & \cdots & \operatorname{cov}(v_2, w_m) \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{cov}(v_n, w_1) & \operatorname{cov}(v_n, w_2) & \cdots & \operatorname{cov}(v_n, w_m) \end{bmatrix}.$$

We also note here that

$$\operatorname{cov}(\vec{w}, \vec{v}) = \operatorname{cov}(\vec{v}, \vec{w})^T.$$

The covariance of the random vector $\vec{v}$ with it self is the $n \times n$ matrix

$$\begin{aligned} \operatorname{cov}(\vec{v}, \vec{v}) = \operatorname{cov}(\vec{v}, \vec{v}) &= \begin{bmatrix} \operatorname{cov}(v_1, v_1) & \operatorname{cov}(v_1, v_2) & \cdots & \operatorname{cov}(v_1, v_m) \\ \operatorname{cov}(v_2, v_1) & \operatorname{cov}(v_2, v_2) & \cdots & \operatorname{cov}(v_2, v_m) \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{cov}(v_n, v_1) & \operatorname{cov}(v_n, v_2) & \cdots & \operatorname{cov}(v_n, v_m) \end{bmatrix} \\ &= \begin{bmatrix} \operatorname{var}(v_1) & \operatorname{cov}(v_1, v_2) & \cdots & \operatorname{cov}(v_1, v_m) \\ \operatorname{cov}(v_2, v_1) & \operatorname{var}(v_2) & \cdots & \operatorname{cov}(v_2, v_m) \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{cov}(v_n, v_1) & \operatorname{cov}(v_n, v_2) & \cdots & \operatorname{var}(v_n) \end{bmatrix}. \end{aligned}$$

For a Gaussian random vector $V = (v_1, \ldots, v_n)$, the density is defined as

$$f_V(\vec{u}) = \det(2\pi\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(\vec{u} - \mu)^T \Sigma^{-1} (\vec{u} - \mu)\right) [6],$$

where Σ is the covariance matrix of V and $\exp(x)$ denotes the function e^x.

2.3 Stochastic Control Systems

This section introduces the concept of stochastic control systems. Many readers are probably familiar with the linear state space model. This model does not necessarily model noise in a system. What follows is a quick introduction of general control systems, and then a discussion of stochastic control systems and an example of a particular stochastic control system model.

A system is a broad term which usually refers to some interaction of objects in the real world. Some examples are a vehicle driving down the highway, or the population of a town plagued by disease, or a robot exploring an unknown building. A model is a mathematical description of the system. The model is used to understand the behavior of a system over time. By having a mathematical description of the system, we can use already-developed mathematics to do things such as estimate the state of the system, given noisy inputs and measurements.

Three vectors are used to model certain aspects of the system. These vectors contain quantities that correspond to these aspects. The "state" vector of the system, $\vec{x}(k)$, describes what is happening in the system at time k using one or more numerical quantities. This vector describes what is happening in this system in reality. Some examples could be a vector containing the vital signs of a human, such as heart rate, white/red blood cell count, number of breaths per minute etc... or the current position (latitude and longitude perhaps) of a boat at sea, or the distance traveled and velocity of a car driving down the highway. The system input, $\vec{u}(k)$ contains numerical quantities that represent inputs to the system. Examples of system inputs would be the acceleration provided by a driver's foot on the accelerator pedal, or the flow rate of insulin provided by an insulin pump. A control system model is described using the state, input, and measurement vectors, and two equations, the state equation and the measurement equation. The state equation is

$$\vec{x}(k+1) = f(\vec{x}(k), \vec{u}(k)) \tag{2.1}$$

The function f depends on the dynamics of the system and describes how the system moves forward in time. Given the current time k, system state, $\vec{x}(k)$, and current inputs, $\vec{u}(k)$, f outputs the next system state $\vec{x}(k+1)$. The measurement equation is

To put things in context, for the example of a car driving down the highway, the state vector could be something like

$$\vec{x}(k) = \begin{bmatrix} d(k) \\ v(k) \end{bmatrix}$$

where $d(k)$ is the car's position at time k and $v(k)$ is the car's velocity at time k. Similarly, we have the control vector

$$\vec{u}(k) = \begin{bmatrix} a(k) \end{bmatrix}$$

where $a(k)$ is the acceleration due to the driver's pressure on the accelerator pedal at time k. The variable k is a non-negative integer that represents time. This is a discrete system, so time moves forward in "steps" of Δt. For simplicity, we always assume that $\Delta t = 1$.

The vector $\vec{y}(k)$ is the measurement, or observation of the system, and is a p dimensional vector:

$$\vec{y}(k) = h(\vec{x}(k)). \tag{2.2}$$

The function h determines the measurement's relationship to the state. This relationship is dependent on what the state and measurement actually are; for instance if the state is the position of a vehicle but the measurement is the velocity of the vehicle, the state must be differentiated with respect to time to yield proper measurement data. Again, using the example of the car on the highway, the measurement equation might be something like

$$\vec{y}(k) = \begin{bmatrix} s(k) \end{bmatrix}$$

where $s(k)$ is the speedometer reading at time k. To complete the model for this system, we must define f and h. First, note that $d(k+1) = d(k) + v(k)$ and $v(k+1) = v(k) + a(k)$, so we may define f as

$$f(\vec{x}(k), \vec{u}(k)) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \vec{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \vec{u}(k). \tag{2.3}$$

Since $\vec{y}(k)$ is a direct measurement of the vehicle's velocity, we define h as

$$h(\vec{x}(k)) = \begin{bmatrix} 0 & 1 \end{bmatrix} \vec{x}(k). \tag{2.4}$$

Plugging these function definitions in to equations (2.1) and (2.2), we get the following system model:

$$\begin{bmatrix} d(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} [a(k)]$$

$$[s(k)] = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} d(k) \\ v(k) \end{bmatrix}.$$

When a system can be described using f and h functions of the form

$$f(k, \vec{x}, \vec{u}) = F(k)\vec{x}(k) + G(k)\vec{u}(k) \tag{2.5}$$

$$h(k, \vec{x}) = H(k)\vec{x}(k), \tag{2.6}$$

where $F(k)$, $G(k)$ and $H(k)$ are matrices, the system is called linear.

This model is deterministic since each state $\vec{x}(k+1)$ can be computed using equation (2.1) as long as $\vec{x}(k), \vec{u}(k)$, are known for all time before $k+1$. If there is some uncertainty in the $\vec{x}(k)$ and $\vec{u}(k)$ vectors, then f cannot completely determine the next state $\vec{x}(k+1)$. This is commonly the case when applying such theory to control systems. For example, in the case of the car, we must have some way of determing how much accceleration $\vec{u}(k)$, in m/s^2, was applied to the car. Nobody can know the actual acceleration that was applied to the car, so we either have to estimate the acceleration using the senses of the human, or use some kind of measurement device, such as an accelerometer. Either of these methods has some uncertainty associated with them. Thus, if we use such measurements or estimations of $\vec{u}(k)$ as an input to f, $\vec{x}(k+1)$ would be slightly off of its actual value.

The measurement vector is built from an observation of the system. Unlike the input vector, the measurement vector does not affect the computation of $\vec{x}(k+1)$. However, since we will use the measurement to estimate $\vec{x}(k)$, the model we use must take the uncertainty of the measurement into account. These two uncertainties can be incorporated into the system by adjusting equations (2.1) and (2.2). The following model has been adjusted to take these uncertainties into account.

The state and measurement equations become

$$\vec{x}(k+1) = f(k, \vec{x}, \vec{u}) + \vec{v}(k) \tag{2.7}$$

$$\vec{y}(k) = h(k, \vec{x}) + \vec{w}(k). \tag{2.8}$$

$\vec{v}$ and $\vec{w}$ are random vectors that express any unpredictable randomness in the system. Specifically, the uncertainty in the measuring device used in the system is encoded by $\vec{w}$, and any uncertainty in how the system evolves in time is encoded in the random vector $\vec{v}$.

These uncertainties are specified using the covariance matrices of $\vec{v}$ and $\vec{w}$. Again, we will return to the car example. The covariance matrix for $\vec{v}(k)$ might be

$$V(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ and}$$

$$W(k) = \begin{bmatrix} 1 \end{bmatrix}.$$

The purpose of $V(k)$ is to describe the uncertainty involved in the state equation. For the car example there is error from the car moving along the road. The car might hit a pot hole or there could be very high error if the driver is estimating the control vector. This error will affect both the position and the velocity of the car at time $k+1$. By defining the matrix with the $\text{cov}\left((,v)_1, v_2\right) = 0$, we are telling the model that the error in the position and the error in the velocity are not related. As long as some values are chosen, the model will contain some notion of the uncertainty in the state equation, it is not extremely important for these values to be accurate.

The purpose of $W(k)$ is to encode any uncertainty involved in taking a measurement. In this case, the measurement only has one component, so this is just the variance of $w_1(k)$, the first component of $\vec{w}(k)$. Again, the value of 1 just tells the model that there is some uncertainty, whereas the actual value of the variance could be different.

Similarly to a linear deterministic control system, a linear stochastic control system is described by the following state and measurement equation. This model is the one that will be used in SLAM.

$$f(k, \vec{x}, \vec{u}) = F(k)\vec{x}(k) + G(k)\vec{u}(k) + \vec{v}(k) \tag{2.9}$$

$$h(k, \vec{x}) = H(k)\vec{x}(k) + \vec{w}(k). \tag{2.10}$$

3 Our Robot

3.1 Robot Configuration

An iRobot Create acts as the base of the robot. The Create is equipped with two differential drive, DC motors and shaft encoders that provide information regarding the motion of the robot. In addition to the Create's shaft encoders, the system uses a Hokuyo laser rangefinder and a USB web-camera as its sensors. The system is controlled using a netbook running LabVIEW 2009. To make space for the various sensors and the netbook, a mechanical platform was placed on the iRobot Create. The platform is made out of four pieces of angled iron. The angled iron pieces act as risers that are bolted to the base of the Create. Two Plexiglas sheets were fitted between the risers, providing space for the netbook. The remainder of this section is dedicated to explaining the various components of the design.

3.1.1 Sony VAIO Netbook With LabVIEW

The robot's software is run on a Sony VAIO netbook. As illustrated in Figure 2, it is the central unit of the robot and all sensors and the iRobot Create interface with it. The Netbook is a compact PC which is ideal in size for the purpose of this project. The Netbook interfaces with the Create using serial communication which is discussed in detail in later sections. The webcam and the Hokuyo Laser Rangefinder are connected to the netbook using USB ports. The netbook is equipped with National Instrument's LabVIEW Robotics 2009. LabVIEW is a visual programming environment and platform that can be used for data acquisition, instrument control and automation. LabVIEW is particularly useful in performing the various computations and autonomous control that this project requires and was also the primary choice given the level of experience the members of the team have had with working in the LabVIEW environment. LabVIEW allows users to create programming blocks called Virtual Instruments (VI). Built-in VIs can be used to run programming loops, perform mathematical and boolean processes, communication and data manipulation amoung

Figure 2: Diagram of all components of the robot and how they are connected.

other tasks and allows users to make their own VIs. Using these built-in VIs, customized VIs were created to build the system architecture and all other system software including the SLAM algorithm, Kalman filter and navigation algorithm.

3.1.2 iRobot Create

As mentioned previously, the base of the robot is an iRobot Create. It is a robotics development platform based on the iRobot Roomba vacuum cleaner. The Create is modified to contain a cargo bay with a 25 pin port for digital and analog input and output. It is additionally equipped with a serial port that can be used to communicate with the Create. The netbook uses RS 232 protocol to communicate with the Create Open Interface (OI). The Create's serial port is configured to receive 8 bit opcodes at 57600 baud, the default rate. The serial port is then used to send initialization code, actuation commands and receive distance and angle information from the shaft encoders. These tasks are explained below and are done through specific VIs programmed in LabVIEW.

1. Initialization: Sending an op code value of 128_{10} starts the Create OI. The next step is to choose the operating mode. The OI can be operated in four modes: Off, Passive, Safe and Full. Sending an opcode value of 131_{10} puts the OI in safe mode enabling standard safety features, meaning that if it is next to the edge of a table and the sensors detect the edge of the table, the robot will shut down.

2. Actuator Control: Sending an op code of value 145_{10} allows the netbook to control the Create's forward and backward motion. The op code is followed by four data bytes that set the wheel speeds in millimeters per second. The first two bytes specify the right wheel speed

Figure 3: Picture showing the front of the robot.

while the next bytes specify the left wheel speed. A positive velocity makes a wheel drive forward while a negative velocity makes the wheel spin backward.

3. Encoder Information: In addition to sending motor commands to the Create, the RS 232 port can be used to obtain distance and angle information. This information is used in the SLAM process. Sending an op code of 142_{10} requests the OI to send a packet of sensor data bytes. This op code is followed by the Packet ID for the packet that needs to be retrieved. Packet 19 returns the distance the Create has travelled in millimeters since the distance was last requested. Positive values reflect forward motion while negative values indicate backward motion. Similarly, packet 20 contains the angle in degrees the Create turned since the last time this information was requested. Counter-clockwise angles are positive while clockwise angle values are negative. The values are stored in a counter, so care must be taken to read them frequently enough so that they do not roll over to 0 and start counting over [1].

3.2 Sensor Setup

A Hokuyo laser rangefinder coupled with a standard Logitech USB webcam are used for the purpose of landmark extraction. Recall that landmark extraction is an integral part of SLAM and refers to the process of extracting various features in the environment that can be recognized by the robot repeatedly to help the mapping and localization processes. A careful study of existing literature shows that there are multiple different approaches that have been taken to landmark extraction. These include the use of a camera rig to obtain panoramic images [2], the use of laser cameras and the technique of segment matching [3], and the use of a single camera to obtain quadrangular images[4]. After conducting a survey of the various landmark extraction techniques, it was realized that most techniques were suited to specific environments and sensor setups. The team then decided to specify the environment that would be used to develop the SLAM algorithm and make sensor choices based on it. The landmarks used would be cylinders of various colors. The final setup contains red, blue, and green landmarks. For these specific landmarks, using a laser rangefinder and

a webcam is simple and appropriate. The webcam is used to detect and distinguish between objects in the environment using their color, while the laser rangefinder is used to get distance information for each detected landmark. Once the team decided on using a laser rangefinder and a camera, a survey was performed to decide on the best possible combination of specific sensors to be purchased. Table 1 below highlights the different options that were discussed. Taking into consideration the

Sensor	**Positives**	**Negatives**
Laser Rangefinder	Accurate and precise data on distance of objects from robot	Complicated algorithms to differentiate between environment and landmarks
Stereo-vision Camera	Able to detect landmarks and get distance information	Not easy to implement - would have to develop a driver
Camera	Easy to detect landmarks by color	More difficult to get distance information

Table 1: Original sensor possibilities

budget allotted to the team and the quality of the scan obtained, the team decided upon a Hokuyo URG-04LX-UG01 laser rangefinder. It has a sensing range of 5.6 meters and the measurement accuracy is known to be within three percent tolerance of the current reading. Additionally the scanning rate is 100 ms across a 240° field of view, which is sufficient for the purpose of this project. Figure 4 shows a single measurement made by the laser rangefinder. Although only one

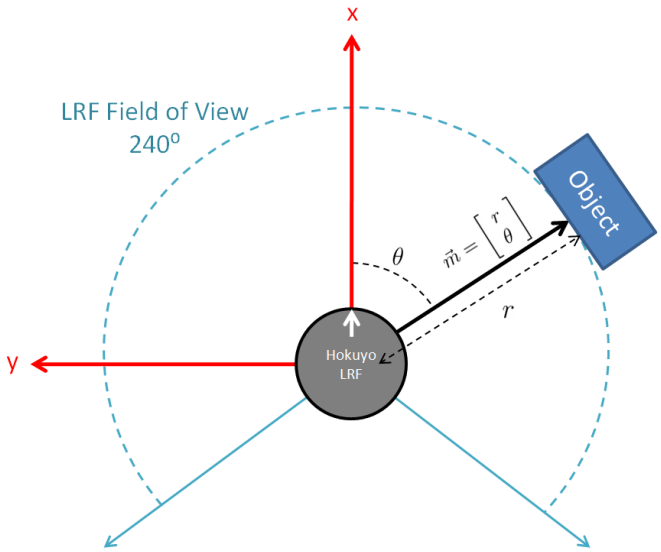

Figure 4: Diagram of the laser rangefinder showing one measurement in detail. The solid black arrow is the (range, bearing) measurement and the blue box is the object being measured.

measurement is shown here, the laser rangefinder makes over 700 of these measurements in each scan, evenly distributed over the 240° field of view. The black arrow is a vector representing the measurement. The distance of the measurement, r, is read in mm, and the angle is read as the angle of the vector $\vec{m}$ from the x axis, ranging from $-120°$ to $120°$. A publicly available LabVIEW VI performs the serial communication with the Hokuyo laser rangefinder in order to extract this data. The data returned is in the form of two arrays: one array contains all the distance measurements,

and the other array contains all of the angles corresponding to each distance.

The Logitech USB camera was chosen for its easy plug and play feature and ease of image extraction through LabVIEW. The image data is returned from the camera as a special LabVIEW type, and built in image processing functions are used to make requried computations.

4 Simultaneous Localization And Mapping (SLAM)

4.1 Localization and Mapping

Before getting into SLAM, localization and mapping must be defined. **Mapping** is a robotics term that refers to constructing a list of objects and their locations in the environment relative to a reference frame [6, p152]. In SLAM, these objects are called landmarks. **Localization** is determining the robot's position in a reference coordinate frame when already given a map. The robot must use its sensors to detect features and to associate them with features from the map [6, p5].

As its name suggests, SLAM does both of these tasks simultaneously. Before SLAM, it was thought that doing these tasks separately would yield better results as localization and mapping were considered to be mathematically unrelated. It has since been found that the opposite is actually true, these two tasks are related, and SLAM takes advantage of this relationship [7].

4.2 Odometer Based Mapping

Before discussing SLAM, we will first introduce a technique for mapping that has a substantial issue due to cumulative error, called odometer based mapping. It is assumed that

1. The robot can make measurements of landmarks and distinguish one landmark from another and

2. The robot can compute its change in position after moving using an on-board odometer.

Figure 5 shows a diagram of this process. The robot at time k takes a measurement of the relative distance to landamrks 1 and 3. From these measurements, the robot can construct a map with the coordinates of two of the landmarks. Then, the robot moves from the location at time k to the one

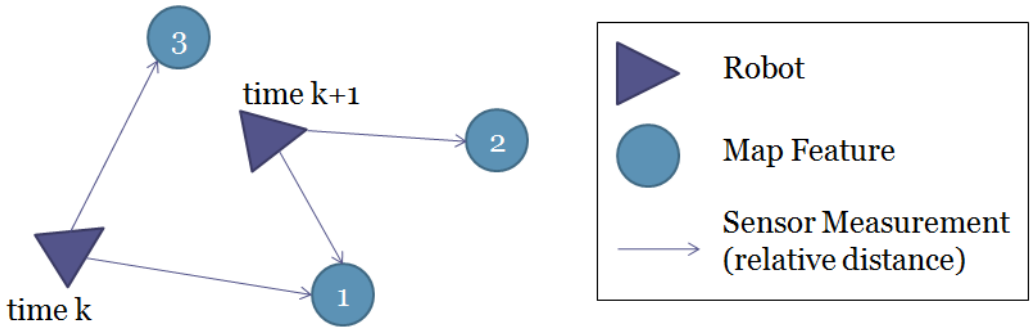

Figure 5: Diagram showing the steps in odometer based mapping.

at time $k+1$ and computes the change in position. However, there is error in the computed change in position, so the estimated position of the robot at time $k+1$ also has some error associated

with it. When the robot gets to the location shown at time $k+1$, the robot takes a measurement of landmark 2, in addition to the already observed landmark 1. The position at time k and the change in position can be used to compute the positions of the two landmarks with respect to the reference coordinate frame. Now the robot has two different position measurements of landmark

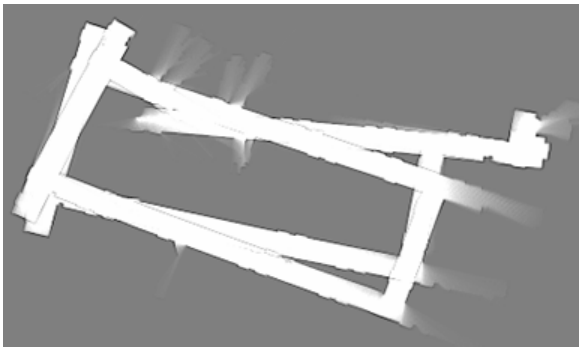

Figure 6: Shows a map of a rectangular building constructed using odometer based mapping.

1 with respect to the reference coordinate frame. The second measurement is polluted by error from the robot's computation of its change in position. In an attempt to intelligently estimate the position of landmark 1, the robot averages the two measurements of the landmark's positions. This noise from the change in position builds up over time, causing the estimated position of landmarks to be skewed. This effect is demonstrated in Figure 6. Since the map in Figure 6 was generated using odometer based mapping, the cumulative error associated with measuring change in position caused the map to become bent.

4.3 The Solution: Kalman Filter SLAM

SLAM solves this problem by statistically combining the robot position data and the landmark measurement data. SLAM has two main steps:

1. Landmark Extraction, and
2. Data Fusion.

Landmark extraction is the process of using sensors to measure the locations of landmarks in the environment. Part of this process, called data association, is is the processes of recognizing the same landmark from different locations. Various techniques for landmark extraction are discussed in [8] [9], and the specific algorithms used for this project will be discussed in detail.

The data fusion step requires a statistical approach. Different applications of SLAM use different statistical techniques. The most commonly used technique (and the one used in this project) is called the Kalman filter. The Kalman filter assumes that the map and robot pose (position and heading) can be represented using one system state in a state-space model.

The same assumptions are made as the previous section on odometer based mapping. They are re-stated here:

1. The robot can make measurements of landmarks and distinguish one landmark from another and
2. The robot can compute its change in position after moving using an on-board odometer.

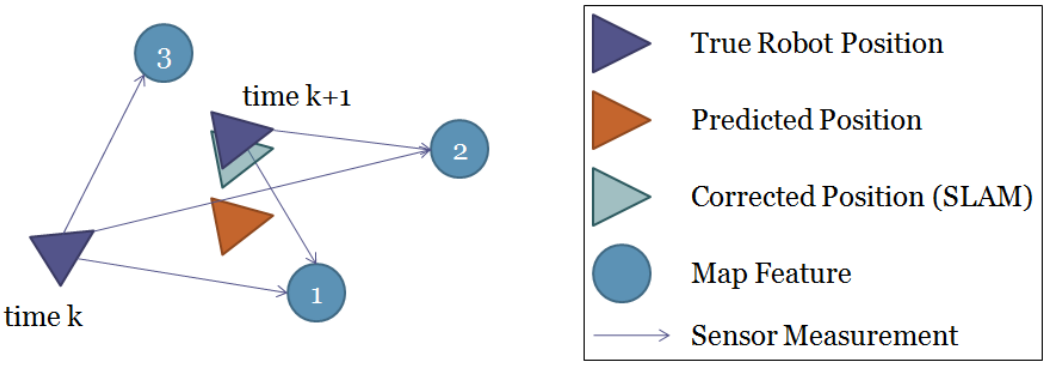

Figure 7: Diagram of the predict-update process.

Kalman filter data fusion has two steps. The first step is to **predict** the robot position using odometer data. Figure 7 shows the predict-update structure of Kalman filter based SLAM. The robot position at time $k+1$ is predicted using odometer data. However, there will be some error as discussed in section 4.2, so instead of the robot predicting it will be at the true position, shown by the purple triangle, it first predicts its next location, shown by the orange triangle, based on system controls. The robot then uses the Kalman filter to incorporate sensor measurements at time $k+1$. This step is called the **update** step. In addition to using these new measurements of landmarks 1 and 2 to compute a better estimate of their positions, the Kalman filter also uses this information to update the predicted robot position. Thus, the predicted position is corrected to a more accurate estimate of its position, shown by the light blue triangle. In this way, SLAM combats the cumulative error in the odometer measurements.

5 Statistical Estimation: Kalman Filter

5.1 Kalman Filter Overview

In order to use the Kalman filter, the system must be described using a stochastic linear state-space model. The model used has some error terms in the state and measurement equations to model the cumulative error in the odometer and the uncertainty in the sensor measurements. Each of the vectors in this state-space model are considered to be random vectors so that uncertainties about each vector may be defined. The consideration of uncertainty will be most important for the noise terms in the system. The state equation is

$$\vec{x}(k+1) = F(k)\vec{x}(k) + G(k)\vec{u}(k) + \vec{v}(k),$$

where $\vec{x}(k)$ is the state vector at time k, $\vec{u}(k)$ is the input vector at time k, $\vec{v}(k)$ is the process noise at time k, $F(k)$ describes the system dynamics at time k, and $G(k)$ describes how the input vector affects the transition to the next state. The process noise is assumed to be white and Gaussian with mean 0 and covariance matrix $V(k)$. Whiteness means that the process noise at any time j is not correlated with the process noise at any other time i.

The measurement equation is

$$\vec{y}(k) = H(k)\vec{x}(k) + \vec{w}(k),$$

where $\vec{y}(k)$ is the measurement at time k, $\vec{w}(k)$ is the measurement noise at time k, and $H(k)$ describes the relationship between the system state at time k and the measurement at time k. Here,

$\vec{w}$ describes the uncertainty in the measurement and is also assumed to be white and Gaussian with mean 0 and covariance matrix $W(k)$. H is assumed to be full row rank for any k.

All of these assumptions may become valid or invalid depending on the specific application. The Kalman filter is commonly re-derived with different assumptions depending on the application [5]. For SLAM, the assumption of linearity is usually too restrictive. The most common form of the Kalman filter used for SLAM is called the Extended Kalman Filter (EKF). The EKF works by linearizing the system and then applying the linear Kalman filter.

The estimate of the system state at time k is called $\hat{x}(k)$. In addition to the state estimate, the filter also provides an estimate of the error covariance matrix, called $P(k)$. This matrix is the covariance of the error vector $\vec{x}(k) - \hat{x}(k)$. It can be thought of as a measure of how good or bad the estimate is at time k. Given the assumptions discussed above, the Kalman filter provides recursive equations for computing $\hat{x}(k+1)$ and $P(k+1)$, given

1. The previous estimate of the system state, $\hat{x}(k)$,
2. The previous estimate of the error covariance, $P(k)$,
3. The system input at time k, $\vec{u}(k)$, and
4. The system measurement at time $k+1$, $\vec{y}(k+1)$.

A flowchart for the algorithm is shown in Figure 8. The equations are given in two steps: prediction

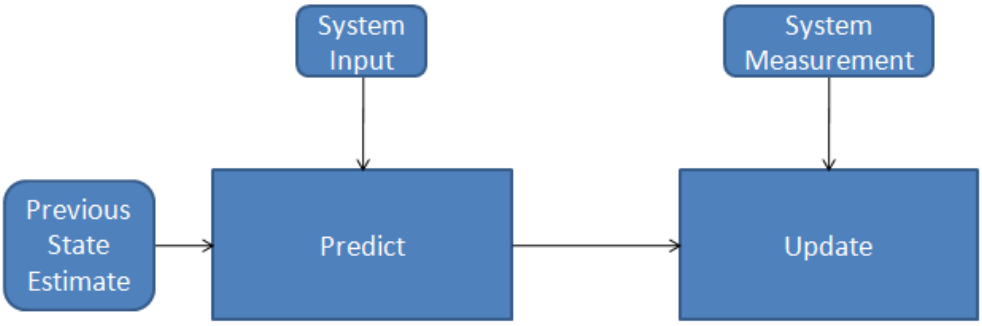

Figure 8: Flowchart showing the steps in the Kalman filter algorithm.

and update. The prediction equations are

$$\hat{x}(k+1|k) = F(k)\hat{x}(k) + G(k)u(k)$$
$$P(k+1|k) = F(k)P(k)F(k)^T + V(k)$$

This is an intermediate prediction of the system state based only on the input, $\vec{u}(k)$. The following equations are the update equations

$$\hat{x}(k+1) = \hat{x}(k+1|k) + Rv$$
$$P(k+1|k) = P(k+1|k) - RH(k+1)P(k+1|k),$$

where

$$v = \vec{y}(k+1) - H(k+1)\vec{x}(k+1|k)$$

$$S = H(k+1)P(k+1|k)H(k+1)^T + W(k+1)$$
$$r = P(k+1|k)H(k+1)^T P(k+1|k)S^{-1}\text{[5, p285]}.$$

These equations incorporate the information from the newest measurement $\vec{y}(k+1)$. Once the system is described using the state and measurement equations above, these prediction and update equations will form the core of the SLAM algorithm [5, p273].

5.2 Kalman Filter SLAM

In this section, the system model for SLAM will be constructed. This model is a slightly expanded version of the one presented in [5, p295]. The state vector will be a $(3+2n)\times 1$ vector, where n is the number of landmarks in the system. The vector is defined as

$$\vec{x}(k) = \begin{bmatrix} x_r(k) \\ y_r(k) \\ \theta_r(k) \\ x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix}$$

where $x_r(k)$, $y_r(k)$, and $\theta_r(k)$ are the robot x position, y position, and heading, respectively, with respect to the reference coordinate frame, and x_i and y_i are the positions of the landmarks. Figure

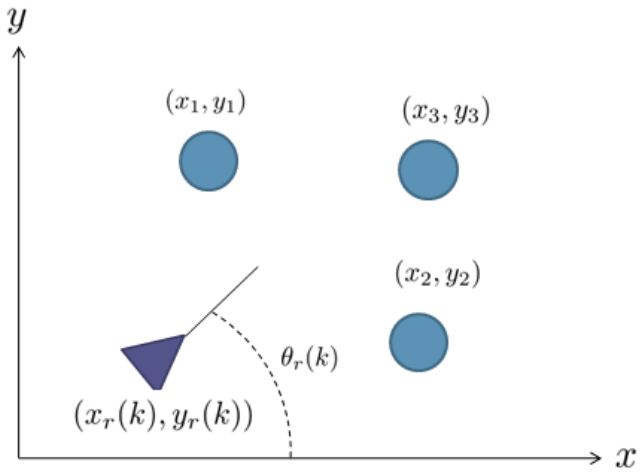

Figure 9: Visualization of the quantities included in the state vector for $n = 3$.

9 visually shows these quantities for a system with 3 landmarks ($n = 3$). The first three elements make up the robot pose and each (x_i, y_i) pair after that is the position of the ith landmark.

The input vector contains the change in robot pose from time k to time $k+1$. This information is read from the odometer. The vector is defined as

$$\vec{u}(k) = \begin{bmatrix} \Delta x_r(k) \\ \Delta y_r(k) \\ \Delta\theta_r(k) \end{bmatrix}$$

where $\Delta x_r(k)$, $\Delta y_r(k)$, and $\Delta\theta_r(k)$ are the change in x position, y position, and robot heading, respectively. Figure 10 shows a visualization of the input vector.

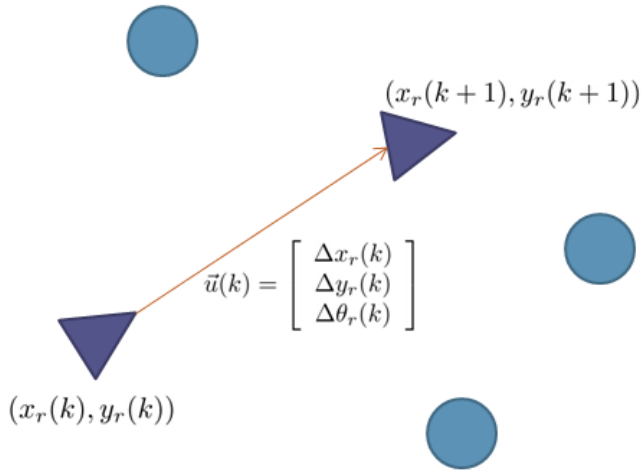

Figure 10: Visualization of the input vector for $n = 3$.

Now $F(k)$ and $G(k)$ will be defined so that the state equation advances the robot position without changing the position of each landmark. $F(k)$ is defined as the $(3+2n) \times (3+2n)$ identity matrix, so that $F\vec{x}(k) = \vec{x}(k)$. $G(k)$ is the $(3+2n) \times 3$ matrix defined by

$$G(k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}.$$

By defining $G(k)$ in this way, the new state vector is computed as the old state vector plus the change in robot pose. So the robot is advanced to its next position as desired, and the landmarks are not affected because of the $2n$ rows of zeros at the end of the input vector.

The process noise $\vec{v}(k)$ is never known directly, since it is in the state equation for the purpose of modeling noise associated with the system advancing from time k to time $k+1$. However, in order to give the Kalman filter some information about this noise, the covariance matrix of $\vec{v}(k)$, called $V(k)$ will be given as an initial condition. The covariance of $\vec{v}$ is a $(3n+2) \times (3n+2)$ matrix defined by

$$V(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

The 1's in the first three diagonals describe the amount of uncertainty in the first three elements in the state vector due to any noise in the state equation. One example of a source of noise is wheel slippage. If the wheels of the robot slip, the odometer reading will be erroneous. The rest of the

matrix is filled with 0's because there is no uncertainty in the transition of landmark position from time k to $k+1$. The landmarks are assumed to be stationary objects, and thus there is no error associated with their location.

Next, we move on to the measurement vector. It is defined as

$$\vec{y}(k) = \begin{bmatrix} x_1 - x_r(k) \\ y_1 - y_r(k) \\ x_2 - x_r(k) \\ y_2 - y_r(k) \\ \vdots \\ x_n - x_r(k) \\ y_n - y_r(k) \end{bmatrix}$$

since the measurement in SLAM is a measurement of the position of each landmark that is in view relative to the robot's position. Figure 11 shows the quantities included in the measurement vector visually. The matrix H relates the measurement vector to the state vector. First, we define H as

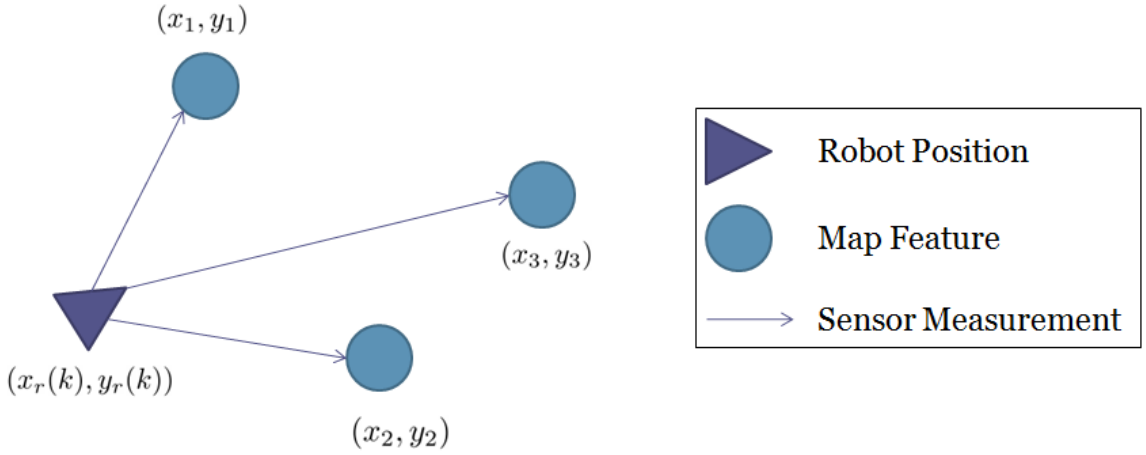

Figure 11: Visualization of the measurement vector for $n = 3$.

a block matrix as follows

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{bmatrix}.$$

Each H_i relates the ith landmark measurement to the system state. So H_i is defined by

$$H_i = \begin{bmatrix} -1 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 & 1 & \cdots & 0 \end{bmatrix}.$$

As an example, let the number of landmarks, n, be 3. Assume that the robot has just used its sensors to detect all landmarks that are currently visible, and it has detected landmark 1 and landmark 3. Then, $\vec{y}(k)$ would be

$$\vec{y}(k) = \begin{bmatrix} x_1 - x_r(k) \\ y_1 - y_r(k) \\ x_3 - x_r(k) \\ y_3 - y_r(k) \end{bmatrix}.$$

So, we want to build H using H_1 and H_3. H_1 and H_3 would be

$$H_1 = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_3 = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and the measurement equation would be

$$\vec{y}(k) = H\vec{x}(k) + \vec{w}(k)$$

$$\begin{bmatrix} x_1 - x_r(k) \\ y_1 - y_r(k) \\ x_3 - x_r(k) \\ y_3 - y_r(k) \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r(k) \\ y_r(k) \\ \theta_r(k) \\ x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} + \vec{w}(k)$$

$$= \begin{bmatrix} x_1 - x_r(k) \\ y_1 - y_r(k) \\ x_3 - x_r(k) \\ y_3 - y_r(k) \end{bmatrix} + \vec{w}(k)$$

Since the expected value of the noise term $\vec{w}(k)$ is 0, the left and right sides match up properly.

The covariance matrix for $\vec{w}(k)$, called $W(k)$, is the $2n \times 2n$ identity matrix

$$W(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Defining $W(k)$ this way tells the Kalman filter that each element of the measurement vector has some uncertainty associated with it.

6 Experiments

The following sections describe in detail the experimental setup, software, and results for each experiment performed as part of this project. Each experiment revolves around using the Kalman filter to estimate the system state for some control system.

6.1 Straight Line SLAM

6.1.1 Experiment Description

In the straight line experiment, the red landmark remains stationary while the robot moves towards it in a straight line. The robot moves using dead reckoning, so the Create's wheels are sent constant

speed values. The state vector now has the robot's x, y coordinates and heading ($\theta(k)$) and the single landmark's x and y coordinates. The vector then takes the following form:

$$\vec{x}(k) = \begin{bmatrix} x_r(k) \\ y_r(k) \\ \theta_r(k) \\ x_1 \\ y_1 \end{bmatrix}$$

Since the system does not have any additional variables, $F(k)$ is the 5×5 identity matrix. The input vector, $\vec{u}(k)$, reflects the change that is caused by the forward motion of robot in the x and y directions and its heading. Hence $\vec{u}(k)$ is as follows:

$$\vec{u}(k) = \begin{bmatrix} \Delta x_r(k) \\ \Delta y_r(k) \\ \Delta \theta_r(k) \end{bmatrix}$$

$G(k)$, which defines the effect of the input vector on the control vector is as follows:

$$G(k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This form ensures that the robot's x, y and θ values are updated by the input vector without changing the one landmark's position.

The measurement vector $\vec{y}(k)$ corresponds to a displacement vector between the robot and the landmark, so it has x and y components. The matrix H is defined for the case where $n = 1$ as in section 5.2. While the noise terms cannot be modeled directly, the covariance matrices for each one must be provided. The process noise covariance, or the covariance of $\vec{v}(k)$ is

$$V(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

This says that there is uncertainty in the robot's change in position, but not in the actual position of the one landmark.The measurement noise covariance, or the covariance of $\vec{w}(k)$ is

$$W(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This covariance matrix encodes the uncertainty involved in measuring the location of the landmark.

6.1.2 Input Vector

Since the input vector is the change in x, y, and θ with respect to the reference coordinate frame, some computation must be done on the data received from the Create to determine $\vec{u}(k)$. The Create's shaft encoders measure two quantities: average distance traveled by both wheels since the

distance register was last polled, and the change in angle in the robot's heading since the heading register was last polled.

Figure 12 shows the quantities involved in this computation. The blue circles with arrows show the robot before (lower-right) and after (upper-left) making a turn. The arrows inscribed in the blue circles indicate the robot's heading. The Create returns $S(k)$, the arc length of this turn, and the change in heading. Using some simple geometry, it is clear that the change in heading is equal to $\theta(k)$. Before computing $\Delta x_r(k)$, $\Delta y_r(k)$, and $\Delta\theta_r(k)$, the change in position with respect to

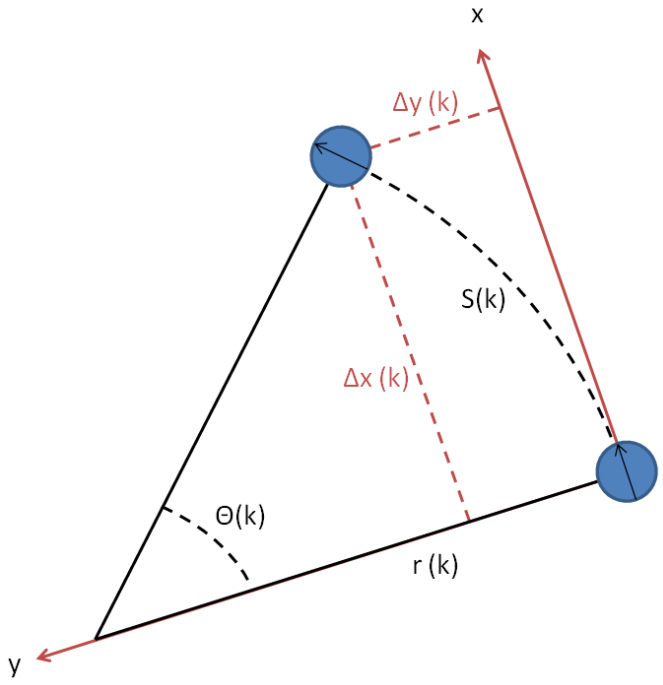

Figure 12: Diagram showing the quantities involved in computing the input vector. $\theta(k)$ and $S(k)$ are measured directly by the Create's shaft encoders.

the (x, y) coordinate frame in Figure 12, denoted $(\Delta x, \Delta y)$, must be computed. First, the turning radius is

$$r(k) = \frac{S(k)}{\theta(k)}.$$

Using this radius and simple trigonometry,

$$\Delta x(k) = r(k)\sin(\theta(k)).$$

Then, using similar logic,

$$\begin{aligned} r(k) - \Delta y(k) &= r(k)\cos(\theta(k) \\ r(k) - r(k)\cos(\theta(k)) &= \Delta y(k) \\ r(k)(1 - \cos(\theta(k)) &= \Delta y(k). \end{aligned}$$

So, to summarize,

$$\begin{bmatrix} \Delta x(k) \\ \Delta y(k) \end{bmatrix} = \begin{bmatrix} \Delta r(k)\sin(\theta(k)) \\ \Delta r(k)(1 - \cos(\theta(k)) \end{bmatrix},$$

where $r(k) = S(k)/\theta(k)$.

To compute $(\Delta x_r(k), \Delta y_r(k))$, $(\Delta x, \Delta y)$ must be converted to the reference coordinate frame. Since this vector is a displacement vector, it suffices to rotate $(\Delta x, \Delta y)$ by $\theta_r(k)$. This is accomplished using a 2×2 rotation matrix of angle $\theta_r(k)$, or

$$\begin{bmatrix} \cos(\theta_r(k)) & -\sin(\theta_r(k)) \\ \sin(\theta_r(k)) & \cos(\theta_r(k)) \end{bmatrix}.$$

So the final control vector is computed by

$$\vec{u}(k) = \begin{bmatrix} \Delta x_r(k) \\ \Delta y_r(k) \\ \Delta \theta_r(k) \end{bmatrix} = \begin{bmatrix} \cos(\theta_r(k)) & -\sin(\theta_r(k)) & 0 \\ \sin(\theta_r(k)) & \cos(\theta_r(k)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \\ \theta(k) \end{bmatrix}.$$

6.1.3 Software Architecture

Overview This experiment is the first one in which all the components of the robot were used. Figure 13 shows the flowchart for the program. The first step in the program is to construct the measurement and update vectors, $\vec{y}(k)$ and $\vec{u}(k)$. $\vec{y}(k)$ is built directly from the landmark extraction code. The vector $\vec{u}(k)$ is read from the iRobot Create as described in section 6.1.2 above. At this point, $\vec{u}(k)$ can be plugged into the Kalman predict step. The predicted state and $\vec{y}(k)$ can then be used in the update step to compute the final prediction for this iteration, $\hat{x}(k)$.

Landmark extraction is performed as in the stationary ship experiment, the Create is driven at a constant speed and the shaft encoders are read as described in section 3.1. The Kalman filter used in the stationary ship experiment was used directly in this experiment as well.

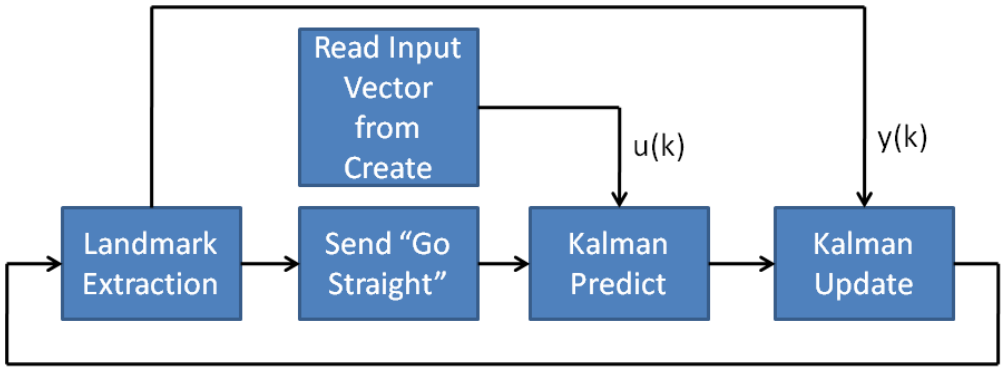

Figure 13: Flowchart showing straight line SLAM software architecture.

Sensor Calibration The webcam and the laser rangefinder were tested and calibrated so that a general relationship between the horizontal location of the landmark in the captured image and its corresponding distance measurement from the laser rangefinder scan was found. The laser rangefinder data is received as an array, and each data point in the array corresponds to a specific angle of the scan. The reading from one specific angle will always be in the exact same location of the array. This information was used to calibrate the webcam and the laser rangefinder. To perform the calibration, objects were put at the very edge of the webcam image and the array location of the distance reading that corresponded to the object was recorded. To sum up, each

data point consisted of the x coordinate in the image (in pixels) and the location in the array of the distance information for that pixel.

(Pixel location, Index of distance data in array)

Originally, many data points at the edge of the image were taken, but it was decided that a good enough approximation could be calculated by simply taking two data points, one at the far left corner of the image and one at the far right corner. Figure 14 illustrates the set-up used to calibrate the webcam and the laser rangefinder. From these two data points a linear relationship between

Figure 14: Diagram showing the calibration of the webcam and laser rangefinder

the horizontal pixel location in the image and the index of the corresponding data in the array was derived. This relationship is shown by Equation 6.1.

$$i_{lrf} = 0.8x_{img} + 280 \tag{6.1}$$

where i_{lrf} is the index of the data in the laser rangefinder array and x_{img} is the horizontal pixel location of the landmark in the image.

Landmark Extraction As previously discussed, to properly perform SLAM a robot must be able find easily identifiable landmarks in its environment and gather information about their position relative to the robot. This is called landmark extraction. Since the focus of this project was to develop a working SLAM algorithm, it was decided that simple, easy to detect landmarks would be used. The webcam can produce an image that can be processed to detect colors fairly easily. Because of this, the landmark used in this experiment was a red cylinder, shown in Figure 15.

The algorithm for detecting the landmark is as follows. The webcam and the laser rangefinder point in the same direction. The webcam takes a picture and the laser rangefinder takes a scan at the same time. The image from the webcam is then sent through a series of image processing operations which detect the red from the landmark and determine its center of mass. The center

Figure 15: Red landmark used for stationary ship and straight line experiments.

of mass, x_{img} is then put into Equation 6.1. This results in an estimate of where in the array the corresponding distance information is located. When this method was tested, there were several problems with it. First of all, the equation did not always yield the exact location of the correct distance information in the scan. Secondly, it was not very robust in that when the robot turned, it would lose its spot in the laser rangefinder scan very quickly. Because of these issues, a final step was added to the process. A 40 element sub-array was taken from the complete laser rangefinder scan array. This sub-array was centered at the result of equation 6.1, i_{lrf}. This made it so that even when the equation did not give the exact location of the distance measurement, the distance measurement was always located in the sub-array. Figure 16 shows how the 40 element sub-array is taken from the complete laser rangefinder scan. Once the 40 element sub-array is extracted from

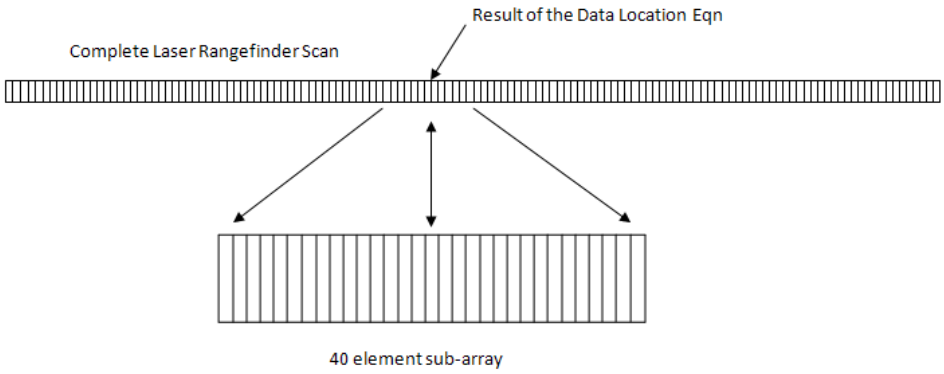

Figure 16: This diagram that shows how sub-array is taken from laser rangefinder scan.

the laser rangefinder scan, it is analyzed element by element looking for a change in distance of over 50 mm. If there is a change in distance of over 50 mm, then there is a landmark present and the distance reading is taken and saved. This distance reading along with the angle at which the measurement was taken are used to calculate the (x, y) position of the landmark relative to the position of the robot as shown in Figure 17. We will call this the $(\hat{x}, \hat{y})$ position for the rest of

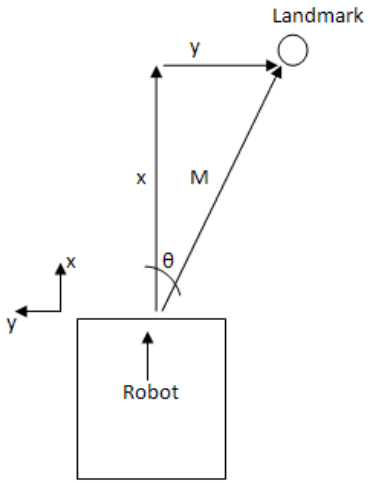

Figure 17: This diagram shows how the (x, y) position a landmark relative to the robot is calculated.

this discussion. Once the $(\hat{x}, \hat{y})$ position is determined, it must be translated so that it is an (x, y) position with regard to the original coordinate system. We will call this (x_i, y_i). In order to do this, the current (x_r, y_r) position of the robot will be taken from the robot state and added to the landmark $(\hat{x}, \hat{y})$. The result of this is the position of the landmark with respect to the original coordinate system, (x_i, y_i). The final result of the landmark extraction algorithm is this (x_i, y_i) for each visible landmark. It is input into the Kalman filter and used in the update section of the Kalman filter.

Kalman Filter The Kalman filter implementation used for this experiment was centered around the `KalmanFilter` data type. This data type contains the current state estimate, $\hat{x}(k)$, the current error covariance estimate, $P(k)$, and the matrices that describe the current system model, $F(k)$, $G(k)$, and $H(k)$. The two main functions are `KalmanPredict` and `KalmanUpdate`. `KalmanPredict` takes the input vector, $\vec{u}(k)$, and executes the prediction equations shown in section 5.1. Similarly, `KalmanUpdate` takes the measurement vector, $\vec{y}(k+1)$, and executes the update equations shown in section 5.1. By doing this, the estimate of the map is updated.

Map Generation The map was generated from the Kalman estimate, $\hat{x}(k)$. At the end of each iteration of the main loop, the robot position, $(x_r(k), y_r(k))$, is saved to an array. This array of (x, y) points is plotted as a line using the LabVIEW waveform graph control. Another array containing the location of the landmark was also passed to this graph control and plotted as solid white dots. A sample map is shown below in figure 18.

6.1.4 Results

Initially, the results of the experiment were not as expected. The covariance matrix did not tend to 0 and the estimate was not reasonable at all. Because of this, the update and predict equations as well as the covariance matrices were examined in greater detail. It was found that the initial covariance matrix was causing the experiment to fail. For the initial trials, the covariance matrix was a 5×5 identity matrix. After some research, it was found that covariance matrices that resulted in successful SLAM were not the identity matrix, rather they had zeros instead of ones in some

places. The process noise covariance matrix used for the successful experiment is as follows:

$$V(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Using this covariance matrix, the experiment yielded results as expected with two constraints. The landmark always had to be in the field of view of the webcam and the laser rangefinder, and the measurement associated with the landmark had to be valid. The second constraint arose due to the fact that the Hokuyo laser rangefinder being used did not always give good measurement readings. When looking at a complete scan, there would be measurements of 0 throughout the scan when there should not have been any. This occurred most often when there were no objects within its range. This would sometimes cause the measurement of the landmark to be 0, which was incorrect. This problem was corrected in later experiments.

The robot was able to detect the landmark and get its (x, y) position, as in the stationary ship experiment. When the robot moved, it was able to keep track of the landmark through the iterations of the program. Also, the predict and update steps of the Kalman filter worked properly, resulting in the robot updating its estimate of its position accurately. All together, this resulted in the robot being able to move towards the landmark while updating its estimate of its position and the position of the landmark relative to a set coordinate system. Figure 18 shows the map

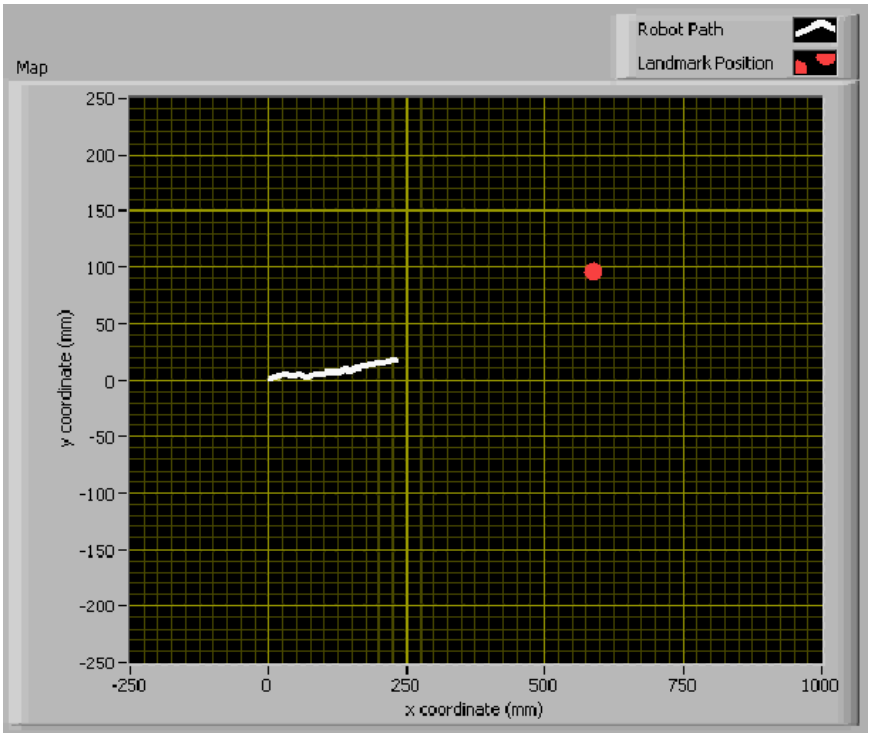

Figure 18: Map output of straight line experiment

generated from the straight line experiment. The red dot is the estimated position of the landmark. The landmark was not directly in front of the robot, as shown by its position. The white line shows the estimate of the path of the robot. The line is not straight because the noise in the landmark

measurements changed the estimate of the path slightly. Despite this, the estimate is still very good.

6.2 Multi Landmark SLAM

6.2.1 Experiment Description

The multiple landmark experiment incorporates all three landmarks into the system. Because of this, the various components of the Kalman filter system needed to be updated to include all of the landmarks. The robot itself still used dead reckoning to move in a straight line the same way it did in the straight line experiment.

6.2.2 Software Architecture

Overview This experiment will be extremely similar to straight-line SLAM. Landmark extraction was modified to include blue and green in addition to red. A "Landmark Management" component was added to the software to handle cases in which landmarks were not visible. The first version of this code relied on a separate landmark database to store landmarks not observed in the current measurement instead of just using the system state, $\vec{x}(k)$. This method required a lot of complex matrix manipulation, and thus was fraught with bugs. Later on it was discovered that by simply manipulating the H matrix the same affect could be achieved with little computational overhead.

Landmark Extraction For multiple landmark SLAM, the landmark extraction algorithm had to be updated to detect multiple landmarks. The basic algorithm, however, remained the same. Instead of having one red landmark, there were three landmarks of different colors, one red, one green, and one blue, shown in Figure 19.

Figure 19: The 3 different landmarks with different colors

Separate sub-vi's were written to detect the green landmark and then the blue landmark in the image from the webcam using the same process used to detect the red landmark. Then, when the image is taken and the laser rangefinder makes its scan, the image is processed three times, once looking for the red landmark, once looking for the blue landmark, and once looking for the green landmark. Each landmark detected is assigned an ID according to its color, and its center of mass

is used to find the distance measurement from the laser rangefinder scan using the same process previously described. The assigning of the ID is very important because it is used to identify the same landmark for each iteration of the code. It is important that the red landmark is always assigned the same ID so that the program will not associate the distance measurement from the red landmark with the ID of the blue landmark. This is called data association. If it is not done properly, it will cause the program to think that the location of a landmark suddenly jumped to a different place, which can cause issues in the update step of the Kalman filter and result in errors in the state estimate.

Landmark Management After the landmark extraction step, a list of N observed landmarks and their ID's are passed along to the Kalman filter step. Before performing the update, the landmarks must be compiled into the measurement vector. For this scheme to work, there must be a correspondence between the indexes in the state vector with the current landmark position and the ID of the landmark as seen by the landmark extraction code. This correspondence is built up dynamically depending on the order in which the landmarks are first detected. The state estimate will be just as explained in section 5.2:

$$\hat{x}(k) = \begin{bmatrix} x_r(k) \\ y_r(k) \\ \theta_r(k) \\ x_1(k) \\ y_1(k) \\ \vdots \\ x_n(k) \\ y_n(k) \end{bmatrix}$$

In addition to the state vector, there is an array that contains the correspondences between the landmark indexes shown in the state vector above and the landmark ID's. The array is defined by

$$\texttt{LM\_IDs} = \begin{bmatrix} \texttt{ID1} & \texttt{ID2} & \ldots & \texttt{IDn} \end{bmatrix},$$

where `ID1` corresponds to the 1st landmark in the system state, `ID2` corresponds to the 2nd landmark in the system state, etc..

Landmark extraction has completed and output its data to the Kalman update step. First, the H matrix must be built for each measurement. As discussed in section 5.2, the H matrix is composed of many different H_i matrices, one for each landmark. The landmark extraction code outputs an array of landmark data in which ea ch element is of the form

$$\texttt{LM\_Meas} = \left(\texttt{LM\_ID}, \begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} Var(x) & Cov(x,y) \\ Cov(x,y) & Var(y) \end{bmatrix} \right),$$

where `LM_ID` is the ID of the landmark as determined by the landmark extraction code, $\begin{bmatrix} x \\ y \end{bmatrix}$ is the measured (x, y) coordinate of the landmark in the reference frame, and the matrix

$$\begin{bmatrix} Var(x) & Cov(x,y) \\ Cov(x,y) & Var(y) \end{bmatrix} \tag{6.2}$$

is the covariance matrix for $\begin{bmatrix} x \\ y \end{bmatrix}$. If the landmark is already in the state vector, the proper index is found by searching for `LM_ID` in the `LM_IDs` array. This index is the i index in H_i. So, the H_i

matrix is built as discussed in section 5.2. The H_i matrix is re-stated here to refresh the reader's memory

$$H_i = \begin{bmatrix} -1 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 & 1 & \cdots & 0 \end{bmatrix}.$$

The next index must also be looked up in `LM_IDs`. It may not be $i+1$, since it is not known which landmarks will be observed at each step. However, there does not need to be an H_i for all landmarks in the state vector. As long as each H_i is defined properly, the measurement equation will be satisfied. All of these H_i's are combined to make up a $2N \times (3n+2)$ H matrix.

The W matrix must also be built for this iteration. This matrix is built from the covariance matrix mentioned in equation (6.2). If $W_1, \ldots, W_N$ is are the covariance matrices, the complete $W(k)$ matrix is a block matrix defined by

$$W(k) = \begin{bmatrix} W_1 & 0 & \cdots & 0 \\ 0 & W_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & W_N \end{bmatrix}.$$

If this is the first time the landmark has been observed, it cannot be looked up in the state vector; it must be added. Landmarks that have never been observed before are added to the state estimate and do not take part in the Kalman update step. To add a landmark to the system, the state vector and error covariance must be expanded, `LM_IDs` must be updated, and the control system matrices must also be updated. To add the landmark to the state vector, the measured landmark position $\begin{bmatrix} x \\ y \end{bmatrix}$ is inserted just below the nth landmark position estimate in $\hat{x}(k)$ as $\begin{bmatrix} x_{n+1}(k) \\ y_{n+1}(k) \end{bmatrix}$:

$$\hat{x}(k) = \begin{bmatrix} x_r(k) \\ y_r(k) \\ \theta_r(k) \\ x_1(k) \\ y_1(k) \\ \vdots \\ x_{n+1}(k) \\ y_{n+1}(k) \end{bmatrix}.$$

The augmented $P(k)$, denoted $P'(k)$, is computed by inserting the covariance matrix from `LM_Meas` at the lower right diagonal of $P(k)$:

$$P'(k) = \begin{bmatrix} P(k) & 0 & 0 \\ 0 & Var(x) & Cov(x,y) \\ 0 & Cov(x,y) & Var(y) \end{bmatrix}.$$

Next the control system model must be modified. First, $F(k)$ is still the identity matrix, except now its size is increased to $(3+2(n+1)) \times (3+2(n+1))$ to match the state vector. Two rows of zeros are added to $G(k)$ to make it the correct size of $(3+2n) \times 3$. Two rows of zeros and two columns of zeros are also added to $V(k)$ to make it the correct size. Unless a new landmark is added, these matrices all stay the same.

Figure 20 shows a flowchart of the landmark management portion of the code. The "Add LM to $\vec{y}(k)$" block also changes $H(k)$ and $W(k)$ as discussed above. Similarly, the "Add LM to state" block also augments $F(k)$, $P(k)$, $G(k)$, and $V(k)$ as discussed above.

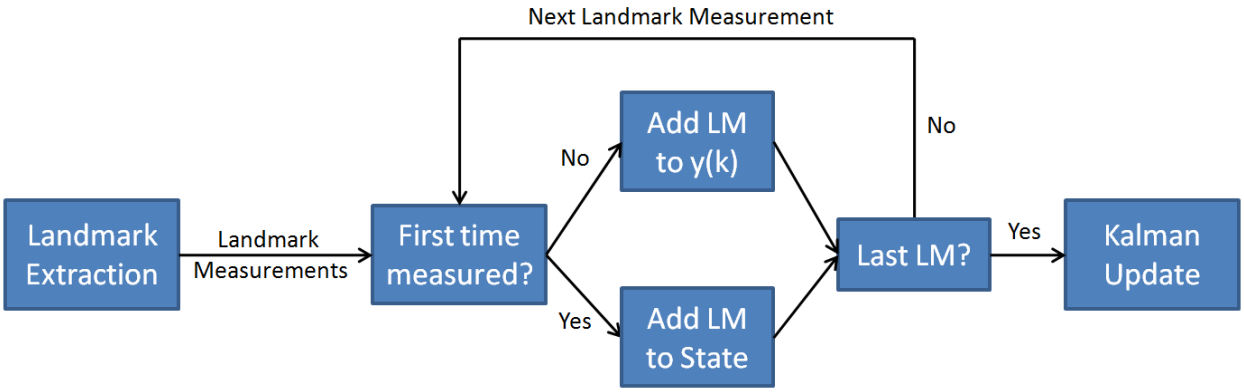

Figure 20: Flowchart for the landmark management code.

Kalman Filter For this experiment, the Kalman filter was streamlined for easier usability. Two new data types were created, called `KalmanEstimate` and `StochasticLinearSystem`. The `KalmanEstimate` type contains the current state estimate, $\hat{x}(k)$, and the current error covariance estimate $P(k)$. The `StochasticLinearSystem` type contains $F(k)$, $G(k)$, $H(k)$, $V(k)$, and $W(k)$.

The Kalman Predict and Update function use the same Kalman predict and update equations as stated in section 5.1, but the input now is both a variable of type `KalmanEstimate` and one of type `StochasticLinearSystem`. This change in convention has improved the organization of the code.

Map Generation Map generation is done exactly the same as for the straight line experiment, except the second array of landmark positions contains the position of the green and blue landmarks, in addition to the red one. Figure 21 shows a map generated after one run of the Multi-Landmark SLAM algorithm.

6.2.3 Results

Initially, the experiment yielded mixed results when the robot was stationary and three landmarks were in view. The location of one of the landmarks was always correct, but the location of the other two landmarks was correct at times and incorrect at times. The cause of this was found to be the incorrect measurements of 0 present in the laser rangefinder scan previously discussed. In the sub-array of 40 elements, there were objects measured in the scan that were not landmarks, such as the wall. When there were several measurement readings of 0 it would register as a transition of greater than 50 mm and would then be used incorrectly as the landmark measurement. To account for this, a cut-off distance was implemented. If a measurement was greater than this distance, it would be set to 0 in the sub-array. This would then not allow it to register as a transition, and the correct landmark measurement could be found from the array. Once this correction was made, the measurements of all three of the landmarks were correct. From that point, a test was done where one landmark was in the field of view of the robot, and the second was added after a period of time. This caused the program to crash due to non-conformability of Kalman filter matrices. Specifically, the dimensions of the error covariance P matrix did not change as they were supposed to with the addition and or removal of landmarks. Instead of re-building the entire system and measurement model every step, the system model, $F(k)$ and $G(k)$, was changed only when there is a new landmark. The list of ID's and landmark measurements are used to build the measurement

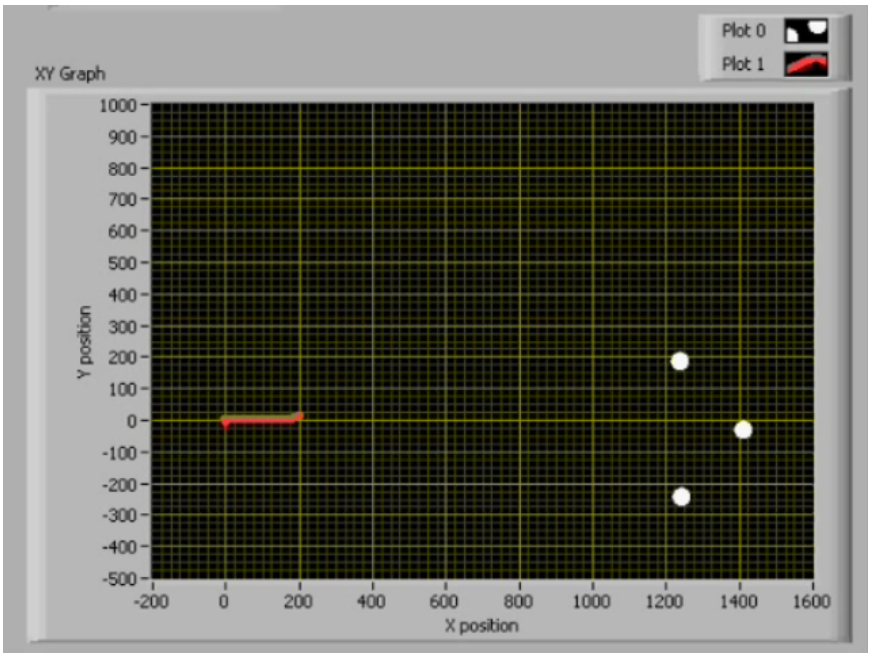

Figure 21: Map generated by the multi-landmark version of the SLAM algorithm

model, the $H(k)$ matrix. Figure 21 shows the results of a run of the experiment. The landmarks are shown by the white dots, and the path of the robot is shown by the red line. Landmarks were able to be added and subtracted as they were seen, and the system state was correctly estimated.

References

[1] "Create Manuals." IRobot Corporation: Home Page. Web. 03 Dec. 2009. <http://www.irobot.com/hrd_right_rail/create_rr/create_fam/createFam_rr_manuals.html>.

[2] Ranganathan, Ananth and Frank Dellaert. *Automatic Landmark Detection for Topological Mapping Using Bayesian Surprise (Technical Report No. GT-IC-08-04)*, College of Computing, Georgia Institute of Technology, 2008.

[3] Arana Arejolaleiba, N., F. Lerasle, M. Briot, C. Lemaire, and J. B. Hayet. "A Smart Sensor Based Visual Landmarks Detection for Indoor Robot". *ICPR02.* 848-51. Print.

[4] Hayet, J.B., F. Lerasle, M Devy, "Visual landmarks detection and recognition for mobile robot navigation," *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* , vol.2, no., pp. II-313- II-318 vol.2, 18-20 June 2003 <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1211485&isnumber=27266>

[5] Choset, Howie, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents).* New York: The MIT, 2005. Print.

[6] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics.* Cambridge, Mass.: MIT, 2006. Print.

[7] Durrant-Whyte, Hugh, and Tim Bailey. "Simultaneous Localization and Mapping: Part 1." IEEE Robotics and Automation Magazine (June 2006): 99-108. Print.

[8] Madhavan, Raj, Hugh Durrant-Whyte, and Gamini Dissanayake. *Natural Landmark-based Autonomous Navigation using Curvature Scale Space.* Proc. of 2002 IEEE International Conference on Robotics & Automation, Washington, D.C. Print.

[9] Dailey, Matthew N., and Manukid Parnichkun. *Landmark-based Simultaneous Localization and Mapping with Stereovision. Tech.*

[10] "Simple Map Utilities: Simple Mapping Utilities (pmap)." USC Robotics Research Lab. Web. 31 Mar. 2010. <http://www-robotics.usc.edu/ ahoward/pmap/index.html>.

[11] Pitman, Jim. *Probability.* New York: Springer-Verlag, 1993. Print.