Wall Following Algorithm for a Mobile Robot Using Extended Kalman Filter

by

Xueming Wang

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 6, 2011

Keywords: Extended Kalman Filter, Wall Following, Mobile Robot

Approved by

Thaddeus Roppel, Chair, Associate Professor of Electrical and Computer Engineering
Prathima Agrawal, Ginn Distinguished Professor of Electrical and Computer Engineering
Robert Dean, Assistant Professor of Electrical and Computer Engineering

Abstract

This thesis proposes a wall-following algorithm for mobile robots using the Extended Kalman Filter. The robot navigates in a known environment where the baseboard of the wall is used as a natural landmark and indentified by using the Hough transform. The prediction phase of the Kalman Filter is implemented using the odometry model of the robot. The update phase directly uses the parameters of the lines detected by the Hough algorithm to correct the robot's pose. A proportional control is then performed to adjust the speeds on the wheels of the robot so that it follows the wall in a straight line. The results provided by the Matlab simulation environment showed that the wall-following algorithm could converge well to the real world values.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Associate Professor Dr. Thaddeus A. Roppel without whose guidance and support this thesis would not be possible. His understanding for my situation and encouragement carried me through my research. I gratefully acknowledge the facilities and financial support provided by Dr. Prathima Agrawal. I also thank Dr. Robert Dean for being a part of my advisory committee and catering time during his busy schedules.

This work was supported in part by the National Science Foundation under grant IIP-0738088.

My thanks goes to my friends Chris Wilson, Aditya Singh and Nida Bano for discussions and valuable suggestions on my research work.

I am grateful to my parents for their consistent support and encouragement during my study.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The primary objective of the work described in this thesis is to add vision-based navigation to the capabilities of the robots developed in the Cooperative Robotics Research (CRR) Lab at Auburn University.

The CRR lab is currently developing its second generation of robots primarily intended for research related to search and rescue missions. These robots utilize a laptop PC as a computational platform, envisioning it will provide sufficient computational capability to experiment with vision-based mobile robot navigation methods.

Navigation can be roughly described as the process of determining a suitable and safe path between a starting and a goal point for a robot travelling between them[2][3]. Different sensors have been used for this purpose, which has led to a varied spectrum of solutions. Visual navigation for mobile robots has become a source of countless research contributions since navigation strategies based on vision can increase the scope of application of autonomous mobile robots.

Vision is a powerful sensor as it can mimic the human sense of vision and allow non-contact measurement of the working environment. This thesis is focused on visual servoing, also known as vision-based robot control. Figure 1.1 shows the block diagram of a typical visual servo system where vision (through a camera) is a part of the control system. In the control loop, a vision system is used for robot control, by providing feedback information about the state of the environment to the controller.

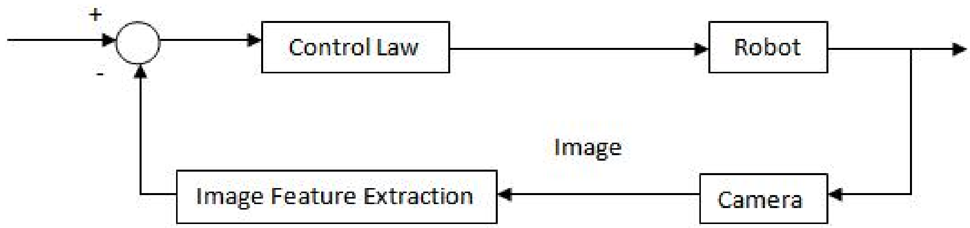

Figure 1.1: Block Diagram of a typical visual servo system

To follow a wall, or in a more general sense to follow the contours of an object, is an important behavior for autonomous mobile robots. Robots operating in an unknown, unstructured environment use their sensors to perceive the surroundings and plan their motions or trajectories accordingly [4]. The basic capability desired for the robots described here is to be able to navigate the hallways of our research facility (3rd floor of Broun Hall on the Auburn University campus). Thus, wall-following behavior is a crucial building block of the entire robot operation.

A particular feature of the walls in the target environment comes in handy in the design of the wall-following algorithm. Along the bottom edge of the walls there are dark gray baseboards of about two and a half inches in width. In this thesis, the webcam built into a laptop mounted on top of the robot is used to take images of the wall. This provides an image containing a clean, high-contrast edge. The orientation of the edge in the image varies predictably with the distance of the robot's center from the wall and with the robot's heading angle. The presence of systematic error (bias) and random error (noise) in real life scenarios as compared to ideal environments demands the use of probabilistic approaches for modeling robotic movements.

The vision-based navigation algorithm described in this thesis has been developed in parallel with hardware development for the most recent generation of robots in the CRR lab. Although not part of this work, it is intended for these results to be incorporated as part of the final design.

In Chapter 2 an overview of the literature in the domain of vision based mobile robot navigation is provided. Chapter 3 encompasses a complete description of the hardware that is used for performing the simulations. This chapter gives a detailed explanation of the vision sensor along with the type of images being studied. It also explains the robot's hardware and the hardware test bed environment on which the simulation models are drawn. Chapter 4 introduces the problem in a mathematical perspective and also provides the notations that are being used through the document. Also a sufficient amount of description and pseudo-code of the algorithms is provided for better comprehension. In Chapter 5 the simulation methods and the results obtained is provided. In Chapter 6 the directions for future work is discussed and a brief conclusion is drafted. An appendix with reference to the source code used for simulation is also given.

Chapter 2

Literature Review

Mobile robot vision-based navigation has been the source of immeasurable research contributions, from the domains of both vision and control. Vision has become more and more widespread in applications such as localization, automatic map construction, autonomous navigation, path following, inspection, monitoring or risky situation detection [5].

2.1 Visual Servoing

One of the earliest papers that talks about visual servoing was from the SRI international labs [6]. A first tutorial on visual servoing was published in 1996 by S. A. Hutchinson, G.D. Hager, and P.I. Corke [7], more recent tutorials were published in 2006 and 2007 by F. Chaumette and S. Hutchinson[8] [9].

Visual servoing techniques are broadly classified into the following types: image based (IBVS),position based (PBVS) and hybrid approach.

2.1.1 Image-Based Visual Servoing

IBVS was proposed by Weiss and Sanderson [10]. The control algorithm is based on the error between current and desired features on the image plane, and does not involve any estimate of the pose of the target. The features may be lines or moments of regions, the coordinates of visual features. IBVS has difficulties [11] with motions of very large rotations, which has come to be called camera retreat [12].

2.1.2 Position-Based Visual Servoing

PBVS is sometimes referred to as Pose-Based VS and is a model-based technique (with a single camera). This is because the pose of the object of interest is estimated with respect to the camera and then a command is issued to the robot controller, which in turn controls the robot. In this case the image features are extracted as well, but are additionally used to estimate 3D information, hence it is servoing in 3D.

2.1.3 Hybrid Approach

Hybrid approaches use some combination of the 2D and 3D servoing [48].

2.2 Indoor Navigation

Many robot control systems have included some information about the environment where the robot has to navigate. The navigation and localization systems generally used fall largely within one of the following three categories: mapless navigation systems, map-building-based navigation systems, map-based navigation systems.

2.2.1 Mapless Navigation

This category includes all navigation approaches that do not need awareness of the environment to run. The movements of the robot rely on the elements observed in the environment (doors, desks, features, walls, etc). Two major techniques being used are appearance-based navigation and optical flow. Appearance-based matching methods are based on the storage of images in a previous recording phase. These images are then used as templates. The robot self-locates and navigates in the environment matching the current viewed frame with the stored templates. Examples of these approaches are provided next.

Jones et al. [25] describe research in which a sequence of images and associated actions are stored in the robot memory. During navigation, the robot recovers the template that best matches the on-line frame. If the match is above a threshold, the robot runs the action associated to that template.

Matsumoto et al. [24] describe VSRR (view sequenced route representation), where a sequence of images is stored to be used as a memory. The robot repeats the same trajectory comparing the on-line scene with all stored images using correlation. This approach basically focuses on how to memorize the views.

Optical flow based methods estimate the motion of objects or features within a sequence of images. Researchers compute optical flow mostly using pioneering techniques from Horn and Schunk [22] and Lucas and Kanade [23].

2.2.2 Map-Building-Based Navigation

The navigation process starts once the robot has explored the environment and stored its representation. The first to consider this technique was Moravec with his Stanford Cart [19]. This system was improved by Thorpe for the robot FIDO [20], and was used to extract features from images. These features were then correlated to generate their 3D coordinates. The features were represented in an occupancy grid of two square meter cells. A topological representation of the environment is an alternative to an occupancy grid. These systems are based on generating a graph of nodes representing the space, and storing metrical information for every node recognized during the navigation process. Thrun [21] combined the best of occupancy grids and topological maps for navigation.

2.2.3 Map-based Navigation

These methods are based on providing the robot with models of the environment. A process known as self-localization and is fundamental for a correct navigation. The main steps are: (i)acquire image information, (ii) detect landmarks in current views (edges, corners, objects), (iii) match observed landmarks with those contained in the stored map according to certain criteria, (iv)update the robot position, as a function of the matched landmarks location in the map.

To solve the localization problem, absolute localization methods contrast with relative localization strategies. In absolute localization methods, the initial position of the robot is unknown. This self-localization problem has been solved either using deterministic triangulation [13], Markov [14], or Montecarlo [15] localization.

In relative localization, it is assumed that, at the beginning of the navigation, the position of the robot is approximately known. Matthies and Shafer [16] used stereo vision to reduce errors. Tsubouchi and Yuta [17] used a CAD model for environment representation. Later on, Christensen et al. [18] also used CAD models for space representation combined with stereo vision to reduce errors.

2.3 Wall-follow Algorithms

The wall-following control problem is characterized by moving the robot along a wall in a desired direction while maintaining a constant distance to that wall. This can be the case in the following situations.

2.3.1 Following an unknown wall

When there is little or no knowledge about the environment, the trajectory may be specified as 'follow the wall on the right until the first doorway'. When world modeling is the specific goal of the robot, it may be necessary to follow a wall to model it completely. The position of the robot can be calculated by means of dead-reckoning.

2.3.2 Following a known wall

When the trajectory of the robot has been planned, it can be followed by means of dead-reckoning. However, dead-reckoning methods suffer from accumulating errors. A way to keep these errors small is by tracking a wall [26]. The planning must include the availability of walls in the route description in that case [27]. This is the reverse of the previous case: now the position of the wall has to be known accurately.

2.3.3 Obstacle avoidance

When the sensors can't provide an overview of the shape or the size of the obstacle, it is not possible to plan an evasive route. Then it becomes necessary to follow the contour of the obstacle until such is possible or until the original route can be resumed.

Chapter 3

Hardware and Software

3.1 Hardware

The Cooperative Robotics Research Lab at Auburn University has a team of robots for general research purposes. These will be referred to here as “Generation II”. The Generation II robot prototype is equipped with a laptop as its main control unit and ultrasonic sensors for distance measurement. It uses an Arduino microcontroller as an agent to transfer commands between the sensors and motors. The simulations presented in this thesis are performed by making use of the existing robot hardware and utilizing the webcam on the laptop.

The following sections will give a brief introduction to the hardware employed in the Generation II robotics testbed for better comprehension of the environment in which the simulations are being performed. There are two fundamental parts to the hardware used in assessing the effect of vision-based wall-following algorithms: the experimental field and the robots themselves. This section will be divided up in the same way. The experimental field will be described first, followed by the robotic hardware.

3.1.1 Experimental Field

The experimental field for Generation II robots is the hallway on the third floor of Broun Hall. The floor plan is shown in Fig. 3.1

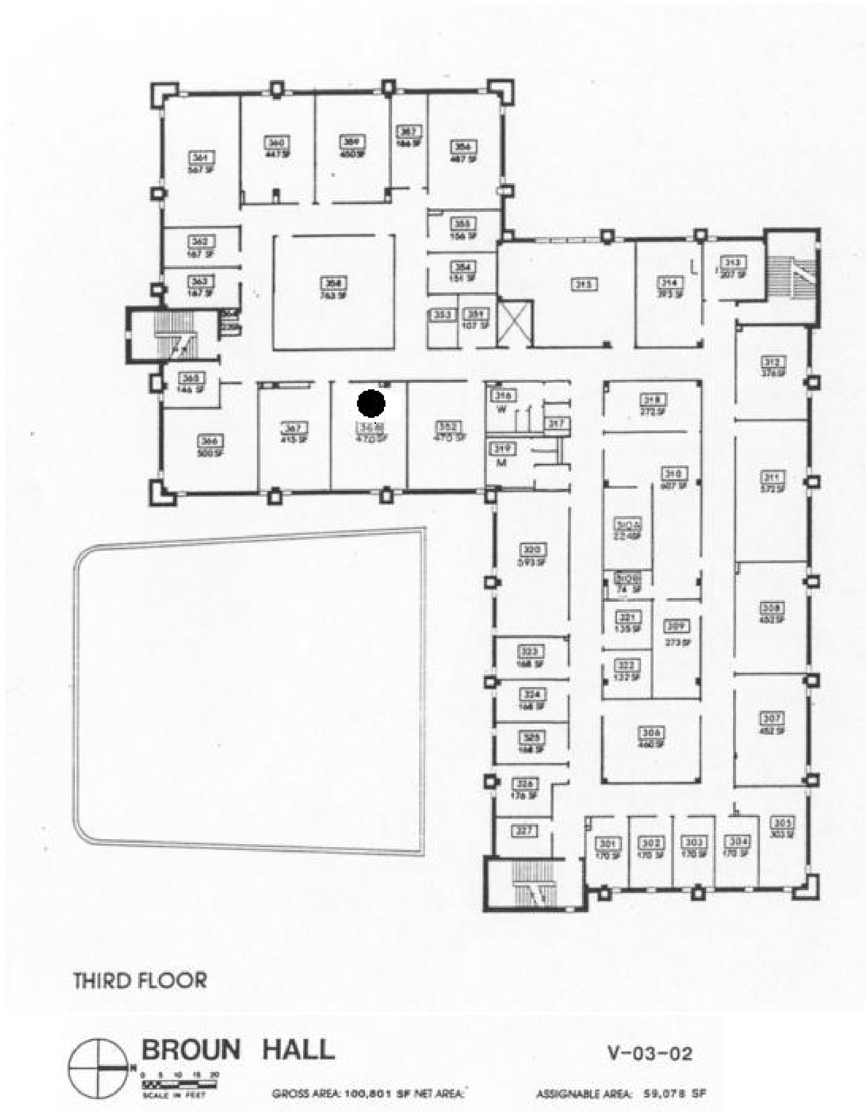

Figure 3.1: Broun Hall Third Floor Floor Plan.The Cooperative Robotics Research Lab is indicated by the large black dot.

3.1.2 Robotic Hardware

The robot hardware assembly is shown in Fig. 3.2. It consists of a base carrying a laptop, sensors, and auxiliary electronics.

Figure 3.2: Robot Hardware

Acer Aspire 1410 Laptop

This laptop is chosen to function as the main control unit. Some of the specifications of the laptop includes: Intel Celeron processor 723/743(1 MB L2 cache, 1.20/1.30 GHz, 800 MHz FSB, 10W), 800 MHz Front Side Bus, Dual-channel DDR 2 SDRAM support with up to 2 GB of DDR2 667 MHz memory, 11.6” HD 1366x768 pixel resolution, 160/250 GB hard disk drive, Integrated Acer Crystal Eye webcam supporting enhanced Acer PrimaLite technology, Wi-Fi Certified network connection, three USB 2.0 ports.

Robot Base

The Max 96 Round Mobile Robot Base Kit was selected as our robot frame. This kit has 1 base and 2 decks with a 16" diameter. The base has dual 12 volt 20in-lb torque drive motors (The Max speed is 39 feet per minute under full load) and is balanced with two casters. The drive wheels are six inches (15 cm) in diameter and each caster wheel is three inches (7.5 cm) in diameter. The maximum recommended payload of this base is 35 lbs (13.6 kgs). Two three channel optical encoders and a power distribution board are also included.

Arduino Mega

The Max 96 Round Robot Base kit initially came with a microcontroller board HC11. After experimenting with this microcontroller we found several issues. First, HC11 connects through a 3 wire serial RS232 and laptops only have USB ports. We used an RS232 to USB adapter to solve this problem. Second, coding with HC11 is not as user-friendly as we expected for mobile robot application. These two issues led us to eventually abandon the HC11 microcontroller and switch to Arduino Mega which has become quite prominent in the robotic world. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing).

The Arduino Mega is a microcontroller board based on the ATmega128. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

The Arduino Mega has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega1280 provides four hardware UARTs for TTL (5V) serial communication. An FTDI FT232RL on the board channels one of these over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer.

IM-15 Gearmotors (DC Permanent Magnet Spur Gearmotors)

Two motors are used for left and right differential driving wheels. Each motor has three connections: Enable, I1 and I2. For the left motor we connect them to Pin 2, 25, 24 respectively on the Arduino Mega board. The motor weighs 15 to 16.5 ounces.

Two Channel Optical Encoders

The encoders we used for the robots are HEDS-5500/5540. They are two channel optical incremental encoders. Each encoder contains a lensed LED source, an integrated circuit with detectors and output circuitry, and a codewheel which rotates between the emitter and detector IC. The outputs of the encoders are two square waves in quadrature.

Parallax PING))) Ultrasonic Sensors

Generation II robots initially used with Infra-red sensors, but due to the output non-linearity we decided to switch to Parallax PING))) ultrasonic sensors. This sensor is very cost-effective and provides an easy method of distance measurement. There is an activity status LED and just one I/O pin which makes it relatively easy to use. Some of its features are: 5V DC supply voltage, 2 cm to 3 m range, positive TTL pulse input trigger, 40KHz burst frequency, 200 us delay before next measurement.

3.2 Software

MATLAB is a scientific programming language and provides strong mathematical and numerical support for the implementation of advanced algorithms. It comes with a large set of toolboxes. In this thesis Matlab is chosen for implementation of simulation. The image processing toolbox was use for simulation.

Chapter 4

Wall-following Algorithm

4.1 Mobile Robot Kinematic Model

The mobile robot used in this experiment has a differential drive system. It has three wheels. There are two drive wheels located at either side of the transverse midline. There is also a front caster wheel and rear caster wheel. Drive wheels can be controlled independently from each other. The kinematic model of the mobile robot is given by the following relations [28] [36],

$$x(k+1) = x(k) + D(k) \cdot cos(\theta(k+1)) \tag{4.1}$$

$$y(k+1) = y(k) + D(k) \cdot sin(\theta(k+1)) \tag{4.2}$$

$$\theta(k+1) = \theta(k) + \Delta\theta(k) \tag{4.3}$$

$$D(k) = v_t(k) \cdot T \tag{4.4}$$

$$\Delta\theta(k) = \omega(k) \cdot T \tag{4.5}$$

$$v_t(k) = \frac{v_L(k) + v_R(k)}{2} \tag{4.6}$$

$$\omega(k) = \frac{v_R(k) - v_L(k)}{b} \tag{4.7}$$

where $x(k)$ and $y(k)$ are coordinates of the center of the robot (mm); $D(k)$ is the distance travelled between time step k and $k+1$ (mm); $v_t(k)$ is the robot translation speed (mm/s); T is sampling time (s); $\theta(k)$ is the angle between the vehicle and x-axis (°); $\Delta\theta(k)$ is the rotation angle between time step k and $k+1$ (°); $v_L(k)$ and $v_R(k)$ are velocities of the left and right wheel, respectively (mm/s); b is the vehicle axle length (mm).

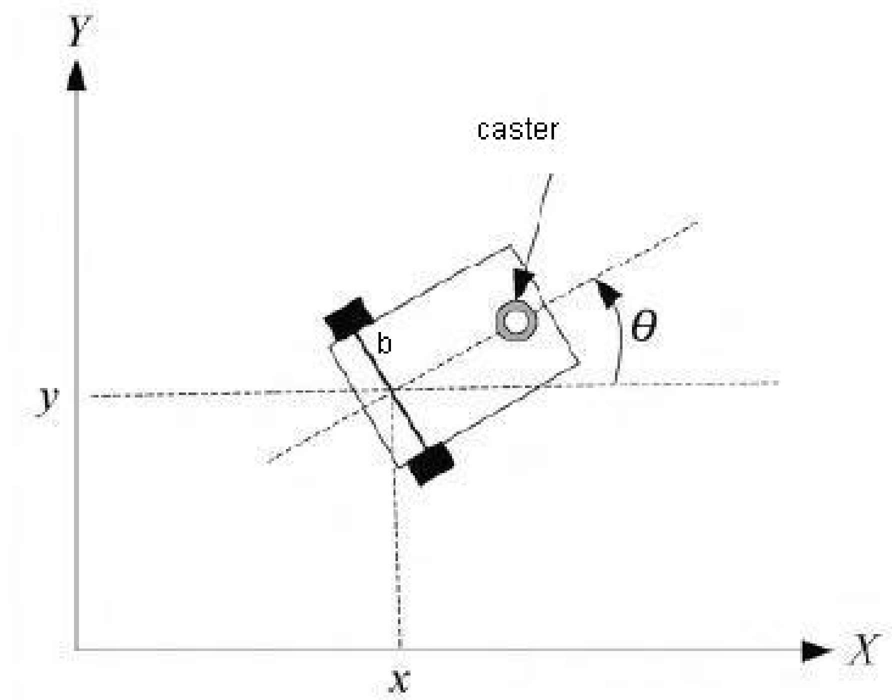

Figure 4.1: Schematic model of a differential-drive mobile robot

Equations (1) to (7) describe the basic odometry pose tracking model and results obtained using this technique are marked as uncalibrated odometry (UO).

4.2 Sensor Model for Detecting Natural Landmarks

There is a distinct feature of the hallway in Broun Hall. Both the wall and floor are white except that the baseboard along the wall is dark gray. This forms a natural landmark for the image sensor system in this thesis. The technique adopted to identify the dark gray line is the Hough transform. This kind of transform is a method employed to identify inside a digital image a class of geometric forms which can be represented by a parametric curve [30].

In an image space, a straight line can be described as $y = mx + b$ and can be graphically plotted for each pair of image points (x, y). In the Hough transform, a main idea is to consider the characteristics of the straight line not as image points $(x_1, y_1), (x_2, y_2)$, etc., but instead in terms of the slope parameter m and the intercept parameter b, which is always denoted as ρ and θ for the lines in the Hough transform.

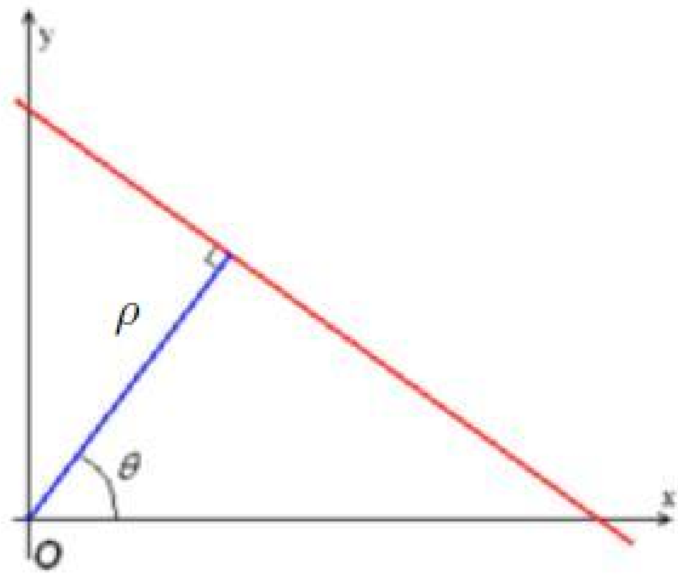

Figure 4.2: ρ and θ in Hough transform

The parameter ρ represents the distance between the line and the origin, while θ is the angle of the vector from the origin to this closest point.

The Hough transform algorithm uses an array, called an accumulator, to detect the existence of a line $y = mx + b$. For each pixel and its neighborhood, the Hough transform algorithm determines if there is enough evidence of an edge at that pixel. If so, it will calculate the parameters of that line, and then look for the accumulator's bin that the parameters fall into, and increase the value of that bin. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted, and their (approximate) geometric definitions read off [31].

4.3 Extended Kalman Filter Based Sensor Fusion [43] [44] [45] [46]

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. The equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. This stage can also be thought of as being a prediction of the state. The measurement update equations are responsible for the feedback – for incorporating a new measurement into the *a posteriori* estimate. Because the measurement is used to improve the estimate obtained by projecting the state forward in time, this stage can also be thought of as being a correction of the predicted state. Figure 4.3 illustrates the process of Kalman filtering.

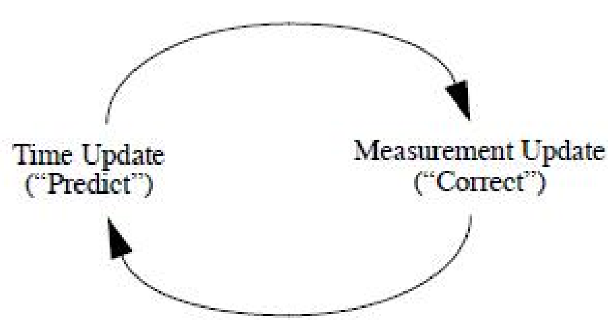

Figure 4.3: The ongoing discrete Kalman filter cycle

One problem with using a Kalman filter in this mobile robotics application is that the kinematic model is non-linear. In something akin to a Taylor series, we can linearize the estimation around the current estimate using partial derivatives of the process. A Kalman filter that linearizes about the current mean and covariances is referred to as an extended Kalman filter or EKF.

4.3.1 Extended Kalman Filter time update stage

Using the kinematic model stated in section 4.1, let's assume that our process has a state vector $x \in \mathbb{R}^n$ and it is governed by the non-linear stochastic difference equation

$$x(k+1) = f(x(k), u(k), \omega(k-1)), \tag{4.8}$$

with a measurement $z \in \mathbb{R}^m$ that is

$$z(k) = h(x(k), v(k)), \tag{4.9}$$

where the random variables $\omega(k)$ and $v(k)$ represent the process and measurement noise respectively. They are assumed to be independent of each other, white and with normal probability distributions

$$p(\omega) \sim N(0, Q) \tag{4.10}$$

$$p(v) \sim N(0, R) \tag{4.11}$$

Q is the process noise covariance matrix and R is the measurement noise covariance matrix.

$$Q = \sigma_Q^2 \cdot I \tag{4.12}$$

$$R = \sigma_R^2 \cdot I \tag{4.13}$$

where σ_Q^2 and σ_R^2 are variances and I is the identity matrix.

$u(k)$ represents control input. The non-linear function f relates the state at the previous time step $k-1$ to the state at the current time step k. The non-linear function h relates the state $x(k)$ to the measurement $z(k)$.

The state vector and control input are used to compute the state vector at the next time step.

$$x(k+1|k) = f(x(k|k), u(k), E\{\omega(k)\}) \tag{4.14}$$

By implementing the robot kinematic model we get

$$f(\hat{x}_{k-1}, u_{k-1}, 0) = \begin{bmatrix} x(k) + D(k) \cdot cos(\theta(k) + \Delta\theta(k)) \\ y(k) + D(k) \cdot sin(\theta(k) + \Delta\theta(k)) \\ \theta(k) + \Delta\theta(k) \end{bmatrix} \tag{4.15}$$

The noise covariance $Q(k)$ was modeled on the assumption that there are two sources of error, angular and translational. Thus $x(k\hat{+}1|k)$ depends on $x(\hat{k}|k)$, $\Delta\theta(k)$, and $D(k)$. In order to translate the uncertainty in $\Delta\theta(k)$ and $D(k)$, a partial differentiation of (4.13) with respect to $\Delta\theta(k)$ and $D(k)$ is done and gives the following jacobian:

$$\nabla f(k) = \begin{bmatrix} -D(k) \cdot sin(\theta(k) + \Delta\theta(k)) & cos(\theta(k) + \Delta\theta(k)) \\ D(k) \cdot cos(\theta(k) + \Delta\theta(k)) & sin(\theta(k) + \Delta\theta(k)) \\ 1 & 0 \end{bmatrix} \tag{4.16}$$

The complete expression for $Q(k)$ is then:

$$Q(k) = \nabla f(k) \cdot Q \cdot \nabla f(k)^T \tag{4.17}$$

Another source of uncertainty is the uncertainty in the position and orientation at time step k, $P(k|k)$, carried forward to time step $k+1$. Another Jacobian is used to determine how the uncertainty is transferred between the time steps. A partial differentiation of (4.13) with respect to $x(k)$, $y(k)$, and $\theta(k)$ is performed and the resulting Jacobian $\nabla f^{'}(k)$ is given by,

$$\nabla f^{'}(k) = \begin{bmatrix} 1 & 0 & -D(k) \cdot sin(\theta(k) + \Delta\theta(k)) \\ 0 & 1 & D(k) \cdot cos(\theta(k) + \Delta\theta(k)) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.18}$$

and thus the complete pre-localization covariance matrix expression for $x(k+1|k)$ becomes:

$$P(k+1|k) = \nabla f^{'}(k) \cdot P(k|k) \cdot \nabla f^{'}(k)^T + Q(k) \tag{4.19}$$

4.3.2 Extended Kalman Filter measurement update stage

The image sensor model stated in section 4.2 is used to improve the mobile robot pose estimation. The model gives information of the distance between the robot center and the wall (y) and the angular rotation of the robot ($\Delta\theta$), thus the measurement matrix H that relates the state to the measurement z_k is simply

$$H = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

The measurement innovation (residual) covariance matrix $S(k)$ is

$$S(k+1) = H \cdot P(k+1|k) \cdot H + R(k+1) \tag{4.21}$$

where $R(k)$ denotes the measurement noise matrix. It's a diagonal matrix with the $r_i(k+1)$ values on the diagonal. The Kalman gain can now be computed as:

$$K(k+1) = P(k+1|k) \cdot H^T \cdot S^{-1}(k+1) \tag{4.22}$$

The variance of the new pose estimate is computed as:

$$P(k+1|k+1) = P(k+1|k) - K(k+1) \cdot S(k+1) \cdot K^T(k+1) \tag{4.23}$$

The form of the EKF to take the latest range readings for the new pose estimate is now:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1) \cdot (z(k+1) - h(k+1)) \tag{4.24}$$

4.4 Speed adjustment

The desired wall-following behavior is to follow the wall in a perfectly straight line. After we get the pose estimate from the EKF step, we compare the estimated pose with the desired pose and then adjust the speed on both wheels so that the robot will drive towards the desired pose.

The distance from the estimated pose at step k to the desired pose at step $k+1$ is:

$$d = \sqrt{(x(k+1) - x(k))^2 + (y_{estimated}(k) - y_{desired}(k))^2} \tag{4.25}$$

and this distance is equal to

$$d = v \cdot T \tag{4.26}$$

where v is the average speed and T is the sampling time.

The rotation angle the robot needs to correct in order to achieve the desired 0 degree rotation angle is simply $\theta_{estimated}$. And the relationship between the angle and the angular speed ω is:

$$0 - \theta_{estimated} = \omega \cdot T \tag{4.27}$$

where again T is the sampling time. We also have the equations that express the relationships between left and right wheel speeds vL and vR and average speed v and angular speed ω:

$$v = \frac{vL + vR}{2} \tag{4.28}$$

$$\omega = \frac{vR - vL}{b} \tag{4.29}$$

where b is the radius of the robot body. By using the equations stated above in this section we can calculate the new set of differential speeds for the robot wheels so that the robot will drive towards the desired straight wall-following line. A top-level block diagram is shown below:

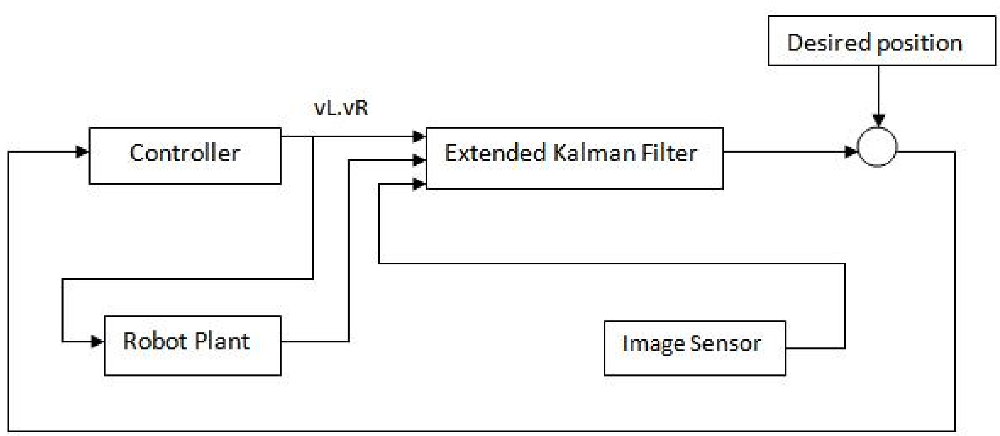

Figure 4.4: Top-level Block Diagram

Chapter 5

Experimental Set-up and Results

The experimental set-up consists of 2 major components: Matlab software on which the simulations are performed and the hardware implementation for the image sensor model.

5.1 Hardware Implementation for Image Sensor Model

5.1.1 The Set-up

The tool used for taking digital images of the environment is a built-in webcam of the laptop PC which sits on top of the robot. To make sure that every time the camera will point at the same angle, I fixed the laptop opening at a 65 degree angle. Also the laptop is positioned at such a location on the robot that the webcam is at the center of the robot.

5.1.2 Digital Image Processing

Distance to wall with no rotation angle

Below are some raw images taken by the camera of the target environment. The lower part of each image is the floor; the dark line is the baseboard; the upper part of the image is the wall.

Figure 5.1: Raw images with no rotation angle

The dark line corresponding to the baseboard appears to shift up as the robot moves away from the wall. After image processing we get the following images accordingly:

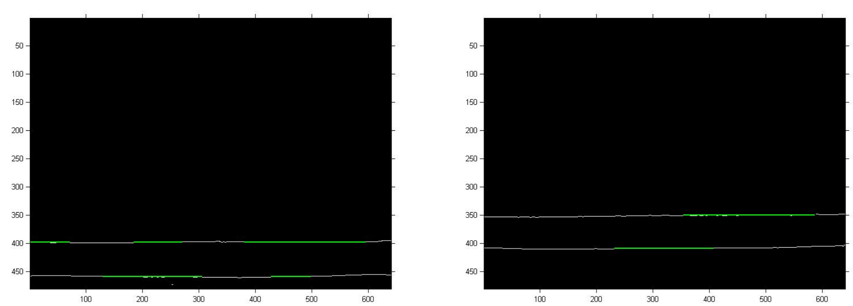

Figure 5.2: distance to wall is 54 cm   distance to wall is 64 cm

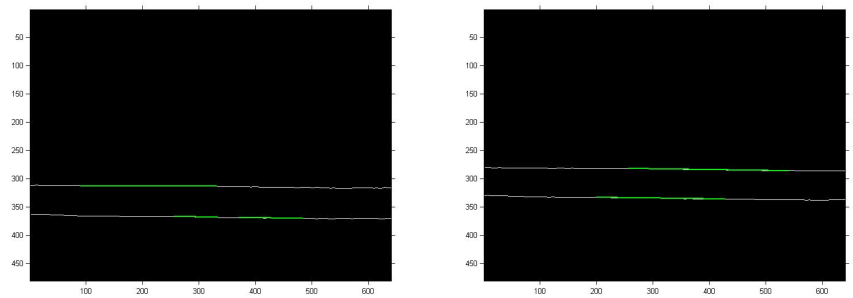

Figure 5.3: distance to wall is 74 cm   distance to wall is 84 cm

Robot Rotation Angle

If the robot body does not directly face the front but has some rotation angle instead, the gray line in the digital image rotates accordingly. The following images are taken when robot body rotates 5 degrees to the left and right.

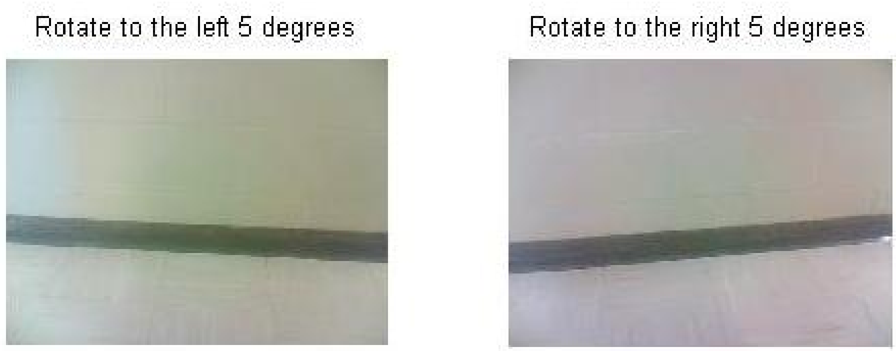

Figure 5.4: Raw images with rotation angles

And the images after digital image processing are:

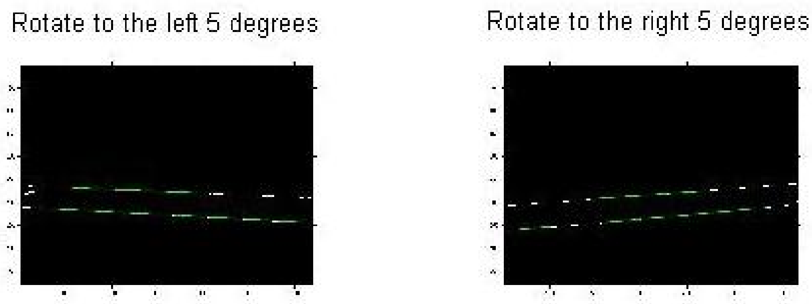

Figure 5.5: Processed images with rotation angles

θ given by Matlab function Hough is the angle of the perpendicular projection from the origin to the line measured in degrees clockwise from the positive x-axis. The range of θ is $-90^\circ \leq \theta < 90^\circ$. The angle of the line itself is $\theta + 90^\circ$. The values of θ given by the Hough transform are rounded up angles. In the simulation the rotation angles are floating-point numbers. The largest value of $\theta + 90^\circ$ the robot has encountered is 30 degrees. Thus we can construct a lookup table treshholding the rotation angles from 0 degree to 30 degrees by taking images off line.

Table 5.1: Rotation angle versus $\theta + 90°$

Actual Rotation Angle (degrees)	$\theta + 90°$ (degrees)
[0, 1.5]	0
(1.5, 3.5]	1
(3.5, 5.5]	2
(5.5, 7.5]	3
(7.5, 9.5]	4
(9.5, 11.5]	5
(11.5, 13.5]	6
(13.5, 15.5]	7
(15.5, 17.5]	8
(17.5, 19.5]	9
(19.5, 21.5]	10
(21.5, 23.5]	11
(23.5, 25.5]	12
(25.5, 27.5]	13
(27.5, 29.5]	14
(29.5, 30]	15
(0, -1.5]	180
(-1.5, -3.5]	179
(-3.5, -5.5]	178
(-5.5, -7.5]	177
(-7.5, -9.5]	176
(-9.5, -11.5]	175
(-11.5, -13.5]	174
(-13.5, -15.5]	173
(-15.5, -17.5]	172
(-17.5, -19.5]	171
(-19.5, -21.5]	170
(-21.5, -23.5]	169
(-23.5, -25.5]	168
(-25.5, -27.5]	167
(-27.5, -29.5]	166
(-29.5, -30]	165

Negative angle indicates rotating to the right.

Distance to wall with rotation angles

In most cases the robot heading with respect to the wall has a rotation angle as shown in figure 5.6.

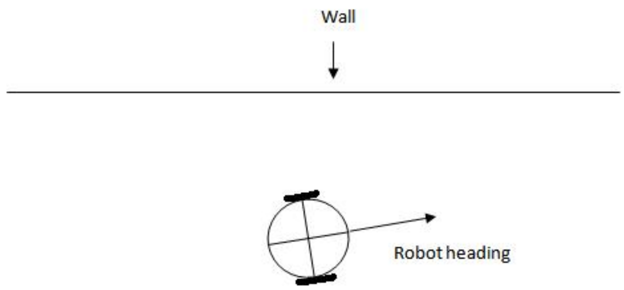

Figure 5.6: Robot heading with rotation angles

When there is rotation angle, the ρ given by the Hough transform is not the corresponding distance of the robot center to the wall. An image taken in this case is shown in figure 5.7.

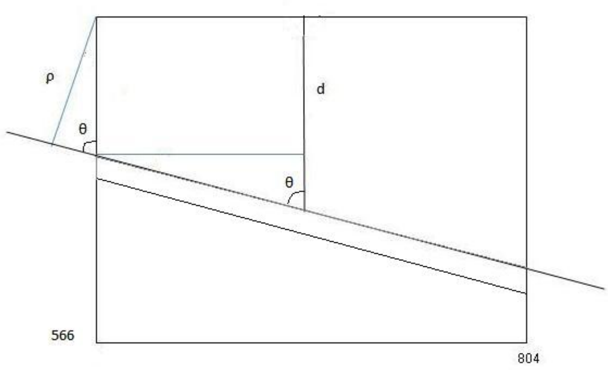

Figure 5.7: Distance to wall with rotation angles

This distance should be the distance from the middle of the line to the top of the image which is d in figure 5.6. The post-processed images are 804 x 566 pixels. The calculation needed to get d is

$$d = \|\rho/sin\theta - (804/2)/tan\theta\| \tag{5.1}$$

A calculation is performed to relate d to the actual distance between the robot center and the wall. The calculation is done using the equation below:

$$distance_{calculated} = (566 - d)/3.445 \tag{5.2}$$

5.2 Results

A series of tests was performed using the wall-following algorithm. The results of the tests are presented and discussed in this section. Each test was run for a duration of 100 time steps at normalized wheel speed. In real time, the robot moves at 67 cm/s. Correspondingly the robot samples at every 67 cm and at the end of 100 steps it will cover a distance of 67 m. Each test result shown below is a pair of plots. Each plot consists of the test results of 20 runs. The first plot of the pair is wall following behavior. The output y is normalized distance from the wall which corresponds to 84 cm in the target environment. The output x is the simulation time step, which correlates to the distance traveled parallel to the wall. The second plot of the pair is robot rotation angle error in degrees. The output y is the rotation angle error. The output x is the simulation time step. σ_Q^2 is the process noise variance and σ_R^2 is the sensor noise variance. The radius of the robot base is 8 inches (20.32 cm) which is normalized to 0.24 in the simulation. The threshold distance of robot center to wall is set to be 0.3, below which is considered too close to the wall.

5.2.1 No process noise, no sensor noise

First the experiments were performed under ideal conditions (no noise). The results are shown in figure 5.8.

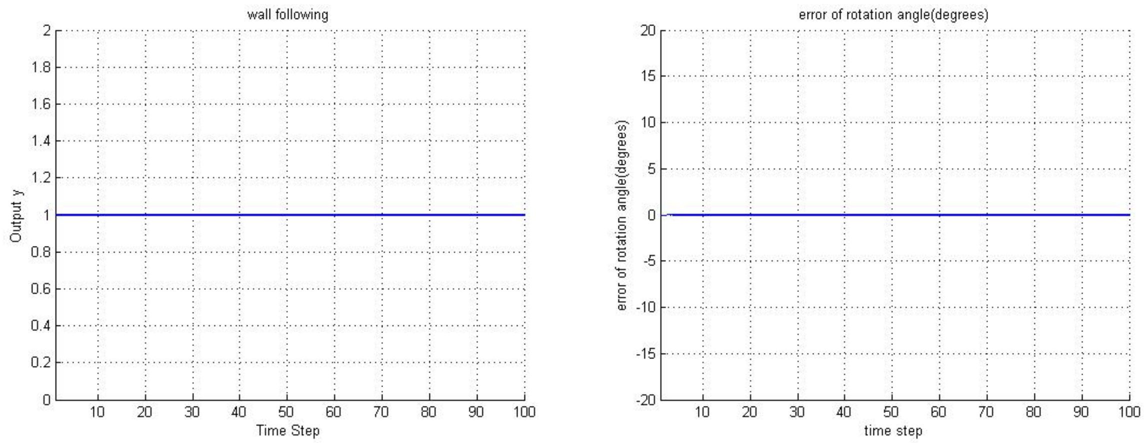

Figure 5.8: $\sigma_Q^2 = 0$, $\sigma_R^2 = 0$

The robot was able to follow the wall with nomalized distance error less than 0.0001 and rotation angle error less than 0.0001 degrees. The system settling time was found to be 0 time steps.

5.2.2 With process noise, no sensor noise

The experiments were then performed with no sensor noise but with increasing process noise. The results are shown from figure 5.9 to 5.11.

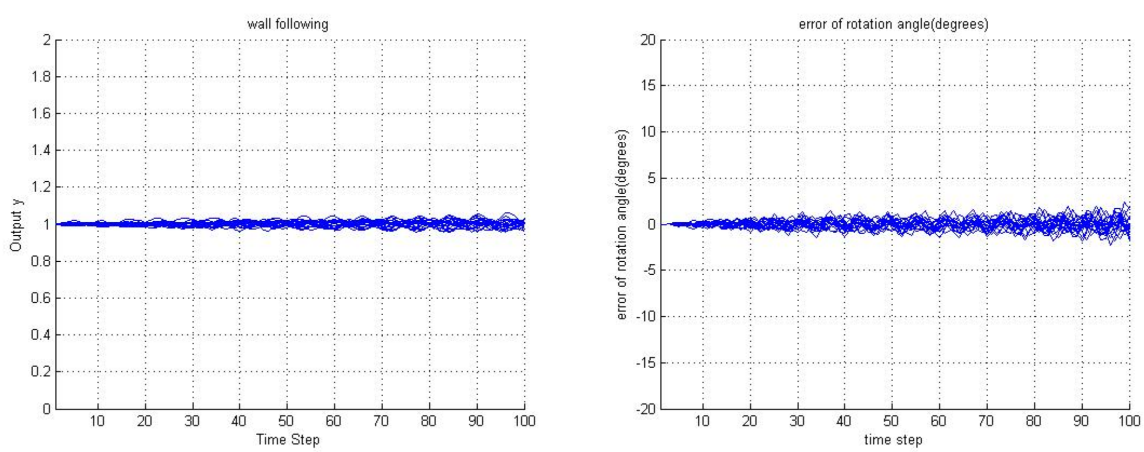

Figure 5.9: $\sigma_Q^2 = 0.001$, $\sigma_R^2 = 0$

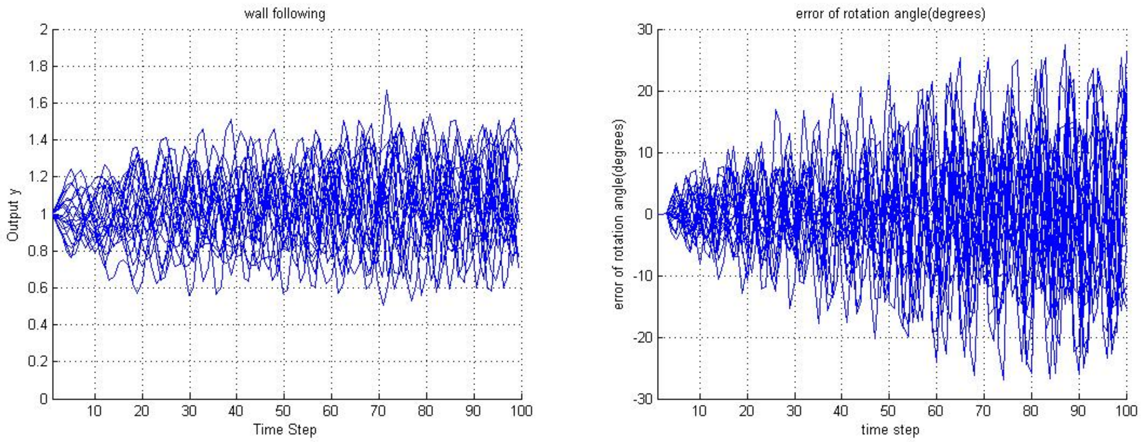

Figure 5.10: $\sigma_Q^2 = 0.015$, $\sigma_R^2 = 0$

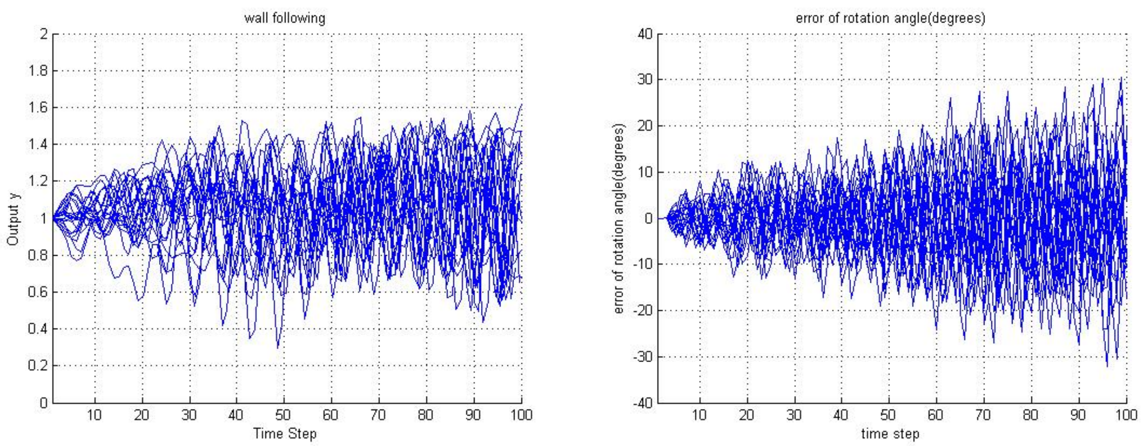

Figure 5.11: $\sigma_Q^2 = 0.016$, $\sigma_R^2 = 0$

When σ_Q^2 was increased to 0.016 there were unsuccessful cases that the distance between robot center and the wall was smaller than the threshold 0.3.

5.2.3 With process noise, with sensor noise

In this case σ_Q^2 is kept at 0.015 and σ_R^2 is increased slowly. The results are shown from figure 5.12 to 5.13.

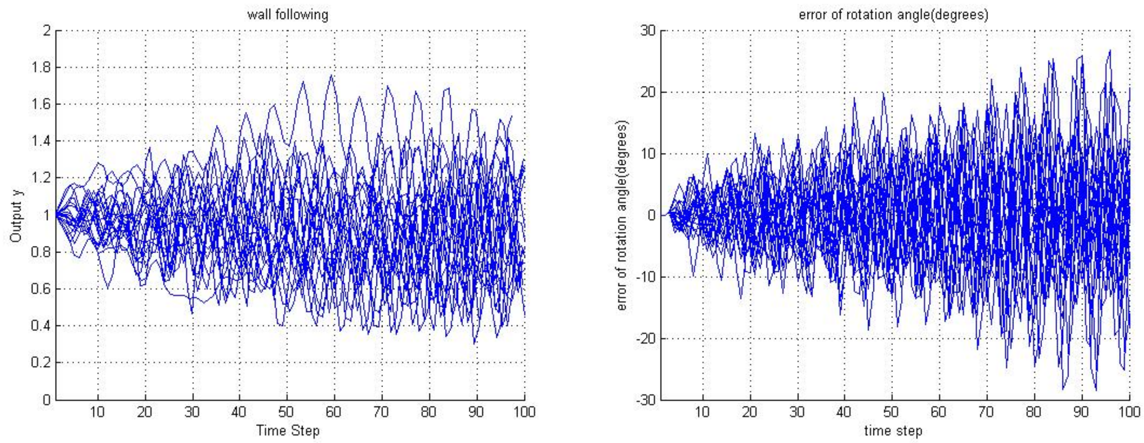

Figure 5.12: $\sigma_Q^2 = 0.015$, $\sigma_R^2 = 0.003$

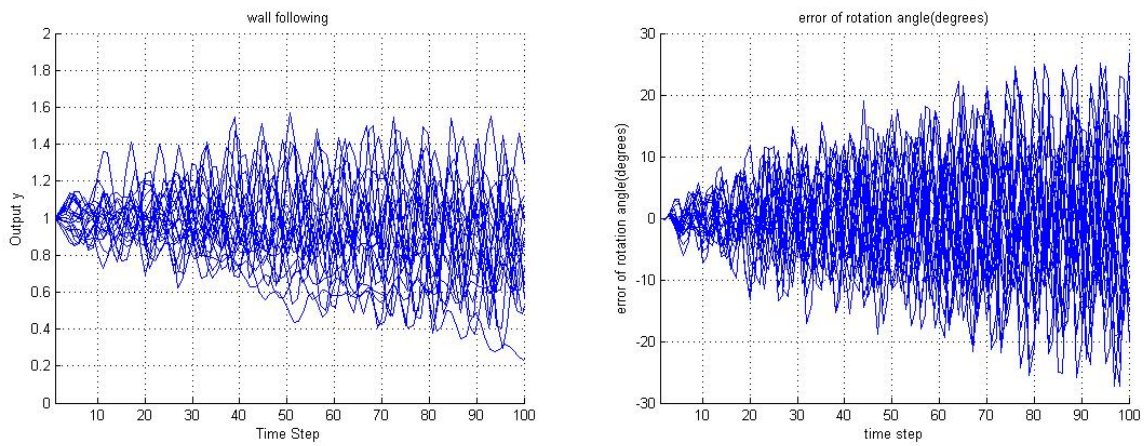

Figure 5.13: $\sigma_Q^2 = 0.015$, $\sigma_R^2 = 0.004$

When σ_R^2 was increased to 0.004 there were unsuccessful cases that the distance between robot center and the wall was smaller than 0.3.

5.2.4 No process noise, with sensor noise

The experiments were then performed with no process noise but with increasing sensor noise. The results are shown from figure 5.14 to 5.16.

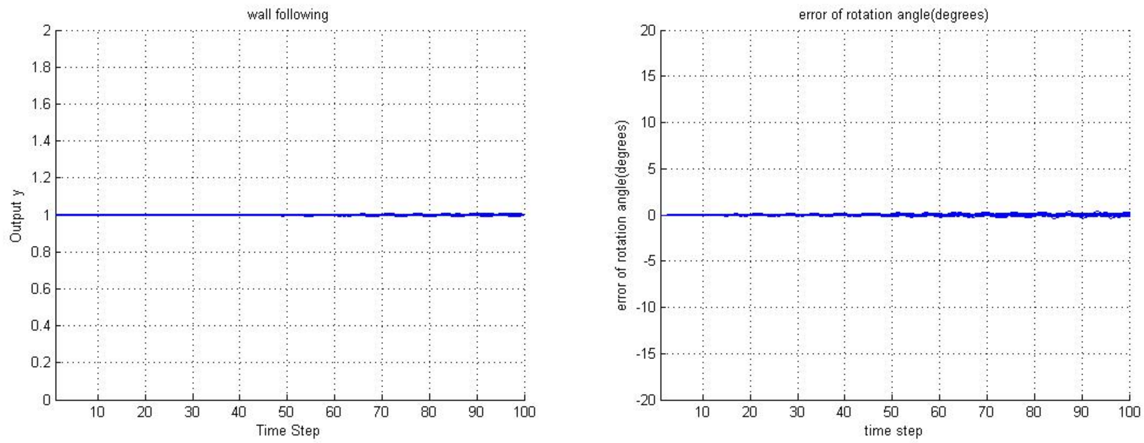

Figure 5.14: $\sigma_Q^2 = 0$, $\sigma_R^2 = 0.001$

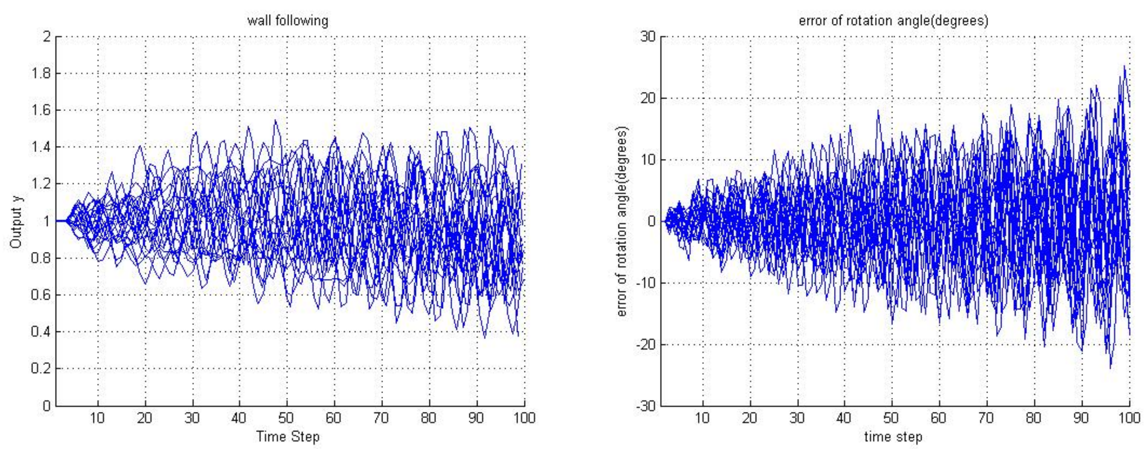

Figure 5.15: $\sigma_Q^2 = 0$, $\sigma_R^2 = 0.08$

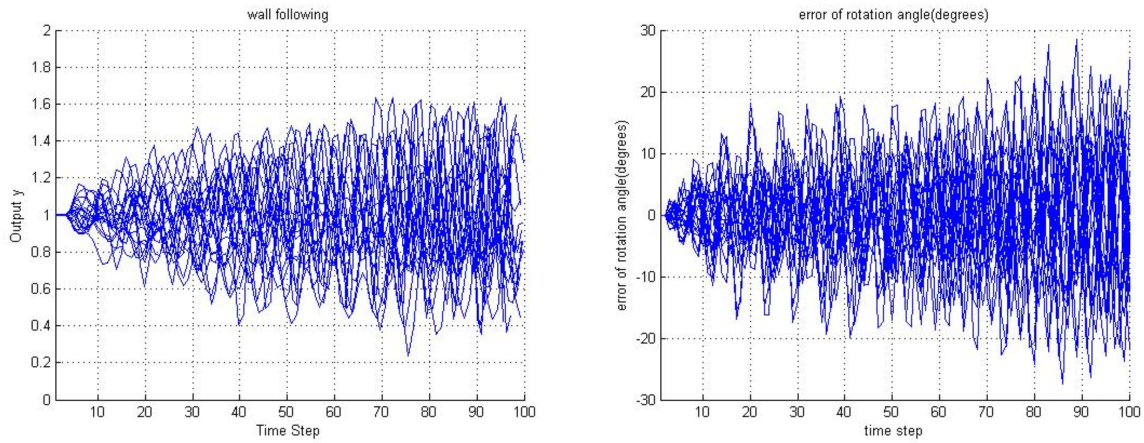

Figure 5.16: $\sigma_Q^2 = 0$, $\sigma_R^2 = 0.09$

When σ_R^2 was increased to 0.09 there were unsuccessful cases that the distance between robot center and the wall was smaller than 0.3.

5.2.5 With process noise, with sensor noise

In this case σ_R^2 is kept at 0.08 and σ_Q^2 is increased slowly. The results are shown from figure 5.17 to 5.18.

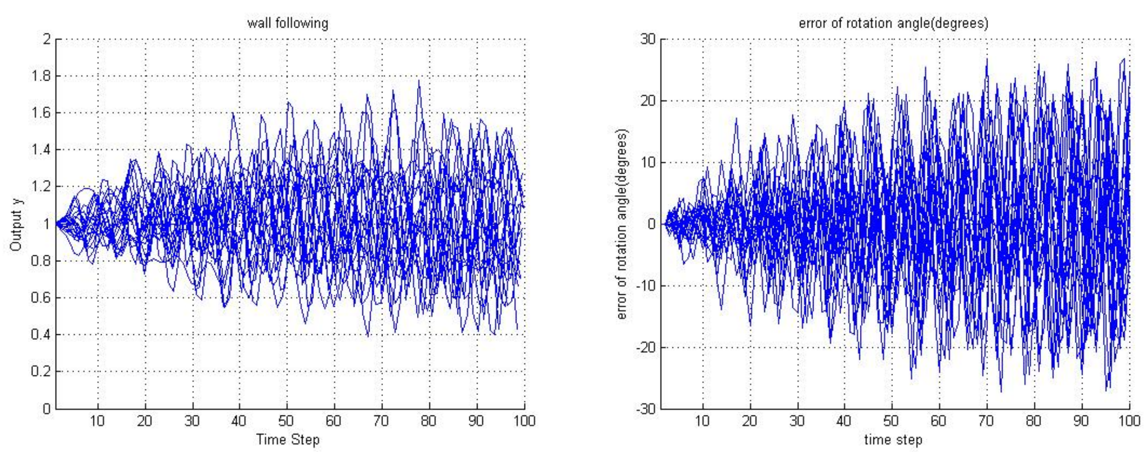

Figure 5.17: $\sigma_Q^2 = 0.009$, $\sigma_R^2 = 0.08$

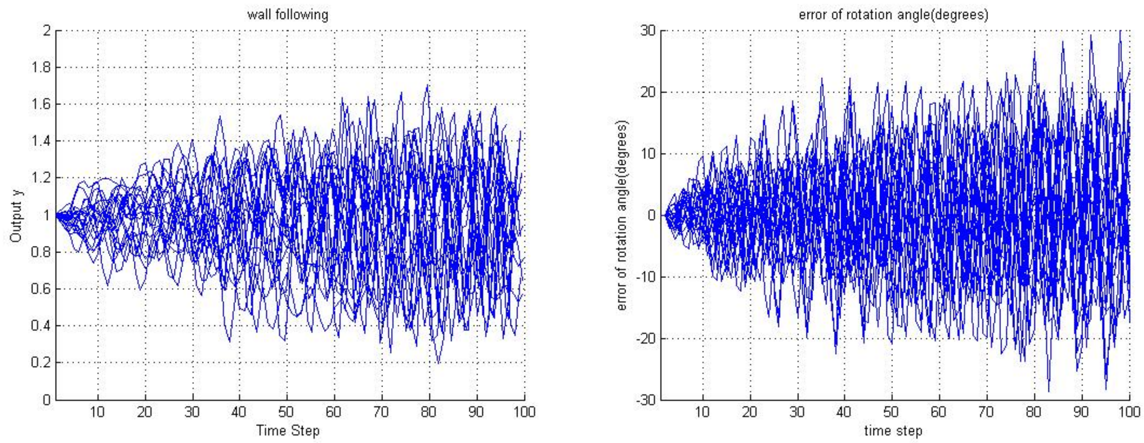

Figure 5.18: $\sigma_Q^2 = 0.01$, $\sigma_R^2 = 0.08$

When σ_Q^2 was increased to 0.01 there were unsuccessful cases that the distance between robot center and the wall was smaller than 0.3.

Chapter 6

Conclusion

This chapter will consist of two sections. The first section will summarize the results. The second section will list possible modifications for future work.

6.1 Summary

In this thesis, it was found that under ideal conditions (no noise), the robot was able to follow the wall with normalized distance error less than 0.0001 and rotation angle error less than 0.0001 degrees. The system settling time was found to be 0 time steps. When only process noise was added, with variance σ_Q^2 higher than 0.015 there were runs that the normalized distance was smaller than 0.3. When σ_Q^2 was kept at 0.015 and sensor noise was then added, there were runs that the normalized distance was smaller than 0.3 when σ_R^2 was bigger than 0.003. When only sensor noise was added, with variance σ_R^2 higher than 0.08, there were runs that the normalized distance was smaller than 0.3. When σ_R^2 was kept at 0.08 and process noise was then added, there were runs that the normalized distance was smaller than 0.3 when σ_Q^2 was bigger than 0.009.

6.2 Future work

A variety of future work can be generated from this algorithm.

6.2.1 Turning at corner

This algorithm is focused on the simpliest case in robot navigation–following a wall in a straight line. In the target environment, the walls are not endless. The robot needs to be able to make a turn at dead ends or corners. This can be done in a similar fashion as the wall-following algorithm by using digital image processing to get the location information of the robot.

6.2.2 Localization

Another distinct feature of the target environment is that all the doors have gray colored door frames. If we use digital image processing edge detection to recogonize door frames and give the robot knowledge of the enviroment, by counting how many door frames the robot has encountered it can then localize itself.

6.2.3 Real-time image processing

For this work, the images were captured and processed offline. Ultimately, it would be desirable to perform image processing in real time.

Bibliography

[1] Chris Wilson, *Hardware Testbed for Collaborative Robotics Using Wireless Communication*. 2009.

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, *Principles of Robot Motion*. The MIT Press, 2005.

[3] Latombe J., *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[4] P. van Turennout, G. Honderd, L.J. van Schelven, *Wall-following Control of a Mobile Robot*. 1992.

[5] Francisco Bonin-Font, Alberto Ortiz, Gabriel Oliver, *Visual Navigation for Mobile Robots: A Survey*. 2008.

[6] Agin,G.J. *Real Time Control of a Robot with a Mobile Camera*. Technical Note 179, SRI International, Feb. 1979.

[7] S.A.Hutchinson, G.D.Hager, P.I.Corke *A tutorial on visual servo control*. IEEE Trans. Robot. Automat, 12(5):651-670, Oct.1996.

[8] F. Chaumette, S. Hutchinson, *Visual Servo Control, Part I: Basic Approaches*. IEEE Robotics and Automation Magazine, 13(4):82-90, December 2006.

[9] F. Chaumette, S. Hutchinson, *Visual Servo Control, Part II: Advanced Approaches*. IEEE Robotics and Automation Magazine, 14(1):109-118, March 2007.

[10] A. C. Sanderson and L.E. Weiss, *Adaptive visual servo control of robots*. In A. Pugh, editor, Robot Vision, Pages 107-116. IFS, 1983.

[11] F. Chaumette *Potential problems of stability and convergence in image-based and position-based visual servoing*. In D. Kriegman, G. Hager, and S. Morse, editors, The confluence of vision and control, volume 237 of Lecture Notes in Control and Information Sciences, pages 66-78, Springer-Verlag, 1998.

[12] P.Corke and S.A.Hutchinson, *A new partitioned approach to image-based visual servo control*. IEEE Trans. Robot.Autom., 17(4):507-515, Aug.2001.

[13] Sugihara K, *Some location problems for robot navigation using a single camera*. Comput. Vis. Graph. Image Process, 42,112-129, 1988.

[14] Thrun S., *Probabilistic algorithms in robotics.* Technical Report CMU-CS-00-126, Carnegie Mellon University, 2000.

[15] Dellaert F., Fox D., Burgard W., Thrun S., *Monte Carlo localization for mobile robots.* In Proc. of IEEE Int'l Conference on Robotics and Automation(ICRA), pp.1322-1328, 1999.

[16] Matthies L. Shafer,S.A., *Error modeling in stereo navigation.* IEEE J. Robot. Autom., 3(3),239-248, 1987

[17] Tsubouchi T., Yuta S., *Map-assisted vision system of mobile robots for reckoning in a building environment.* In Proc. of IEEE Int'l Conf. on Robotics and Automation(ICRA), pp.1978-1984, 1987.

[18] Christensen H.I., Kirkeby, N.O, Kristensen S., Knudsen L., *Model-driven vision for indoor navigation.* Robot. Auton. Syst.12, 199-207, 1994.

[19] Moravec H.P., *The Stanford cart and the CMU rover.* In Proceedings of IEEE, vol.71, pp.872-884, 1983.

[20] Thorpe C., *FIDO, vision and navigation for a mobile robot.* In PhD dissertation, Dept Computer Science, Carnegie Mellon University, 1983.

[21] Thrun S., *Learning metric-topological maps for indoor mobile robot navigation.* Artif.Intell, 99(1),21-71, 1998.

[22] Horn, B.K.P., Schunck, B.G., *Determining optical flow.* Artif.Intell, 17(1-3), 185-203, 1981.

[23] Lucas B.D., Kanade T., *An iterative image registration technique with an application to stereo vision.* In Proc. of DARPA Image Understanding Workshop, pp.121-130, 1981.

[24] Matsumoto Y., Inaba M., Inoue H., *Visual navigation using view sequenced route representation.* In Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA), vol.1, pp.83-88, 1996.

[25] Jones S.D., Andresen D., Crowley J.L., *Appearance based process for visual navigation.* In Proc. of IEEE Int'l Conf. on Intelligent Robots and Systems(IROS), pp.551-557, 1997.

[26] Durrant-Whyte, H.F. and J.J. Leonard, *Navigation by Correlating Geometric Data.* Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems(IROS), Tsukuba,Japan, pp.440-447, 1989.

[27] Habib, M.K. and S. Yuta, *Efficient On-line Path Planning Algorithm and Navigation for a Mobile Robot.* Int. Journal of Electronics, Vol.69, No.2, pp. 187-210, 1990.

[28] E.Ivanjko, I.Petrovic, *Enxtended Kalman Filter based Mobile Robot Pose Tracking using Occupancy Grid Maps.* IEEE Melecon, Dubrovnik, Croatia, May 12-15, 2004.

[29] P.Goel, S.I.Roumeliotis, G. Sukhatme *Robust localization using relative and absolute position estimates*. In proc. of IEEE/RSJ International Conf. on IROS, 1999.

[30] Gonzales, R.C., *Processamento de Imagens Digitais*. Edgard Blucher, 2000.

[31] Shapiro, Linda and Stockman, George, *Computer Vision*. Prentice-Hall, Inc., 2001.

[32] Santana A.M.,de Souza, A.A.S.,Britto R.S.,Alsina P.J.,Medeiros A.A.D., *Localization of a Mobile Robot Based on Odometry and Natural Landmarks Using Extended Kalman Filter*. International Conference on Informatics in Control, Automation and Robotics, Funchal, Madeira, Portugal, 08,2008.

[33] Iraj Hassanzadeh,Mehdi Abedinpour Fallah, *Design of Augmented Extended Kalman Filter for Real Time Simulation of Mobile Robots Using Simulink*. Proceeding of the 6th International Symposium on Mechatronics and Applications(ISMA09), Sharjah, UAE, March 24-26,2009.

[34] Edouard Ivanjko, Andreja Kitanov, Ivan Petrovic, *Model Based kalman Filter Mobile Robot Self-Localization*. Robot Localization and Map Building, Hanafiah Yussof(Ed.) ISBN:987-953-7619-83-1, 2010.

[35] J.Borenstein, H.R.Everett, L.Feng, *Where am I? Sensors and Methods for Mobile Robot Positioning*. Ann Arbor, MI 48109, University of Michigan, 1996.

[36] P.Goel, S.I.Roumeliotis, G. Sukhatme, *Robust localization using relative and absolute position estimates*. In proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), 1999.

[37] J.Borenstein, L.Feng, *Measurement and correction of systematic odometry errors in mobile robots*. IEEE Transactions in Robotics and Automation, vol. 12, 1996.

[38] N.Roy and S. Thrun, *Online self-calibration for mobile robots*. in In Proceedings of the IEEE International Conference on Robotics and Automation(ICRA), 1999.

[39] K.Konolige, *Markov localization using correlation*. in International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 1999.

[40] G.Dudek, M.Jenkin, *Computational Principles of Mobile Robotics*. Trumpington Street, Cambridge CB2 1RP: Cambridge University Press, 2000.

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

[42] Jacobs,O.L.R., *Introduction to Control Theory, 2nd Edition*. Oxford University Press, 1993.

[43] Kalman,R.E., *A New Approach to Linear Filtering and Prediction Problems*. Transaction of the ASME, Journal of Basic Engineering, pp. 35-45, March, 1960.

[44] Lewis,Richard, *Optimal Estimation with an Introduction to Stochastic Control Theory.* John Wiley & Sons, Inc. 1986.

[45] Maybeck,Peter S., *Stochastic Models, Estimation, and Control, Volume 1.* Academic Press, Inc., 1979.

[46] Sorenson,H.W., *Least-Squares estimation:from Gauss to Kalman.* IEEE Spectrum, vol.7, pp.63-68, July, 1970.

[47] Duda,R.O.,P.E.Hart, *Use of the Hough Transformation to Detect Lines and Curves in Pictures.* Comm. ACM, Vol 15, pp. 11-15, Jan,1972.

[48] E.Malis, F. Chaumette and S. Boudet, *2.5 D visual servoing.* IEEE Transactions on Robotics and Automation, 15(2):238-250, 1999