FEDERAL UNIVERSITY OF CEARÁ

CENTER OF TECHNOLOGY

ELECTRICAL ENGINEERING DEPARTMENT

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING

MASTER IN ELECTRIC ENGINEERING

MARCUS DAVI DO NASCIMENTO FORTE

REFERENCE TRAJECTORY TRACKING CONTROL OF A NONHOLONOMIC MOBILE ROBOT WITH INERTIAL SENSOR FUSION

FORTALEZA

2018

MARCUS DAVI DO NASCIMENTO FORTE

REFERENCE TRAJECTORY TRACKING CONTROL OF A NONHOLONOMIC MOBILE ROBOT WITH INERTIAL SENSOR FUSION

Master's thesis presented to the Graduate Program in Electrical Engineering from Federal University of Ceará, as part of the requisites to obtain the Master's degree in Electrical Engineering. Concentration Area: Electrical Engineering

Supervisor: Prof. Dr. Fabrício Gonzalez Nogueira

Co-Supervisor: Prof. Dr. Wilkley Bezerra Correia

FORTALEZA

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F842r Forte, Marcus Davi do Nascimento.
REFERENCE TRAJECTORY TRACKING CONTROL OF A NONHOLONOMIC MOBILE ROBOT WITH INERTIAL SENSOR FUSION / Marcus Davi do Nascimento Forte. – 2018.
88 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2018.
Orientação: Prof. Dr. Fabrício Gonzalez Nogueira.
Coorientação: Prof. Dr. Wilkley Bezerra Correia.

1. Robô Móvel. 2. Fusão Sensorial. 3. Controle LQR. 4. Filtro de Kalman. I. Título.

CDD 621.3

MARCUS DAVI DO NASCIMENTO FORTE

REFERENCE TRAJECTORY TRACKING CONTROL OF A NONHOLONOMIC MOBILE ROBOT WITH INERTIAL SENSOR FUSION

Master's thesis presented to the Graduate Program in Electrical Engineering from Federal University of Ceará, as part of the requisites to obtain the Master's degree in Electrical Engineering. Concentration Area: Electrical Engineering

Approved on: July 31, 2018

EXAMINATION BOARD

Prof. Dr. Fabrício Gonzalez Nogueira (Supervisor)
Federal University of Ceará (UFC)

Prof. Dr. Wilkley Bezerra Correia (Co-Supervisor)
Federal University of Ceará (UFC)

Prof. Dr. Bismark Claure Torrico
Federal University of Ceará (UFC)

Prof. Dr. Marcus Vinicius Silvério Costa
Federal University of Rural do Semi-Árido
(UFERSA)

To God,

to my parents, Marcus and Rosana,

To my beloved, Larissa

ACKNOWLEDGEMENTS

I thank God for blessing me with numerous opportunities and the passion to study.

I thank my parents for providing me with everything I need for accomplishing my dreams and give me such great motivation and life example.

I thank my beloved Larissa for supporting me in happy and difficult times.

I thank my supervisor, Fabrício and co-supervisor, Wilkley for providing me exceptional knowledge and expertise that gave me more than enough motivation for this thesis.

I thank my friends from GPAR for sharing their knowledge, giving me counsel and friendship.

I thank my brothers and sisters from Church for the great care and support they have for me.

Tudo o que fizerem, façam de todo o coração,
como para o Senhor, e não para os homens.

(Colossenses 3:23)

ABSTRACT

This work presents a study on differential drive wheeled mobile robots regarding its localization estimation using sensor fusion techniques and control over a reference trajectory. The robot's posture is an extremely important variable to be estimated, specially for autonomous mobile robots as it has to drive along a path without manual intervention. Posture estimation provided by individual sensors has shown to be inaccurate or straightforward mismatched, leading to a need of data fusion from different sources. Sensors such as accelerometer, gyroscope and magnetometer each have its own intrinsic constructive limitations. However, by combining all their flaws into a linear model allows optimal estimators, such as Kalman Filter (KF), to be used and produce estimates close to real life behavior. For the trajectory tracking and disturbance rejection, a linear control strategy for a linearized mobile robot model is applied. The system is modeled with error states to be carried out by a Linear Quadratic Regulator (LQR) controller along with a feedforward reference control action so that the reference trajectory is accordingly tracked.

Keywords: Mobile Robot. Sensor Fusion. LQR Control. Kalman Filter.

RESUMO

Este trabalho apresenta um estudo a respeito de robôs móveis sobre rodas com tração diferencial, com ênfase na estimação da localização do robô através de técnicas de fusão sensorial e no controle sobre trajetórias de referência. A posição do robô é uma variável de extrema importância a ser estimada, especialmente para robôs autônomos, uma vez que este deve se dirigir sobre uma trajetória sem intervenção manual. A estimação fornecida por sensores individualmente mostra-se imprecisa ou até completamente errônea, fazendo-se necessário aplicar fusão de dados de diferentes fontes. Sensores como acelerômetro, giroscópio e magnetómetro possuem limitações construtivas inerentes à natureza do sensor. Contudo, combinando-se as falhas de cada sensor em um modelo linear permite o uso de estimadores ótimos, tal como filtro de Kalman, para estimação próxima a valores reais. Para o controle da trajetória e rejeição de distúrbios, um controlador linear aplicado sobre o modelo linearizado do robô. O sistema é modelado considerando erros como estados para serem controlador por um Regulador Linear Quadrático, juntamente com uma ação de alimentação direta de tal forma a garantir o seguimento de uma trajetória de referência.

Palavras-chave: Robô Móvel. Fusão Sensorial. Controle LQR. Filtro de Kalman.

LIST OF FIGURES

LIST OF ACRONYMS

EKF	Extended Kalman Filter
GPS	Global Positioning System
IMU	Inertial Measuring Unit
INS	Inertial Navigation System
KF	Kalman Filter
LIDAR	Light Detection and Ranging
LPV	Linear Parameter Varying
LQR	Linear Quadratic Regulator
LSE	Least Square Error
MCU	Microcontroller Unit
MEMS	Microelectromechanical System
MIMO	Multiple Input Multiple Output
NWMR	Nonholonomic Wheeled Mobile Robot
PF	Particle Filter
SISO	Single Input Single Output
SLAM	Self Localization and Mapping
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
WMR	Wheeled Mobile Robot
ZVU	Zero Velocity Update

LIST OF SYMBOLS

ω	Angular velocity
v	Linear velocity
ω_r	Feedforward Angular velocity
v_r	Feedforward Linear velocity
e	Mobile robot body frame error state
$\bar{e}$	Mobile robot body frame error linearized state
x, x_c	Robot Navigational X Component
y, y_c	Robot Navigational Y Component
J	Joint Space Transformation Matrix
L	Distance of Motorized Wheels
r	Wheel Radius
$R(.)$	Rotation Matrix
A, B, C, D	State-Space Matrices
K	Linear Controller Gain
$K(t)$	Non-linear Controller Gain
ξ	Non-linear Controller Damping
ω_n	Non-linear Controller Natural Frequency
Q	LQR State Weight Matrix
R	LQR Control Weight Matrix
ϕ	(Robot) Yaw Angle
ψ	Roll Angle
θ	Pitch Angle
M_v	Three Dimensional Vector Rotation
M_f	Three Dimensional Frame Rotation
a	Acceleration
a_m	Measured Frame Acceleration

a^b	Body Frame Acceleration
a^n	Navigational Frame Acceleration
b^x	Bias of device x
v^x	Noise of device x
ω_m	Measured Angular Velocity
B_m	Measured Magnetic Field
A_k, B_k, C_k	Discrete State-Space Matrices
Q_k	Kalman Filter Process Covariance Matrix
R_k	Kalman Filter Measurement Covariance Matrix
P_k	Error Covariance Matrix
K_k	Kalman Filter Gain Matrix

CONTENTS

1 INTRODUCTION

Mobile robot control is a highly regarded theory used in any mobile robot related research, not being exclusive to the control theorists. This is due to the fact the other areas, such as artificial intelligence or computer vision require some way to validate their studies when applied to mobile robotics, which cannot be done without controlling the mobile robot along some reference trajectory or path. One might mention the usage of a neural network to find the shortest route to an objective. Another example is the employment of a camera to identify and avoid obstacles along a pre-programmed path or even create a path based on the camera imaging.

There are two main approaches for controlling robots in general: Kinematic and Dynamic. Dynamic modeling consists of describing motion while also knowing the cause of the forces that cause the motion. It considers the robot's mass, attrition, inertial moments, and so on. Kinematic modeling cares not for such causes, being interested only in the motion itself. It considers.

For trajectory control purposes, both approaches are viable alternatives for designing a control structure. However, since dynamic modeling requires the knowledge of physical properties parameters like mass or inertial moment , which may be subjected to modeling uncertainty, the kinematic modeling is simpler and most used.While not considering mobile robot physical parameters directly, the kinematic model may include some dynamic characteristics, for instance motor behavior, that can be contemplated in the control design.

As mentioned, kinematic modeling deals with motion with no regards to its cause. In mathematical terms, it can be described as a function, $p = f(q)$, where q is the joint space and p is the robot task space. This function is referred to as *Direct Kinematics*. The opposite way, $q = f^{-1}(p)$ is known as *Inverse Kinematics*. Normally, the task space p, commonly the Cartesian coordinates, is used to define the mobile robot coordinates, and the joint space q (velocities, for instance) defines which variables lead to p. Finally, the function f defines mathematically the way q turns into p. In general, direct kinematics is used to determine the *end-effect* coordinates when q is applied, and inverse kinematics is used to know what q should be to a given *end-effect*. In terms of control theory, the inverse kinematic is adopted to devise a control law q such that the robot moves to the desired task space p given a f^{-1}.

1.1 Robot History

The word “Robot” is believed to be first used in 1921 by the Czech novelist Karel Capec for his play, “Rossum’s Universal Robots”, in which the so called “Robot” would take over the world. The play conceived the idea that in the near future all workers would be fully automated and would eventually obtain self-awareness. The play became extremely popular, which contributed to the dissemination of the word (NEHMZOW, 2003). The concept idea of the word “Robot” remains to this day, being defined as an autonomous machine capable of executing complex tasks based on its programming.

There are several different types of robots, each designed to perform specific set of tasks in a certain environment. The most important ones may be listed as:

1. Manipulator Robots
2. Wheeled Mobile Robot (WMR)
3. Legged Mobile Robots
4. Underwater Robots
5. Aerial Robots

Any robot possess the same basic component structure: a programmable device (computer), sensors and actuators. The differences lie on the application the robot is subjected to.

Manipulator Robots were among the first kind to be used, mostly in industry. The main usage of this type of robot is to replace human task in the repetitive or strength-demanding movement required in industrial facilities. The always operate within a bounded, or non-movable, work space. Another useful application is the manual/remote operation of a robotic arm to perform precise movement (Medical) or object manipulation in hazardous areas (Space).

Wheeled robots appeared in the 1950s. One of them being the focus of the researcher Grey Walter who was able to conceive a robot with only a few sensors, motors and two vacuum tube analog computed. He demonstrated that even by using simple components, the mobile robots could exhibit complex behaviors. He also argued that they had a tendency to explore their environment autonomously (NEHMZOW, 2003). Nowadays, mobile robots are present in numerous areas, such as autonomous navigation, mapping, cleaning, object manipulation, supervision, and many others.

Legged robots are similar to wheeled robots, the difference only being the movement actuator. There are a few settings that favors the usage of legged robots. A few, very specific applications require the robot to move along a path full of small ground obstacles, too tilted

or that has an irregular terrain for a wheeled robot to move on, requiring a more sophisticated approach.

Underwater robot are quite useful for underwater exploration, specially because humans cannot navigate without already requiring some equipment. Some of the latest underwater robots are even able to navigate on some deep floors of the ocean. Other applications are also mentioned, such as cleaning, surveillance, underwater machinery supervision and so on.

Aerial Robots are among the latest mobile robot technological advances, namely Unmanned Aerial Vehicle (UAV). They perform similar task to the aforementioned mobile robots, but have an extra degree of freedom and several other physical properties that has to be accounted for when designing such robots. Weight, lift, drag and thrust have to be mathematically modeled in order to well represent real life behavior. The popular Drones are multirotor UAVs capable of producing lift by rotating propellers. Complex control strategies have to be employed to maintain stable attitude (three dimensional orientation) and height. They have many applications in terrain mapping, topology, supervision, and others.

1.2 Sensor Fusion

Although the term *sensor fusion* is mostly used in combining physical sensory data, and *data fusion* might suggest combining any types of data (i.e image processing), there is no clear difference between the two terms in the literature, since both refer to the same concept idea. This thesis applies *sensor fusion* as it is more adequate.

The sensor fusion itself is a simple concept:to combine measurement data from sensors of different nature to produce a better estimate of the measurement as opposed to individual sensor readings. This technique is suitable for applications where multiple sensors are present and/or affected with some unwanted characteristics. There are many different methods for combining data from multiple sources into an estimated system state as seen in Raol (2009). Most of the time, stochastic properties inherent to the fusing sensors must be known. A popular sensor fusion technique considers to apply the Kalman Filter (KF), which is an optimal estimator in the sense that it minimizes the model's covariance error matrix. It is also a model based estimator, whose system dynamics have to be mathematically modeled. There are three main Kalman Filtering types: the original, linear KF from Kalman (1960); the Extended Kalman Filter (EKF), a modified KF for non-linear model dynamics, from Julier (1997) and Unscented Kalman Filter (UKF), an alternative to EKF less prone to instability, from Wan e Merwe (2000). There

are other modified versions of the KF adapted to specific applications (RAOL, 2009).

Sensor fusion is extremely important for autonomous mobile robots or other navigational platforms, as it provides accurate pose estimation on the robot's location. Many different sensors are known to be used in such applications, i.e accelerometer, gyroscopes, magnetometers, Global Positioning System (GPS), and so on. Some of the mentioned sensors have intrinsic constructive limitation, such as the gyroscope biased output. However, by combining gyroscope data with the output of a magnetometer, which is heavily noised but has no bias, one may remove both limitation by fusing the sensors' data. Also, fusion techniques are helpful to avoid cumulative error effect from *Dead Reckoning* computation of velocity and position of an object. Dead reckoning is the process of computing one's position or velocity by solely using previous position or velocity data, respectively. However, if the estimation of a given object's position is not quite correct, the erroneous estimation will still be used for computation of later position and so on, thus accumulating the error. Using KF greatly helps to correctly estimate desired variables so that dead reckoning drift is either mitigated or straightforwardly canceled.

1.3 Related Work

Mobile robot trajectory tracking control is the object of study since its conception in the late 1950s. Most works propose non-linear strategies for the problem.Klancar *et al.* (2005) uses a pole-placement method for designing a stable controller with a specified performance, while proposing a trajectory planning algorithm. The same pole-placement strategy is seen in Kanayama *et al.* (1990) originally. Other strategies, such as adaptive control Diniz (2016) and predictive control Ogawa (2014) have also been used for Nonholonomic Wheeled Mobile Robot (NWMR) control over reference trajectory. These works, however, do not mention how the pose of the mobile robot is acquired, an important information required in practical applications. Hence, sensor fusion techniques would be a good alternative for solving the pose estimation problem for mobile robots.

The are plenty of works involving mobile robots and sensor fusion strategies. Early works such as Chenavier e Crowley (1992) made use of EKF to combine camera data (vision) and odometry data for a pose estimation of a wheeled mobile robot. Latter on Barshan e Durrant-Whyte (1993) were among the first works to use inertial sensors in mobile robotics. Until then, inertial sensors were used only in expensive aerial or aerospace applications. After technological advances and the coming of cheaper integrated circuits, engineers started developing cheaper

inertial sensors embedded into a silicon chip using Microelectromechanical System (MEMS) technology.MEMS consists basically in constructing extremely small mechanical components whose motion produces some electrical signal in a transducer. The works of Kang *et al.* (1994) and Kam *et al.* (1997) provided a baseline for studying sensor fusion applied to mobile robots, the latter being a review work. In Liu e Pang (2001) an accelerometer is used to estimate a mobile robot position by double integration. The accelerometer biasing issue is discussed, as well as a method for rejecting it, allowing an integration of its data to estimate the mobile robot position. Although not a permanent solution, Liu e Pang (2001) provided ground work for using accelerometer-estimated pose.

More work is done with other robot platforms like robotic arms (JASSEMI-ZARGANI; NECSULESCU, 2002) or multipod robots (LIN *et al.*, 2006). Further work combines not only inertial data with odometry and includes active beacons as seen in Lee *et al.* (2009). The latter work exhibited good performance, but is not applicable for outdoor navigation. Majority of works found in literature apply sensor fusion to provide a good attitude estimation without much concern for an actual pose estimation. A solution to the outdoor localization is the use of GPS data to be considered in the fusion model. GPS data might be accurate but may introduce extreme delay or outright not work. Khatib *et al.* (2015) uses EKF to mix up encoder, IMU, digital compass and GPS data for pose estimation.

Finally, recent works contemplate computer vision theory and sensor fusion to combine data from a digital camera and inertial sensors for mobile robot localization and navigation, as seen in Alatise e Hancke (2017) and Silva e Wimalaratne (2017). Neural networks (LIMA *et al.*, 2017) and Particle Filter (PF) (XUE *et al.*, 2017) have also been explored. UKF have been used to fuse data for tracking a robotic arm (ATRSAEI *et al.*, 2018).

1.4 Objectives

It has been noticed that most works mentioned in the Section 1.3 and others will rarely handle mobile robot control, regarding only localization or pose estimation. On the other hand this work is meant to combine IMU data to estimate a NWMR pose while also providing a suitable linear controller to effectively reject disturbances perceived by the IMU. The main focus of this work are:

- Explore the NWMR mathematical model;
- Design a Linear Quadratic Regulator (LQR) controller for the NWMR;

- Analysis of an IMU;
- To design and implement the sensor fusion model;
- Validation data with simulations and experiments.

1.5 Outline

This work is organized as follows:

- Chapter 1 – An introduction and contextualization of the main topics of this thesis is discussed;
- Chapter 2 – The NWMR mathematical model is explored.
- Chapter 3 – The LQR controller is designed. Tuning and other control theory related topics are present as well;
- Chapter 4 – This Chapter presents an introduction to inertial navigation and presents the used sensors;
- Chapter 5 – In this Chapter the sensor fusion model is derived and discussed;
- Chapter 6 – Both simulation and experimental results. A small subsection is dedicated to the hardware specifications of the experimental robot;
- Chapter 7 – Conclusions about the obtained results and the discussed theory are brought, along with suggested future work.

2 MOBILE ROBOT MATHEMATICAL MODEL

This Chapter presents the kinematic equations that describe the NWMR motion. The goal is to find a state-space model with which one may devise a linear controller. A state-space model is preferred due to its intrinsic multivariable nature of the mobile robot as it has two velocities input and up to three outputs regarding its pose.

It should be clear that this work handles reference tracking objectives, instead of path following objectives. Reference tracking means that the robot has to be at a reference point at a certain time, and path following simply requires that the robot follows along a path without any time restriction. Each concept raises a different model. Recent path following works proposed strategies using *Guiding Vector Fields* (KAPITANYUK *et al.*, 2018) for nonholonomic mobile robots. For omnidirectional mobile robots (WANG *et al.*, 2016) the path following task is slightly different than its nonholonomic counterpart as it adds more degrees of freedom. Aguiar *et al.* (2004) provides a very good read on the subject. Aguiar *et al.* (2004) also mentions that the reference tracking problem is slightly more complex than path following because of time restriction.

2.1 Dynamic System

The nonholonomic mobile robot mathematical model is described in terms of forward kinematics, i.e. what inputs u to the system lead the robot to some pose p. Figure 1 shows the top-down vision of a NWMR whose pose is defined as the global/navigational (referred to axis X and Y) Cartesian coordinates including the robot's heading, composing the task space $p = [x, y, \phi]'$. Note that p has its Cartesian coordinates coincident with the robot's gravity center C. With simple trigonometry, one describes ϕ with ω and decompose the velocity vector v along X and Y as

$$\dot{x_c} = v.\cos\phi, \tag{2.1}$$

$$\dot{y_c} = v.\sin\phi, \tag{2.2}$$

$$\dot{\phi} = \omega \tag{2.3}$$

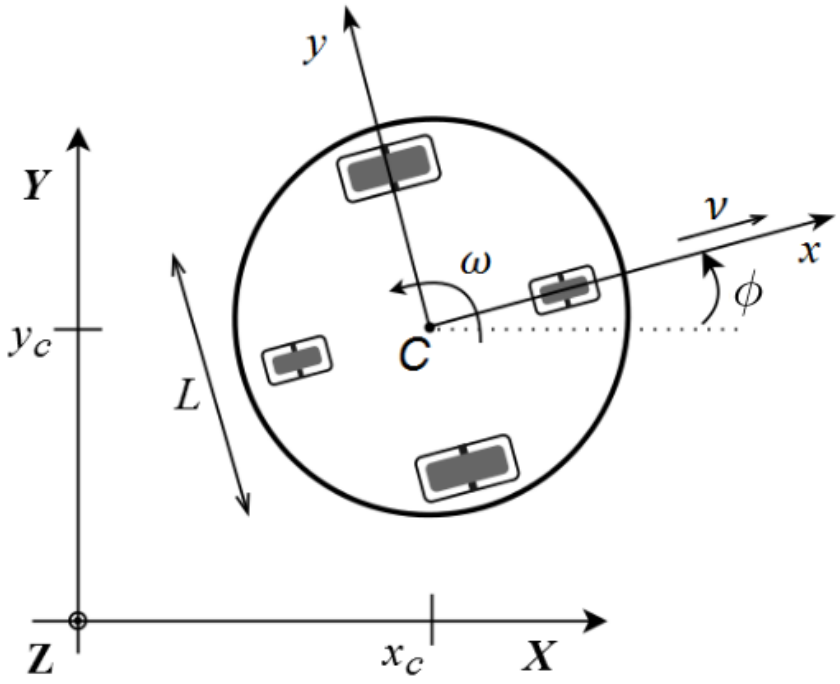
or, in matrix form,

$$\begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \rightarrow \dot{p} = J.q. \tag{2.4}$$

We also describe its discrete-time model with sampling time T_s (needed for digital, practical implementation)

$$\begin{bmatrix} x_c \\ y_c \\ \phi \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ \phi \end{bmatrix}_k + Ts \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}_k \rightarrow p_{k+1} = p_k + J.q \tag{2.5}$$

Figure 1 – Robot Architecture



Source: The author.

Consider the continuous model of Eq. (2.4). The joint space q is defined by the mobile robot velocities $q = [v\ \omega]'$. Now, expressing $v = \frac{v_r+v_l}{2}$, $\omega = \frac{v_r-v_l}{L}$ for $v_r = r.\dot{\phi}_r$ and $v_l = r.\dot{\phi}_l$, we write an alternative, equivalent form of Eq. (2.4) in terms of the wheel speeds $\left[\dot{\phi}_r\ \dot{\phi}_l\right]$ and their radius r

$$\begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} (r/2)\cos\phi & (r/2)\cos\phi \\ (r/2)\sin\phi & (r/2)\sin\phi \\ r/L & -r/L \end{bmatrix} \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix}. \tag{2.6}$$

We can explicitly depict the nonholonomic constraint by taking Eq. (2.1) and Eq. (2.2) and isolating v, leading to

$$\dot{x_c}\sin\phi = \dot{y_c}\cos\phi. \tag{2.7}$$

The equation (2.7) can be easily interpreted by applying intuitive ϕ, e. g., 0 or π for a better understanding of the nonholonomic constraint - the robot cannot move laterally with respect to its body frame.

Simple as it shows up, model in Eq. (2.4) is not suitable for a linear controller design because of the presence of nonlinear terms $cos(\phi)$ and $sin(\phi)$ so that it is not a state-space model of the form $\dot{x} = A.x + B.u$. An option for tracking a reference trajectory would be to apply the inverse kinematic method so that we write $q = f^{-1}(p)$, which would allow us to designate a control law q_r so that a reference p_r is achieved. By observing Eq. (2.4), an inverse f^{-1} of the space transformation matrix $f : q \to p$ cannot be calculated by inverting J because it is not a square matrix. This is not a problem, since we can apply the generalized inverse (TZAFESTAS, 2014). The generalized inverse $J^{\dagger}$ of J is given by

$$J^{\dagger} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.8}$$

By taking $q_r = J^{\dagger}.\dot{p_r}$ we express the control law

$$\begin{aligned} v_r &= \dot{x_r}\cos\phi_r + \dot{y_r}\sin\phi_r \\ \omega_r &= \dot{\phi}_r, \end{aligned} \tag{2.9}$$

or, equivalently,

$$\begin{aligned} v_r &= \pm\sqrt{\dot{x_r}^2 + \dot{y_r}^2} \\ \omega_r &= \frac{d}{dt}(\text{atan2}(\dot{y_r}, \dot{x_r}) + n\pi). \end{aligned} \tag{2.10}$$

Results from Eq. (2.10) may be readily reachable by examining Figure 1. The linear velocity vector v can be seen as a Pythagorean composition of the moving robot's center $\dot{x_c}$ along X and $\dot{y_c}$ along Y, with the $\pm$ indicating reverse or forward drive direction. The angular velocity ω is then the rate of change of the angle at each point according to the motion $(\dot{x_c}, \dot{y_c})$. The term $n\pi$ may also defines drive direction, with $n = 0$ for forward and $n = 1$ for reverse.

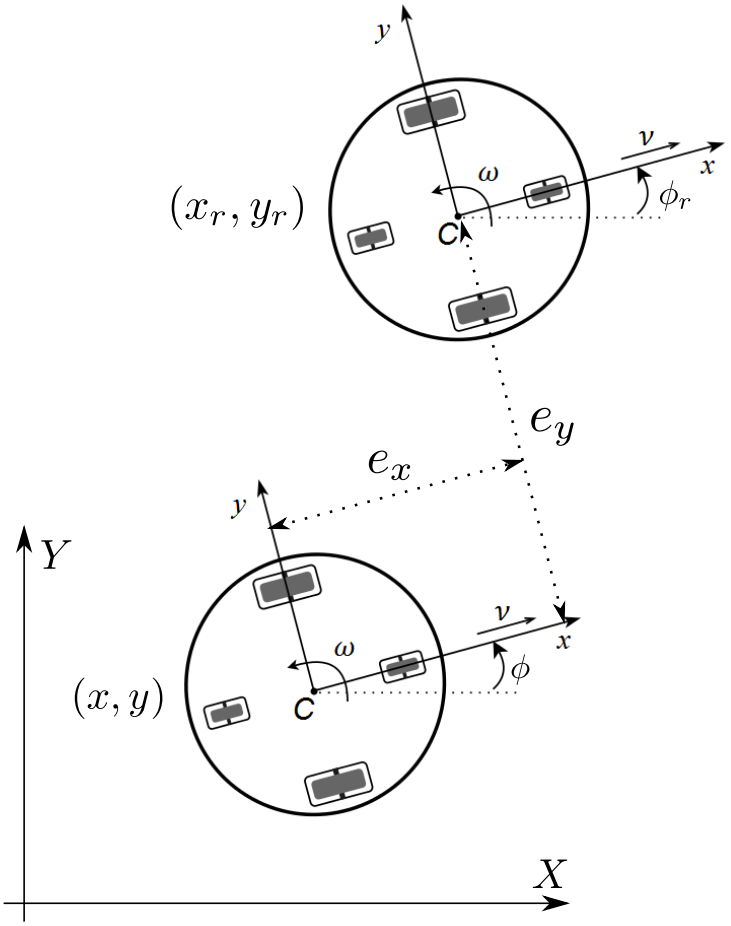
The main problem with the use of the inverse kinematic itself is that it does not reject any kind of disturbance because there is no feedback signal, only a reference frame p_r. It can be

seen as a feed-forward action that drives the robot along a desired path $p_r(t)$ at some interval $t \in [0,T]$

However, defining a virtual reference robot p_r as a tracking objective for our system model allows for a new dynamic model which expresses the error model. Thus, one may specify an error state $e = [e_x\ e_y\ e_\phi]'$.

The error vector is an expression of the global error $p_r - p_c$ referred to the local frame, or body frame, of the robot. Note that $p_c = (x_c, y_c, \phi_c) = (x, y, \phi)$ is the current mobile robot pose, and $p_r = (x_r, y_r, \phi_r)$ is the reference, or desired, pose.

Figure 2 – Robot Error Frame



Source: The author.

Let $R(\phi)$ define a counterclockwise rotation matrix of a vector referred to a same frame of reference

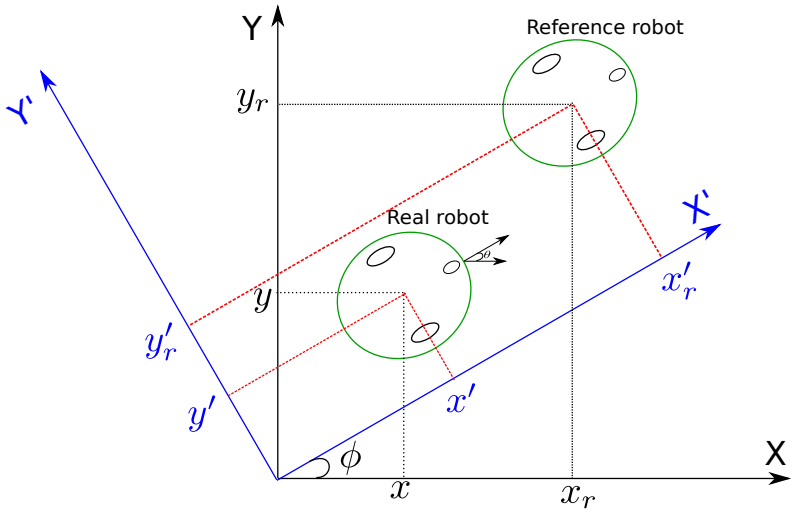
$$R(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.11}$$

From Figure 2, the error is written as

$$e = \begin{bmatrix} e_x \\ e_y \\ e_\phi \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R(\phi)^T} \begin{bmatrix} x_r - x \\ y_r - y \\ \phi_r - \phi \end{bmatrix}. \tag{2.12}$$

The reason for this transformation is because the global frame of reference (X,Y) is being moved to a local frame of reference (X',Y') centered at the same origin but counterclockwise-rotated by ϕ. The local frame is not fixed and rotates with the robot. The use of the transpose transform[1] is required because a whole frame (X,Y) is being rotated counterclockwise, instead of rotating a vector or point. Rotating a frame counterclockwise is seen is as if the vectors or points with respect to the moving frame, in this case, (X,Y), were being rotated clockwise, which is the transformation matrix (2.12).

Figure 3 – Rotating Robot Frame



Source: The author.

As an example, consider the Figure 3. The fixed, navigational coordinates are (X,Y). The robot's coordinate with respect to (X,Y) is (x,y). The mobile robot frame, represented by (X',Y'), is the navigational frame rotated counterclockwise by ϕ. If we define a vector $v = (x,y)$, we compute v' (same v but with respect to (X',Y')) by applying a clockwise rotation using inverse of Eq. (2.11), $v' = (x',y') = R^{-1}(\phi)v$. The same applies to all points or vectors referred to a frame being rotated. This is how we describe the same point using a different, rotated set of coordinates.

[1] Clockwise rotation is done by applying $R^T(\phi)$

To express the system's model in the state-space form $\dot{x} = A.x + B.u$, one computes the derivative of Eq. (2.12).

$$\begin{aligned}
\dot{e}_x &= \frac{d}{dt}(\cos\phi(x_r - x)) + \frac{d}{dt}(\sin\phi(y_r - y)) \\
&= \frac{d}{dt}(\cos\phi(x_r)) - \frac{d}{dt}(\cos\phi(x)) + \frac{d}{dt}(\sin\phi(y_r)) - \frac{d}{dt}(\sin\phi(y)) \\
&= -\sin\phi\dot{\phi}(x_r) + \cos\phi(\dot{x}_r) + \sin\phi\dot{\phi}(x) - \cos\phi(\dot{x}) + \\
&+ \cos\phi\dot{\phi}(y_r) + \sin\phi(\dot{y}_r) - \cos\phi\dot{\phi}(y) - \sin\phi(\dot{y})
\end{aligned} \tag{2.13}$$

Note the presence of familiar terms, such as inverse kinematic velocity from Eq. (2.9) but referred to the real robot v and the rotated error e_y. Isolating known terms and taking $\phi = \phi_r - e_\phi$ and leads to

$$\begin{aligned}
\dot{e}_x = \dot{\phi}\underbrace{(-\sin\phi(x_r - x) + \cos\phi(y_r - y))}_{e_y} - \underbrace{(\cos\phi\dot{x} + \sin\phi\dot{y})}_{v} + \cdots \\
\cdots + \cos(\phi_r - e_\phi)\dot{x}_r + \sin(\phi_r - e_\phi)\dot{y}_r.
\end{aligned} \tag{2.14}$$

Applying proper trigonometric identities on above equation and knowing that $\dot{\phi} = \omega$ leads to

$$\begin{aligned}
\dot{e}_x &= \omega e_y - v + \dot{x}_r(\cos\phi_r\cos e_\phi + \sin\phi_r\sin e_\phi) + \dot{y}_r(\sin\phi_r\cos e_\phi - \cos\phi_r\sin e_\phi) \\
&= \omega e_y - v + \cos e_\phi \underbrace{(\dot{x}_r\cos\phi_r + \dot{y}_r\sin\phi_r)}_{v_r} + \sin e_\phi \underbrace{(\dot{x}_r\sin\phi_r - \dot{y}_r\cos\phi_r)}_{0}
\end{aligned} \tag{2.15}$$

Following this procedure for $\dot{e}_y$ one gets

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\phi \end{bmatrix} = \begin{bmatrix} \cos e_\phi & 0 \\ \sin e_\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} + \begin{bmatrix} -1 & e_y \\ 0 & -e_x \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.16}$$

The errors are directly driven by the current mobile robot velocities v and ω, with which one may devise a control law so that the state vector e may be regulated. There are different ways of deriving a control law to maintain stability (regulation). Tzafestas (2014) suggests the use of a candidate Lyapunov function $V(e) = \frac{1}{2}(e_x^2 + e_y^2) + (1 - \cos e_\phi)/K_y$ with $K_y > 0$. As $V(e) > 0$, one computes $\dot{V}(e)$ so that $\dot{V}(e) < 0$. Selecting the control law

$$\begin{aligned}
v &= v_r\cos e_\phi + K_x e_x \\
\omega &= K_\phi \sin e_\phi + K_y v_r e_y + \omega_r,
\end{aligned} \tag{2.17}$$

satisfies the Lyapunov requirements and guarantees stability for any $K_x, K_y, K_\phi > 0$. Rewriting the control law Eq. (2.17),

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v'_r \\ \omega_r \end{bmatrix} + \begin{bmatrix} v' \\ \omega' \end{bmatrix} = \begin{bmatrix} v_r \cos e_\phi \\ \omega_r \end{bmatrix} + \begin{bmatrix} K_x e_x \\ K_y e_y v_r + K_\phi \sin e_\phi \end{bmatrix} = u_r + u', \tag{2.18}$$

allows the visualization of the separate signals from the feedforward action u_r and the feedback action u'. In this analysis, the feedforward action is responsible for driving the robot along a reference trajectory, and the feedback is responsible for the body frame disturbances rejection.

Chapter 3 will propose the computation of a new signal $u' = -Ke$ that is linear and guarantees stability with closed-loop performance tuned by a weighting matrix.

3 CONTROL STRATEGY FOR NWMR TRAJECTORY TRACKING

In this Chapter a control strategy is proposed for controlling NWMR along a reference trajectory. More specifically, a linear control is desired due to its simplicity. A well known non-linear control from Klancar *et al.* (2005) is discussed as well for comparison purposes.

Mobile robot control early works proposed what is called *Curvature based* controllers (SALICHS *et al.*, 1991), at the same time as Lyapunov function based controllers (KANAYAMA *et al.*, 1990), the latter being the approach in this work. Curvature based controllers are mostly a path following strategy, with no time constraints whatsoever. Recent works still contemplate curvature based controllers by using new technologies such as computer vision to keep track of the curvature along a desired path (WANG; SEKIYAMA, 2015).

However, trajectory tracking control proved to be more interesting for autonomous navigational applications. The need for online obstacle avoidance or simply the need for the mobile robot to reach its endpoint at a specified time is most critical for such applications.

3.1 Mobile Robot Model Linearization

A linearized model is required to design a linear control with some required performance specification. Rewriting Eq. (2.16) with the closed loop control inputs from Eq. (2.17) leads to

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\phi \end{bmatrix} = \begin{bmatrix} -K_x e_x + (K_\phi \sin e_\phi + K_y v_r e_y + \omega_r) e_y \\ -(K_\phi \sin e_\phi + K_y v_r e_y + \omega_r) e_x + v_r \sin e_\phi \\ -(K_\phi \sin e_\phi + K_y v_r e_y) \end{bmatrix} \tag{3.1}$$

The closed loop non-linear model from Eq. (3.1) can be linearized using Taylor series by truncating all terms beyond quadratic power. Taking one of the states, $\dot{e}_x = f_x(e_x, e_y, e_\phi)$, its Taylor series expansion around operating point $e_o = (e_{ox}, e_{oy}, e_{o\phi})$ without quadratic power terms is given by

$$\dot{e}_x \approx f_x(e_o) + \left(\frac{\partial f_x}{\partial e_x}\right)\Big|_{e=e_o} (e_x - e_{ox}) + \left(\frac{\partial f_x}{\partial e_y}\right)\Big|_{e=e_o} (e_y - e_{oy}) + \left(\frac{\partial f_x}{\partial e_\phi}\right)\Big|_{e=e_o} (e_\phi - e_{o\phi}). \tag{3.2}$$

The chosen operating point for linearization is $e_{ox} = e_{oy} = e_{o\phi} = 0$ (mobile robot is exactly at its tracking trajectory point). Computing linearized state $\dot{\tilde{e}}_x$ of Eq. (3.2) description yields

$$\dot{e}_x \approx \dot{\bar{e}}_x = 0 + (-K_x)(e_x - e_{ox}) + (K_\phi \sin e_\phi\big|_{e_\phi = e_{o\phi}} + 2K_y v_r e_y\big|_{e_y = e_{oy}} + \omega_r)(e_y - e_{oy}) + \\ + (K_\phi \cos e_\phi e_y\big|_{e_y = e_{oy}, e_\phi = e_{o\phi}})(e_\phi - e_{o\phi}) = -K_x e_x + \omega_r e_y \quad (3.3)$$

Following the same procedure for e_y and e_ϕ forms the linearized state-space model of the mobile robot

$$\dot{\bar{e}} = \begin{bmatrix} -K_x & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & -v_r K_y & -K_\phi \end{bmatrix} \bar{e}, \quad (3.4)$$

or, more explicitly,

$$\dot{\bar{e}} = \underbrace{\begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix}}_{A} \bar{e} + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} -K_x & 0 & 0 \\ 0 & -v_r K_y & -K_\phi \end{bmatrix}}_{u' = \begin{bmatrix} v' \\ \omega' \end{bmatrix} = -K\bar{e}} \bar{e}. \quad (3.5)$$

The problem now resides in finding a suitable state-feedback matrix K based on matrices A, B. System matrices A, B produces a controllable (full rank controlability matrix) system if either feedforward velocity v_r, ω_r is non-zero. The chosen approach was modeling the problem as a regulation problem. Since the feed-forward already takes care of the steady state error, one must only worry about the linearized error $\bar{e}$. It is important to ratify that the analysis hereafter is made concerning the linearized states $\bar{e}$, and not the real states e. Although relatable, they are not the same states, and the designed controller is designed for the linearized states. Regardless, the linearized model analysis is used only for the gain computation. For practical implementation, the designed controller will be used alongside the real, non-linear states.

3.2 Linear Quadratic Regulation Control

The LQR controller is a well known optimal controller in the literature. As the control theory advances in the 1960s, there was an increasing demand for strategies suited for multivariable systems. The classical control proved itself to be very useful for Single Input Single Output (SISO) systems but not much effective for Multiple Input Multiple Output (MIMO) systems (FRIEDLAND, 1985). Another usefulness for LQR is that some multivariable systems

may be difficult to be approached as a pole placement problem. Even though its mathematically simple to perform, arbitrary pole placement might not always yield good performance results due to its non-intuitive tuning, hence an optimal strategy is quite advantageous in some cases. As our mobile robot system is a multivariable one, with two velocities input and at most three outputs (pose), LQR appears itself to be a suitable controller for such purpose. Although the literature presents many different strategies, from non-linear controllers (KLANCAR *et al.*, 2005), to adaptive controllers (PONS *et al.*, 1994) or neural-network (PHAM *et al.*, 2017) controllers, simple controllers are still an interesting option due to ease of implementation. For NWMR it is quite common to have controllers implemented in an embedded digital system. Such systems are most of the time limited by computational resources, e.g memory, clock, so a simple controller is desired.

The LQR controller was selected to solve the regulation problem described by Eq. (3.5). Let

$$J = \int_t^T [x'(\tau)Qx(\tau) + u'(\tau)Ru(\tau)]d\tau \tag{3.6}$$

be a cost function, one seeks a gain K such that $u(t) = -Kx(t)$ minimizes J for all $t \in [t,T]$. The matrices Q and R are weighting matrices, with which one may tune the computed gain. The solution to this problem is found on many text books, such as Friedland (1985) and Dorf e Bishop (2000). Given a state space system

$$\dot{x} = Ax + Bu, \tag{3.7}$$

the associated Riccati equation that solves the linear quadratic problem of Eq. (3.6) is given by

$$A^T P + PA - PBR^{-1}B^T P + Q = 0, \tag{3.8}$$

where P is the symmetric semi-definite positive matrix solution for Eq. (3.8). The state-feedback controller gain is then

$$K = -R^{-1}BP. \tag{3.9}$$

Gain K is simply a vector or matrix by which one multiplies the measured or estimated state vector x. It is a fixed gain and is easily implemented in practice. Systems that do not prove access to states require an estimator. An optimal estimation is an analogous to the LQR but regarding estimation instead of control. Later sections bring one of the most used optimal estimation in practice, namely the KF.

3.2.1 Remarks

Matrices A and B from Eq. (3.5) form the state-space model to be considered in the LQR controller design, which is tuned for the linearized state vector $\bar{e}$. It should be mentioned that the controller is adjusted for a specific, constant set of feedforward inputs (v_r, ω_r). Although v_r and ω_r are in the model matrix A, it is expected that small variations around chosen values on either or both, from the feedforward portion, will not affect controllability.

Matrix A may be defined angular feedforward input in trajectories with small curvature, which is the case presented in this work. Resulting gain matrix K for such system will be similar to the form from Eq. (3.5).

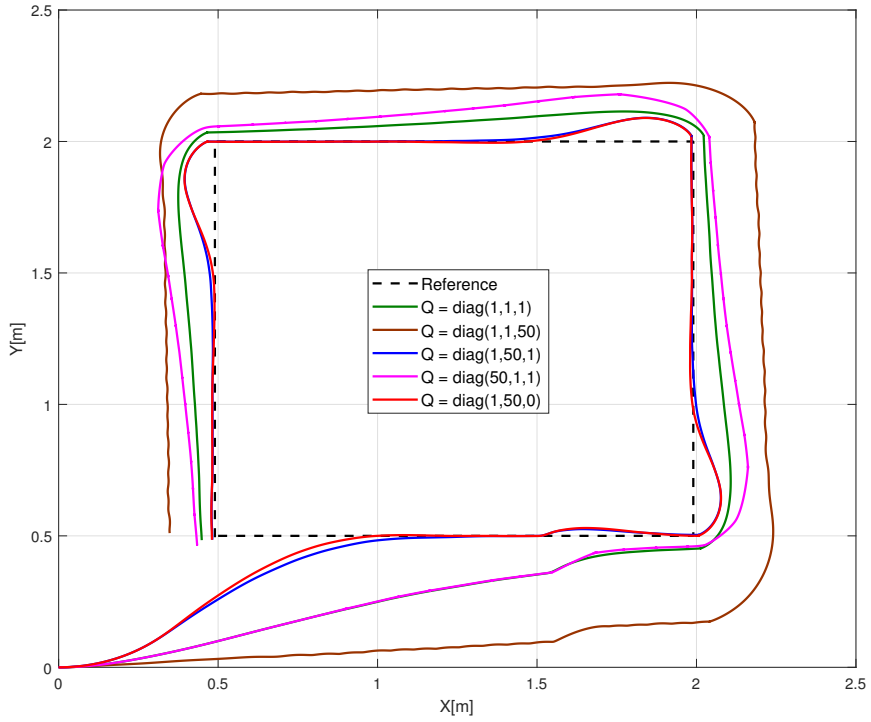
Also, it is not uncommon to perform trajectory tracking with the same feedforward velocity at all times. For reference tracking of line shapes, or with small curvature, one may set $\omega_r = 0$ from matrix A before computing gain matrix K. In case of time varying reference inputs $[v_r \ \omega_r]'$, a gain scheduler for gain matrix K is advised. A Linear Parameter Varying (LPV) may prove useful for this application by tuning $K(\rho)$ for each set of velocity (vertex) $A(\rho)$ and composing a general controller with the linear combination of vertex controllers. This is known as a polytopic approach and will be discussed in future work.

3.2.2 Controller Tuning

The LQR design process allows a certain autonomy for computing an optimal gain K. Since the stability is guaranteed by making $K > 0$, the only requirement is the choosing of matrices Q and R. Thus, the setting $R = I$ and $Q = diag(\alpha_x, \alpha_y, \alpha_\phi)$ is chosen. Parameters α are chosen based on the intuitive rule: Each α_i weights its respective error e_i. So a higher α_x will increase control effort to minimize e_x and so on. Figure 4 exhibits such behavior. This simulation is the tracking of a square-shape trajectory with the same reference velocities at all points.

Observe that not any value for Q results in good performance. It was noticed that large α_x and α_ϕ are usually bad choices for the closed-loop system operation. In fact, setting $\alpha_\phi = 0$ proved to be a better tuning for the matrix in all cases. Higher performances were reached by setting a large value for α_y, a small value for α_x and $\alpha_\phi = 0$ as shown by the red curve. It is important that for practical purposes, too high values for any Q produces high values in K, which is not desired due to excessive control effort that may causes saturation.

Figure 4 – Q-Tuning

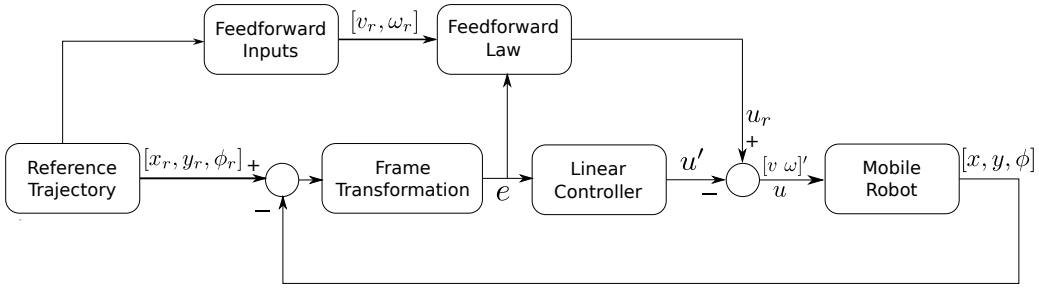


Source: The author.

3.2.3 Controller Scheme

Figure 5 presents the block diagram describing the controller and the system model. The control law $u = [v\ \omega]'$ is given by Eq. (2.17). It has a feedforward component u_r and the

Figure 5 – Linear Control Scheme



Source: The author.

state-feedback component $u' = [v'\ \omega']$.

In summary, the nonlinear controller is reached by the following steps

1. Define a reference trajectory $[x_r(t), y_r(t), \phi_r(t)]$ at each time t;
2. Define reference input velocities $[v_r(t)\ \omega_r(t)]$ at each time t according to Eq. (2.9) or (2.10);

3. Calculate gain matrix K by specifying Q, R and defining A, B as in Eq. (3.5)
4. Compute transformed error e from matrix of Eq. (2.12) after reading robot pose $p = [x(t), y(t), \phi(t)]$;
5. Compute feedback control signal $u' = -K.e$ and feedforward signal u_r according to Eq. (2.18);
6. Apply control signal $u = u_r - u'$ into the mobile robot;
7. Repeat step 4. to 7. at each sampling time until robot reaches end of trajectory;

An important observation is that the reference trajectory of steps 1. and 2. may come from other system such as a path planning layer or obstacle avoidance. Regarding step 6., most NWMR have differential drive, that is, each wheel may turn independently of the other. An expression that relates control inputs $u = [v\ \omega]'$ to each wheel's velocity is given by

$$\dot{\phi}_r = v + \omega L/2 \tag{3.10}$$

$$\dot{\phi}_l = v - \omega L/2 \tag{3.11}$$

Now, consider step 4. The current robot pose $p = [x, y, \phi]'$ is a signal used in the feedback loop. This ratifies the importance of good pose estimation. Many works mentioned in Chapter 1 propose control strategies with no regards on how the mobile robot pose is obtained. It is reasonable to assume that older papers used odometry for that purpose. However, it is known that odometry has many problems on long trajectory as the robot slips or skids after some time. Recent papers have been working on different ways of obtaining precise estimates of robot pose as mentioned in Chapter 1. In Chapter 4 the sensor fusion technique used to yield better pose estimates with IMU sensors is established.

3.3 A Non-Linear Control Strategy

Many different strategies propose non-linear control for the mobile robot system of (2.16). The use of the Lyapunov function to derive a control law as mentioned in the previous section in itself is one type of non-linear control, whose gains K_x, K_y, K_θ are to be calculated based on some rules. More details on the following rules are found in Klancar *et al.* (2005)

Taking the term $K_{2,2} = -\text{signum}(v_r)K_y$ from gain matrix K and computing the characteristic polynomial of system from Eq. (3.5) leads to

$$\det(sI - A + BK) = s^3 + (K_x + K_\phi)s^2 + (K_x K_\phi + K_y v_r + \omega_r^2)s + K_x K_y v_r + K_\phi \omega_r^2, \tag{3.12}$$

with which one can compare to a pre-specified performance polynomial

$$(s+2\zeta\omega_n)(s^2+2\zeta\omega_n+\omega_n^2), \tag{3.13}$$

resulting in the equations

$$K_x+K_\phi = 4\zeta\omega_n, \tag{3.14}$$

$$K_xK_\phi+K_yv_r+\omega_r^2 = 4\zeta^2\omega_n^2+\omega_n^2, \tag{3.15}$$

$$K_xK_yv_r+K_\phi v_r^2 = 2\zeta\omega_n^3. \tag{3.16}$$

A solution can be found by making $K_x = K_\phi = 2\zeta\omega_n$. Now one may determine $K_y = \frac{\omega_n^2+\omega_r^2}{|v_r|}$. The closed loop system poles are constant and given by

$$s_1 = -2\zeta\omega_n \tag{3.17}$$

$$s_2 = \zeta\omega_n+\omega_n\sqrt{(\zeta^2-1)} \tag{3.18}$$

$$s_3 = \zeta\omega_n-\omega_n\sqrt{(\zeta^2-1)}. \tag{3.19}$$

The natural frequency ω_n should be higher than the maximum allowed mobile robot angular velocity ω_r. If v_r is close to zero, K_y goes to infinity and therefore a gain scheduling should be chosen for K_y as $K_y = K_y(t) = g|v_r(t)|$. System characteristic frequency then becomes

$$\omega_n = \sqrt{\omega_r^2(t)+gv_r^2(t)}, \tag{3.20}$$

whose controller gains are $K_x(t) = K_\phi(t) = 2\zeta\omega_n(t)$ and $K_y(t) = g|v_r(t)|$. Notice that the gains may be time-varying since $v_r(t)$ and $\omega_r(t)$ also be time varying depending on the feedforward law. This differs from the linear controller because linear controller is tuned for constant velocities v_r and ω_r. The tuning parameters for this controller are g and ζ. Its control structure is identical to Figure 5, the only difference being the Linear controller block is now a non-linear controller block defined by gains $K_x(t), K_y(t), K_\phi(t)$. This non-linear control strategy is used in this thesis to compare with the proposed linear controller.

4 INERTIAL NAVIGATIONAL SYSTEM

This Chapter introduces a few valuable terminologies used in Inertial Navigation System (INS) such as *inertia* or *inertial frame of reference*, as well as other general topics. Then, the INS sensors are discussed with details regarding what and how each works and by what means we might use those to establish a proper NWMR pose estimation. A basic matrix algebra that is needed for understanding some INS equations is discussed as well.

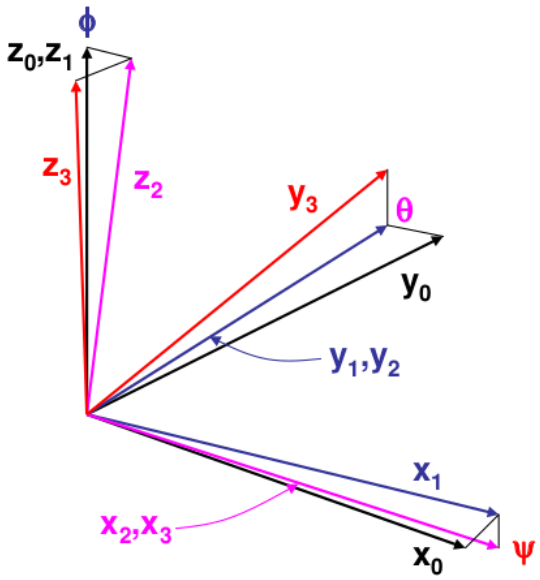
INS refers to the use of an IMU and a computer that performs dead reckoning computations. The computer is responsible for filtering and processing data from IMU. The output of an INS is normally an object's estimated velocity, attitude and position with respect to some starting point. As already mentioned in the introduction, dead reckoning is prone to error accumulation with time as all sensors present some bias and noise. Each sensors' advantages and disadvantages regarding dead reckoning and how to avoid error accumulation using KF is discussed. An attitude estimation model using accelerometer, gyroscope and magnetometer is discussed, as well as linear position and velocity estimation using accelerometer.

4.1 Three Dimensional Rotations

An important concept to understand inertial systems is the idea of Euler Angles. Euler Angles were developed by Leonhard Euler to describe rigid body rotations with respect to a fixed coordinate system. They can also be used to represent the orientation of a mobile frame of reference. Figure 6 shows the rotations about a reference frame (x_0, y_0, z_0) by the Euler angles associated with each axis, (ψ, θ, ϕ), namely, *roll, pitch, yaw*, respectively. The colors help identify the correspondent rotation.

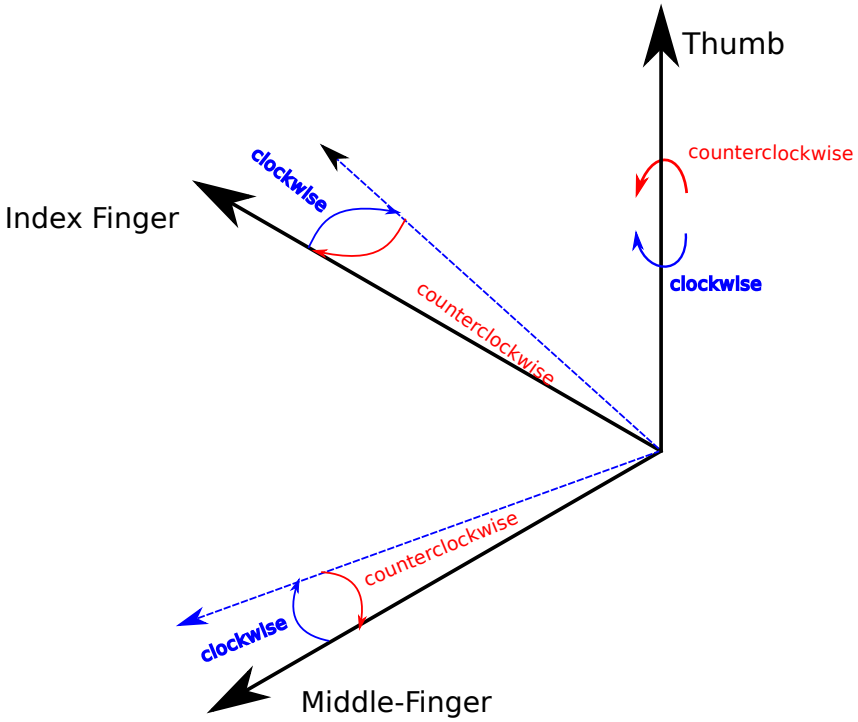
Figure 6 presents also a sequence of a rotating frame. The first frame rotation, $(x_0, y_0, z_0) \rightarrow (x_1, y_1, z_1)$ is a counterclockwise rotation about the Z axis. Thus, it is important to define orientation to make sense of clockwise and counterclockwise terms. For this purpose it is suggested the use of the right hand rule, with the fixed axis about which one rotates being the thumb, as depicted by Figure 7. Which this standard, one may use the expression clockwise and counterclockwise accordingly, clockwise being moving the hand in the direction middle-finger $\Rightarrow$ index finger, and vice versa.

Figure 6 – Euler Angles



Source: Hover e Triantafyllou (2010)

Figure 7 – Right hand rule



Source: The author.

The following matrices define rotations about each axis.

$$R_\psi = R(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{4.1}$$

$$R_\theta = R(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.2}$$

$$R_\phi = R(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

As one might notice, transpose of Eq. (4.3) was used in Chapter 2 to convert the errors from the global reference frame to the robot frame. Rotation matrices can be used for both vector/point rotation or frame rotations. An interesting property is observed with these transformation matrices - they are orthonormal matrices. An orthogonal matrix R satisfies $R^{-1} = R^T$.

The application of a rotation R to a vector v with respect to the same frame of reference is simply done by multiplying

$$v_{rotated} = Rv. \tag{4.4}$$

Likewise, one can represent a fixed vector v in a frame of reference with respect to a new frame, rotating the original one by R using

$$v_{rotatedframe} = R^T v. \tag{4.5}$$

Multiplication of all rotation matrices from Eqs. (4.1) (4.2) (4.3) results in a full description of a three dimensional rotation M_v

$$M_v = R(\psi)R(\theta)R(\phi) = \begin{bmatrix} c\theta c\phi & -c\theta s\phi & s\theta \\ c\psi s\phi + s\psi s\theta c\phi & c\psi c\phi - s\psi s\theta s\phi & -s\psi c\theta \\ s\psi s\phi - c\psi s\theta c\phi & s\psi c\phi + c\psi s\theta s\phi & c\psi c\theta \end{bmatrix} = R(\phi,\theta,\psi), \tag{4.6}$$

with 'c' representing the cosine function and 's' representing the sine function. The multiplication order relates to the order of rotation if the equation is read from right to left. Naturally, to move entire reference frames, use $M_f = R^T(\psi)R^T(\theta)R^T(\phi)$ with the individual transposes. For completeness M_f is also written

$$M_f = R^T(\psi)R^T(\theta)R^T(\phi) = \begin{bmatrix} c\theta c\phi & c\theta s\phi & -s\theta \\ -c\psi s\phi + s\psi s\theta c\phi & c\psi c\phi + s\psi s\theta s\phi & s\psi c\theta \\ s\psi s\phi + c\psi s\theta c\phi & -s\psi c\phi + c\psi s\theta s\phi & c\psi c\theta \end{bmatrix}. \tag{4.7}$$

To summarize, if one has a point or vector defined in a three dimensional navigational system, p_n, and wants to represent the same point but referred to a three dimensional body frame of reference, simply apply $p_b = M_f p_n$. To perform the opposite operation, apply $p_n = M_f^T p_b$. This is an extremely important result since IMU have sensors whose data may come with respect to either an inertial frame (i.e. gravity) or body frame while driving though a navigational frame. Therefore, some frame transformations or conversions are a necessity for correctly handling the measurement data.

The main problem with Euler angles is that the rotations are not unique. That is, if one swaps any rotation matrices multiplication order, the entire M matrix changes. Matrix of Eq. (4.6) and (4.7) are commonly used in literature, specially in an aerial vehicle attitude control. The multiplication order of the matrices are such that attitude is controlled also following a sequence. In other words, yaw ϕ is controlled first, then pitch θ and finally roll ψ. There are a total of six possible three dimensional rotation matrices. This might cause problems such as the *Gimbal Lock*, in which 90^o rotations causes and axis to overlap, effectively leading to a loss of degree. The are a few solutions. One of them is limiting that angle at which the axis can rotate, and performing the appropriate corrections. Another one is using a new set of number system called *Quartertions*. More details can be found on NXP Semiconductors (2015c).

4.2 Inertial Sensors

The term inertial sensors is often used to denote the combination of an accelerometer and a gyroscope. An IMU is a device that contain inertial sensors and some other measuring unit that provide data related to orientation or position. Modern smartphones, cars, airplanes, submarines frequently use inertial sensors to obtain some estimate of their position or orientation. It is common to encounter controllers that require those measurements to be able to operate well. For instance, airplanes are equipped with IMU to estimate its orientation and provide a signal for pitch/roll angles regulation controller. Some cars may also come with IMU in case a GPS satellite is unable to transmit the car's localization, so that they provide an estimate while GPS signal is lost. Without any kind of external guidance or reference, an object's velocity and position is estimated using the already discussed dead reckoning. Until data from external sources arrive, dead reckoning is extremely useful to provide a temporary pose estimation. Therefore, the use of a good estimation model is highly encouraged.

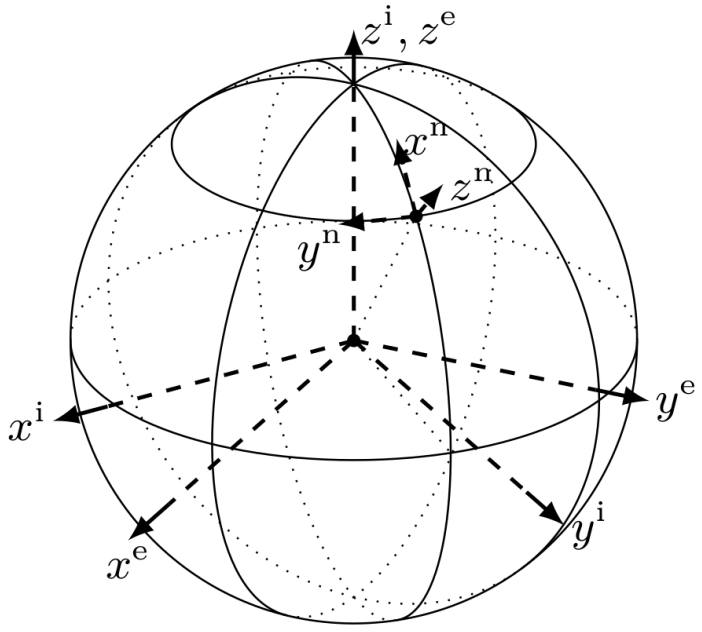
Inertia is the property of bodies to maintain constant translational and rotational

velocity unless disturbed by external forces or torques, respectively. Let us define the frames of reference used in inertial navigation (KOK *et al.*, 2017).

- **Body frame of reference** *b* is the coordinate frame of a moving riding body. The mobile robot error model, for example was described by this frame of reference. Inertial sensors measure accelerations and velocities with respect to this frame.
- **Navigational frame of reference** *n* is the a fixed coordinate one might be interested to navigate in. The mobile robot navigates in this frame of reference, as the reference pose p_r is given with respect to the navigational frame. Its origin is at the surface of the earth.
- **Inertial frame of reference** *i* is a stationary frame with respect to which an IMU measure its linear acceleration and angular velocities. Its origin is located at the center of earth and axis are aligned with the stars.
- **Earth frame of reference** *e* is coincident with the inertial frame but rotates with earth. It is aligned with the *Z* axis of the inertial frame.

Figure 8 presents some of the aforementioned frame of reference coordinate systems. The superscript represents its coordinate system letter (i.e x^i for inertial frame *x*-axis).

Figure 8 – Frames of reference



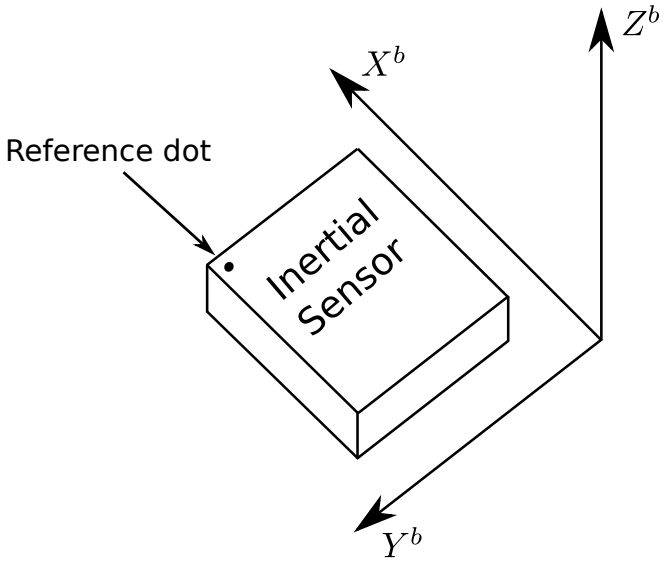
Source: Kok *et al.* (2017)

R^{nb} denotes a rotation of the body frame to the navigational frame. One readily writes the opposite transformation $R^{bn} = (R^{nb})^T$. The same idea covers any other frame transformation. If the origins are different, simply add and offset vector accordingly, nevertheless offset operations

are rarely required. The analysis on the following section will discard small terms such as earth rotation and Coriolis effect on the sensors.

Sensors belonging to an IMU device provide all of its data with respect to the body frame of measurement. Modern sensors are normally encapsulated into a semiconductor package resembling an electronic component. Figure 9 presents the axes of measurements of one IMU sensor.

Figure 9 – Measurement frame of reference of IMU



Source: The author.

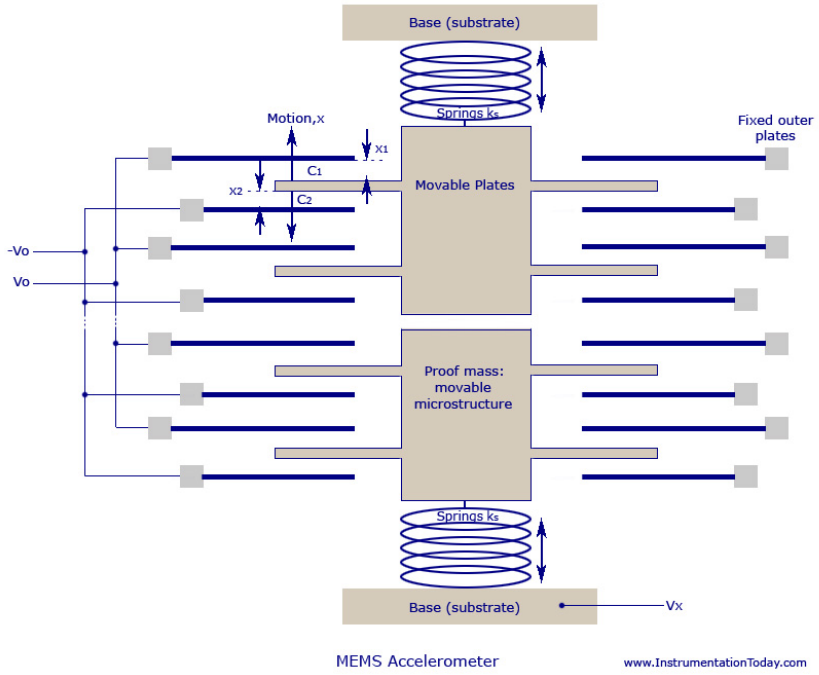
This is an important feature to point out, since a mobile robot navigate on a navigation plane, thus frame transformations are required to convert body frame inertial data to navigational frame data.

4.2.1 Accelerometer

An accelerometer measures the specific force f with respect to its body frame (KOK *et al.*, 2017). This device can be seen as a collection of springs attached to a mass that is sensitive to acceleration on each of its axis.

Whenever a force acts upon the device, the “mass” moves and its distance to its resting point is measured. The measured distance is proportional to the acceleration resulting from the applied force. Current technology allows the construction of such devices with very small mechanical elements, namely MEMS. Figure 10 shows the constructive elements of an accelerometer with MEMS technology.

Figure 10 – Accelerometer MEMS model



Source: (MEMS...,)

A movable proof mass is attached to a spring. Transducers detect its relative position and generates an electrical signal to be interpreted by a microelectronic circuit. Modern accelerometers are capable of measuring accelerations with respect to three dimensional body frame axes, (X^b, Y^b, Z^b).

A general expression of an accelerometer measurement model output along one if its axes is given by

$$a_m = a^b + a_g^b + b^a + v^a, \tag{4.8}$$

where a^b is the linear body acceleration component, a_g^b is body frame gravity component. b^a is the bias and v^a a represents measurement noise. The measurement noise v^a is typically a white Gaussian noise with zero mean and constant variance, which can be computed by taking real sensor measurements. The bias b^a can be modeled as well, either being considered constant or slowly time-varying random walk (KOK *et al.*, 2017).

Generally one would want to remove the gravity component a_g^b. This can be done by transforming its vector into the body frame of reference, and then manually removing it. Expressing the transformed gravity vector $a_g^b = R^{bi} a_g^i$, with R^{bi} being a rotation matrix with known angles and $a_g^i = [0\ 0\ g]^T$. Since a a vector is being expressed in terms of a different frame of reference, one would use Eq. (4.7) as our R^{bi}. To effectively remove its effect on an

accelerometer in order to obtain exclusively linear accelerations, a gravity free acceleration $\hat{a}^b$ is computed

$$\hat{a}^b = a_m - R^{bi} a_g^i = a^b + b^a + v^a. \tag{4.9}$$

Two significant remarks can be observed. First, the rotation matrix R^{bi} must be computed with prior knowledge of all the Euler angles. Second, since a_g^i contains only one term in its last row, the Euler angle ϕ has no effect on the transformed a_g^b. Still, one must compute θ and ψ. Interestingly enough, the accelerometer itself is capable of providing these two angles with the use of gravity . Let $a = [a_x \; a_y \; a_z]^T$ be the measurement output of a three axes accelerometer. The measurement due to gravity is given by

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} c\theta c\phi & c\theta s\phi & -s\theta \\ -c\psi s\phi + s\psi s\theta c\phi & c\psi c\phi + s\psi s\theta s\phi & s\psi c\theta \\ s\psi s\phi + c\psi s\theta c\phi & -s\psi c\phi + c\psi s\theta s\phi & c\psi c\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \sin\psi\cos\theta \\ \cos\psi\cos\theta \end{bmatrix} \tag{4.10}$$

Expressing θ and ψ based on the known output vector $[a_x \; a_y \; a_z]^T$. Dividing second row by third row of Eq. (4.10) returns

$$\frac{a_y}{a_z} = \tan\psi \Rightarrow \psi = \text{atan}\left(\frac{a_y}{a_z}\right) + v^a. \tag{4.11}$$

To compute the pitch, divide the first row by the sum of squares of the second and third row.

$$\frac{a_x}{a_y^2 + a_z^2} = \frac{-\sin\theta}{\sin^2\psi\cos^2\theta + \cos^2\psi\cos^2\theta} = \frac{-\sin\theta}{\cos^2\theta}. \tag{4.12}$$

Taking the square root of the denominator yields

$$\frac{a_x}{\sqrt{a_y^2 + a_z^2}} = -\tan\theta \Rightarrow \theta = -\text{atan}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) + v^a \tag{4.13}$$

The important of denoting there results in terms of inverse tangent is because inverse sine or cosine may have two solutions for the same angle, and tangent has unique solution for a given angle.

Provided one has good estimates of the accelerometer bias b^a, its subtraction from Eq. (4.9) gives

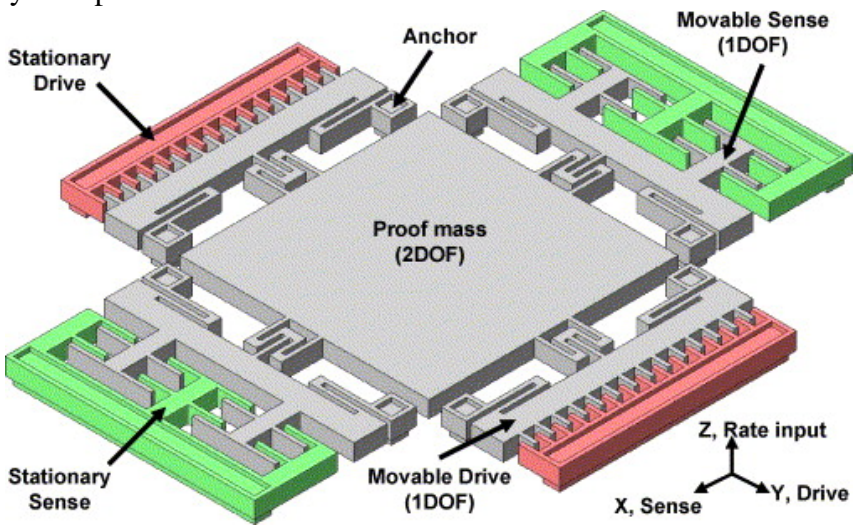
$$\hat{a}^b = a_m - R^{bi} a_g^i - \hat{b}^a + v^a \approx a^b + v^a \tag{4.14}$$

If noise variance of v^a is known, filters may be applied to mitigate its effect on the measurement data. Assuming small estimation errors, integration and double integration may be performed to estimate velocity and position, respectively, of a body using Eq. (4.14) (with respect to the body frame). There are other ways of better estimating the noise and bias, which is the use of other sensor in a sensor fusion model. For instance, the accelerometer may be used to compute roll and pitch angles, but its noise could corrupt the results. A gyroscope has much less noise on its output and is also able to provide roll and pitch angles with a small bias. By combining both sensors, noise and bias effect of the two sensors for estimating roll and pitch are effectively mitigated.

4.2.2 Gyroscope

Gyroscope sensors measure angular rates referred to the body frame (KOK *et al.*, 2017). In fact, it measures inertial angular rates but expressed in the body frame of reference. However, one can neglect the earth rotation rate and the resulting measurements are body measurements with respect to the body frame. Similar to the accelerometer, a gyroscope is also constructed using MEMS technology. A 2DOF gyroscope model is shown in Figure 11. Note that angular motions provoke movements on the Proof mass, making it possible for transducers to perceive the effect and convert it into an electrical signal to be processed. Similarly to accelerometers, modern gyroscopes provide angular rates about the three body frame axes.

Figure 11 – Gyroscope MEMS model



Source: (ALPER *et al.*, 2006)

A general gyroscope measurement model output about one of its body axes is given

by

$$\omega_m = \omega^b + b^g + v^g, \tag{4.15}$$

where ω^b is the actual body angular rate, b^g is the gyroscope bias and v^g its measurement noise. Gyroscopes are not affected by the gravity, which in turn require no transformation whatsoever. Thus, only by knowing its bias b_g an estimate of the body angular rate can be written as

$$\hat{\omega}^b = \omega_m - \hat{b}^g + v^g \approx \omega^b + v^g \tag{4.16}$$

Once again, noise effect may be mitigated by using filters. It is known that the gyroscope bias is often time varying, which requires a time varying estimation as well. In Chapter 5 the aforementioned characteristics are modeled into a state-space fusion model suitable for Kalman filtering.

Furthermore, if one wants to estimate an angle instead of angle rates, this is easily done by integrating our estimation output. Considering the body frame gyroscope angular rates estimates as $\hat{\omega}^b = [\hat{\omega}_x \ \hat{\omega}_y \ \hat{\omega}_z]^T$, the three dimensional Euler angles estimates using gyroscope measurements as computed using

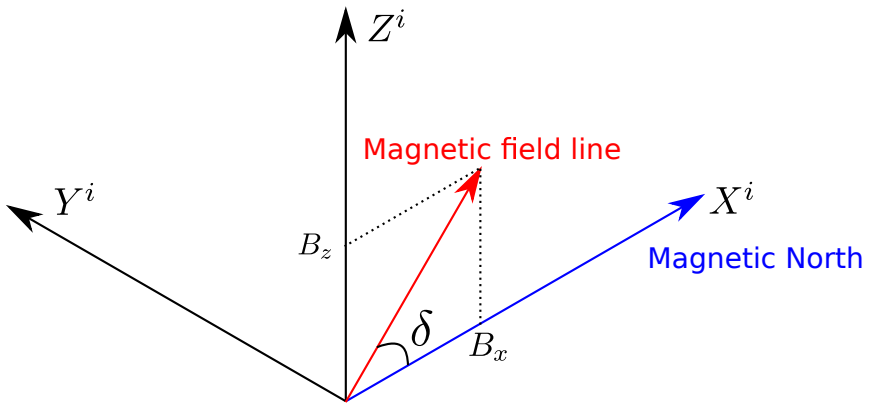
$$\begin{bmatrix} \hat{\psi} \\ \hat{\theta} \\ \hat{\phi} \end{bmatrix} = \int \begin{bmatrix} \hat{\omega}_\psi \\ \hat{\omega}_\theta \\ \hat{\omega}_\phi \end{bmatrix} dt. \tag{4.17}$$

Consider now the mobile robot problem. Assume that the robot will navigate throughout a plane terrain. In that case, there is no use for estimating ψ and θ since the robot will not move in these directions. An estimate of ϕ, however, would prove extremely useful for two reasons. The first being that ϕ is one of the pose states of the mobile robot, and could be directly computed by the gyroscope. The second being that the error frame transformation require that same angle to compute the body frame error model of Eq. (2.12). Having access to that state would undoubtedly increase mobile robot navigation estimate accuracy.

4.2.3 Magnetometer

While not considered an inertial sensor, magnetometer is present in many IMU devices lately. The magnetometer provides another source for computing yaw ϕ which might be useful for neglecting gyroscope bias. It was seen that the accelerometer cannot provide that angle, and the gyroscope alone would have its bias corrupt the yaw estimate on the long run.

Figure 12 – Earth magnetic field effect on magnetometer



Source: The author.

Combining both gyroscope and magnetometer to provide better yaw estimations have shown good results for mobile robot application, as seen in Forte *et al.* (2018).

The working principle of the magnetometer is pretty simple - it measures magnetic field with an impressive sensitivity. So much so that detects earth's magnetic field at its surface. It is quite similar to a needle compass but with more axes. Nevertheless, its sensitivity is also the sensor's weakness as it is highly susceptible to noise. The calibration process of a magnetometer greatly impacts on its resulting outputs for that reason.

A general expression for magnetometer measurement model is now presented. First one writes the earth's magnetic field vector B^i referred to the inertial frame regarding the *magnetic inclination* δ and amplitude B,

$$B^i = B \begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix}. \tag{4.18}$$

The presence of the magnetic inclination angle δ is because the magnetic field lines of earth are not parallel with its surface, and the sensor measures the inclined field line. Also, this angle varies with the latitude and longitude position of the measurement. Figure 12 shows this effect.

Assume that the sensors will not rotate its pitch and roll, and that the sensor has no external disturbance whatsoever. In practice, if well calibrated, the magnetometer behaves similarly to the following analysis.

The magnetometer outputs its measurements referred to the body frame similarly to inertial sensors. Applying the frame rotation of the yaw angle from Eq. (4.3) (Transpose because

it is a frame rotation), one expresses the sensor measured output $B_m = R^{bi}(\phi)B^i$

$$B_m = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B\cos\delta \\ 0 \\ B\sin\delta \end{bmatrix} = \begin{bmatrix} B\cos\delta\cos\phi \\ -B\cos\delta\sin\phi \\ B\sin\delta \end{bmatrix} \tag{4.19}$$

Dividing the second row by the first row of matrix Eq. (4.19) results in

$$\frac{B_y}{B_x} = -\tan\phi \Rightarrow \phi = -\text{atan}\left(\frac{B_y}{B_x}\right) + v^m, \tag{4.20}$$

where v^m is measurement noise. Note that the inclination angle does not affect the yaw angle from the magnetometer. Eq. (4.20) provided a new way of computing the much needed ϕ for out mobile robot. With this result, combination with the gyroscope integrated ϕ is possible.

If one considers that the magnetometer has rotated about all three axes, the matrix $R^{bn}(\phi)$ becomes $R^{bn}(\phi,\theta,\psi)$. The θ and ψ angles, in this case, are calculated using accelerometer, gyroscope or both, as already seen. It is possible to implement the so called *Tilt Compensation* by de-rotating the magnetometer measurement vector using accelerometer/gyroscope pitch and roll. Mathematically, one writes $B_m = R_\psi^T R_\theta^T R_\phi^T B^i \Rightarrow R_\phi^T B^i = R_\theta R_\psi B_m$.

$$\begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B\cos\delta \\ 0 \\ B\sin\delta \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} \tag{4.21}$$

$$\begin{bmatrix} B\cos\delta\cos\phi \\ -B\cos\delta\sin\phi \\ B\sin\delta \end{bmatrix} = \begin{bmatrix} B_x\cos\theta + B_y\sin\psi\sin\theta + B_z\cos\psi\sin\theta \\ B_y\cos\psi - B_z\sin\psi \\ -B_x\sin\theta + B_y\cos\theta\sin\psi + B_z\cos\psi\cos\theta \end{bmatrix} \tag{4.22}$$

Finally, the computation of the tilt-compensated yaw ϕ_c is similar to Eq. (4.20) (dividing the same rows), given by

$$\phi_c = -\text{atan}\left(\frac{B_y\cos\psi - B_z\sin\psi}{B_x\cos\theta + B_y\sin\psi\sin\theta + B_z\cos\psi\sin\theta}\right) \tag{4.23}$$

4.3 Inertial Sensor Calibration

Calibration procedures are almost always required, otherwise the sensor output data becomes quite meaningless. Every sensor comes with some manufacturing flaw. Knowing

that, manufacturers publish calibration documentation for all the sensors they construct. An accelerometer, for example, when inert on a horizontal plane, should output gravity's effect on the Z axis with approximately $9.8 m/s^2$ positive. If not, calibration is required so that it measures what is expected. The same idea applies to other sensors.

4.3.1 Accelerometer Calibration

Accelerometer calibration is quite straightforward. Although it is not possible to eliminate some of its unwanted properties(bias, noise), one can assure that its measurements are correctly aligned with its axis. The calibration steps are described as follows:

1. Place the accelerometer or IMU on a table that is aligned with the X, Y coordinates of the navigational frame and do not move the sensor.
2. Take multiple measurements on all its axes $a_m = [a_x \; a_y \; a_z]^T$ and compute their average value vector $[a_x^{avg} \; a_y^{avg} \; a_z^{avg}]^T$.
3. Compose an offset correction vector $a_o = [0 \; 0 \; 9.8]^T - [a_x^{avg} \; a_y^{avg} \; a_z^{avg}]^T$.

Thereafter, simply sum the offset value with the raw measurements to compute the calibrated output

$$a_c = a_m + a_o. \tag{4.24}$$

The signal a_c should be used for any accelerometer accelerations data processing. Multiplicative error may be considered non existing.

4.3.2 Gyroscope Calibration

Gyroscope calibration is also as simple as the accelerometer. The only difference being that the expected output when inert is $\omega_m = 0$.

1. Place the gyroscope or IMU on a table and do not move the sensor;
2. Take multiple measurements on all its axes $\omega_m = [\omega_x \; \omega_y \; \omega_z]^T$ and compute their average value vector $[\omega_x^{avg} \; \omega_y^{avg} \; \omega_z^{avg}]^T$.
3. Compose an offset correction vector $\omega_o = [0 \; 0 \; 0]^T - [\omega_x^{avg} \; \omega_y^{avg} \; \omega_z^{avg}]^T$.

The calibrated gyroscope measurements are then given by

$$\omega_c = \omega_m + \omega_o \tag{4.25}$$

The signal ω^c should be used for any gyroscope angular rate data processing. Multiplicative error may be considered non existing.

4.3.3 Magnetometer Calibration

Magnetometer calibration is slightly more complicated that calibration of the inertial sensors. Because magnetometer are extremely sensitive to external magnetic disturbances, their measurement output is easily corrupted. There are two main issues that may disturb correct magnetometer reading which can be compensated in the calibration process. Then, each effects are modeled in order to remove them in the calibration process. each effect More details can be found on NXP Semiconductors (2015a) and NXP Semiconductors (2015b).

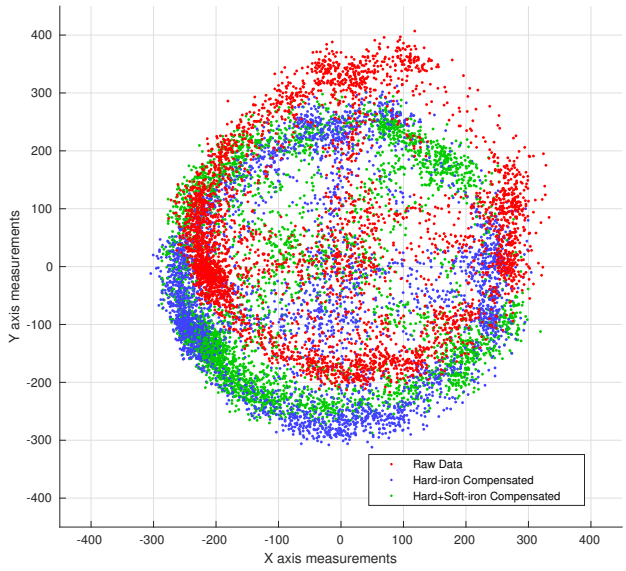
- Hard-Iron is a magnetic interference generated by the permanently magnetized components on the magnetometer circuit board and other fixed metallic components around its position. These components generate small magnetic fields detectable by the magnetometer. If one assumes that the components are always fixed with respect to the body frame of the magnetometer the hard-iron effect behaves as an additive field vector.
- Soft-Iron is a magnetic interference created by the induction of temporary magnetic field into normally unmagnetized ferromagnetic components, such as batteries or steel shields. These effects are harder to model since the induced depend on the relative IMU orientation to earth's geomagnetic field.

The hard-iron effect is modeled as an additional vector present on all measurements. The soft-iron effect is modeled as a multiplicative matrix, or scaling gain, all measurements. The additive effect due to hard-iron is denoted by V, the multiplicative effect is denoted by W and the magnetic field due to earth's has amplitude B inclined by δ. Let B_m be the measurement vector after frame transformation $R(\phi,\theta,\psi)$

$$B_m = WR(\phi,\theta,\psi)B\begin{bmatrix}\cos\delta\\0\\\sin\delta\end{bmatrix}+V. \tag{4.26}$$

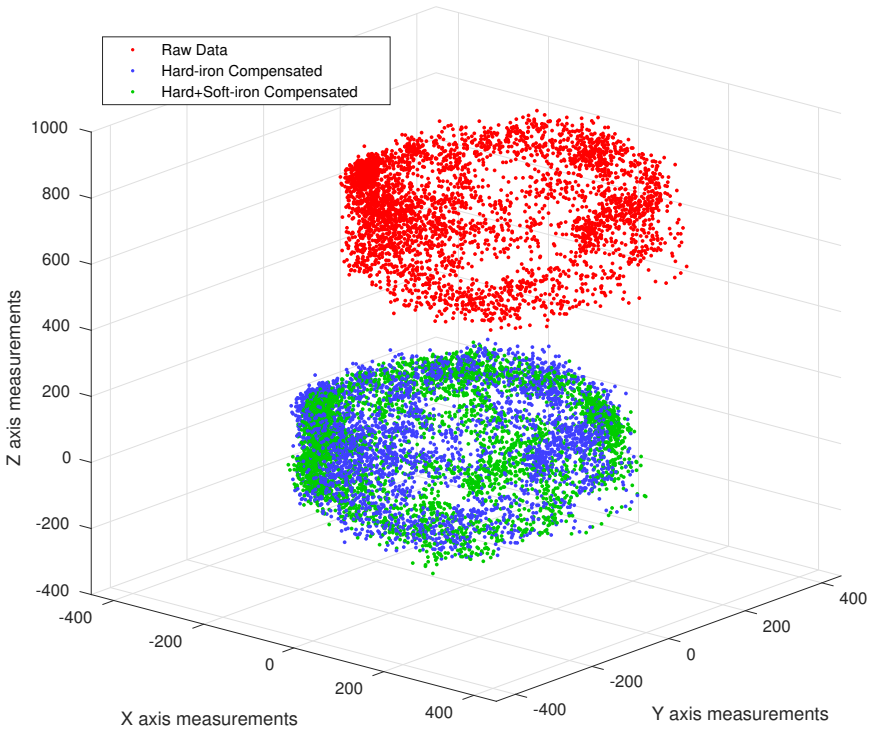
Application note NXP Semiconductors (2015b) shows that an well calibrated magnetometer after arbitrary rotation has its measurement output describe a sphere located at the origin, namely the *Measurement Locus*. Non-calibrated magnetometers usually describe an ellipsoidal shaped surface. The following Figures 13 and 14 compares real raw data with calibrated data from the same magnetometer used in the mobile robot.

Figure 13 – Magnetometer Calibration on plane XY



Source: The author.

Figure 14 – Magnetometer Calibration on space XYZ

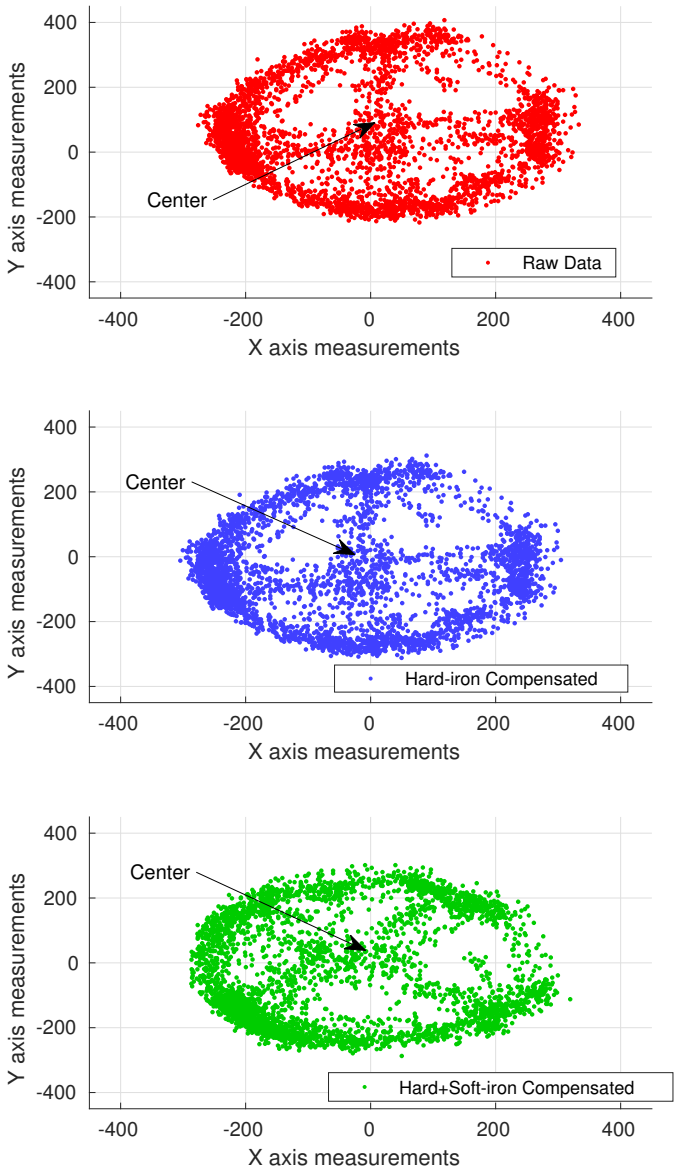


Source: The author.

Note how the red dots points of raw data are shifted to the positive directions of Y

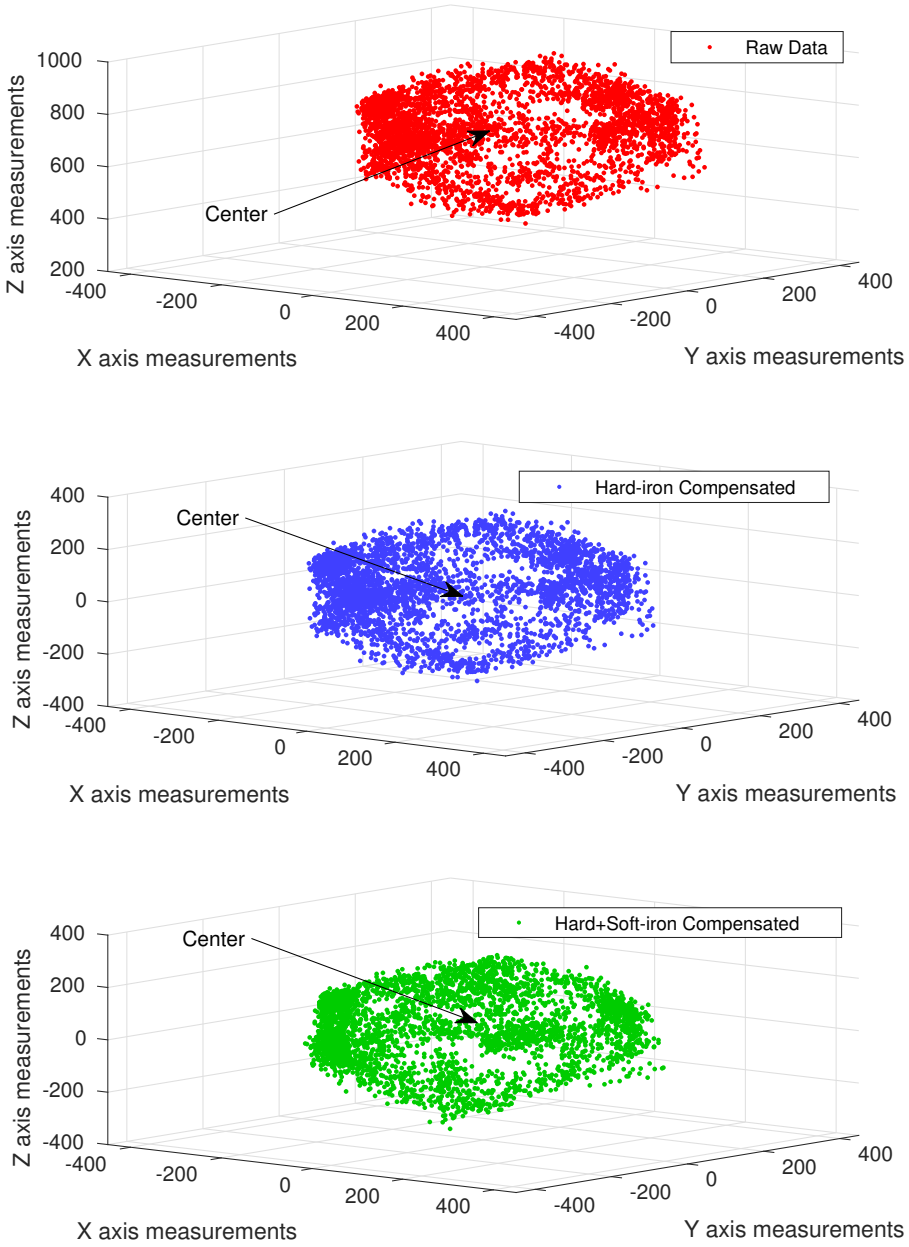
and Z, but still quite spherical (which is good). Subtraction of the hard-iron offset from raw data results in blue dot data points. In addition, observe how similar are the data points comparing the individual hard-iron compensation data with the hard-iron plus soft-iron compensation. This means that the soft-iron matrix is close to the identity I matrix and can be neglected. The following Figures 15 and 16 presents the same calibration results separately.

Figure 15 – Magnetometer Calibration on plane XY - Separate



Source: The author.

Figure 16 – Magnetometer Calibration on space XYZ - Separate



Source: The author.

The hard-iron effect shifts the ellipsoid origin, and the soft-iron is responsible for its ellipsoidal shape NXP Semiconductors (2015a). Hard-iron and soft-iron effect are computed by comparing Eq. (4.26) with the general expression of an ellipsoid. Let R be the vector on its

surface, R_0 be its origin and A a symmetric matrix, the expression is given by

$$(R-R_0)^T A(R-R_0) = constant. \tag{4.27}$$

Taking Eq. (4.26) and moving V and W to the left hand side of the equation leads to

$$W^{-1}(B_m - V) = R(\phi,\theta,\psi)B \begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix} \tag{4.28}$$

Rewriting Eq. (4.28) in quadratic form (assume W is symmetric) gives

$$\left(W^{-1}(B_m - V)\right)^T W^{-1}(B_m - V) = \left(R(\phi,\theta,\psi)B \begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix} \right)^T R(\phi,\theta,\psi)B \begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix} = B^2. \tag{4.29}$$

Observe the similarity of Eq. (4.29) with (4.27). The problem now resides on using an ellipse fit algorithm to either recursively or using a data set to compute matrix W and vector V. STMicroelectronics (2016) presents a Matlab^{TM} code which can be translated to a micro-controller language for online computation. Once W and V are computed, calibrated magnetometer measurements are given by

$$B_c = W^{-1}(B_m - V) \tag{4.30}$$

By neglecting the effect of soft-iron (make $W = I$) one can easily compute the hard-iron effect $V = [V_x\ V_y\ V_z]^T$ using the Least Square Error (LSE) method by taking measures $B_m = [B_{mx}\ B_{my}\ B_{mz}]^T$. Magnitude vector B is also computed within this approach. Rewriting Eq. (4.29)

$$(B_m - V)^T(B_m - V) = B^2 \Rightarrow B_m^T B_m - 2B_m^T V + V^T V - B^2 = 0 \tag{4.31}$$

Define a residue of measurement $r[k]$ as the evaluation of Eq. (4.31)

$$r[k] = B_{mx}[k]^2 + B_{my}[k]^2 + B_{mz}[k]^2 - 2B_{mx}[k]V_x - 2B_{my}[k]V_y - 2,B_{mz}[k]V_z + V_x^2 + V_y^2 + V_z^2 - B^2 \tag{4.32}$$

$$r[k] = (B_{mx}[k]^2 + B_{my}[k]^2 + B_{mz}[k]^2) - \begin{bmatrix} B_{mx}[k] \\ B_{my}[k] \\ B_{mz}[k] \\ 1 \end{bmatrix}^T \begin{bmatrix} 2V_x \\ 2V_y \\ 2V_z \\ B^2 - V_x^2 - V_y^2 - V_z^2 \end{bmatrix}. \tag{4.33}$$

Eq. (4.33) is appearing in the least square problem format. Forming a measurement vector at each time $r = [r[0]\ r[1]\ ...\ r[M-1]]^T$, Eq. (4.33) becomes

$$\begin{bmatrix} r[0] \\ r[1] \\ ... \\ r[M-1] \end{bmatrix} = \begin{bmatrix} B_{mx}[0]^2 + B_{my}[0]^2 + B_{mz}[0]^2 \\ B_{mx}[1]^2 + B_{my}[1]^2 + B_{mz}[1]^2 \\ ... \\ B_{mx}[M-1]^2 + B_{my}[M-1]^2 + B_{mz}[M-1]^2 \end{bmatrix} - \\ - \begin{bmatrix} B_{mx}[0] & B_{my}[0] & B_{mz}[0] & 1 \\ B_{mx}[1] & B_{my}[1] & B_{mz}[1] & 1 \\ ... & & & \\ B_{mx}[M-1] & B_{my}[M-1] & B_{mz}[M-1] & 1 \end{bmatrix} \begin{bmatrix} 2V_x \\ 2V_y \\ 2V_z \\ B^2 - V_x^2 - V_y^2 - V_z^2 \end{bmatrix}, \tag{4.34}$$

Or, more abstractly,

$$r = Y - X\beta, \tag{4.35}$$

where Y is the vector of known dependent variables, X is the matrix of known magnetometer measurements and β is the solution to the least square error problem. Defining $P = r^T r$ as a cost function to be minimized, one computes its derivative relative to the desired parameter vector β. The X matrix has to take measurements from different angles, otherwise noise from the same angle may corrupt the parameter estimation. In other words, one should rotate the magnetometer about all Euler axis to effectively form the measurement surface.

$$P = r^T r = (Y - X\beta)^T (Y - X\beta) \tag{4.36}$$

$$\partial P / \partial \beta = 0 \Rightarrow \beta = (X^T X)^{-1} X^T Y \tag{4.37}$$

This method is online computable if one takes the current measurement $r[i]$ and form the matrices X and Y at the time of the measurement and summing with the previous ones instead of storing M-sized vectors and matrices. However, it might be excessively demanding for an embedded system, it is suggested calculating the LSE offline by storing sufficient measurement vectors in a computer and calculating β to be used in the embedded system.

Another, simpler method is by offsetting the entire shape of measurements to the origin. Assuming the measurements form a sphere (soft-iron is the identity matrix), numerous measurements are taken while rotating the magnetometer about Euler angles and stored the maximum and minimum of each axis. Then, simply shift the measurement output by

$$B_{cx} = B_{mx} - (V_x^{max} + V_x^{min})/2, \tag{4.38}$$

$$B_{cy} = B_{my} - (V_y^{max} + V_y^{min})/2, \tag{4.39}$$

$$B_{cz} = B_{mz} - (V_z^{max} + V_z^{min})/2, \tag{4.40}$$

where $B_c = [B_{cx}\ B_{cy}\ Bcz]^T$ is the calibrated magnetometer output.

5 SENSOR MODELS

Once calibration is correctly applied to each sensor output, it is now possible to process their data accordingly for pose or attitude estimation. The modeling of a sensor is similar to the modeling of a control system - a state-space approach might be used to describe the dynamics of a sensor with all its characteristics included. Not only that, but multiple sensors dynamics may be included into a single state-space model. Within that single model, the flaws of each sensor are included so that proper estimation of such flaws are also realized. State-space sensor modeling is extremely important as it provided the mathematical means for linear or non-linear estimation. This Chapter introduces one of the most important linear filtering strategies used in literature for state estimation, namely, KF, as well as its usage for fusing sensor data.

5.1 Kalman Filtering

The KF is a highly regarded optimal state estimator used in numerous applications other than control theory. In fact, most of its usage exists in the area of sensor/data fusion, where there is no closed loop system, only supervision or estimation. For control purposes, KF is often, but not exclusively, used to estimate a state vector in a heavily noisy systems which have no accessible states for a state-feedback control. The mathematical model of KF includes some stochastic analysis, in which one might add process noise and measurement noise stochastic properties. Normal state estimators provide no means for dealing with the noise effect. Not only that, but KF is also an algorithm. It means that it can be either be used to compute optimal gains on the steady state of a linear system or be used to compute a time varying gain recursively. In 1960, Rudolf Emil Kalman proposed a solution to the Wiener problem (estimating noisy processes with least square error) for a discrete time analysis. Norbert Wiener was among the first to propose a stochastic analysis for linear filtering design, but his work was rarely used in practice due to its complexity. Kalman's published work on linear filtering Kalman (1960) provided the much needed groundwork for practical implementation with an algorithm included. Note that linear filtering applies not only for signals but also for data sets, such as images, as well.

Let us now discuss the KF proposed state estimation. Let

$$x_{k+1} = A_k x_k + w_k \tag{5.1}$$

$$y_k = C_k x_k + v_k \tag{5.2}$$

be a discrete time state space model, where x_k is the state at time k, y_k the output vector, A_k is the state matrix, C_k the output matrix, w_k is the process noise assumed to be white Gaussian with known covariance and v_k is the measurement noise, also assumed to be white Gaussian with known covariance.

Define covariance matrices for signal w_k and v_k as

$$E[w_k w_k^T] = \begin{cases} Q_k, \ i = k \\ 0, \ i \neq k \end{cases} \tag{5.3}$$

$$E[w_k w_k^T] = \begin{cases} R_k, \ i = k \\ 0, \ i \neq k \end{cases} \tag{5.4}$$

$$E[w_k v_k^T] = 0 \text{ for all k and i,} \tag{5.5}$$

where $E[.]$ is the expectation operator. Assume one has an initial state estimate at time t_k and that estimate is based on all knowledge prior to t_k, denoted by $\hat{x}_k^-$. The superscript minus denotes its a priori characteristic. Now define the estimation error

$$e_k^- = x_k - \hat{x}_k^-, \tag{5.6}$$

with associated covariance matrix given by

$$P_k^- = E[e_k^- e_k^{-T}] = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T]. \tag{5.7}$$

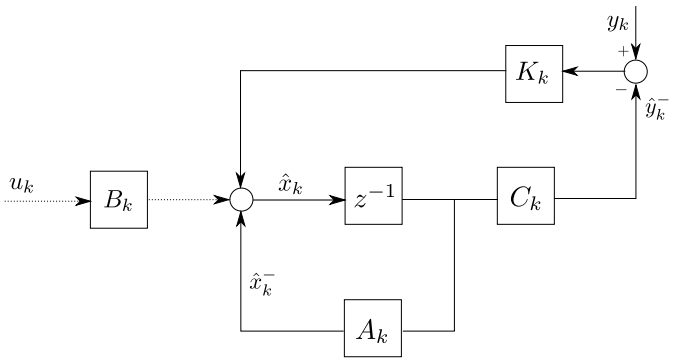
with a prior estimate $\hat{x}_k^-$, the measurement z_k is used to further improve the estimation using

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C_k \hat{x}_k^-) \tag{5.8}$$

Note the similarity with the conventional linear estimators from control textbooks. In fact, the Kalman filter has the same structure of a state observation. See Figure 17.

While it might be confusing looking at the control structure alone due to the existence of *a priori* and *a posteriori* estimates, the algorithm that will be discussed later on will clarify the computation sequence.

Figure 17 – Kalman Filter structure



Source: The author.

Rewriting the covariance matrix P_k but in terms of a known output vector y_k, namely, *a posteriori*

$$\begin{aligned} P_k &= E[e_k e_k^T] = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = \\ &= E\left\{ \left[(x_k - \hat{x}_k^-) - K_k(C_k x_k + v_k - C_k \hat{x}_k^-)\right] \left[(x_k - \hat{x}_k^-) - K_k(C_k x_k + v_k - C_k \hat{x}_k^-)\right]^T \right\}. \end{aligned} \tag{5.9}$$

By performing the indicated expectation and taking note that $(x_k - \hat{x}_k^-)$ is part of the *a priori* covariance error matrix P_k^-, results in

$$\begin{aligned} P_k &= (I - K_k C_k) P_k^- (I - K_k C_k)^T + K_k R_k K_k^T = \\ &= P_k^- - K_k C_k P_k^- - P_k^- C_k^T K_K^T + K_k (C_k P_k^- C_k^T + R_k) K_k^T \end{aligned} \tag{5.10}$$

Note that the second and third term of Eq. (5.10) is linear on K_k and the fourth term is quadratic on K_k. One wishes to minimize the trace of P_k since it is the sum of the mean square error in the estimates of all elements of the state vector (BROWN; HWANG, 1997). Therefore, using matrix differentiation properties,

$$\frac{d(tr(P_k))}{dK_k} = -2(C_k P_k^-)^T + 2K_k(C_k P_k^- C_k^T + R_k). \tag{5.11}$$

Setting Eq. (5.11) to zero yields

$$K_k = P_k^- C_k^T (C_k P_k^- C_k^T + R_k)^{-1}. \tag{5.12}$$

Eq. (5.12) is known as the *Kalman gain*. With it, substitute the K_k terms in Eq. (5.10) to calculate an optimal covariance matrix

$$\begin{aligned} P_k &= P_k^- - P_k^- C_k^T (C_k P_k^- C_k^T + R_k)^{-1} C_k P_k^- = \\ &= P_k^- - K_k (C_k P_k^- C_k^T + R_k) K_k^T = \\ P_k &= (I - K_k C_k) P_k^- \end{aligned} \tag{5.13}$$

Eq. (5.13) is only valid for an optimal gain K_k, while (5.10) is valid for any K_k be it either optimal or sub-optimal. The simplest form $P_k = (I - K_k C_k)P_k^-$ is most used for practical applications.

Finally, the next *a priori* error covariance matrix P_{k+1}^- is projected in terms of a known, previous error covariance matrix P_k. Given $e_{k+1}^- = x_{k+1} - \hat{x}_{k+1} = (A_k x_k + w_k) - A_k \hat{x}_k = Ae_k + w_k$, one writes

$$\begin{aligned} P_{k+1}^- &= E[e_{k+1}^- {e_{k+1}^-}^T] = E[(A_k e_k + w_k)(A_k e_k + w_k)^T] = \\ &= A_k P_k A_k^T + Q_k. \end{aligned} \tag{5.14}$$

The Kalman Filter algorithm is now ready to be composed.

5.1.1 Recursive Kalman Filter Algorithm

The recursive KF algorithm is solely based on the already presented equations. It merely defined a sequence of computation, defined within two stages: **predict** and **update**. This algorithm is largely used in the literature and practical applications, specially sensor fusion applications. Its discrete-time approach also favors digital implementation. Its simplicity permits an embedded system implementation without much concern for computational resource limitation. The sequence is as follows

Predict

1. $\hat{x}_k^- = A_k \hat{x}_{k-1} + B_k u_k$
2. $P_k^- = A_k P_{k-1} A_k^T + Q_k$

Update

3. $K_k = P_k^- C_k (C_k P_k^- C_k^T + R_k)^{-1}$
4. $\hat{x}_k = \hat{x}_k^- + K_k (y_k - C_k \hat{x}_k^-)$
5. $P_k = (I - K_k C_k) P_k^-$

Matrix B_k is included to clear up that the existence of an input matrix does not affect the analysis, and that the KF also works for non-autonomous systems.

5.1.2 Steady-state Kalman Filter

There is also a steady-state solution to the KF optimal estimator design. It models the KF as an algebraic matrix Riccatti equation. This is done by substituting first form of Eq.

(5.13) into Eq. (5.14) (considering $P_{k+1}^{-} = P_k^{-}$, steady state).

$$P_{k+1}^{-} = A_k(P_k^{-} - P_k^{-}C_k^T(C_kP_k^{-}C_k^T + R_k)^{-1}C_kP_k^{-})A_k^T + Q_k, \tag{5.15}$$

whose solution for $P_{k+1}^{-} = P_k^{-}$ is the symmetric positive error covariance matrix at steady state. The steady-state Kalman gain is given by Eq. (5.12) using steady state P_k^{-}.

5.2 Sensor Models

The sensor fusion models for mobile robot pose estimation are proposed within this section. The models include an attitude estimation model for estimation of roll ψ, pitch θ and yaw ϕ. An accelerometer velocity and position estimation along an axis from body frame of reference is presented as well. Yaw ϕ estimation will provide the mobile robot kinematic model with accurate angular motion detection for all the required frame transformation matrices and also for mobile robot's own heading direction. Position and velocity estimations are very useful for detecting linear disturbances that may affect the mobile robot as it tracks a reference trajectory.

5.2.1 Magnetometer and Gyroscope Fusion

This is the same model presented in Forte *et al.* (2018). First, let the gyroscope z axis, whose integration yields the yaw angle, be defined as

$$\omega_\phi^g = \omega_\phi + b_\phi^g + v_\phi^g \tag{5.16}$$

where ω_ϕ^g is the gyroscope sensor measurement output, ω_ϕ is the actual angular rate, b_g the gyroscope bias and v_g is measurement noise assumed to be white Gaussian. Yaw ϕ is computed by integrating Eq. (5.16). Using numerical integration with sampling time T_s,

$$\phi^g{}_k = \phi^g{}_{k-1} + T_s\omega_{\phi\,k-1}^g = \phi^g{}_{k-1} + T_s(\omega_\phi + b_\phi^g + v_\phi^g)_{k-1} \tag{5.17}$$

Note that both bias b_ϕ^g and noise v_ϕ^g are being integrated. The magnetometer model is assumed to output its yaw directly using formula Eq. (4.20) but with heavy noise v^m included.

$$\phi_k^m = \phi_{k-1}^m + v^m \tag{5.18}$$

Now onto the fusion of the readings into a sensor model. Consider ϕ_k^m our model output with magnetometer noise. System input is the gyroscope reading ω_ϕ^g. Assuming the gyroscope bias

b_ϕ^g is time-varying, one would want to estimate bias as well, so its considered one of the model's states. The state space sensor fusion model is then given by

$$\begin{aligned} x_{\phi\,k+1} &= \begin{bmatrix} \phi \\ b_\phi^g \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \phi \\ b_\phi^g \end{bmatrix}_k + \begin{bmatrix} T_s \\ 0 \end{bmatrix} \omega_{\phi\,k}^g + v_\phi^{g\,\prime} \\ y_{\phi\,k} &= \phi_k = \begin{bmatrix} 1 & 0 \end{bmatrix} + v^m \end{aligned} \tag{5.19}$$

Both magnetometer and gyroscope stochastic properties have been included into the same model Eq. (5.19). Integrated noise $v_\phi^{g\,\prime}$ is considered a Wiener process noise, even though gyroscope noise is known to be very small. A KF is then used to optimally estimate our states $x_k = [\phi \; b_g]$. The covariance matrices Q_k (proposed) and R_k are given by

$$Q_k = q \begin{bmatrix} \frac{T_s^2}{2} & 0 \\ 0 & T_s \end{bmatrix} \qquad R_k = var(v^m) \tag{5.20}$$

The selected matrix Q_k is similar to the Brownian motion process noise matrix of Eq. (5.24) because there is an white Gaussian noise v_ϕ^g integration.

5.2.2 Accelerometer and Gyroscope Fusion

The model is similar to the previous one, except there are now two angles as the output of the system. Using Eq. (5.17) but for each axis, the fusion model is written as

$$\begin{aligned} x_{\theta,\psi\,k+1} &= \begin{bmatrix} \theta \\ b_\theta^g \\ \psi \\ b_\psi^g \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -T_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ b_\theta^g \\ \psi \\ b_\psi^g \end{bmatrix}_k + \begin{bmatrix} T_s & 0 \\ 0 & 0 \\ 0 & T_s \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_\theta^g \\ \omega_\psi^g \end{bmatrix}_k + v_{\theta,\psi}^{g\,\prime} \\ y_{\theta,\psi\,k} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + v_{\theta,\psi}^a \end{aligned} \tag{5.21}$$

Roll ψ and pitch θ are now directly calculated by accelerometer using formulas from Eqs. (4.11) and (4.13), composing output vector y_k. The same principle of computing Q_k and R_k from previous section also applies for this instance

$$Q_k = q \begin{bmatrix} \frac{T_s^2}{2} & 0 & 0 & 0 \\ 0 & T_s & 0 & 0 \\ 0 & 0 & \frac{T_s^2}{2} & 0 \\ 0 & 0 & 0 & T_s \end{bmatrix} \qquad R_k = var(v_{\theta,\phi}^a) = \begin{bmatrix} var(v_\theta^a) & 0 \\ 0 & var(v_\psi^a) \end{bmatrix} \tag{5.22}$$

A KF is also used for the presented model to provide optimal estimation.

Full attitude (ψ, θ, ϕ) estimation using fusion models of Eqs. (5.19) and (5.21) is now provided. With these estimations one is able to provide all angles required for rotation matrices from Eqs. (4.6) and (4.7). Thus, any three dimensional frame transformation operation is measurable. Although not within the scope of this thesis, the sensor fusion techniques may prove useful for other applications, such as drones.

5.2.3 Accelerometer Data Integration Model

Integration and double integration may be performed on the measurement output of the accelerometer to obtain linear velocity and position, respectively. However, the noise and bias will also be part of the integration, regardless of how well calibrated the accelerometer is. For that reason, raw integration is not advised. It is possible nonetheless to designate a state space model that describes the integrations as well as the noises and biases for Kalman Filter estimation (This is not a fusion model since there is only one sensor at hand). Consider the following discrete model with sampling time T_s (bias is neglected)

$$\begin{aligned} \begin{bmatrix} p \\ v \\ a \end{bmatrix}_{k+1} &= \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ v \\ a \end{bmatrix}_k + w_k \\ y_k &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v \\ a \end{bmatrix}_k + v_k \end{aligned} \tag{5.23}$$

This model can be used as a position estimation model regarding one axis of the accelerometer. Its states $[p\ v\ a]^T$ correspond to position, velocity and acceleration, respectively. Brown e Hwang (1997) even provides a covariance matrix Q_k model for w_k based on the mathematics of Wiener (or Brownian motion). The covariance is given by

$$Q_k = q \begin{bmatrix} \frac{T_s^5}{20} & \frac{T_s^4}{8} & \frac{T_s^3}{6} \\ \frac{T_s^4}{8} & \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^3}{6} & \frac{T_s^2}{2} & T_s \end{bmatrix}, \tag{5.24}$$

with q being a tuning parameter. The covariance of v_k is estimated through actual measurements of the sensor at use. This model is practically ready for a KF implementation, since all the

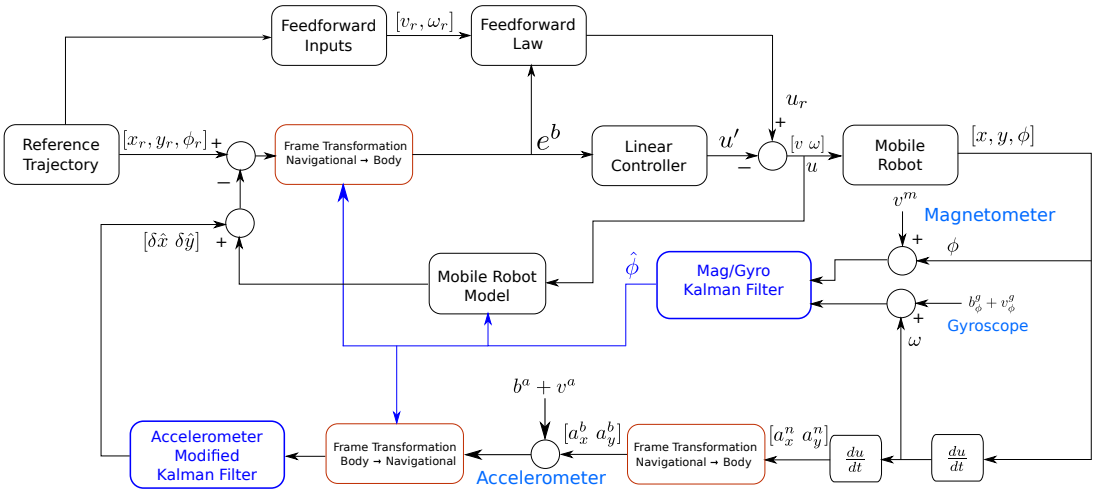
matrices of the model required are defined. An important observation - it is not possible to compute a steady-state KF since the model has its unstable poles not observable - observability matrix $\mathscr{O} = [C\ CA\ CA^2]^T$ is rank 1. However, the recursive KF presents no issues.

Note that the model is still susceptible to position drift due to its random walk nature or gravity if one considers that the mobile robot moves over an irregular terrain. Even with removal of gravity using Eq. (4.14), there may be many noise disturbances in the rotation matrix as its angle elements are also estimates. One must consider other strategies for correctly applying double integration. Applications note NXP Semiconductors (2007) suggest many useful approaches to solve that problem. Even though information may be lost by implementing some of its filters, the position estimation will be reliable enough for our mobile robot application.

6 CONTROL AND SENSOR FUSION RESULTS

This Chapter provides the results of both linear controller and sensor fusion techniques. Simulation regarding IMU sensors are performed with all of each sensors constraints included. We also compare a raw sensor reading with the KF output. The complete schematic of the sensor fusion is shown in Figure 18.

Figure 18 – Complete Control and Fusion Scheme



Source: The author.

Observe that the estimated $\hat{\phi}$ is being used in blocks that require the yaw angle. The bottom frame transformation is just to show that the accelerometer measurement is with respect to the body frame. Accelerometer KF is the same described by Eq. (5.23). Useful notes from NXP Semiconductors (2007) were considered and included in the accelerometer modifier Kalman Filter to avoid drift integration. The modified Kalman filter consists of a discriminator block that discard acceleration values below a certain threshold and a Zero Velocity Update (ZVU). ZVU accumulates past measured accelerations into a buffer and if all values are zero, the velocity state is set to zero. Although no drift will be accumulated, acceleration information will be lost, and not every external disturbance will be detected by the conditioner. The conditioner output $[\delta\hat{x}\ \delta\hat{y}]$ is a double integrated position estimation with respect to navigational frame that is added to the odometer model to perceive some disturbances that odometry alone does not consider.

KF from magnetometer and gyroscope is the same described by Eq. (5.19)

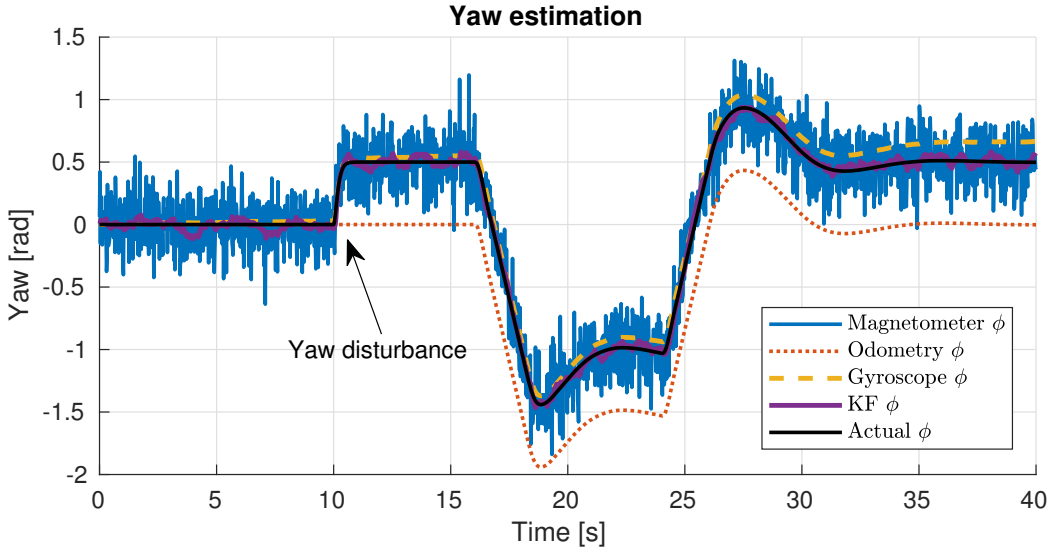
6.1 Simulation Results

Simulations were performed using Matlab Simulink environment.Both mobile robot and sensor fusion were integrated into one same simulation. The simulation allows for mobile robot reference specification, disturbance application, noise addition, and other features. The user can choose whether odometry, odometry plus magnetometer and gyroscope, odometry plus entire IMU or real feedback is used, and all the curves for each pose estimation is drawn.

6.1.1 Magnetometer Gyroscope Fusion

Eq. (5.19) models the magnetometer and gyroscope fusion. The following simulation describes a NWMR tracking two interconnected L-shaped curves. Gyroscope bias and magnetometer noise are similar to practical devices. Figure 19 presents the yaw ϕ estimations and its actual value during the simulation.

Figure 19 – Yaw ϕ estimation



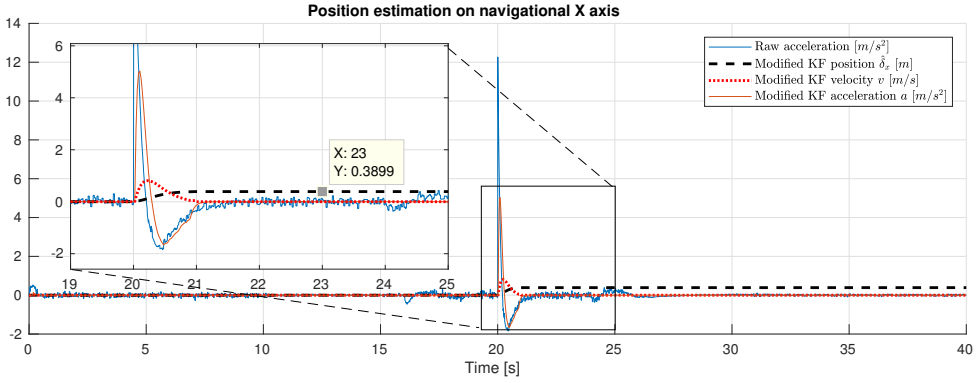
Source: The author.

In this case, an angular disturbance of 0.5 *rad*, or approximately 28.6 degrees, was applied at $t = 10s$. As expected, odometry-estimated ϕ is unable to cope with external angular disturbances while the sensors are able to. Notice the pure gyroscope estimated ϕ beginning to drift away by the end of the simulation. No matter how small the bias is, it will drift with time. Magnetometer provided noisy but drift-less data, and the KF estimation was very close to the real robot ϕ angle.

6.1.2 Accelerometer Integration and Conditioning

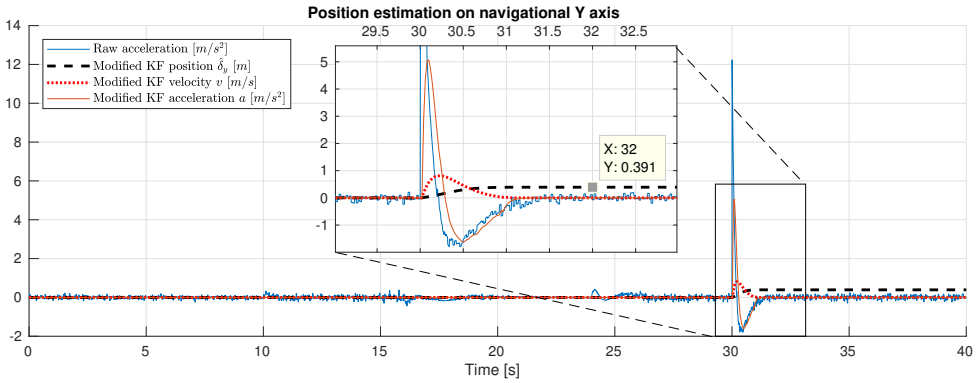
The accelerometer KF of Eq. (5.23) was used on the noisy and slightly biased accelerometer measuring output. The incorporated discriminator and ZVU blocks allows an additional tuning requirement so that drift is canceled. The two tuning parameters of these blocks are the discriminator acceleration threshold value and the number of elements in the ZVU block. After some trial and error, the value for discriminator threshold $d = 0.5\ m/s^2$ and buffer size $N = 5$ elements showed satisfactory performance. These additional, non-linear blocks were modeled using a S-Function alongside a KF embedded into a single block. At time $t = 20s$ a "kick" pushes the robot along navigational X direction for $0.5m$ m. At time $t = 30s$ another "kick" pushed the robot along navigational Y along for $0.5m$ as well. Figure 20 and 21 present the resulting estimations.

Figure 20 – Accelerometer position estimation on X^n



Source: The author.

Figure 21 – Accelerometer position estimation on Y^n



Source: The author.

The two $0.5m$ kicks were successfully detected by the accelerometer. As expected, the non-linear blocks that avoid drift filtered out some information inside the modified KF, whose integrated acceleration resulted in approximately $0.39m$ shift detection for the two kicks. A drawback is that small disturbances accelerations will be discarded by the discriminator block and the resulting pose shift will not be detected.

It is worth remarking that the aforementioned method is applied to the two measurement axes X^b and Y^b of an accelerometer. This means that the robot will effectively reject linear disturbances. However, since the odometry velocity v points to the same direction of body frame axis X^b, it is possible adding both odometry-estimated position and accelerometer-estimated position might end up overestimating the actual robot position along its body frame X^b since they are being added. A solution to this problem would be to include another model that copes with the two estimated positions along body frame axis X^b so that no overestimation arises. This will be covered in future work.

6.1.3 Complete Model (Control + Fusion)

Now the fusion models and the controller blocks are used altogether. The feedback pose is obtained from the gyroscope and magnetometer fusion block and modified accelerometer KF block. Comparison with the non-linear controller from Klancar *et al.* (2005) is included.

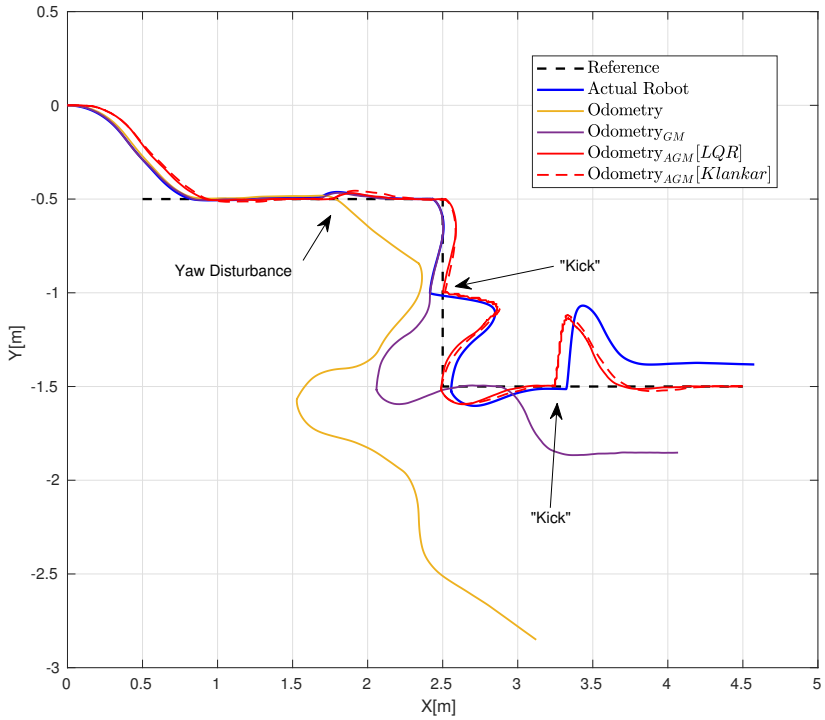
Table 1 – Kalman Filters and Controller parameters

LQR	Mag/Gyro Kalman Filter
$Q = diag(1,50,0)$ $R = I$	$Q_k = (5.24)\ q = 0.01\ T_s = 0.02s$ $R_k = 0.025$
Klankar	Accelerometer Kalman Filter
$\xi = 0.6$ $g = 40$	$Q_k = (5.20)\ q = 0.01\ T_s = 0.02s$ $R_k = 0.01$

Source: The author.

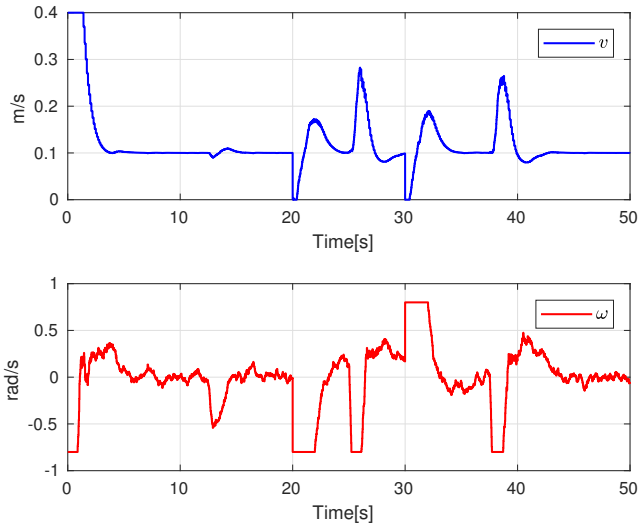
For the first test, two interconnected L-shaped curves of $2.8m$, $0.9m$ and $2.8m$, respectively, are drawn. The robot's feedforward velocities is set to $v_r = 0.1\ m/s$, $\omega_r = 0\ rad/s$ at all times, and no angular velocity. Table 1 shows the tuning parameters for LQR controller and Kalman Filters. Finally, the two "kicks" from the accelerometer modified Kalman filter are applied here as well. A controller signal saturation of $v_{max} = 0.4\ m/s$ and $\omega_{max} = 0.8\ rad/s$ was considered for a more realistic simulation. The reference trajectory tracking simulation result is shown in Figure 22, with velocities shown in Figure 23 and navigational errors in Figure 24.

Figure 22 – Mobile robot simulation - XY Plane



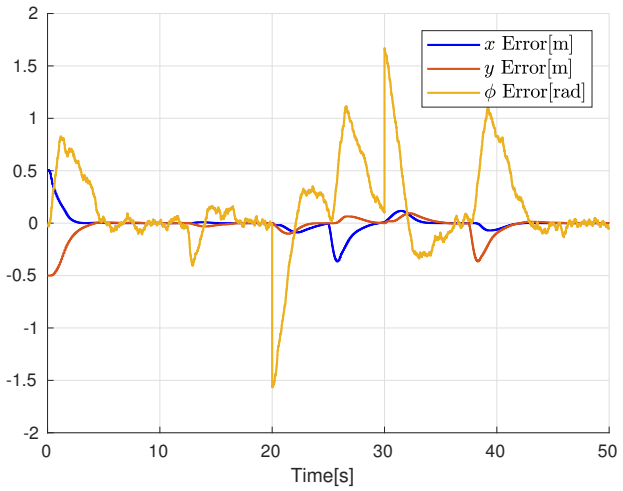
Source: The author.

Figure 23 – LQR Mobile robot simulation - Velocities



Source: The author.

Figure 24 – LQR Mobile robot simulation - Errors



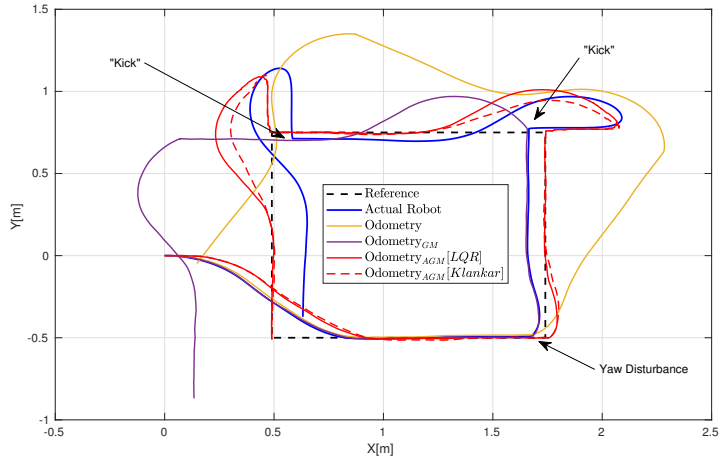
Source: The author.

The control is applied regarding estimates from the complete model labeled as $\text{Odometry}_{\text{AGM}}$ (Odometry plus accelerometer, gyroscope, magnetometer). The curve named $\text{Odometry}_{\text{GM}}$ do not consider accelerometer estimated positions, and curve named Odometry has no inertial navigation at all. The actual robot curve describe where the robot is after its disturbances. One may assume this pose is being acquired using some very precise equipment or measurements, and it expresses how far the estimates are from the actual robot pose.

While LQR is a simpler alternative controller than the non-linear controller, it has shown similar performance of that of its non-linear counterpart. The sensor fused controller with integrated accelerometer finished the trajectory much closer to the one without accelerometer. The angular disturbance was rejected by all fused controllers. The “kicks” were also rejected by the accelerometer integrated fused model, but not entirely due to the information loss of the drift rejection modified KF. In Figure 23 one can see the magnetometer noise effect on the controller signal, but not too detrimental. In Figure 24 one sees the controller successfully minimizing coordinate errors.

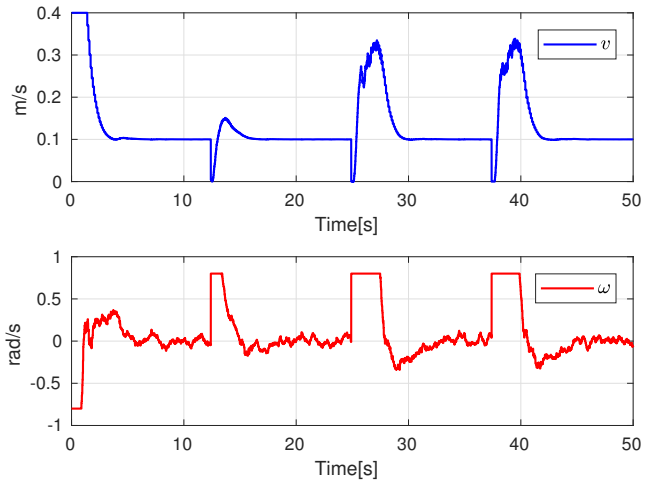
Another similar test was done with a square-shape curve. The same velocity reference $v_r = 0.1\ m/s$ was used. Figures 25, 26 and 27 display the results.

Figure 25 – Mobile robot simulation 2 - *XY* Plane



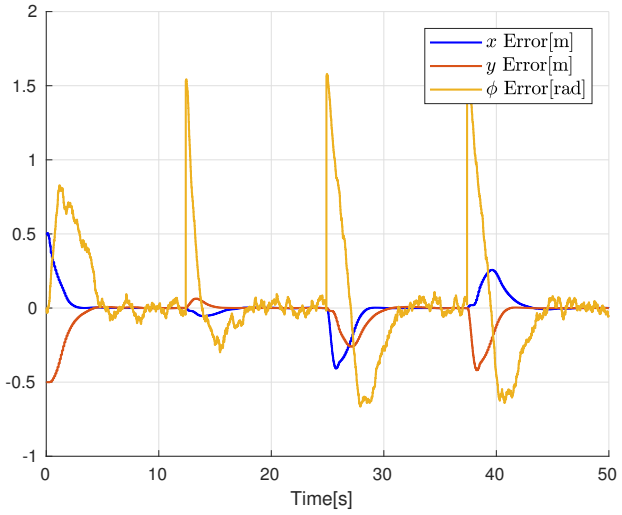
Source: The author.

Figure 26 – LQR Mobile robot simulation 2- Velocities



Source: The author.

Figure 27 – LQR Mobile robot simulation 2 - Errors

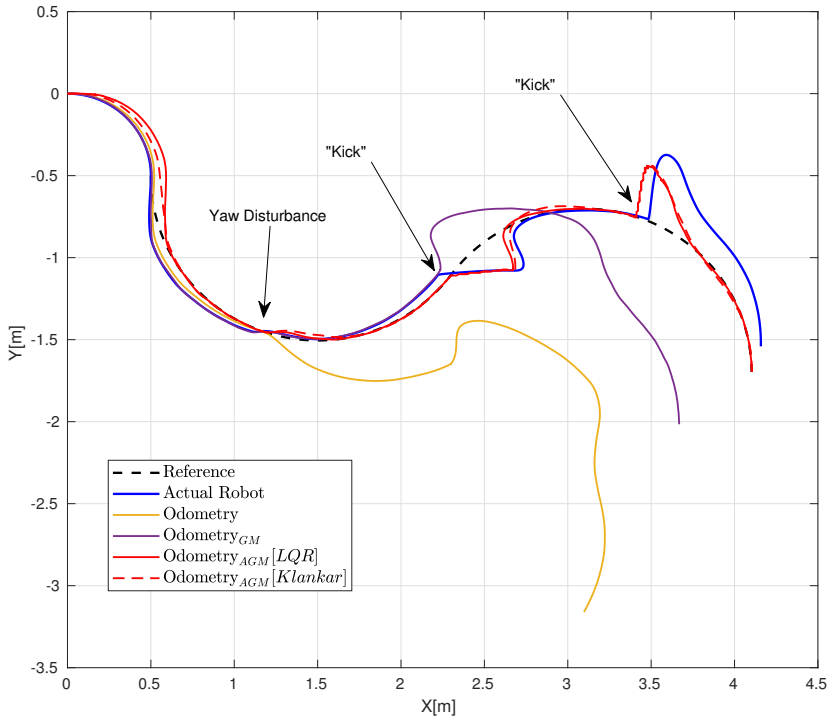


Source: The author.

Once again all the angular disturbances are rejected with fused models, and linear disturbances are rejected, although not entirely, with accelerometer integrated data. The non-linear controller showed a slightly better behavior compared to the linear controller, probably because its gain on the laterally error e_y is higher during the time of disturbance. If necessary, one could tune the LQR to reject e_y error more intensively by choosing a higher value on the α_y element of weight matrix Q, but it would implicate in a less robust and more aggressive controller.

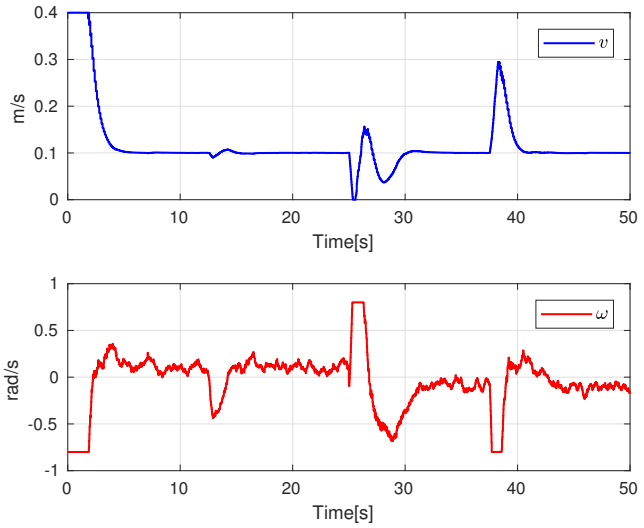
Finally, an S-curved-shape trajectory was tracked, with $v_r = 0.1\ m/s$ and $\omega_r = 0.1\ rad/s$ for half the time and $\omega_r = -0.1\ rad/s$ for the second half. The LQR tuning remained with its state matrix considering only linear velocity.

Figure 28 – Mobile robot simulation 3 - *XY* Plane



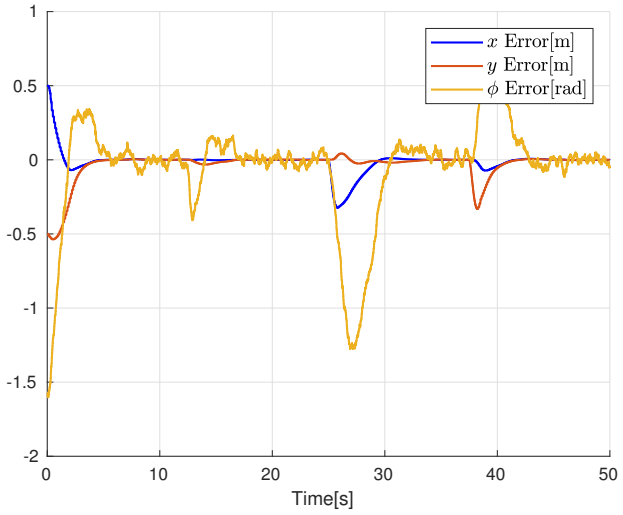
Source: The author.

Figure 29 – LQR Mobile robot simulation 3 - Velocities



Source: The author.

Figure 30 – LQR Mobile robot simulation 3 - Errors



Source: The author.

6.2 Experimental Results

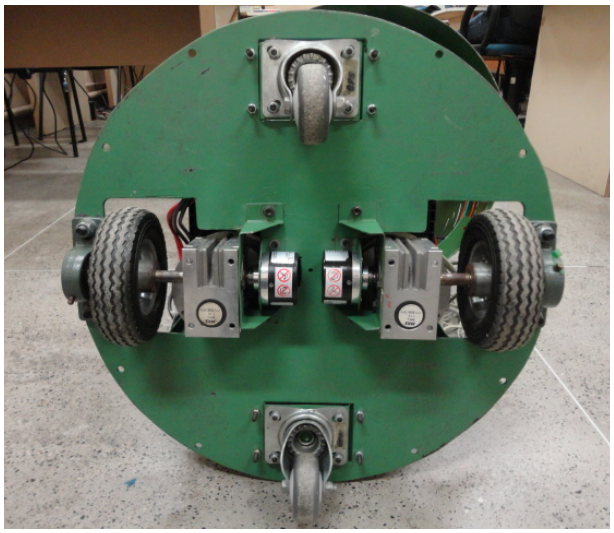
6.2.1 Real Robot Specifications

The test robot, whose name is a testament to its size and color, *Hulk*, weights about 70 Kg. It has a distance $L = 0.4m$ between its wheels and each active wheel has a radius of $R = 0.07m$. Details on constructive features of this robot can be seen in Sousa (2016). Figure 31 presents the robot and Figure shows its differential driving wheels.

Figure 31 – Test Mobile Robot *Hulk*

Source: The author.

Figure 32 – Test Mobile Robot *Hulk* wheels



Source: The author.

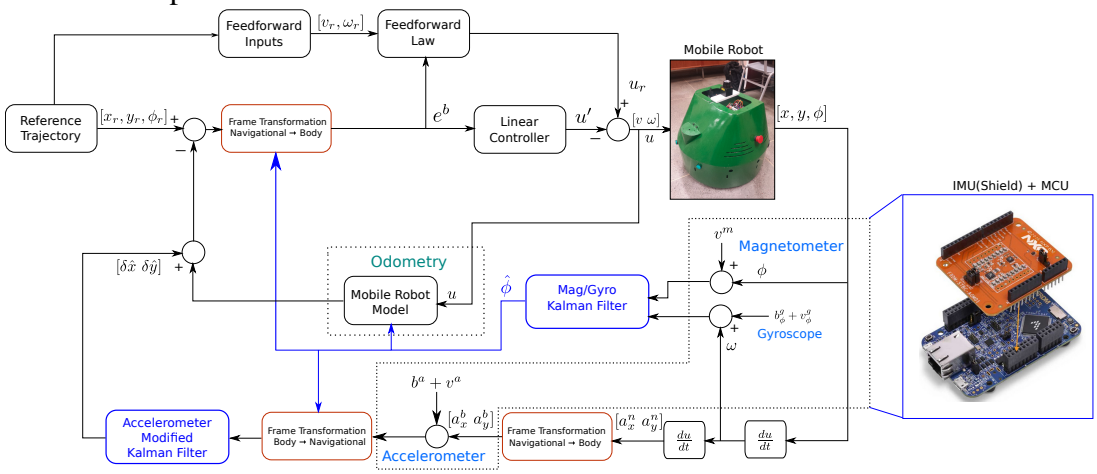
It comprises the following components

- Two 45AH lead-acid batteries, connected in series for 24 V power supply;
- Two 30A high torque DC motors;
- One 24V 150A dual-channel DC motor driver HDC2450;
- Two active wheels;

- Two passive wheels;
- Two 1200 pulses per revolution quadrature encoders;
- Electrical protection;
- One FRDM-K64F 120 Mhz 1 MB Flash Microcontroller Unit (MCU) with Bluetooth module;
- One BRKT-STBC-AGM01 IMU board with accelerometer, gyroscope and magnetometer;
- One(Optional) Hokuyo URG-04LX-UG01 Lidar scanner.

The tests were performed on an indoor laboratory. The laboratory proved quite challenging due to its highly slipping flood, which also contributed to the validation of the studied topics. Diagram from Figure 18 is redrawn to highlight experimental components of the system in Figure 33.

Figure 33 – Complete Control and Fusion Scheme



Source: The author.

6.2.2 Complete Model (Control + Fusion)

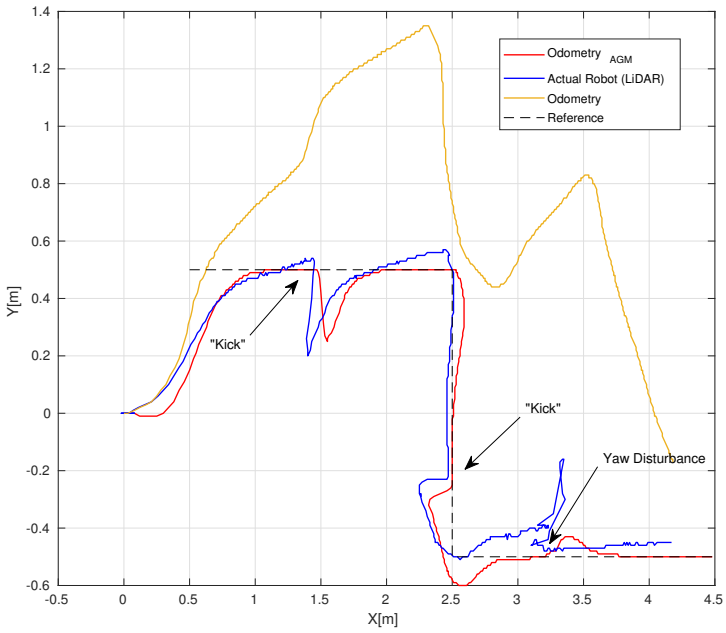
Experimental tests were performed using the same tuning parameters and for the LQR controller and both KF from simulations.

The experiment is the reference tracking of a two interconnected L-shaped curves but with smaller distances. The three paths are $2.5m$, $1m$ and $2.0m$ long. At time $t = 10s$ approximately a linear disturbance ("kick") is applied. At $t = 28s$ approximately another linear disturbance is applied. Finally, at $t = 36s$ an angular disturbance is applied by rotating the robot counterclockwise.

The actual mobile robot pose is acquired using a Light Detection and Ranging (LIDAR) sensor and a well established Self Localization and Mapping (SLAM) (KOHLBRECHER

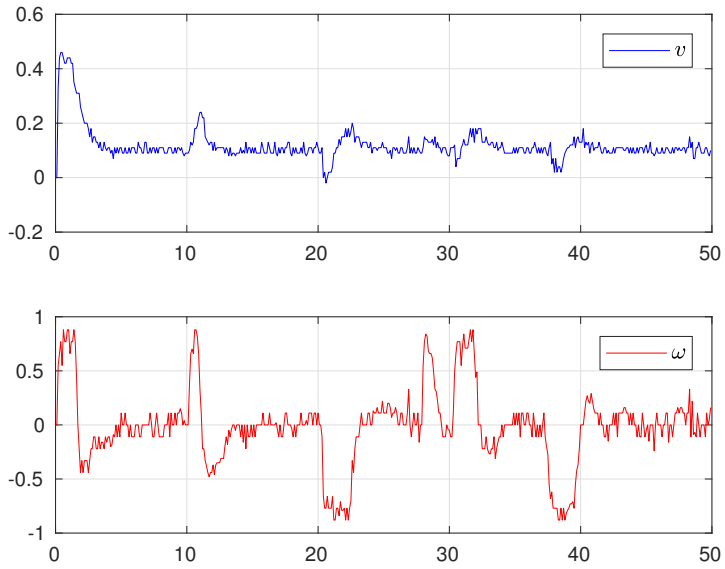
et al., 2011) algorithm. Although this technique is known to produce quite accurate pose estimates, it is still subjected to some uncertain behavior that may corrupt the estimated pose. During tests, for example, the performed “kicks” sometimes made the algorithm believe the robot was moving extremely fast for some reason, thus ruining pose estimates of the actual robot. Nonetheless, successful tests were achieved after devising a way to apply disturbances to the robot without interfering the readings from LIDAR.

Figure 34 – LQR Mobile robot experiment - *XY* Plane



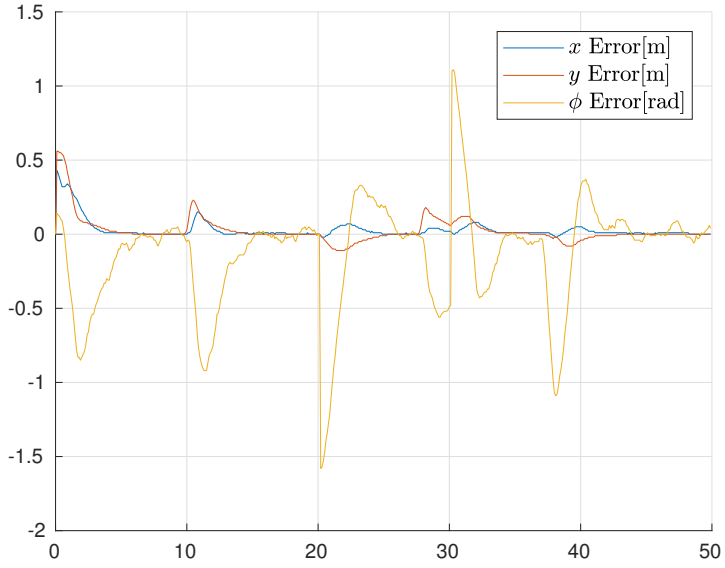
Source: The author.

Figure 35 – LQR Mobile robot experiment - Velocities



Source: The author.

Figure 36 – LQR Mobile robot experiment - Errors



Source: The author.

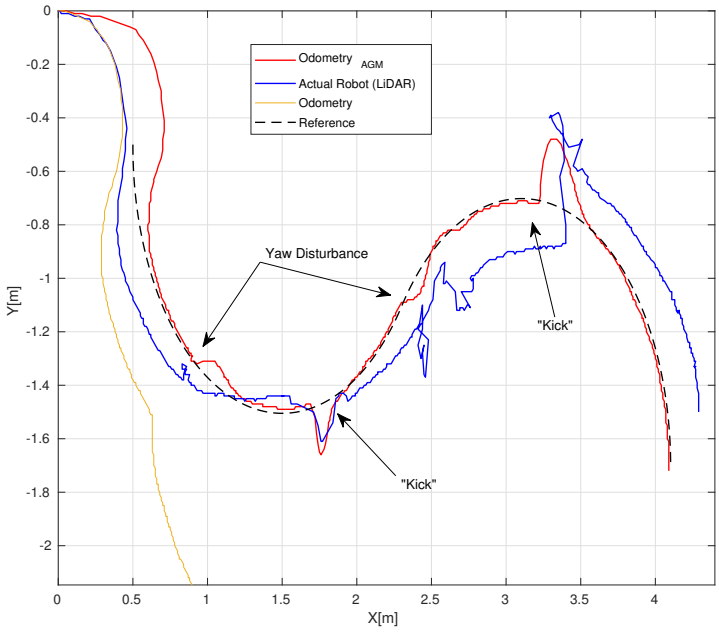
Experimental curves have shown similar results compared to simulations. The angular disturbance was successfully rejected by the magnetometer and gyroscope sensor fusion model. The controller also exhibited satisfactory response to the 90 degree sudden change on angular reference. The slightly oscillatory initial response is due to irregular floor on which the robot performed the experiment. The applied “kick” were also rejected, but not entirely, as

expected due to the loss of information from the discriminator and ZVU blocks from modified KF.

Notice the quite noisy behavior of the LIDAR data. Notice also that actual measurements from LIDAR do not describe the mobile robot dynamics, only its pose. This is clearly visible in Figure 34 near the application of the yaw disturbance, as the blue curve exhibits an unusual response. While not within the scope of this work, a sensor fusion between inertial data, odometry and LIDAR sensor is certainly an interesting strategy for a more accurate pose estimation than relying solely on LIDAR estimates. Nevertheless, the LIDAR estimates are assumed to be good representatives of the mobile robot actual pose.

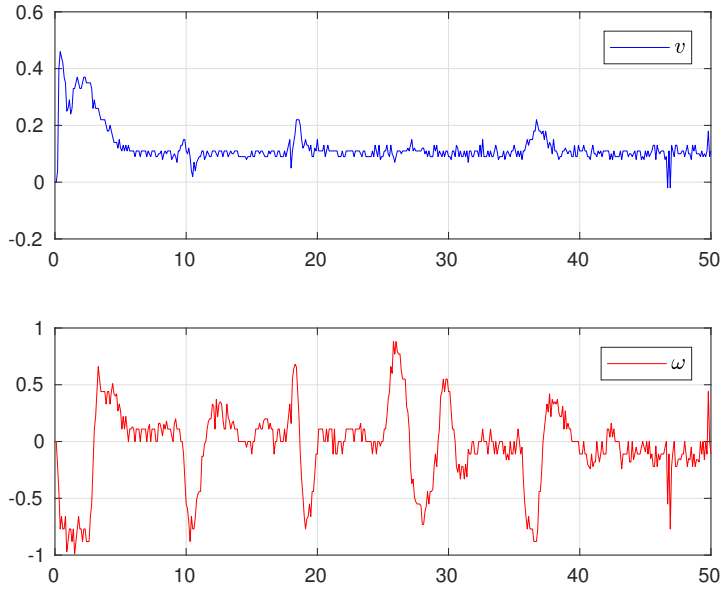
Another experimental setup was done, this time using the S-shaped curve from the simulations.

Figure 37 – LQR Mobile robot experiment 2 - XY Plane



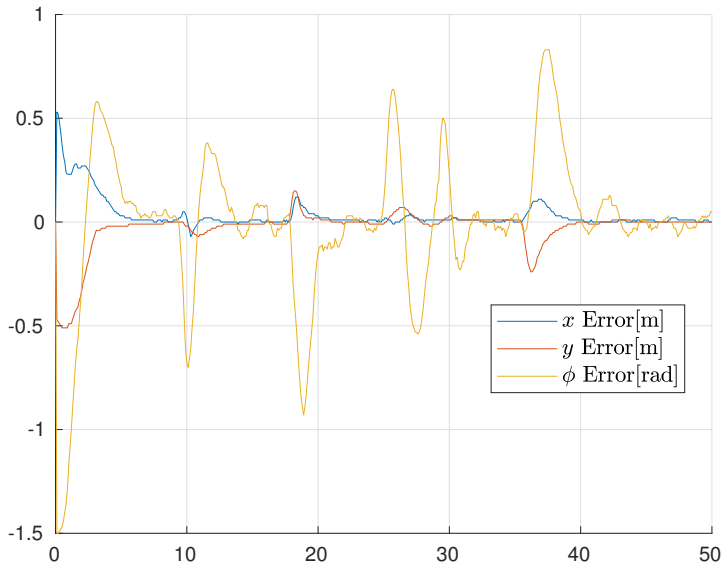
Source: The author.

Figure 38 – LQR Mobile robot experiment 2 - Velocities



Source: The author.

Figure 39 – LQR Mobile robot experiment 2 - Errors



Source: The author.

Two angular disturbances at times $t = 10s$ and $t = 26s$, and two "kicks" at times $t = 18s$ and $t = 36s$, were introduced along the path. The robot successfully rejected all disturbances and remained in the reference path. Once again, "kicks" were detected but not entirely rejected, assuming the LIDAR sensor is estimating correctly.

6.2.3 Practical Remarks

Since optometry is partially used to estimate mobile robot pose, a discrete model has to be programmed the MCU responsible for the entire control and fusion implementation. The model was presented in Eq. (2.5). The chosen sampling time for the computation of mobile robot pose was $10Hz$, as its a reasonable rate for closed loop performance and previously used in the same robot (LIMA *et al.*, 2016). The wheel velocity is converted into distance using $v = r\omega$, where r is the wheel radius and ω the wheel speed.

As mentioned in previous section, an IMU board was used to acquire inertial data and magnetic data. The board support different sampling frequencies up to $400Hz$, of which a $50Hz$ sampling rate was chosen. There is no particular reason for that choice - any frequency higher than the controller's $10Hz$ would be suitable. A frequency decimation of $1/5$ was implemented so that the controller of $10Hz$ would only be computed and executed after five samples of the IMU.

MCU communicates with the motor driver using RS-232 protocol and the necessary circuitry ICs. The data is written and read synchronously, which means that if either the MCU or the Motor Driver stop working, the program is loop-locked to avoid accidents.

IMU calibration should done before every test. For the magnetic calibration, the robot rotates around its own axis, acquire maximum and minimum readings on every axis and correct the ellipse-shape curve of measurements to the origin for Hard-Iron correction.Soft-iron correction is assumed to be negligible. Accelerometer and gyroscope calibration is done when robot is inert.

The KF used for the gyroscope-magnetometer fusion was implemented using Eq. (5.19). The gyroscope "contribution" to the fused yaw ϕ is its discrete integration of the measured angular velocity ω_{ϕ}^{g}. At any point the robot may have turned over its own axis more than once. While the magnetometer measurement belongs to the range $[-\pi, \pi]$ from the atan function Eq. (4.20), the integrated gyroscope might surpass that range. For that reason an additional processing is required to constantly compare the error between the two measurements compute the fused data while remaining in range $[-\pi, \pi]$. A code snippet is provided in Appendix A.

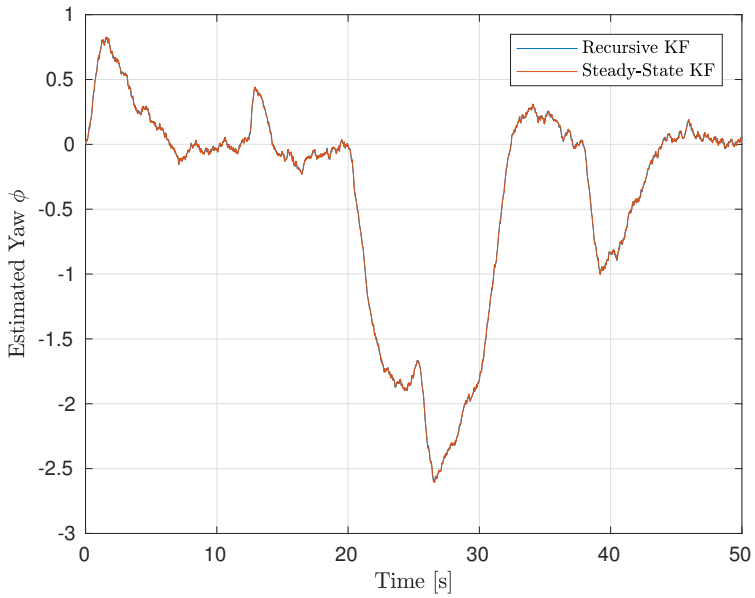
Still on the topic of gyroscope-magnetometer KF, the steady-state KF gain approach was preferred as opposed to the recursive one for two reasons. The first is that the implementation of a simple steady state KF gain is much faster and simpler. The second is that simulations have shown that both alternatives provides very similar estimations, as Figure 40 proves, which is

taken from simulation of two interconnected L-shape curve.

Recursive KF digital implementation is done as Section 5.1.1. Another code snippet with implementation in C language is provided in Appendix A. The adopted MCU FRDM-K64F manufacturer, NXP, supplies users with a powerful digital signal processing library for handling numerous otherwise complex operations, like matrix operations. The library was used to write the recursive KF code. Implementation requires a few temporary matrices and a previous initialization of each matrix structure with its rows, columns and data pointer.

Data is acquired in real time using Bluetooth module HC-06. It communicates with the MCU by a simple serial protocol and the paired computer receives it in an emulated serial port environment. Fortunately, Matlab has an object specific for this application, which facilitated the data acquisition from mobile robot.

Figure 40 – Comparison of steady-state and recursive KF



Source: The author.

7 CONCLUSION

A strategy for NWMR indoor navigation using IMU and odometry combined by sensor fusion was presented. The development of a kinematic and inverse kinematic model of the NWMR exhibited its non-linearity and challenges regarding the trajectory tracking control of such system. Nonetheless, a linearization was performed in order to design a linear control system for a specified performance.

The LQR controller was designed for the body frame error model so that the mobile robot reject any disturbance related to its pose. A feedforward control law was devised based on inverse kinematic that is responsible for the reference tracking. The problem was modeled as a regulation problem which the LQR regarding an error state vector which one aims to minimize along the trajectory tracking.

Then the mathematical model of sensors present in an IMU was discussed with details. Accelerometer, gyroscope and magnetometer sensors were introduced and mathematically modeled. With the models in hand, the sensor fusion was possible by means of a KF optimal estimator. The designed KF showed its great usefulness for models with process or measurement noise. Gyroscope and magnetometer were combined to provide a yaw ϕ angle for the robot and the accelerometer was used to estimate otherwise undetectable linear disturbances.

Finally, both controller and sensor fusion techniques were combined into the a single mobile robot model, and simulations and experiments were performed. The estimation of the yaw ϕ angle was critical for the computation of transformation matrices required, as well as the robot heading direction itself. The accelerometer double integration was a challenging task but handy to cope with drift issue.

The results were very satisfactory. The controller was able to reject any angular and linear disturbances, and showed similar performance to non-linear controllers. Initial conditions far from target reference and some measurement noise were not a problem for the controller.

7.1 Recommendations for Future Work

Much work is needed regarding the conditioning of accelerometer data to correctly estimate mobile robot linear disturbances. A model that combines accelerometer and odometry, instead of simply adding both models estimation, is needed to avoid overestimation. The author suggests the use of an EKF with the non-linear model of the mobile robot to fuse odometry

and LIDAR sensor. Also, a more detailed fusion of slow-rate data of a GPS and high-rate data IMU is important to be studied. Yet another suggested future work is the generalization of the used control strategy and sensor fusion for other robotic platforms, like drones or robotic arms. Furthermore, an LPV controller might be suitable for time-varying feedforward velocities that compose linear matrix A from linearized system so that the mobile robot maintains performance for different sets of velocities.

BIBLIOGRAPHY

AGUIAR, A. P.; DAčIć, D. B.; HESPANHA, J. P.; KOKOTOVIć, P. Path-following or reference tracking?: An answer relaxing the limits to performance. **IFAC Proceedings Volumes**, v. 37, n. 8, p. 167 – 172, 2004. ISSN 1474-6670. IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 5-7 July 2004. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1474667017319705>.

ALATISE, M. B.; HANCKE, G. P. Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter. **Sensors**, v. 17, 2017. ISSN 1424-8220. Disponível em: <http://www.mdpi.com/1424-8220/17/10/2164>.

ALPER, S. E.; AZGIN, K.; AKIN, T. High-performance soi-mems gyroscope with decoupled oscillation modes. **19th IEEE International Conference on Micro Electro Mechanical Systems**, p. 70–73, 2006.

ATRSAEI, A.; SALARIEH, H.; ALASTY, A.; ABEDINY, M. Human arm motion tracking by inertial/magnetic sensors using unscented kalman filter and relative motion constraint. **Journal of Intelligent & Robotic Systems**, v. 90, n. 1, p. 161–170, May 2018. ISSN 1573-0409. Disponível em: <https://doi.org/10.1007/s10846-017-0645-z>.

BARSHAN, B.; DURRANT-WHYTE, H. F. An inertial navigation system for a mobile robot. In: **Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on**. [S.l.: s.n.], 1993. v. 3, p. 2243–2248 vol.3.

BROWN, R. G.; HWANG, P. Y. C. **Introduction to random signals and applied kalman filtering: with MATLAB exercises and solutions; 3rd ed.** New York, NY: Wiley, 1997. Disponível em: <https://cds.cern.ch/record/680442>.

CHENAVIER, F.; CROWLEY, J. L. Position estimation for a mobile robot using vision and odometry. In: **Proceedings 1992 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 1992. p. 2588–2593 vol.3.

DINIZ, T. L. **In portuguese: Controle adaptativo auto-ajustável para controle de trajetórias de um robô móvel com rodas**. Dissertação (Mestrado) — Universidade Federal do Ceará, 7 2016.

DORF, R. C.; BISHOP, R. H. **Modern Control Systems**. 9th. ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2000. ISBN 0130306606.

FORTE, M. D.; CORREIA, W. B.; NOGUEIRA, F. G.; TORRICO, B. C. Reference tracking of a nonholonomic mobile robot using sensor fusion techniques and linear control. **IFAC-PapersOnLine**, v. 51, n. 4, p. 364 – 369, 2018. ISSN 2405-8963. 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control PID 2018. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2405896318303884>.

FRIEDLAND, B. **Control Systems Design: An Introduction to State-Space Methods**. [S.l.]: McGraw-Hill Higher Education, 1985. ISBN 0070224412.

HOVER, F. S.; TRIANTAFYLLOU, M. S. **System Design for Uncertainty**. 2010. Available at<https://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-electromechanical-robotic-systems-fall-2009/course-text/>.

JASSEMI-ZARGANI, R.; NECSULESCU, D. Extended kalman filter-based sensor fusion for operational space control of a robot arm. **IEEE Transactions on Instrumentation and Measurement**, v. 51, n. 6, p. 1279–1282, Dec 2002. ISSN 0018-9456.

JULIER, J. K. U. S. J. New extension of the kalman filter to nonlinear systems. **Proc.SPIE**, v. 3068, p. 3068 – 3068 – 12, 1997. Disponível em: <https://doi.org/10.1117/12.280797>.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Transactions of the ASME–Journal of Basic Engineering**, v. 82, n. Series D, p. 35–45, 1960.

KAM, M.; ZHU, X.; KALATA, P. Sensor fusion for mobile robot navigation. **Proceedings of the IEEE**, v. 85, n. 1, p. 108–119, Jan 1997. ISSN 0018-9219.

KANAYAMA, Y.; KIMURA, Y.; MIYAZAKI, F.; NOGUCHI, T. A stable tracking control method for an autonomous mobile robot. In: **Proceedings., IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 1990. p. 384–389 vol.1.

KANG, D.; LUO, R. C.; HASHIMOTO, H.; HARASHIMA, F. Position estimation for mobile robot using sensor fusion. In: **Multisensor Fusion and Integration for Intelligent Systems, 1994. IEEE International Conference on MFI '94.** [S.l.: s.n.], 1994. p. 647–652.

KAPITANYUK, Y. A.; PROSKURNIKOV, A. V.; CAO, M. A guiding vector-field algorithm for path-following control of nonholonomic mobile robots. **IEEE Transactions on Control Systems Technology**, v. 26, n. 4, p. 1372–1385, July 2018. ISSN 1063-6536.

KHATIB, E. I. A.; JARADAT, M. A.; ABDEL-HAFEZ, M.; ROIGARI, M. Multiple sensor fusion for mobile robot localization and navigation using the extended kalman filter. In: **2015 10th International Symposium on Mechatronics and its Applications (ISMA)**. [S.l.: s.n.], 2015. p. 1–5.

KLANCAR, G.; MATKO, D.; BLAZIC, S. Mobile robot control on a reference path. In: **Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.** [S.l.: s.n.], 2005. p. 1343–1348. ISSN 2158-9860.

KOHLBRECHER, S.; STRYK, O. von; MEYER, J.; KLINGAUF, U. A flexible and scalable slam system with full 3d motion estimation. In: **2011 IEEE International Symposium on Safety, Security, and Rescue Robotics**. [S.l.: s.n.], 2011. p. 155–160. ISSN 2374-3247.

KOK, M.; HOL, J. D.; SCHÖN, T. B. Using inertial sensors for position and orientation estimation. **CoRR**, abs/1704.06053, 2017. Disponível em: <http://arxiv.org/abs/1704.06053>.

LEE, T.; SHIN, J.; CHO, D. Position estimation for mobile robot using in-plane 3-axis imu and active beacon. In: **2009 IEEE International Symposium on Industrial Electronics**. [S.l.: s.n.], 2009. p. 1956–1961. ISSN 2163-5137.

LIMA, M. V. P.; SILVA, P. R. M.; QUIROZ, C. H. C.; KURKA, P. R. G. Neural network regularization of an inertial odometry estimation for position control of a mobile robot. In: **2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)**. [S.l.: s.n.], 2017. p. 1–6.

LIMA, T. A.; FORTE, M. D. do N.; NOGUEIRA, F. G.; TORRICO, B. C.; PAULA, A. R. de. Trajectory tracking control of a mobile robot using lidar sensor for position and orientation estimation. In: **2016 12th IEEE International Conference on Industry Applications (INDUSCON)**. [S.l.: s.n.], 2016. p. 1–6.

LIN, P.-C.; KOMSUOGLU, H.; KODITSCHEK, D. E. Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits. **IEEE Transactions on Robotics**, v. 22, n. 5, p. 932–943, Oct 2006. ISSN 1552-3098.

LIU, H. H. S.; PANG, G. K. H. Accelerometer for mobile robot positioning. **IEEE Transactions on Industry Applications**, v. 37, n. 3, p. 812–819, May 2001. ISSN 0093-9994.

MEMS Accelerometer. <http://www.instrumentationtoday.com/mems-accelerometer/2011/08/>. Accessed: 2018-07-25.

NEHMZOW, U. **Mobile Robotics: A Practical Introduction**. [S.l.]: Springer, 2003.

NXP SEMICONDUCTORS. **Implementing Positioning Algorithms Using Accelerometers**. [S.l.], 2007. Rev. 0.

NXP SEMICONDUCTORS. **Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference**. [S.l.], 2015. Rev. 4.

NXP SEMICONDUCTORS. **Layout Recommendations for PCBs Using a Magnetometer Sensor**. [S.l.], 2015. Rev. 4.

NXP SEMICONDUCTORS. **Quaternion Algebra and Rotations**. [S.l.], 2015. Rev. 1.

OGAWA, M. A. **In portuguese: Controle preditivo aplicado ao seguimento de trajetória de robô móvel com rodas**. Dissertação (Mestrado) — Universidade Federal do Ceará, 4 2014.

PHAM, T. T.; LE, D. H.; NGUYEN, C.-N.; NGUYEN, T. D.; TRAN, C. C. Optimizing the structure of rbf neural network-based controller for omnidirectional mobile robot control. In: **2017 International Conference on System Science and Engineering (ICSSE)**. [S.l.: s.n.], 2017. p. 313–318.

PONS, N.; BOURDON, G.; DELAPLACE, S. Contribution of fuzzy and feedback control to a mobile robot adaptative navigation. In: **1994 Proceedings of IEEE International Conference on Control and Applications**. [S.l.: s.n.], 1994. p. 65–69 vol.1.

RAOL, J. R. **Multi-Sensor Data Fusion with MATLAB**. 1st. ed. Boca Raton, FL, USA: CRC Press, Inc., 2009. ISBN 1439800030, 9781439800034.

SALICHS, M. A.; PUENTE, E. A.; GACHET, D.; MORENO, L. Trajectory tracking for a mobile robot-an application to contour following. In: **Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON '91., 1991 International Conference on**. [S.l.: s.n.], 1991. p. 1067–1070 vol.2.

SILVA, C. S.; WIMALARATNE, P. Towards a grid based sensor fusion for visually impaired navigation using sonar and vision measurements. In: **2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)**. [S.l.: s.n.], 2017. p. 784–787.

SOUSA, R. L. S. **In portuguese: Desenvolvimento de um robô móvel não-holonômico com controlador não-linear para seguimento de trajetórias**. Dissertação (Mestrado) — Universidade Federal do Ceará, 9 2016.

STMICROELECTRONICS. **Ellipsoid or sphere fitting for sensor calibration**. [S.l.], 2016. Rev. 2.

TZAFESTAS, S. G. 2 - mobile robot kinematics. In: TZAFESTAS, S. G. (Ed.). **Introduction to Mobile Robot Control**. Oxford: Elsevier, 2014. p. 31 – 67. ISBN 978-0-12-417049-0. Disponível em: <http://www.sciencedirect.com/science/article/pii/B978012417049000002X>.

WAN, E. A.; MERWE, R. V. D. The unscented kalman filter for nonlinear estimation. In: **Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)**. [S.l.: s.n.], 2000. p. 153–158.

WANG, J.; CHEPINSKIY, S. A.; KRASNOV, A. J.; ZHANG, B.; LIU, H.; CHEN, Y.; KHVOSTOV, D. A. Geometric path following control for an omnidirectional mobile robot. In: **2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)**. [S.l.: s.n.], 2016. p. 1063–1068.

WANG, P.; SEKIYAMA, K. Curvature based velocity control system for mobile robot. In: **IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society**. [S.l.: s.n.], 2015. p. 005211–005216.

XUE, G.; WANG, J.; CHEN, Q.; REN, X. Position estimation for planar servo mechanical systems based on sensor data fusion. In: **2017 9th International Conference on Modelling, Identification and Control (ICMIC)**. [S.l.: s.n.], 2017. p. 488–493.

APPENDIX A – IMPLEMENTATION CODE SNIPPETS

```
//IMU data is read before this function
float GetFusedYaw(const IMUData_t* Data){
    static const float K[] = {0.0604, -0.0868}; //Kalman Gain
    static float YAW=0,BIAS=0;
    float error;
    //yk
    float MAG = GetMagYaw(Data); //Compute atan2 from By,Bx
    //xk = Axk_-1 + Buk_-1 (Predict)
    YAW = YAW + (Data->Wz-Data->OffWz)*GyroResolution*Deg2Rad*dt - BIAS*dt;

    error = MAG-Wrap2PI(YAW); //Compute correct error (yk - xk')
    if(fabs(error) > PI){
        if(error > 0.0)
        error = error - 2*PI;
        else
        error = error + 2*PI;
    }

    //xk = xk' + K(yk - xk') (Update)
    YAW = YAW + K[0]*(error);
    BIAS = BIAS + K[1]*(error);

    YAW = Wrap2PI(YAW);
    return YAW;
}
```

Figure 41 – Code snippet for gyroscope and magnetometer angle processing

```
//Parameters -> Control signal, Output measurement, Kalman Structure with matrices (A,B,C,Qn,...)
void KalmanFilter( float* system_input, float* system_output,KalmanLinear_t* Kalman){
    Kalman->U.pData = system_input; //Control signal
    Kalman->Y.pData = system_output; //Output measurement

    //PREDICT
    //x = Ax + Bu
    arm_mat_mult_f32(&Kalman->A, &Kalman->X, &Kalman->X_est); // X = A*X
    arm_mat_mult_f32(&Kalman->B, &Kalman->U, &Kalman->tmp0); //tmp = B*U
    arm_mat_add_f32(&Kalman->X_est, &Kalman->tmp0, &Kalman->X_est); //Xe = A*X + B*U
    //Pk = APkA' + Qn
    arm_mat_mult_f32(&Kalman->A, &Kalman->Pk, &Kalman->Pk); // Pk = A*Pk
    arm_mat_trans_f32(&Kalman->A, &Kalman->tmp1); // tmp1 = A'
    arm_mat_mult_f32(&Kalman->Pk, &Kalman->tmp1, &Kalman->Pk); // Pk = A*Pk*A'
    arm_mat_add_f32(&Kalman->Pk, &Kalman->Qn, &Kalman->Pk); // Pk = A*Pk*A' + Qn

    //UPDATE
    //y = y - Cx
    arm_mat_mult_f32(&Kalman->C, &Kalman->X_est, &Kalman->Y_est); // Yest = C*Xest
    arm_mat_sub_f32(&Kalman->Y, &Kalman->Y_est, &Kalman->Y_est); // Yest = Y - CXest
    //S = C*Pk*C' + Rn
    arm_mat_mult_f32(&Kalman->C, &Kalman->Pk, &Kalman->tmp2); //C*Pk
    arm_mat_trans_f32(&Kalman->C, &Kalman->tmp5); // C'
    arm_mat_mult_f32(&Kalman->tmp2, &Kalman->tmp5, &Kalman->tmp3); // C*Pk*C'
    arm_mat_add_f32(&Kalman->tmp3, &Kalman->Rn, &Kalman->tmp3);// C*Pk*C' + Rn
    //S = inv(S);
    arm_mat_inverse_f32(&Kalman->tmp3, &Kalman->tmp4);
    //K = Pk*C'*S
    arm_mat_mult_f32(&Kalman->Pk, &Kalman->tmp5, &Kalman->tmp5); // Pk*C'
    arm_mat_mult_f32(&Kalman->tmp5, &Kalman->tmp4, &Kalman->Kk); // Kk = Pk*C'*inv(S);
    //X = X + Ky
    arm_mat_mult_f32(&Kalman->Kk, &Kalman->Y_est, &Kalman->tmp0);
    arm_mat_add_f32(&Kalman->X_est, &Kalman->tmp0, &Kalman->X);
    //Pk = (I- K*C)*Pk
    arm_mat_mult_f32(&Kalman->Kk, &Kalman->C, &Kalman->tmp1);
    arm_mat_sub_f32(&Kalman->I, &Kalman->tmp1, &Kalman->tmp1);
    arm_mat_mult_f32(&Kalman->tmp1, &Kalman->Pk, &Kalman->Pk);
    //Resulting states are stored in Kalman.X.pdata[];
}
```

Figure 42 – Code snippet KF implementation