# SLIPPAGE ESTIMATION OF A TWO WHEELED MOBILE ROBOT USING DEEP NEURAL NETWORK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

İSMAİL ÖZÇİL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2018

Approval of the thesis:

**SLIPPAGE ESTIMATION OF A TWO WHEELED MOBILE ROBOT USING DEEP NEURAL NETWORK**

submitted by **İSMAİL ÖZÇİL** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. M. A. Sahir Arıkan
Head of Department, **Mechanical Engineering**

Assoc. Prof. Dr. E. İlhan Konukseven
Supervisor, **Mechanical Engineering, METU**

Assist. Prof. Dr. A. Buğra Koku
Co-supervisor, **Mechanical Engineering, METU**

**Examining Committee Members:**

Assist. Prof. Dr. Kıvanç Azgın
Mechanical Engineering, METU

Assoc. Prof. Dr. E. İlhan Konukseven
Mechanical Engineering, METU

Assist. Prof. Dr. A. Buğra Koku
Mechanical Engineering, METU

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Engineering, METU

Assist. Prof. Dr. Andaç Töre Şamiloğlu
Mechanical Engineering, Başkent University

Date: 07.09.2018

## ABSTRACT

## SLIPPAGE ESTIMATION OF A TWO WHEELED MOBILE ROBOT USING DEEP NEURAL NETWORK

Özçil, İsmail
M.S., Department of Mechanical Engineering
Supervisor : Assoc. Prof. Dr. E. İlhan Konukseven
Co-Supervisor : Assist. Prof. Dr. A. Buğra Koku

September 2018, 59 pages

Mobile robot navigaiton is an important task for the operations of the mobile robots. Due to the wheel slippages, performance of the dead reckoning in estimating speed of the robot and the position of the robot is not sufficient. To overcome the errors in navigation estimates, usage of the recurrent deep neural networks is porposed. Neural networks are used to understand the behaviour of the linear and nonlinear systems. Since wheel-ground interaction will be modeled with non-linear models and the estimating parameters of those models are difficult, usage of the neural networks is preferable since they do not require system models and parameters. In this work, a recurrent deep neural network is proposed to estimate the speed and yaw angle of the 2 wheeled differentially driven mobile robot. By recording data from the training experiments of the navigation of the mobile robot, network is trained. After that, performance of the network is evaluated by plotting and tabulating outputs of the network, sensor data calculation and ground truth. Finally, results are compared with the results from the literature.

Keywords: mobile robot, ground vehicle, dead reckoning, navigation, recurrent deep neural network

# ÖZ

## DERİN SİNİR AĞI KULLANILARAK İKİ TEKERLEKLİ MOBİL ROBOTUN TEKERLEK KAYMALARININ TAHMİN EDİLMESİ

Özçil, İsmail
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. E. İlhan Konukseven
Ortak Tez Yöneticisi : Yrd. Doç. Dr. A. Buğra Koku

Eylül 2018 , 59 sayfa

Hareketli robot yöngüdümü ve konumlandırılması hareketli robotlar için önemlidir. Tekerleklerdeki kaymalara bağlı olarak yalnızca enkoder ve ataletsel ölçüm birimleri ile yapılan gözü kapalı konum hesapları ve hız hesapları yüksek hatalara sahip olmaktadır. Bu hataları azaltmak için geri beslemeli yapay sinir ağlarının kullanımı önerilmiştir. Tekerlek ile yer arasında olan etkileşim denklemleri doğrusal olmadığı, bu denklemlerdeki katsayıların kestirimi zor olduğu ve böyle durumlarda yapay sinir ağlarının kullanımı herhangi bir sistem modeli veya denklem gerektirmediği için yapay sinir ağlarının kullanımı tercih edilebilmektedir. Bu çalışmada iki tekerlekli diferansiyel sürüşlü robotun hızını ve yalpa açısını tahmin edebilmek için geri beslemeli yapay sinir ağının kullanımı önerilmiştir. Alıştırma deneylerindeki robotun üzerindeki algılayıcılar ile toplanan veriler kullanılarak bu yapay sinir ağı eğitilmiştir. Sonrasında eğitimde kullanılmayan deneylerin verileri kullanılarak, yapay sinir ağının hız ve yalpa açılarındaki başarımı yer doğrulaması, enkoder ve ataletsel ölçüm birimi ile kıyaslanıp değerlendirilmiştir.

Anahtar Kelimeler: mobil robot, yer aracı, gözü kapalı tahmin, yöngüdüm, geri beslemeli yapay sinir ağı

To my family...

# ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, Dr E. İlhan Konukseven and my co-supervisor Dr. A. Buğra Koku for their help, guidance, expertise and kindness. With their expertise and guidance I learned about mechatronics. They also provided me the resorces of Mechatronics Laboratory to conduct experiments about my thesis.

I would like to thank to members of Mechatronics laboratory for their patience and friendship.

I acknowledge my family for all their support.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

For the indoor and outdoor wheeled mobile robots, position information of the robot is important since all tasks depend on positioning of the robot. For the indoor applications, vision information is used to calculate position of the robot, but there may not be available camera getting image of whole working ground of the robot, or the camera on the robot may not get known objects to determine position every time. Also, image processing algorithms are dependent on illumination of the environment and illumination may not be same through the room. For the outdoor applications, GPS is widely used, however it may not be available according to Ward and Iagnemma due to surrounding buildings, trees or weather conditions [1]. Frequency of the GPS data may not be sufficient or ground truth data may be interrupted. During those times, dead reckoning is used. Dead reckoning is estimating position of the mobile robot by using on-board sensors of the mobile robot like IMU, encoder, etc. However, using those sensors over a time, errors start to propogate and position and velocity estimates starts to have large error due to the sensor drifts and wheel slippages. Hence, improving performance of dead reckoning algorithms are important to make better estimations where ground truth data is not supplied to the robot.

Also, frequency of the GPS data may not be sufficient for slip detection and anti-slippage applications. Slippage is directly affects the trajectory robot is following, hence it should be compensated to follow the desired trajectory. Moreover, sensing and compensating wheel slip also improves traction of the robot and it may prevent unnecessary power usage where wheels of the robot are slipping. Hence, motivation of the thesis is to estimate wheel slippages and improve speed and yaw angle estimations during the robot operation. By this way, at the instances where absolute position data of the robot are not available, robot motion information will have less error.

## 1.2 Literature Survey

Using only odometry data to estimate position and velocity of the robot causes high errors since odometry is unable to detect wheel slips. Moreover, error on the position estimation using odometry is subjected to cumulative error, which makes estimations with large errors as measurement time increases. Chenavier and Crowley [2] used mobile robot with a camera mounted on the robot in an environment which contains some known objects with known positions to determine the position of the mobile robot. Gustafssont [3] used non-driven wheels to determine actual velocity of the vehicle. But, this method cannot be applied on all the mobile robots since there are robots with all of their wheels are driven. Seyr and Jakubek [4] used odometry and IMU to improve position estimates. First, they compare measurements done by the IMU and odometry, then if the difference is too high, they assume wheels of the robot is slipping. In the slipping case, they give more weight to IMU measurements while calculating position of the robot, but if the difference is low, they assume slip in the wheels are small, and they give more weight to odometry while calculating position of the robot. Chonnaparamutt, Winai and Kawasaki [5] used fuzzy estimator to estimate postion of a pruning robot. They also used a fuzzy controller to control speed of the robot. Ward and Iagnemma [1] used GPS data to improve velocity estimates of the robot, they update their velocity estimates at nearly 1 Hz, when they got GPS data. Hwang et al.[6] used images of a object taken by the on-board camera of the mobile robot to correct velocity and position estimates done by the odometry and IMU. Ojeda et al.[7] used current drawn by the DC motors connected to wheels to determine the slippage and amount of the slippage. Their method requires a mobile robot with at least 4 driven wheels. While cruising a constant speed, they increase speed of the one of the wheels and record speed of the wheel and current drawn by that wheel. By this way, they can determine the current drawn and velocity difference function for that surface, and they can use that function to estimate slip later. One disadvantage of their method is that method only compensates longitudinal tire slip, lateral slip cannot be determined by the current drawn by the motor. Other disadvantage is the function they found for current drawn and velocity difference is linear for outdoor terrain. For indoor applications and constructed surfaces like asphalt coated roads, there is not such function. On those surfaces, nonlinear functions obtained by Dugoff Formula [8] and Magic Formula [9] are used and obtaining parameters for those formulae are difficult in the online applications. Bayar et al.[10] used camera positioned above the working area of a 2 wheeled mobile robot to determine the coefficients of the traction, lateral and roll force equations derived in previous works. Later, they used those coeeficients to determine position of the robot. Moreover, they showed that in different surfaces, current profile of the DC motors connected to wheels differs. Zabaleta et al.[11] used optical mouse sensors to improve position estimations of the rehabilitation robot they used in thier work. However, this robot requires to work on a special textured surface. Bonarini and Matteucci [12] presented equations for a robot using 2 optical mouses to improve position estimation, but they only presented modelling, there is not any experiments. Sekimori and Miyazaki [13] used multiple optic mouses to get position of the mobile robot. Optical mouses can be used to improve position estimations, but optical mouse sensors should be positioned in a certain narrow height from the surface, installation and operation of those sensors in unstructured surfaces is not ideal since road surface may not be smooth and distance from sensor to ground may constantly change. Fujimoto et al. [14] used yaw moment observer to estimate cornering stiffness of an electric vehicle. But this method requires model of the vehicle, and model parameters may not be available for a small robot for indoor applications. Matuso [15] used neural networks to estimate tire-road friction force. This arcticle shows that neural networks are useful with the non-linear systems with high number of unknowns. Hence for robotic applications with small robots where parameters of the robot is unknown, neural netwoks can be used. Melzi and Sabbioni[16] used a simple neural network with one hidden layer and 10 sigmoid neurons to determine slip angle of a vehicle. They used 3 neural networks in that structure, but they changed number of inputs for second and third network. In the end, they compared performances of those networks. Cirovic et al.[17] used neural network to adjust brake actuation pressure according to the adhesion coefficient between the tire and road. Tahami et al.[18] used neural network to generate yaw angle reference to the fuzzy controller that control the torque and slip of the wheels in case of an emergency like obsticle avoidance for an all-wheel drive electric vehicle.

## 1.3 Outline of the Thesis

In this chapter, motivation of the thesis and the previous work done related to this thesis are presented. In the next chapter, two wheeled differentially driven robot used in the experiments, the experimental set-up, set-up components, sensors used in the experiments and the basic equations used to manipulate sensor data are presented. Also, results of the experiments obtained by different sensor data are plotted. In the third chapter, neural networks and structure of two of the basic deep neural networks explained. Later, structure, inputs and the training of the deep recurrent neural network used in this work is explained. In the fourth chapter, estimated speed, covered distance, estimated yaw angle and estimated position of the robot calculated using odometry data, IMU data and results of the proposed deep recurrent neural network is plotted and compared. In the last chapter, results, discussion and future work is presented.

# CHAPTER 2

# EXPERIMENTAL SET-UP

## 2.1 Introduction

In this chapter, robot used in the slip detection and used to collect data is introduced and schematic of the robot is represented. The set-up and test ground of the robot is explained, also set-up components are introduced. Image processing algorithm used to get position of the robot and used as ground truth is explained. Basic equations to manipulate sensor data collected from the robot are represented. In the end of the chapter, using those equations, results of a experiment are represented to compare measurements done by different sensors.

To obseve effects of wheel slip on the position estimates of a mobile robot, a set-up consisting a 2 wheeled mobile robot and a camera recording the motion of the robot are used. Camera is used to make ground truth and to compare inertial measurements and odometry to ground truth to evaluate their performance. Later, this ground truth data are used to train deep neural network to make position estimations from the on-board sensors of the mobile robot. Experiments are done on a flat surface, i.e. there is no inclination on the surface which robot operates. During the experiments, surface properties are not changed and mechanical parameters of the mobile robot like wheels, wheel radii, center of mass, mass, inertia are not changed. Since changing one or more of those parameters will change system model, predicted model will not work after that point.

During the experiments, to see effects of both lateral and longitudinal wheel slip, different angular velocity ramp input are sent controllers of the DC motors connected to wheels, hence robot will make circular motion or near-circular motion in presence of high wheel slips. In different experiments, different ramp inputs are sent to wheels.

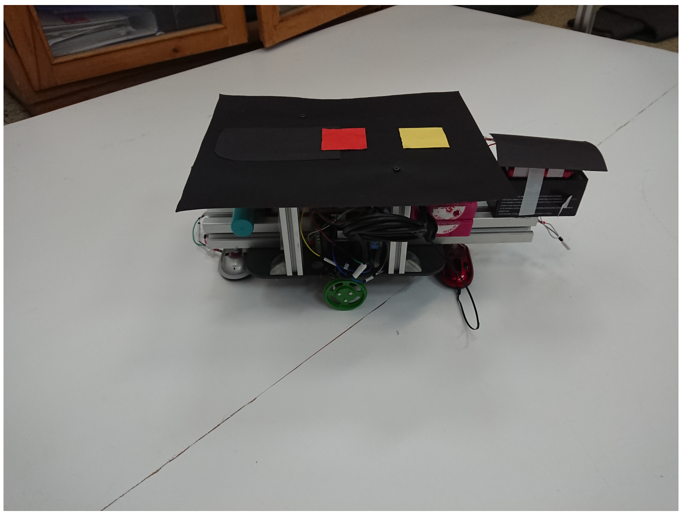


Figure 2.1: Two wheeled differentially driven mobile robot

## 2.2 Set-Up Components

2 wheeled differentially driven robot is used in experiments. There are 2 wheels on the mobile robot which are connected to DC motors with 896 ppr encoders. Odometry data are obtained using those encoders. There also one caster wheel to stabilize robot. To get inertial measurement, an inertial measurement unit (IMU) is placed on the mobile robot. This IMU contains a 3 axis accelerometer, a 3 axis gyro and a 3 axis magnetometer. Encoder, IMU and DC motor current measurement data are measured and sent simultaneously. In this thesis, those measurements alongside with the current measurements of the DC motors are combined to make better position and velocity estimamtes of the mobile robot. However, to evaluate performance of the estimates and measurements, a camera is placed on the top of the experiment area. Experiment area is 1.85 m x 1.85 m square area. The camera used to get position has 1280x720 pixels resolution and gives images with 30 frames per second.

Two ACS 712 analog current sensors are connected to DC motors driving the wheels. Analog current sensor output is measured by a 16 bit ADC. DC motors are controlled by Faulhaber motion controllers and speed of the DC motors are controlled by a closed-loop control system.

Sensor data are collected by Arduino Uno and then sent to the Raspberry pi2 placed on the mobile robot. Also, control of the robot is done by the same Raspberry pi2. A sigma profile aluminum beam is connected to chasis of the robot. Heaviest component of the system is battery, thus place of the battery is used to change center of gravity of the robot. By moving center of gravity of the robot to forward or backward, handling behaviour of the robot is adjusted. If the center of gravity is close to front of the robot, then robot tends to understeer. If the center of the gravity is close to rear of the robot, then robot tends to oversteer. Battery is placed on the sigma profile beam, and since that beam strecth out from beyond front of the robot to beyond rear of the robot, handling characteristic can be adjusted by moving battery on the beam. In this work, behaviour of the robot is expected to be understeer and robot is expected to start sliding in relatively low speeds since maximum speed of the robot is below $2\,m/s$. When the desired characteristic is achieved, battery is fixed at that location on the beam.

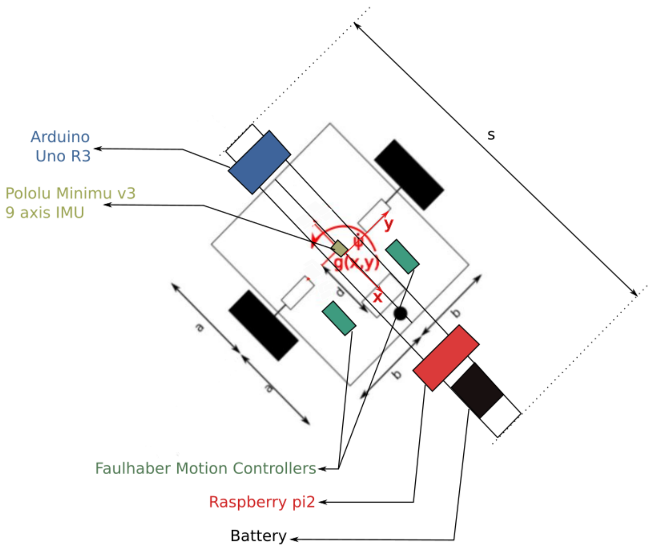
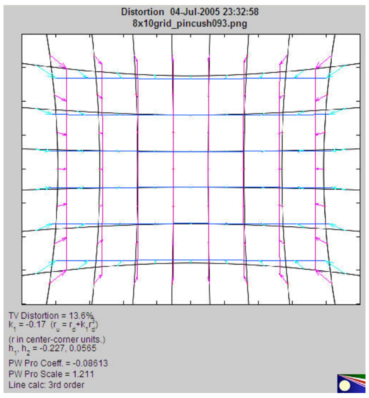


Figure 2.2: Schematic of the Mobile Robot

To prepare ground truth, location data is collected through the camera placed on the top side of the set-up. To process data, python environment with OpenCV 3 library is used. One red and one yellow square is placed on the robot (Figure 2.1). While finding the center of the robot, image processing algorithm finds the red square placed on the center axis of the robot. To find direction of the robot, algorithm finds centers of the red square and yellow square, then finds the angle of the vector starting from center of the yellow square to center of the red square.

Since a flat surface is being captured by the camera, areas around the center represents greater area in the image, areas at the corners represented by a smaller area in the image.



```
Distortion  04-Jul-2005 23:32:58
8x10grid_pincush093.png

TV Distortion = 13.6%
k1 = -0.17  (ru = rd+k1rd3)
(r in center-corner units.)
h1, h2 = -0.227, 0.0565
PW Pro Coeff. = -0.08613
PW Pro Scale = 1.211
Line calc: 3rd order
```

Figure 2.3: Grid showing lines on the ground and their representations on the image [1]

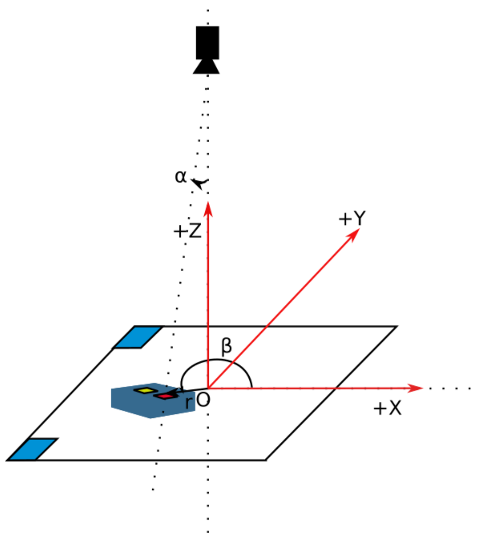


Figure 2.4: Placement of camera and Test Ground

[1]Reprinted from Distortion Module Documentation, In *imatest*,n.d., Retrieved August 1, 2018, from `http://www.imatest.com/docs/distortion_ instructions/`

Since the angle of the lens is 34.5 degree, this distortion is not as much as wide angle lenses, however the effects should not be neglected. To calculate this distortion, camera angle is used. Firstly, camera is placed above the test set-up and the angle of the focal axis is normal to test ground. Also camera is placed above the center of the test ground.
Focal length of the camera to test ground is calculated by the two reference points on the test ground. Image processing algorithm finds the centers of the blue rectangles, then according to the actual distance between those rectangles, it calculates the focal length of the camera using the following equations.

$$\alpha_{cam} = 34.5\, degrees = 0.6\, radians \tag{2.1}$$

$$d\alpha = \frac{\alpha_{cam}}{0.5\sqrt{1280^2 + 720^2}} \tag{2.2}$$

$$\alpha = \sqrt{x_r^2 + y_r^2} d\alpha \tag{2.3}$$

First, positions of the blue rectangles were found with the unknown focal length, then by using the known distance between blue rectangles, focal length is calculated. In the following equations, $\beta$ is the angle of a vector with +x axis.

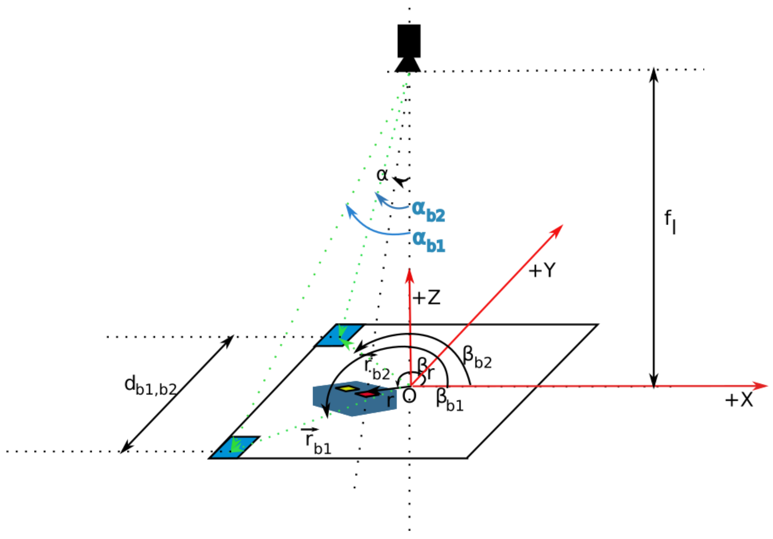


Figure 2.5: Representations of Vectors and Dimensions used in Calculation of Focal Length and Coordinates of the Robot

$$
\begin{aligned}
r_{b1} &= f_l \cot \alpha_{b1} \\
r_{b2} &= f_l \cot \alpha_{b2} \\
\beta_{b1} &= \arctan \frac{p_{y,b1}}{p_{x,b1}} \\
\beta_{b2} &= \arctan \frac{p_{y,b2}}{p_{x,b2}} \\
x_{b1} &= r_{b1} \cos \beta_{b1} \\
y_{b1} &= r_{b1} \sin \beta_{b1} \\
x_{b2} &= r_{b2} \cos \beta_{b2} \\
y_{b2} &= r_{b2} \sin \beta_{b2} \\
d_{b1,b2} &= \sqrt{(x_{b2} - x_{b1})^2 + (y_{b2} - y_{b1})^2}
\end{aligned}
$$

Here, $d_{b1,b2}$ is known, hence from those equations, focal length $f_l$ is calcualted. Then, position of the robot ond yaw angle $\psi$ of the robot is calculated using following formulas.

$$
\begin{aligned}
r_r &= f_l \cot \alpha_r && (2.4) \\
r_y &= f_l \cot \alpha_y && (2.5) \\
\beta_r &= \arctan \frac{p_{y,r}}{p_{x,r}} && (2.6) \\
\beta_y &= \arctan \frac{p_{y,y}}{p_{x,y}} && (2.7) \\
x_r &= r_r \cos \beta_r && (2.8) \\
y_r &= r_r \sin \beta_r && (2.9) \\
x_y &= r_y \cos \beta_y && (2.10) \\
y_y &= r_y \sin \beta_y && (2.11) \\
\psi &= \arctan \frac{y_y - y_r}{x_y - x_r} && (2.12)
\end{aligned}
$$

In these calculations, all position measurements and angular measurements are done according to the ground reference frame.

## 2.3 Mathematical Modelling

Kinematics equations f the mobile robot is required to manipulate sensor data. Sensors are placed in different parts of the mobile robot, hence their measurements are not according to the center of the robot, except IMU which is placed on the vertical axis passing through geometrical center of the mobile robot. Moreover, measurements done according to the body fixed refrence frame of the mobile robot should be converted to the gruond reference frame.

### 2.3.1 Kinematics of Mobile Robot

In this section, kinematics equations relating the body fixed reference frame to ground reference frame are explained. Since sensors and the markers for the camera are placed on the robot, velocity is measured according to the body fixed frame of the mobile robot. To calculate absolute position of the robot, those velocity values should be converted to ground reference frame.

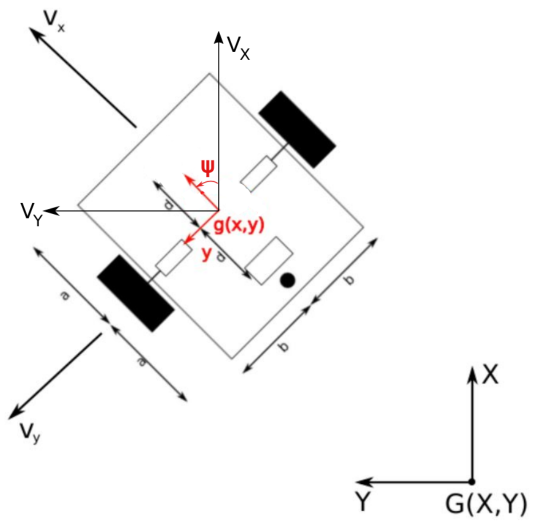


Figure 2.6: Ground(G(X,Y)) and body fixed(g(x,y)) reference frames

In the Figure2.6, $b = 75\,mm$ and $a = 125\,mm$.

In the following equations, $V_X$ and $V_Y$ represents velocity of the robot in the $+x$

and $+y$ direction of the ground reference frame respectively. $V_x$ and $V_y$ represents velocity of the robot in the $+x$ and $+y$ direction of the body fixed reference frame respectively. $\psi$ and $\dot{\psi}$ represents the yaw angle and yaw rate of the robot.

$$V_X = V_x \cos\psi - V_y \sin\psi \tag{2.13}$$

$$V_Y = V_x \sin\psi + V_y \cos\psi \tag{2.14}$$

### 2.3.2 Basic Equations

Velocity of the robot according to odometry data is calculated by using following equations. In those equations, $V_{e1x}$ and $V_{e2x}$ represent velocity of the wheel 1 and 2 of the mobile robot in the $+x$ direction of body fixed reference frame.$V_{ex}$ represents the speed of the robot in the $+x$ direction of the robot done by the odometry measurements, $\dot{\psi}_e$ represents the yaw angle rate of the robot measured by the combination of the data from encoders.

$$V_{ex} = \frac{V_{e1x} + V_{e2x}}{2} \tag{2.15}$$

$$\dot{\psi}_e = \frac{V_{e2x} - V_{e1x}}{2b} \tag{2.16}$$

### 2.3.3 Experiment Procedure

Experiments are done on the $1.85\,m\,x\,1.85\,m$ experiment ground and during the experiment, reference velocity input is sent to the motion controllers of the DC motors connected to wheels of the robot. During the experiment, IMU and encoder data are collected by the raspberry pi2, and ground truth data is collected using the camera placed over the experiment set-up. Before starting experiment, position of the robot is detected by the camera and sent to previously defined start point on the experiment ground. After the experiment, again robot position is detected by the camera and sent to the start point. By this way, experiments are done automatically up to the time when battery of the robot is low or there is an error in the system.

### 2.3.4 Graphical Representation of Sensor Outputs

Sensor data are acquired by Arduino Uno R3, then those data are stored in a text file during the experiment, later that text file is sent to the main computer which is getting images of the robot. Finally, main computer calculates position estimates and prints outputs as a graph. Here, outputs of the different sensors are compared. To see where wheel slips starts, instead of position graphs velocity graphs are presented. From those graphs it can be seen when slips starts from the difference between camera measurement and odometry measurement. In the following figures, comparisons of robot velocity estimates and yaw angle estimates are represented.
In this experiment, different speed input was given to DC motors connected to wheels of the robot. In low slip conditions, motion of the robot is expected to be spiral. Reference for the angular speeds of the each wheel are ramp input, hence each wheel of the robot turns with a constant angular acceleration, but each of the two wheels have different angular acceleration. But after some time, due to the longitudinal and lateral slips, robot goes out of this spiral path. In the end of the experiment, power sent to the DC motors are cut, thus in the end of the experiment robot slows due to damping in the system and friction forces. Also, there is no inclination on the test surface, hence gravity does not disturb lateral forces. In the following test, expected acceleration of the robot is $0.07\,m/s^2$ , according to the angular speed input sent to the speed controllers of the DC motors connected to wheels.

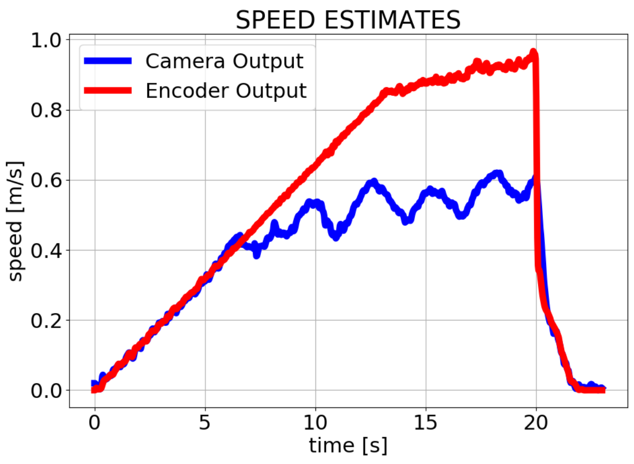


Figure 2.7: Comparison of Velocity Estimates Obtained by Odometry and Camera

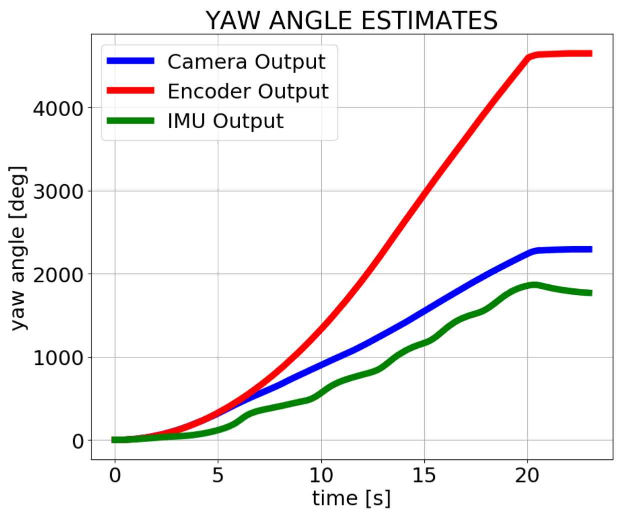


Figure 2.8: Comparison of Yaw Angle Estimates Obtained by Odometry, Camera and IMU

In the Figure 2.7, red line represents speed calculated from the odometry data and the blue line represents speed calculated from images captured by the camera on top of the test set-up. As it can be seen from the figure, slip starts around $5\,s$. After that time, there is significant difference between speed measured with camera and speed measured by odometry. To improve position estimates done by on-board sensors of the mobile robot, a slip estimation algorithm is required.

In the Figure 2.8, red line represents yaw angle of the robot calculated from odometry data, blue line represents yaw angle calculated from images of the robot during experiment and the green line represents the IMU measurement. As expected, slippage starts at $t = 5s$ since after that time, odometry and camera calculations start to differ. In this experiment, IMU measurements performed better than odometry measurements since they are closer to the results of the image processing algorithm. Also, from this figure it can be seen that steering behavior of the robot is understeer since yaw rate sent to the wheels do not translate to actual yaw performance of the robot. In fact, the center of mass of the robot is adjusted to be close to front end of to robot to make this understeer behavior. Thus, this understeer on the robot motion was expected. If the center of mass of the robot was adjusted to close to rear end of the robot, then oversteer would be in motion of the robot and yaw measured by the ground truth would be greater than the yaw angle measured by odometry.

Another effect of the center of mass position is since it is close to front end, robot starts sliding at lower speeds compared to the case in which center of mass was adjusted to close to geometrical center of the robot.

In the second experiment, acceleration of the robot is expected to be $0.037\,m/s^2$ according to the speed input sent to speed controllers of the DC motors. Here, while computing the expected acceleration of the robot, eqn(2.15) is used. In the low slip regions, as in the first experiment, robot is expected to drive with $0.037\,m/s^2$, but as the speed of the robot is increased, after some time longitudinal and lateral traction force limits are reached, then acceleration of the robot is expected to lower than the expected acceleration calculated by the eqn(2.15). In this experiment, slippage is expected to start later than first experiment because of the acceleration input sent in this experiment is lower than the first experiment. This causes traction forces to reach their peak point later, and slippage occur later.

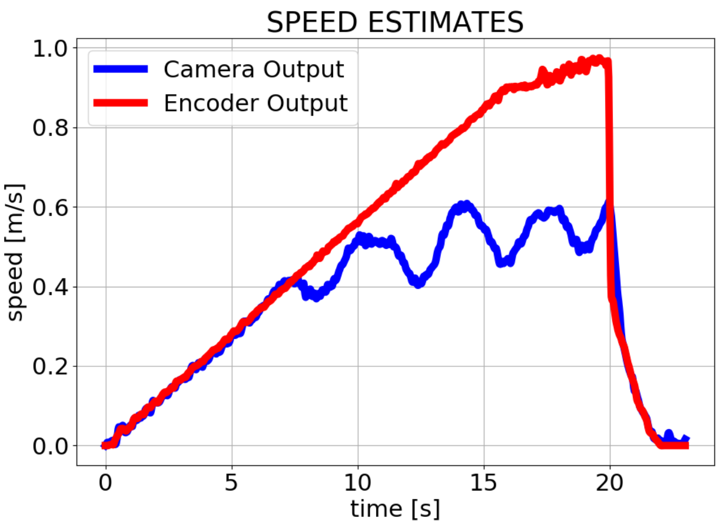


Figure 2.9: Comparison of Velocity Estimates Obtained by Different Sensors

In this experiment, a lower acceleration was given to mobile robot, hence velocities are lower compared to first experiment. Slippage starts around $t = 11s$ (Figure 2.9) .

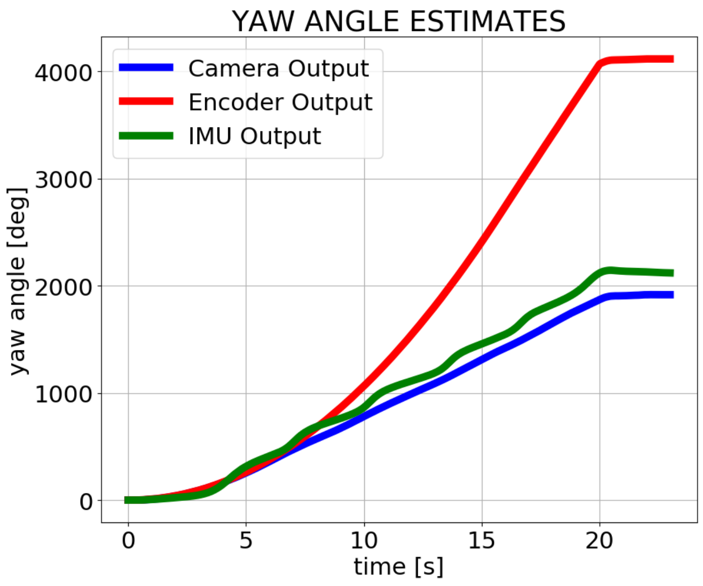


Figure 2.10: Comparison of Yaw Angle Estimates Obtained by Different Sensors

IMU measurements repsresented by the green line performs relatively bad at the begining of the experiment (Figure 2.10). Up to around $t = 8s$, it shows very little angular movement. Odometry measurement is again has error due to wheel slippage. Thus, inertial measurements or odometry should not be trusted alone while making position estimates and both have errors.

## 2.4 Conclusion and Discussions

In this chapter, 2 wheeled differentially driven robot, set-up and test surface used in the experiments, components and sensors used in the experiment and basic equations used to manipulate sensor data is represented. Later, results of the experiments with graphical outputs are explained. Performance of the different sensors compared to ground truth is mentioned. In the next chapter, Deep Neural Networks, and structure of two of the widely used and simplest Deep Neural Networks are explained. Later, Deep Neural Network used in this work to estimate speed of the robot and yaw angle of the robot is explained.

# CHAPTER 3

# DEEP NEURAL NETWORKS

## 3.1 Introduction

In this chapter, feedforward neural networks and recurrent neural networks are explained. Activation functions used in the networks are introduced, advantages and disadvantages of using them are explained. Then, deep recurrent neural network used in the work is expressed, and the inputs of the network is tabulated. Lastly, structure of the network is explained.
Deep neural networks or Deep Learning is used in a variety of areas like image recognition [19], speech recognition[20], image processing and enhancement [21]. Schmidt and Roth[21] show that deep learning can be used to identify traffic signs, which is a important task in autonomous driving and navigation.

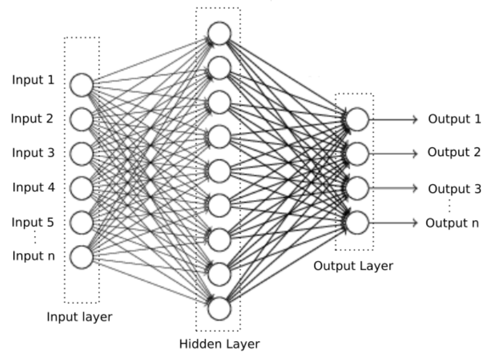


Figure 3.1: Simple Neural Network with a Single Hidden Layer

Deep networks are a type of artificial neural network, having more than one hidden layer. These networks were inspired by the structure of the animal brains. In the animal brains, there are multiple neurons which are connected by each other by synapses. Similar to brain structure, artificial neural networks have neurons represented by circular patterns in the below image, and those neurons are connected by the weights, which are shown by arrows in the figure. Using the previously done experiments and previously collected data, activation values for the neurons and the weights of the connections are found. While finding those parameters, mean square error is used as loss function and Adam optimaziton method is used.

## 3.2 Feed Forward Networks and Recurrent Networks

According to Schmidhuber[22], one of the simplest deep networks are feedforward neural network and recurrent neural network. In the feedforward neural networks, information flows in one direction and it makes no loop movement. They are the simplest model of neural networks. They are widely used in areas like image classification. In the python and TensorFlow environment, they are easy to implement and train.

In the Figure 3.2, a multi-layer deep feedforward neural network is represented. Information is taken by the input nodes, after that point calculations are made and information is passed to the all the layers in front of the current layer. There is no back movement and feedback loop. Flow of the information is represented by the arrows, and as it can be seen from the figure, a neuron takes information from the neurons one layer before it, and if that neuron is activated, it passes information to the neurons one layer after it. They are trained using to the training data, and through the training, optimal values are found according to loss function and optimization method. For the activation function, rectified linear unit (ReLU) is used. According to Glorot et al.[23], compared to sigmoid function, rectified linear unit function has better gradient propogation. Moreover, since it is a very simple function, it does not bring much computational load to algrorithm. According to the lecture notes of Sert[24], compared to other activation functions, ReLU (3.1) is around 6 times faster in computation. Moreover, it does not suffer from gradient vanishing. Sigmoid (3.2) function is another function that can be used for gating, however, since gradient vanishing is a big problem for this function, it is not used. Tangent hyperbolic function

(3.3) is another activation function, but again gradient vanishing is a big problem for this function. Hence it is not used in this study. Self-gated activation function (3.2) is another activation function, but since it is a new function in deep neural network applications and it is not widely used as ReLU, it is not used as activation function in this study.

$$\begin{aligned} f(x) &= max(0, x) && (3.1) \\ f(x) &= \frac{1}{1+e^{-x}} && (3.2) \\ f(x) &= \tanh(x) && (3.3) \\ f(x) &= xsigmoid(\beta x) && (3.4) \end{aligned}$$

Because of the computational simplicity and gradient propogation properties of ReLU, in this study ReLU is used instead of these activation functions.
In the Figure 3.2, a multi-layer neural network is represented. To improve the performance of the system, more than one hidden layers can be added to network.

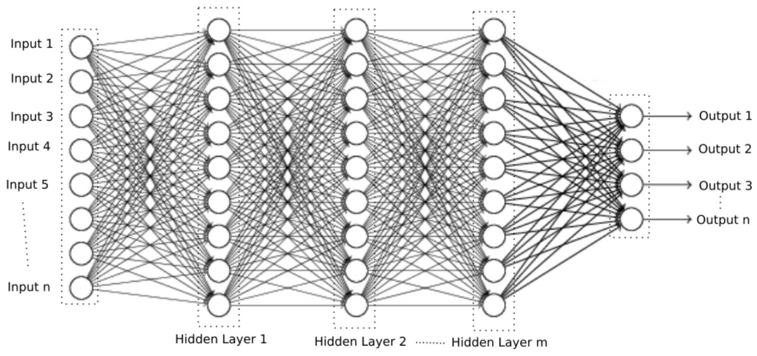


Figure 3.2: Simple Neural Network with Multiple Hidden Layers

The other type of neural networks is recurrent neural networks. Unlike feedforward neural networks, recurrent neural networks have feedback loop within them. They have long short term memory(LSTM) or gated recurrent units, hence they can handle batch of inputs, thus they are widely used in speech recognition [20]. Like feedforward neural networks, recurrent neural networks can be composed of multiple hidden layers. Figure 3.3 shows a multi-layer recurrent neural network. Performance of the fittet function can be increased by adding more hidden layers. However, adding too much layers can cause over-fitting and increase training time. Hence, hidden layers should be added to system one by one starting from the one hidden layer. When the desired fitting is achieved, required number of hidden layers is found. Also, number of neurons in a hidden layer should be determined carefully. If there are too much neurons, then that may cause increase in training time since number of computations will increase.

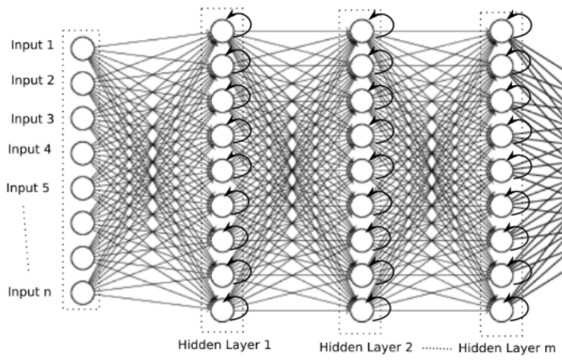


Figure 3.3: Recurrent Neural Network with Multiple Hidden Layers

## 3.3 Deep Network Used in the Study

Since system is dynamic, recurrent neural network is decided to be used. It is known that tire slips are non-linear functions, and also there is noise on each sensor used in the system, hence a multi layer recurrent neural network is used to capture non-linearity of the system. After that, experiments are done and data are collected from those experiments. In the experiments, robot made a near-circular motion. DC motors connected to wheels of the robot are controlled by closed-loop controllers. Different reference velocity values are sent to inner and outer wheels of the robot, hence motion is near circular. For each inner and outer speed set, 15 experiments are done. In total, 140 experiments are done for one side of the robot,i.e same inner and same outer wheel. After data are collected from those experiments, some of the experiments are chosen to be validation data for the recurrent network model. Those data are not sent to model while training, they are only used to validate and evaluate the performance of the model. Validation experiments are chosen randomly.

### 3.3.1 Inputs to the Network

Since Recurrent Neural Network is used in the study, input should be scaled. All of the inputs are scaled between 0 and 1, but before scaling, a low pass filelter used to filter high frequency noise on the inputs. Those inputs are measured by the sensors like IMU, camera and encoders, because of this they have some noise on them. Even the camera captures have some noise, and since they give position information and later they are used to obtain speed information, errors propogate. To decrease level of noise, all the measurements are sent to second order low-pass butterworth filter. For this application, second order low-pass filter with 50 Hz sampling rate and 7.5 Hz cutoff frequency is chosen.

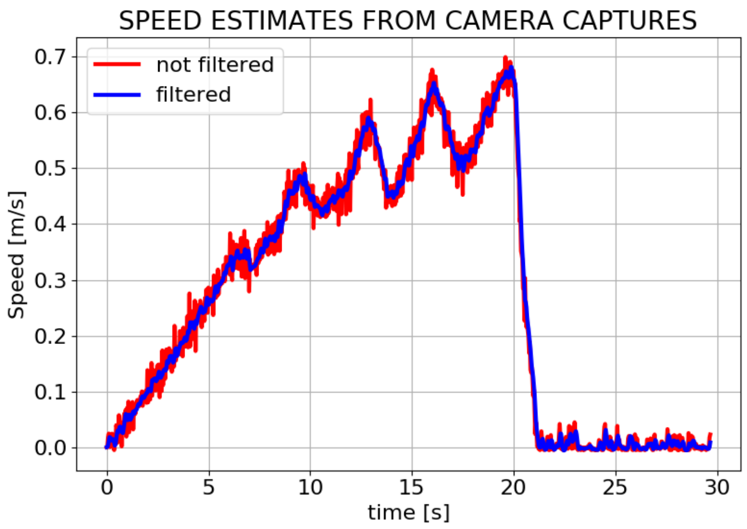


Figure 3.4: Comparison of Filtered and Unfiltered Speed Estimates obtained from Image Captures

There are 11 inputs in total sent to the recurrent neural network. Those inputs are time at the each measurement set are taken, time step(time increment between now and previous measurement), speed of the each wheel and center of the robot (2.15) with respect to the ground measured by the encoders, since encoders are used in these measurements, they only supply longitudinal speed of the wheel with respect to the ground without considering effects of the slippage, yaw angle of the robot from the encoder measurements of the each wheel and speed difference between them (2.16), current drawn by the DC motors connected to the wheels, acceleration of the robot along the x and y directions of the body-fixed reference frame measured by the IMU placed in geometrical center of the robot, and yaw angle with respect to the body-fixed reference frame again measured by the IMU. While training the network, sample outputs are the camera capture output, which are speed of the robot calculated from the position estimates (2.8) (2.9) from the image processing algorithm, and the yaw angle estimates (2.12) from the image processing algorithm.

Table 3.1: Inputs and Outputs of the Network

| Label | Measurement | Sensor Used |
|---|---|---|
| Input 1 | Time | |
| Input 2 | Time Step | |
| Input 3 | Longitudinal Speed Center | Encoder 1&2 |
| Input 4 | Longitudinal Speed Wheel 1 | Encoder 1 |
| Input 5 | Longitudinal Speed Wheel 2 | Encoder 2 |
| Input 6 | Yaw Angle | Encoder 1&2 |
| Input 7 | Yaw Angle | IMU |
| Input 8 | Acceleration +x | IMU |
| Input 9 | Acceleration +y | IMU |
| Input 10 | Current Motor 1 | Current sensor 1 |
| Input 11 | Current Motor 2 | Current sensor 2 |
| Output 1 | Speed | Camera |
| Output 2 | Yaw Angle | Camera |

Current measurement, IMU readings and encoder readings are collected simultameously, however, images of the robot are not captured simultaneously to those readings. Sample rate of the images are also lower than the sapmle frequency of other measurements. Because of this, position of the robot at the reading times of the encoders and other sensors are estimated by the interpolation of the image data. Position and yaw angle from the image readings are calculated, after that those values sent to second order low-pass butterworth filter, then functions for those readings are constructed using image capture times and values at the encoder readings are calculated from the interpolation of the reading times and capture times.

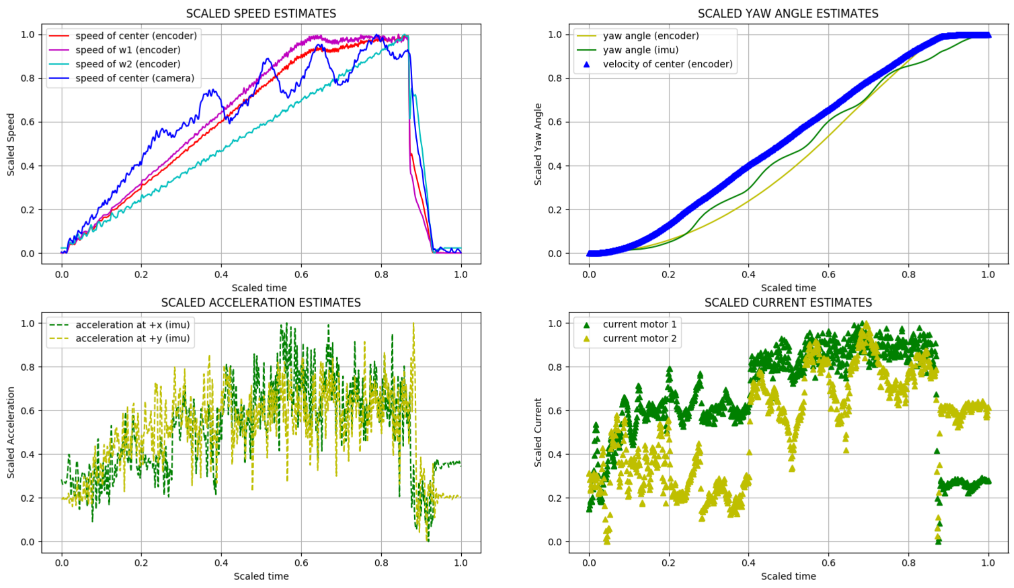


Figure 3.5: Scaled Measurements Used to Traing Network

### 3.3.2 Network Structure

There are 11 inputs in total supplied to the network, and there are 2 outputs, which are speed of the center of the robot and yaw angle of the robot. When combined, these two outputs make velocity vector of the robot, which can be used to generate position output of the robot.
While calculating number of hidden layer required, following formula is used.

$$N_h = \frac{N_s}{\alpha(N_i + N_o)} \tag{3.5}$$

In this equation, $N_s$, $N_i$ $N_o$ are number of hidden layers, number of inputs and number of outputs, respectively. While determining number of cells in each hidden layer, lower than the twice of the number of inputs are chosen to reduce time to calculate weights in the network while training. Hence, there are 20 cells in each hidden layer.

The network used in this study composed of 5 hidden layers. In each layer, there are 20 recurrent cells. Training is done with 120 experiments. 10 experiment are saved as validation data and those are not used in training. While training, 500 epochs are used.

Since all the inputs are scaled then sent to network, output of the network is scaled too. This is not a big problem cases where data are observed first, then the remaining data are estimated. In those cases, observed values are used for re-scaling. However, in this case, camera output is only used to train the network. For the validation data, camera output is only used to evaluate the performance of the trained network. Hence for those experiments, re-scaling is a problem. For re-scaling, a region or a point should be known to re-scale output of the network. Here, only known point is at $t = 0$, $v = 0$ and $\psi = 0$, however those cannot be used to re-scale data. In (2.7) and (2.8), velocity and yaw agnle estimates done by the data from encoders are same as the camera output up to the point where slip begins. But in the beginning of the experiment and at the beginning of the movement, odometry gives reliable data. For re-scaling, this charecteristic can be used.

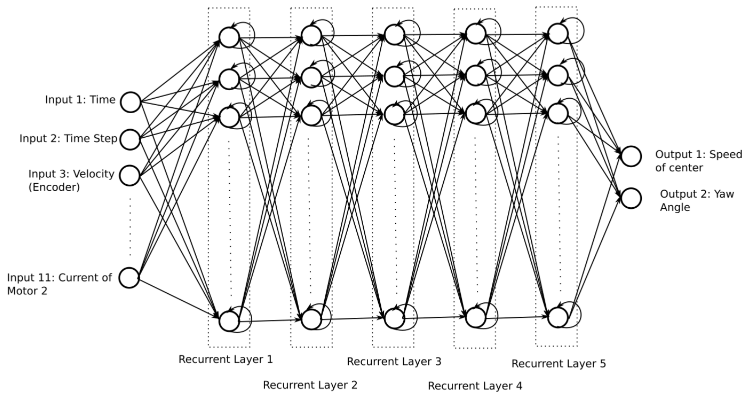


Figure 3.6: Structure of the Recurrent Neural Network Used in the Study

Re-scaling is done by using the first 200 data points of the experiment. From those points, speed and yaw angle estimates are calculated from odometry data. Then, mean of those speed and yaw angle estimates are calculated. Mean of the speed and yaw angle output of the network is calculated and whole output data re-scaled to have same mean values for those 200 points. By this way, whole data are re-scaled.

## 3.4 Conclusion and Discussion

Feedforward and Recurrent Deep Neural Networks are explained. Then, activation function and loss function optimization method used in the deep recurrent neural network used in the work is represented. Inputs sent to the network are tabulated, then filtering and scaling of the inputs are represented. In the end of the chapter,structure of the network and training process are explained.

# CHAPTER 4

# RESULTS OF THE RECURRENT DEEP NEURAL NETWORK

## 4.1 Introduction

In this chapter, results of the algorithm obtained from the experiments those did not used to train the algorithm are presented. Firstly, speed estimates of the camera, odometry measurements and results of the algorithm are compared. Then, using those speed estimates, distance covered by the robot is plotted by those three estimates. Thirdly, yaw angle estimate of the robot is plotted by the camera, IMU, odometry and result of the algorithm is plotted. Later, position estimates of the camera, odometry and result of the algorithm which is combination of distance estimates of the algorithm and yaw angle estimate of the algorithm is presented. Lastly, performance of the odometry, IMU and proposed network output evaluated by the ground truth are tabulated and discussed.

## 4.2 Plots and Comparisons

Results of the 7 experiments are plotted and compared. Those experiments are chosen randomly and acceleration input sent to the wheels of the robot are different in those experiments, hence, speed profile and yaw angle profiles are different. Also, in each experiment, acceleration input sent to the each wheel are different too, because of this, in low slip conditions, robot is expected to follow a near-circular path. In all of the experiments, there is no inclination on the surface. Same surface and same wheels are used in all experiments, moreover parameters like mass, mass center, inertia are not changed, thus wheel-ground interaction parameters are expected to be same in all experiments.

### 4.2.1 Experiment 1

In this experiment, acceleration of the robot is expected to be $0.064\,m/s^2$ according to equation (2.15). The speed estimates in experiment 1 is plotted in Figure (4.1). Ground truth is camera, hence errors should be evaluated according to camera data. As it can be seen from the figure, estimates from odometry or encoder data give larger difference to camera data when compared to output of the recursive deep leraning algorithm. Moreover, both camera data and output of the networks starts to differ from odometry data around $t = 6\,s$, which means that algorithm is able to capture start of the slippage. However, as expected, output of the network does not follow the ground truth exactly and it has some error. Despite that, network is able to estimate start of the slippage and able to decrease the error of the odometry.

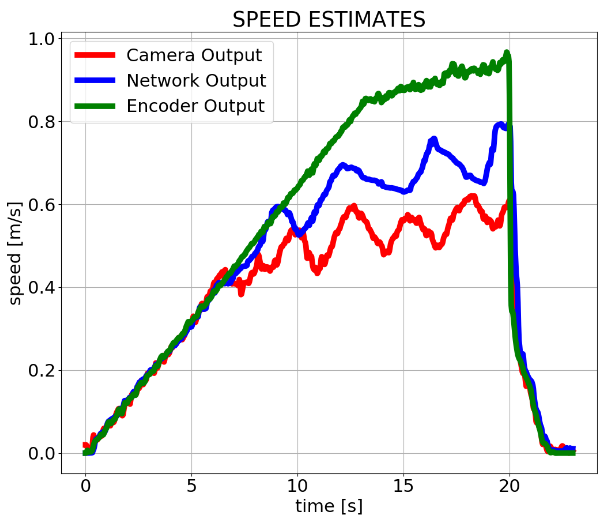


Figure 4.1: Speed Estimates of Experiment 1

The data in the Figure 4.1 are integrated and the total distance covered by the robot is plotted (Figure 4.2). Errors in the speed estimates are accumulated, hence output of the algorithm gives better results compared to odometry or encoder data. From Figure 4.1, it is known that slippage starts around $t = 6\,s$, hence covered distance estimates start to differ after $t = 6\,s$.

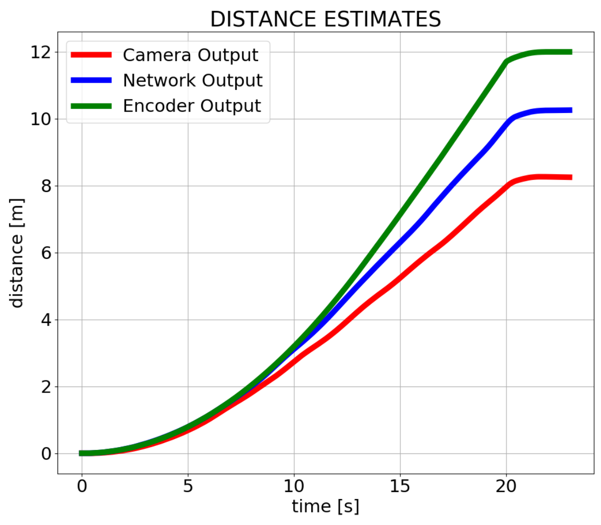


Figure 4.2: Covered Distance Estimates of Experiment 1

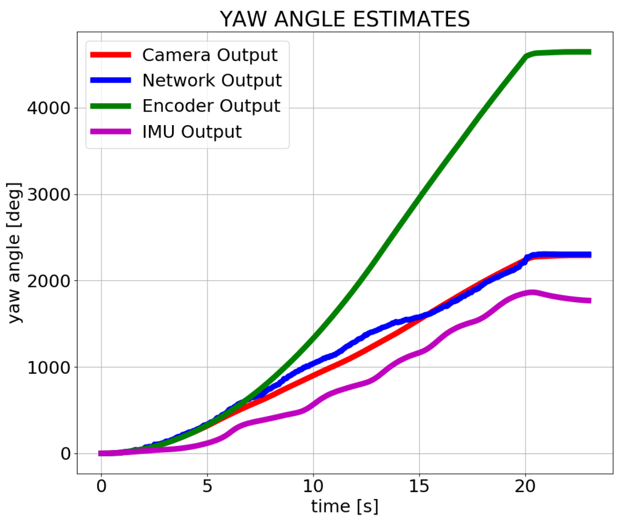


Figure 4.3: Yaw Angle Estimates of Experiment 1

Yaw angle estimates of the robot done by IMU, odometry or encoder data and output of the recursive deep network are compared to camera data (Figure 4.3). Biggest difference to camera data is seen on odometry data. IMU output is closer to camera data compared to encoder data, however, output of the network gives closest result to camera data.

In Figure 4.4, position estimates of the robot which is combination of distance covered and yaw angle estimates are presented. As expected, due to not being able to detect longitudinal and lateral slips of the robot, encoder data gives circular motion. Camera data shown by the red line, is the ground truth data. Output of the algorithm is not as close to camera data as expected. It is due to errors in the estimates of velocity and yaw angle causes accumulated errors in integration.

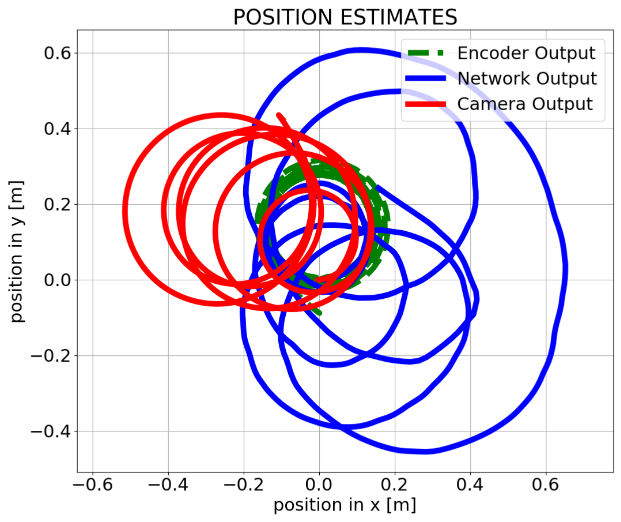


Figure 4.4: Position Estimates of Experiment 1

### 4.2.2 Experiment 2

Acceleration input sent to the motion controlllers of the DC mototrs connected to wheels of the robot are lower than the experiment 1. According to the equation 2.15, robot is expected to have $0.056\, m/s^2$ acceleration. In experiment 2, similar to experiment 1, speed estimate of the network is closer to camera data compared to encoder data. Since acceleration of the robot is lower in this experiment, slippage is expected to start later, and as it can be seen from the Figure 4.5, slippage starts around $t = 7\, s$. However, in this experiment, network captures slippage with a lag. According to the network, slippage starts around $t = 8\, s$. It captures start of slippage with around $1\, s$

delay.

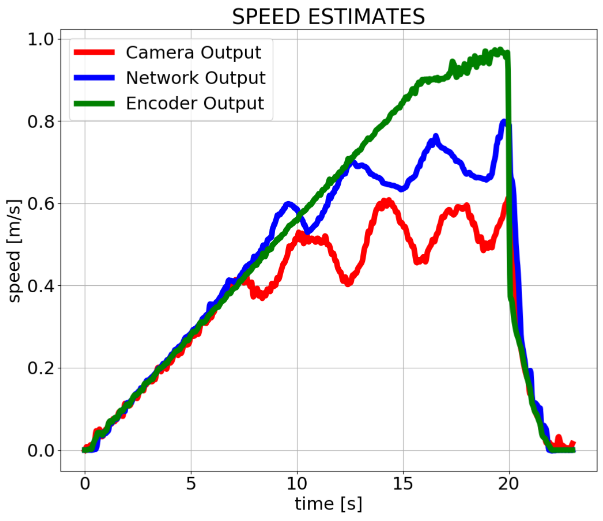


Figure 4.5: Speed Estimates of Experiment 2

Since speed estimates of the network is closer to camera data, it has lower accumulated error and covered distance is closer to camer data as expected(Figure 4.6).

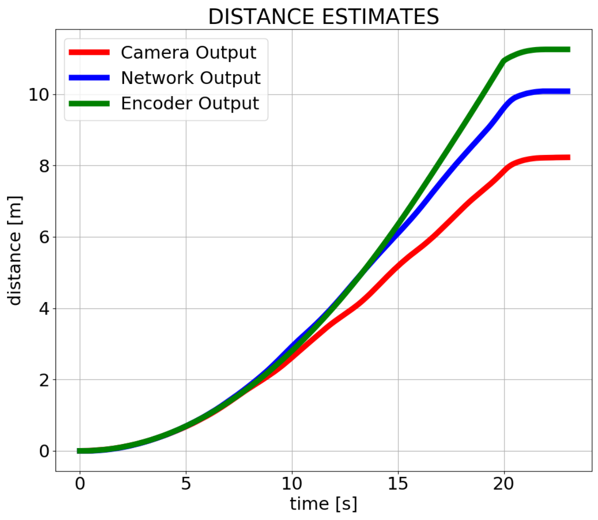


Figure 4.6: Covered Distance Estimates of Experiment 2

Yaw estimates of the network is closest to camera data. Again, IMU output is closer than encoder data, but it is worse than network output(Figure 4.7).

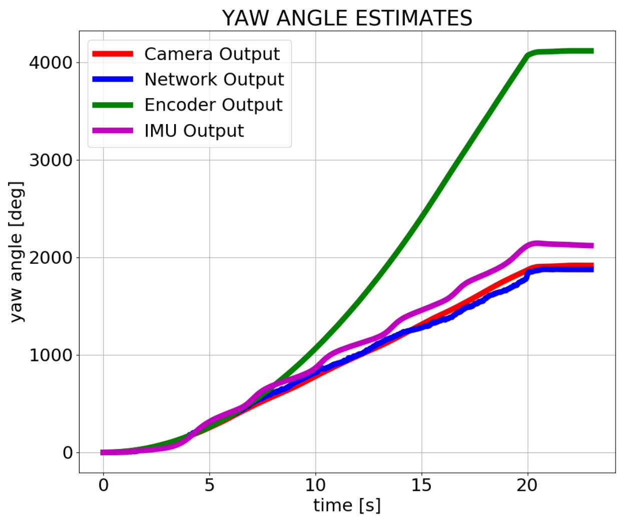


Figure 4.7: Yaw Angle Estimates of Experiment 2

Accumulated errors in speed estimates and yaw angle estimates resulted in errors big errors in the position estimate(Figure 4.8).

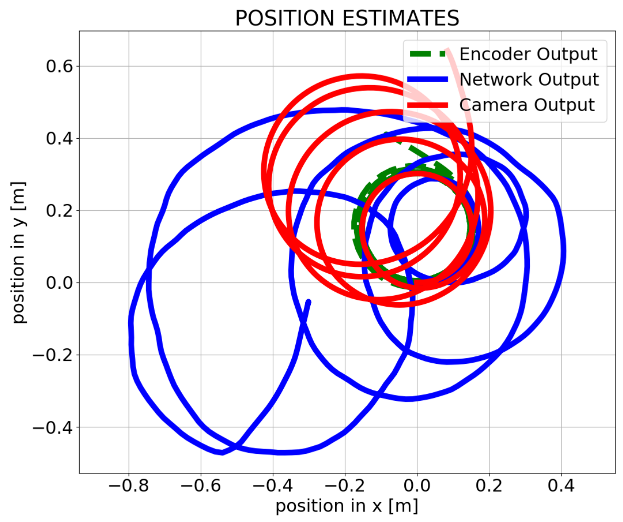


Figure 4.8: Position Estimates of Experiment 2

### 4.2.3 Experiment 3

Compared to first and second experiment, acceleration input sent to the motion controllers of the DC motors are lower than previous experiments in this experiment. According to the equation 2.15, robot is expected to $0.049m/s^2$ acceleration in low slip conditions. In this experiment, speed changes are captured by the deep network, however, there are some errors in the magnitude of speed(Figure 4.9). Slippage starts around $t = 8\,s$, however after brief slippage, slippage diminishes, then slippage starts around$t = 12\,s$, later than the experiment 1 and experiment 2, as expected. Network is able to capture the change at $t = 12\,s$, but the speed estimates at that point have some error. When slip starts around $t = 8s$, output of the deep network gives higher error compared to encoder output. After $t = 10s$, output of the deep network starts perform better than the encoder data.

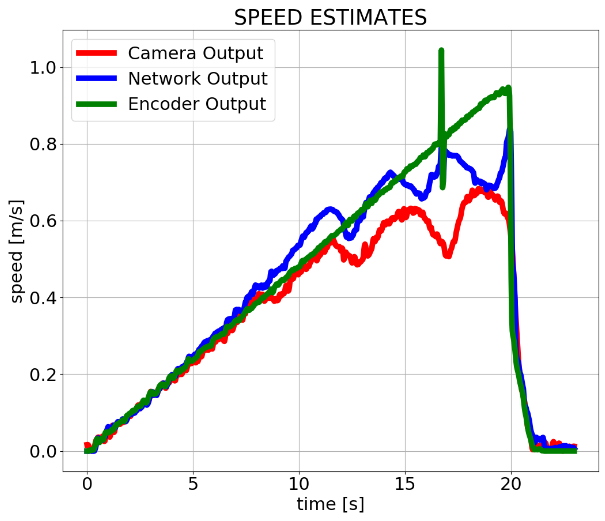


Figure 4.9: Speed Estimates of Experiment 3

Higher errors in the speed estimation makes covered distance estimates worse than encoder measurements. However after $t = 17s$, since performance of algorithm is better in high slip regions compared to odometry, distance estimates are closer to camera data (Figure 4.10).

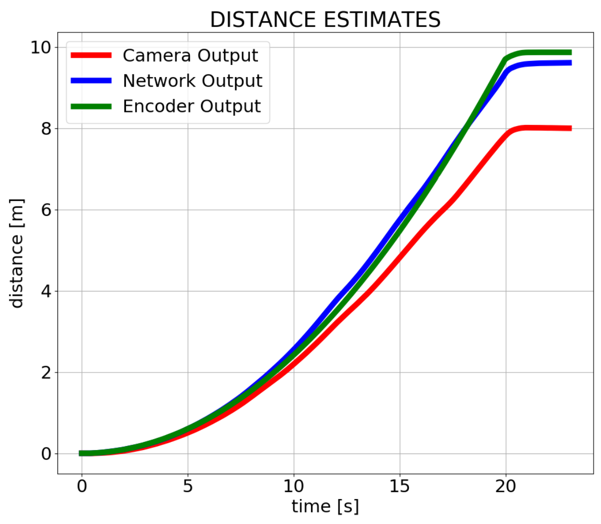


Figure 4.10: Covered Distance Estimates of Experiment 3

In the Figure 4.11, yaw angle esitmates of the experiment 3 is shown. In this experiment, output of the network has performed better compared to IMU output and odometry, but in the end of the experiment, IMU has given slightly closer result to camera.

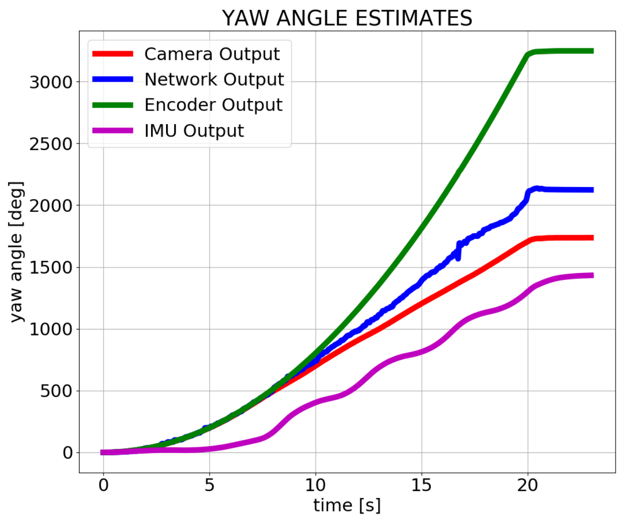


Figure 4.11: Yaw Angle Estimates of Experiment 3

Again, errors in the speed estimation and yaw angle estimation have accumulated and caused large errors in the position estimates(Figure 4.12). But, the path of the network output makes larger near-circular patterns, close to the camera output. Here, path output of the odometry is circular, and the radius of this path is smallest. As explained before, center of mass of the robot is adjusted so that handling behaviour of the robot is understeer. From the odometry output, since it is not be able to capture lateral and longitudinal slips, cornering radius is expected to be smallest, and actually it has the smallest cornering radius as it can be seen from Figure 4.12. Since cornering radius of the network output is close to the cornering radius of the ground truth, it can be said that network is able to capture lateral and longitudinal slips and reduce the error.

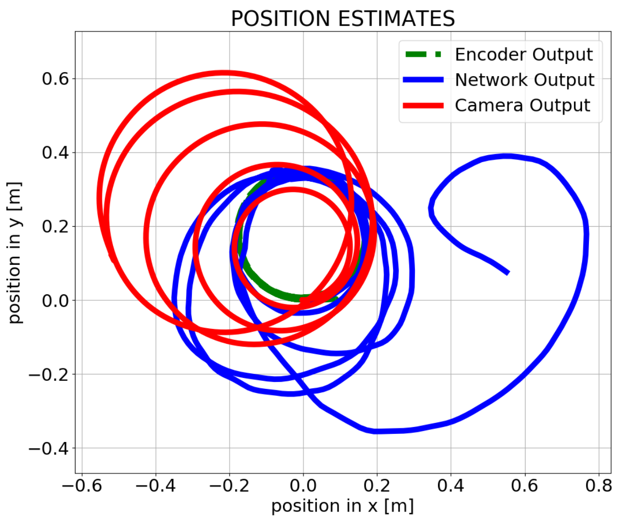


Figure 4.12: Position Estimates of Experiment 3

### 4.2.4 Experiment 4

According to the equation 2.15 and the acceleration input sent to the motion controllers, robot is expected to $0.036m/s^2$ acceleration in low slip conditions. Hence, slip is expected to start later than slippage start time in the previous experiments. In this experiment, speed estimates of the network is closer than previous experiments. Network guessed speed more closer to camera data (Figure 4.13).

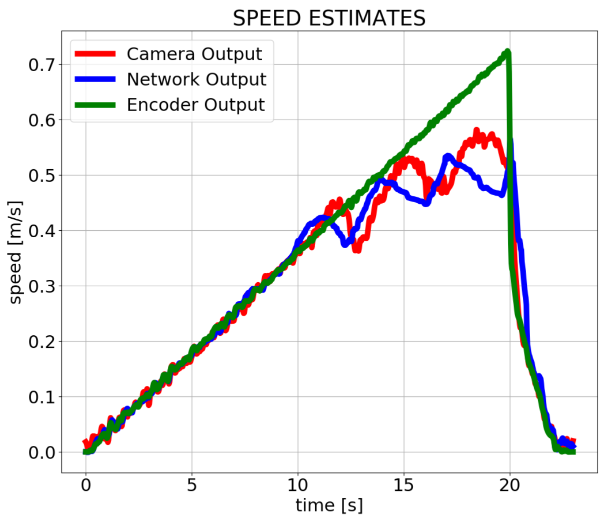


Figure 4.13: Speed Estimates of Experiment 4

Since speed estimates are closer to camera output, distance covered is closer to camera estimates too. Most of the error of the odometry estimate is compansated by the network (Figure 4.14).

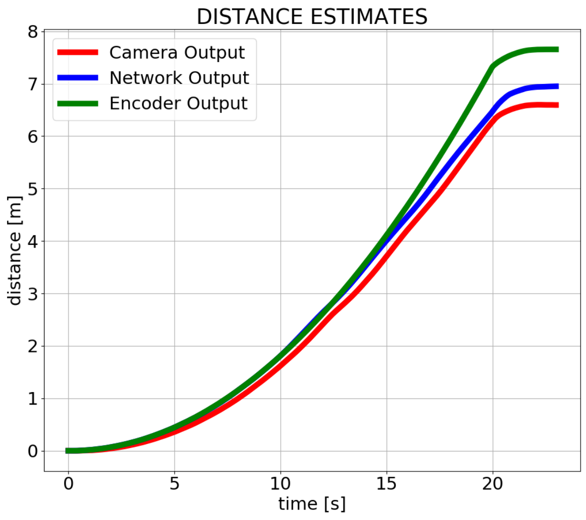


Figure 4.14: Covered Distance Estimates of Experiment 4

The yaw estimates of the network is significantly better compared to odometry and IMU (Figure 4.15).

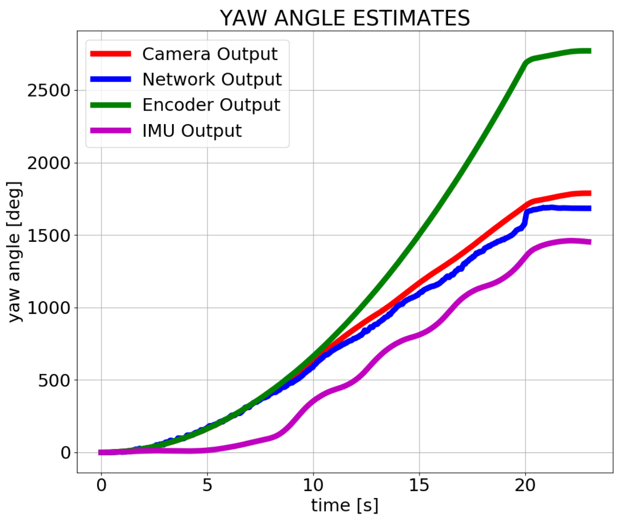


Figure 4.15: Yaw Angle Estimates of Experiment 4

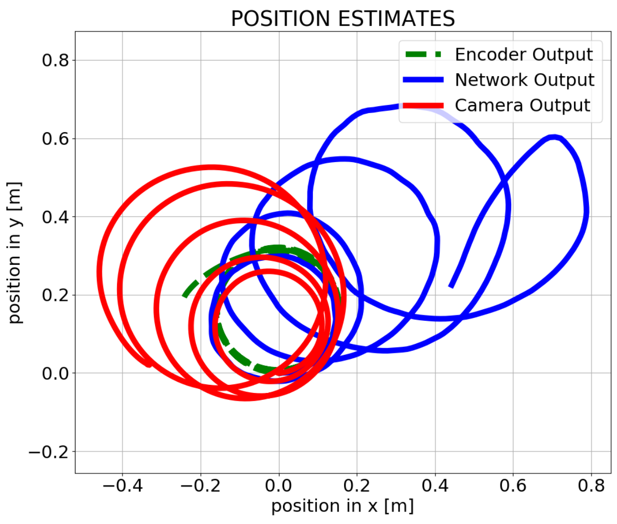


Figure 4.16: Position Estimates of Experiment 4

Again, in this experiment, odometry data reprresents path without lateral and longitudinal slips. Performance of the trained network when capturing and compesating lateral and longitudinal wheel slips can be evaluated after examining covered distance(Figure 4.14) and position(Figure 4.16) plots. Since covered distance estimate of the network is closer to the ground truth data, it is able to compansate some of the longitudinal wheel slip. Moreover, since yaw angle estimates of the network is much closer to the ground truth and radius of cornering of the network output is close to the ground truth, it is able to capture and compensate some of the lateral slip.

### 4.2.5 Experiment 5

According to the equation 2.15 and the acceleration input sent to the motion controllers, robot is expected to $0.056m/s^2$ acceleration like in experiment 1, in low slip conditions. But in this experiment, acceleration input difference sent to inner and outer wheels are higher than experiment 1, hence robot is expected to make sharper turn. Compared to experiment 1, acceleration input of the outer wheel is slightly increased and the acceleration input of the inner wheel is slightly decreased to have a sharper turning motion during the experiment. Because of the sharper turning motion, lateral slip will be more effective in this experiment compared to experiment 1.

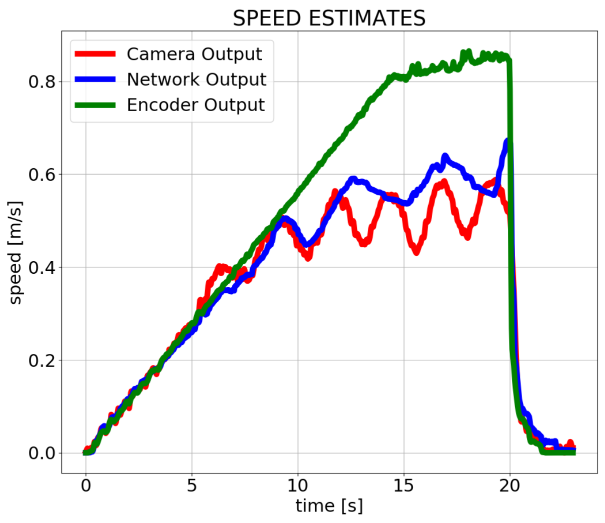


Figure 4.17: Speed Estimates of Experiment 5

The speed estimates made by network is closer to camera data than odometry data, but the later speed fluctuations are not captured by the algorithm. Despite this, even in this regions output of the deep network is significantly closer to ground truth than odometry data(Figure 4.17).

Compensation of some of the errors in the speed estimate resulted in significant improvement in the covered distance estimates. As it can be seen from Figure 4.18, output of the network is considerably better than odometry estimates. If the speed estimates done by the network is close to the ground data, which means that it is able to capture and compensate wheel slips, covered distance calculated by those estimates will be closer to the ground data. Since covered distance estimates are significantly closer to ground truth compared to odometry data, recurrent deep neural network is able to capture and compensate some of the wheel slips.

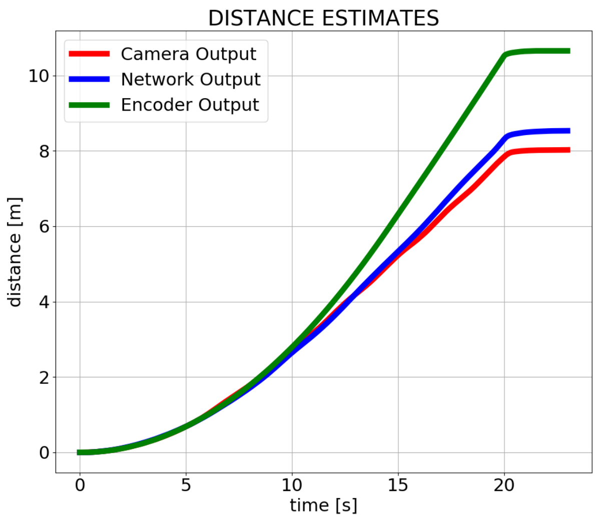


Figure 4.18: Covered Distance Estimates of Experiment 5

Like in the Figure 4.15, yaw angle estimates of the network is so considerably closer to camera data compared to IMU and odometry (Figure 4.19). While computing yaw estimates from the odometry data, equation 2.16 is used, thus due to wheel slips, there will be high error in odometry yaw angle estimates. Because of the closeness of the network output to ground data and yaw estimate calculated using the odometry data and yaw angle calculated using IMU data are inputs of the network, it can be said that network is able to capture and compansate some of the wheel slips and compose IMU and odometry data to estimate more accurately yaw angle of the robot.

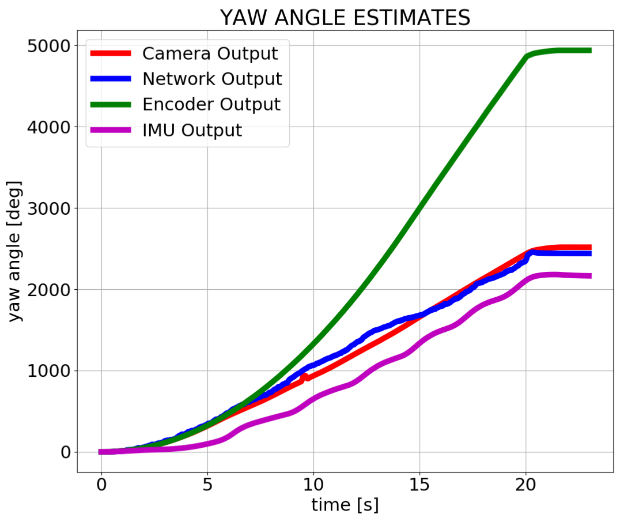


Figure 4.19: Yaw Angle Estimates of Experiment 5

Like previous experiments, due to the cumulative errors while calculating positions, position estimates were as not good as expected.

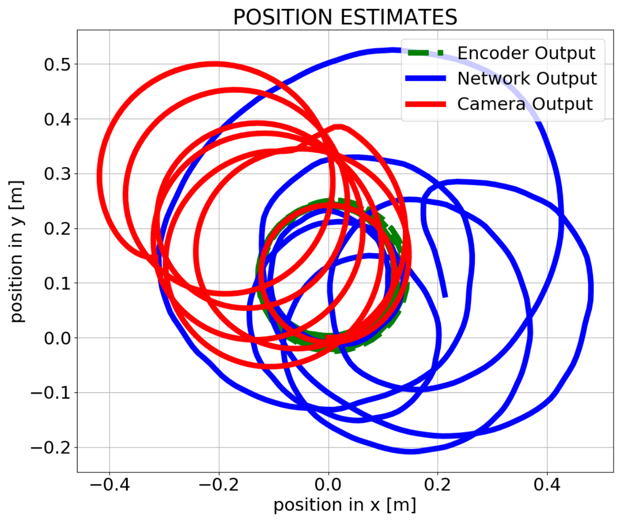


Figure 4.20: Position Estimates of Experiment 5

### 4.2.6 Experiment 6

In experiment 6, the accelration of the robot is expected to be $0.049m/s^2$ according to equation 2.15 and the acceleration input sent to the motion controllers of the DC motors.

Trained network is able to capture the start of the slippage and in the beginning of the slippage, it is able to guess speed of the robot relatively closely. But after around $t = 12\,s$, network is not able to guess the speed of the robot as in the beginning of the experiment and up to $t = 12\,s$ (Figure 4.21). Although network output does not follow ground data exactly, it is considerably closer to ground data compared to odometry output.

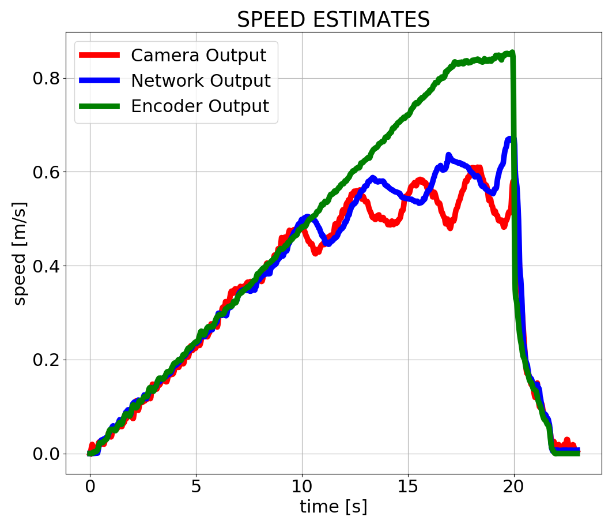


Figure 4.21: Speed Estimates of Experiment 6

Trained network compensated some of the slip in the speed estimations (Figure 4.21). Reducing the speed estimation errors will result in the error in covered distance estimates. Since output of the network is close to the ground truth data in speed estimates, covered distance calculated by the trained network is significantly closer to the ground truth data than odometry data (Figure 4.22). Because of the errors in speed estimates, covered distance estimates calculated from the network output have errors, but those error is relatively small compared to the odometry data.

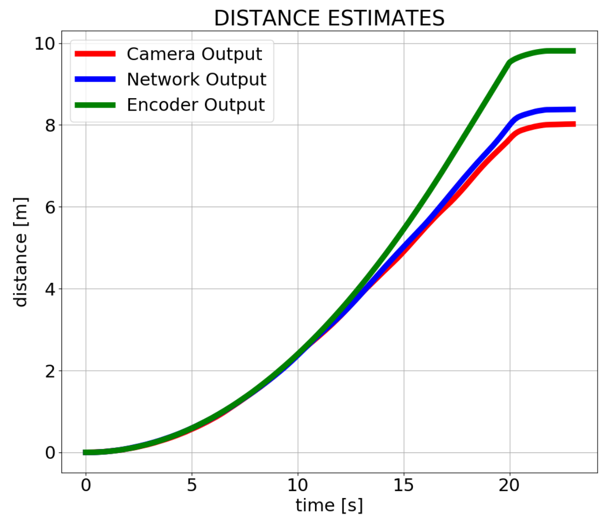


Figure 4.22: Covered Distance Estimates of Experiment 6

Yaw estimates from the trained recurrent neural network are consideraby closer to the ground truth data compared to the IMU measurements and odometry calculations(Figure 4.23).

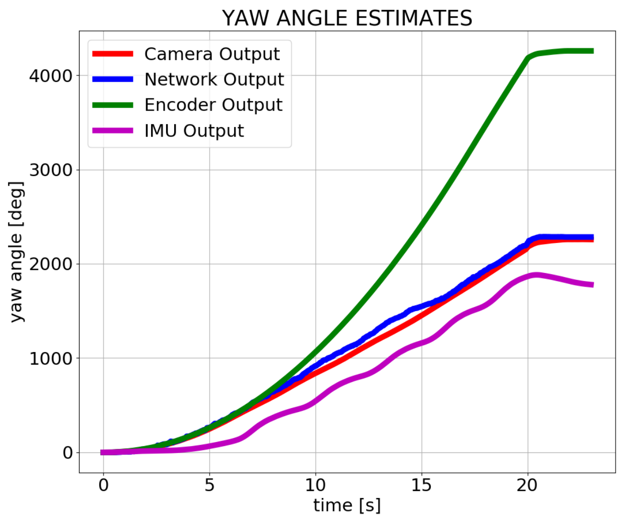


Figure 4.23: Yaw Angle Estimates of Experiment 6

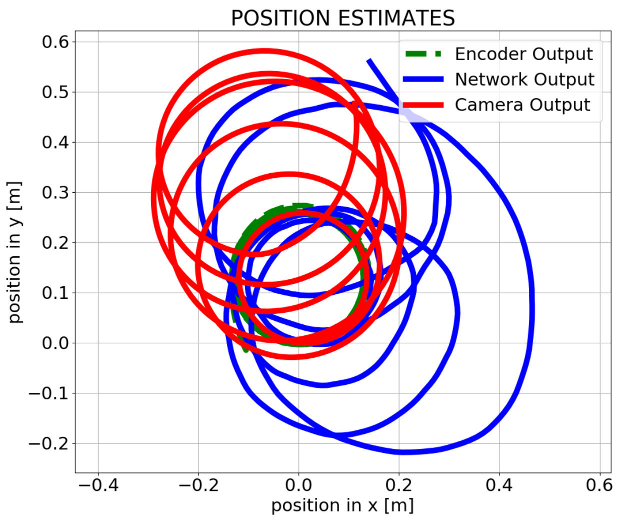


Figure 4.24: Position Estimates of Experiment 6

Position estimates calculated from the network output have high error due to the cumulative errors in the speed estimate and the yaw angle estimates. Although network is able to estimate and reduce the slippage at a lower level, it does not completely corrects speed and yaw angle estimates. But larger path radius compared to odometry and closer covered distance to ground truth means it is able to estimate some of the slips.

### 4.2.7 Experiment 7

In the last experiment, acceleration of the robot is expected to be $0.04m/s^2$ according to equation 2.15 and the acceleration input sent to the motion controllers of the DC motors connected to wheels of the robot.
In this expeiment, network guesses speed of the robot and yaw angle of the robot much better compared to other experiments, and covered distance and position estimates are much closer to the camera data. Speed estimates of the network is close to the ground data, as expected(Figure 4.25).

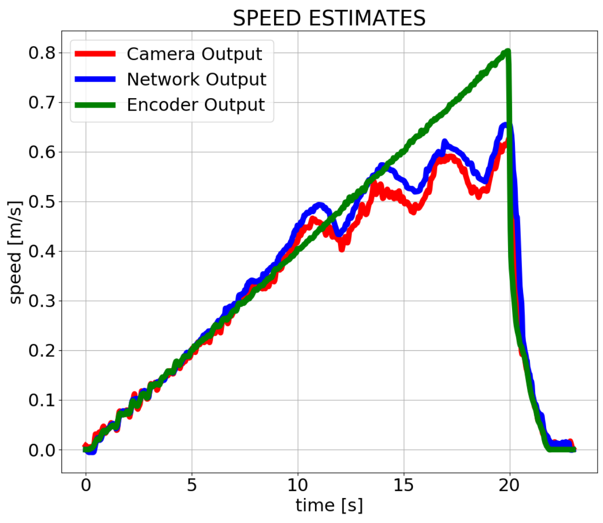


Figure 4.25: Speed Estimates of Experiment 7

Error of the odometry data while calculating covered distance is reduced by the algorithm by enhancing speed estimates(Figure 4.26).

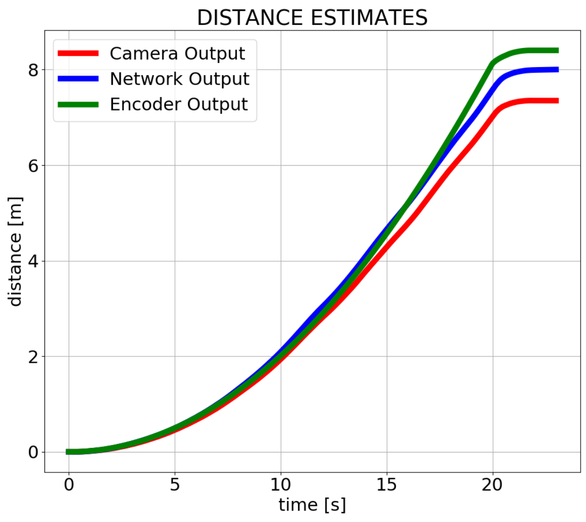


Figure 4.26: Covered Distance Estimates of Experiment 7

The estimate done by the network is consideraby better than yaw estimates of IMU and odometry(Figure 4.27).

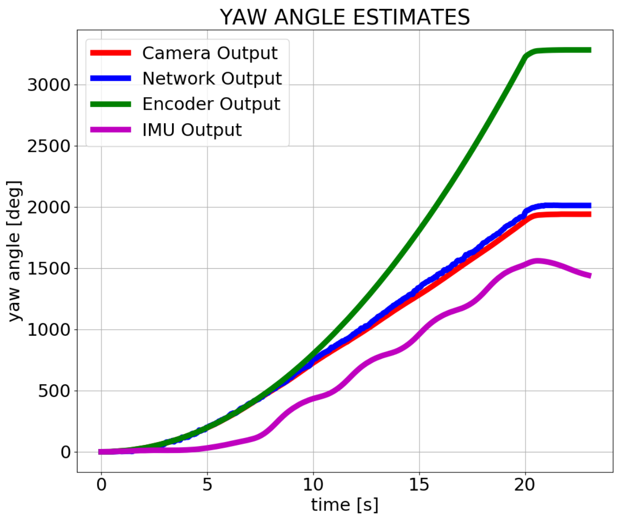


Figure 4.27: Yaw Angle Estimates of Experiment 7

Since speed and yaw angle estimates have relatively low errors compared to 6 experiments presented before, position estimates are closer to the camera output(Figure 4.28).

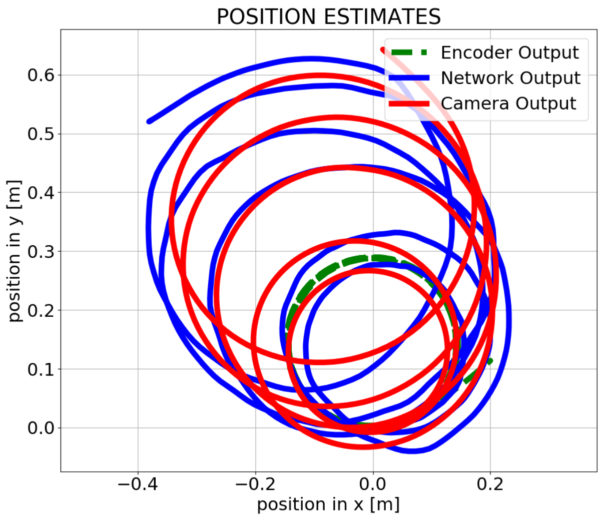


Figure 4.28: Position Estimates of Experiment 7

Since position estimates of the network are close to the ground truth, network is able to reduce errors due to slippage. Covered distance, yaw angle and the position estimates are close to the ground truth, moreover radius of the path of position estimates of the network is close to the radius of path obtained from the ground truth. Hence, it can be said that network is able to compensate lateral and longitudinal slips of the robot.

## 4.3 Error Tables

In this section, errors of the odometry measurements and the output of the trained network is compared and tabulated for speed of the robot, covered distance by the robot and yaw angle of the robot. For the yaw angle comparison, output of the IMU is also considered. In the following equations, $V_{ex}$, $V_c$, $V_{network}$ represent speed of the robot measured by the encoders, camera and speed output of the network, $d_e$, $d_c$, $d_{network}$, represent distance covered by the robot measured by encoders, camera and output of the network, $\psi_e$, $\psi_c$, $\psi_{imu}$, $\psi_{network}$ represent yaw angle of the of the robot measured by encoders, camera IMU and output of the network respectively. $e_{s,e}$, $e_{s,network}$ represents speed measurement errors of encoders and network, $ep_{s,e}$, $ep_{s,network}$ represents speed measurement error percentages of encoders and network, $e_{d,e}$, $e_{d,network}$ represents covered distance measurement errors of encoders and network, $ep_{d,e}$, $ep_{d,network}$ represents covered distance measurement error percentages of encoders and network, $e_{y,e}$, $e_{y,network}$, $e_{y,imu}$ represents yaw angle measurement errors of encoders, network and IMU, $ep_{y,e}$, $ep_{y,network}$, $ep_{y,imu}$ represents yaw angle measurement error percentages of encoders, network and IMU respectively.

$$e_{s,e} = \frac{1}{m}\sum_{i=1}^{m}\left|V_{ex}(i) - V_c(i)\right| \tag{4.1}$$

$$e_{s,network} = \frac{1}{m}\sum_{i=1}^{m}\left|V_{network}(i) - V_c(i)\right| \tag{4.2}$$

$$e_{d,e} = \frac{1}{m}\sum_{i=1}^{m}\big|d_e(i) - d_c(i)\big| \tag{4.3}$$

$$e_{d,network} = \frac{1}{m}\sum_{i=1}^{m}\big|d_{network}(i) - d_c(i)\big| \tag{4.4}$$

$$e_{y,e} = \frac{1}{m}\sum_{i=1}^{m}\big|\psi_e(i) - \psi_c(i)\big| \tag{4.5}$$

$$e_{y,network} = \frac{1}{m}\sum_{i=1}^{m}\big|\psi_{network}(i) - \psi_c(i)\big| \tag{4.6}$$

$$e_{y,imu} = \frac{1}{m}\sum_{i=1}^{m}\big|\psi_{imu}(i) - \psi_c(i)\big| \tag{4.7}$$

$$ep_{s,e} = \frac{1}{m}\sum_{i=1}^{m}\frac{\big|V_{ex}(i) - V_c(i)\big|}{V_c(i)} \tag{4.8}$$

$$ep_{s,network} = \frac{1}{m}\sum_{i=1}^{m}\frac{\big|V_{network}(i) - V_c(i)\big|}{V_c(i)} \tag{4.9}$$

$$ep_{d,e} = \frac{1}{m}\sum_{i=1}^{m}\frac{\big|d_e(i) - d_c(i)\big|}{d_c(i)} \tag{4.10}$$

$$ep_{d,network} = \frac{1}{m}\sum_{i=1}^{m}\frac{\big|d_{network}(i) - d_c(i)\big|}{d_c(i)} \tag{4.11}$$

$$ep_{y,e} = \frac{1}{m}\sum_{i=1}^{m}\frac{\big|\psi_e(i) - \psi_c(i)\big|}{\psi_c(i)} \tag{4.12}$$

$$ep_{y,network} = \frac{1}{m}\sum_{i=1}^{m}\frac{\big|\psi_{network}(i) - \psi_c(i)\big|}{\psi_c(i)} \tag{4.13}$$

$$ep_{y,imu} = \frac{1}{m}\sum_{i=1}^{m}\frac{\big|\psi_{imu}(i) - \psi_c(i)\big|}{\psi_c(i)} \tag{4.14}$$

For error equations, all of the measurement points are used and difference to ground truth is calculated and mean of the error values of the error values for each measurement method are found and tabulated. For the error percentages, same procedure is followed, but the data points where denominator term is close to zero is excluded due to the near-singularity. To prevent cancelling-out due to the sign difference, absolute values of the errors and error percenteges are used in error calculations.

Table 4.1: Errors of the Speed Estimates

| Experiment # | Odometry Error (m/s) | Trained Network Error (m/s) |
|---|---|---|
| Experiment 1 | 0.16358 | 0.08626 |
| Experiment 2 | 0.13586 | 0.08161 |
| Experiment 3 | 0.08305 | 0.06938 |
| Experiment 4 | 0.04711 | 0.03259 |
| Experiment 5 | 0.12101 | 0.03449 |
| Experiment 6 | 0.08672 | 0.02779 |
| Experiment 7 | 0.05281 | 0.02808 |

Mean error of the speed estimates are significantly better in the outputs of the recurrent deep network. From this result, proposed recurrent deep neural network can be said to reduce errors due to the slippages of the wheels.

Table 4.2: Errors of the Covered Distance Estimates

| Experiment # | Odometry Error (m) | Trained Network Error (m) |
|---|---|---|
| Experiment 1 | 1.38230 | 0.77261 |
| Experiment 2 | 0.99202 | 0.68340 |
| Experiment 3 | 0.63162 | 0.66094 |
| Experiment 4 | 0.37933 | 0.19023 |
| Experiment 5 | 0.86379 | 0.15810 |
| Experiment 6 | 0.53440 | 0.10066 |
| Experiment 7 | 0.32364 | 0.25909 |

Since speed estimates are enhanced, covered distance estimates are also enhanced. Mean of the errors of the deep network is better than the odometry measuremets.

Table 4.3: Errors of the Yaw Angle Estimates

| Experiment # | Odometry Error (deg) | Trained Network Error (deg) | IMU Error (deg) |
|---|---|---|---|
| Experiment 1 | 938.72 | 57.80 | 287.15 |
| Experiment 2 | 802.82 | 33.79 | 115.61 |
| Experiment 3 | 494.99 | 136.70 | 270.10 |
| Experiment 4 | 298.99 | 47.31 | 253.95 |
| Experiment 5 | 929.08 | 60.12 | 259.46 |
| Experiment 6 | 714.56 | 48.77 | 242.10 |
| Experiment 7 | 430.95 | 31.94 | 260.41 |

Mean of the yaw angle errors are considerably decreased by the proposed recurrent deep neural network. Output of the deep network is significantly better than odometry and IMU estimates in all experiments.

Table 4.4: Durations to Reach 50% Speed Estimation Errors for IMU and Network Output

| Experiment # | IMU Duration (s) | Trained Network Duration (s) |
|---|---|---|
| Experiment 1 | 0.81 | 16.44 |
| Experiment 2 | 2.74 | 11.96 |
| Experiment 3 | 0.76 | 16.84 |
| Experiment 4 | 0.81 | 20.36 |
| Experiment 5 | 2.02 | 20.74 |
| Experiment 6 | 1.97 | 20.81 |
| Experiment 7 | 1.45 | 20.50 |

In the above table, reaching durations of 50% speed estimation errors for IMU and trained network is compared. From the table, it can be seen that durations to reach 50% speed estimation error is largely increased by the network, which means this system can be used for longer durations in absence of the ground truth.

## 4.4 Comparisons to Literature Results

There are previously done works in literature for indoor and outdoor applications. Both 2 wheeled, 4 wheeled and 6 wheeled robots are used in those works and experiments. In tthis section, results of the literature will be compared to the results obtained in this work. Since robot in this work is adjusted to slide in relatively low speeds and robot speeds, robot types, ground surface properties, etc. are not the same with the literature, comparisons will not show exact performance of the proposed network compared to literaure, however it will give general idea about the performance of the network.

Chonnaparamutt, Winai and Kawasaki [5] used fuzzy estimators to estimate speed of a pruning robot. In the experiments, robot is moved around $0.1\,m/s$, and in average encoder measurements have $11.09\,\%$ error and fuzzy estimator has $2.74\,\%$ speed estimation error.

Sekimori and Miyazaki [13] used four optical mouse sensors to improve covered ditance and yaw angle estimations of a differentially driven mobile robot. They moved robot in a $2.5m$ path and this path consists of straigths and 90 degree turns. Average speed of the robot is $500\,mm/s$ and in the end of the experiments, encoders are measured covered distance by an average of 239.4 mm (9.58%), optical mouse sensors measured covered distance by $61.6\,mm$ average error (2.46%). Encoders measured yaw angle by $19.26\,deg$ average error (5.35%) and optical mouse sensors measured yaw angle by $8.58\,deg$ average error(2.38%).

Seyr and Jakubek [4] combined odometry and IMU measurements to estimate position of a two wheeled differentially driven robot. They did experiments in a rectangular path. Total distance covered during those experiments is $2.4\,m$ and in the end of the experiments, they estimated distance by $2.5\,\%$ average error.

Ojeda et al.[7] used driven DC motor currents to determine wheel slippages of a 6 wheeled outdoor terrain vehicle. Robot they used covered around $14\,m$ during the experiment and they estimated position of the robot with $0.6\,\%$ error.

Since those results are presented in percentages, error percentege tables for the proposed method is presented below.

Table 4.5: Error Percentages of the Speed Estimates

| Experiment # | Odometry Error (%) | Trained Network Error (%) |
|---|---|---|
| Experiment 1 | 31.60 | 16.40 |
| Experiment 2 | 26.69 | 16.73 |
| Experiment 3 | 14.67 | 13.01 |
| Experiment 4 | 9.28 | 6.36 |
| Experiment 5 | 24.23 | 6.83 |
| Experiment 6 | 16.43 | 5.26 |
| Experiment 7 | 10.13 | 5.63 |

In all of the experiments, error percentages of the encoders decreased significantly. From the speed error percentages, it can be seen that even for longer runs and higher speeds, network output is comparable to Chonnaparamutt, Winai and Kawasaki [5] in speed estimates especially in experiments 4, 5, 6. In this work, robot is designed to slide in low speeds, robot is operated in much higher speeds than pruning robot tested by Chonnaparamutt, Winai and Kawasaki and robot covereed much higher distance at those experiments. Despite that, results of the proposed network is comparable.

Table 4.6: Error Percentages of the Covered Distance Estimates

| Experiment # | Odometry Error (%) | Trained Network Error (%) |
|---|---|---|
| Experiment 1 | 19.71 | 12.75 |
| Experiment 2 | 11.87 | 9.52 |
| Experiment 3 | 9.98 | 12.23 |
| Experiment 4 | 8.11 | 5.96 |
| Experiment 5 | 9.35 | 2.25 |
| Experiment 6 | 6.02 | 1.36 |
| Experiment 7 | 4.92 | 5.44 |

Covered distance error percentages are comrarable to works of Sekimori and Miyazaki [13], Seyr and Jakubek [4] in experiments 4, 5 and 6. Network gives better results in latest experiments since recurrent neural network updates itself and latest trainings are more effective. Work of Ojeda et al.[7] gives much better results, however robot they used is completely different since it has 6 driven wheels. Again, comparison, covered distance bby the robot in this work is much higher than literature except Ojeda et al.[7].
Yaw angle estimates are improved significantly by the network. Sekimori and Miyazaki [13] improved average error of yaw angle estimates from $5.35\%$ to $2.38\%$. In this work, this improvement is much more notable. Moreover, despite running experiments in higher mobile robot speeds and at longer distances, results of the output are comparable to work of Sekimori and Miyazaki [13].In addition to these, robot used in this work is adjusted to slide in low speeds and experiments are done in higher speeds as it can be seen from tha high errors in the encoders measurements of this work. Later in this section, results of the network and the literature will be compared again for covered distance of $5.5\,m$ .

Table 4.7: Error Percentages of the Yaw Angle Estimates

| Experiment # | Odometry Error (%) | Trained Network Error (%) | IMU Error (%) |
|---|---|---|---|
| Experiment 1 | 43.67 | 6.20 | 24.72 |
| Experiment 2 | 42.06 | 3.31 | 9.81 |
| Experiment 3 | 24.39 | 7.47 | 29.81 |
| Experiment 4 | 14.42 | 3.56 | 25.43 |
| Experiment 5 | 38.45 | 5.28 | 23.09 |
| Experiment 6 | 32.24 | 4.46 | 22.12 |
| Experiment 7 | 19.80 | 1.93 | 24.85 |

Up to this point, experiment results are compared to literature, but in literature, robot covered around $2.5\,m$ distance. Since in dead reckoning errors accumulate, it will be better to compare results in similar distances. Hence ın the same 7 experiments are used again, however in this time, experiments are not evaluated after robot is covered

$5.5\,m$ distance.

Table 4.8: Error Percentages of the Speed Estimates for 5.5 m Covered Distance

| Experiment # | Odometry Error (%) | Trained Network Error (%) |
|---|---|---|
| Experiment 1 | 22.82 | 12.24 |
| Experiment 2 | 16.07 | 12.49 |
| Experiment 3 | 8.13 | 10.77 |
| Experiment 4 | 7.75 | 4.86 |
| Experiment 5 | 15.64 | 4.76 |
| Experiment 6 | 8.29 | 3.47 |
| Experiment 7 | 6.19 | 4.95 |

Trained network gives error percentage around $4.5\%$ in experiments 4, 5, 6 and 7. It is close to the result obtained by Chonnaparamutt, Winai and Kawasaki [5] despite using different robot and higher speeds which will increase slippages in the wheels.

Table 4.9: Error Percentages of the Covered Distance Estimates for 5.5 m Covered Distance

| Experiment # | Odometry Error (%) | Trained Network Error (%) |
|---|---|---|
| Experiment 1 | 13.29 | 10.15 |
| Experiment 2 | 5.79 | 6.25 |
| Experiment 3 | 7.65 | 10.52 |
| Experiment 4 | 7.39 | 6.16 |
| Experiment 5 | 4.19 | 1.58 |
| Experiment 6 | 2.77 | 0.91 |
| Experiment 7 | 3.55 | 5.00 |

Covered distance results of the network in experiments 4, 5, 6 and 7n are comparable to Sekimori and Miyazaki [13], Seyr and Jakubek [4].

Table 4.10: Error Percentages of the Yaw Angle Estimates for 5.5 m Covered Distance

| Experiment # | Odometry Error (%) | Trained Network Error (%) | IMU Error (%) |
|---|---|---|---|
| Experiment 1 | 28.50 | 7.48 | 26.18 |
| Experiment 2 | 24.20 | 3.05 | 9.47 |
| Experiment 3 | 12.98 | 4.76 | 30.88 |
| Experiment 4 | 10.62 | 3.30 | 25.73 |
| Experiment 5 | 24.83 | 6.01 | 24.81 |
| Experiment 6 | 19.78 | 4.89 | 23.74 |
| Experiment 7 | 12.61 | 1.68 | 25.73 |

Yaw angle estimates are again comparable to Sekimori and Miyazaki [13] . However, since test surface conditions, wheel types, dynamics of the robot used in the experiments and experiment durations are different, to compare and evaluate performances of the different algorithms, same experiment with same robots on the same test ground should be done.

## 4.5 Conclusion and Discussion

In this chapter, results of the speed estimations, covered distance estimations, yaw angle estimations and position estimations of the robot obtained from the trained network, odometry and ground truth are plotted and compared for 7 experiments those not used in the training process of the recurrent deep neural network. For the yaw angle comparisons and plots, output of the IMU is also considered. Performance of the estimates are compared by tabulating the errors from ground truth for each experiments. Lastly, results of the this work and the lierature is compared.

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

In this work, errors on the speed and yaw angle data due to the slippages on the wheels of a 2 wheeled differentially driven mobile robot are reduced using a recurrent deep neural network. In the first chapter, literature survey and the motivation of this thesis is explained. Test set-up and the robot used in the experiments, sensors used in the experiments, basic equations used to mainupulate sensor data and comparison of the data obtained from different sensors are presented in chapter 2. Two of the basic neural networks, the recurrent deep neural network used in this work, inputs of this deep neural network and training process of the proposed network is explained in chapter 3. In the chapter 4, results of the proposed network and comparison of the network output with odometry, IMU data and the lierature results are presented. Moreover, errors of the network output, odometry and the IMU data are tabulated and compared. Proposed network's ability to capture and compensate errors due to the wheel slips in the speed estimates and the yaw angle estimates is evaluated according to the error comparisons. Reduced errors in speed and yaw angle estimates mean that is is better than dead reckoning, hence deep networks can be used to get position estimates in when the ground truth data is interrupted. Network is able to reduce erros in all experiments and it can be used to improve navigation of two wheeled mobile robots. Since most of the mobile robots have encoders and also an IMU, after a simple training of the network, navigation data can be obtained from the trained network with lower error compared to odometry and IMU. Since it is not requires system model and system parameters, it is easy to apply in different systems. Specifically for the systems where frequency of the ground data is low or interrupted, this method can be used in the intervals between receiving ground truth data. Also, since proposed network is recurrent, received data can be used to train network to adapt changing road or surface conditions.

While training network, to improve performance of the network, loads on the tires of the robot are added to training data. Accelerometer data of the IMU with the known weight and center of mass of the robot are used to obtaing normal loads on the wheels. However, since information used in this calculation except the center of mass and weight, which are constants and are already sent to network, this information is redundant. Hence it did not improve the performance of the system. Moreover, to improve position estimates, robot position is used as output of the network instead of the speed and the yaw angle of the robot. In position calculations, current calculation depends on all the previous position calculations. Because of this, Errors in the beginning of the experiment propogates through the experiment and this reduces performance of the nerwork significantly. Output of the network trained by the position outputs are worse than the network trained by the speed and yaw angle of the robot.

## 5.2 Future Work

Same surface, same robot and same set of wheels are used in this work. For different tires, ground surface conditions and for different inclinations, method can be tested. Using two cascade networks, first surface type can be determined and later according to the type of the surface, second network can be adjusted to provide required navigation data for the robot.

Secondly, to make proposed network applicable to other mobile robots too, sensors chosen to be easy to purchase and easy to use. Numbers of the sensors can be increased and different sensors can be added to the system. For example, using a camera on the robot, optic flow can be used as position input to the system, by this way accuracy of the system can be improved.

Thirdly, network can be tested for four wheeled outdoor robots to evaluate performance of the system for outdoor conditions with more than 2 wheeled robots. Since surface conditions of the road will change constantly, recurrent deep neural network's ability to adapt changes can be observed by this way.

# REFERENCES

[1] C. C. Ward and K. Iagnemma, “A dynamic-model-based wheel slip detector for mobile robots on outdoor terrain,” *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 821–831, 2008.

[2] F. Chenavier and J. Crowley, “Position estimation for a mobile robot using vision and odometry,” *IEEE International Conference on Robotics and Automation*, pp. 2588–2593, 1992.

[3] F. Gustafssont, “Slip-based tire-road friction estimation*,” vol. 33, no. 6, pp. 1087–1099, 1997.

[4] M. Seyr and S. Jakubek, “Proprioceptive navigation, slip estimation and slip control for autonomous wheeled mobile robots,” *2006 IEEE Conference on Robotics, Automation and Mechatronics*, 2006.

[5] W. Chonnaparamutt and H. Kawasaki, “Fuzzy systems for slippage control of a pruning robot,” pp. 1270–1275, 2009.

[6] W. Hwang, J. Park, H.-i. Kwon, and M. Anjum, “Vision tracking system for mobile robots using two Kalman filters and a slip detector,” *2010 International Conference on Control Automation and Systems (ICCAS),*, pp. 2041–2046, 2010.

[7] L. Ojeda, D. Cruz, G. Reina, and J. Borenstein, “Current-based slippage detection and correction for mobile robots and planetary rovers,” vol. 22, no. 2, pp. 366–378, 2006.

[8] A. Bhoraskar and P. Sakthivel, “A review and a comparison of Dugoff and modified Dugoff formula with Magic formula,” *2017 International Conference on Nascent Technologies in Engineering, ICNTE 2017 - Proceedings*, no. 1, pp. 1–4, 2017.

[9] H. Pacejka and E. Bakker, " The Magic Formula Tyre-Model," *Proceedings of 1st International Colloquium on Tyre Models for Vehicle Dynamics Analysis*, pp. 1–18, 1991.

[10] G. Bayar, a. B. Koku, and E. I. Konukseven, "Dynamic modeling and parameter estimation for traction, rolling, and lateral wheel forces to enhance mobile robot trajectory tracking," *Robotica*, pp. 1–17, 2014.

[11] H. Zabaleta, D. Valencia, J. Perry, J. Veneman, and T. Keller, "Absolute position calculation for a desktop mobile robot based on three optical mouse sensors,"

[12] A. Bonarini and M. Matteucci, "Dead reckoning for mobile robots using two optical mice," 2004.

[13] D. Sekimori and F. Miyazaki, "Precise dead-reckoning for mobile robots using optical mouse sensors," pp. 145–151, 2007.

[14] H. Fujimoto, N. Takahashi, A. Tsumasaka, and T. Noguchi, "A DVS-MHE approach to vehicle side-slip angle estimation," *Control Applications, . . .*, vol. 2006, no. 2, pp. 4553–4558, 2006.

[15] J. Matuso, I. Petrovic, and N. Pericaculty, "Neural network based tire/road friction force estimation," *Engineering Applications of Artificial Intelligence*, vol. 21, pp. 442–456, 2008.

[16] S. Melzi and E. Sabbioni, "On the vehicle sideslip angle estimation through neural networks: Numerical and experimental results," *Mechanical Systems and Signal Processing*, vol. 25, no. 6, pp. 2005–2019, 2011.

[17] V. Ćirović, D. Aleksendric, and D. Smiljanic, "Longitudinal wheel slip control using dynamic neural networks," *Mechatronics*, vol. 23, no. 1, pp. 135–146, 2013.

[18] F. Tahami, R. Kazemi, and S. Farhanghi, "A novel driver assist stability system for all-wheel-drive electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 52, no. 3, pp. 683–692, 2003.

[19] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural

network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 2012.

[20] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, no. 6, pp. 6645–6649, 2013.

[21] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. June, pp. 2774–2781, 2014.

[22] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[23] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 315–323, 2011.

[24] E. Sert, B. Tuncer, S. B. Okcu, E. Halıcı, F. Gharbalchi, M. Atay, and U. Halıcı, "Hands on Deep Learning," 2017.