

# **Design of Soccer Player to find Distance and Angle on Khepera Platform using Extended Kalman Filter**

*A Thesis*

*submitted in partial fulfilment of the requirement for the award of the Degree of Master of Technology in Intelligent Automation and Robotics under Electronics and Telecommunication Engineering*

*By*

**Tipu Sultan**

**Registration No. 137306 of 2016-17**  
**Examination Roll No. M6IAR19015**

*Under the Guidance of*

**Prof. Amit Konar**

**Department of Electronics and Telecommunication Engineering**  
**Jadavpur University**  
**Kolkata-700032**

May, 2019

FACULTY OF ENGINEERING AND TECHNOLOGY  
JADAVPUR UNIVERSITY

**CERTIFICATE**

This is to certify that the dissertation entitled "**Design of Soccer Player to find Distance and Angle on Khepera Platform using Extended Kalman Filter**" has been carried out by **TIPU SULTAN** ( University Registration No: **137306 of 2016-17**, Examination Roll No: **M6IAR19015** under guidance and supervision and be accepted in partial fulfilment of the requirement for the award of the Degree of Master of Technology in Intelligent Automation and Robotics under Electronics and Telecommunication Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree to any other University or Institute.

-----  
Prof. Amit Konar  
Thesis Supervisor  
Dept. of Electronics & Telecommunication Engineering  
Jadavpur University

-----  
Dr. Sheli Sinha Chaudhuri  
Head of the Department  
Dept. of Electronics & Telecommunication Engineering  
Jadavpur University

-----  
Prof. Chiranjib Bhattacharjee  
Dean  
Faculty of Engineering and Technology  
Jadavpur University

**FACULTY OF ENGINEERING AND TECHNOLOGY  
JADAVPUR UNIVERSITY**

**CERTIFICATE OF APPROVAL\***

The foregoing thesis is hereby approved as a creditable study of an engineering subject and presented in a manner satisfactory to warrant acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for which it is submitted.

**Committee on final examination for the evaluation of the thesis**

-----  
Signature of the Examiner

-----  
Signature of the Supervisor

\*Only in the case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY  
JADAVPUR UNIVERSITY

**DECLARATION OF ORIGINALITY AND COMPLIANCE OF  
ACADEMIC ETHICS**

I declare that this thesis entitled “**Design of Soccer Player to find Distance and Angle on Khepera Platform using Extended Kalman Filter**” contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Technology in Intelligent Automation and Robotics under Electronics and Telecommunication Engineering.

All information has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: **TIPU SULTAN**

Examination Roll No: **M6IAR19015**

Thesis Title: **Design of Soccer Player to find Distance and Angle on Khepera Platform using Extended Kalman Filter**

-----  
Signature of the Candidate

# ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Prof. Amit Konar, Department of Electronics and Telecommunication Engineering, Jadavpur University, for providing me the opportunity to work under his guidance for the final year thesis work. I am highly obliged to him for his phenomenal support and valuable advice helped me to exercise constructive and prodigious approach to the research problem.

I would like to convey my profound appreciation to Dr. Sheli Sinha Chaudhuri, Head of the Department of Electronics and Telecommunication Engineering, Jadavpur University, for her constant support and encouragement. I would also like to express my sincere appreciation and gratitude to all the professors of this department for their priceless opinion and guidance.

I am grateful to my seniors Arnab Rakshit, Lidia Ghosh, Mousumi Laha and Sayantani Ghosh for their unswerving assistance and support. Specifically, Sayantani Ghosh for all the inspiration and moral support which offered me towards fulfilment of this thesis.

I would also like to extend my sincere thanks to my parents for their love, care, support and guidance throughout my life.

Date:

-----

**TIPU SULTAN**

M. Tech in Intelligent Automation and Robotics  
Dept. of Electronics & Telecommunication Engineering  
Examination Roll No: **M6IAR19015**  
Jadavpur University

# *Table of Contents*

<b>Chapter 1</b> .....	1
<b>INTRODUCTION</b> .....	1
1.1 A Brief Insight into Soccer Robotics .....	1
1.2 Differential Drive Mobile Robot .....	1
1.3 Categories of Wheeled Mobile Robots .....	1
1.4 Differential Drive Kinematics .....	3
1.4.1 Forward Kinematics for Differential Drive Robots .....	4
1.4.2 Inverse Kinematics of a robot of a mobile robot .....	5
1.5 Applications .....	6
1.6 Scope of the Thesis .....	6
1.7. Conclusion .....	6
References .....	7
<b>Chapter 2</b> .....	11
<b>KHEPERA IV</b> .....	11
2.1 Introduction .....	11
2.2 Historical Background .....	11
2.3 Specifications of Khepera .....	11
2.4 Accessories of the Robot .....	13
2.5 Safety Precautions .....	29
2.6 Conclusion .....	29
References .....	30
<b>Chapter 3</b> .....	31
<b>SET UP &amp; PROGRAMMING</b> .....	31
3.1 Introduction .....	31
3.2 Communication with Robot .....	33
3.3 Set Up for Programming Interface .....	38

3.4 Programming Requisites .....	40
3.5 Conclusion .....	43
References.....	44
<b>Chapter 4</b> .....	<b>45</b>
<b>Socket Communication</b> .....	<b>45</b>
4.1 Introduction.....	45
4.2 TCP/IP Protocol Suite .....	45
4.3 Establishment of Socket Communication .....	48
4.4 Socket Programming in C .....	49
4.5 Socket Programming in MATLAB .....	51
4.6 Conclusion .....	52
References.....	53
<b>Chapter 5</b> .....	<b>54</b>
<b>Kalman &amp; Extended Kalman Filter</b> .....	<b>54</b>
5.1 Introduction.....	54
5.2 What is estimation? .....	54
5.2.1 Bayes Theorem.....	54
5.2.2 Overview of the Probability Rules .....	55
5.2.3 Mean & Covariance.....	55
5.2.4 MAP - Maximum A-Posteriori Estimation.....	56
5.2.5 MMSE – Minimum Mean Squared Error .....	56
5.2.6 RBE- Recursive Bayesian Estimation.....	56
5.2.7 LSQ- Least Squares Estimation .....	57
5.3 Kalman Filter .....	58
5.3.1 What is a Kalman Filter?.....	58
5.3.2 How It Came to Be Called a Filter .....	58
5.3.3 Its Mathematical Foundations .....	59
5.4 Uses of Kalman Filter.....	59

5.5 Optimal Estimation Methods.....	59
5.5.1 Kalman Filter culmination.....	60
5.5.2 Linear Kalman Filter to Extended Kalman Filter .....	61
5.6 Kalman filter in target tracking and Interception by Mobile Robots .....	62
5.6.1 Schematic diagrams of tracking scheme .....	62
5.6.2 Measurements of the Input to Kalman Filter .....	62
5.7 Extended Kalman Filter.....	64
5.7.1 Predicting target position using Extended Kalman Filter ..	65
5.8 Conclusion .....	66
References.....	67
<b>Chapter 6</b> .....	<b>68</b>
<b>Object Recognition</b> .....	<b>68</b>
6.1 Introduction.....	68
6.2 Edge Detection .....	68
6.2.1 Stages of Edge Detection .....	69
6.2.2 Types of Edge Detectors .....	71
6.2.3 First Order Edge Detection Operators.....	71
6.2.4 Second Order Edge Detection Operators .....	72
6.2.5 Edge Linking Algorithms .....	73
6.2.6 Hough Transform .....	73
6.3 Image Morphology .....	75
6.4 Region Descriptors .....	76
6.5 Conclusion .....	78
References.....	79
<b>Chapter 7</b> .....	<b>81</b>
<b>Implementation of Soccer Goal Keeper</b> .....	<b>81</b>
7.1 Introduction.....	81



7.2 Methodologies Utilized for Implementation .....	81
7.2.1 An overview of the proposed scheme for prediction .....	81
7.2.2 Ball Recognition .....	82
A. Ball Recognition using MATLAB function.....	82
B. Algorithm used in Ball Recognition.....	82
C. Distance and Angle Calculation .....	85
7.2.3 Algorithm used in MATLAB EKF function.....	86
7.3 Results.....	87
7.4 Output from EKF function written in MATLAB.....	88
7.5 Outcome.....	89
7.6 Conclusion .....	89
Reference .....	90
<b>Chapter 8</b> .....	<b>92</b>
<b>Conclusion and Future Direction</b> .....	<b>92</b>
8.1 Summary of the work .....	92
8.2 Future Scope .....	93
<b>APPENDIX</b> .....	<b>i</b>
User guide to run various scripts, codes and function of MATLAB .	i

# *List of Figures*

Figure 1.1 Types of differential drive robots: (a) Pioneer (b) Bigtrak (c) Edison (d)Khepera.....	2
Figure 1.2 Differential Drive kinematics .....	3
Figure 1.3 Forward kinematics of differential drive motor .....	4
Figure 2.1 (a) Left, (b) Rear, (c) Right view of the robot.....	13
Figure 2.2 - 2.3 Bottom view, Top view.....	13
Figure 2.4 Infrared Sensors viewed from Bottom. ....	15
Figure 2.5 IR value vs Distance.....	16
Figure 2.6 Ultrasonic sensors viewed from top. ....	17
Figure 2.7 Contact pads viewed from bottom.....	19
Figure 2.8 Reed relay location .....	20
Figure 2.9 Directions of Detectable Accelerations (TOP view).....	22
Figure 2.10 Directions of Detectable Angular Rates (TOP view).....	23
Figure 2.11 Motor Block with Wheel.....	24
Figure 2.12 Duty Cycle with PWM .....	24
Figure 2.13 Example of Speed Profile.....	27
Figure 2.14 Speed profile using position control.....	28
Figure 2.15 Position Profile .....	28
Figure 3.1 Installation of Light Tool Chain.....	32
Figure 3.2 Execution of the make command .....	33
Figure 3.3 Display Information after Execution of the minicom command .....	34
Figure 3.4 Serial Port Setup Configuration Display.....	34
Figure 3.5 Final Step for Connection via USB .....	34
Figure 3.6 Display after Editing the file ~/etc/wpa_supplicat/wpa_supplicant-wlan0.conf.....	35
Figure 3.7 Display showing the use of address given by Access point.....	35
Figure 3.8 Check for IP address of Robot .....	36
Figure 3.9 Line added to /etc/exports file .....	36
Figure 3.10 Command for mounting the shared directory .....	36
Figure 3.11 Check of mounted folder properties .....	37
Figure 3.12 File transfer using scp.....	37
Figure 4.1 TCP/IP and OSI model.....	45
Figure 4.2 Server-Client Model .....	48

Figure 5.1 Foundational concepts in Kalman filtering .....	59
Figure 5.2 Lifelines of some important contributors to estimation .....	60
Figure 5.3 Schematic of tracking scheme .....	62
Figure 5.4 The tracker observing the motion of the target .....	63
Figure 5.5 The recursive Kalman filter cycle .....	63
Figure 6.1 Types of edges- a) Step edge, b) Ramp edge, c) Spike edge, d) Roof edge .....	68
Figure 6.2 Edge detection Process .....	70
Figure 6.3 Lines in Hough transform.....	74
Figure 7.1 Classical CHT Voting Pattern .....	83
Figure 7.2 Voting Mode: Multiple Radii, Along Direction of Gradient .....	84
Figure 7.3 Ball detection.....	87
A.1 Screen depicting the C code for Soccer Robot .....	I
A.2 Screen for C program compilation.....	II
A.3 Screen showing script which calls BallDetection function .....	III
A.4 Screen showing script which shows the BallDetection Function.....	IV
A.5 Screen showing extended kalman filter. ....	V

## *List of Tables*

Table 2.1 Specifications of the Robot.....	11
Table 2.2 Battery Specification.....	18
Table 2.3 Pin out of the contact pads.....	19
Table 2.4 Camera Specification.....	20
Table 2.5 Specifications of Microphones .....	21
Table 2.6 Specifications of Loudspeaker.....	21
Table 2.7 Specifications of Gumstix Overo FireSTORM-P COM .....	21
Table 2.8 Parameters for defining the Ramp .....	26
Table 3.1 Basic functions and Variables used for Programming .....	41
Table 4.1 Socket creation variables and declaration .....	49

# Chapter 1

## INTRODUCTION

### 1.1 A Brief Insight into Soccer Robotics

During the past few decades, Soccer Robotics [1-5] [58-60] has contributed immensely for the development and growth of various technologies in the fields of Computational [6] and Artificial Intelligence [7]. The primary motivation behind this field is to design autonomous mobile robots [3-10] that can accustom and act in accordance to the real-world environment. The thesis deals with the design of a soccer robot using a differential drive mobile robot [11-15] having certain verifications that are required to perform a particular task in our case predict the distance and angle of the next state. A basic description of differential drive mobile robots has been depicted in the following sections.

### 1.2 Differential Drive Mobile Robot

The differential drive robot is a two-wheeled mobile robot that moves with the aid of separately driven wheels consisting of independent actuators for each wheel. Its name has been derived from the fact that the motion vector of the robot is sum of the independent wheel motions. Its direction of movement can be changed by varying the relative rate of rotation of its wheels without any requirement of additional steering motion. Moreover, this robot also consists of a motion-driven small swivel wheel known as the caster wheel that helps to balance or stabilize its motion.

### 1.3 Categories of Wheeled Mobile Robots

On the basis of the mobility degree ( $m$ ) and steer-ability degree ( $s$ ) classification method by Campion [16], the mobile robots are of the following classes -

**Type (3,0) robots or Omni-directional robots:** These robots have no steering wheels ( $s=0$ ) and are equipped only with Swivel or caster wheels. They have full mobility in the plane ( $m=3$ ), which means that they are able to move in any direction without any reorientation.

**Type (2,0) robots:** They have no steering wheels ( $s=0$ ) but either one or several fixed wheels with a common axle. The common axle restricts mobility to a two-dimensional plane ( $m=2$ )

**Type (2,1) robots:** They have no fixed wheels and at least one steering wheel. If there is more than one steering wheel, their orientations must be coordinated ( $s=1$ ). Therefore, mobility restricted to a two-dimensional plane ( $m=2$ ).

**Type (1,1) robots:** They have one or several fixed wheels on a common axle and also one or several steering wheels, with two conditions for the steering wheels, their centres must not be located on the common axle of the fixed wheels and their orientations must be coordinated ( $s=1$ ). Mobility is restricted to a one-dimensional plane determined by the orientation angle of the steering wheel ( $m=1$ ).

**Type (1,2) robots:** These robots have no fixed wheels, but at least two steering wheels. If there are more than two steering wheels, then their orientation must be coordinated in two groups ( $s=2$ ). Mobility is restricted to a one-dimensional plane ( $m=1$ ) determined by the orientation angles of the two steering wheels.

Some of the well-known differential drive mobile robots are shown below:



(a)



(b)



(b)

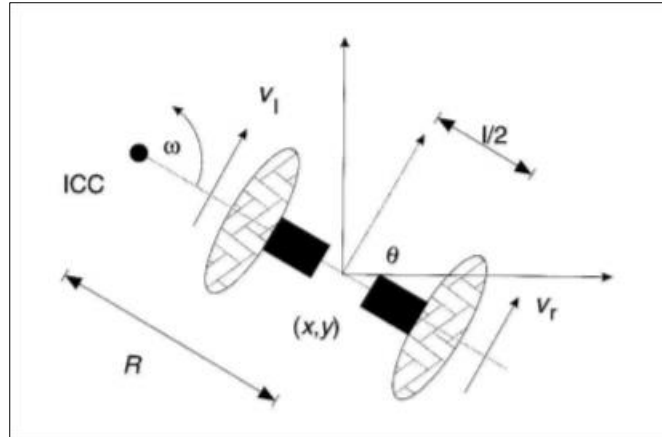


(d)

**Figure 1.1 Types of differential drive robots: (a) Pioneer (b) Bigtrak (c) Edison (d)Khepera IV**

## 1.4 Differential Drive Kinematics

Many mobile robots use a drive mechanism known as differential drive. It consists of 2 drive wheels mounted on a common axis, and each wheel can independently be driven either forward or backward. While we can vary the velocity of each wheel, for the robot to perform rolling motion, the robot must rotate about a point that lies along their common left and right wheel axis. The point that the robot rotates about is known as the ICC - *Instantaneous Center of Curvature* (see figure 1.2).



**Figure 1.2 Differential Drive kinematics**

By varying the velocities of the two wheels, we can vary the trajectories that the robot takes. Because the rate of rotation  $\omega$  about the ICC must be the same for both wheels, we can write the following equations:

$$\omega(R + l/2) = V_r \quad (1.1)$$

$$\omega(R - l/2) = V_l \quad (1.2)$$

where  $l$  is the distance between the centres of the two wheels,  $V_l$ ,  $V_r$  are the right and left velocities along the ground, and  $R$  is the signed distance from the ICC to the midpoint between the wheels. At any instance in time we can solve for  $R$  and  $\omega$ :

$$R = \frac{l}{2} \frac{V_r + V_l}{V_r - V_l} \quad (1.3)$$

$$\omega = \frac{V_r - V_l}{l} \quad (1.4)$$

There are three interesting cases with these kinds of drives.

1. If  $V_l = V_r$ , then we have forward linear motion in a straight line.  $R$  becomes infinite, and there is effectively no rotation -  $\omega$  is zero.
2. If  $V_l = -V_r$ , then  $R = 0$ , and we have rotation about the midpoint of the wheel axis - we rotate in place.
3. If  $V_l = 0$ , then we have rotation about the left wheel. In this case  $R = \frac{l}{2}$ . Same is true if  $V_r = 0$ .

Note that a differential drive robot cannot move in the direction along the axis - this is a singularity. Differential drive vehicles are very sensitive to slight changes in velocity in each of the wheels. Small errors in the relative velocities between the wheels can affect the robot trajectory. They are also very sensitive to small variations in the ground plane, and may need extra wheels (caster wheels) for support.

### 1.4.1 Forward Kinematics for Differential Drive Robots

Let us assume the robot is at some position  $(x, y)$ , headed in a direction making an angle  $\theta$  with the X axis. We assume the robot is centred at a point midway along the wheel axle. By manipulating the control parameters  $V_l, V_r$ , we can get the robot to move to different positions and orientations. With the knowledge of these control parameter and using equation (1.3), the ICC location can be written as:

$$C = [x - R \sin \theta, y + R \cos \theta] \quad (1.5)$$

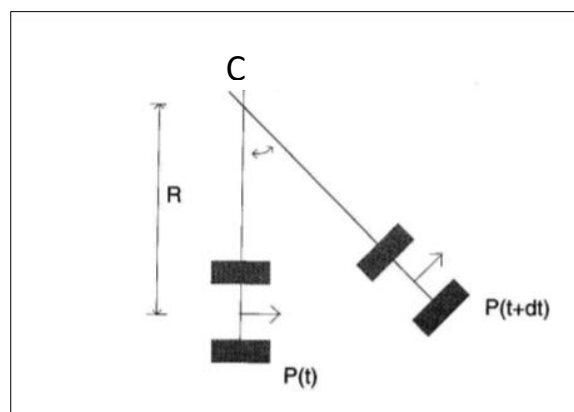
And at time  $t + \Delta t$  the robot's pose will be:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) & 0 \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - C_x \\ y - C_y \\ \theta \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \\ \omega\Delta t \end{bmatrix} \quad (1.6)$$

This equation thus describes the motion of a robot rotating a distance  $R$  about ICC with an angular velocity of  $\omega$ .

Another way to understand this as depicted in figure 1.3 is that the motion of the robot is equivalent to: -

- 1) Translating the ICC to the origin of the coordinate system,
- 2) Rotating about the origin by an angular amount  $\omega$ .



**Figure 1.3 Forward kinematics of differential drive motor**



### 1.4.2 Inverse Kinematics of a robot of a mobile robot

In general, we can describe the position of robot capable of moving in a particular direction  $\theta_t$  at a given velocity  $V(t)$  as:

$$x(t) = \int_0^t V(t) \cos(\theta(t)) dt \quad (1.7)$$

$$y(t) = \int_0^t V(t) \sin[\theta(t)] dt \quad (1.8)$$

$$\Theta(t) = \int_0^t w(t) dt \quad (1.9)$$

For the special case of a differential drive robot the equation become:

$$x(t) = \frac{1}{2} \int_0^t [v_r(t) + v_l(t)] \cos[\theta(t)] dt \quad (1.10)$$

$$y(t) = \frac{1}{2} \int_0^t [v_r(t) + v_l(t)] \sin[\theta(t)] dt \quad (1.11)$$

$$\Theta(t) = \frac{1}{l} \int_0^t [v_r - v_l] dt \quad (1.12)$$

The inverse kinematics problem can be described as the way by which the robot can be controlled to reach a given configuration  $(x, y, \theta)$ . Unfortunately, a differential drive robot imposes what are called non-holonomic constraints [18-20] on establishing its position. For example, the robot cannot move laterally along its axle. A similar non-holonomic constraint is a car that can only turn its front wheels. It cannot move directly sidewise, as parallel parking a car requires a more complicated set of steering maneuvers. So, it's not possible simply an arbitrary robot pose  $(x, y, \theta)$  and find the velocities that will get it to a desired location.

For the special cases of  $v_r = v_l = v$  (robot moving in a straight line) the motion equations become:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + v \cos(\theta) \Delta t \\ y + v \sin(\theta) \Delta t \\ \theta \end{bmatrix} \quad (1.13)$$

If,  $v_r = -v_l = v$  then the robot rotates in place and equations become:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + 2v\Delta t/t \end{bmatrix} \quad (1.14)$$

This provides a strategy of moving the robot in straight line, then rotating for a turn in place, and then moving straight again thus forming a navigation strategy for differential drive robots.

## 1.5 Applications

Differential drive mobile robots are used in a wide range of applications, some of which are listed below:

- They are utilized extensively in various rehabilitative domains [21]
- Hospitals have been using autonomous mobile robots to move materials [22]
- Warehouses have installed mobile robotic systems to efficiently move materials from stocking shelves to order fulfilment zones [23]
- They are used as military robots for various military applications, from transport to search, rescue and attack [24]
- They are used as domestic robots that perform certain household tasks such as vacuuming or gardening [25]
- Mobile robots are also used in sports to entertain the audience [26]

## 1.6 Scope of the Thesis

Soccer robotics is a research field that fosters the development of autonomous mobile robots having the capability to interact efficiently with the physical environment. The soccer game was first proposed by A.K. Mackworth in the year 1993 [27]. Since then, a large number of soccer symposiums and leagues have been conducted by two leading international associations namely Robocup [28-36] and FIRA [37-40]. In these leagues, various types of robots have been used as soccer players like four legged robots [41-45], two legged humanoid robots [47-52] etc. The major real-life areas that are tackled through this game include real time planning [53-54], reasoning [55], behaviour learning [56-57], sensor fusion, strategy acquisition etc.

Though extensive research has been carried in this field, however there still exists a dearth of literature regarding implementation of soccer player using small sized differential drive robots. Moreover, most of the soccer robots still face real time issues like object recognition under a noisy environment, appropriate coordinate update [58-60] during navigation etc. Hence this thesis aims to design a soccer player goal keeper using the recently developed Khepera IV robot which predicts the position i.e., distance and angle of the next state. Though real time approach is not done in this thesis but it dealt with prediction approach using Extended Kalman Filter.

The objective of this thesis is divided into three main parts. First, a vigorous method is discussed that handles the robot basic hardware related components which are required for the thesis and theoretical aspect of EXTENDED KALAMAN FILTER. Second, novel methods for object recognition are depicted that are highly robust to noise while working in real life scenario. Third, the planning, strategy for implementing the goal keeper decision using extended Kalman filter for prediction of distance and angle of the soccer player using the function developed using MATLAB.

## 1.7. Conclusion

This chapter provides basic idea regarding Soccer Robotics and its significance in research domain. It also discusses the preliminary idea regarding differential drive mobile robots, its kinematical modelling. Lastly, it discusses the major motivation behind this thesis work. Thus, this chapter helps to understand the basic concepts of differential drive robots and its application in the field of robotics.

## References

- [1] J.H. Kim, D-H. Kim, Y. Kim, K.-T Seow, Soccer Robotics (Springer Tracts in Advanced Robotics), New York: Springer Verlag, pp. 326, 2004.
- [2] H. H. Lund, "Robot soccer in education". *Adv Robot*, vol. 13, no. 8, pp. 737-752, 2000.
- [3] J.-H. Kim, H. S. Shim. H.-S. Kim, M, J. Jung, I-H. Choi, J. o. Kim, "A cooperative multi-agent system and its real time application to robot soccer", *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 638 643, 1907.
- [4] M.-J. Jung, H. S. Kim, H. S. Shim, J. H. Kim, "Fuzzy rule extraction for shooting action controller of soccer robot", *Proc. IEEE Int. Fuzzy System Conf.*, vol. 1, pp. 556-561 1999.
- [5] Sukhan Lee, Javier Bautista, "Motion Control For Micro-Robots Playing Soccer Games", *Proceedings of the 1998 IEEE Int. Conference on Robotics & Automation Leuven, May 1998*.
- [6] A. Konar, *Comput. Intelligence: Principles Techniques and Applications*, New York: Springer, 2005.
- [7] A. Konar, *Artificial Intelligence & Soft Computing: Behavioural & Cognitive Modelling of the Human Brain*, Florida, Boca Raton: CRC Press LLC, 2000.
- [8] C. M. Wang, "Location Estimation and Uncertainty Analysis for Mobile Robots". *Proc. IEEE International Conference on Robotics and Automation*, pp. 1230-1235, 1988.
- [9] Y. Kanayama, N. Miyake, "Trajectory Generation for Mobile Robots" in *Robotics Research*. The MIT Press, vol. 3, pp. 333-340, 1986.
- [10] Y. Kanayama, A. Nilipour, C. Lelm, "A Locomotion Control Method for Autonomous Vehicles", *Proc. IEEE Conference on Robotics and Automation*, pp. 1315-1317, 1988.
- [11] S. Roh, H. Choi, "Strategy for navigation inside pipelines with differential-drive in pipe robot". *Proc IEEE Int. Conf Robotics Automation*, pp. 2575-2580, 2002,
- [12] D. Chwa, "Tracking control of differential-drive wheeled mobile robots using a backstepping-like feedback linearization", *IEEE Trans. Sust. Man Oybern. A Syst. Humans*, vol 40, no. 6, pp. 1285-1295, Nov. 2010.
- [13] R. Segon, C. HyoukRyeol, "Differential-drive in-pipe robot for moving inside urban gas pipelines", *IEEE Trans. Robot. Automat*, vol. 21, no. 1, pp. 1-17, 2005.
- [14] D. J. Balkcom, M T Mason, "Extremal trajectories for bounded velocity differential *drive robots*", *IEEE International Conference on Robotics and Automation*, 2000.
- [15] D. J. Balkcom, M. T. Mason, "Time optimal trajectories for differential drive vehicles", *Int. J. Robot, Res.*, vol. 21, no 3, pp. 199-217, Mar. 2002.

- [16] Campion, G. Bastin, and B. D'Andréa-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, February 1996.
- [17] G. Dudek, M. Jenkins, Computational Principles of Mobile Robotics, U.K., Cambridge: Cambridge Univ. Press, 2000.
- [18] I. Kolmanovsky, N. H. McClamroch, "Developments in nonholonomic control problems", *IEEE Control Syst. Mag.*, vol. 15, no. 6, pp. 20-36, Dec. 1995.
- [19] Y.-C. Chang, B.-S. Chen, "Adaptive tracking control design of nonholonomic mechanical systems", *Proc Conf. Decision Control*, pp. 4739-4744, 1996.
- [20] T. Fukao, H. Nakagawa, N. Adachi, "Adaptive tracking control of a nonholonomic mobile robot", *IEEE Trans. Robot. Autom.*, vol. 16, no. 5, pp. 609-615, Oct. 2000.
- [21] R. Kumar, P. Berkelman, Gupta, A. Barnes, P.S. Jensen, L.L. Whitcomb, R.H. Taylor, "Preliminary experiments in cooperative human/robot force control for robot assisted microsurgical manipulation", *Proc. IEEE Int. Conf. Robotics Automation*, pp. 610-617, 2000.
- [22] J. M. Evans, "HelpMate: An autonomous mobile robot courier for hospitals", 1994 *Int. Conf on Intelligent Robots and Systems (IROS '94)*, pp. 1695-1700, Sept. 1994.
- [23] F. Y. Zheng, Recent Trends in Mobile Robots, Singapore: World Scientific, 1993.
- [24] J. Vaganay, M. J. Aldon, "Attitude estimation for a vehicle using inertial sensors", *Ist IFAC Int Workshop on Intell. Autonomous Vehicles*, pp. 89-94, 1993.
- [25] O. Yong-Joo, Y. Watanabe, "Development of small robot for home floor cleaning", *Proc. 41st SICE Annu. Conf.*, vol. 5, pp. 3222-3223, 2002.
- [26] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel (Hahnel), G Lakemeyer, D Schulz, W. Steiner, S. Thrun, "Experiences with an interactive museum tour guide robot", *Artif Intell.*, vol. 114, no. 1-, 2000.
- [27] A. Mackworth, "On Seeing Robots" in *Computer Vision: Systems Theory and Applications*, Singapore: World Scientific Press, 1993. H. Kitano, *RoboCup-97: World Cup Robot Soccer I*, LNAI, Springer-Verlag, 1998.
- [28] H. Kitano, "RoboCup: A Challenge AI Problem", *AI Magazine*, Spring 1997.
- [29] H Kitano, "RoboCup Synthetic Agent Challenge 97", *Proc. of IJCAI-97*, 1997.
- [30] S. Tadokoro, "The RoboCup Rescue Project: a multi-agent approach to the disaster mitigation problem", *IEEE International Conference on Robotics and Automation (ICRA00)*, 2000.
- [31] Satoshi Tadokoro, Hiroaki Kitano et al., *The RoboCup-Rescue concept*, The RoboCup Rescue Committee, 1999.
- [32] N. M. Mayer, M. Asada, "RoboCup humanoid challenge", *Int. J. Humanoid Robot.*, vol. 5, no. 3, pp. 335-351, 2008.

- [33] "RoboCup soccer humanoid league rules and setup for the 2014 competition in Joao Pessoa", 2014, [online] Available: <https://www.robocuphumanoid.org>
- [34] D. Budden, P. Wang, O. Obst, M. Prokopenko, "Simulation leagues: Analysis of competition formats" in *RoboCup 2014: Robot Soccer World Cup XVIII.*, Springer, 2014.
- [35] "Middle Size Robot League Rules and Regulations for 2015 Version-17.220141231 5", *MSL Technical Committee*, 5 2015.
- [36] Gupta Gourab Sen, M Muthu kumarr, Wai Lum Chee, Khiang Tan Ser, Quek Jiang Woei, "*The SPECIAL CUBS: Multiagent Co-operative Systems for playing Robot Soccer*", *FIRA Robot World Cup France '98 Proceedings*, pp. 33-35.
- [37] Young D. Kwon, Shin Dong Min, Jin M. Won, Jin S. Lee, "Multi Agents Cooperation Strategy for Soccer Robots", *FIRA Robot World Cup France '98 Proceedings*, pp. 1-6, July 1998.
- [38] P. Thomas, G. Springer, R.J. Stonier, P. Wolfs, "Boundary-avoidance and attack strategy in robot soccer", *FIRA Robot World Cup France '98 Proceedings*, pp. 27-32, July 1998.
- [39] FIRA.org: About fira/brief history. <http://www.fira.net/contents/sub01/sub011.asp>
- [40] C. Zhou and PK. Yue, Robo-Erectus, "A low cost autonomous humanoid soccer robot, *Advanced Robotics*, vol 18,no. 7, pp. 717-720, 2004.
- [41] M. S. Kim and W. Uther, "Automatic gait optimisation for quadruped robots," *Proceedings of the Australasian Conference on Robotics and Automation, Brisbane, Australia, December 2003*.
- [42] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," November 2003.
- [43] J. Chen, E. Chung, R. Edwards, N. Wong, E. Mak, R. Sheh, M. S. Kim, A. Tang, N. Sutanto, B. Hengst, C. Sammut, and W. Uther, "A description of the run swift 2003-legged robot soccer team," University Of New South Wales, Tech. Rep., 2003 [Online]. Available: <http://www.cse.unsw.edu.au/robocup/reports.html>
- [44] P. Stone, K. Dresner, S. Erdogan, P. Fidelman, N. Jong, N. Kohl, G. Kuhlmann, E. Lin, M. Sridharan, D. Stronger, and G. Hariharan, "Ut austin villa 2003: A new robocup fourlegged team," University of New South Wales, Tech. Rep., 2003. [Online]. Available: <http://www.cs.utexas.edu/users/AustinVilla/legged/2003/>
- [45] M. Fujita and H. Kitano, "Development of an autonomous quadruped robot for robot entertainment," *Autonomous Robots*, vol. 5, no. 1, pp. 7-18, 1998.
- [46] M. J. Quinlan, S. K. Chalup, R. H. Middleton, "Techniques for improving vision and locomotion on the sony aibo robot", *Australasian Conference on Robotics and Automation, December 2003*.
- [47] C. Zhou, "Rules for the humanoid league", *Adv. Robot.*, vol. 18, no. 7, pp. 721-724, 2004.
- [48] R. Matsumura, H. Ishiguro, "Development of a high-performance humanoid soccer robot", *Int. J. Humanoid Robot*, vol. 5, no. 3, pp. 353-373.

- [49] S. Behnke, J. Stückler, "Hierarchical reactive control for humanoid soccer robots", *Int. J. Humanoid Robot*, vol. 5, no. 3, pp. 375-396.
- [50] M. Friedmann, J. Kiener, S. Petters, D. Thomas, O. von Stryk, H. Sakamoto, "Versatile high-quality motions and behaviour control of a humanoid soccer robot", *Int. J. Humanoid Robot.*, vol. 5, no. 3, pp. 417-436, 2008.
- [51] Sven Behnke Michael Schreiber, Maren Bennewitz, Jörg Stückler, Hauke Strasdat, Kabanab ROBOTIK 2006), 06/2006 Johannes Schwenk, "Designing a team of soccer-playing humanoid robots", *Proceedings 37th International Symposium on Robotics (ISR 2006) and 4th German Conference on Robotics (ROBOTIK 2006)*, 06/2006.
- [52] K. Hirai, "Current and Future Perspective of Honda Humanoid Robot", *Proc. of IROS-97*, 1997.
- [53] Byoung-Tak Zhang and Sung-Hoon Kim.(1997) "An evolutionary method for active Automation, pp. 312-317, 1997. Learning of mobile robot path planning". *Computational Intelligence in Robotics and Automation*, pp. 312-317, 1997.
- [54] R. Barman, S. Kingdon, J. Little, A. K. Mackworth, D. Pai, M. Sahota, H. Wilkinson, Y. Zhang. "Dynamo real-time experiments with multiple mobile robots", *Proceedings of Intelligent Vehicles Symposium*, pp. 261-266, 1993.
- [55] T. Balch, R. C. Arkin, "Behavior-based formation control for multirobot systems", *IEEE Transactions on Robotics and Automation*, vol. 14, no. 12, pp. 926-939, 1998.
- [56] TH Lee, "A practical Fuzzy Logic Controller for the Path Tracking of Wheeled Mobile Robots", *IEEE Control Systems Magazine*, pp. 60-65, Apr. 2003.
- [57] D. Gu, H. Hu, "Evolving Fuzzy Logic Controllers for Sony Legged Robots", *RoboCup 2001 Lecture notes in artificial intelligence*, vol. 2377, pp. 356-361, 2000.
- [58] L. Jetto, S. Longhi and G. Venturini, "Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots," in *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 219-229, April 1999.
- [59] F. Chenavier and J. L. Crowley, "Position estimation for a mobile robot using vision and odometry," *Proceedings 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992, pp. 2588-2593 vol.3.
- [60] J. Lu, C. Li and W. Su, "Extended Kalman filter based localization for a mobile robot team," *2016 Chinese Control and Decision Conference (CCDC)*, Yinchuan, 2016, pp. 939-944.

# Chapter 2

## KHEPERA IV

### 2.1 Introduction

Khepera IV [1-3] is a miniature differential wheeled mobile robot that acts as an important tool for indoor research and educational purposes, developed by K-Team [4], Switzerland. It aids in real world testing of various algorithms and methodologies like trajectory planning [5-7], obstacle avoidance [8-9], object recognition [10-12] etc.

### 2.2 Historical Background

Observing the evolution of robotic technology in 90's, Prof. Jean-Daniel Nicoud, head of the Micro computing Laboratory (LAMI) [13-14] of the Swiss Federal Institute of Technology of Lausanne (EPFL), decided to start development of a mobile robot occupying less than one cubic inch of volume. So, in 1991 this laboratory developed a miniature mobile robot named Khepera. In the beginning, the robot became affectionately known as "cafard" by the people of the lab, which is French for "cockroach". However, Prof Nicoud thought that this name was lacking some elegance and hence finally named it "Khepera" [15-16] which was the name of an Egyptian god having the head of a cockroach.

### 2.3 Specifications of Khepera

The technical details and specifications of the robot are provided in the table below:

**Table 2.1 Specifications of the Robot**

ELEMENTS	TECHNICAL INFORMATION
Processor	Linux core running on a 800MHz ARM Cortex-A8 Processor with C64x Fixed Point DSP core and additional microcontroller for peripherals management
RAM	512 MB
Flash	512 MB plus additional 4GB for data
Motion	2 DC brushed motors with incremental encoders (roughly 147 pulses per mm of robot motion) and gearbox
Sensors	8 Infra-red proximity and ambient light sensors with up to 25cm range, 4 Infrared ground proximity sensors for line following applications and fall avoidance, 5 Ultrasonic sensors with range 25 cm to 2 meters, 3-axis accelerometer and 3-axis gyroscope

Audio	2x1 embedded microphone 100 to 10,000 Hz, 1x 0.7W speaker (400-20,000Hz)
Speed	Max 1 m/s in openloop and 0.8 m/s with Factory default PID speed controller. Min 0.003 m/s with Factory default PID speed controller
LED	3 programmable RGB LED on top of the robot
Video	Integrated color camera (752x480 pixels, 30FPS)
AC adapter power	9V @ 2.5A
Autonomy	Approximately 7 hours, Additional turrets will reduce battery life
Battery	Embedded battery, 7.4V Lithium Polymer, 3400mAh
Size	Diameter: 140 mm, Height: 58 mm
Docking	Ready for docking (Power input and I2C communication)
Communications	1x USB 2.0 host (500mA), 1x USB 2.0 device, 802.11 b/g WiFi, Bluetooth 2.0 EDR
Weight	540g
Max, payload	Approx. 2000 g
Extension Bus	Expansion modules can be added to the robot using the KB-250 bus.
Ground clearance	4 mm. Use only on hard and flat surfaces
Development Environment for Autonomous Application	GNU C/C++ compiler, for native on-board applications.
Turn radius	0 cm
Operating temperature	0-40°C
Other Languages	Python 2.79 and higher
3 Axis Gyroscope and Accelerometer	Gives powerful array of information of position, velocity, acceleration, orientation
Bluetooth and WiFi	WiFi with IEEE 802.11 standard included, Bluetooth version 3.0 has been included for wireless communication
High Quality and High Accuracy DC Motor	2 DC motor of max 1.96 W is used with gear ratio 19:1 and an efficiency of 78%



## 2.4 Accessories of the Robot

### 2.4.1 A Quick Look into the features

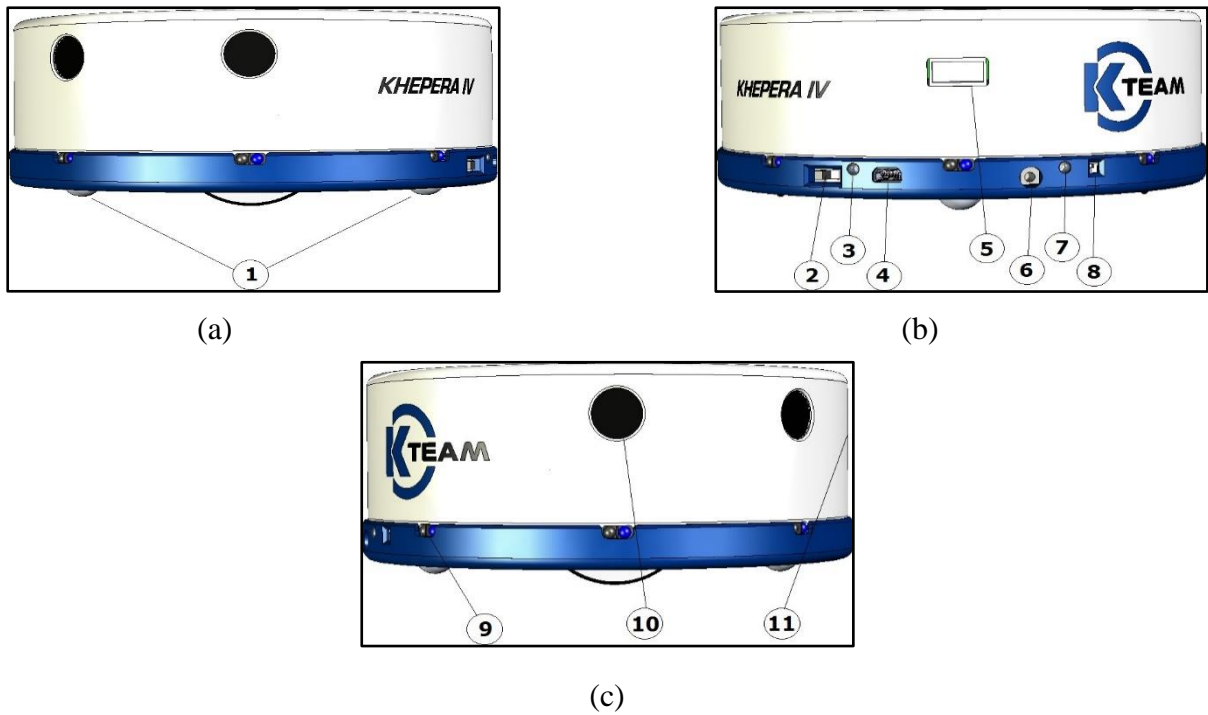


Figure 2.1 (a) Left, (b) Rear, (c) Right view of the robot

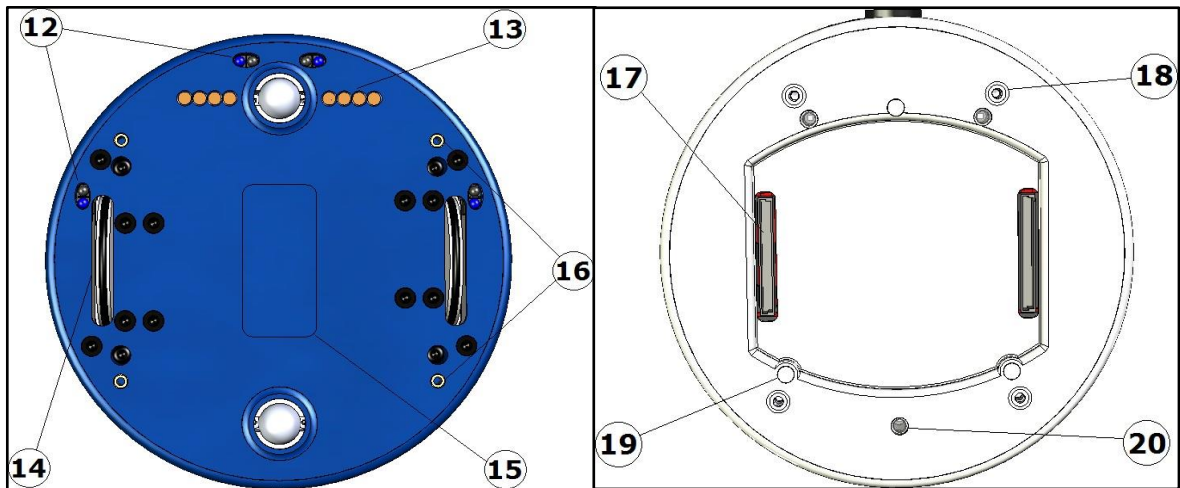


Figure 2.2 Bottom view

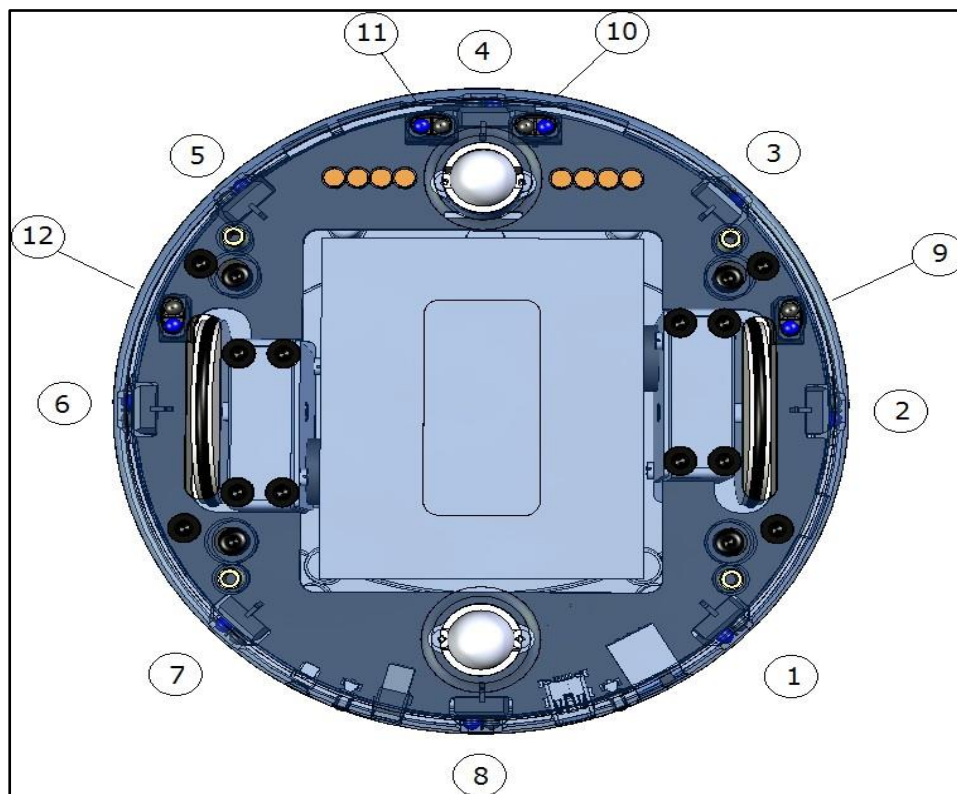
Figure 2.3 Top view

1. **Caster wheels:** The robot consists of two caster wheels that provides it with stability even in the presence of high payload or with long cantilever extension modules.
2. **On/Off switch:** This switch enables to turn on the robot by moving it inwards and turn off the robot by moving it outwards. Moreover, this switch has its effect on the internal regulators of the robot rather than the battery i.e., the robot can be charged even if it's in the off condition.
3. **Status LED:** This bicolour indication LED indicates the state of the robot. When it is turned on, the green LED will stay on and the red light will blink until the system is ready. Moreover, this LED is also user-controllable.
4. **Mini-USB B connector:** This connector creates a communication link between the robot and a computer. However, it is not possible to charge the robot by this way.
5. **USB A connector:** This is a USB host-competent connector. It can be plugged into any USB device as required as long as it doesn't draw more than 500mA of current.
6. **Power supply jack:** This is the 0.65mm centre positive jack used to charge the internal battery of the robot. Its input voltage is 9V. Current drawn by the robot is 1A, so a 1.5+A adapter is required.
7. **Charging status LED:** This is a bicolour indication LED representing the state of the charging. There are two modes, depending if the robot is powered or not. If the robot is off and the AC adapter is plugged, the red LED will be on as long as the battery is charging. Once the charge terminated, the LED will turn off. If the robot is on and the AC adapter is plugged, the red LED will be on as long as the battery is charging. Once the charge terminated, the red LED will turn off and the green LED will turn on.
8. **Reset button:** This button serves to reset the whole robot, including the extension modules.
9. **Infrared sensors:** There are 8 infrared sensors all around the robot, each separated from its neighbour by an angle of 45°, that enables it to detect obstacles or measure ambient light. With these sensors, the robot is able to see obstacles from 2 to approx. 250mm, depending on the calibration, the ambient conditions and the obstacle colour.
10. **Ultrasonic sensors:** There are 5 ultrasonic sensors, having an angle of 45° between each pair, that enables it to see obstacles from approx. 250 to approx. 2500mm.
11. **Camera:** The robot has a colour camera in front with user-changeable lens It can be used to take pictures or videos that can be further processed.
12. **Bottom infrared sensors:** Four infrared sensors are provided on the bottom of the robot which prevents the robot from falling down and can also be used to follow a line.
13. **Contacts for docking station:** The Khepera IV has some apparent contacts below its body i.e. it can be used as a docking station to charge its battery or to communicate with it. The signals that are provided are: Battery out (controlled by a reed relay), 9V in, 12C.
14. **Wheels:** The robot is differential driven, with 2 wheels equipped with O-ring. The wheels are driven by DC motors with encoder and gearbox.
15. **Sticker:** It provides the serial number of the robot.

16. **Bottom M3 Nuts:** There is the possibility to fix an extension to the robot from below.
17. **KB-250 Extension connectors:** These two connectors are used to connect extension modules to the robot.
18. **Top M3 Nuts:** There is the possibility to fix an extension to the robot from above.
19. **Magnets:** There are three magnets used mainly to attach the gripper, but they can be used for any other module.
20. **RGB LED:** There are three RGB LED on the top of the robot, all are user-controllable. They are primarily intended for robot pattern recognition with a colour camera mounted on the ceiling of the experiment room.

## 2.4.2 Detailed Functionalities

### I. Infrared Sensors



**Figure 2.4 Infrared Sensors viewed from Bottom.**

Khepera IV has 8 infrared sensors placed all around the robot and 4 placed on the bottom. They are positioned and numbered as shown in figure 2.4.

These sensors embed an infrared light emitter and a receiver. The twelve sensors are TCRT5000 reflective optical sensors from Vishay Telefunken. Measuring range is from 2 to 250mm. Each sensor is separated from its neighbour from an angle of 45°.

This kind of sensors allows two measures:

- The normal ambient light. This measure is made using only the receiver part of the device, without emitting light with the emitter. A new measurement is made every 5ms. The value returned at a given time is the result of the last measurement made.
- The light reflected by obstacles (=proximity). This measure is made by emitting light using the emitter part of the device. The returned value is the difference between the measurement made while emitting light and the light measured without light emission (ambient light). A new measurement is made every 5ms. The value returned at a given time is the result of the last measurement made.

## II. Ambient light measurement

Ambient light measurement is strongly influenced by the robot's environment. Depending on the light source type, colour, and distance, ambient light measurement profile might vary. It is not recommended to use light source with large emission in the infrared range, as this could confuse the IR sensors. Value range is 0 to 1023, 0 stands for no light and 1023 for full light.

## III. Reflected light measurements (proximity)

Sensors are mainly meant to detect obstacles around the Khepera. Measurements for reflected light depend on objects reflectivity and on ambient light conditions. Object colours, materials and surfaces do have an influence on the sensor's response. Moreover, as any sensor, IR sensors are subject to environmental noise. For all these reasons, graphics below are given for information only and should not be considered as references, Value range is 0 to 1023, 0 stands for no obstacle, 1023 for very near obstacle. Here's an example of value with a white paper used as an obstacle:

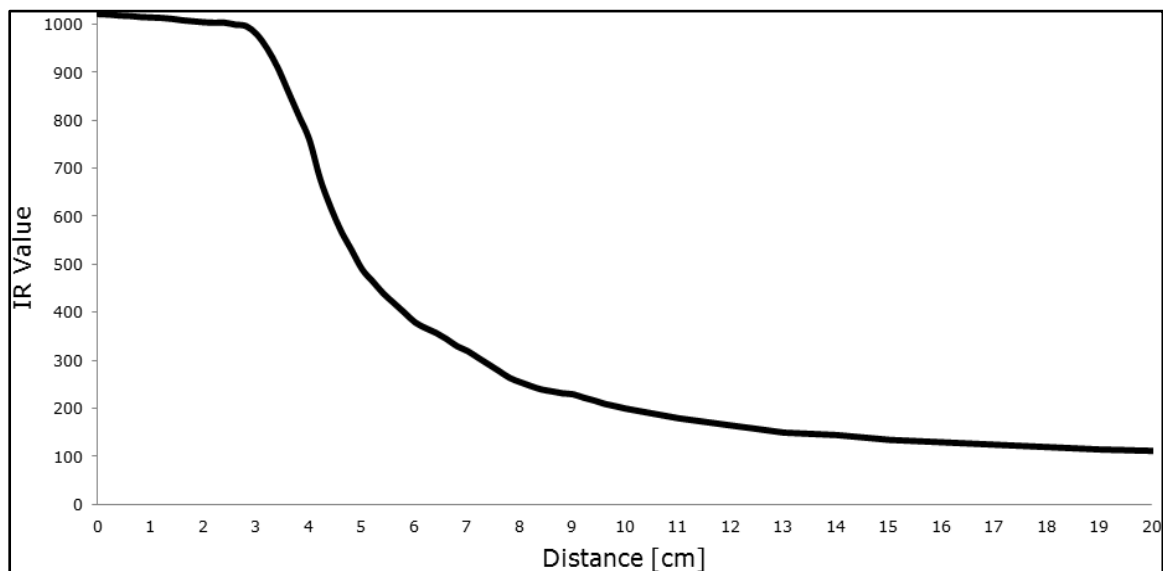
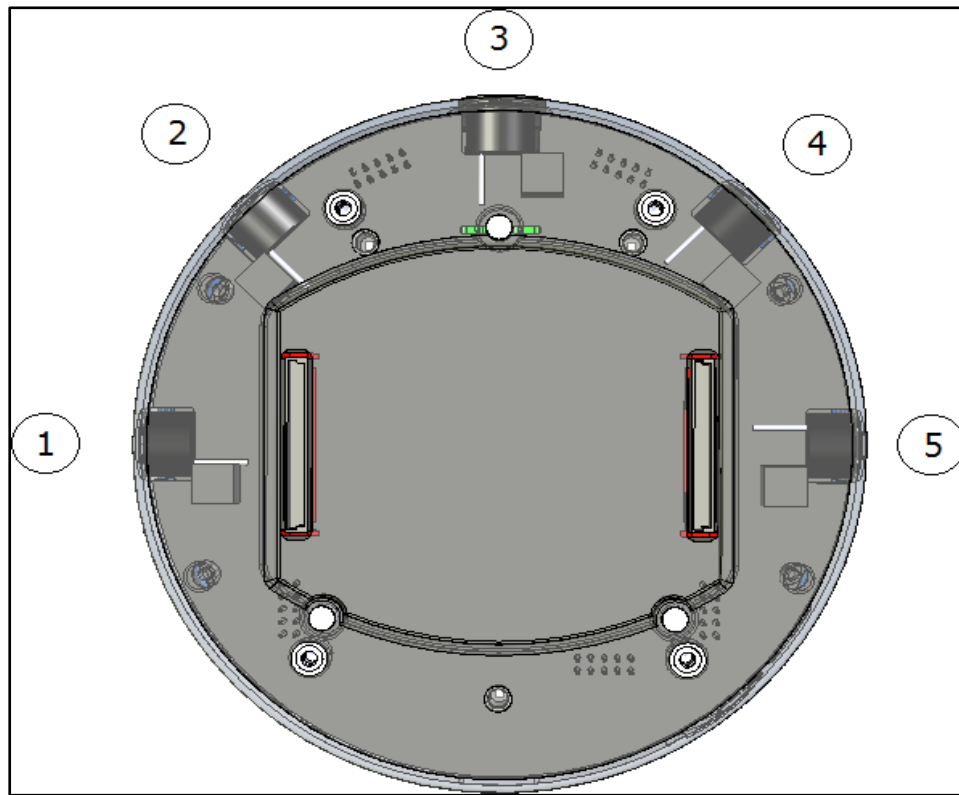


Figure 2.5 IR value vs Distance

The IR value never falls to 0, as even with no obstacle, the IR reflects on the floor and adds a static value. As all the sensors are not exactly the same, the solution is to perform a calibration of the IR with no obstacle in front. With this calibration, the user will be able to improve detection of obstacle at distance greater than 20cm.

#### IV. Ultrasonic sensors

Five sensors are placed around the robot and are positioned and numbered as shown in figure below:



**Figure 2.6 Ultrasonic sensors viewed from top.**

These sensors are transceivers, meaning that they can emit and receive the pulses. The ultrasonic Sensors are powered by an 85 Vpp source. The nominal frequency of these transducers is 40kHz +/- 1 kHz. The returned value is the distance to the object in centimetres, with a tolerance of +/-2cm. Measuring range is from 25 to 200cm. Every transducer is separated from its neighbour from an angle of 45°. Each sensor can be disabled in order to get higher refresh rate for a particular one (or group). One sensor measure takes 20ms All 5 sensors need 100ms to be read.

## V. Battery

The Khepera IV is equipped with an internal non-removeable Lithium-Polymer battery. It is built in a 2S1P configuration, giving 7.4V nominal, 8.4V charging voltage and a capacity of 3400mAh.

**Table 2.2 Battery Specification**

Nominal voltage	7.4V
Cut-off voltage	6.0V
Charging voltage	8.4V
Nominal capacity	3400mAh
Max discharge current	3400mA (1C)
Charging current	1100mA
Time for a complete charge	about 4 to 5 hours

Using its embedded power, the robot is able to run completely autonomously for more than 5 hours with motors at 100% and 7 hours with motors off, running with a basic configuration. When additional equipment is used, the autonomy is reduced as Khepera's extensions like the gripper rely on Khepera's batteries as a power source. There is no specific power management system on the Khepera. When the battery voltage falls under 6V, the battery opens itself the circuit to avoid a deep discharge of the cells. Users can implement their own software power management system to handle extensions to shutdown properly before this case happens.

The battery can be charged from 3 places:

- From the jack
- From the contacts situated below the robot
- From the KB-250 extensions connectors.

The battery is charged through an internal battery charge IC that needs 9V of input voltage. During its constant current phase, the battery is charged with a 1.1A current. To charge via the contact pads situated below the robot, only use the 9V in and Ground pads. To charge via the KB250 extensions connectors, use only pin 44 of 1701 and a ground pin. Max voltage is 9.5VDC the charge status is indicated on the charging status LED. During the charge, the red LED turns on. The red LED will turn off when the charge is complete. The charge will not be enabled if the external supply is plugged when the battery is above 7.95V. In this case, the red LED will never turn on, neither the green LED. If the robot is turned on, the dsPIC will check the end-of-charge status in the Fuel Gauge. When this status is set (~5-10 second after the red LED turns off), the charge status LED will turn green.

The Fuel Gauge (DS2781) returns different information available by the application:

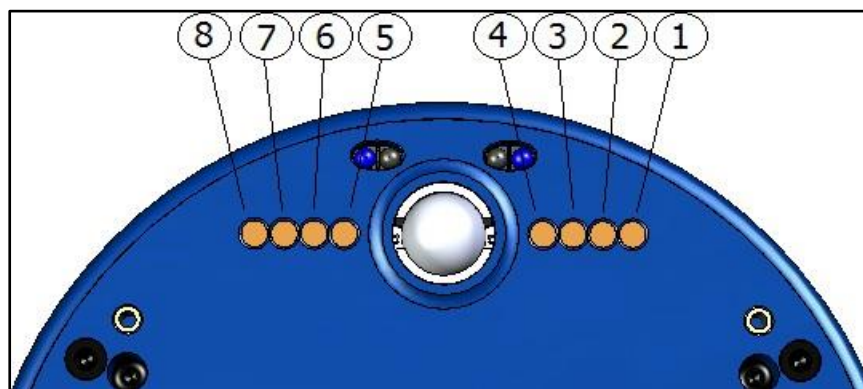
- Battery status register
- Absolute remaining capacity (unit 1.6mAh)
- Relative remaining capacity (0-100%)
- Battery current (updated every 3.5s). The resolution is 78.125[uA]. A positive value means a charging current
- Average current (updated every 28s)

- Temperature with a resolution of 0.125°C.
- Battery voltage with a resolution of 9.76mV

The battery current is measured through a 20[mΩ]resistor. The battery temperature is accurate only when the charge is not active. During the charge, internal component heat will perturb the measure. It is required to consider that the returned value is approximately 10°C higher than the real battery temperature (especially when charging current is 1A). As the charge is not automatically protected against temperature, it is needed to verify before starting a charge.

## VI. Contact pads

Eight contact pads are situated below the robot and can be used to connect either to an extension module or to a docking station.



**Figure 2.7 Contact pads viewed from bottom**

**Table 2.3 Pin out of the contact pads**

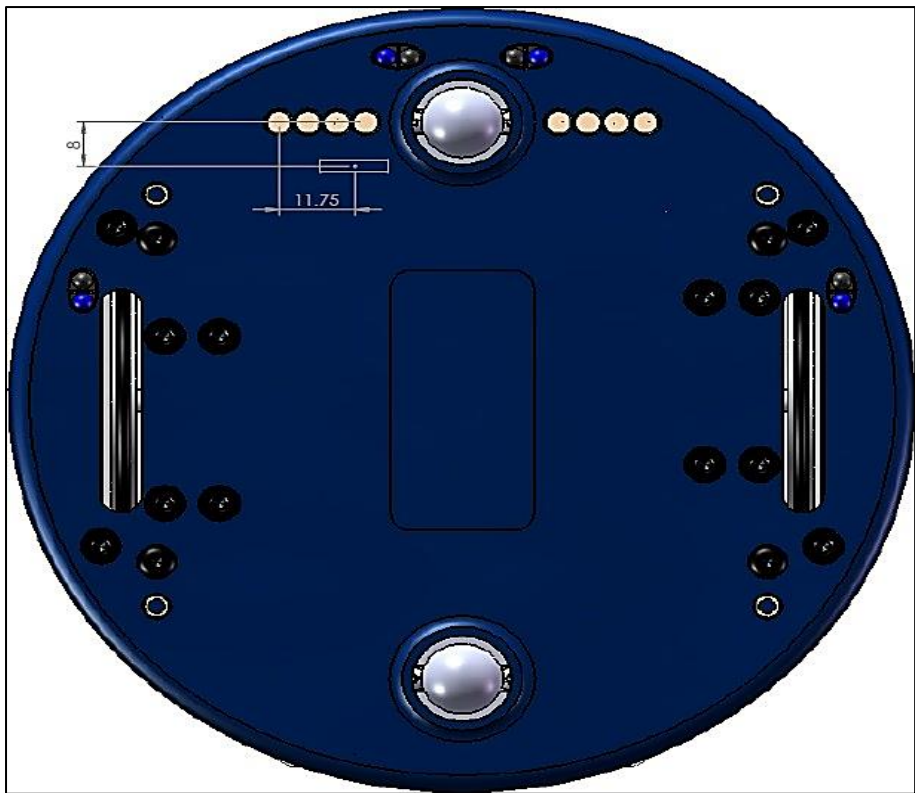
<i>Pos.</i>	<i>Signal</i>
1	9V out
2	9V in
3	I <sup>2</sup> C Serial clock
4	Ground
5	Ground
6	I <sup>2</sup> C Serial data
7	9V in
8	9V out

The internal battery can be used to power external devices. Signal is "9V out", present on pads number 1 and 8. As this is a power output, this signal is controlled by a reed relay to avoid any short-circuit. In order to be able to draw current from the battery, it is first required to activate the relay with a magnet. Please refer to the drawing for the exact location of the relay. Max current is 1.5A but this value depends on operating current of the robot. Voltage is accordingly to the battery state minus voltage drop of one Schottky diode and the Rdson of two MOS-P transistors. The internal battery can be charged from a docking station. Signal is "9V in", present on pads number 2 and 7. Communication with the outside is done via I2C. Clock signal is on pad 3, while data signal is on pad 6. Level is 0 to 3.3V. Use level adapter if needed.

Ground signal is present on pins 4 and 5. The signals are routed in a symmetrical way so that if the robot is in the wrong direction on the docking, no damage occurs.

### VII. Reed relay location

The diagram below represents the location of the reed relay. Unit is [mm]. The coordinates are about the centre of the relay.



**Figure 2.8 Reed relay location**

### VIII. Camera

The Khepera IV is equipped with a front color camera, disposed below the front ultrasonic sensor. The sensor is a MT9V034C12ST from Aptina. It's a 1/3" WVGA CMOS sensor.

**Table 2.4 Camera Specification**

Active imager size	4.51x2.88mm
Active pixels	752x480

The default lens has a focal length of 2.1mm, with IR cut filter and fixed focus. The mounting thread is M12x0.5. Diagonal field of view is 150°, horizontal is 131° and vertical is 101°.

### IX. Microphones

The Khepera IV is equipped with an amplified Microphone PU0414HRSH-SB from Knowles. It's directly connected to the Overo Analog right SUB MIC input.



**Table 2.5 Specifications of Microphones**

Gain	20dB
Sensitivity (typ)	-22dbv/Pa
Directivity	Omnidirectional
Supply voltage	2.5V

## **X. Loudspeaker**

A SMS-1308MS-2-R loudspeaker from PUI Audio is mounted on the Khepera IV. This speaker is driven by a 1 W low distortion power amplifier. The speaker is connected on the HSOLF output of the OVERO. The OVERO can also mute the amplifier with GPIO64 (0 = MUTE, 1 = ampli on).

**Table 2.6 Specifications of Loudspeaker**

Speaker Power	0.7W (max 1W)
Impedance	8 Ohm
Output SPL	88dBA
Distortion (max)	5%
Resonant frequency	850Hz $\pm$ 20%
Frequency range	400 ~ 20'000Hz

## **XI. Gumstix Overo FireSTORM-P COM**

The Khepera IV was designed to embed a Gumstix Overo processor board. The computer-on module mounted by default in the Khepera IV is the Gumstix Overo FireSTORM-P COM. This computer-on-module has an additional DSP to perform special tasks, Bluetooth & WiFi capabilities (SMD antenna mounted on the Khepera).

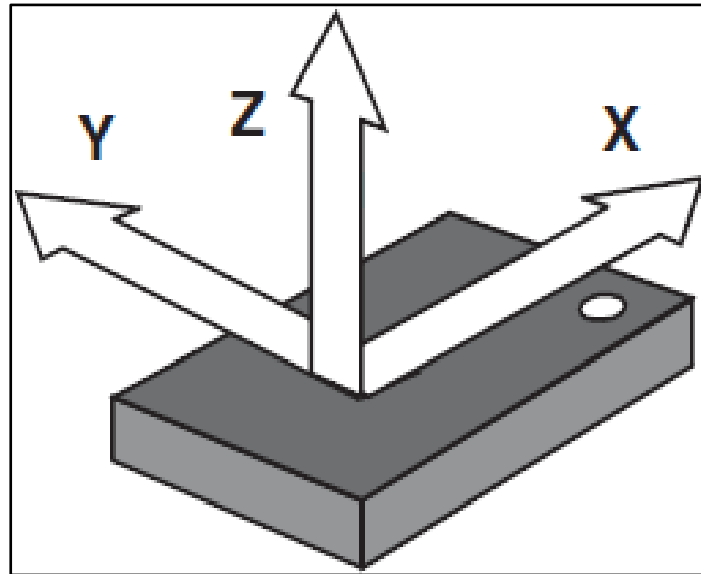
**Table 2.7 Specifications of Gumstix Overo FireSTORM-P COM**

Architecture	ARM Cortex-A8
NAND Flash	512MB
Processor	Texas Instruments OMAP3730 @800 MHz
DSP	C64x Fixed Point DSP 660, 800MHz
Wifi	802.11b/g/n included
Bluetooth	version 3.0 included

The Gumstix is provided with a Linux system already installed (Angström distribution).

## XII. Accelerometer

The accelerometer mounted on the Khepera IV is a LSM330DLC from ST. This device includes in one package a 3D accelerometer and a 3D gyroscope. The device is exactly at the centre of the robot (placed on the rotation centre). The device is located on the TOP of the main PCB. The accelerometer is oriented with the pin 1 to the front right; this returns a positive value for X axis gravity. when going forward. The Y axis is positive on the left, and finally Z axis is negative with the gravity.

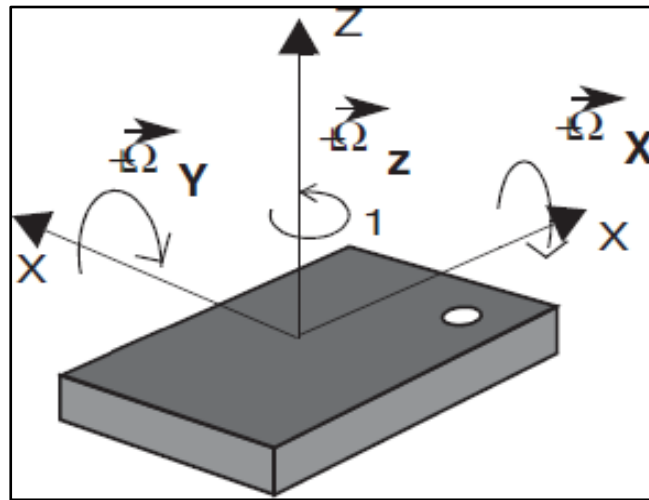


**Figure 2.9 Directions of Detectable Accelerations (TOP view)**

The accelerometer returns 12-bit data (two's complement) with a range of  $\pm 2g$ . This means a value of  $1g$  will return a value of  $16384$ . The data rate is configured to  $100Hz$ , as the dsPIC of the Khepera refreshes 10 values at a time, the user needs to read every  $100ms$  ( $10Hz$ ) to obtain fresh data.

## XII. Gyroscope

The gyroscope of the Khepera IV is included in the same package as the accelerometer. The directions of detectable angular rates are defined around the accelerometer axis.



**Figure 2.10 Directions of Detectable Angular Rates (TOP view)**

The data format is on 12 bits too, the full range is configured at  $\pm 2000\text{dps}$  ( $360\text{dps} = 5'898$ ) and the data rate is configured at  $95\text{Hz}$ . The gyroscope data is read by packets of 10 values at a time, which means the user can read new data every  $105\text{ms}$  to obtain fresh value. The output has a to be multiplied by  $0.066$  to have  $(\text{deg/s})$  units.

## XIV. USB Device (mini-USB B connector)

A mini-USB B connector provides access to a USB-to-serial adapter (FT234XD from FTDI) to access directly to the `ttyS2` of the Gumstix. Using a terminal provides the access to the boot of the system. When connecting a computer to this connector for the first time, the local system will ask for driver. FT234XD driver can be found at <http://www.ftdichip.com/Products/ICs/FT234XD.html>.

## XV. MicroSD

A 4GB Micro SD card is provided inside the Khepera IV. The robot will boot on it and use this card. It already contains the OS, kernel and boot files.

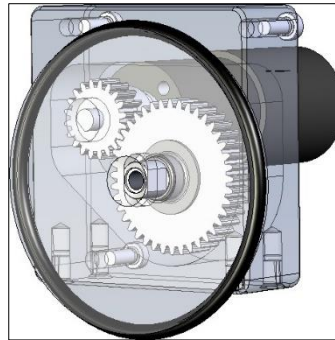
## XVI. RGB LED

Three RGB LED (19-337/R6GHBHC-A01/2T from Everlight) are mounted on the TOP of the main board. Each of these LED has a light guide on it. The LED are driven by a dedicated LED driver (LTC3219 from Linear) which provides a resolution of 6 Bits (0-63) for each colour.

These LED can be used to locate the Khepera IV with a camera (at the ceiling) and differentiate each robot (in swarm application). As the LED are placed on an isosceles triangle, the direction of each robot can also be detected.

## XVII. Motors

The Khepera IV has got 2 DC motors in order to drive its two wheels. The motors have 1.96W nominal power. The integrated gearbox has a reduction ratio of 19:1 and an efficiency of 78%. There is another gearbox within the carter of the motor block, with a ratio of 2:1 and efficiency of 85%. Total ratio is then 38:1 and efficiency is 66.3%, meaning that there is 1.3W of a usable mechanical power by wheel.

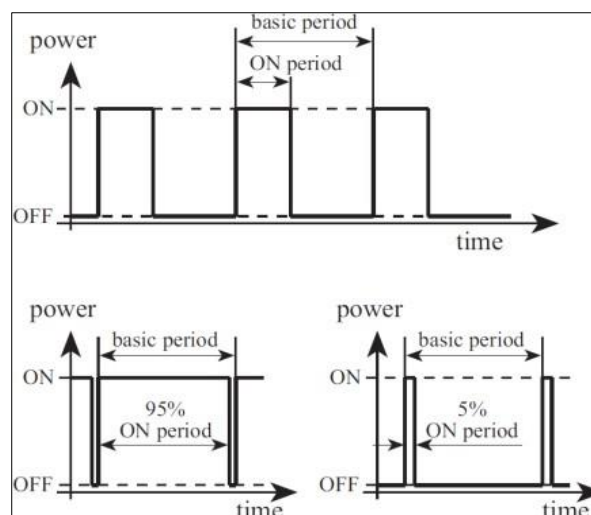


**Figure 2.11 Motor Block with Wheel**

The encoder has a 128-pulse by turn resolution. With the reduction ratio of 38:1 and an internal hardware 4x multiplier, there are 19'456 pulses by wheel turn. As the diameter of the wheel is 42mm (perimeter is then 131.94mm), this gives 147.4 pulses by millimeter. Or 1 pulse is 0.006782mm (6.7818um).

*Reminder: 1 revolution = 131.94mm = 19456 pulses.*

Both motors are controlled via Pulse Width Modulation (PWM) at 20kHz. This technique switches the motor ON and OFF at a given frequency and during a given time by this way, the motor reacts to the average of the power supply, which can be modified by changing the period the motor is switched ON. This means that only the ratio between ON and OFF periods is modified, as illustrated in the figure below:



**Figure 2.12 Duty Cycle with PWM**

The dsPIC calculates the PWM to apply to each motor in speed control and position control. The user can override the PID and apply directly a desired PWM to the motor using the open loop command. Default PID settings applied to the Khepera IV controller are:

$$K_p: 10$$

$$K_i: 5$$

$$K_d: 1$$

These values will be used by the PID speed controller. In position control, the PID is the same as the position controller calculates a speed order then calls the speed controller to reach this order. User can modify these values to improve behaviour to his particular use. When selecting a type of control, this mode will be applied to both motors. It's not possible to set the left motor in speed control and the right motor in another mode. To put the motor in idle mode (no more current drawn by the motors), use the open loop control with parameters set to 0. In speed control, even with a parameter of 0, the controller will struggle against any movement.

### XVIII. Speed control

Both DC motors are controlled by a PID controller executed every 10ms in an interrupt routine of the dsPIC. Every term of this controller (Proportional, Integral, Derivative) is associated to a constant, setting the weight of the corresponding term:  $K_p$  for the proportional,  $K_i$  for the integral,  $K_d$  for the derivative. The controller has as input the speed value of the wheels and controls the motor to keep this wheel speed. The speed modification is made as quick as possible, in an abrupt way. No limitation in acceleration is considered in this mode. The speed unit corresponds to the difference measured in position between the two controllers' routine (10ms). Here's the formula to convert the speed unit to metric unit.

<i>Refresh time:</i>	10ms
<i>Wheel diameter:</i>	42mm
<i>Revolution resolution:</i>	19'456 [pulses]
$Speed[mm/s] = \frac{v_{pulses} \cdot \phi_{Wheel} \cdot \pi}{t_{Refresh} \cdot Nb_{pulses}} = \frac{v_{pulses} \cdot 42 \cdot \pi}{0.01 \cdot 19456} = 0.678181 \cdot v_{pulses}$	

The minimum speed order to ensure a correct control is 5 (=3mm/s). Under this value, the control is not very stable with default PID parameters. User can modify the PID to try to improve the behaviour for this kind of very low speed. The maximum speed order is approximately 1'200 (=813mm/s). It's still possible to move the robot faster if the control mode is set to open loop. In this case, the maximum speed will vary with the battery voltage and the payload.

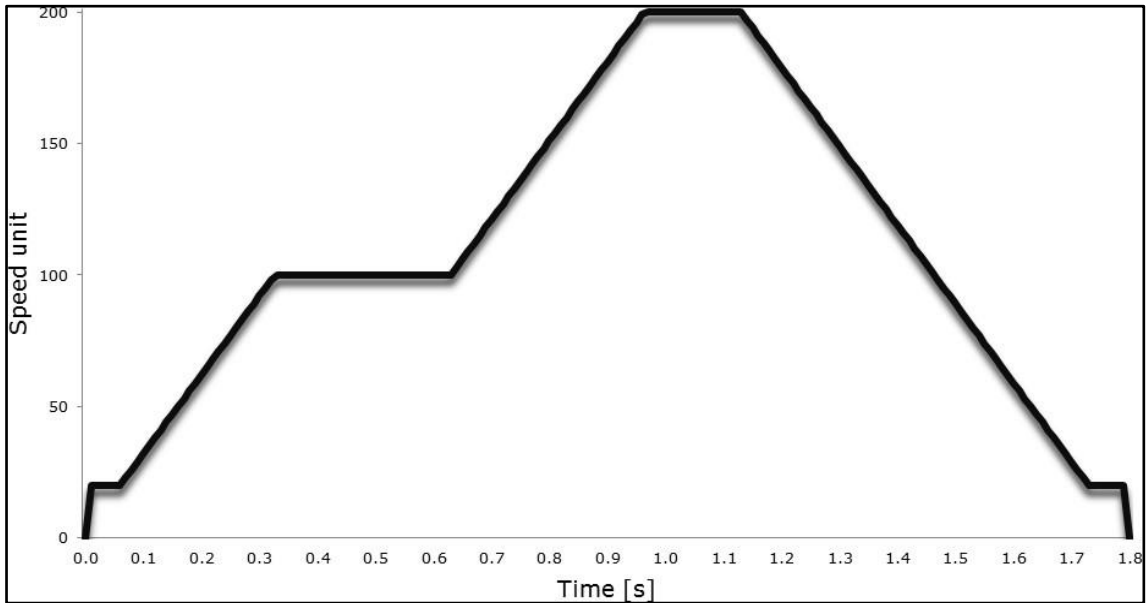
## **XIX. Speed profile control**

This type of control uses the same PID as standard speed controller but adds an acceleration ramp to travel from the actual speed to the new speed order. The ramp used in this mode can be configured with the speed profile parameters. Three parameters define the ramp:

**Table 2.8 Parameters for defining the Ramp**

Acc_Inc	value of increment to add or subtract every Acc_Div +1 control loop (value from 1 to 255). Default = 3
Acc_Div	defines the number of control loops where no increment is added to the speed order. For example, a value of 0 means that at every control loop, the speed will be increased by Acc_Inc. A value of 4, means that every 5 control loops (50ms) the speed will be modified. (value from 0 to 255) Default = 0.
Min_Speed_Acc	this parameter defines the minimum speed used by the controller. This value avoids setting a speed too low where the controller is not efficient. If the order value is smaller than this parameter, the controller will automatically limit the speed to Min_Speed_Acc. Do not set values lower than 1. Default = 20.

Here's an example of speed profile with default parameters (Acc\_Inc = 3, Acc\_Div = 0, Min\_Speed = 20).

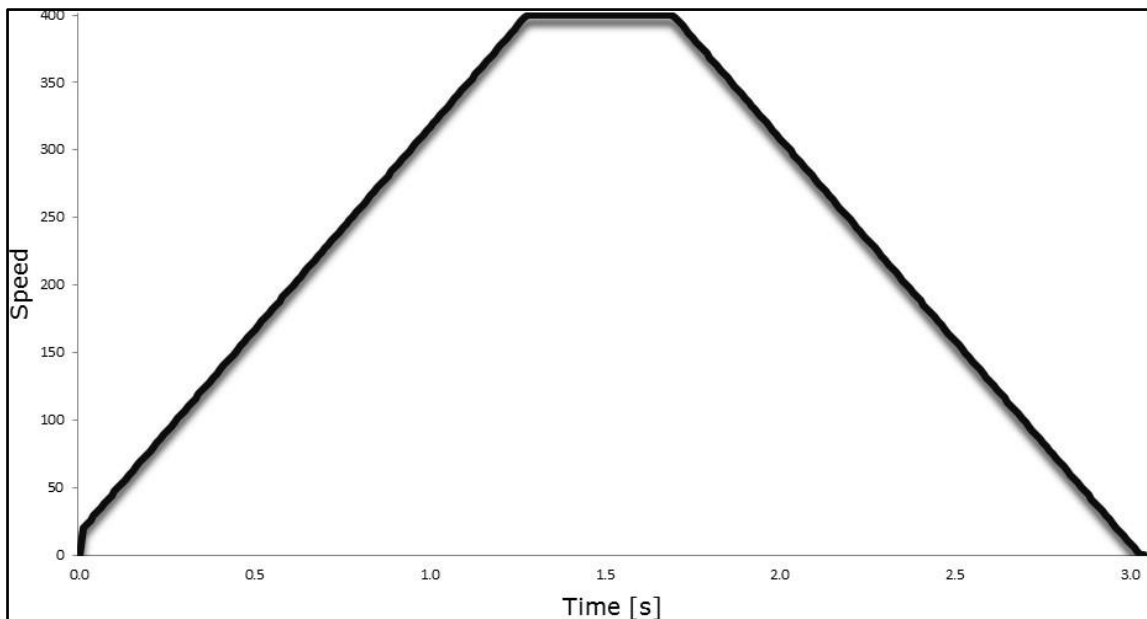


**Figure 2.13 Example of Speed Profile**

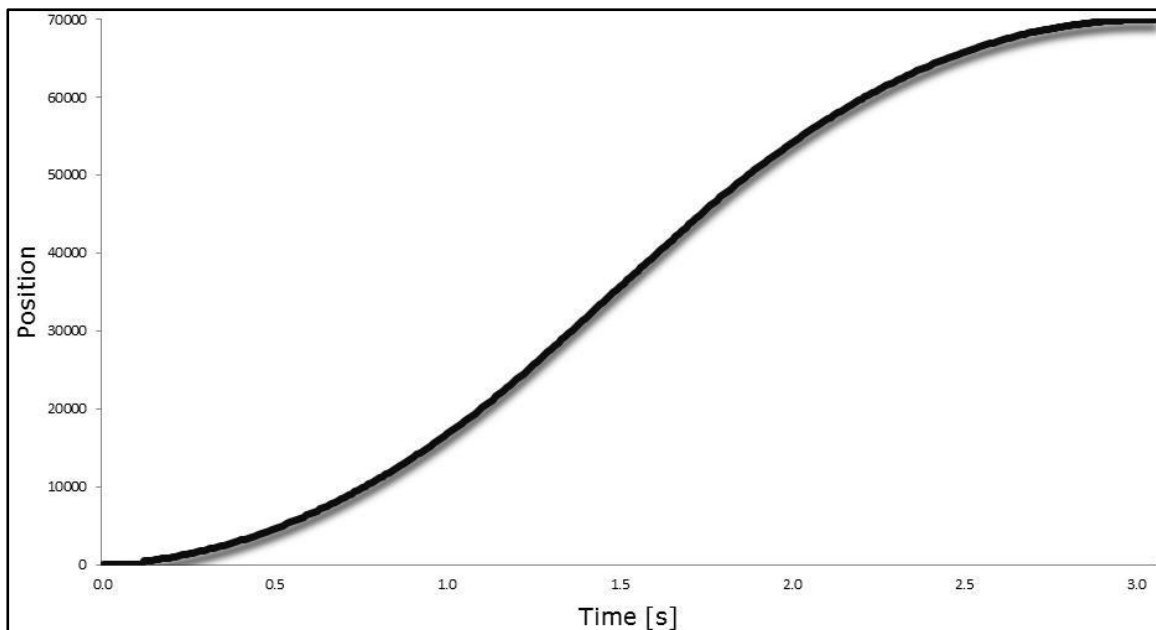
A speed profile order has been set to 100. After 300ms at constant speed, a new speed order of 200 is set. The motor keeps this speed during 200ms, and finally decreases until reaching 0. This curve corresponds to the order sent to the PID speed controller. The real speed of the motor will depend on the payload and the PID reactivity. A higher Acc\_Div parameter will increase the time between two steps to allow the PID to reach the speed order. A value of 0 means that the effective motor speed will always be late on the order during the acceleration. This type of control must be preferred to the simple speed profile in order to avoid high current peaks and preserve the mechanical parts. The user needs to adapt the parameters Acc\_Inc and Acc\_Div to match the desired behaviour (high Acc\_Inc for a reactive profile, high Acc\_Div for a smooth profile).

## XX. Position control

In this mode, the robot will calculate a speed order (which will be processed by the PID) to move the robot using an acceleration ramp, a constant speed, and finally a deceleration ramp.



**Figure 2.14** Speed profile using position control



**Figure 2.15** Position Profile

This example shows a travel of 475mm (70'000 pulses) using the default parameters.

The position control mode when the same parameters as the speed profile control to calculate the acceleration ramp. The `Min_Speed_Acc` parameter is used only at the start. When reaching the target position, the speed is limited by `Min_Speed_Dec` parameter (default = 1). In addition to these three parameters, the travel speed can be configured through "Speed Order" parameter (default = 400). Finally, the "Pos\_Margin" parameter defines the threshold when the position



controller stops completely the motor (set 0 to the speed controller). A low margin will increase the precision, but will add an instability to the control. It is not recommended to set this parameter below the default value (10). To calculate the real distance travelled by the motor, use the formula below:

<i>Wheel diameter:</i>	42mm
<i>Revolution resolution:</i>	19'456 [pulses]
$Position[mm] = P_{pulses} \cdot \frac{\phi_{Wheel} \cdot \pi}{Nb_{pulses}} = P_{pulses} \cdot \frac{42 \cdot \pi}{19456} = \frac{P_{pulses}}{147.453}$	

The position is stored in a signed 32 bits data, which means that the maximum position order is +/- 2<sup>31</sup> pulses (=14'563m), When performing straight travel (name distance on each wheel), the best solution is to reset the position encoder before sending the target position command.

## XXI. Open loop

This control mode disables the PID controller and sets directly the PWM to the two motors. This can be useful if the application wants to calculate its own PID. The range of this command is +/-2'940 where 2'940 correspond to 100% of PWM in forward direction and -2'940 in backward direction. If the application wants to disable the motor (to decrease current consumption), the best way is to use this mode and set the PWM to 0. The motor will be in free wheel mode.

## 2.5 Safety Precautions

Here are some recommendations on how to correctly use the robot:

- Keep the robot away from wet area. Contact with water could cause malfunction and/or breakdown.
- Store your robot in a stable position. This will avoid the risk of falls, which could break it or cause damage to a person.
- Use only the official charger or the cable which is delivered with the robot. Do not try to use another charger; this can cause irreversible damage to the battery and or the electronics.
- Do not attach any connector while the robot is powered. To avoid any damage, make all connections when the robot power is off.
- Never leave the robot powered when it is unused. When you have finished working with Khepera, turn it off. It will save the battery life.
- Do not manually force any mechanical movement. Avoid forcing by any mechanical way, the movement of the wheels or any other part.
- Never open the case. Only qualified technicians are allowed to do so.

## 2.6 Conclusion

This chapter discusses the components of Khepera IV robot along with its detailed functionalities. It also provides the mechanical drawings of the robot that describe the various precautions are listed at the end of this chapter dimensions of its components which can be utilized for performing any experiment. The safety precautions are listed at the end of this chapter.

## References

- [1] J. M. Soares, A. P. Aguiar, A. M. Pascoal, A. Martinoli, "A distributed formation-based localization algorithm: design implementation and wind tunnel evaluation", *IEEE Int. Conf. on Robotics and Automation*, pp. 1830-1836, 2015.
- [2] E. Peralta, F. Fabregas, G. Farias, H. Vargas, S. Dormido, "Development of a khepera iv library for the V-REP simulator", *11th IFAC Symposium on Advances in Control Education*, June 2016.
- [3] [www.k-team.com/mobile-robotics-products/khepera-iv/manuals-downloads](http://www.k-team.com/mobile-robotics-products/khepera-iv/manuals-downloads)
- [4] <https://www.k-team.com/>
- [5] Y. Hu, S. X. Yang, "A knowledge based genetic algorithm for path Planning of a mobile robot". *Proc. IEEE Int. Conf. Robotics & Automation*, April 2004.
- [6] Y. Q. Qin, D. B. Sun, N. Li, Y. G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator", *Proc. 3th Int. Conf Machine Learning and Cybernetics*, pp. 2473-2478, 2004.
- [7] L. Lu, D. Gong, "Robot path planning in unknown environments using particle swarm optimization", *4th IEEE Int. Conf. Natural Computation*, pp. 422-426, 2008.
- [8] Reina, J. Gozalez, "Characterization of a radial laser scanner for mobile robotic navigation", *Proc. IEEE/RSJ Int. Conf. on Intelligent robots and systems*, pp. 579-585, 1997.
- [9] W. Feiten, R. Bauer, G. Lawitzky, "Robust Obstacle Avoidance in Unknown and Cramped Environments", *Proc. IEEE Intl. Conference on Robotics and Automation*, pp. 2412-2417, 1994.
- [10] J. MacCormick, A. Blake, "A probabilistic exclusion principle for tracking multiple objects", *Proc. of 7th International Conference on Computer Vision (ICCV)*, pp. 572-587, 1999.
- [11] H. Murase, S. Nayar, "Visual learning and recognition of 3-d objects from appearance", *Intl. J. Computer Vision*, vol. 14, pp. 5-24, 1995.
- [12] C. Papageorgiou, T. Poggio, "A trainable system for object detection", *Intl. J Computer Vision*, vol. 38, no. 1, pp. 15-33, 2000.
- [13] F. Mondada, E. Franzi, A. Guignard, "The Development of Khepera", *Proceedings of the 1st International Khepera Workshop*, vol. 64, pp. 7-13, 1999.
- [14] [https://www.researchgate.net/publication/37438219\\_The\\_Development\\_of\\_Khepera](https://www.researchgate.net/publication/37438219_The_Development_of_Khepera)
- [15] <http://www.egyptianmyths.net/khepera.htm>
- [16] [https://en.wikipedia.org/wiki/Khepera\\_mobile\\_robot](https://en.wikipedia.org/wiki/Khepera_mobile_robot)

# Chapter 3

## SET UP & PROGRAMMING

### 3.1 Introduction

The setup step is a very important aspect which includes the organization of the necessary elements required for programming and controlling the robot, though the real time application is not done but to implement some part of the thesis this setup is necessary. There are mainly two development packages used for this process- Light tool chain and Full tool chain [1-2] Moreover, there are important components that are extremely necessary for the setup process which are:

- a) **Cross compiler** which is installed in the computer of the user
- b) **Khepera IV** robot where the code will be executed.
- c) **Establishment of communication channel** (wired or wireless) through which communication between the computer and robot will take place.

#### 3.1.1 Required Hardware

1. Computer with USB and WiFi consisting of Linux operating system (Kernel 2.6.x and higher) with at least 3 GB on /opt account and at least 50 MB on user account.
2. Khepera IV robot.

#### 3.1.2 Required Software

1. Linux OS with packages like GCC (GNU C/C+ Compiler), MINICOM (Terminal Emulation), LRZSZ (Communication Package) should be installed.
2. The following two files must be installed either from the Khepera IV DVD or K-Team [3] website:
  - i. Cross-compiler Light toolchain: *poky-glibc-i686-khepera4-image-cortexa8hf-vfp-neontoolchain- 1.sh*
  - ii. Robot library sources: *libkhepera-2 1.bz2*

#### 3.1.3 Development Directory

The development directory is the base folder for Khepera IV development comprising with scripts, links, make files which are very much essential for compile the program in cross compiler. The creation of new development directory Khepera4 is by the following command:

```
~$ mkdir Khepera4
```

```
~$ cd Khepera4/
```

*mkdir folder\_name* is used to make directory while *cd folder\_name* is used to change the directory to *folder\_name* directory.

### 3.1.4 Installation of Cross-Compiler (Light Tool chain)

The steps and snapshot for light tool chain installation are illustrated below.

1. Cross compiler is first installed in `/opt` folder. However, it can be installed in any other folder.
2. For this process, the execution permission of the `.sh` file has to be changed. This is done by the command `chmod +x`.
3. `.sh` file is a shell script used for sequential execution of linux commands hence, it is interpreted by shell (interpreter of terminal commands). After changing the permission of `poky-glibc-1686-khepera4-image-cortexa8hf-vfp-neon-toolchain-1.sh` file, this file is required to be executed.
4. Checking of the correct installation of cross compiler is performed.
5. The executed file is required to be **sourced** in the folder where it has been executed (`/opt/poky/1.8` for our case). **Source** is used to read and execute a command from a file and make environment variables available.
6. After executing the above steps, the version of cross compiler can be found (which is GCC 4.9.2 for our case)

```
khepera@khepera-desktop:~/Khepera/software/light_tools$ ls
poky-glibc-1686-khepera4-image-cortexa8hf-vfp-neon-toolchain-1.sh
khepera@khepera-desktop:~/Khepera/software/light_tools$ chmod +x poky-glibc-1686
-khepera4-image-cortexa8hf-vfp-neon-toolchain-1.sh
khepera@khepera-desktop:~/Khepera/software/light_tools$ sudo ./poky-glibc-1686-k
hepera4-image-cortexa8hf-vfp-neon-toolchain-1.sh
[sudo] password for khepera:
Enter target directory for SDK (default: /opt/poky/1.8):
The directory "/opt/poky/1.8" already contains a SDK for this architecture.
If you continue, existing files will be overwritten! Proceed[y/N]?y
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
khepera@khepera-desktop:~/Khepera/software/light_tools$ source /opt/poky/1.8/env
ironment-setup-cortexa8hf-vfp-neon-poky-linux-gnueabi
khepera@khepera-desktop:~/Khepera/software/light_tools$ arm-poky-linux-gnueabi-g
cc --version
arm-poky-linux-gnueabi-gcc (GCC) 4.9.2
Copyright (C) 2014 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Figure 3.1 Installation of Light Tool Chain

### 3.1.5. Installation of Robot Library

All the inbuilt functions and commands are within this library. This library is already installed inside the robot. However, the same prototype library is required to be installed in the PC that will be used for compilation. The following steps are required to install the library on the development system:

1. Extract the robot library `libkhepera-2.1.bz2` in the development folder (**Khepera4** for my case).
2. After getting into the extracted folder whole library need to be compiled using the make file command Make file is used for simplifying the building program executables that may need various modules. It is a user-defined file. Make file is also used to make utility while compiling and linking programs modules. It is a special format file that helps to do many tasks

automatically depending upon the need. It is not mandatory to execute the commands line by line as of `.sh` file used to do. Here when **make clean** command will be executed then it will execute some specific commands which are not as same as the **make all** command execution.

```
root@khepera-desktop:~/Khepera/software/library# ls
libkhepera-2.1.bz2
root@khepera-desktop:~/Khepera/software/library# tar -xjf libkhepera-2.1.bz2 -C
~/Khepera4
khepera@khepera-desktop:~/Khepera4$ ls
libkhepera-2.1
khepera@khepera-desktop:~/Khepera4$ cd libkhepera-2.1/
khepera@khepera-desktop:~/Khepera4/libkhepera-2.1$ make clean
make[1]: Entering directory '/home/khepera/Khepera4/libkhepera-2.1/src'
Cleaning
make[2]: Entering directory '/home/khepera/Khepera4/libkhepera-2.1/src/tests'
Cleaning
make[2]: Leaving directory '/home/khepera/Khepera4/libkhepera-2.1/src/tests'
make[2]: Entering directory '/home/khepera/Khepera4/libkhepera-2.1/src/utils'
Cleaning
make[2]: Leaving directory '/home/khepera/Khepera4/libkhepera-2.1/src/utils'
make[1]: Leaving directory '/home/khepera/Khepera4/libkhepera-2.1/src'
khepera@khepera-desktop:~/Khepera4/libkhepera-2.1$ make all
```

**Figure 3.2 Execution of the make command**

### 3.2 Communication with Robot

Communication with the robot is a vital notion that aids in transferring the cross compiled executable code to the robot and interfacing with any hardware or software,

#### 3.2.1 Establishment of Communication Medium

The communication between the robot and the PC can be done through a wire medium (USB) [4-5] or wireless medium (Bluetooth, WiFi) [6-10]. Both types of communication establishment are discussed below:

##### A. Communication via USB

The setup of communication with the robot through USB involves the following steps:

1. Installation of the Linux package *lrzsz* following command:

```
~$ sudo apt-get install lrzsz
```

2. Installation of the Linux package *minicom* containing several commands to get into the robot and by utilizing the terminal to run the following command:

```
~$ sudo apt-get install minicom
```

3. Turning the robot on

4. Run minicom with the command: `~$ sudo minicom -s`. After this the **RETURN** key is Pressed.



## B. Communication via Wi-Fi

For real-time applications, Wi-Fi forms an important media for communication between the robot and the computer. Before setting the Wi-Fi of the robot, user's personal computer should have an existing Wi-Fi which can be able to create a network (hotspot), i.e. the ssid. Moreover, password should be known for that Wi-Fi network.

### 1. Wi-Fi configuration of robot: Edit the file

~/etc/wpa\_supplicant/wpa\_supplicant-wlan0.conf using vi command as given below.

~\$ vi etc/wpa\_supplicant/wpa\_supplicant-wlan0.conf

```
# Adjust this file for your network. See [1] for
# more details.
# [1] http://linux.die.net/man/5/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

#Connect to a WPA2 protected network

network={
    ssid="khepera123"
    proto=WPA2
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
#    group=CCMP TKIP
#    scan_ssid=1
    psk="khepera123456"
    priority=5
}
```

Figure 3.6 Display after Editing the file ~/etc/wpa\_supplicant/wpa\_supplicant-wlan0.conf

By default, the system uses the address given by the access point using the DHCP=v4 command

~\$ vi /etc/systemd/network/wifi.network

```
# Automatically bring-up default wifi interface as a wireless client.
# If possible, get an address using DHCP. This works in tandem with
# wpa_supplicant to establish a network connection.
# See http://www.freedesktop.org/software/systemd/man/systemd.network.html
[Match]
Name=wlan0

[Network]
DHCP=v4
```

Figure 3.7 Display showing the use of address given by Access point

After setting these files reboot the robot using **reboot** command.

2. **Checking of connectivity and IP address of robot:** To check the wlan IP Address of the robot use the command ~\$ **ifconfig**.



```
wlan0    Link encap:Ethernet HWaddr 78:A5:04:2D:C4:F9
         inet addr:10.42.0.176 Bcast:10.42.0.255 Mask:255.255.255.0
         inet6 addr: fe80::7aa5:4ff:fe2d:c4f9/64 Scope:Link
```

**Figure 3.8 Check for IP address of Robot**

To check the connectivity of the robot, ping the robot from the user's personal computer and also ping the computer from the robot using **ping** command.

3. **Remote access:** Robot can be controlled remotely using Wi-Fi. For that establish a connection between the robot and the PC using **ssh** command,

```
~$ ssh root@KHEPERA_IP
```

where KHEPERA\_IP is the robot IP address. After this command give the password to accept the authenticity.

4. **NFS Configuration:** Network File System (NFS) is used for sharing files and folders across an established network. It can be configured as a centralized storage and computation solution for different networks which need not require to be running on the same Operating System. It is secured with Firewalls. NFS mount needs at least two machines. The machine hosting the shared folder is called as server and which connects to it is called as the client. Here some of the folders of user's personal computer needs to be shared with the robot. For this case user's PC acts as a server and the robot acts as client. Before establishing this NFS configuration the IP address of robot and user's personal computer must be known in advance. The directory to be shared between the personal computer and the robot must be declared to the NFS service in the **/etc/exports** configuration file. To do this the following line should be added to the **/etc/exports** file on the computer using **vi** editor:

```
/home/khepera 10.42.0.176/255.255.255.0(rw no_root_squash sync)
```

**Figure 3.9 Line added to /etc/exports file**

where **/home/khepera** is the folder in the user's PC which will be shared with the robot. **10.42.0.176** is the IP Address and **255.255.255.0** is the Subnet Mask of the robot which acts as a client here. **'rw'** allows all the clients to read and write the files to the shared directory of the server. **'no\_root\_squash'** is used to tell all the root users that they are connected to that designated directory. **'sync'** will confirm the shared directory once the changes are committed. After adding these lines **nfs-server** on the server side (user's PC here) need to restart using the following command

```
~$ sudo service nfs-kernel-server restart
```

Next step is to mount the shared directory of the personal computer to the robot file system.

The following command is used to do this.

```
root@khepera4_1157: ~# mount 10.42.0.1:/home/khepera home/root
```

**Figure 3.10 Command for mounting the shared directory**



10.42.0.1 is the IP address of the server (user PC here) /home/khepera is the folder of the server that need to be mounted. /home/root is the folder of the client (robot here) where it will be mounted folder's properties using **df** command. It should look something like the following.

```

root@khepera4_1157:~# df
Filesystem            1K-blocks    Used Available Use% Mounted on
/dev/root              3615352    311136   3117236  10% /
devtmpfs              241344      0    241344   0% /dev
tmpfs                 249744      0    249744   0% /dev/shm
tmpfs                 249744      0    249744   0% /run
tmpfs                 249744      0    249744   0% /sys/fs/cgroup
tmpfs                 249744      0    249744   0% /tmp
tmpfs                 249744      4    249740   1% /var/volatile
tmpfs                 49952       0     49952   0% /run/user/0
10.42.0.1:/home/khepera 306045952 42779648 247697408 15% /home/root

```

Figure 3.11 Check of mounted folder properties

Now the directory is changed to /home/root directory to access the mounted folder.

### 3.2.2 Procedure for Transferring a File to the Robot

#### A. Minicom Console via USB

In the Minicom console, hold the keys "Ctrl + A" and then press "Z" It will show Minicom command summary of different functions. To send files press the keys and then select "Z-Modem" after that navigate with the arrow keys, 2x "spacebar" to change directory and 1x "spacebar" to select the file which need to be send to the robot. Then press the RETURN key successfully transferring the file to the robot it should show **Transfer Successful**.

#### B. File Transfer using Wi-Fi

After establishing the network connection between robot and user's personal computer via Wi-Fi execute the following command to transfer a file in user's personal computer to robot:

```
~$ scp FILE root@KHEPERA_IP:/home/root
```

where **FILE** is the file to transfer and **KHEPERA\_IP** is the robot's IP address.

```

khepera@khepera-desktop:~/Khepera4/libkhepera-2.1/template$ scp template root@10.42.0.176:/home/root
Password:
template                                                    100% 10KB 10.1KB/s 00:00

```

Figure 3.12 File transfer using scp

### 3.2.3 Procedure for Transferring a File to the Computer

FILENAME is the file you would like to send. In the Minicom console at the prompt of the robot, type the following command, where

```
~$ lsz FILENAME
```

The file **FILENAME** is sent to the last directory Minicom used.

### 3.2.4 Use of Camera Module

For using Camera module ensure that the driver of camera (mt9v032.so) is loaded. For taking snapshot **v412grab** program is required with some input parameters. For that following commands need to be executed inside the robot.

```
~$ media-pipes.sh
```

```
~$ media-formats.sh 752 480
```

And to take one image 'image.jpg' at 85% jpeg quality with the following command:

```
~$ v412grab -d /dev/video6 -o image.jpg -W 752 - H 480 -q 85 -I -1
```

where **image.jpg** is the output image file in JPEG format, **752** is the width of the image pixel, **480** is the height of the image in pixel, **85** is the quality jpeg in %, **/dev/video6** is the video device, **-I -1** image frame rate.

### 3.2.5 Video Steaming

For video streaming vlc Gstreamer should be installed first. To install it use the following command:

```
~$ sudo apt-get install gstreamer0.1*
```

Before streaming first set the pipes and image size with the following commands:

```
~$ media-pipes.sh
```

```
~$ media-formats.sh 752 480
```

For streaming execute the following commands on robot side first:

```
~$ gst-launch v412src device=/dev/video6 ! autoconvert ! jpegenc quality=70 !  
multipartmux ! tcpserver sink port=5000
```

Then run on the robot side:

```
~$ vlc tcp://ROBOT_IP:5000
```

### 3.3 Set Up for Programming Interface

The most vital components required for this setup are Computer with USB and Wi-Fi and Linux operating system (Kernel 2.6.x and higher) with at least 3 GB on /opt account and at least 50 MB on user account. It is mainly used for cross - compiler programming. As mentioned, this cross compiler mainly supports C/C++ programming languages. Corresponding GCC/G++ compiler is required for compiling these programs. There are many libraries and pre-processor directives specially made for programming this Khepera IV robot. While compiling the scripts these pre-processor directives consisting of declaration of various functions need to be included in compiling command. After compiling the scripts several object files (.o file) are generated. After that executable file (.exe file) will be generated by linking these object files with various libraries where the definition of existing Khepera functions and other functions are written. Here compiler is not same as GCC/G++, **arm-poky-linux-gnucabi-gcc/g++** cross-compiler is used here. Each compilation command becomes very big and which makes it difficult to remember. For that Make file comes in rescue. Using export command, a part of command can

be treated as variable. So, a big command can be divided in many sub-commands using export command. Using colon (:) and dollar (\$) command is used to create different compilation command or target command used for different purpose. Default target is all. **Make/ Make all** used to execute same commands. Details description of Make file is attached in Appendix.

Two script writers have been used for this purpose. Any one of it is recommended to write script file and compile it and make one executable file

### **A. Visual Studio Code**

It's very easy to use development editor. In it both programming in user PC and execution of **.exe** file in the robot can be done in different terminals.

Step 1: Download Visual Studio Code and install it.

Step 2: Install C/C++ Compiler from Add-ons.

Step 3: Write a program and save it with **.c** or **.cpp** extension.

Step 4: Store the make file on the same folder and run **make all** command.

Step 5: One executable file (**.exe** file) (let's say **template**) will be generated after compiling. To run this executable file inside robot use the following command.

```
~$ ./template
```

### **B. Eclipse Neon**

Its setup is relatively complicated with respect to Visual Studio code. There is no need of makefile to compile the code. For that we can develop the code anywhere and generate executable file. Here debugging is little easy with respect to Visual Studio Code.

Step 1: Install the Java Runtime Environment (JRE).

```
~$ sudo add-apt-repository ppa:webupd8team/java
```

```
~$ sudo apt-get update
```

```
~$ sudo apt-get install oracle-java8-installer
```

Step 2: Download the linux version of "Eclipse Neon IDE" for C/C++ Developers and install it.

Step 3: Ensure khepera light toolchain is already installed. Then Run Eclipse and go to file menu "File => C Project" or C++ then click Next button.

Step 4: In the next window "C Project", press the "Advanced Settings" button and on the "C/C++ Build => Settings", In "Cross Settings":

*Prefix: arm-poky-linux-gnueabi*

*Path: /opt/poky/1.8/sysroots/1686-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/*

Step5: On "Cross GCC Compiler" Includes => Include paths, add:

*/opt/poky/1.8/sysroots/cortexa8hf-vsp-neon-poky-linux-gnueabi/usr/include*

Step 6: On Miscellaneous, replace "Other flags" with the following

*-c-march-armv7-a -mfloat-abi-hard -mfpu-neon -mtune-cortex-a8 --sysroot=/opt/poky/1.8/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi*

Step 7: On "The Cross GCC Linker", on Miscellaneous, replace "Linkers flags" with:  
*march=armv7-a -mfloat-abi=hard -mfpu-neon -mtune-cortex-a8 --sysroot=/opt/poky/1.8/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi*

Step 8: On "Cross GCC Linker => Libraries 'at' Libraries (-1), add khepera with the + button on the upper right.

Step 9: At "Libraries search path", add: */opt/poky/1.8/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi/usr/lib*

Step 10: Press "Finish" button.

Step 11: Go to menu "File => New => Source file" choose test.c/test.cpp as filename.

Step 12: After writing the code Press the Green RUN button.

Step 13: One executable file (.exe file)(let's say test) will be generated after compiling. To run this executable file inside robot use the following command:

**~\$ ./test**

## **3.4 Programming Requisites**

### **3.4.1 Libraries and Preprocessors**

Khepera IV API include all the libraries after installing the cross-compiler in /opt/poky folder. The declaration of all in build khepera functions are inside "khepera/khepera.h".

### **3.4.2 Basic Functions**

All the important functions are described in the table below. However, an important variable must be discussed which is used in almost all programs is **dsPic**. It is a pointer variable used to store microcontroller access which is variable of data type **knet\_dev\_t**. It gives access to the microcontroller of khepera. Moreover, it is a static variable.

Apart from **dsPic** variable there are many static and Global variable used for programming. By understanding the functions, the glimpse of those variables can be easily obtained. Apart from the variables there are also several khepera defined data types (using **typedef** function), few Enumerator and several Macros.

Few lists of frequently used functions are tabulated below. What the function do is written in the Remarks section. The input data types, formal arguments of that functions and output data types are also listed corresponding to that particular function.

**Table 3.1 Basic functions and Variables used for Programming**

Function name	Input type		Output type	Remarks	Return	
kh4_set_speed	int	left,	integer	left motor speed(units:encoder)	Returns A value: <0 on error >=0 on success	
	int	right,		right motor speed(units:encoder)		
	knet_dev_t*	hDev		It is a handle to an opened knet socket (Khepera4:dsPic)		
kh4_SetMode	int	regtype,	integer	type of control	Returns A value: <0 on error >=0 on success	
	knet_dev_t*	hDev		It is a handle to an opened knet socket (Khepera4:dsPic)		
kh4_SetRGBLeds	int	char	integer	left_R,	left led, R colour on 6 bits	Returns A value: <0 on error >=0 on success
				left_G,	left led, G colour on 6 bits	
				left_B,	left led, B colour on 6 bits	
				right_R,	right led, R colour on 6 bits	
				right_G,	right led, G colour on 6 bits	
				right_B,	right led, B colour on 6 bits	
				back_R,	back led, R colour on 6 bits	
				back_G,	back led, G colour on 6 bits	
	back_B,	back led, B colour on 6 bits				
	knet_dev_t*	hDev	It is a handle to an opened knet socket.(Khepera4:dsPic)			
kb_change_term_mode	int	dir	void	1= mode changed to non-blocking, 0 mode reverted to previous	Return none	
timeval_diff	struct timeval*	difference,	long	difference between the two times in structure timeval type,	Returns difference between the times in [us]	
	struct timeval*	end_time,		end time		
	struct timeval*	start_time,		start time		

kb_clrscr	void		void	Clear the console screen	none
kb_change_term_mode	int dir		void	Change terminal for getchar to return immediately dir 1=mode change to non-blocking, 0 mode reverted to previous	Returns none
ctrlc_handler	int sig		void	static	Returns none
kb_kbhit	void		integer	Test if any key was pushed	Returns -1 if error occurred >=0 number of characters to read
kb4_get_speed	int*	left	integer	kh4_get_speed get motors speed left motor speed(units: encoder)	Returns a value: <0 on error >=0 on success
	int*	right		right motor speed (units: encoder)	
	knet_dev_t*	hDev		It is a handle to an opened knet socket(Khepera4:dsPic)	
kb_change_term_mode	int dir		void	Change terminal mode for get char to return immediately Dir 1=mode changed to non-blocking, 0 mode reverted to previous	Return none
main	int argc,char *argv[]		integer	Provide a basic test program to control a K-Team pan tilt camera with a KoreMotor using the KoreBot library	Returns none
kb4_init	int	argc,	integer	kh4_init initializes some things like the GPIO40 pin.This function needs to be called BEFORE any other functions	Returns a value: <0 on error =0 on success
	char*	argv[]			

### 3.4.3 Steps for Program Execution

This section provides the basic steps for compiling and running the program after it has been written in the editor. Further details regarding this section are provided in Appendix part at the end of this thesis.

Step 1: Write the program in the editor and save it as **program-template.c**

Step 2: Open a new terminal and enter into the path where the program has been saved using cd command as shown below:

```
cd Khepera4_development  
cd libkhepera-2.1 cd template  
cd template
```

Step 3: Compile the program using **make** command

Step 4: Open another new terminal where execution of the program will be done.

Step 5: Establish a connection between the robot and computer through Wi-Fi by the command:

```
ssh root @10.42.0.176
```

Step 6: Mount the program using the following command: mount 10.42.0.1:/home/khepera /home/root

Step 7: Enter into the path again where the program has been written by the following commands:

```
cd /home/root  
cd khepera4_development  
cd libkhepera-2.1  
cd template
```

Step 8: now run the program using the following command and then press enter. **./template**

### 3.5 Conclusion

This chapter describes the entire setup process required to work and program with Khepera IV robot. Then description of various variables and functions useful for programming are also explained vividly. Thus, this chapter would act as a useful guide for novice users of this robot.

## References

- [1] [https://www.k-team.com/mobile-robotics-products/khepera-iv/manuals\\_downloads](https://www.k-team.com/mobile-robotics-products/khepera-iv/manuals_downloads)
- [2] [http://ftp.k-team.com/KheperaIV/software/light\\_tools/](http://ftp.k-team.com/KheperaIV/software/light_tools/)
- [3] [https://www.k-team.com/mobile-robotics-products/khepera-iv/linux\\_os](https://www.k-team.com/mobile-robotics-products/khepera-iv/linux_os)
- [4] <https://en.wikipedia.org/wiki/USB>
- [5] [https://www.cablestogo.com/learning\\_connector-guides/usb](https://www.cablestogo.com/learning_connector-guides/usb)
- [6] E. Amir, H. Balakrishnan, S. Seshan, R. Katz, "Efficient TCP over Networks with Wireless Links", *The First International ACM Conference on Mobile Computing and Networking, 1995*.
- [7] A. Bakre, B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts", *Proceedings of the International Conference on Distributed Computing Systems, 1995*,
- [8] "IEEE 802.11", *Wireless Access Method and Physical Specification, May 1993*,
- [9] H. Wang, N. Moayeri, "Finite state markov channel -a useful model for radio communication channels", *IEEE Trans. Vehic. Tech.*, pp. 163-171. Feb 1995,
- [10] D. DUCHAMP, N. Reynolds, "Measured performance of a wireless lan", *17<sup>th</sup> Conference on Local Computer Networks*, pp. 494-499, 1992.



# Chapter 4

## Socket Communication

### 4.1 Introduction

Socket programming [1] is defined as a notion that enables communication between two nodes (i.e. the processing computer and the robot). The basic idea of this type of communication is that one socket (node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server. It utilizes the TCP/IP protocol suite [2-4] due to its good failure recovery, its capability to add networks without interrupting existing services, high error-rate handling performance and provides low data overhead.

### 4.2 TCP/IP Protocol Suite

TCP/IP defines the acronym that is commonly used for the set of network protocols that compose the Internet Protocol suite. It is the conceptual model and set of communications protocols used on the Internet and similar computer networks. Moreover, it is commonly known as TCP/IP because the foundational protocols in the suite are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The TCP/IP protocol suite maps to a four-layer conceptual model known as the DARPA [5-6] model, which was named after the U.S. government agency that initially developed TCP/IP. The four layers of the DARPA model are: Application, Transport, Internet, and Network Interface. Each layer in the DARPA model corresponds to one or more layers of the seven-layer OSI model as shown in the figure below.

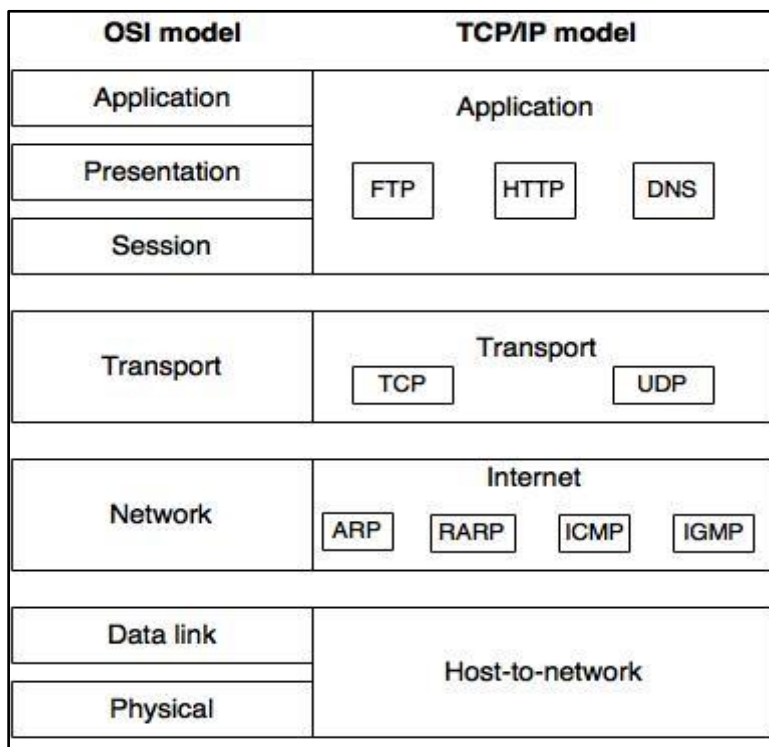


Figure 4.1 TCP/IP and OSI model

### **4.2.1 Physical and Data Link Layers**

At these layers, TCP/IP does not define any specific protocol. It supports all the standard and proprietary protocols. A network in a TCP/IP can be a local-area network or a wide area network.

### **4.2.2 Network Layer**

At this layer, TCP/IP supports the Internet Protocol. IP, in turn, uses four supporting protocols which are ARP [7], RARP [8], ICMP [9], and IGMP [10].

#### **I. Internet Protocol (IP)**

This protocol is the transmission mechanism used by TCP/IP. It is an unreliable and connectionless protocol providing best effort delivery service. The term best effort means that the IP does no error checking or tracking. It assumes the unreliability of the underlying layer and does its best to provide a transmission to its destination but without any guarantees. IP transports called datagrams, each of which is transported separately. Datagrams can travel along different routes and can arrive out of sequence or be duplicated. IP does not keep track of the routes and has no facility for reordering the datagrams once they have arrived at the destination.

#### **II. Address Resolution Protocol (ARP)**

This protocol is used to associate a logical address with a physical address. On a typical physical network such as LAN, each device address, usually imprinted on a link is identified by a physical address or a station imprinted on the network interface card (NIC). ARP is used to find the physical address of the node when its Internet address is known.

#### **III. Reverse Address Resolution Protocol (RARP)**

It allows its host to discover its Internet address when it knows only its physical address. It is used when a computer is connected to the network for the first time or when a diskless computer is booted.

#### **IV. Internet Control Message Protocol (ICMP)**

This protocol is mechanism used by hosts and gateways to send notification of datagram problems back to the sender, thus providing the service of sending query and error reporting messages.

#### **V. Internet Group Message Protocol (IGMP)**

This protocol is used to facilitate the simultaneous transmission of a message to a group of recipients.

### **4.2.3 Transport Layer**

Earlier, the transport layer was represented in TCP/IP by two protocols: TCP and UDP. IP is a host to host protocol, meaning it can deliver a packet from one physical device to another. UDP and TCP are the transport level protocols responsible for delivery of a message from a process

(running program) to another process. A new transport layer protocol, SCTP, has been devised to meet the needs of some of the newer applications.

### **A. User Datagram Protocol (UDP)**

This protocol is the simpler of the two standard TCP/IP transport protocols. It is a process to process protocol that adds only port addresses, check sum, error control, and length of information to the data from the upper layer,

### **B. Transmission Control Protocol (TCP)**

It provides full transport layer services to applications. It is a reliable stream transport protocol. At the sending end of each transmission, TCP divides a stream of data into smaller units called segments. Each segment includes a sequence number for reordering after recipient, together with an acknowledgment number for the segments received. Segments are carried across the internet inside of the IP datagrams. At the receiving end, TCP collects each datagrams as it comes in and reorders the transmission based on sequence numbers.

### **C. Stream Control Transmission Protocol (STCP)**

This protocol provides support for newer applications such as voice over the Internet. It combines the best features of UDP and TCP.

## **4.2.4 Application Layer**

The Application layer allows applications to access the services of the other layers, and it defines the protocols that applications use to exchange data. The Application layer contains many protocols, and more are always being developed. The most widely known Application layer protocols help users exchange information are:

- The Hypertext Transfer Protocol (HTTP) [11] transfers files that make up pages on the World Wide Web.
- The File Transfer Protocol (FTP) [12] transfers individual files, typically for an interactive user session.
- The Simple Mail Transfer Protocol (SMTP) [13] transfers mail messages and attachments.

Additionally, the following Application layer protocol helps to use and manage TCP/IP networks:

- Domain Name System (DNS) [14] protocol resolves a host name, such as 'www.microsoft.com', to an IP address and copies name information between DNS servers.
- The Routing Information Protocol (RIP) [15] is a protocol that routers use to exchange routing information on an IP network.
- The Simple Network Management Protocol (SNMP) [16] collects and exchanges network management information between a network management console and network devices such as routers, bridges, and servers.

### 4.2.5 Advantages of using TCP/IP Protocol Suite

- a) It is a freely available protocol and not a secret protocol that is owned by a single company. This makes it possible for anyone with sufficient technical knowledge to improve it.
- b) It is compatible with virtually all modern operating systems, and thus it enables almost any system to communicate with any other system.
- c) It is also compatible with virtually all types of computer hardware and network configurations.
- d) It is routable protocol, which means that it can determine the most efficient path for every packet as it moves through the network. This makes TCP/IP highly scalable and thus the size of the network virtually unlimited (e.g., the Internet).
- e) It provides reliable data delivery. Reliable means that it can guarantee that the data is delivered to its intended destination (e.g., through the use of error checking and retransmission of corrupted or missing packets).
- f) The use of a single (and relatively simple) addressing scheme, referred to as IP addressing, allows administrator to transfer their knowledge of TCP/IP to any TCP/IP network without the need to learn a new addressing scheme.

### 4.3 Establishment of Socket Communication

For implementation of any task assigned to Khepera robot, socket communication is established between the robot that works in C programming platform and the working computer [17-18] that utilizes the MATLAB software for various processing activities. However, socket communication in general, be it for any platform, has some basic steps as explained below. It is also represented diagrammatically in the figure 4.2.

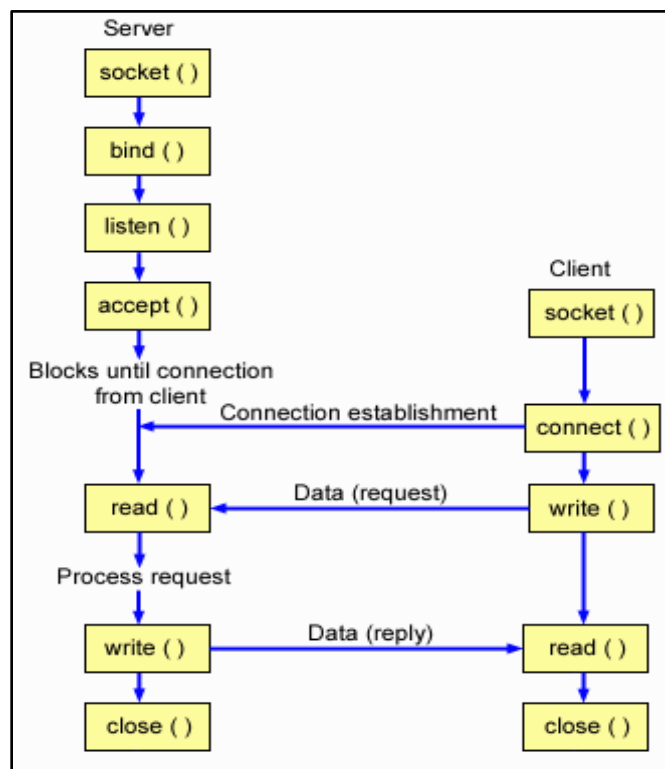


Figure 4.2 Server-Client Model

### 4.3.1 Server Model

The Server process takes a request from the client end. After getting this request, this process will perform the required processing, gather the requested information, and send it to the requestor client. Once done, it becomes ready to serve another client. The basic steps for this process are:

- i. Creation of a socket using the **socket()** system call.
- ii. Binding the socket to an address using the **bind()** system call. For a server socket on the Internet an address consists of a port number on the host machine.
- iii. Listening for connections using the **listen()** system call.
- iv. Accepting a connection with the **accept()** system call. This call typically blocks the connection until a client connects with the server.
- v. Send and receive data by the use of **read()** and **write()** system calls.

### 4.3.2 Client Model

The Client process involves the production of a request for information. After getting the response this process may terminate or may do some other processing. The basic steps for this process are:

1. Creation of a socket using the **socket()** system call.
2. Connection of the socket to the address of the server using the **connect()** system call.
3. Send and receive data by the use of the **read()** and **write()** system calls respectively.

## 4.4 Socket Programming in C

The stages for implementing a server and client in C programming platform is discussed below:

### 4.4.1 Stages for Server Implementation

#### A. Socket Creation

In the beginning, a socket function needs to be declared to get the socket descriptor which is done by the following code:

```
int sockfd = socket(domain, type, protocol)
```

**Table 4.1 Socket creation variables and declaration**

Domain	AF_UNIX-connection inside same machine or AF_INET -connection with different machine
Type	SOCK_STREAM - TCP connection or SOCK_DGRAM - UDP connection
Protocol	Define here when there is any additional protocol. Otherwise, define it as 0

Next, decision regarding which struct needs to be used based on what domain (Linux for our case) is used.

```

struct sockaddr_in
{
short int sin_family;
int sin_port;
struct in_addr sin_addr;
}

```

### B. Binding the Socket

After creation of the socket, bind function binds the socket to the address and port number specified in addr by the following code:

```
int bind(int sockfd, const struct sockaddr* addr, socklen_t addrlen);
```

### C. Listening to Connections

It puts the server socket where it waits for the client to approach the server to make a connection. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

```
int listen(int sockfd, int backlog);
```

backlog	Defines the maximum length to which the queue of pending connections for sockfd may grow.
---------	---

### D. Accepting a Connection

It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.

```
int new_socket=accept(int sockfd, struct sockaddr * addr, socklen_t * addrlen);
```

### E. Read/Write Data

Once a connection has been established, server will either read or write to the connection using the code:

```
i=read(newsockfd, buffer,255);
```

Socket_description	Put server socket description depending on reading data from client
read buffer	Content of the data retrieved
read buffer length	Length of the output string

```
w=write(newsockfd,"I got your meassage",18);
```

socket_description	Put server socket description depending on sending data to client
write buffer	Data to be sent
write buffer length	Length of the output string

## 4.4.2 Stages for Client Implementation

### A. Socket Creation

The procedure for socket creation at the Client end is exactly same as that of the server side.

### B. Connection

The **connect()** system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server's and port is specified in addr.

```
int connect(int sockfd, const struct sockaddr * addr, socklen_t addrlen);
```

### C. Read/Write Data

Once a connection has been established, the client will either read or write to the connection using the code:

```
n=read(newsockfd, buffer,255);
```

socket description	Put client socket description depending on reading data from server
read buffer	Content of the data retrieved
read buffer length	Length of the output string

```
n= write(sockfd, buffer,strlen(buffer));
```

socket_description	Put client socket description depending on sending data to client
write buffer	Data to be sent
write buffer length	Length of the output string

## 4.5 Socket Programming in MATLAB

The procedure for implementing a server and client in MATLAB are discussed below:

### 4.5.1 Server Implementation

First acceptance of a connection is done from a machine on a particular port no. say 3000 which is implemented using the code below:

```
t=tcipip('0.0.0.0',3000,'NetworkRole','server');
```

Here, the IP address '0.0.0.0' means that the server will accept the first machine that tries to connect.

Then, a connection is opened by the following code:

```
fopen(t);
```

This will not return until a connection is received.

Then finally the data is read and confirmed visually by plotting it by the code below:

```
data=fread(t, t.BytesAvailable);
```

```
plot(data);
```

## 4.5.2 Client Implementation

The client code is written on a different script file. First, the data is created and visualized by the following set of codes:

```
data = sin(1:64);  
plot(data)
```

Then, a client interface is created and opened by the following set of codes:

```
t= tcpip('localhost', 3000, 'NetworkRole', 'client');  
fopen(t);
```

The data is finally written to the server session by the following code:

```
fwrite(t, data);
```

## 4.6 Conclusion

This chapter provides a detailed explanation of the socket communication implementation. First, a vivid description has been provided regarding the TCP/IP protocol suite to have a clear idea regarding the underlying networking used for this communication. Then, a detailed illustration has been provided for implementation of socket and client nodes. Thus, this chapter helps to develop the required interfacing while working with robots like Khepera.



## References

- [1] Warren W. Gay, "Linux socket programming by example", Que, 2000.
- [2] B. A. Forouzan, "TCP/IP Protocol Suite", Second Edition, McGraw Hill, 2003.
- [3] P. Porra, A. Valdes, "Live Traffic Analysis of TCP/IP gateways". The proceeding of ISOC symp. on Network and Distributed System Security (NDSS'98,) March 1998.
- [4] K. McCloghrie, and Rose, "M. Management Information Base for Network Management of TCP/IP-based Internets", RFC 1066, 1988.
- [5] L Feinstein, D. Schnackenberg, "Statistical Approaches to DDOS Attack Detection and Response", Proceedings of DARPA Information Survivability Conference and Expastion (DISCEX 03), April 2003.
- [6] J. Postel, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.
- [7] Plummer, D., "An Ethernet Address Resolution Protocol", RFC 826, MIT-LCS. November 1982.
- [8] R. Finlayson, T.Mann, J.C. Mogul, and M.Theimer, "Reverse Adress Resolution Protocol", RFC 902, June 1984.
- [9] J. Postel, "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", RFC 792, USC/Information Sciences Institute, September 1981.
- [10] B. Cain, S. Deering, A. Thyagarajan, Internet Group Management Protocol Version 3, Nov 1999.
- [11] R Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Bemers-Lee, "Hypertext Transfer Protocol, HTTP/1.1", RFC 2616, June 1999.
- [12] J. Postel, and J. Reynolds, "File Transfer Protocol (FTP)", STD 5, RFC 959, October 1985.
- [13] K. Moore, "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)". RFC 3461, January 2003.
- [14] P. Mockapetris, "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [15] G. Malkin, "RIP Version 2 Protocol Applicability Statement", RFC 1722, Xylogics, Inc., November 1994.
- [16] W. Stallings, SNMP, SNMPv2. SNMPv3 and RMON I und 2. Rending, MA:Addison Wesley, 1999,
- [17] M. Xue, C. J. Zhu, "The socket programming and software design for communication based on client/server. Proceedings of the 2009 Pacific-Asia Conference on Circuits Communications and System PACCS 2009", Proceedings of the 2009 Pacific-Asia Conference on Circuits Communications and System PACCS 2009, pp. 775-777, 2009.
- [18] K. L. Eddie Law. R. Leung, "A design and implementation of active networks socket programming", Microprocessors and Microsystems, vol. 27, no, 5-6, pp. 277-284. June 2003.

# Chapter 5

## Kalman & Extended Kalman Filter

### 5.1 Introduction

This chapter mainly deals with Kalman and extended Kalman filter [1-4] which was implemented for the soccer goal keeper for ball tracking and interception using the Khepera IV. Kalman filter is the analytical implementation of Bayesian filtering [14] recursions for linear Gaussian state space models. At the beginning of the chapter it describes the Bayesian filters, brief description on probability then our thesis objective using Kalman Filter i.e. to predict distance and angle. But to increase the accuracy in range and position estimation along near linear trajectories it is achieved by employing extended Kalman filter as Kalman filter [11] works best for linear systems and extended Kalman filter in non-linear systems [5-10].

### 5.2 What is estimation?

Estimation is the process by which we infer the value of a quantity of interest, (in our case these are distance and angle) represented by  $x$ , by processing data that is in some way dependent on  $x$ . In all the below cases we require estimation.

- Measured data corrupted by noise—uncertainty in input transformed into uncertainty in inference (e.g. Bayes rule)
- Quantity of interest not measured directly (e.g. odometry in skid-steer robots)
- Incorporating prior (expected) information (e.g. best guess or past experience)
- Open-loop prediction (e.g. knowing current heading and speed, infer future position)
- Uncertainty due to simplifications of analytical models (e.g. performance reasons linearization)

#### 5.2.1 Bayes Theorem

In probability theory and statistics, **Bayes' theorem** [13] (alternatively **Bayes' law** or **Bayes' rule**) describes the probability of an event, based on prior knowledge of conditions that might be related to the event, which is described below:

$$P(B|A) = P(A|B) P(B)/P(A),$$

where  $P(B|A)$  is the posterior probability and  $P(A|B)$  is the likelihood. This is a fundamental rule for machine learning (pattern recognition) as it allows to compute the probability of an output  $B$  given measurements  $A$ . The prior probability is  $P(B)$  without any evidence from measurements. The likelihood  $P(A|B)$  evaluates the measurements given an output  $B$ . Seeking the output that maximizes the likelihood (the most likely output) is known as the maximum likelihood estimation (ML). The posterior probability  $P(B|A)$  is the probability of  $B$  after taking the measurement  $A$  into account. Its maximization leads to the maximum a-posteriori estimation (MAP).

### 5.2.2 Overview of the Probability Rules

The Product rule:  $P(A, B) = P(A|B) P(B) = P(B|A) P(A)$

The Sum rule:  $P(B) = \sum_A P(A, B) = \sum_A P(B|A)P(A)$

Random events A, B are independent,  $P(A, B) = P(A) P(B)$ , and the independence means:  $P(A|B) = P(A)$ ,  $P(B|A) = P(B)$ . A, B are conditionally independent,  $P(A, B|C) = P(A|C)P(B|C)$

The Bayes theorem:

$$P(A|B) = P(A, B)/P(B) = P(B|A)P(A)/P(B) = P(B|A)P(A) / \sum_A P(B|A)P(A)$$

### 5.2.3 Mean & Covariance

The **mean** of a discrete random variable  $X$  is a weighted average of the possible values that the random variable can take. Unlike the sample mean of a group of observations, which gives each observation equal weight, the mean of a random variable weights each outcome  $x_i$  according to its probability,  $p_i$ .

In statistics, **variance** refers to the spread of a data set. It's a measurement used to identify how far each number in the data set is from the mean.

**Covariance** provides insight into how two variables are related to one another. More precisely, covariance refers to the measure of how two random variables in a data set will change together.

A positive covariance means that the two variables at hand are positively related, i.e. they move in the same direction.

A negative covariance means that the variables are inversely related, i.e. they move in opposite directions.

Expectation = the average of a variable under the probability distribution.

$$\text{Continuous definition of Expectation: } E(x) = \int_{-\infty}^{\infty} x f(x) dx \quad (5.1)$$

$$\text{Discrete definition of Expectation: } E(x) = \sum_x x P(x) \quad (5.2)$$

In probability theory and statistics, covariance is a measure of the joint variability of two random variables. Mutual covariance  $\sigma_{xy}$  of two random variables  $X, Y$  is given by:

$$\sigma_{xy} = E((X - \mu_x)(Y - \mu_y)) \quad (5.3)$$

Covariance matrix is symmetric i.e.  $\Sigma = \Sigma^T$  and positive semidefinite (as the covariance matrix is real valued, the positive-semidefinite means that  $x^T M x \geq 0$  for all  $x \in \mathbb{R}$ . For  $n$  variables  $X_1, \dots, X_n$  is

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_{1n}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{n1}^2 & \cdots & \sigma_n^2 \end{bmatrix} \quad (5.4)$$

### 5.2.4 MAP - Maximum A-Posteriori Estimation

In many cases, we already have some prior (expected) knowledge about the random variable  $x$ , i.e. the parameters of its probability distribution  $p(x)$ .

With the Bayes rule, we go from prior to a-posterior knowledge about  $x$ , when given the observations  $z$ :

$$p(x|z) = p(z|x) p(x) / p(z)$$

$$p(x|z) = \text{likelihood} \times \text{prior/normalizing constant} \sim C \times p(z|x)p(x)$$

Given an observation  $z$ , a likelihood function  $p(z|x)$  and prior distribution  $p(x)$  on  $x$ , the maximum a posteriori estimator MAP finds the value of  $x$  which maximizes the posterior distribution  $p(x|z)$ :

$$\hat{x}_{MAP} = \underset{x}{\text{argmax}} p(z|x)p(x) \quad (5.5)$$

### 5.2.5 MMSE – Minimum Mean Squared Error

We want to find such a  $\hat{x}$ , an estimate of  $x$ , that given a set of measurements  $Z_k = \{z_1, z_2, \dots, z_k\}$  it minimizes the mean squared error between the true value and this estimate.

$$\hat{x}_{MMSE} = \underset{x}{\text{argmin}} \mathcal{E} \{(\hat{x} - x)^T (\hat{x} - x) | Z^k\} = \mathcal{E}\{x | Z^k\} \quad (5.6)$$

The purpose of the above step in MMSE estimate as given a set of measurements is the mean of that variable conditioned on the measurements. In LSQ the  $x$  is a unknown constant but in MMSE  $x$  is a random variable.

### 5.2.6 RBE- Recursive Bayesian Estimation

RBE [14] is the natural extension of MAP to time-stamped sequence of observations being processed at each time step. In RBE we use the priory estimate and current measurement to compute the posteriori estimate  $\hat{x}$ . When the next measurement comes, we use our previous posteriori estimate as a new prior and proceed with the same estimation rule. Hence for each time-step  $k$  we obtain an estimate for its state given all observations up to that time (the set  $Z^k$ ). Using the Bayes rule and conditional independence of measurements ( $z_k$  being single measurement at time  $k$ ):

$$p(x, Z^k) = p(x|Z^k) = p(Z^k|x) = p(Z^{k-1}|x)p(Z_k|x)p(x) \quad (5.7)$$

We express  $p(Z^{k-1}|x)$  and substitute for it to get:

$$p(x, Z^k) = \frac{p(Z_k|x)p(x|Z^{k-1})p(Z^{k-1})}{p(Z^k)} \quad (5.8)$$

RBE is extension of MAP to time-stamped sequence of observations. We obtain RBE as the likelihood of current  $k$ th measurement  $\times$  prior which is our last best estimate of  $x$  at time  $k - 1$  conditioned on measurement at time  $k - 1$  (denominator is just a normalizing constant).

$$p(x, Z^k) = \frac{p(Z_k|x)p(x|Z^{k-1})p(Z^{k-1})}{p(Z^k)} = \frac{\text{current likelihood} * \text{last best estimate}}{\text{normalizing constant}}$$

### 5.2.7 LSQ- Least Squares Estimation

A. Given measurements  $\mathbf{z}$ , we wish to solve for  $\mathbf{x}$ , assuming linear relationship:

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

If  $\mathbf{H}$  is a square matrix with  $\det \mathbf{H} \neq 0$  then the solution is trivial:

$$\mathbf{x} = \mathbf{H}^{-1} \mathbf{z},$$

otherwise (most commonly), we seek such solution  $\hat{\mathbf{x}}$  that is closest (in Euclidean distance sense) to the ideal:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{H}\mathbf{x} - \mathbf{z}\|^2 = \underset{\mathbf{x}}{\operatorname{argmin}} \{(\mathbf{H}\mathbf{x} - \mathbf{z})^T (\mathbf{H}\mathbf{x} - \mathbf{z})\} \quad (5.9)$$

### B. Weighted Least Squares Estimation

If we have information about reliability of the measurements in  $\mathbf{z}$ , we can capture this as a covariance matrix  $\mathbf{R}$  (diagonal terms only since the measurements are not correlated):

$$\mathbf{R} = \begin{bmatrix} \sigma_1^2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_1^2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

In the error vector  $\mathbf{e}$  defined as  $\mathbf{e} = \mathbf{H}\mathbf{x} - \mathbf{z}$  we can weight each its element by uncertainty in each element of the measurement vector  $\mathbf{z}$ , i.e. by  $\mathbf{R}^{-1}$ . The optimization criteria then become:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{R}^{-1}\mathbf{H}\mathbf{x} - \mathbf{z}\|^2 \quad (5.10)$$

In the same way we obtain the weighted least squares:

$$\mathbf{x} = (\mathbf{H}^{-1}\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} \quad (5.11)$$

The world is non-linear, nonlinear model function  $\mathbf{h}(\mathbf{x}) \rightarrow$  non-linear LSQ:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{h}(\mathbf{x}) - \mathbf{z}\|^2 \quad (5.12)$$

We seek such  $\delta$  that for  $\mathbf{x}_1 = \mathbf{x}_0 + \delta$  the  $\|\mathbf{h}(\mathbf{x}_1) - \mathbf{z}\|^2$  is minimized.

We use Taylor series expansion:

$$\mathbf{h}(\mathbf{x}_0 + \delta) = \mathbf{h}(\mathbf{x}_0) + \nabla\mathbf{H}_{\mathbf{x}_0}\delta \quad (5.13)$$

$$\|\mathbf{h}(\mathbf{x}_1) - \mathbf{z}\|^2 = \|\mathbf{h}(\mathbf{x}_0) + \nabla\mathbf{H}_{\mathbf{x}_0}\delta\|^2 = \|\nabla\mathbf{H}_{\mathbf{x}_0}\delta - (\mathbf{z} - \mathbf{h}(\mathbf{x}_0))\|^2 \quad (5.14)$$

Where  $\nabla\mathbf{H}_{\mathbf{x}_0}$  is Jacobian of  $\mathbf{h}(\mathbf{x})$ :

$$\nabla\mathbf{H}_{\mathbf{x}_0} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \dots & \frac{\partial h_n}{\partial x_m} \end{bmatrix} \quad (5.15)$$

The extension of LSQ to the non-linear LSQ can be formulated as an algorithm below:

1. Start with an initial guess  $\hat{\mathbf{x}}$ .
2. Evaluate the LSQ expression for  $\delta$  (update the  $\nabla H_{\hat{\mathbf{x}}}$  and substitute).

$$\delta := (\nabla H_{\hat{\mathbf{x}}}^T \nabla H_{\hat{\mathbf{x}}})^{-1} \nabla H_{\hat{\mathbf{x}}}^T [\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}})] \quad (5.16)$$

3. Apply the  $\delta$  correction to our initial estimate:  $\hat{\mathbf{x}} := \hat{\mathbf{x}} + \delta$
4. Check for the stopping precision: if  $\|\mathbf{h}(\hat{\mathbf{x}}) - \mathbf{z}\|^2 > \epsilon$  proceed with step (2) or stop otherwise.

## 5.3 Kalman Filter

### 5.3.1 What is a Kalman Filter?

Theoretically Kalman Filter is an estimator for what is called the linear-quadratic problem, which is the problem of estimating the instantaneous "state" of a linear dynamic system perturbed by white noise-by using measurements linearly related to the state but corrupted by white noise. The resulting estimator is statistically optimal with respect to any quadratic function of estimation error.

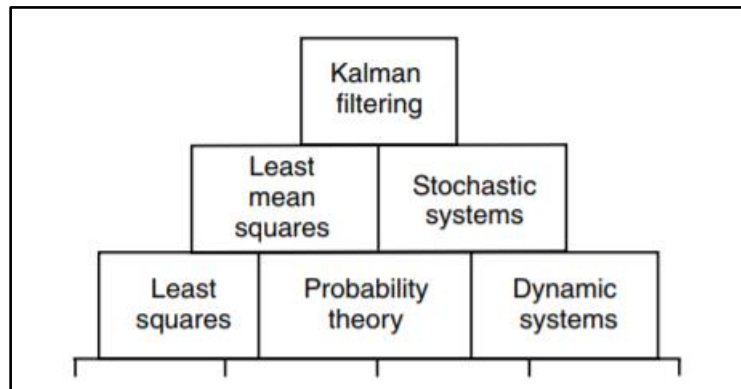
Practically, it is certainly one of the greatest discoveries in the history of statistical estimation theory and possibly the greatest discovery in the twentieth century. It has enabled mankind to do many things that could not be done without it, and it has become as indispensable as silicon in the makeup of many electronic systems. Its most immediate applications have been for the control of complex dynamic systems such as continuous manufacturing processes, aircraft, ships, or spacecraft. Kalman filter provides a mean for inferring the missing information from indirect (and noisy) measurements. The Kalman filter is also used for predicting the likely future courses of dynamic systems that people are not likely to control, such as the flow of rivers during flood, the trajectories of celestial bodies, or the prices of traded commodities.

### 5.3.2 How It Came to Be Called a Filter

It might seem strange that the term "filter" would apply to an estimator. More commonly, a filter is a physical device that removes unwanted fractions of mixtures. The word comes from the same medieval Latin stem for the material was used as a filter for liquids. Originally a filter solved the problem of separating unwanted components of gas liquid solid mixtures. In the era of crystal radios and vacuum tubes, the term was applied to analog circuits that "filter" electronic signals. These signals are mixture of different frequency components, and these physical devices preferentially attenuate unwanted frequencies.

With Kalman filtering the term assumed a meaning that is well beyond the original idea of separation of the components of a mixture. It has also come to include the solution of an inversion problem, in which one knows how to present the measurable variables a function of the variables of principal interest. In essence it inverts this functional relationship and estimates the independent variables as inverted functions of the dependent (measurable) variables. These variables of interest are allowed to be dynamic, with dynamics that are only partially predictable.

### 5.3.3 Its Mathematical Foundations



**Figure 5.1 Foundational concepts in Kalman filtering**

Figure 5.1 depicts the essential subjects forming the foundations for Kalman filtering theory. Although it shows Kalman filtering as the apex of a pyramid, it is itself but part of the foundations of another discipline—modern control theory—and a proper subset of statistical decision theory. Most of them are already explained in the chapter so far.

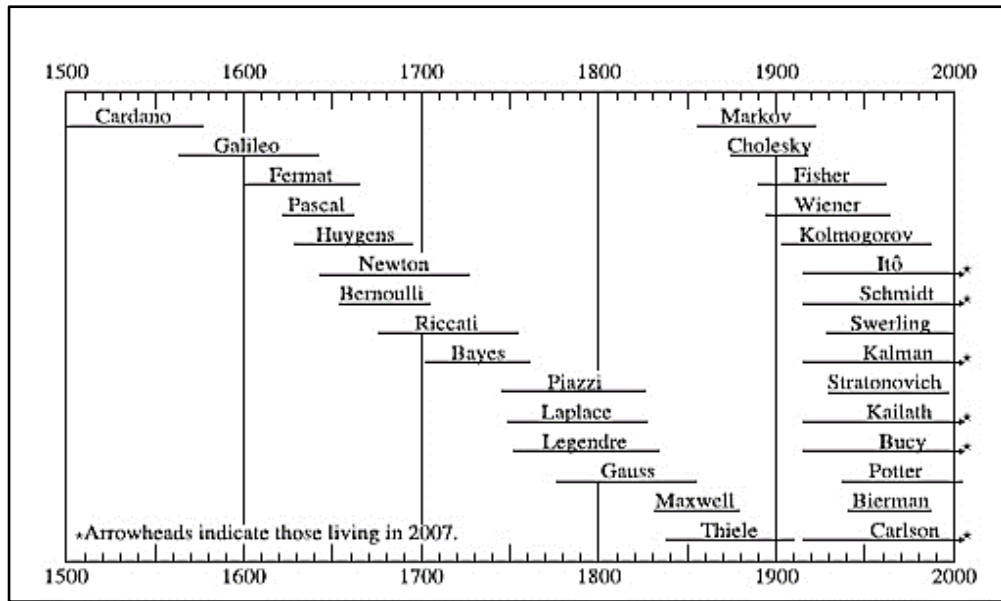
### 5.4 Uses of Kalman Filter

It is used as a tool almost exclusively for two purposes:

- a) **Estimating the state of dynamic systems.** Nearly all physical systems are dynamic to some degree. The kalman filter allows us to estimate state of dynamic systems with certain types of random behaviour by using such statistical information.
- b) **Performance analysis of estimation systems.** The objective of design analysis is to determine how best to use these sensor types for a given set of design criteria. These criteria used to measure the estimation accuracy and system cost.

### 5.5 Optimal Estimation Methods

Kalman filter is the result of an evolutionary process of ideas from many creative thinkers over many centuries. here some of the seminal ideas in this process, the discoverers of which are listed in historical perspective in figure 5.2 below. This list is by no means exhaustive. The figure covers only half a millennium, and the study and development of mathematical concepts goes back beyond history.



**Figure 5.2 Lifelines of some important contributors to estimation**

An important figure in probability theory and the theory of random processes in the 20<sup>th</sup> century was the Russian academician Andrei Nikolaevich Kolmogorov. Starting around 1925, working with Aleksandr Yakovlevich Khinchin and others, he re-established the foundations of probability theory on measure theory, which had originated as the basis for integration theory and became the accepted mathematical basis of probability and random processes. Along with Norbert Wiener, he is credited with founding much of the theory of prediction, smoothing and filtering of Markov processes, and the general theory of ergodic processes. His was the first formal theory of optimal estimation for systems involving random processes.

### 5.5.1 Kalman Filter culmination

It is the culmination of progression of models and associated optimal estimation methods for dynamic processes.

- a) Wiener–Kolmogorov models use the power spectral density (PSD) in the frequency domain to characterize the dynamic and statistical properties of a dynamic process.
- b) Control theorists use linear differential equations as dynamic system models. This led to the development of mixed models, in which the dynamic system functions as a “shaping filter” excited by white noise. Coefficients of the linear differential equations determine the shape of the output PSD, and the shape of the PSD defines the Wiener–Kolmogorov estimator. This approach allows the dynamic system model to be time-varying. These linear differential equations can be modelled as a system of first-order differential equations in what has come to be called **state space**.

With the additional assumption of finite dimensionality, Kalman was able to derive the Wiener–Kolmogorov filter as what we now call the Kalman filter.



### 5.5.2 Linear Kalman Filter to Extended Kalman Filter

Linear models in the non-linear environment is **BAD**.

Non-linear models in the non-linear environment is **BETTER**.

Assume the following the non-linear system model function  $f(x)$  and the non-linear measurement function  $h(x)$ , we can reformulate:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{(k),k}) + \mathbf{v}_{(k)} \quad (5.17)$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_{(k)}, \mathbf{u}_{(k),k}) + \mathbf{w}_{(k)} \quad (5.18)$$

$\mathbf{x}_k$  : State Vector

$\mathbf{v}_{(k)}$  : Measurement noise vector

$\mathbf{w}_{(k)}$  : Process noise vector

$\mathbf{f}(\cdot)$  : Process non-linear vector function

$\mathbf{h}(\cdot)$  : Observation non-linear vector function

The main idea behind EKF is to linearize the non-linear model around the “best” current estimate. This is realized using a Taylor series expansion. Assume an estimate  $\hat{\mathbf{x}}_{(k-1|k-1)}$  then

$$\mathbf{x}(k) \approx \mathbf{f}(\hat{\mathbf{x}}(k-1|k-1), \mathbf{u}_{(k),k}) + \nabla \mathbf{F}_x [\mathbf{x}_{(k-1)} - \hat{\mathbf{x}}(k-1|k-1)] + \dots + \mathbf{v}_{(k)}$$

Where the term  $\nabla \mathbf{F}_x$  is a Jacobian of  $f(x)$  with respect to  $x$  evaluated at  $\hat{\mathbf{x}}(k-1|k-1)$ :

$$\nabla \mathbf{F}_x = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \quad (5.19)$$

The same holds for the observation model, i.e. the predicted observation  $\mathbf{z}_{(k|k-1)}$  is the projection of  $\hat{\mathbf{x}}_{(k|k-1)}$  through the non-linear measurement model.

Hence assume an estimate  $\hat{\mathbf{x}}_{(k|k-1)}$  then

$$\mathbf{z}_{(k)} \approx \mathbf{h}(\hat{\mathbf{x}}(k|k-1), \mathbf{u}_{(k),k}) + \nabla \mathbf{H}_x [\hat{\mathbf{x}}(k|k-1) - \mathbf{x}_{(k)}] + \dots + \mathbf{w}_{(k)}$$

Where the  $\nabla \mathbf{H}_x$  is Jacobian of  $h(x)$  with respect to  $x$  evaluated at  $\hat{\mathbf{x}}(k|k-1)$ :

$$\nabla \mathbf{H}_x = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \dots & \frac{\partial h_n}{\partial x_m} \end{bmatrix} \quad (5.20)$$

## 5.6 Kalman filter in target tracking and Interception by Mobile Robots

Target tracking is a problem of common interest for researchers of numerous domains. The concept of target tracking classically emerged from the disciplines of control engineering. The concerned problem in the present context is to predict the trajectory of a moving target mainly soccer ball, soccer player by a given tracker. Usually the speed of response of the tracker is higher or at least comparable to that of the target. Since the intelligent target and the tracker both use the same level of technology, it is expected that their speed of response is more or less comparable. The Khepera IV employs a video camera to capture the images of the moving target mainly soccer ball for segmentation and localization of the target in its image. It then identifies the location of the target by a range finder and consequently plans a path towards the target by using the knowledge of the obstacle map in its workspace.

### 5.6.1 Schematic diagrams of tracking scheme

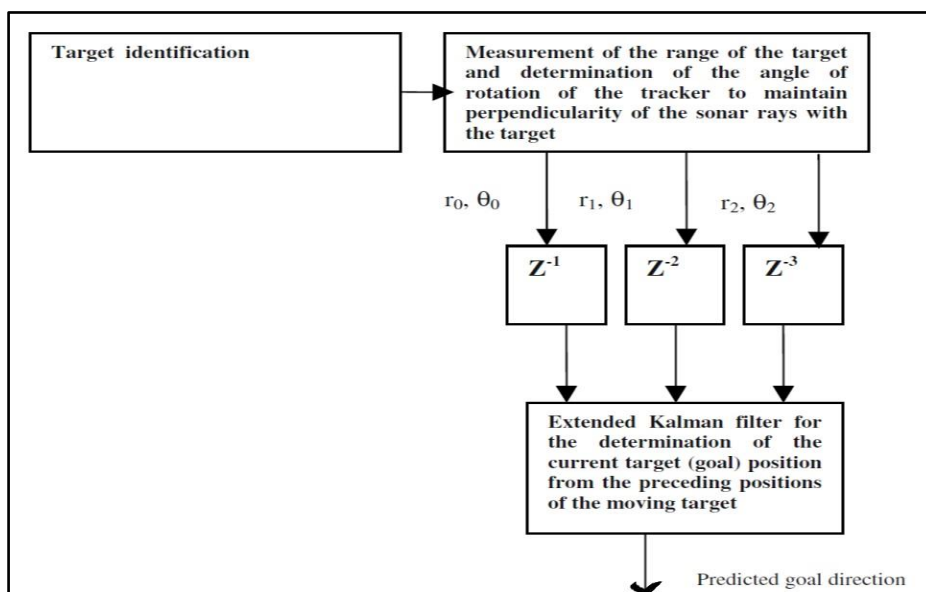
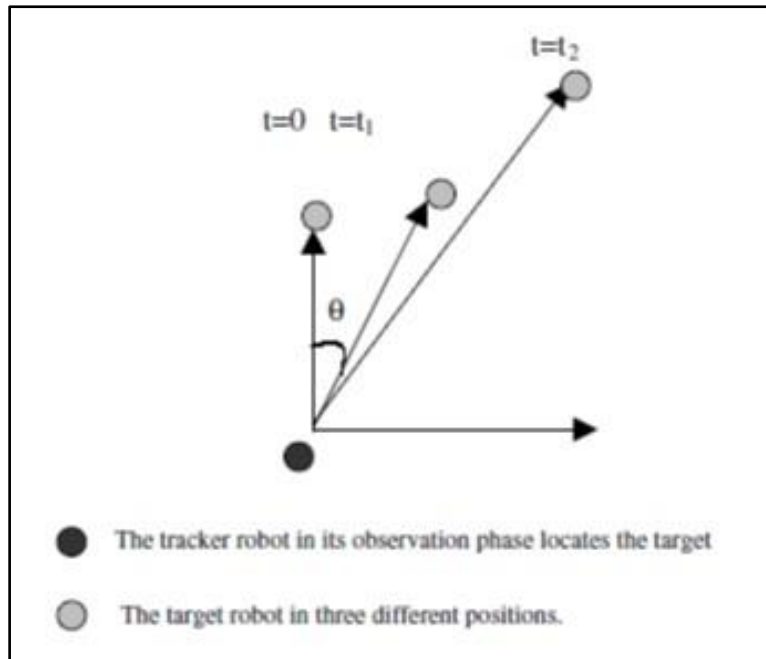


Figure 5.3 Schematic of tracking scheme

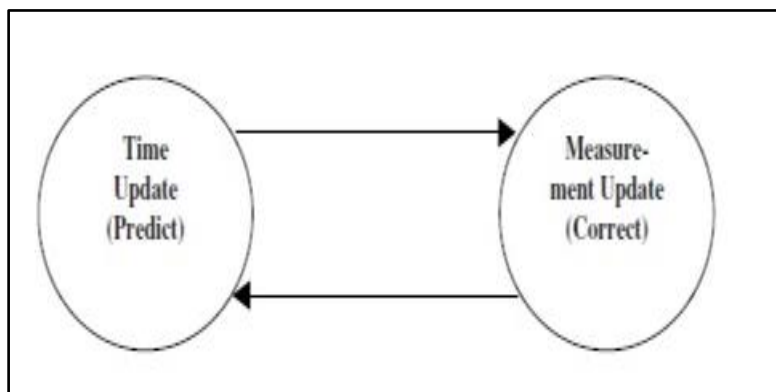
### 5.6.2 Measurements of the Input to Kalman Filter

The Kalman filter employed in the tracking system can predict the current position of the target from its preceding positions. The accuracy in prediction by a Kalman filter greatly depends on the time gap between successive data samples. Since the speed of the robots is considerably high (around 813mm/ second), the predicted current position of the target may suffer from inaccuracy. To overcome the above problem the tracker is designed to work in 2 phases. In the first phase, the tracker is given a controlled rotation around its z-axis so that it can direct its vision system to grab 3 successive frames of the target. Let the polar co-ordinates of these 3 positions be  $(r_0, \theta_0)$ ,  $(r_1, \theta_1)$ ,  $(r_2, \theta_2)$  respectively.



**Figure 5.4 The tracker observing the motion of the target**

Figure 5.4 describes 3 successive positions of the target in the measurement update phase of the tracker. The time  $t=0$ ,  $t=t_1$ ,  $t=t_2$  correspond to the image sampling times associated with these measurements. After the measurement update phase is over, the tracker switches to prediction phase. In the prediction phase, the tracker determines the current position of the target from its preceding positions  $(r_i, \theta_i)$  for  $i=0$  to 2. Once the prediction phase is over, the tracker starts its next measurement update phase, and the process continues until the tracker intercepts the target. A schematic view of a tracking cycle comprising of the measurement update phase and prediction phase is presented in figure 5.5.



**Figure 5.5 The recursive Kalman filter cycle**

## 5.7 Extended Kalman Filter

In the present section, the principle of extended Kalman filtering is just outlined to demonstrate its application in target tracking.

An extended Kalman filter [12] is a digital filter that attempts to minimize the measurement noise for estimating a set of unknown parameters, linearly related with a set of measurement variables. The most important significance of this filter is that it allows recursive formulation and thus improves accuracy of estimation up to users' desired level at the cost of new measurement inputs.

Let,

$\mathbf{f}_i(\mathbf{x}_i, \mathbf{a}) = \mathbf{0}$  be a set of measurement equations describing relationships among an estimator vector  $\mathbf{a}$  and measurement variable vector  $\mathbf{x}_i$ .

$\mathbf{x}_i^* = \mathbf{x}_i + \mathbf{I}_i$ , where  $\mathbf{I}_i$  is a white Gaussian type measurement noise such that  $\mathbf{E}[\mathbf{I}_i \mathbf{I}_j^T] = \mathbf{0}$  positive symmetric matrix  $\Lambda_i$ , and  $\mathbf{E}[\mathbf{I}_i \mathbf{I}_j^T] = \mathbf{0}$ ,

$\mathbf{a}_{i-1}^* = \mathbf{a} + \mathbf{s}_{i-1}$ , where  $\mathbf{s}_{i-1}$  is a gaussian type measurement noise such that  $\mathbf{E}[\mathbf{s}_{i-1}] = \mathbf{0}$ ,

is a white Gaussian type estimation noise.

$\mathbf{E}[\mathbf{s}_{i-1} \mathbf{s}_{j-1}^T] =$  positive symmetric matrix  $\mathbf{s}_{i-1}$ , and  $\mathbf{E}[\mathbf{s}_{i-1} \mathbf{s}_{j-1}^T] = \mathbf{0}$ .

Expanding  $f_i(\mathbf{x}_i, \mathbf{a})$  by Taylor's series around  $(\mathbf{x}_i^*, \mathbf{a}_{i-1}^*)$ , we find

$$\mathbf{y}_i = \mathbf{M}_i \mathbf{a} + \mathbf{w}_i \quad (5.21)$$

$$\mathbf{M}_i = \frac{\partial f_i}{\partial \mathbf{a}} \quad (5.22)$$

$$\mathbf{w}_i = \left( \frac{\partial f_i}{\partial \mathbf{x}} \right) (\mathbf{x}_i - \mathbf{x}_i^*) \quad (5.23)$$

$$\mathbf{W}_i = \mathbf{E}[\mathbf{w}_i \mathbf{w}_i^T] = \left( \frac{\partial f_i}{\partial \mathbf{x}} \right) \Lambda_i \left( \frac{\partial f_i}{\partial \mathbf{x}} \right)^T \quad (5.24)$$

$$\text{Let } \mathbf{S}_i = \mathbf{E}[(\mathbf{a}_i - \mathbf{a}_i^*)(\mathbf{a}_i - \mathbf{a}_i^*)^T] \quad (5.25)$$

An attempt to minimize  $S_i$  yields the filter equations [32], given by:

$$\mathbf{a}_i^* = \mathbf{a}_{i-1}^* + \mathbf{K}_i (\mathbf{y}_i - \mathbf{M}_i \mathbf{a}_{i-1}^*) \quad (5.26)$$

$$\mathbf{K}_i = \mathbf{S}_{i-1} \mathbf{M}_i^T (\mathbf{W}_i + \mathbf{M}_i \mathbf{S}_{i-1} \mathbf{M}_i^T)^{-1} \quad (5.27)$$

$$\mathbf{S}_i = (\mathbf{I} - \mathbf{K}_i \mathbf{M}_i) \mathbf{S}_{i-1} \quad (5.28)$$

Given  $\mathbf{S}_0$  and  $\mathbf{a}_0$ , the Kalman filter recursively updates  $\mathbf{a}_i$ ,  $\mathbf{K}_i$ ,  $\mathbf{S}_i$  until error covariance matrix  $\mathbf{S}_i$  becomes insignificantly small, or all the number of data points have been submitted. The  $\mathbf{a}_i$  thus obtained after termination of the algorithm is the estimated value of the parameters.

### 5.7.1 Predicting target position using Extended Kalman Filter

Let the input measurement vector  $x$  be given by

$$x = \begin{pmatrix} r \\ \theta \\ t \end{pmatrix} \quad (5.29)$$

Where  $r$  is the perpendicular distance of the target from the tracker at time  $t$ ,

$\theta$  is the angular displacement of the target measured with respect to the axis of sensor of the tracker, i.e.  $\theta$  is the angular shift of tracker with respect to the target from time  $t=0$  to the current time of observation,

$t$  is the time elapsed measured from the beginning of the observation phase

Let the measurement equations in the present context be

$$f_i = \begin{pmatrix} at^2 + bt + c - r \cos \theta \\ pt^2 + qt + s + r \sin \theta \end{pmatrix} = \mathbf{0} \quad (5.30)$$

where

$c$  and  $s$  are the initial displacements of the target with respect to sensor axis and its perpendicular direction,

$b$  and  $q$  are the velocity in the corresponding directions, and

$a$  and  $p$  are the time rate of change of velocity in the corresponding directions.

For determination of the current position of the target we need to evaluate  $a$ ,  $b$ ,  $c$ ,  $p$ ,  $q$ ,  $s$  from the measured  $(r, \theta)$ s at time  $t=0$ ,  $t=t_1$ , and  $t=t_2$ , respectively. The estimated values of  $a$ ,  $b$ ,  $c$ ,  $p$ ,  $q$ ,  $s$  then may be substituted in the measurement equations to evaluate  $r \cos \theta$  and  $r \sin \theta$  at time  $t \geq t_2$

The estimator in the present context thus is given by

$$a = \begin{pmatrix} a \\ b \\ c \\ p \\ q \\ s \end{pmatrix} \quad (5.31)$$

For the evaluation of the estimator we, however, need to determine the following derivatives:

$$\frac{\partial f_i}{\partial x} = \begin{pmatrix} -\cos \theta & r \sin \theta & 2at + b \\ -\sin \theta & -r \cos \theta & 2pt + q \end{pmatrix} \quad (5.32)$$

$$\frac{\partial f_i}{\partial a} = \begin{pmatrix} t^2 & t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & t^2 & t & 1 \end{pmatrix} \quad (5.33)$$

Let

$$\Lambda_i = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{pmatrix} \quad (5.34)$$

Where  $\alpha$  = the variance of the noise measurement of range r.

$\beta$  = the variance of the noise in measurement of angle  $\theta$ ,

and  $\gamma$  = the variance of the noise in measurement of time t.

## 5.8 Conclusion

In this chapter we have discussed about Kalman Filter and extended Kalman filter. Briefly described about the derivation and different variables which are used to derive the algorithm. The algorithm will be discussed in the later chapter. The historical background of Bayesian to recursive Bayesian how the Kalman filter had been came into existence. This chapter gave an idea on Kalman Filter what is its use and answer to how, why extended Kalman filter is required for non-linear systems.

## References

- [1] Reza Zekavat; R. Michael Buehrer, "An Introduction to Kalman Filtering Implementation for Localization and Tracking Applications," in *Handbook of Position Location: Theory, Practice, and Advances*, , IEEE, 2019, pp.143-195.
- [2] Jaiswal, R.K. & Jaidhar, C.D. *Wireless Netw* (2017) 23: 2021.
- [3] Tian Tian, Shuli Sun, Na Li, Multi-sensor information fusion estimators for stochastic uncertain systems with correlated noises, *Information Fusion*, Vol. 27, 2016, Pages 126-137, ISSN 1566-2535.
- [4] Lin Chai, "Neural network aided adaptive federal kalman filtering for intelligent integrated navigation application - [Not available for publication]," *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*, Salt Lake City, UT, USA, 2004, pp. 177-177.
- [5] A. Barhoumi, T. Ladhari and F. M'sahli, "The extended Kalman filter for nonlinear system: Application to system of waste water treatment," *2014 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, Hammamet, 2014, pp. 445-450.
- [6] N. Y. Ko and S. Jeong, "Fused Navigation of Unmanned Surface Vehicle and Detection of GPS Abnormality," *Journal of Institute of Control, Robotics and Systems*, vol. 22, no. 9, pp. 723–732, Sep. 2016.
- [7] R. Faragher, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation," *IEEE SIGNAL PROCESSING MAGAZINE*, vol. 29, no. 5, pp. 128-132, 2012.
- [8] S. Riisgaard and M. R. Blas, "SLAM for Dummies (A Tutorial Approach to Simultaneous Localization and Mapping)," 2005.
- [9] MATLAB, "Understanding Kalman Filters, Part 5: Nonlinear State Estimators," Youtube, 17 May 2017. [Online]. Available: <https://youtu.be/Vefia3JMeHE>. [Accessed 21 May 2018].
- [10] Extended Kalman filtering and weighted least squares dynamic identification of robot, *Control Engineering Practice*, Vol. 9, Issue 12, December 2001, Pages 1361-1372.
- [11] J. Heikkonen and E. Oja, "Self-organizing maps for visually guided collision-free navigation," *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, Nagoya, Japan, 1993, pp. 669-672 vol.1.
- [12] *Artificial Vision for Mobile Robots Stereo Vision and Multisensory Perception* by Nicholas Ayache
- [13] Jeffreys, Harold (1973). *Scientific Inference* (3rd ed.). Cambridge University Press. p. 31. ISBN 978-0-521-18078-8.
- [14] Chen, Zhe. (2003). *Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond*. Statistics. 182. 10.1080/02331880309257.

# Chapter 6

## Object Recognition

### 6.1 Introduction

Object recognition [1-6] is a vital concept that comprises of the abilities of robots to recognize various things and entities while performing a particular task. This technique aids to pick out and identify objects from inputs like video and still camera images. Robots thus can understand their environments better and can perform more complex tasks. Methods used for object identification include 3D models, component identification, edge detection and analysis of appearances from different angles. Object recognition is at the convergence points of robotics, machine vision, neural networks and AI [7-12].

### 6.2 Edge Detection

Edges [13-15] play an important role in many image processing applications. They provide an outline of the object. In the physical plane, edges correspond to the discontinuities in depth, surface orientation, change in material properties, and light variations. These variations are present in the image as gray scale discontinuities. An edge is a set of connected pixels that lies on the boundary between two regions that differ in gray value. The pixels on an edge are called edge points. Most edges are unique in space that is, their position and orientation remain the same in space when viewed from different points. When an edge is detected the unnecessary details are removed, while only the important structural information is retained. An edge is typically extracted by computing the derivative of the image function. This consists of two parts magnitude of the derivative, which is an indication of the strength or contrast of the edge, and the direction of the derivative vector, which is a measure of edge orientation. Some of the edges that are encountered in image processing are as follows:

- a) **Step edge:** It is described as an abrupt intensity change.
- b) **Ramp edge:** It is described by a gradual change in intensity.
- c) **Spike edge:** This edge represents a quick change and immediately returns to the original intensity.
- d) **Roof edge:** This edge is not instantaneous over a short distance.

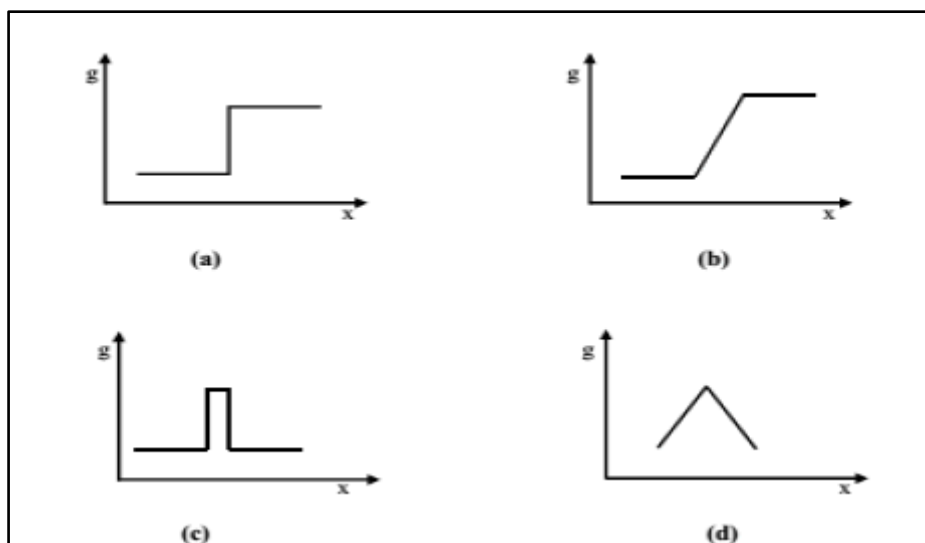


Figure 6.1 Types of edges- a) Step edge, b) Ramp edge, c) Spike edge, d) Roof edge



### 6.2.1 Stages of Edge Detection

The basic idea of edge detection is to detect the sharp edges in image brightness, which can capture the important events and properties. This is done in three stages as described below:

#### A. Filtering

It is better to filter the input image to get maximum performance for the edge detectors. This stage may be performed either explicitly or implicitly. It involves smoothing, where the noise is suppressed without affecting the true edges. In addition, this phase uses a filter to enhance the quality of the edges in the image. Normally, Gaussian filters are used as they are proven to be very effective for real-time images,

#### B. Differentiation

This phase distinguishes the edges pixels from other pixels. The idea of edge detection is to find the difference between two neighbourhood pixels. If the pixels have the same value, the difference is 0. This means that there are no transitions between the pixels. The non-zero difference indicates the presence of an edge point. A point is defined as an edge point (or edge) if its derivative is greater than the user-specified threshold and encounters a sign change in the derivative. The first derivative is  $\frac{\partial f}{\partial x} = \lim_{\Delta x \rightarrow 0} \frac{f(x) - f(x - \Delta x)}{\Delta x}$ , images are discrete. Hence,  $\Delta x$  should be discrete and should be at least 1. Therefore, the gradient vector is (called *grad of x*) should be equal to  $f(x) - f(x-1)$ , if the intensities are same, the derivative is 0. The non-zero element indicates the presence of images. In the case of the second derivatives, the zero crossings indicate the presence of edges.

Images are two dimensional. Hence, the gradient vector of  $f(x,y)$  is also two dimensional. The gradient vector of  $f(x,y)$  at location  $(x,y)$  is a vector that consists of the partial derivatives of  $f(x,y)$  as follows:

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \quad (6.1)$$

$$\nabla f(x, y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \quad (6.2)$$

The magnitude of this vector, generally referred to as the gradient  $\nabla f$ , is

$$\nabla f(x, y) = \text{mag}(\nabla f(x, y)) = \left[ (g_x)^2 + (g_y)^2 \right]^{1/2} \quad (6.3)$$

Edge strength is indicated by the edge magnitude. The direction of the gradient vector is useful in detecting a sudden change in image intensity. The common practice is to approximate the gradient with absolute values that are simpler to implement, as follows:

$$\nabla f(x, y) = |g_x| + |g_y| \quad (6.4)$$

$$\nabla f(x, y) = \max(g_x, g_y) \quad (6.5)$$

The gradient direction can be given as:

$$\theta = \tan^{-1} \left( \frac{g_y}{g_x} \right) \quad (6.6)$$

### C. Localization

In this stage the detected edges are localized. The localization process involves determining the object location of the edge. In addition, this stage involves edge linking and edge thinning steps to measure that edge is sharp and connected. The sharp and connected edges are then displayed.

The prerequisite of localization stage is normalization of the gradient magnitude. The calculated gradient can be scaled to be very specific range say, 0-K by performing this operation. For example, the value of the constant K may be an integer, say 100.  $N(x,y)$  is called the normalized edge image and is given by:

$$N(x,y) = \frac{G(x,y)*K}{\max_{i=1 \text{ to } n, j=1 \text{ to } n} G(i,j)} \quad (6.7)$$

The normalized magnitude can be compared with a threshold value T to generate the edge map.

The edge map is given as:

$$E(x,y) = \begin{cases} 1 & \text{if } N(x,y) > 1 \\ 0 & \text{otherwise} \end{cases} \quad (6.8)$$

The edge map is the displayed or stored for further image processing operations.

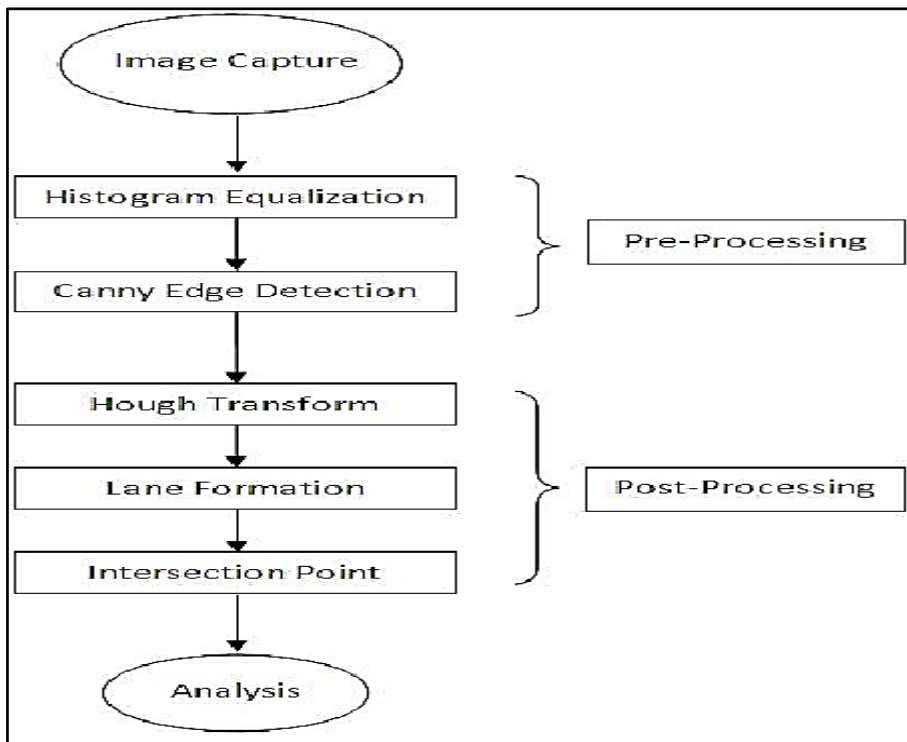


Figure 6.2 Edge detection Process

## 6.2.2 Types of Edge Detectors

The edge detection is implemented in all kinds of edge detectors. In image processing, four types of edge detection operators are available which are:

1. **Derivative filters:** These filters use the differentiation technique to detect the edges.
2. **Template matching filters:** These filters use templates that resemble the target shapes and match with the image. Gradient operations are isotropic in nature as they detect edges in all directions. Hence, template matching filters are used to perform directional smoothing as they are very sensitive to directions. If there is a match between the target shape or directions and the masks, then a maximum gradient value is produced. By rotating the template sensitive in all eight directions, masks, are produced. Point detection and line detection masks are good examples of template matching filters.
3. **Gauss derivative filters:** They are very effective for real time images and are used along with the derivative filters.
4. **Patten fit approach:** This is another approach where a surface is considered as a topographic surface, with the pixel value representing altitude. The aim is to fit a pattern over a neighbourhood of a pixel where the edge strength is calculated. The properties of the edge points are calculated based on the parameters.

## 6.2.3 First Order Edge Detection Operators

Local transitions among different image intensities constitute an edge. Therefore, the aim is measuring the intensity gradients. Hence, edge detectors can be viewed as gradient calculators. Based on the differential geometry and vector calculus, the gradient operator is represented as:

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \quad (6.9)$$

Applying this to the image  $f$ , one gets:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (6.10)$$

In differences between the pixels are quantified by the gradient magnitude. The direction of the greatest change is given by the gradient vector. This gives the directions of the edge. Since the gradient functions are continuous functions, the discrete versions of the continuous functions can be used. This can be done by finding the differences. The approaches in 1D are as follows. Here.  $\nabla_x$  and  $\nabla_y$  are the movements in  $x$  and  $y$  directions respectively.

$$\text{Backward difference} = \frac{f(x) - f(x - \Delta x)}{\Delta x} \quad (6.11)$$

$$\text{Forward difference} = \frac{f(x + \Delta x) - f(x)}{\Delta x} \quad (6.12)$$

$$\text{Central difference} = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \quad (6.13)$$

The differences can be obtained by applying the following masks, assuming  $\Delta x = 1$ .

$$\text{Backward differences} = f(x) - f(x-1) = [1-1] \quad (6.14)$$

$$\text{Forward difference} = f(x+1) - f(x) = [-1+1] \quad (6.15)$$

$$\text{Central difference} = \frac{1}{2} * [10-1] \quad (6.16)$$

These differences can be extended to 2D as  $g_x = \frac{\partial f}{\partial x}$  and  $g_y = \frac{\partial f}{\partial y}$ . Then the magnitude is given by:

$$\nabla f(x, y) = \text{mag}(\nabla f(x, y)) = \left[ (g_x)^2 + (g_y)^2 \right]^{1/2} \quad (6.17)$$

The gradient direction is given by:

$$\theta = \tan^{-1} \left( \frac{g_y}{g_x} \right) \quad (6.18)$$

Now let us discuss most important first order edge detection operators, that is the Sobel operator. The Sobel operator also relies on central difference. This can be viewed as an approximation of the first Gaussian derivative. This is equivalent to the first derivative of the Gaussian blurring image obtained by applying 3 x 3 mask to image. Convolution is both commutative and associative, and is given by:

$$\frac{\partial}{\partial x} (f * G) = f * \frac{\partial}{\partial x} G \quad (6.19)$$

A 3 x 3 digital approximation of the Sobel operator is given as:

$$\nabla f = |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)| \quad (6.20)$$

The masks are as follows:

$$M_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (6.21)$$

$$M_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (6.22)$$

### 6.2.4 Second Order Edge Detection Operators

Edges are considered to be present in the first derivative when the edge magnitude is large compared to threshold value. In the case of the second derivative, the edge pixel is present at a location where the second derivative is zero. This is equivalent to saying that  $f''(x)$  has a zero crossing which can be observed as a sign change in pixel differences. The Laplacian algorithm is one such zero-crossing algorithm. However, the problems of the zero-crossing algorithms are many, the problem with laplacian masks is that they are sensitive to noise as there is no magnitude checking- even a small ripple causes the method to generate as an edge point. Therefore, it is necessary to filter the image before the edge detection process is applied. This method produces two-pixel thick edges, although generally, one pixel thick edges are preferred. However, the advantage is that there is no need for the edge thinning process as the zero-crossings themselves specify the location of the edge points. The main advantage is these operators are rotationally invariant.

The second order derivative is given as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (6.23)$$

This  $\nabla^2$  operator is called the Laplacian operator. The Laplacian of the 2D function  $f(x,y)$  is also defined as:

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \quad (6.24)$$

### 6.2.5 Edge Linking Algorithms

The detector often does not produce continuous edges. Often, the detected edges are not sharp continuous presence of noise and intensity variations. Therefore, the idea of edge [16] is to use the magnitude of the gradient operator to detect the presence of edges and connect it to a neighbourhood to avoid breaks. Continuity is ensured by techniques such as variation thresholding and edge relaxation. The adjacent pixels  $(x,y)$  and  $(x',y')$  are connected as they have properties such as the following:

1. Similar gradient magnitude

$$| \|\nabla f(x,y)\| - \|\nabla f(x',y')\| | \leq T \quad (6.25)$$

2. Similar gradient orientation

$$| \varphi(\nabla f(x, y)) - \varphi(\nabla f(x', y')) | \leq A \quad (6.26)$$

Thus, A is the angular threshold. Edge linking is a post-processing technique that is used to link edges. The idea of using detection algorithm to extract the edges and using as appropriate threshold for combining them is known as edge elements extraction by thresholding. The threshold selection can be static, dynamic or adaptive.

### 6.2.6 Hough Transform

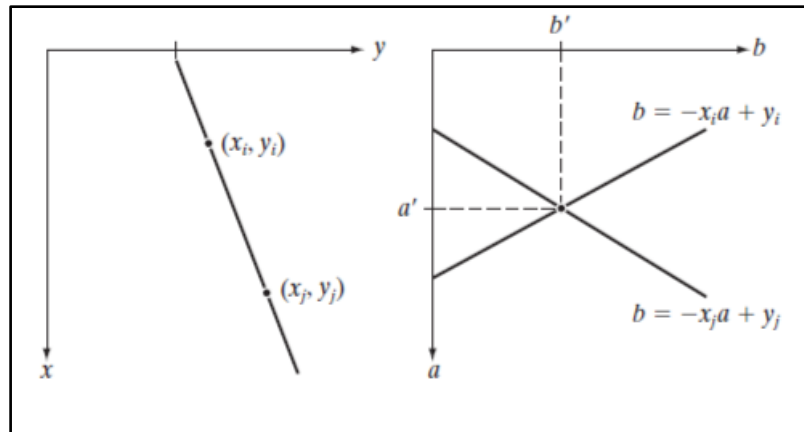
The Hough transform [17-20] takes the images created by the edge detection operators. Most of the edge map generated by the edge detection algorithms is disconnected. The Hough transform can be used to connect the disjoint edge points. Unlike the edge linking algorithms, the transform does not require any prerequisites to connect the edge points. It is used to fit the points as plane curves. The plane curves typically are lines, circles and parabolas. The line equation is given as:

$$y = mx + c \quad (6.27)$$

Where, m is the slope and c is the y-intercept of the line. However, the problem is that there are lines that can be drawn connecting these points. Therefore, an edge point in an x - y should be transformed into a different c - m plane. The trick here is to write the line equation as:  $c = (-x) m + b$ . All the edge points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  in the x - y needs to be fitted. Hence, the x - y plane should be transformed into a different c - m points are lines in the c - m plane. The objective is to find the intersection point. A common intersection points indicates that edge points are part of the same line. If A and B are connected by a line in the spatial domain, then they will be intersecting lines in the Hough transform. This is illustrated in the figure 6.3. To check whether they are intersecting lines, the c-m plane is, portioned as accumulator lines. To find this, it can be assumed that the c-m plane can be partitioned as an accumulator array. At the end of this process, the accumulator values are checked. The largest

value of the accumulator is called  $(m', c')$ . The significance is that this point gives us the line equation of the  $(x, y)$  space:

$$y = m'x + c' \quad (6.28)$$



**Figure 6.3 Lines in Hough transform**

The Hough transform can be stated as follows:

1. Load the image
2. Find the edges of the image using any edge detector
3. Quantize the parameter space P
4. Repeat the following for all pixels of the image.

If the pixel is an edge pixel, then

$$(a) \quad c = (-x)m + y$$

$$(b) \quad P(c,m) = P(c,m) + 1$$

5. Show the Hough space.
6. Find the local maxima in the parameter space
7. Draw the line using the local maxima.

The major drawback with the algorithm is that it does not work for vertical line, as they have a slope of infinity. Therefore, it is better to convert this line into polar coordinates. The line in polar form can be represented as  $\rho = x \cos \theta + y \sin \theta$ , where  $\theta$  is the angle between the line and the x-axis, and  $\rho$  is the diameter.

The modified hough transform for the polar coordinate line is as follows:

- i. load the image.
- ii. Find the edges of the image using any edge detector
- iii. Quantize the parameter space P.
- iv. Repeat the following for all pixels of the image:
  - if the pixel is an edge pixel, then for all  $\theta$
  - a) Calculate  $\rho$  for the pixel (x,y) and  $\theta$
  - b) Increment the position ( $\rho, \theta$ ) in the accumulator array P.
- v. Show the Hough space
- vi. Find the local maxima in the parameter space.
- vii. Draw the line using the local maxima

Similarly, the Hough transform for the other edges can also be found. The Hough transform for circle detection can be given as follows:

$$(x - x_o)^2 - (y - y_o)^2 - r^2 = 0 \quad (6.29)$$

The parameter space is three dimensional as the there are three unknowns. So, the polar form circle detection algorithm can be written as follows:

1. Load the image
2. Find the edges of the image using any edge detector.
3. Quantize the parameter space P
4. Repeat the following for all pixels of the image:
  - If the pixel is a edge pixel, then for all values of r, calculate
  - a)  $x_o = x - r \cos \theta$
  - b)  $y_o = y - r \sin \theta$
  - c)  $P(x_o, y_o, r) = P(x_o, y_o, r) + 1$
5. Show the Hough space.
6. Find the local maxima in the parameter space.
7. Draw the line using the local maxima
8. Draw the circle using the local maxima

In this manner, a circle can be fit to cover the edge points. The Hough transform is a powerful algorithm and can be modified to fit any generalized shape having disjoint edge points.

### 6.3 Image Morphology

Morphological image processing [21-22] is a collection of non-linear operations related to the shape or morphology of features in an image. Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all the possible locations in the image and it is compared with the corresponding neighbourhood of pixels. Some operations test whether the element "fits" within the neighbourhood, while others test whether it "hits" or intersects the neighbourhood.

### 6.3.1 Dilation

Dilation can be applied to binary as well as grey-scale images. The basic effect of this operator on a binary image is that it gradually increases the boundaries of the region, while the small holes present in the images become smaller. In short, dilation can be considered as a union operation of all translations of the image A caused by the elements specified in the structuring element B.

$$A \oplus B = \bigcup_{b \in B} (A)_b \quad (6.30)$$

Where, b is any point that belongs to image B. As the dilation operator is commutative, hence we can write:

$$B \oplus a = \bigcup_{a \in A} (B)_a \quad (6.31)$$

Here, a is any point corresponding to the image A. This operation is also known as Minkowski addition. Dilation increases the size of the object and also thickens the object.

### 6.3.2 Erosion

Erosion is another important morphological operator. This is also known as Minkowski subtraction. The objective of this operator is to make an object smaller by removing its outer layer of pixels. If a black pixel, has a white neighbour, then all the pixels are made white. This can be described mathematically.

$$A \ominus B = \bigcap A_B \quad (6.32)$$

The erosion takes the image and the structuring element as inputs and thins the object.

### 6.3.3 Opening

The opening operation is an erosion operation followed by dilation operation. This operation can be defined as:

$$A \circ B = \bigcap \{B_w; B_w \subseteq A\} \quad (6.33)$$

Opening is useful for smoothing the edges, breaking the narrow joints and thinning the protrusions that are present in an image.

### 6.3.4 Closing

This operation is defined as dilation followed by erosion operation. This is mathematically denoted as:

$$A \bullet B = (A \oplus B) \ominus B \quad (6.34)$$

Closing is useful for attaching the narrow joints and thickening the protrusions that are present in an image.

## 6.4 Region Descriptors

The region descriptors [23-24] include descriptions that characterize the object. These include approaches such as histogram, geometrical, topological and structural features. We shall now specifically discuss geometrical or shape descriptors in details.



### 6.4.1 Shape Features

Shape is one of the most important features of an object. It is difficult to describe a shape as most of the real-world objects do not exhibit standard geometrical shapes. Hence we correlate unknown shapes with known objects, for example, how circular or spherical an object is, to convey the shape information. A shape is the external or visual appearance of an object which is characterized by boundaries. Patterns such as textures characterize its surface. There is a subtle difference between object shapes and image shapes. An object shape is independent of position, orientation, and scale. This idea can be extended to image shapes if independence of grey level is taken into account.

Shape region can be described by many parameters. Objects can exhibit various shapes. Shape region feature (or geometric features) characterize the appearance of an object. The common shape region features are listed as follows:

#### A. Area

The area of an object refers to all pixels inside the object. This is shift invariant, but not size invariant. The area of a binary image is defined as the number of pixels in the binary image  $B$  and is given as follows:

$$A = \sum_{i=1}^n \sum_{j=1}^m B(i, j) \quad (6.35)$$

Here,  $B(i, j)$  is the binary image. The number of pixels or the sum of the pixels is the same and is the area of the binary image.

#### B. Perimeter

The perimeter of an object is the number of pixels present on the boundary of the object. In a binary image, the perimeter is the number of foreground pixels that touches the background in the image. Perimeter can also be defined as the length of the path through the boundary pixels. Perimeter is shift invariant and rotation invariant, but is not size invariant.

#### C. Shape Factor

This is also known as compactness and is given as:

$$Compactness = \frac{(Perimeter)^2}{Area} \quad (6.36)$$

#### D. Area to Perimeter Ratio

This ratio is a metric that indicates the roundness, circularity, or thinness ratio. This metric indicates how close an object is to a circle. It is a dimensionless quantity and ranges between 0 and 1. If its value is 1, then the object is a perfect circle. This measure is insensitive to scaling transformations.

#### E. Major Axis Length

The longest line that can be drawn through the object connecting the two farthest points in the boundary is called its major axis. If the major axis end points are  $(x_1, y_1)$  and  $(x_2, y_2)$  then the length of the object is the major axis length and can be calculated as:

$$major\ axis\ length = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (6.37)$$

The orientation of the object can be obtained using the angle between major axis and x-axis of the image and is described as:

$$\mathbf{orientation} = \mathbf{tan}^{-1} \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad (6.38)$$

### **F. Minor Axis Length**

The minor axis basically provides the object width. Minor axis is the longest line that can be drawn through the object while maintaining perpendicular with the major axis. This is calculated using the formula:

$$\mathbf{minor\ axis\ length} = \sqrt{(x'_2 - x'_1)^2 + (y'_2 - y'_1)^2} \quad (6.39)$$

Where,  $(x'_1, x'_2)$  and  $(y'_1, y'_2)$  are the end points of the minor axis.

The ratio of the minor axis width and the major axis width length is called object elongation. The value of this parameter is 1 when the object is roughly square or circle.

### **G. Bounding Box**

Bounding Box [25] is defined as the smallest rectangle that can encompass an object. The computation of a minimum bounding rectangle is a difficult task, but approximations can be used. There are two types of boundary boxes. One is called the Feret box and another is called the minimum bounding rectangle.

The area of the box that completely surrounds an object is called then bounding box area which is calculated by:

$$\mathbf{bounding\ box\ area} = \mathbf{major\ axis\ length} * \mathbf{minor\ axis\ length} \quad (6.40)$$

## **6.5 Conclusion**

This chapter summarizes the important methodologies required for object recognition. Minute descriptions of these techniques along with elementary mathematical operations illustrated helps to grasp these mythologies much easily. Moreover, these methods provide useful applications while working with real time images captured by robots.

## References

- [1] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, New York, 1972.
- [2] F. Arman and J. K. Aggarwal, "CAD-based vision: object recognition in cluttered is using recognition strategies", *Computer Vision, Graphics and Image using recognition strategies*, Computer vision, Graphics and Image Processing, vol.58, no. 1, pp. 33-48, 1993.
- [3] D. Huber and M. Hebert. "3D Modeling Using a Statistical Sensor Model and Stochastic search,". *Proc. IEEE Int 'I Conf Computer Vision and Pattern Recognition*, pp. 858-865,2003.
- [4] R Donamukkala,D. Huber, A. Kapuria, and M. Hebert, "Automatic Class Selection and styping for 3-D Object Classification," *Proc. Int'l Conf 3-D Digital Imaging and Modeling*, pp. 64-71, 2005.
- [5] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection", *ICCV*, 2009.
- [6] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, "Latent hierarchical structural learning for object detection", *CVPR*, 2010,
- [7] A. Frome, D. Huber, and R. Kolluri, "Recognizing objects in range data using regional point descriptors," *ECCV*, vol. 1, pp. 1-14, 2004.
- [8] X. Ren, L. Bo, and D. Fox, "RGB-(D) scene labeling: Features and algorithms," *CVPR*, 2012.
- [9] H. Jegou, M. Douze, and C. Schmid., "Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search", *European Conference on Computer Vision*, pp. 304–317, Marseille, 2008.
- [10] D Nister and H Stewenius, "Scalable Recognition with a Vocabulary Tree", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 2161-2168. *IEEE*, 2006.
- [11] Zheng Song, Qiang Chen, Zhongyang Huang, Yang Hua, and Shuicheng Yan, "Contextualizing object detection and classification", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1585-1592. *IEEE*, June 2011.
- [12] Zhihui Du, Weigiang Yang, Yinong Chen, Xin Sun, Xiaoying Wang, and Chen Xu,"Design of a Robot Cloud Center", *International Symposium on Autonomous decentralized Systems* pp. 269–275, March 2011.
- [13] Lina Shi, Tu Feng, Zhu Hong, "A Method for the Detection of Moving Targets in Complex Background", *Electronic Engineer*, pp. 45-47, Jan 2006.
- [14] Rafael C. Gonzalez Richard E. Woods, "Digital Image Processing (Second Edition)" in, *Beijing Publishing House of Electronics Industry*, Mar 2005.
- [15] F A Pellegrino. "Edge Detection Revisited," *IEEE Trans On System Man and Cybernetics*, vol.34.no.3 pp. 1500-1517, 2004.

- [16] Lin Q, Han Y. Han H, "Real-time lane departure detection based on extended edge Having algorithm," Second International Conference Computer Research and Development, IEEE, 2010, pp. 725-730.
- [17] D H. Ballard, "Generalizing the hough transform to detect arbitrary shapes", Pattern Recognition, 13(2):111-122, 1981.
- [18] F. M. G. da Costa, L. da F. Costa, "Straight line detection as an optimization problem an approach motivated by the jumping spider visual system", Biologically Motivated Computer Vision, Germany, Berlin:Springer, vol. 1811, pp. 32-41, 2000,
- [19] J. Song, M. R. Lyu, "A Hough transform based line recognition method utilizing both parameter space and image space", Pattern Recognit., vol. 38, no. 4, pp. 539-552, 2005.
- [20] L. A. F. Fernandes, M. M. Oliveira, "Real-time line detection through an improved Hough transform voting scheme", Pattern Recognit., vol. 41, no. 1, pp. 299-314, 2008,
- [21] A. Pardo., "Semantic image segmentation using morphological tools", Proc. of IEEE Conf. on Image Proc., pp. 745-748, 2002.
- [22] X. Huang, L. Zhang, "A multidirectional and multiscale morphological index for automatic building extraction from multispectral GeoEye-1 imagery", Photogramm. Eng. Remote Sens., vol. 77, no. 7, pp. 721-732, Jul. 2011.
- [23] S. Belongie, J. Malik, J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 2, no. 4, pp. 509-522, Apr. 2002.
- [24] A. Opelt, M. Fussenegger, A. Pinz, P. Auer, "Weak Hypotheses and Boosting for Generic Object Detection and Recognition", Proc. Eighth European Conf. Computer Vision, pp. 71-84, 2004.
- [25] V. Lempitsky, P. Kohli. C. Rother, T. Sharp, "Image segmentation with a bounding box prior", MSR-TR-2009-85.

# Chapter 7

## Implementation of Soccer Goal Keeper

### 7.1 Introduction

The chapter provides the entire technical details regarding the implementation of soccer goal keeper using the Khepera IV mobile robot [1-3]. All steps are described vividly to have a clear understanding of the designing procedure. MATLAB code and the function for extended kalman filter [24-25] developed to implement the goal keeper prediction of range and position information. This chapter contains step by step information on how the whole prediction is done right from the beginning of taking the image by khepera to prediction using MATLAB function.

### 7.2 Methodologies Utilized for Implementation

The steps involved in the implementation of the soccer robot goal keeper are discussed in this section. Moreover, the techniques for such an implementation are also illustrated in this segment.

#### 7.2.1 An overview of the proposed scheme for prediction

**Step 1.** 2 mobile robots identical in all respects have been configured as the target (for kicking the ball) and the tracker.

**Step 2.** The target robot is controlled to move on a fixed trajectory by a control program running at its desktop server to kick the ball at different instance.

**Step 3.** The tracker robot on the other hand receives sensory information by using the video camera and the ultrasonic sensors.

**Step 4.** The received real time video frames collected by the tracker robot are first transferred to its server.

**Step 5.** The server pre-processes the image and segments it into objects of interest (here the ball).

**Step 6.** The individual images are segmented and subsequent localization of each image for the ball is done.

**Step 7.** After the target is identified in the image, its distance and orientation with respect to some reference axis of the tracker needs to be determined.

**Step 8.** After distance and orientation is measured from each image then the values were passed to the extended Kalman filter function.

**Step 9.** It processes the values and it updates the estimation vector till a pre-condition is met.

**Step 10.** It then gives the next distance and angle values by processing the matlab function for extended kalman filter.

## 7.2.2 Ball Recognition

### A. Ball Recognition using MATLAB function

The step begins by capturing an image through the camera by TCP/IP protocol [4-5], socket communication and processing it through the matlab BallDetection function.

Image processing in MATLAB for ball recognition has the following steps:

1. The image captured, read and converted to gray scale.
2. The contrast of this image is enhanced using the contrast-limited adaptive histogram equalization (CLAHE) [6-15]. The development and necessity of this method is described below:

Under certain scenarios the grayscale distribution is highly localized, then it becomes a difficult task to transform very low-contrast images by full histogram equalization. In these cases, the mapping curve might be mapped to significantly different grayscales. This issue is then resolved by limiting approach with the adaptive Histogram Equalization results in Contrast Limited Adaptive Equalization (CLAHE). Thus, this method enhances the contrast of the image much more efficiently as compared to histogram equalization and adaptive histogram equalization techniques.

3. Finally, the ball is detected by the user of Circular Hough Transform [16-20].

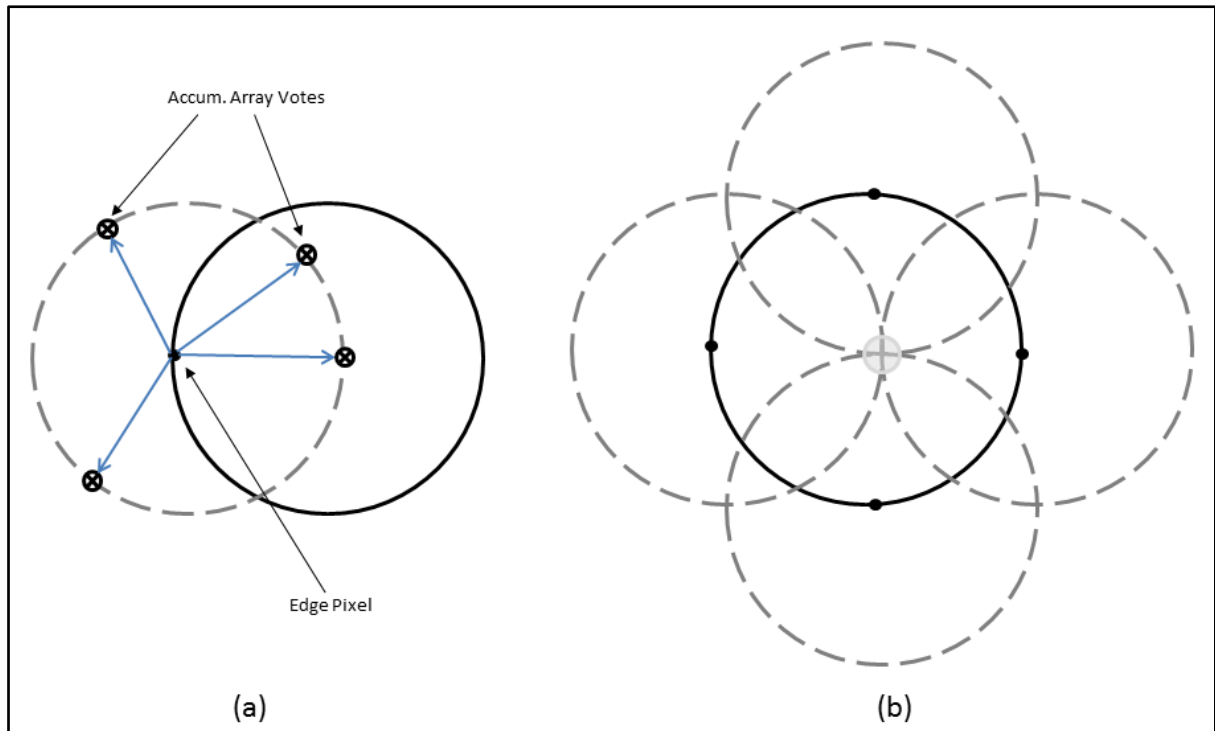
### B. Algorithm used in Ball Recognition

*imfindcircles* uses a Circular Hough Transform (CHT) based algorithm for finding circles in images. This approach is used because of its robustness in the presence of noise, occlusion and varying illumination.

The CHT is not a rigorously specified algorithm, rather there are a number of different approaches that can be taken in its implementation. However, by and large, there are three essential steps which are common to all.

#### I. Accumulator Array Computation

Foreground pixels of high gradient are designated as being candidate pixels and are allowed to cast 'votes' in the accumulator array. In a classical CHT implementation, the candidate pixels vote in pattern around them that forms a full circle of a fixed radius. Figure 7.1 (a) shows an example of a candidate pixel lying on an actual circle (solid circle) and the classical CHT voting pattern (dashed circles) for the candidate pixel.



**Figure 7.1 Classical CHT Voting Pattern**

## II. Center Estimation

The votes of candidate pixels belonging to an image circle tend to accumulate at the accumulator array bin corresponding to the circle's center. Therefore, the circle centers are estimated by detecting the peaks in the accumulator array. Figure 7.1 b shows an example of the candidate pixels (solid dots) lying on an actual circle (solid circle), and their voting patterns (dashed circles) which coincide at the center of the actual circle.

## III. Radius Estimation

If the same accumulator array is used for more than one radius value, as is commonly done in CHT algorithms, radii of the detected circles have to be estimated as a separate step.

`imfindcircles` provides two algorithms for finding circles in images: Phase-Coding (default) and Two-Stage.

The common computational features shared by both algorithms are as follow:

### a) Use of 2-D Accumulator Array

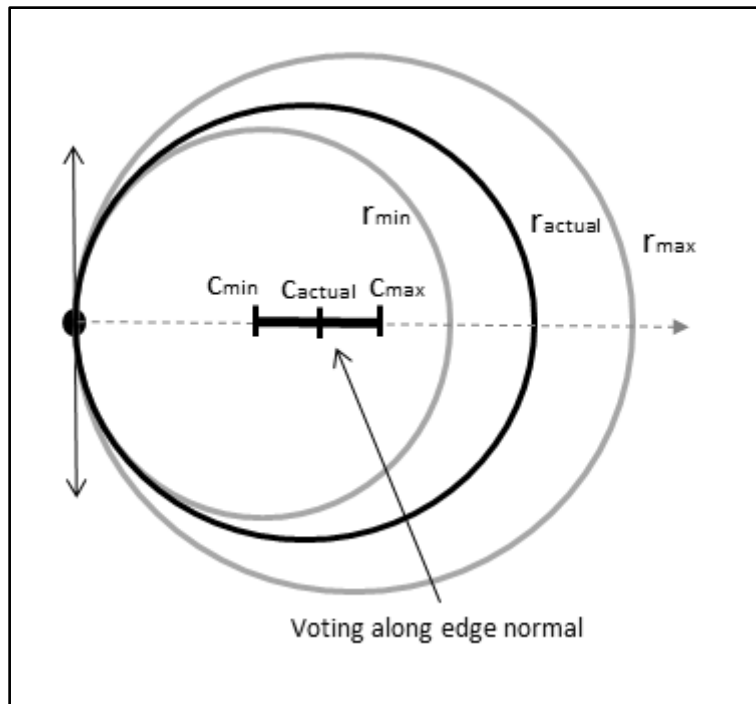
The classical Hough Transform requires a 3-D array for storing votes for multiple radii, which results in large storage requirements and long processing times. Both the Phase-Coding and Two-Stage methods solve this problem by using a single 2-D accumulator array for all the radii. Although this approach requires an additional step of radius estimation, the overall computational load is typically lower, especially when working over large radius range. This is a widely adopted practice in modern CHT implementations.

**b) Use of Edge Pixels**

Overall memory requirements and speed is strongly governed by the number of candidate pixels. To limit their number, the gradient magnitude of the input image is threshold so that only pixels of high gradient are included in tallying votes.

**c) Use of Edge Orientation Information**

Another way to optimize performance is to restrict the number of bins available to candidate pixels. This is accomplished by utilizing locally available edge information to only permit voting in a limited interval along direction of the gradient (Figure 2).



**Figure 7.2 Voting Mode: Multiple Radii, Along Direction of Gradient**

$r_{min}$	Minimum search radius
$r_{max}$	Maximum search radius
$r_{actual}$	Radius of the circle that the candidate pixel belongs to
$C_{min}$	Center of the circle of radius $r_{min}$
$C_{max}$	Center of the circle of radius $r_{max}$
$C_{actual}$	Center of the circle of radius $r_{actual}$



The two CHT methods employed by function `imfindcircles` fundamentally differ in the manner by which the circle radii are computed.

- **Two-Stage**

Radii are explicitly estimated utilizing the estimated circle centers along with image information. The technique is based on computing radial histograms [21-22].

- **Phase-Coding**

The key idea in Phase Coding [23] is the use of complex values in the accumulator array with the radius information encoded in the phase of the array entries. The votes cast by the edge pixels contain information not only about the possible center locations but also about the radius of the circle associated with the center location. Unlike the Two-Stage method where radius has to be estimated explicitly using radial histograms, in Phase Coding the radius can be estimated by simply decoding the phase information from the estimated center location in the accumulator array.

### C. Distance and Angle Calculation

After the recognition of the ball, it is important to know its distance from the robot and its angle of orientation with respect to the robot. These factors are calculated by the following formulae:

$$distance = \frac{f*dr*t}{o*s} \quad (7.1)$$

Where

**f** is the focal length of the robot's camera in mm

**r** is the real height of the ball

**i** is the image height in pixels

**o** is the object height within the image in pixels.

**s** is the sensor height of robot's camera in mm

The angle of inclination of the ball with respect to the robot can be calculated by the following formulae:

$$a' = \left(\frac{192}{2}\right) - y \quad (7.2)$$

$$b' = 144 - x \quad (7.3)$$

$$angle = \tan^{-1} \frac{a'}{b'} \quad (7.4)$$

### 7.2.3 Algorithm used in MATLAB EKF function

#### Function EKF (r<sub>0</sub>, θ<sub>0</sub>, r<sub>1</sub>, θ<sub>1</sub>, r<sub>2</sub>, θ<sub>2</sub>)

#### Begin

##### 1. Initialize

- a)  $W_0 = \left(\frac{\partial f_i}{\partial x}\right) \Lambda_i \left(\frac{\partial f_i}{\partial x}\right)^T$  where  $\Lambda_0$  and  $\left(\frac{\partial f_0}{\partial x}\right)$  are available from expression (5.32) and (5.34).
- b)  $S_0$  as a diagonal matrix with large positive diagonal values.
- c)  $a_0$  to be zero;
- d)  $M_0 = \left(\frac{\partial f_0}{\partial a}\right)$  vide expression (5.33)
- e) Loop iteration index  $i:=1$ .

##### 2. Repeat

- a) Input new measurement  $x_i$  and evaluate  $y_i$  by expression (5.21)
- b) Update  $K_i$ ,  $a_i$ ,  $S_i$  in order using expression (5.27), (5.26) and (5.28);

Until  $\text{abs}(a_i - a_{i-1}) < \text{a pre-defined threshold}$ .

##### 3. Determine

- a)  $at^2 + bt + c$  at time  $t=t_3 > t_2$
- b)  $pt^2 + qt + s$  at time  $t=t_3 > t_2$

For known  $a, b, c, p, q$ , and  $s$ ;

#### End

where

**c** and **s** are the initial displacements of the target with respect to sensor axis and its perpendicular direction,

**b** and **q** are the velocity in the corresponding directions, and

**a** and **p** are the time rate of change of velocity in the corresponding directions.

In the third step of **determine** we evaluate the values of  $r\cos\theta$  and  $r\sin\theta$  at time  $t \geq t_2$ .

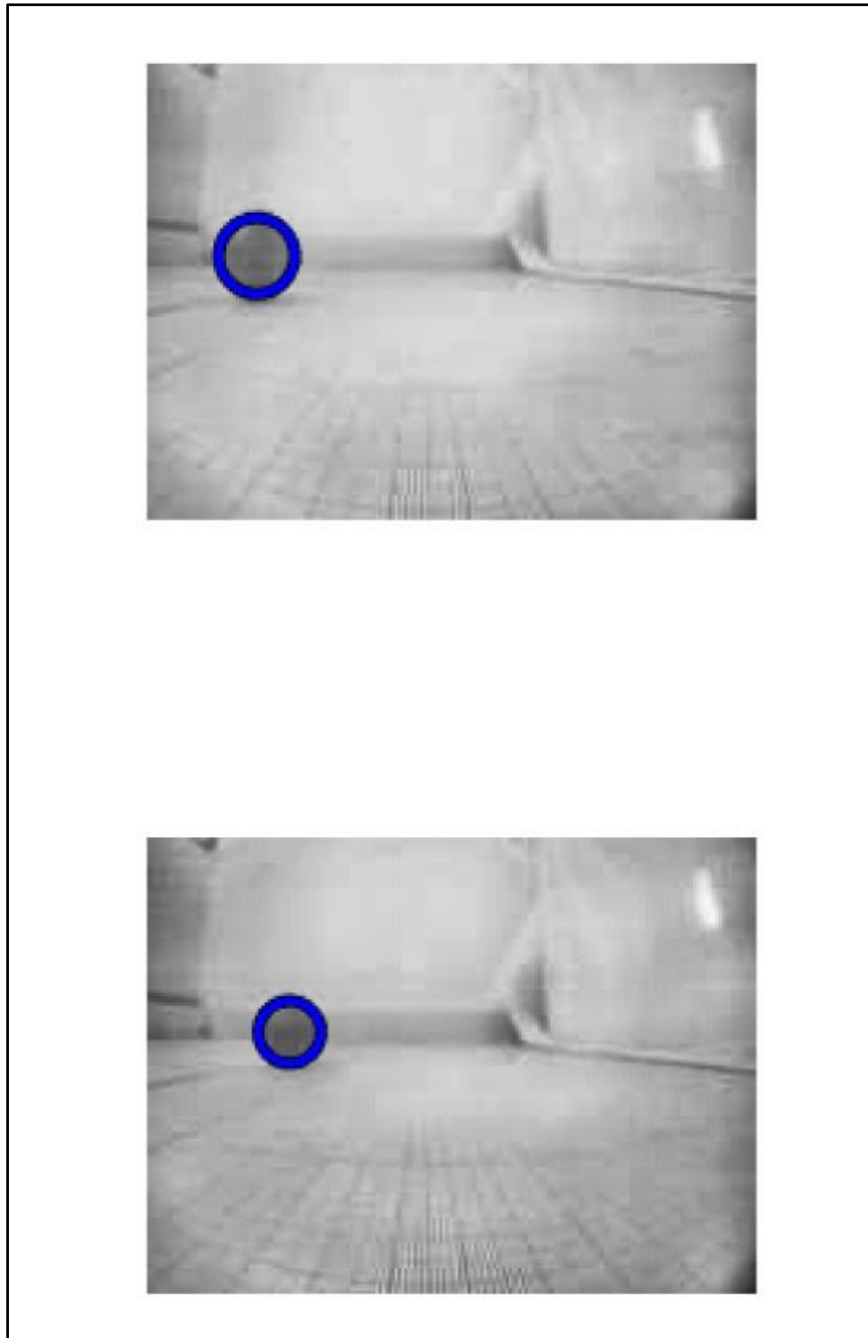
And to measure the predicted angle use the following equation:

$$\text{Angle} = \text{pred\_x}(2,1) + \text{atan}(Y(i)/X(i)) - 5$$

Where **pred\_x** = Predicted value for input measurement vector.

**Y** and **X** are calculated from the above algorithm last step i.e determine state.

### 7.3 Results



**Figure 7.3 Ball detection**

## 7.4 Output from EKF function written in MATLAB

**The predicted range is:**

Y =

-99.7812 521.4616 384.2512

**The actual range is:**

r =

330 364 379

**The predicted angle is:**

A =

24.8854 19.7528 13.5505

**The actual angle is:**

theta =

31 24 18

## 7.5 Outcome

As seen above the predicted distance is:

**384.2512**

Actual distances for all the images taken by KHEPERA IV:

124.001047096430 151.048302441459 173.840297877355 209.565561441147  
272.021877183786 330.815923238659 364.344160717200  
379.851978457323 388.476689097816

In our case actual distance calculated from image by BallDetection function:

**388.4766**

The predicted angle is:

**13.5505**

Actual Angle measured by BallDetection function for all images:

3.10574440131613 20.4197152223547 28.9818659013446 34.1996160268671  
36.3529028768892 31.6438363165663 24.8482487455750  
18.5362652409631 13.3859260960704

In our case actual angle calculated for the instant will be:

**13.3859**

So, it is seen the values predicted are near but not accurate completely.

## 7.6 Conclusion

This chapter provided the entire technical details regarding the design of soccer goal keeper using the Khepera IV mobile robot. All steps are described vividly to have a clear understanding of the designing procedure and the predicted values from matlab code is given which were obtained from function developed to implement the goal keeper prediction code which predicted the distance and angle information.

## Reference

- [1] L. Jetto, S. Longhi and G. Venturini, "Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots," in *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 219-229, April 1999.
- [2] F. Chenavier and J. L. Crowley, "Position estimation for a mobile robot using vision and odometry," *Proceedings 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992, pp. 2588-2593 vol.3.
- [3] J. Lu, C. Li and W. Su, "Extended Kalman filter based localization for a mobile robot team," *2016 Chinese Control and Decision Conference (CCDC)*, Yinchuan, 2016, pp. 939-944.
- [4] J. Broenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Trans. Robot. Autom.*, vol. 12 no. 6, pp. 869-880, Dec. 1996.
- [5] N. Doh, H. Choset, and W. K. Chung, "Accurate relative localization using odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, 2003, pp. 1606-1612.
- [6] W. Z. W. Ismail, and K. S. Sim, "Contrast enhancement dynamic histogram equalization for medical image processing application," *2011 Wiley Periodicals*, vol. 21, pp. 280-289, 2011.
- [7] J.A. Stark and W.J. Fitzgerald, "An Alternative Algorithm for Adaptive Histogram Equalization," *Graphical Models and Image Processing*, vol.56, pp.180-185, 1996.
- [8] Stephen M. Pizer, E. Philip Amburn and John D. Austin, "Adaptive Histogram Equalization and Its Variations," *Computer vision, Graphic, And Image Processing*, Vol. 39. pp. 355-368, 1987.
- [9] Agung W. Setiawan, Tati R. Mengko, and Oerip S. Santoso, " Color Retinal Image Enhancement using CLAHE," *ICISS International Conference*, pp. 1-3, 2013.
- [10] N. Otsu, "A threshold selection method from graylevel histogram," *IEEE Trans on Systems, Man, and Cybernetics*, Vol. 9, No. 1, pp. 6266, 1979.
- [11] Seung Jong Kim, Byong Seok Min and Dong Kyun Lim, " Determining Parameters in Contrast Limited Adaptive Histogram Equalization, ", *SERSC International Conference*, Vol. 21, pp 204-207, 2013.
- [12] K. Zuiderveld, "Contrast limited adaptive histogram equalization," *Graphics Gems IV*, pp. 474-485, 1994
- [13] Joung-Youn Kim, Kim, Lee-Sup Kim, and Seung-Ho Hwang, "An Advanced Contrast ement Using Partially Overlapped SubBlock Histogram Equalization".
- [14]K. Gu, G. Zhai. W. Lin, M. Liu, "The analysis of image contrast: From quality nt to automatic enhancement", *Cybernetics IEEE Transactions*, vol. PP, no. 99, pp. 1-1, 2015

- [15] A. Reza, "Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement", *Journal of VLSI signal processing systems for signal image and video technology*, vol. 38, no. 1, pp. 35-44, 2004.
- [16] S. Abelazeem, M. Hafez, G. Auda, "Using circular Hough transform for detecting microaneurysms in fluorescein angiograms of the ocular fundus", *International Conference on Industrial Electronics*, 19–21 December 2001.
- [17]. J. K. Pedersen, "Circular Hough Transform," Aalborg University, *Vision, Graphics and Interactive Systems*, November 2007.
- [18] V. F. Leavers, "Which Hough transform?", *CVGIP: Image Understanding*, vol. 58, pp. 250-264, 1993.
- [19] S. Abdelazeem, "Microaneurysm detection using vessels removal and circular Hough transform", *IEEE Proc. 19th Natl. Radio Sci. Conf.*, vol. 1-2, pp. 421-426, 2002-Mar.
- [20] M. Soltany, S.T. Zadeh, H.R. Pourreza, "Fast and Accurate Pupil Positioning Algorithm using Circular Hough Transform and Gray Projection", *2011 Int. Conf on Computer Communication and Management Proc. Of CSIT*, vol. 5, pp. 556-561, 2011.
- [21] H.K Yuen, .J. Princen, J. Illingworth, and J. Kittler. "Comparative study of Hough transform methods for circle finding." *Image and Vision Computing*. Volume 8, Number 1, 1990, pp. 71–77.
- [22] E.R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*. Chapter 10. 3rd Edition. Morgan Kauffman Publishers, 2005.
- [23] T.J Atherton, D.J. Kerbyson. "Size invariant circle detection." *Image and Vision Computing*. Volume 17, Number 11, 1999, pp. 795-803.
- [24] *Computational Intelligence Principles, Techniques and Applications* Prof. Dr. Amit Konar.
- [25] *Learning the Extended Kalman Filter version 1.0.0.0* by Yi Cao.

# Chapter 8

## Conclusion and Future Direction

### 8.1 Summary of the work

The most important aspect depicted by this thesis work is the design of a soccer player using the recently developed Khepera IV mobile robot. The fundamental motivation behind this research is to develop prediction of the next state of the target from its few preceding positions. The work vividly describes the ball recognition part mathematical aspect algorithms which are used for more complex environments. Then the values obtained from each image captured by KHEPERA are processed to calculate distance and angle. Later these values are passed to extended Kalman filter to predict distance and angle of the next state, for which the mathematical derivation, various definition, historical aspect of Kalman filter and extended Kalman Filter various matrix partial derivatives used for developing the algorithm of extended kalman filter are discussed in different chapters. The MATLAB function for the filter has been written in such a way that it can be understood by a novice user. Each line of the code has comments to make more sense it is divided in initialization, prediction and update steps. Though I had passed the values manually from the values obtained from the ball detection in set of three to reduce complexity. The code checks a pre-defined condition after every loop iteration and if it is met then it updates the different arrays and matrix used for prediction otherwise it comes out of the loop. This work acts as an excellent test bed for the prediction and interception part of goal keeper design of the soccer robotics.

Chapter 1 gives an introductory concept regarding the differential drive robots, their categories kinematical modelling. It then provides a short literature review regarding the development of robotics and the types of robots used previously. Then the main objective of this thesis work is discussed.

Chapter 2 describes the entire Khepera IV robot in great details. It comprises of its specifications, functionalities and the working principle of its various components. This chapter acts a useful guide that aids in understanding the basic components and the functions of Khepera IV robot.

Chapter 3 comprises of the entire set up part required to communicate with the robot. This chapter provides in depth description of how to install all the necessary directives, libraries etc. Then it discusses how the communication pathway between the robot and the computer can be built using either USB cable or Wi-Fi. It then provides the detailed description of the functions and variables used for programming. And finally, it provides the basic steps required for writing the program and executing the codes.

Chapter 4 describes the essence of socket communication. In this chapter the protocol used for interfacing is discussed in great details. Then the steps required for creating a client and server are discussed for two software platforms i.e. C and MATLAB. Basic codes required for such implementation are also included in this chapter.



Chapter 5 describes mainly the concept of our thesis i.e. Kalman and Extended Kalman filter its historical aspect all the mathematical steps required for extended kalman filter derivation, state variable, probability, random values and noise.

Chapter 6 provides the algorithms of object recognition. Various techniques utilized for object recognition. Mathematical details of circular Hough transform.

Chapter 7 it is main chapter of the thesis work that comprises of the design of soccer robot player using KHEPERA IV. Algorithm used to implement Extended Kalman Filter in MATLAB. Mathematical details of ball recognition and extended kalman filter techniques are vigorously described calculation of distance and angle of next state are also shown. Finally, the output of ball detection function and predicted values for distance and angle from extended Kalman filter are shown and compared.

Thus, this thesis work has been able to accomplish and fulfil the desired objectives and has provided important results that are extremely useful for prediction of next states.

## **8.2 Future Scope**

Though the thesis work has provided vital results that are extremely useful, however, there exists an immense scope for improvement. Some of the future directives are listed below:

- The Real time implementation of prediction can be implemented to make more use of the filter.
- Static obstacles can be placed within the field along with the ball. This will produce a greater challenge for the soccer robot to find the ball.
- Make fellow soccer players which will approach ball efficiently by avoiding the obstacles.
- Develop Back Propagation Neural Net for prediction.
- Develop player interception and ball detection simultaneously for more complex task.
- Two teams consisting of a combination of Khepera robots can be modelled along with two goal keepers. This problem would require various complex actions like kicking the ball to the opposite goal, passing the ball to its fellow mate and other complex reasoning and decision-making tasks.
- 3D reconstruction from multiple 2D images of the robot work space by using Kalman Filter.

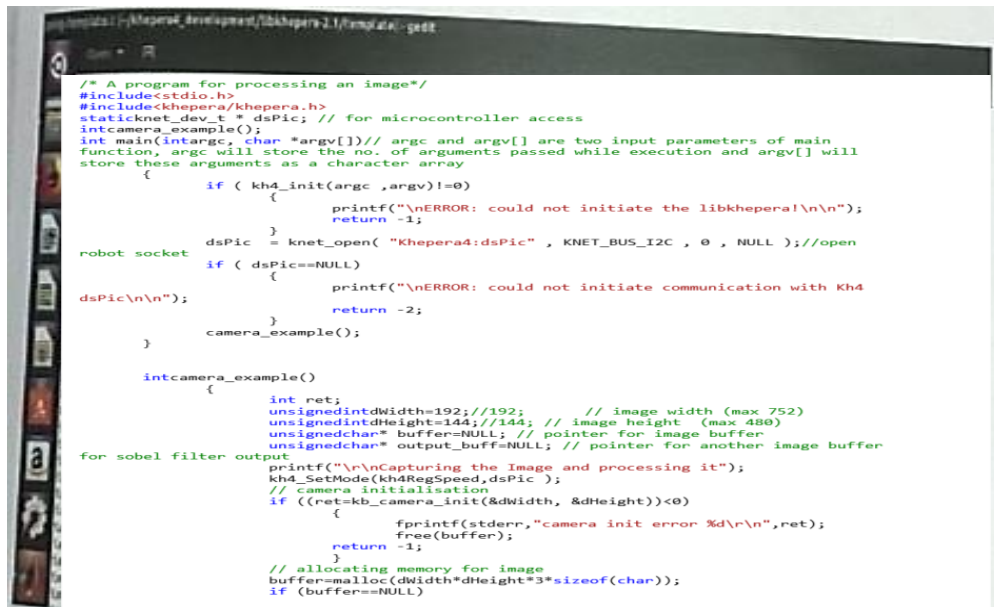
# APPENDIX

## User guide to run various scripts, codes and function of MATLAB

The section provides step wise guidance for the execution of the codes given in the accompanying CD at the end of the thesis.

### Step 1:

Write the C code for processing image using soccer robot in the text file within the template folder as shown below and save it as **prog-template.c**.



```
/* A program for processing an image*/
#include<stdio.h>
#include<khepera/khepera.h>
staticknet_dev_t * dsPic; // for microcontroller access
intcamera_example();
int main(intargc, char *argv[])// argc and argv[] are two input parameters of main
function, argc will store the no. of arguments passed while execution and argv[] will
store these arguments as a character array
{
    if ( kh4_init(argc ,argv)!=0)
    {
        printf("\nERROR: could not initiate the libkhepera!\n\n");
        return -1;
    }
    dsPic = knet_open( "Khepera4:dsPic" , KNET_BUS_I2C , 0 , NULL );//open
robot socket
    if ( dsPic==NULL)
    {
        printf("\nERROR: could not initiate communication with Kh4
dsPic\n\n");
        return -2;
    }
    camera_example();
}

intcamera_example()
{
    int ret;
    unsignedintdWidth=192;//192; // image width (max 752)
    unsignedintdHeight=144;//144; // image height (max 480)
    unsignedchar* buffer=NULL; // pointer for image buffer
    unsignedchar* output_buff=NULL; // pointer for another image buffer
for sobel filter output
    printf("\r\nCapturing the Image and processing it");
    kh4_SetMode(kh4RegSpeed,dsPic );
    // camera initialisation
    if ((ret=kb_camera_init(&dWidth, &dHeight))<0)
    {
        fprintf(stderr,"camera init error %d\r\n",ret);
        free(buffer);
        return -1;
    }
    // allocating memory for image
    buffer=malloc(dWidth*dHeight*3*sizeof(char));
    if (buffer==NULL)
```

A.1 Screen depicting the C code for Soccer Robot

## Step 2:

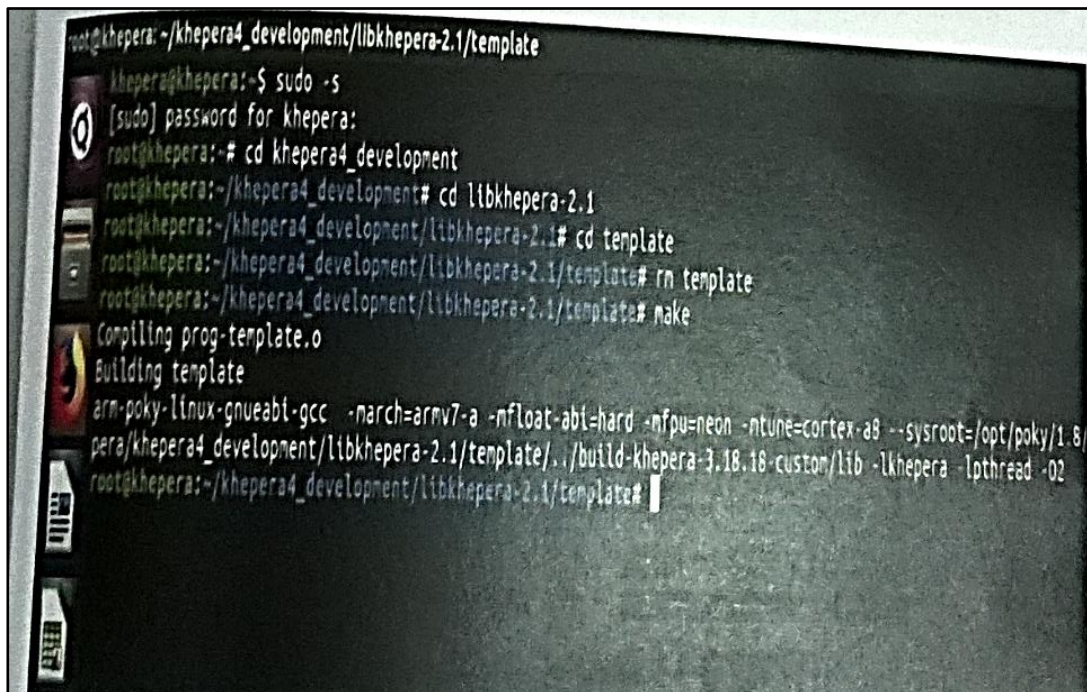
Open a new terminal and complete the code by first entering into the path where the program has been written by using the following commands:

```
cd khepera4_development
```

```
cd libkhepera-2.1
```

```
cd template
```

Then remove the previously written code by using **rm template** and compile the present code by **make** command. If the compilation is successful then message with no error will be shown as depicted below:

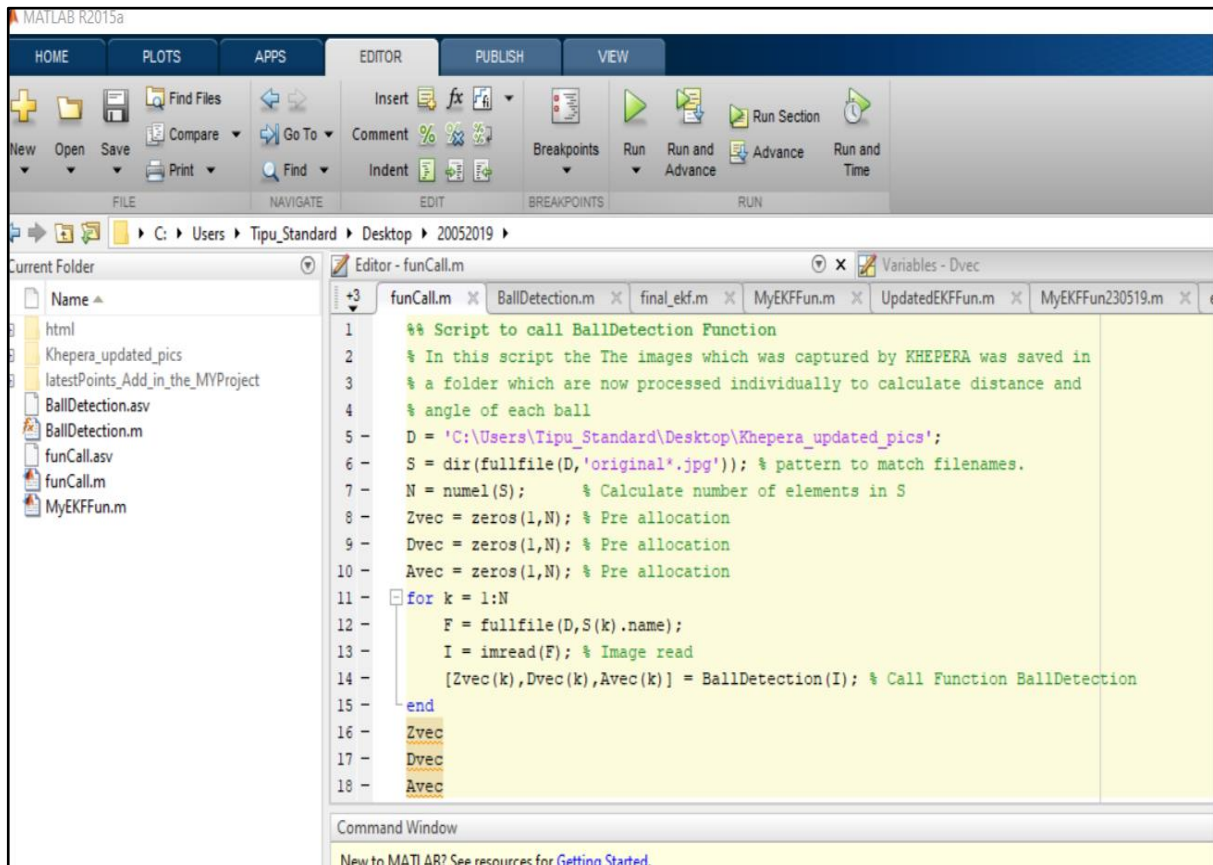


```
root@khepera: ~/khepera4_development/libkhepera-2.1/template
khepera@khepera:~$ sudo -s
[sudo] password for khepera:
root@khepera:~# cd khepera4_development
root@khepera:~/khepera4_development# cd libkhepera-2.1
root@khepera:~/khepera4_development/libkhepera-2.1# cd template
root@khepera:~/khepera4_development/libkhepera-2.1/template# rm template
root@khepera:~/khepera4_development/libkhepera-2.1/template# make
Compiling prog-template.o
Building template
arm-poky-linux-gnueabi-gcc -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a8 --sysroot=/opt/poky/1.8/
pera/khepera4_development/libkhepera-2.1/template/./build-khepera-3.18.18-custom/lib -lkhepera -lpthread -O2
root@khepera:~/khepera4_development/libkhepera-2.1/template#
```

## A.2 Screen for C program compilation

### Step 3:

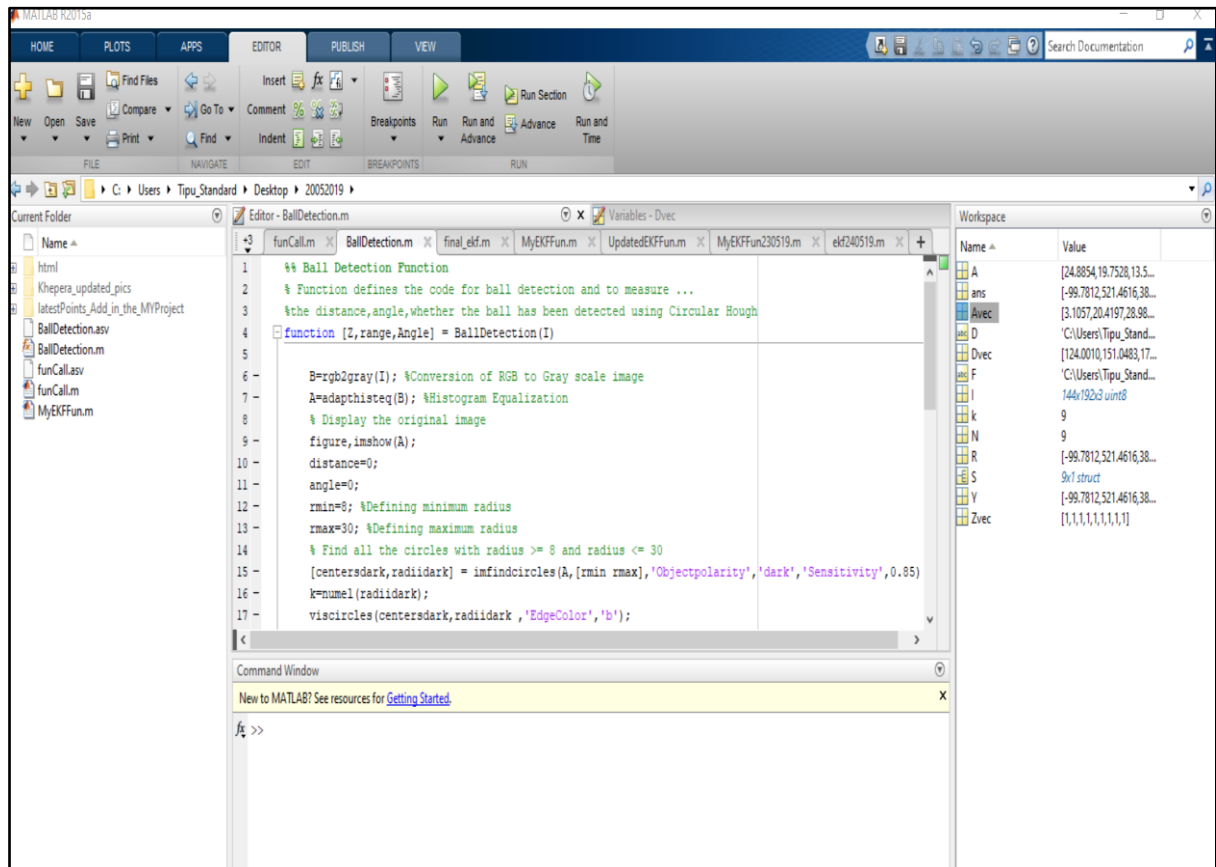
Write the code in script for function calling which calls BallDetection function in MATLAB.



**A.3 Screen showing script which calls BallDetection function**

#### Step 4:

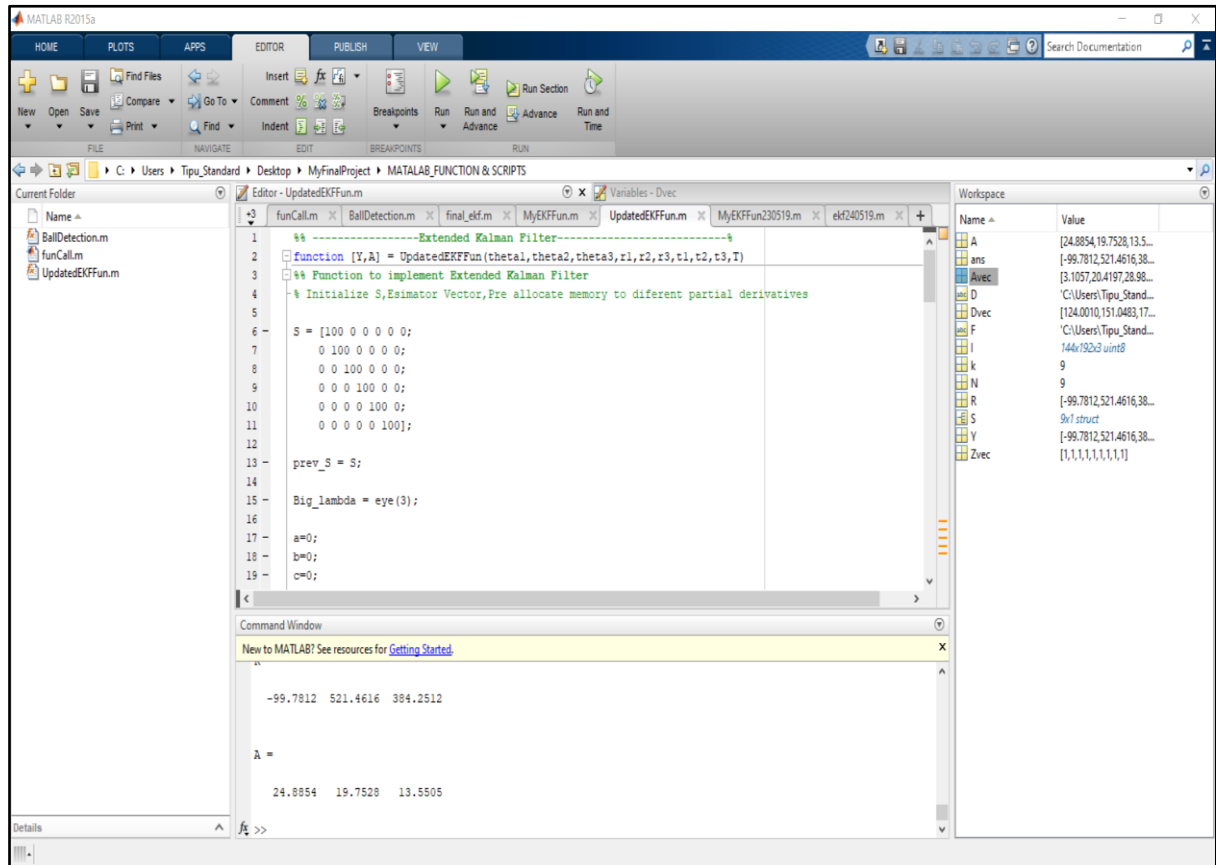
The BallDetection function screen is shown below which calculates distance and angle values stores them in Avec and Dvec for each image that is taken by the KHEPERA IV.



A.4 Screen showing script which shows the BallDetection Function.

## Step 5:

Write the code for Extended Kalman Filter the screen below shows the EKF filter to predict the distance and angle of the next state.



```
1 %% -----Extended Kalman Filter-----  
2 function [Y,A] = UpdatedEKFFun(theta1,theta2,theta3,r1,t2,r3,t1,t2,t3,T)  
3 %% Function to implement Extended Kalman Filter  
4 % Initialize S,Estimator Vector,Pre allocate memory to diferent partial derivatives  
5  
6 S = [100 0 0 0 0 0;  
7 0 100 0 0 0 0;  
8 0 0 100 0 0 0;  
9 0 0 0 100 0 0;  
10 0 0 0 0 100 0;  
11 0 0 0 0 0 100];  
12  
13 prev_S = S;  
14  
15 Big_lambda = eye(3);  
16  
17 a=0;  
18 b=0;  
19 c=0;
```

Workspace

Name	Value
A	[24.8854, 19.7528, 13.5...
ans	[-99.7812, 521.4616, 38...
D	[3.1057, 20.4197, 28.98...
Dvec	'C:\Users\Tipu_Stand...
F	[124.0010, 151.0483, 17...
I	146/1923 uint8
k	9
N	9
R	[-99.7812, 521.4616, 38...
S	9x1 struct
Y	[-99.7812, 521.4616, 38...
Zvec	[1,1,1,1,1,1,1]

Command Window

```
New to MATLAB? See resources for Getting Started.  
--99.7812 521.4616 384.2512  
  
A =  
  
24.8854 19.7528 13.5505
```

A.5 Screen showing extended kalman filter.