GESTURE-BASED GRAPH-THEORETIC ROBOT FORMATION CONTROL

by

Mahroo Sajid

A Thesis presented to the Faculty of the
American University of Sharjah
College of Engineering
In Partial Fulfilment
of the Requirements
for the Degree of

Master of Science in
Mechatronics Engineering

Sharjah, United Arab Emirates

June 2022

Declaration of Authorship

I declare that this thesis is my own work and, to the best of my knowledge and belief, it does not contain material published or written by a third party, except where permission has been obtained and/or appropriately cited through full and accurate referencing.

Signed Mahroo Sajid

Date June 15, 2022

Approval Signatures

We, the undersigned, approve the Master's Thesis of Mahroo Sajid.

Thesis Title: Gesture-based Graph-theoretic Robot Formation Control

Date of Defense: May 17, 2022

Name, Title and Affiliation **Signature**

Dr. Shayok Mukhopadhyay
Associate Professor, Department of Electrical Engineering
Thesis Advisor

Dr. Mohammad Jaradat
Professor, Department of Mechanical Engineering
Thesis Committee Member

Dr. Oualid Hammi
Professor, Department of Electrical Engineering
Thesis Committee Member

Dr. Shayok Mukhopadhyay
Program Coordinator
Mechatronics Engineering Graduate Program

Dr. Lotfi Romdhane
Associate Dean for Graduate Affairs and Research
College of Engineering

Dr. Fadi Aloul
Dean
College of Engineering

Dr. Mohamed El-Tarhuni
Vice Provost for Research and Graduate Studies
Office of Research and Graduate Studies

Acknowledgements

I would like to thank my advisor Dr. Shayok Mukhopadhyay for his guidance and infallible support. I am grateful to the respected committee members for their valuable insights and feedback. Last and certainly not the least, I am indebted to the esteemed professors of the Mechanical and Electrical Engineering departments at the College of Engineering for adding to my knowledge and honing my skills, and the American University of Sharjah Graduate Teaching Assistantship Program for affording the opportunity to accomplish this degree.

Dedication

To my parents, sisters, Qasim,

Rehan, Safa and Roshanay

Abstract

Multi-robot systems have many diverse applications including but not limited to surveillance, search and rescue operations, hazardous material handling, agriculture etc. Formation control is a key requirement out of the many facets of robotic swarm control. While the formation control problem has been extensively explored, including vastly different and various approaches; consensus-based formation control has definitive advantages in terms of providing a robust and distributed control algorithm. In the proposed work, the notion of consensus-based control of the formations of a robotic network utilizing bare-handed gestures is explored. While similar examples can be found in recent research endeavors (including selection and command of groups of UAVs, UGVs and under-water robots), most utilize electronic or specialized devices for the purpose of Human-Swarm Interaction (HSI). This is despite the major breakthroughs in the field of image recognition and classification. The proposed gesture-recognition approach is based on an algorithm capable of discerning the finger count (number of open digits) in 2.5 seconds (on average) with 88.75% accuracy thus affording the user a vocabulary of up to five (5) individual commands / directives to assemble and control the robotic swarm. The gesture recognition algorithm was modified to increase the detection accuracy to 90.8%.

Keywords: Human-Swarm Interaction; Formation Control; Hand-Gesture recognition.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
BLOB	Binary Large Object
COG	Centre of Gravity
HMI	Human-Machine Interaction
HSI	Human-Swarm Interaction
IMU	Inertial Measurement Unit
ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

Chapter 1. Introduction

1.1. Introduction

The topic of formation control of multi-robot systems or swarms has been explored in depth in earlier research. This is due to the utility of formation control in a broad range of applications involving robotic agents: including surveillance, search and rescue operations, exploration, conducting hazardous operations etc. The methodology for achieving formation control being employed in the proposed work is based on the concept of "consensus". In networks of agents, "consensus" means to reach agreement with respect to a certain quantity of interest.

Significant advances have been made in the field of image processing, image recognition and specifically bare-hand gesture recognition. Until late 1990s, the researchers had to face problems with image recognition due to insufficient processing power of earlier machines and poor-quality cameras producing noisy images. That was the main reason for the earlier research to be focussed on sensor-based or device-based gesture control. Devices such as data gloves, coloured gloves, arm bands and smart watches are still being used to control and command groups of robots, due to ease of implementation and reliable results.

Relatively recently, successful bare-hand gesture recognition has become possible due to the exponential increase in computing capabilities and significantly improved vision technology resulting in high quality images and faster image-processing. Various techniques have been developed to extract 'skin' pixels from an image, apply filtering to get a noise-free area of interest containing the gesture and subsequently perform feature extraction and classification to accurately identify the gesture. However, very few examples were found which combine bare-handed gesture control (for Human Swarm Interaction-HSI) with graph-theoretic formation control of a group of robots.

1.2. Overview

Formation control requires specifying formations and is usually programmed using computational devices. There is no quick way for a non-expert user to get robot swarms organized using simple means that do not involve using a computer or an electronics device. However, it is easily imagined that if adoption of robot swarms is to become a reality for the general members of public in the future, then command and control of

swarms must be made convenient. Bare-handed gesture-based control is a step in the right direction, as the use of cumbersome devices can be avoided.

1.3. Thesis Objectives

In the proposed work, the aim is to achieve various formations of a group of robots using bare-hand gestures as commands / directives, eliminating the use of any electronic or specialized devices (such as colored gloves, watches etc.). By using the consensus protocol to achieve formation control, we have a distributed algorithm which can achieve any geometric formation based on the robot positions acquired from beacons in real-time. Thus, the advantage of this system lies in the easy means of commanding the swarm.

1.4. Research Contribution

The contributions of this research work can be summarized as follows:

- Implementation of consensus-based formation control of a group of robots, based on the concepts of Graph Theory.
- Devising and implementing a method for hand-gesture recognition, in order to control a group of robots using bare-handed gestures.

1.5. Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 gives a background on the major concepts of Graph Theory and consensus-based rendezvous and formation control algorithms along with an overview of the progress made in the field of image processing and gesture-recognition. Moreover, related works to these topics are discussed. Chapter 3 goes deeper into the Consensus theory and the control laws devised for achieving Rendezvous and Formation Control. The implementation and hardware-related details are presented in Chapter 4. Chapter 5 describes the code implementation in ROS Python API. The results are presented in Chapter 6. Finally, Chapter 7 concludes the document and highlights potential for future research.

Chapter 2. Background and Literature Review

2.1. Background

A formal theoretical framework for analysis of consensus-based algorithms for multi-agent systems, was presented in [1]. The three main approaches employed by researchers in the past for formation control include leader-follower based approach, virtual-structure based approach and behavior-based approach. However, all three approaches have their flaws: leader-follower based approach has lack of robustness as the failure of the leader may result in the collapse of the whole formation, virtual-structure based approach suffers from large computational overhead [2]. It was later shown that the traditional leader–follower, virtual structure and behavior-based formation approaches can be regarded as special cases of 'consensus'-based approaches [3]. Moreover, the weakness of the traditional formation control approaches can be overcome by employing graph theoretic, consensus-based multi-agent control [4].

Since the development of consensus algorithms, other researchers have taken on more complex problems in this domain. Building on the graph theoretic approach and consensus protocol, in [4] a control strategy for time-varying formation control of unmanned aerial vehicles is presented. In consensus-based approaches, the quantity for which consensus is sought- also termed as the "information state"- (it could be the inter-agent distances, the rendezvous time, the direction of motion of a robotic swarm, the probability that a military target is destroyed etc.) is updated according to the information states of the neighbouring agents. This requires the existence of an underlying communication network which is usually denoted by a graph. For the case where this communication network can 'switch' or change, makes the achievement of formation control more complicated as compared to a fixed network. In [5], researchers have explored the idea of non-rigid formation control using the concept of 'complex' graph Laplacian matrix. Non-rigid formation control removes the constraints related to location, scale and rotation resulting in formation "shape" control.

However, there is a possibility of complications arising such as the impracticality of having agents to sense their surroundings continuously [6]. A common method to avoid this problem is to have the agents periodically broadcast and sample. However, this method has a drawback as it requires synchronized communication among a possibly

large network of agents. As pointed out in [7], real-life systems have non-linear second-order dynamics and are subject to external disturbances and modeling errors which can cause problems in physical implementation of such systems.

2.2. Earlier Research on Human Swarm Interaction

Very few papers were found on bare-hand control of a group of robots/ UAVs. In [8], Face detection was used to select an individual robot from a group of robots and motion gestures then commanded the selected robot. Another implementation was found in [9], where the user commands a swarm of small robots to form different formations. While the user does not wear any control device, a Microsoft Kinect Depth sensor was used for body tracking in addition to an overhead camera to provide absolute positioning reference. A similar example is seen in [10] for the control of industrial robotic arms using a Kinect v2 sensor. The developed algorithm recognizes the body movements and voice commands of the user, which are then used to control the ABB robotic arms.

Most of the earlier research is focussed on guiding robotic swarms either using handheld or wearable devices such as smartwatches, data gloves or specialized gear such as coloured gloves. In [11] a wearable armband provides input regarding the shape, scale and rotation of the intended formation in the form of EMG and IMU sensor data. A case-based gesture interface for Multiagent formation control using a smart watch interface is presented in [12]. Up and right directives result in vertical and horizontal straight-line formations of the LTA (Lighter than Air) agents. A formation control algorithm based on the concept of spring force and potential field is employed in [13] for formation control of multi-robot teams using a Data Glove.

Researchers have explored various approaches in selecting and commanding a sub-group of robots from a swarm using hand gestures. A colored glove-based interface was employed to select a sub-group of UAVs from a swarm of UAVs using specified gestures [14].

More recent research also focuses on making Human-Machine Interaction possible through specialized devices and gloves. A three-axis gyroscopic sensor has been used to provide both gesture recognition and trajectory tracking features [15]. A Flexible Epidermal Tactile Sensor Array (FETSA) has been developed for real-time processing of hand gestures based on the wrist muscle movement of the user [16]. In [17] deep

learning technique is employed to identify hand gestures, based on the IMU, electromyographic (EMG) and finger pressure data. Similarly, a wearable glove has been developed in [18] for Human-UAV interaction.

2.3. Hand-Gesture Recognition

The invention of glove-based control interface was the first step towards using movement or configuration of human hands as a control input to electronic devices. These data gloves came with sensors such as accelerometers and fibre-optic bend sensors, which detected the bending of finger joints to determine the opening or closing of fingers. However, even though data gloves proved very successful in the fields of graphic animation and motion capture for movies, they can be cumbersome as the user has to wear a specialized device. Since the earlier phase of research, which began as early as 1980's, other techniques have been developed such as using colored markers or specialized colored gloves which aid in gesture-recognition. Despite having these solutions, nothing compares with bare-hand gesture recognition as it is a natural component of the non-verbal communication which takes place daily amongst humans.

Despite the volume of research conducted on the topic and persistent interest of researchers, there were many hurdles faced by earlier researchers such as low-quality cameras and low-processing power of computers. Using a single camera often resulted in optical occlusion, thus multiple inputs from two (stereo setting) or more than two cameras were used to avoid this problem. However, multiple cameras require calibration and synchronization of their respective views of a particular object.

Relatively recently, many improvements in the hardware technologies have enhanced the possibility of dynamic hand gesture recognition in real-time. A number of techniques for feature extraction from gestures have been developed since then, such as: Fourier Descriptors, HOG (Histogram of Oriented Gradients), Contour Descriptors, Hu moment descriptors etc. as well as specific classification techniques such as: HMM (Hidden Markov Models), PCA (Principal Component Analysis), Linear and Nonlinear SVM (Support Vector Machine), Neural Networks among many others. In [19], thermal images of various hand gestures are captured using a specialized thermal camera attached a GPU board manufactured by NVIDIA. A deep Convolutional Neural Network is used to recognize the gestures. The forehead skin pixels are used as a template to detect a hand gesture in [20], the convex defects of the hand contour are

then used to find the fingertip pointing direction. A novel method of hand gesture recognition is described in [21] where the IR waves reflected by the hand are used to interpret the hand gesture. A specialized capacitive sensor is used in [22] for hand gesture recognition.

2.3.1. Skin segmentation

The first step in the general roadmap towards Hand Gesture Recognition is Image Pre-processing. Pre-processing usually involves skin-detection which results in a binary (black and white) image so that the unwanted background information can be removed from the image. One major obstacle in successful skin segmentation arises from the change in luminance levels in images captured under varying lighting conditions in addition to ensuring successful skin detection for all skin tones.

While the RGB (Red Green Blue) color space is more suitable to viewing images by human eye, there is a large luminance correlation between the Red Green and Blue components which renders it vulnerable to failure in skin detection under varying light conditions. For this purpose, other color spaces which have separate luminance and chrominance channels such as $\mathrm{YC_bC_r}$, YUV, HSI (Hue, Saturation and Intensity), HSV and HSL etc. [23].

We used $\mathrm{YC_bC_r}$ color space as it is not susceptible to variation in illumination conditions. The transformation from RGB to $\mathrm{YC_bC_r}$ is straightforward:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

The following thresholds for skin detection were determined using $\mathrm{YC_bC_r}$ in [24].

$$77 \leq \mathrm{C_b} \leq 127 \qquad 133 \leq \mathrm{C_r} \leq 173 \quad (2)$$

Applying these thresholds to an input image might not result in a perfect separation of skin vs. non-skin pixels and there might be graininess due to noise or other background objects detected as skin due to similar tones. In order to obtain a single hand-shaped BLOB (Binary Large Object) for further processing, Morphological Filtering is applied.

2.3.2. Morphological filtering

Mathematical Morphology is a class of non-linear shape-based operations which involve a structuring element which is applied over the original image. The non-zero values in the structuring element define the 'neighbourhood' or the pixels, which the function is applied to. The output is applied to the centre pixel in the segment of original image, which overlaps with the centre of the structuring element. Dilation and erosion are two of the most used morphological operations. In dilation, the output is the maximum value of the neighborhood pixels, determined by the structuring element. In erosion, the output is the minimum of all the neighbourhood pixel values.

Mathematical Morphology allows for the creation of very powerful shape-based filters; the shape is determined by the shape of the structural element (i.e. the location of non-zero bits in the structural element determines its shape) [25]. Mathematically, it can be expressed as:

$$\mathbf{O}(u,v) = f(\mathbf{I}[u+i, v+j]), \quad \forall (i,j) \in S \quad \forall (u,v) \in \mathbf{I} \tag{3}$$

where S is the structuring element, and I is the input image. For instance, in Figure 2-1, the structural element (represented by S), is being applied to the pixel neighbourhood (neighbourhood being determined by the size and shape of S) of the input image. The maximum or minimum operation (depending on whether dilation or erosion is being performed), is applied to the input pixels corresponding to non-zero pixels in the "mask" or structural element, i.e. in the case of a binary image, if any of the pixels in the neighbourhood has a value of 1, the result would be 1 (for dilation operation). Similarly, if erosion is being performed and any of the pixels in the neighbourhood area is 0, the result would be 0. The result determines the value of the "center" pixel (center of neighbourhood) in the output image.

Erosion is written as:

$$\mathbf{O} = \mathbf{I} \ominus S \tag{4}$$

where the operator $\ominus$ signifies minimum function. Dilation is expressed as:

$$\mathbf{O} = \mathbf{I} \oplus S \tag{5}$$

where the operator ⊕ signifies maximum function. These operations can be better

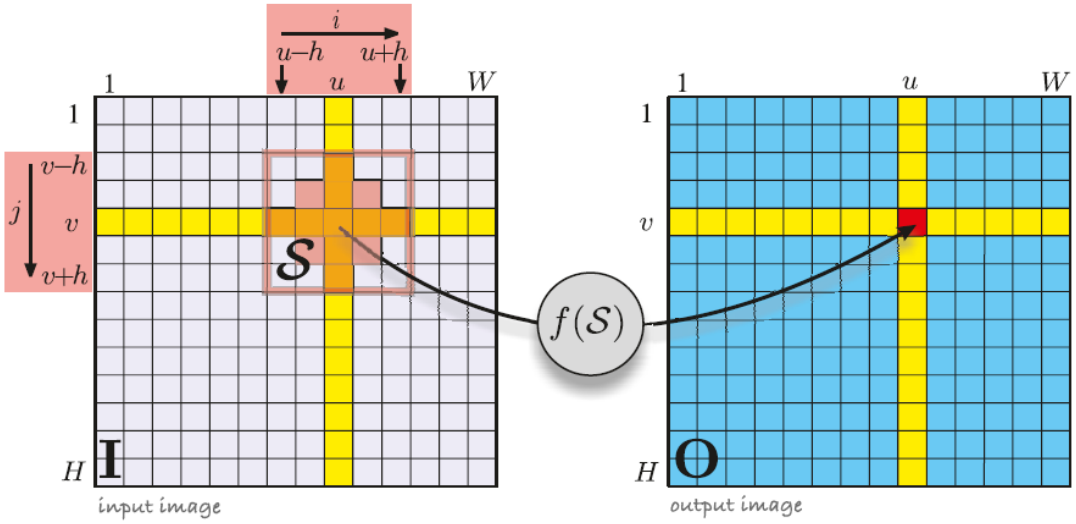


Figure 2-1: Pink blocks define the structural element. Result stored in red block [25].

understood with the help of an example. In Figure 2-2, the first column contains the input binary images while the fourth column displays the applied structural elements. The second column shows the result of erosion operation with the structural element. We can see that the shapes which are inconsistent with the shape of the structural element are eliminated from the images. Even the shapes which are similar to the structural element experience a decrease in size due to the 'erosion' operation. These features can be restored to their original sizes after application of dilation subsequent to the application of erosion. If we apply dilation after erosion, this process is named as 'opening' since it opens up gaps in the image and has an overall 'cleaning effect'.

$$\mathbf{I} \circ \mathbf{S} = (\mathbf{I} \ominus S) \oplus S \tag{6}$$

Application of erosion after dilation is termed as 'closing'.

$$\mathbf{I} \bullet \mathbf{S} = (\mathbf{I} \oplus S) \ominus S \tag{7}$$

The results of Figure 2-3 show that closing has an effect of closing the gaps in an input image. However, the results depend on the size and shape of the structural element.

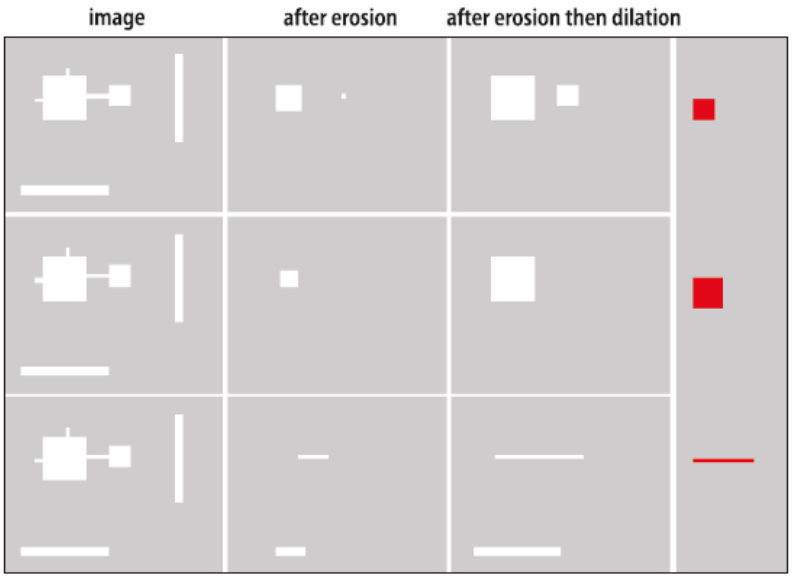


Figure 2-2: 'Opening' operation with different structural elements [25].

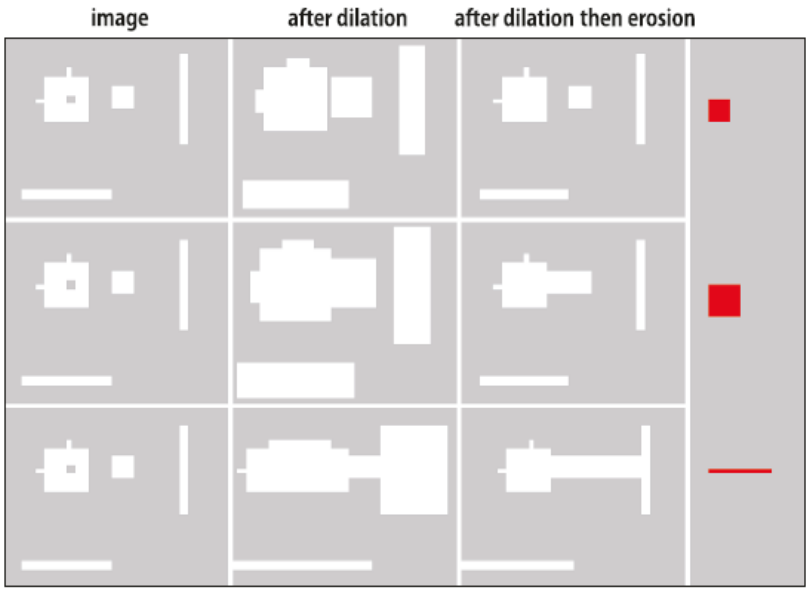


Figure 2-3: 'Closing' operation with different structural elements [25].

The opening and closing operations, when applied using a suitable structural element can result in noise removal from an input image. In Figure 2-4: (a) is the original image; (b) is the result of opening the original image with a circular structural element of radius 3; (c) is the result of applying closing subsequent to opening and (d) shows the output when the order is reversed i.e. opening is applied after closing. We can see that ‘closing after opening’ results in favorable results as compared to ‘opening after closing’. The initial opening operation fills the holes inside the black rectangular object while

increasing the size of noisy specks, the specks are removed after the subsequent application of closing, resulting in an improved image. Thus, morphological filtering techniques can be employed to remove noise from the segmented hand-gesture image. However, the size and shape of the structural element plays a major role in getting acceptable results.

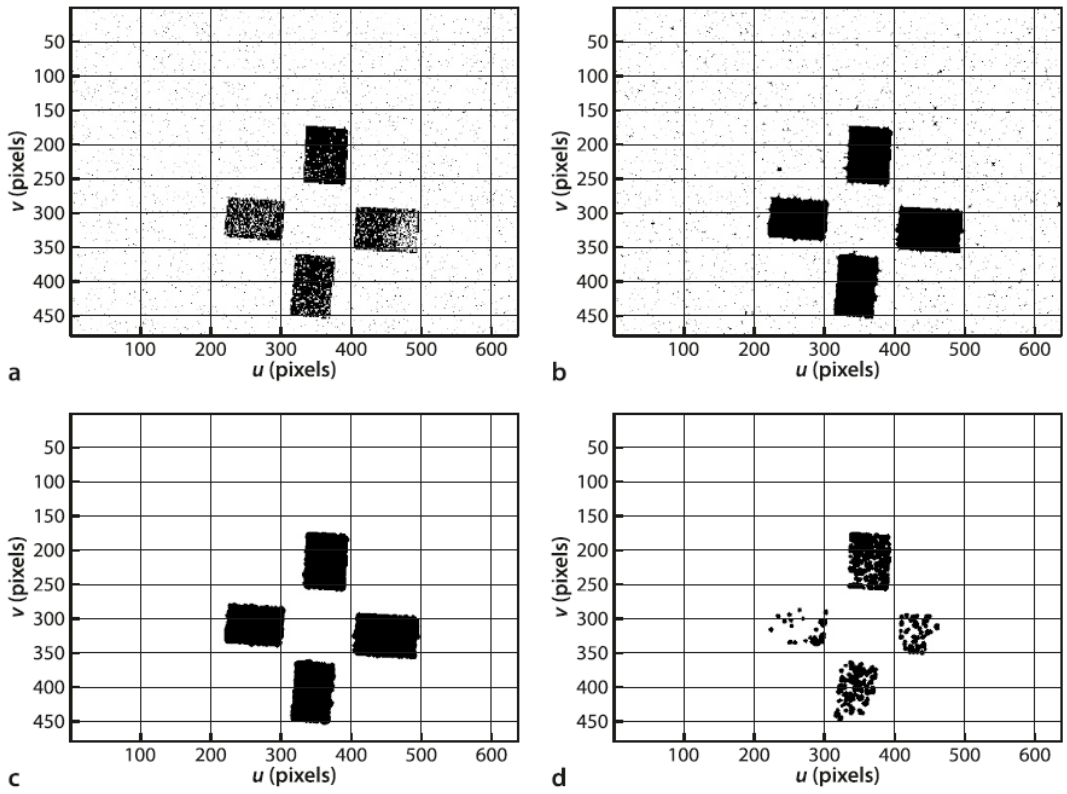


Figure 2-4: (a) Original Image (b) Opening operation (c) Closing after Opening (d) Opening after Closing [25].

2.3.3. Contour description using 1D sequence

Various methods have been developed by researchers to define the contour of the gesturing hand. A feature extraction approach based on Curvature Scale Space for scale, translation and rotation invariant recognition of hand poses is described in [26]. While the results are claimed to be up to 98% accurate, this is a computationally intensive approach. A very fast technique has been proposed in [27] for identifying the number of open fingers in a certain gesture. By employing their proposed algorithm, computationally intensive steps such as feature extraction and subsequent matching with image database can be avoided resulting in super-fast and reasonably accurate

results. Another advantage is that the algorithm is scale, rotation and translation invariant.

The hand gesture recognized by the robot/ computer is in the form of open fingers or 'counts' e.g., count of one means that any of the five digits is open. While this indicates reduced precision and fewer available gestures, the advantage lies in the absence of the requirement to memorize any particular gesture involving the use of a particular set of digits. The user simply needs to know how many open fingers correspond to a certain command.

The first step (after obtaining a noise-free, binary hand image) is to find the center of gravity (COG) of the pixels constituting hand-gesture. The next step is to find the farthest point from the COG, belonging to the hand-gesture. The coordinates $\bar{x}$ and $\bar{y}$ of the center of gravity can be found by using:

$$\bar{x} = \frac{M_{10}}{M_{00}} \qquad \bar{y} = \frac{M_{01}}{M_{00}} \tag{8}$$

where $M_{ij}$is the raw image moment of order (i,j). Then we find the longest distance from the COG to coordinates of any pixel constituting the gesture, which would be the tip of the longest open finger. A circle is drawn with center at COG and a radius which is given as:

$$r = 0.7l \tag{8}$$

where l is the longest distance to the farthest point on the contour from the COG. This circle will intersect all of the open digits. Figure 2-5 and Figure 2-6 show the centre of the hand gesture and the circle drawn to intersect all the open digits. Then we can simply trace the image along the boundary of this circle and find any transitions from black to white. The number of total transitions minus one (the wrist) would give us the number of open fingers or 'counts' in the gesture.

2.4. Graph Theory

The work by the famous Swiss mathematician Leonhard Euler on the "Königsberg Bridge Problem" is considered by many to be the beginning of the field of Graph theory [28]. In the beginning the concepts related to Graph theory were mainly used for

recreation but by the mid-1800s people began to realize the importance of this field in

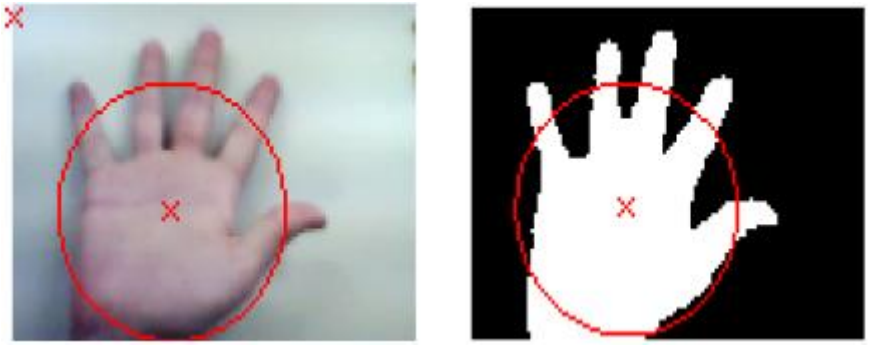


Figure 2-6: Gesture with five open digits, x marks the COG [27].

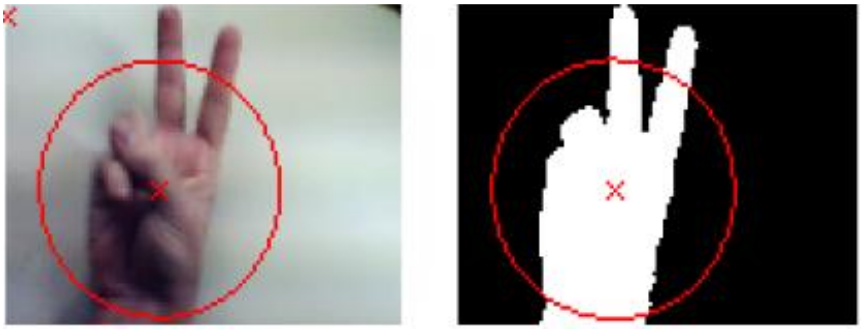


Figure 2-5: Gesture with two open digits [27].

modelling everyday phenomena.

For the last 50 years, Computer Science became a major provider of problems to Graph Theory and, simultaneously, it became the major consumer of solutions to such problems. In practical applications, graphs involved have not five, and not even ten, or twenty, but hundreds and thousands of vertices and edges [29]. It is not possible for humans to solve problems on such a large scale and computers were used to solve these problems.

2.4.1. Basic concepts

A graph G consists of a pair (V, E) where V is a nonempty finite set whose elements are called vertices and E is a set of unordered pairs of distinct elements of V [30]. The elements of E are called edges of the graph G. The vertices and edges of G are referred to as V(G) and E(G) and the notation for an edge can be simplified as $\{v_i, v_j\}$ or $v_i v_j$ or even $i\ j$. Figure 2-7 shows some examples of a graph.

If two vertices are connected by an edge, they are called adjacent, otherwise they are called disjoint. For a given vertex v_i, the number of all vertices adjacent to it is called the degree of the vertex v_i. In other words, we can define the neighbourhood $N_i \subseteq V$ of

the vertex v_i as the set $\{v_j \in V \mid v_i v_j \in E\}$, that is the set of all vertices which are adjacent to v_i. For a given vertex v_i, the number of all vertices adjacent to it is called the degree of the vertex v_i, denoted as $d(v_i)$. The maximum degree over all vertices is called the maximum degree of G, denoted by $\Delta(G)$.

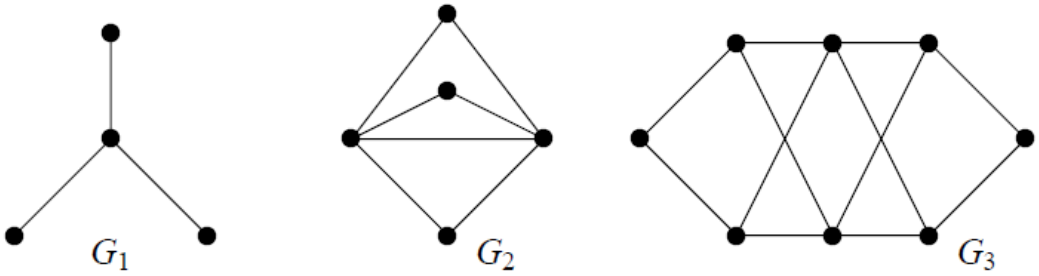


Figure 2-7: Examples of a Graph [29].

Adjacency Matrix is a square matrix, with number of rows and number of columns equal to the number of vertices in graph G. If vertex v_i is adjacent to vertex v_j, then (i, j) entry (element at i^{th} row, j^{th} column) is 1, otherwise it is 0.

For instance, for the graph in Figure 2-8, the Adjacency Matrix denoted by A(G) is given as:

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

Degree Matrix is also a square matrix, with number of rows and number of columns equal to the number of vertices in graph G. It is a diagonal matrix with the degree of each vertex $d(v_i)$ on the main diagonal. For the graph in Figure 2-8, the degree matrix $\Delta(G)$ is given as:

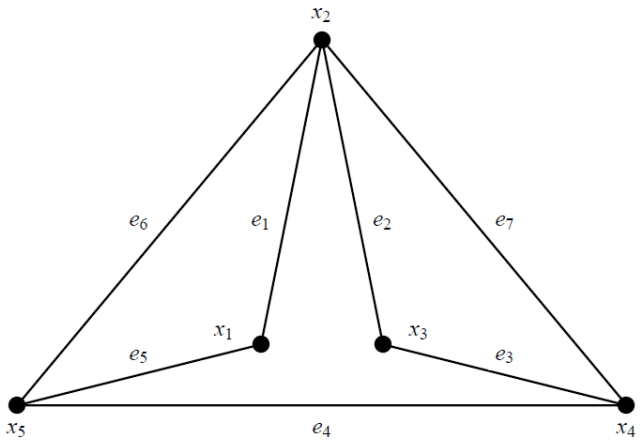


Figure 2-8: Graph with vertices denoted by x and edges by e [29].

$$\Delta(G) = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Laplacian Matrix is another important graph matrix which can be defined in different ways, but the most straightforward representation (for undirected graph) is given as:

$$L(G) = \Delta(G) - A(G) \tag{9}$$

where $\Delta(G)$ is the degree matrix and $A(G)$ is the adjacency matrix of the graph. For Figure 2-8, the Laplacian matrix $L(G)$ is given as:

$$L(G) = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ -1 & -1 & 0 & -1 & 3 \end{bmatrix}$$

It can be observed that the row sum of all rows of the Laplacian matrix is zero.

Algebraic graph theory associates algebraic objects, such as matrices and polynomials, to graphs, and by doing so makes available a range of algebraic techniques for their study. The degree, adjacency, incidence, and Laplacian matrices associated with a graph are examples of objects in algebraic graph theory. The Graph Laplacian Matrix $L(G)$ is symmetric and positive semidefinite, hence its real eigenvalues can be ordered as:

$$\lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_n(L) \tag{10}$$

with $\lambda_1(L) = 0$. The graph G is connected if and only if $\lambda_2(L) > 0$ [31].

2.4.2. Types of graphs

So far, we have discussed graphs without considering the orientation of the edges. If all the edges in a graph are undirected, we call it an undirected graph, or simply a graph. However, if the edge orientation is defined, the graph is called a directed graph or a digraph. Figure 2-9 displays examples of a directed and undirected graph.

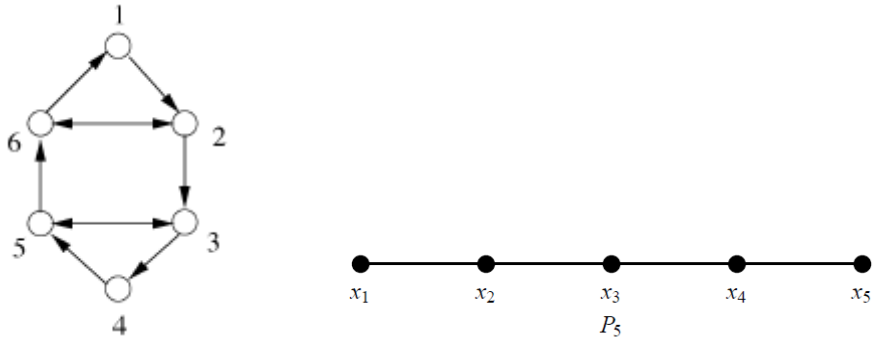


Figure 2-9: (a) Digraph (b) Path Graph

A graph in which all vertices can be numbered from left to right, such that there is precisely one edge connecting every two consecutive vertices and there are no other edges, is called a path. It is denoted by P_n where n is the number of vertices.

An undirected graph is called connected if in it any two vertices are connected by some path, otherwise it is called disconnected. It means that in a disconnected graph there always exists a pair of vertices having no path connecting them. A disconnected graph can be seen in Figure 2-10. For a directed graph (digraph), it is *weakly connected,* if all the arrow heads are removed to get an undirected graph; and the resulting undirected graph is connected.

A digraph is *strongly connected,* if there is a directed path from any vertex v_i, to all the other vertices.

A connected graph in which every vertex has degree 2 is called a cycle. It is denoted by C_n where n is the number of vertices, as seen in Figure 2-11 (a).

A graph in which every pair of vertices is an edge, is called complete denoted by K_n where n is the number of vertices. The graph in Figure 2-11 (b) is a complete graph with four vertices. It is called complete because we cannot add any new edge to it.

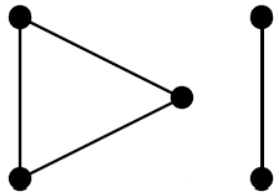


Figure 2-10: Disconnected Graph

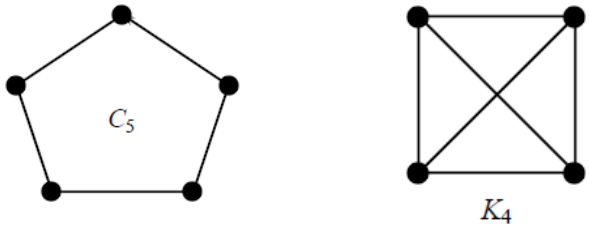


Figure 2-11: (a) Cycle Graph (b) Complete Graph

A digraph is weakly connected if its unoriented version is connected. This means that if we remove the arrowheads from a digraph, and it results in a connected graph, the original digraph is a weakly connected digraph. Example of a weakly connected digraph can be seen in Figure 2-12 (b).

A digraph is strongly connected if between every pair of distinct vertices, there is a directed path, as shown in Figure 2-12 (a).

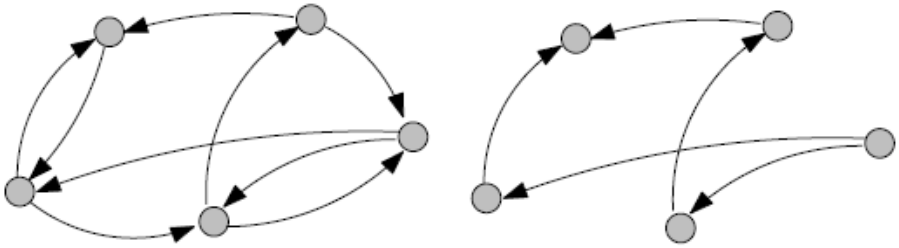


Figure 2-12: (a) Strongly Connected Digraph (b) Weakly Connected Digraph

A digraph is balanced if for every vertex, the in-degree and out-degree are equal. This means that every vertex has the same number of arrows originating from it, as those

terminating on it. Examples of balanced and unbalanced graphs can be seen in Figure 2-13.

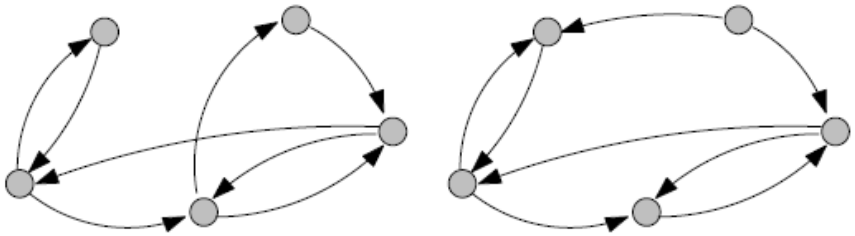


Figure 2-13: (a) Balanced Digraph (b) Unbalanced Digraph

A digraph is a rooted out-branching if:

1. It does not contain a directed cycle.
2. It has a root vertex v_r such that for every other vertex v included in the digraph, there is a directed path from v_r to v.

A digraph and its rooted out-branching subgraph can be seen in Figure 2-14.

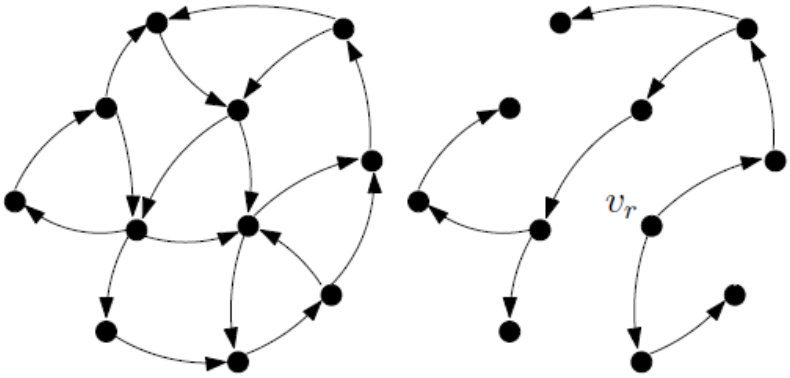


Figure 2-14: (a) Original Digraph (b) Rooted out-branching subgraph

2.5. The Consensus Protocol

A consensus algorithm (or protocol) is an interaction rule that specifies the information exchange between an agent and all of its neighbours on the network [1]. The collective dynamics of the group of agents following a simple consensus algorithm to reach an agreement regarding the state, can be expressed as:

$$\dot{x} = -Lx \tag{11}$$

where $L = [l_{ij}]$ is the graph Laplacian of the network and its elements are defined as:

$$l_{ij} = \begin{cases} -1, & j \in N_i \\ |N_i|, & j = i \end{cases} \quad (12)$$

Here, N_i denotes the neighbourhood of node i and $|N_i|$ denotes the number of neighbors of node *i* (or out-degree of node *i*). This is same as the definition of the Laplacian Matrix given earlier.

It follows from Gershgorin's disc theorem [32] that, for an undirected graph, all of the nonzero eigenvalues of L are positive (L is positive semidefinite), whereas, for a directed graph, all of the nonzero eigenvalues of L have positive real parts. Therefore, all of the nonzero eigenvalues of −L have negative real parts. For an undirected graph, 0 is a simple eigenvalue of L if and only if the undirected graph is connected [33]. For a directed graph, 0 is a simple eigenvalue of L if the directed graph is strongly connected [34, Proposition 3], although the converse does not hold. For an undirected graph, $\lambda_2(L)$ is the algebraic connectivity, which is positive if and only if the undirected graph is connected. The algebraic connectivity quantifies the convergence rate of consensus algorithms [35].

A digraph must be strongly connected to ensure convergence. A strongly connected graph is one where there is a directed path from any node to all the rest of the nodes. Whereas, for an undirected graph, just connectedness is enough for convergence. However, for a directed graph (digraph), the condition for reaching *average consensus* (consensus equilibrium is the average of initial equilibrium states) is that the digraph should be strongly connected in addition to being balanced [3]. But a weakly connected and balanced digraph is automatically strongly connected, hence it contains a rooted directed spanning tree (proof in [31]).

2.5.1. Rendezvous

The consensus protocol ensures that the information states of the agents comprising a network/ graph come into agreement. The rendezvous problem is defined as having a team or group of vehicles/ agents/ robots tasked to arrive at a specified location. When the consensus algorithm is applied to the positions of agents (in both x and y coordinates separately), they can converge at a particular point, The agents do not need to know their own positions or the position of the point of rendezvous, they only need to know the distance from their respective neighbours (depending on the graph topology). As

long as the graph is connected (for an undirected sensing topology), the agents will be able to converge at a particular point.

2.5.2. Rendezvous with delta-disk topology

Earlier, we did not consider any conditions on the existence of edges between agents. In real-life, the robotic agents will be equipped with sensors to detect the relative displacement from their neighbours, and all sensors have a certain sensing range. This results in a geometric condition on the existence of edges between neighbouring robots:

$$\{v_i, v_j\} \in E \text{ if and only if } \| x_i - x_j \| \leq \Delta \tag{13}$$

where, Δ is the sensing range/ radius of each robot. By incorporating this condition, our graph becomes a delta-disk proximity graph. Such graphs are dynamic in nature, as edges may appear or disappear when agents move in or out of the sensing (or communication) range of each other [31].

The linear control laws formulated earlier, to achieve Rendezvous or formation control for a network with inter-agent connections governed by the Delta-disk proximity graph, are not successful anymore. The algorithm needs to be modified to take the geometric range constraints into account.

Let $\ell_{ij}(x)$ be the edge vector between nodes i and j; moreover, if we are constrained by distance δ as the maximum possible distance between any two agents for the existence of an edge between them; the edge-tension function $\mathcal{V}_{ij}$ is defined as:

$$\mathcal{V}_{ij}(\delta, x) = \begin{cases} \frac{\| \ell_{ij}(x) \|^2}{\delta - \| \ell_{ij}(x) \|} & if \{v_i, v_j\} \in E, \\ 0 & otherwise. \end{cases} \tag{14}$$

With,

$$\frac{\partial \mathcal{V}_{ij}(\delta, x)}{\partial x_i} = \begin{cases} \frac{2\delta - \| \ell_{ij}(x) \|}{(\delta - \| \ell_{ij}(x) \|)^2} (x_i - x_j) & if \{v_i, v_j\} \in E, \\ 0 & otherwise. \end{cases} \tag{15}$$

Thus, a control law of the following form is obtained:

$$\dot{x}_i(t) = -\sum_{j \in N_i} \frac{\partial \mathcal{V}_{ij}(\delta, x)}{\partial x_i} = -\sum_{j \in N_i} \frac{2\delta - \| \ell_{ij}(x) \|}{(\delta - \| \ell_{ij}(x) \|)^2} (x_i - x_j) \quad (16)$$

However, since for a Dynamic Graph, the graph topology changes as the agents move in or out of each other's sensing range. It can be seen from the equation above that the tension energy becomes infinite when $\| \ell_{ij} \| = \delta$. In order to avoid this problem, we introduce hysteresis into the system that is; the effect of an edge is included in the total tension energy only when $\| \ell_{ij} \| \leq (\Delta - \epsilon)$ where ϵ is the pre-defined switching threshold.

2.6. Formation Control

Formation Control is usually the first problem that needs to be addressed when controlling a team of robots.

The desired formation which we are planning to achieve can be specified as,

$$\mathrm{Z} = \{\xi_1, \xi_2, \dots, \xi_n\}, \qquad \xi_i \in R^p \quad i = 1,2, \dots, n \quad (17)$$

Where:

$$\| \xi_i - \xi_j \| = d_{ij} \qquad i, j = 1,2, \dots, n \quad i \neq j \quad (18)$$

Where d_{ij} represents the desired inter-agent distance between agents i and j. Any collection of points $x_1, x_2, \dots, x_n$ in R^p thus satisfies the formation specification if $x_i = \xi_i + \tau$ for all $i = 1,2, \dots, n$ for some arbitrary translation $\tau \in R^p$.

The underlying graph topology may be *static* or *dynamic*; for a dynamic graph, the edge set *E(t)* can change when the agents move around in the environment. Whether the graph is static or dynamic, our goal with formation control is for the agents to move in such a way that: $\| x_i - x_j \|$ converges asymptotically to d_{ij}.

If we are dealing with a dynamic graph, we have the additional condition that while the agents move to the intended formation, none of the edges included in the desired edge set E_f should be lost at any time during the maneuvering i.e. $E_f \subseteq E(t)$ for all $t \geq T$ for some finite $T \geq 0$.

2.6.1. Continuous-time formation control

As long as the condition of convergence for the agreement protocol is fulfilled (for an undirected graph, it should be connected), we get the following distributed formation control strategy:

$$\dot{x}_i(t) = -\sum_{j \in N_i} (x_i(t) - x_j(t)) - (\xi_i - \xi_j) \tag{19}$$

Here N_i is the set of nodes adjacent to vertex i in the formation graph G, $x_i(t)$ is the position of agent i and $x_j(t)$ is the position of agent j where $j \in N_i$. As long as the convergence condition is fulfilled, all agents will be driven to the desired inter-agent distances $d_{ij} = \xi_i - \xi_j$.

2.6.2. Discrete-time formation control

The discrete-time consensus algorithm is given as:

$$X[k+1] = X[k] - \epsilon\, L\, X[k] \tag{21}$$

Or:

$$X[k+1] = (I - \epsilon\, L)\, X[k] \tag{21}$$

Where P is the transition probability matrix called the Perron matrix.

$$P = I - \epsilon L \tag{23}$$

Where $0 < \epsilon < \frac{1}{d_m}$ and d_m is the maximum degree of a node present in the network.

For a digraph G with the maximum degree d_m, the properties of the Perron Matrix P are given in [1] as:

1. P is a non-negative, row-stochastic matrix with a trivial eigenvalue of 1.
2. All eigenvalues of P are in a unit circle.
3. P is a doubly stochastic matrix if G is a balanced graph.
4. P is a primitive matrix if G is strongly connected and $0 < \epsilon < \frac{1}{d_m}$

The control law for continuous-time Formation Control is given in (16). The same law can be re-written to include the graph Laplacian matrix in the following form:

$$\dot{x}_i(t) = -L\, x_i(t) + L\, Z \tag{24}$$

Where column vector Z contains the desired robot positions. When we discretize this equation, we end up with the following form:

$$X[k+1] = X[k] - \epsilon\, L\, X[k] + \epsilon\, L\, \mathrm{Z}[k] \tag{25}$$

2.6.3. Formation control with delta-disk topology

Similar to the way in which a control law was formulated for the Rendezvous problem, which preserves connectedness even for the case of a Δ-constrained dynamic graph, a control methodology needs to be developed for the formation control problem. Given a desired formation, the goal of distributed formation control is threefold:

A. The desired edge set $E_d(t)$ should be a sub-graph of the dynamic edge set *E(t)* for all $t \geq T$ for some finite $T \geq 0$.
B. The distances $\| \ell_{ij}(t) \| = \| x_i(t) - x_j(t) \|$ converge asymptotically to the desired distances $\| d_{ij} \|$ for all edges included in the desired edge set $\mathrm{E_d}$.
C. Each agent only uses the local sensor information i.e. the detected inter-agent distances.

For a set of arbitrary positions $\xi_i \in R^n$ that are consistent with the desired inter-agent distances: $d_{ij} = \xi_i - \xi_j$ for all $\{v_i, v_j\} \in E_d$; the displacement from ξ_i at time t is given as:

$$y_i(t) = x_i(t) - \xi_i \tag{26}$$

If $\lambda_{ij}(t)$ is defined as:

$$\lambda_{ij}(t) = \ell_{ij}(t) - d_{ij} \tag{27}$$

Where $\ell_{ij}(t) = x_i(t) - x_j(t)$ as defined earlier. If the edge tension function is defined as:

$$\mathcal{V}_{ij}(\delta - \| d_{ij} \|, y) = \begin{cases} \dfrac{\| \lambda_{ij} \|^2}{\delta - \| d_{ij} \| - \| \ell_{ij} \|} & if\ \{v_i, v_j\} \in E_d, \\ 0 & otherwise. \end{cases} \tag{28}$$

Thus, a control law of the following form can be derived:

$$\dot{x}_i(t) = -\sum_{j \in N_{G_d}(i)} \frac{2\left(\delta - \| d_{ij} \|\right) - \| \ell_{ij}(t) - d_{ij} \|}{(\delta - \| d_{ij} \| - \| \ell_{ij}(t) - d_{ij} \|)^2} (x_i(t) - x_j(t) - d_{ij}) \quad (29)$$

The proof that this control law guarantees connectedness and convergence can be found in [36].

Chapter 3. Methodology

3.1. Overview

The implemented system is a robotic swarm / network sensitive to bare-handed gesture-based commands, perceived through the v2 camera module connected to a raspberry pi controller. The global positioning coordinates are generated by the Marvelmind beacons (installed on all mobile robots). The underlying network will be a static network, as the communication links between individual robots will not change as inter-agent distances increase or decrease.

The Raspberry Pi controller which acts as a controller for the group of robots, has a camera module attached to it. It has a ROS Node running on it, which performs the gesture identification. An image is captured by the camera every second. As the execution is fast, the controller is able to generate command inputs to the swarm in real-time.

3.2. Gesture Vocabulary

Our gesture-recognition algorithm is proposed to afford us five (5) individual hand gestures (finger counts). Table 3-1 links the individual gestures to the associated directives perceived by the swarm.

Table 3-1: Gesture Vocabulary based on the number of open digits recognized

Finger Count	Associated Command
1	Robots to assemble in a linear (static) formation
2	Robots to begin moving along the y-axis in a linear formation
3	Robots to assemble in a square (static) formation
4	Robots to begin moving along the y-axis in a square formation
5	Halt and Abort action on any on-going assembly task

Figure 3-1 shows a group of four Kobuki robots in an initial random configuration which subsequently assemble into a square formation, shown in Figure 3-2.

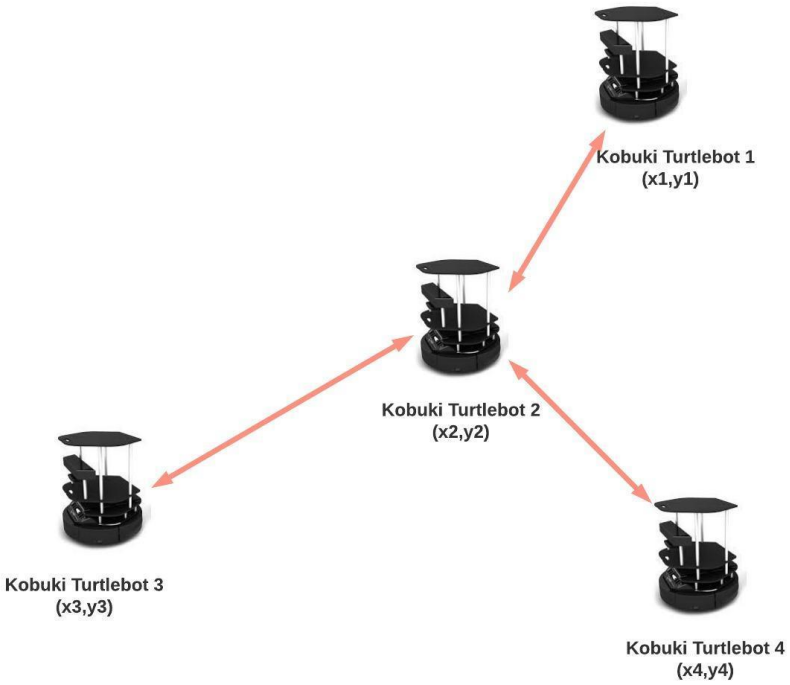


Figure 3-1: Kobuki Turtlebots in Random Configuration

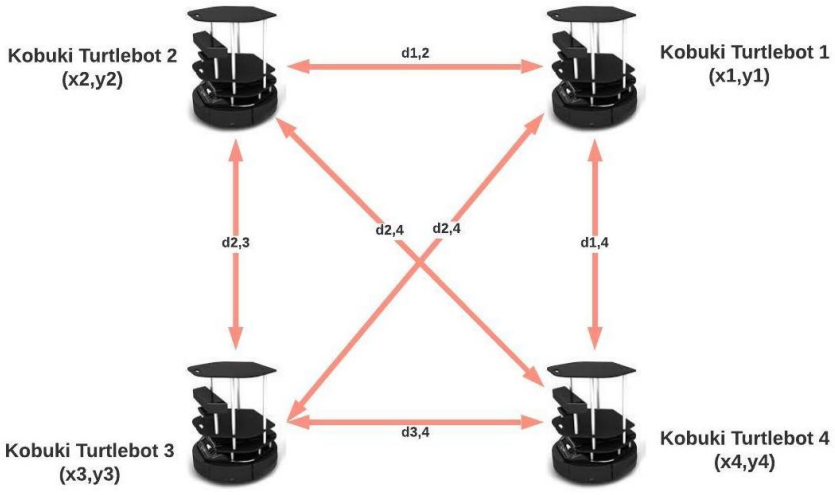


Figure 3-2: Kobuki Turtlebots in Square Formation

Whenever the controller will detect a hand gesture with at least one open digit, the status of the command input will change, and the group of robots will act according to the encoded action linked to that particular command. In case of an invalid input, there is no change in the group behavior. A diagram showing the process flow is given in Figure 3-3.

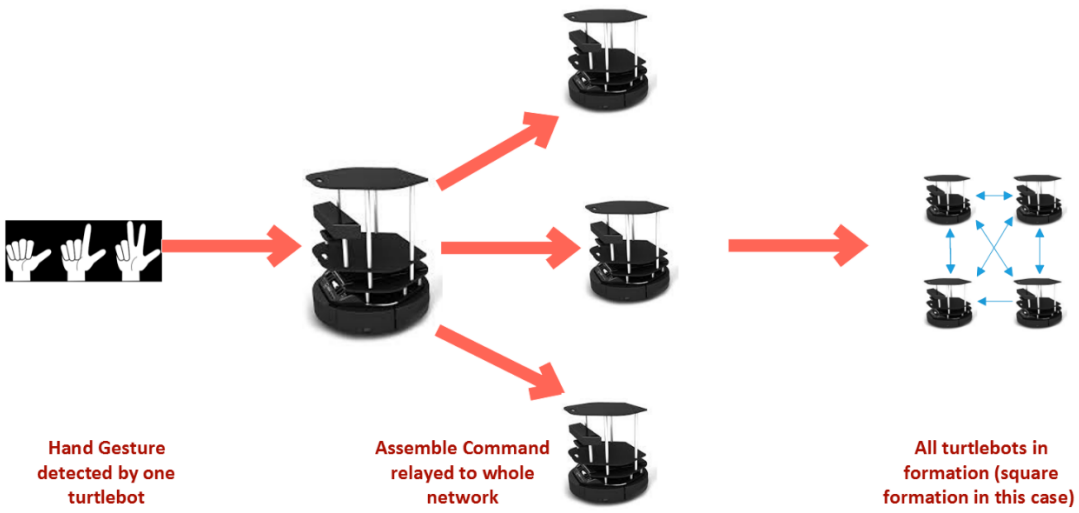


Figure 3-3: Process Flow for the Move into Formation directive

3.3. ROS Python Nodes

Two kinds of ROS Nodes have been created: one for the Gesture Recognition part which runs on the separate Raspberry Pi Controller and the other type is for the Formation Control algorithm. Each agent/ mobile robot runs the Formation Control ROS Node, however, there is a slight change depending on the vertex number that the mobile robot associates with, in the underlying communication graph. This is because the assignment of vertices/ nodes is not random but is pre-assigned.

Both types of ROS Nodes have been written in Rospy which is a Python client library for ROS.

3.4. Formation Control Node

The rospy, numpy, math and time libraries are imported in the beginning; in addition to the message files containing the message structures of the relevant topics. The following four topics are subscribed to:

1. /hedge_pos_ang
2. /odom
3. /mobile_base/events/bumper
4. /finger_count

The topic /hedge_pos_ang is published by the hedge_rcv_bin node which comes with the ROS package for the Marvelmind beacons (marvelmind_nav). It publishes the x and y coordinates of the mobile and stationary beacons in real-time, which are currently active and visible on Dashboard. The /odom topic is published to by the Kobuki robot which includes the odometry data such as the distance covered and the orientation of the robot. The topic /mobile_base/events/bumper is also published to by the Kobuki robot. The robot has one front bumper, one on the left and one on right. Whenever there is a collision with an object, the bumper is pressed, which can be detected by subscribing to the /mobile_base/events/bumper topic. This feature has been used to halt movement in the direction of the object with which the collision occurred and make the robot turn by 90 degrees. The /finger_count topic is not a pre-built topic being used by the installed packages, rather it is a topic created by the Gesture Recognition Node. The Gesture Recognition Node, which runs on the Raspberry Pi with a camera connected to it, publishes the number of fingers it has detected in the last recorded hand gesture, on this topic. The Formation Control Nodes running on all the mobile robots are thus able to receive this information and react to it, by subscribing to the topic /finger_count.

The only topic on which data is published by the Formation Control Node is the /mobile_base/commands/velocity provided by the installed kobuki ROS package. The linear and angular velocity commands can be published on this topic. The communication flowchart between various ROS nodes has been shown in Figure 3-4.

In the main loop, the node and the Subscribers and Publishers are initialized. Then, depending on the finger count received from the controller, one of the five functions is selected. Each function implements the algorithm for a different kind of robot formation and there is a function which stops all mobile robots. So, we can have four different kinds of formations and the fifth command can halt the robots.

The code for achieving a formation evaluates next position for the robot, based on the result of the formation control law (29). The position is determined by the x and y

coordinates, so the consensus equation must be evaluated for both coordinates. This evaluation is done based on the current robot positions received from the mobile beacons.

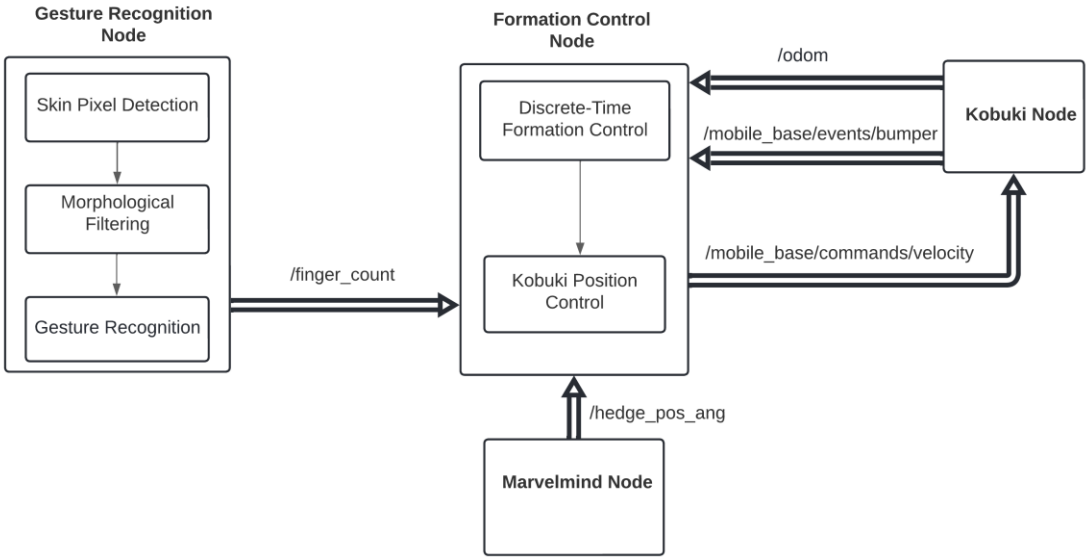


Figure 3-4: Flow Chart of communication between ROS Nodes

3.4.1. Kobuki position control

The next step is to evaluate the linear and angular velocity values required to reach the target position.

$$v_l = k_1 \frac{\Delta s}{k_2 |cos\Delta\theta|} \tag{30}$$

$$\Delta s = \sqrt{\left(x_{target} - x_{current}\right)^2 + \left(y_{target} - y_{current}\right)^2} \tag{31}$$

$$\Delta\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) - \theta_{current} \tag{32}$$

Where v_l is the linear velocity value given to the Kobuki robot through the /mobile_base/commands/velocity topic, k_1 and k_2 are constants, Δs is the distance between the target and current positions and $\Delta\theta$ is the angle difference between the required and current heading/ orientation of the robot. The current heading value $\theta_{current}$ of the robot is determined based on the orientation reading received from the /odom topic. The published orientation is in quaternions which has to be converted to Euler angles using the function euler_from_quaternion( ).

Figure 3-5 shows the diagram and orientation angles of a Kobuki robot. The angular velocity is given as:

$$v_a = k_3 \, \Delta\theta \tag{33}$$

Where v_a is the angular velocity value relayed to the robot by publishing to the topic /mobile_base/commands/velocity, k_3 is a constant and $\Delta\theta$ is the difference in the current and target angles/ heading of the robot.

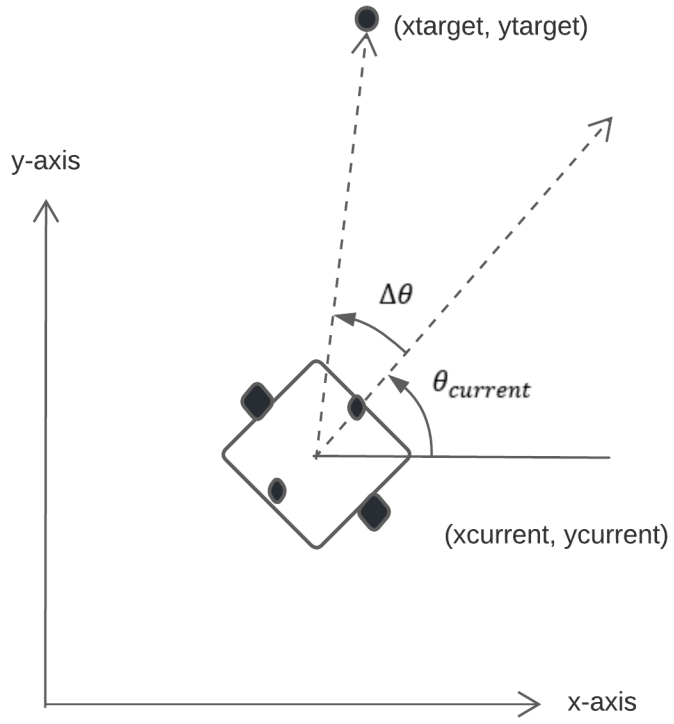


Figure 3-5: Position Control of the Kobuki mobile robot

In order to have a moving formation, a constant value is added to the y coordinate evaluated in accordance with the formation control law. This results in a moving formation along the y-axis.

3.5. Gesture Recognition Node

In the Gesture Recognition Node, OpenCV-Python library needs to be imported in addition to the rospy, math and time Python libraries. The raspistill shell command is used to capture an image using the V2 camera. The image width and height hence the resolution can be given as an argument to the raspistill command.

The first step in image processing is to convert the RGB image to an image in the YCrCb color space. This is done using the cvtColor( ) function from OpenCV library. As explained earlier, the YCrCb color space is less susceptible to changes in illumination.

The next step is to apply the threshold given in (1) for skin-pixel identification. The skin pixels are reasonably accurately identified, however there is some misidentification. Changes in light conditions can also lead to pixel misidentification. Most of the misidentified pixels are random small patches which can be removed using filtering techniques. Here morphological filtering is used for removing the noise from the image. The shape and size of the structuring element can be defined with the OpenCV function getStructuringElement( ). The Opening and Closing functions can be applied similarly using the morphologyEx( ) function which results in significant noise removal.

Next step is to find the contours present in the image which is done using the findContour( ) function. The skin-pixel containing blob (binary linked object) with the largest contour size will be the hand image blob. The longest contour can be found using the arcLength( ) function.

In order to find the coordinates of the center of the hand image blob, the concept of image moment needs to be understood. The raw image moment of order (i,j) is given as:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y) \tag{34}$$

The area of the blob is the (0,0) order image moment. The centroid of the blob is thus given by:

$$\bar{x} = \frac{M_{10}}{M_{00}} \qquad \bar{y} = \frac{M_{01}}{M_{00}} \tag{35}$$

Then we need to find the point on the hand image which is farthest from the centroid. This should give us the fingertip of the longest open digit. We need this distance in order to draw a circle centred on the centroid, which is able to intersect all the open digits. A circle is then drawn with a radius given in equation (36). While the original

paper [27] suggests a radius length given in equation (9), best results were achieved with a circle of radius given in equation (36):

$$r = 0.65\, l \tag{36}$$

Where l is the longest length from the centroid to the farthest point on the contour. The overlap (AND operation) of this circle with the hand image gives us the finger segments and the wrist segment. These segments can be counted, and one is subtracted (to exclude the wrist segment) to get the number of open digits. The results of the noise-removal and segment-detection operations applied on an input hand-gesture image can be seen in Figure 3-6.

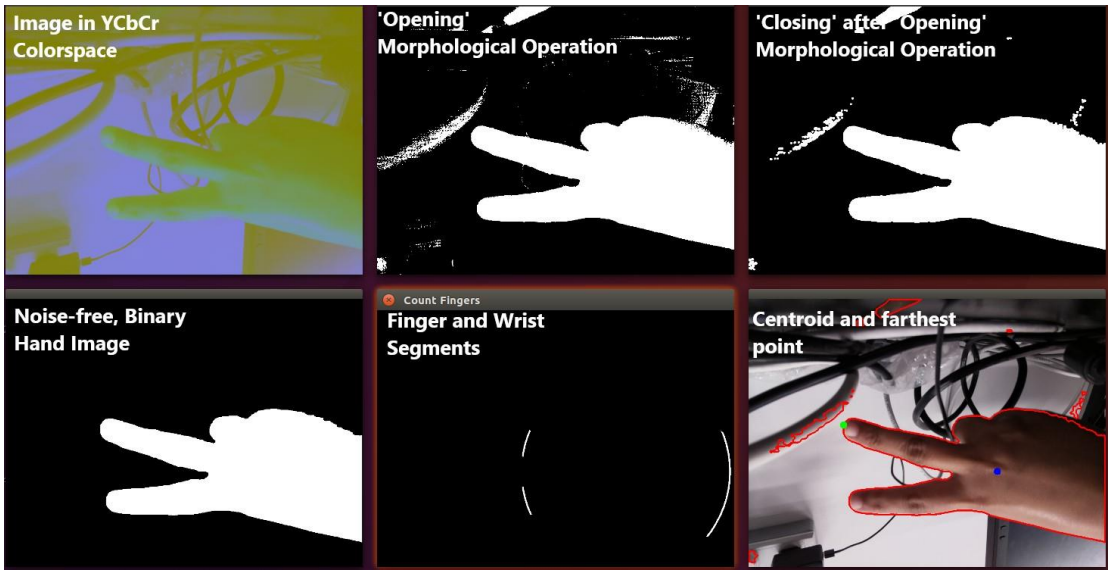


Figure 3-6: Results of finger count detection algorithm

The detection results are good, however, the r versus l ratio needs to be adjusted in order to get all digits detected. This can be challenging sometimes as the thumb and the little finger are shorter than the rest of the fingers. In order to overcome this problem, an ellipse can be drawn rather than drawing a circle. The ellipse needs to be oriented along the longest finger. The lengths of both the major and the minor axis of the ellipse need to be specified.

$$major - axis = 0.65\, l \qquad minor - axis = 1.6(0.32\, l) \tag{37}$$

Where 0.32 is the approximate ratio of the length l to half of palm width. This gives us an ellipse with suitable dimensions. The orientation angle of the ellipse also needs to be specified. This is found using the inverse tangent function applied on the coordinates

of the contour point farthest from the centroid. This basically aligns the ellipse along the longest digit.

$$\theta_{ellipse} = \tan^{-1}\left(\frac{l_y - \bar{y}}{l_x - \bar{x}}\right) \quad (38)$$

Where l_x and l_y are the x and y coordinates of the farthest point on the contour from the centre of the hand image blob.

Replacing the circle with ellipse results in significant improvement in accuracy. The number of open digits found in the hand gesture are finally published on the ROS topic /finger_count.

Chapter 4. Experimental Setup

4.1. Hardware Setup

The intended implementation of the network involves four Kobuki turtlebots. The Kobuki mobile platform needs a microcontroller board (connected through Serial Port) or a computer (connected through USB port) to operate.

4.1.1. Kobuki turtlebot

Kobuki is a low-cost mobile base which has been designed for education and robotics-related research, as seen in Figure 4-1. Power connectors are provided for an external computer as well as additional sensors (Bumper Sensor, Cliff Sensor, Wheel Drop Sensor etc.) and actuators. Gyroscope is also included which enables precise navigation.

4.1.1.1. Components

- Kobuki Base
- 1x4S1P battery 2200 mAh
- Battery charger
- USB communication cable

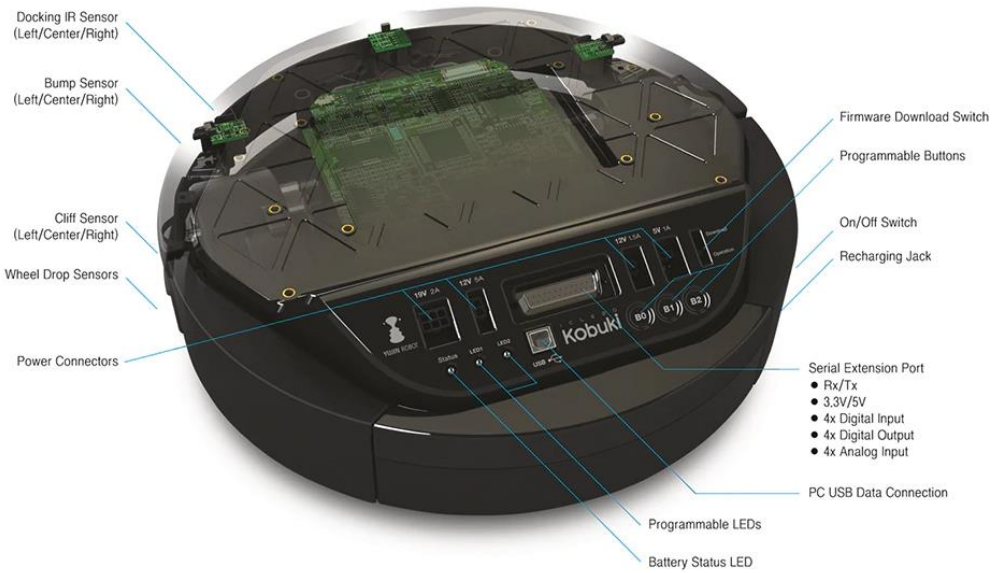


Figure 4-1: Various features and sensors available on a Kobuki mobile platform

4.1.1.2. Functional specifications

- Maximum translational velocity: 70 cm/s
- Maximum rotational velocity: 180 deg/s (>110 deg/s gyro performance will degrade)
- Payload: 5 kg (hard floor), 4 kg (carpet)
- Cliff: will not drive off a cliff with a depth greater than 5cm
- Threshold Climbing: climbs thresholds of 12 mm or lower
- Rug Climbing: climbs rugs of 12 mm or lower
- Expected Operating Time: 3/7 hours (small/large battery)
- Expected Charging Time: 1.5/2.6 hours (small/large battery)
- Docking: within a 2mx5m area in front of the docking station

4.1.1.3. Hardware specifications

- Motor Overload Detection: disables power on detecting high current (>3A)
- Odometry: 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm
- Gyro: factory calibrated, 1 axis (110 deg/s)
- PC Connection: USB or via RX/TX pins on the parallel port
- Bumpers: left, center, right
- Cliff sensors: left, center, right
- Wheel drop sensor: left, right
- Power connectors: 5V/1A, 12V/1.5A, 12V/5A
- Expansion pins: 3.3V/1A, 5V/1A, 4 x analog in, 4 x digital in, 4 x digital out
- Audio: several programmable beep sequences
- Programmable LED: 2 x two-coloured LED
- State LED: 1 x two coloured LED [Green - high, Orange - low, Green & Blinking - charging]
- Buttons: 3 x touch buttons
- Battery: Lithium-Ion, 14.8V, 2200 mAh (4S1P - small), 4400 mAh (4S2P - large)
- Firmware upgradeable: via USB
- Sensor Data Rate: 50Hz
- Recharging Adapter: Input: 100-240V AC, 50/60Hz, 1.5A max; Output: 19V DC, 3.16A

- Netbook recharging connector (only enabled when robot is recharging): 19V/2.1A DC
- Docking IR Receiver: left, centre, right
- Diameter: 351.5mm / Height: 124.8mm / Weight: 2.35kg (4S1P - small)

4.1.1.4. Software specification

- C++ drivers for Linux and Windows
- ROS node
- Gazebo Simulation

4.1.2. Raspberry pi 4 microcontroller board

Each Kobuki robot is equipped with a Raspberry Pi 4 Model B microcontroller. The Raspberry Pi 4 comes in 1GB, 2GB, 4GB and 8 GB RAM variants, we have used the ones with 8GB RAM for more processing power. Raspberry Pi 4 is a Quad core 64-bit System-on-Chip Controller. It has a standard 40 pin GPIO header. It has 2 USB 3.0 ports, 2 USB 2.0 ports and 2 micro-HDMI ports. The board has a voltage requirement of 5V DC via either USB-C connector or GPIO header and a current rating of 3 A.

The Raspberry Pi Controllers have Ubuntu Mate 20.04 installed on it. The reason for choosing Ubuntu Mate OS rather than the mainstream Debian Raspberry Pi OS is because Ubuntu Mate is compatible with ROS as opposed to the Raspberry Pi OS.

The various hardware features and connectors of the Raspberry Pi 4 Model B SOC, such as the USB 2.0 and USB 3.0 connectors, micro-HDMI ports, micro-SD card slot and the camera port can be seen in Figure 4-2.

The controller boards installed on each of the robots need to be powered up. The Kobuki robots have different power outlets connected to the Kobuki's Lithium-Ion battery. There are power connectors providing 5V/1A, 12V/1.5A, 12V/5A & 3.3V/1A. The Raspberry Pi 4 has a higher current requirement at 3 A, than the earlier controllers which required 2.5 A. The LM2965 Buck Converter was used which is an adjustable voltage regulator with the capability of converting an input ranging from 4.0-40V to anywhere between 1.5-35V and a provides a rated current of 2-3A. The buck converter gets input from the 12V/5A power connector of the Kobuki robot and steps the voltage down to 5 V while providing a current up to 3A to match the power requirements of the

Raspberry Pi 4 controller. Figure 4-3 shows the Raspberry Pi board being powered by the Kobuki through the LM2956 Buck Converter.

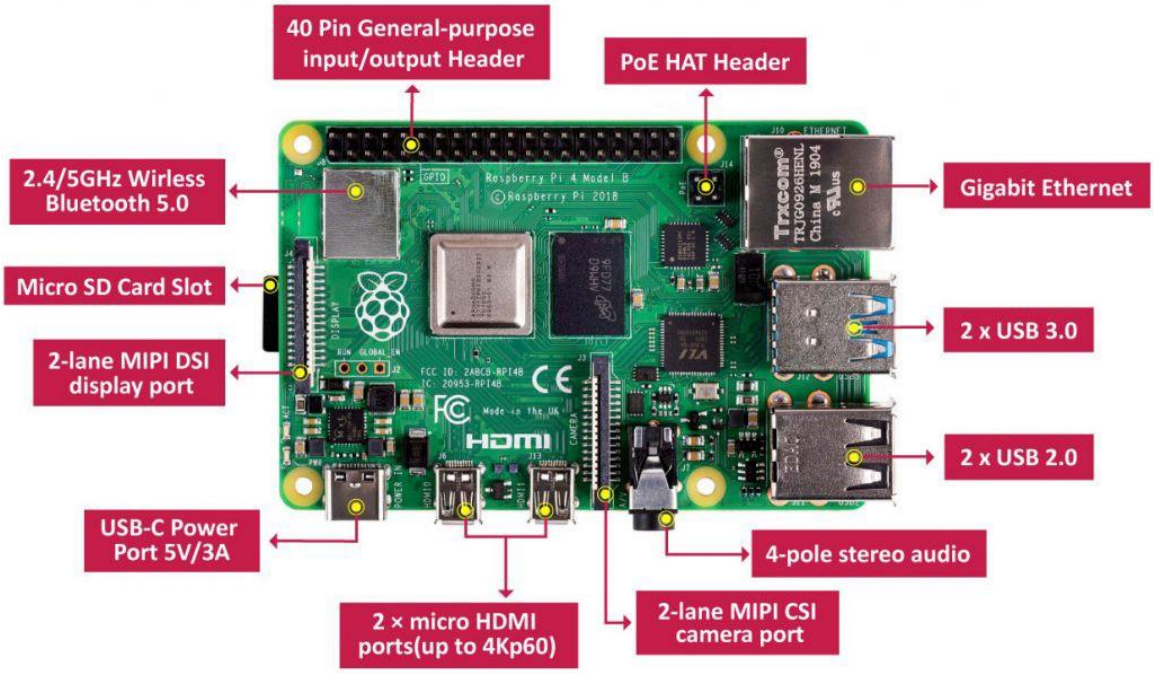


Figure 4-2: Raspberry Pi 4 Model B Controller

4.1.3. Raspberry pi camera module v2

Raspberry Pi foundation currently manufactures two types of camera modules: the 8 MP camera and the 12 MP High Quality (HQ) camera. The 8 MP camera also has the NoIR version which is a night-vision camera. The Raspberry Pi Camera Module 2 (8 MP) also known as the v2 Camera Module has a Sony IMX219 8-megapixel sensor which enables it to take high resolution photos and videos. The Raspberry Pi Camera v2 can be connected to the CSI port on the board through a ribbon connector. Figure 4-4 shows the Raspberry Pi Camera v2 connected to the Raspberry Pi 4 controller board. The camera is accessed through either the MMAL library, V4L library or the Picamera Python library.

The raspberry Pi controllers installed on the mobile robots have the Ubuntu Mate 20.08 (arm64) OS installed on them which is a 64-bit operating system. MMAL (Multimedia Abstraction Layer) is a C library designed by Broadcom, which is used by the Videocore IV GPU on Raspberry Pi. Raspicam commands such as raspistill, raspivid and raspiyuv which are the command line tools for operating the camera module, require the MMAL. Since the Raspberry Pi 4 controllers are 64-bit controllers, the

Figure 4-3: Raspberry Pi 4 Model B powered by the 12V/5A connector on Kobuki via the LM2956 Buck Converter

arm64 version of Ubuntu Mate suits them well in terms of performance, however the MMAL library is a 32-bit library which cannot run on a 64-bit OS. For this reason, the Raspberry Pi 4 board with the v2 camera interfaced to it, needed to have the 32-bit version of the Ubuntu Mate OS (armhf).

4.1.4. Marvelmind indoor navigation system

The Marvelmind indoor positioning system provides location data with a precision of ±2 cm. The system can be used to get real-time position data from drones, UAVs, UGVs etc. Other applications can be in industrial/ construction facilities to keep track of workers with the beacons installed on their helmets or construction vehicles such as forklifts etc.

The system consists of mobile and stationary beacons connected over radio network with a modem connected to a PC. The positioning is based on trilateration of ultrasonic pulses being emanated from either the stationary or mobile beacons (depending on the chosen network topology).

Stationary beacons serve as a global position reference, preferably installed on ceiling/ vantage point with unobstructed view of all the other beacons. The Mobile beacons are supposed to be installed on moving equipment such as robots, copters, UAVs etc.

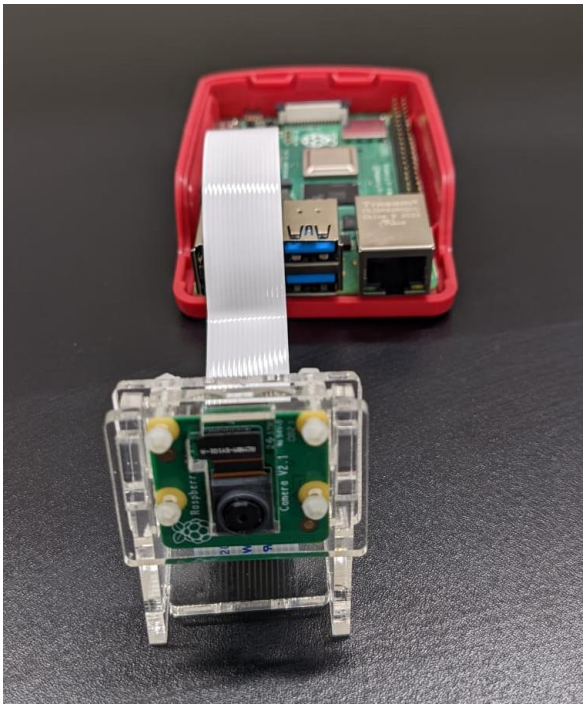


Figure 4-4: Raspberry Pi 4 with Camera Module v2 attached through a ribbon cable

A variety of beacons are manufactured by Marvelmind with different features and specifications. The industrial beacons are dedicated transmitters or receivers and are more sturdy, thus suitable for outdoor use. There are Super Beacons, which can act as both receivers and transmitters and have good sensitivity and reception. The Super Beacons also have an option of external microphone extension for better sound reception. There are dedicated transmitter or receiver mini beacons too. The mini-RX beacon has increased sensitivity since it acts as a dedicated receiver. The HW v4.9 and v5.1 beacons are somewhat similar, with the 5.1 being an improved version of the HW v4.9 beacons.

The HW v4.9 beacon set has been used here for navigation and localization of the mobile robots. The HW v4.9 beacons can act as a receiver or transmitter, depending on the chosen architecture. It is cheaper than the industrial version and more suitable for academic/ hobby projects. It supports either 433 or 915/868 MHz frequency bands. The

reception range of HW v4.9 beacons is around 30 metres and they come equipped with a 1000mAh Lithium Polymer battery. They can be purchased as a set of 1 modem and 5 beacons. The 433 MHz HW v4.9 starter set with 1 modem and 5 v4.9 beacons can be seen in Figure 4-5. The beacons can be charged with a micro-USB cable.

The proposed setup with each of the Kobuki turtlebots equipped with its own raspberry Pi 4 controller and a mobile beacon; and a pair of stationary beacons forming the reference axis; can be seen in Figure 4-6.

Figure 4-5: Marvelmind HW v4.9 Beacon Starter Set

4.1.4.1. Marvelmind dashboard

Marvelmind has its visualization software application by the name of Dashboard, which generates a map of the activated stationary and mobile beacons. It has many useful features such as mobile beacon tracking and other tuning options. The beacon addresses can be set using the settings pane on the right. The hedgehog mode can be enabled to make the beacon function as a mobile beacon depending on your preference. If the hedgehog mode is disabled, the beacon behaves as a stationary beacon.

A few important steps that need to be followed when setting up the system are as follows. It is important to activate the stationary beacons first and then freeze the map. After the map is frozen, the mobile beacons can be activated and become a part of the map or submap. It is also important to erase the map currently saved in the modem, whenever there is a change in the positions of the stationary beacons. If the stationary

beacons are moved, the old map needs to be erased and a new map needs to generated and frozen in the Dashboard. This can be done easily using the Save map, Erase map and Freeze/Unfreeze map buttons provided on the application.

Another important feature is the battery status of each beacon which is visible in the settings pane. In order to have good beacon performance and to avoid fluctuations, it is important to have the beacons charged to at least 3.7 V. The Clear tab can be used to clear any earlier paths created by the mobile beacons. We can zoom in or out of the map area depending on the area of interest.

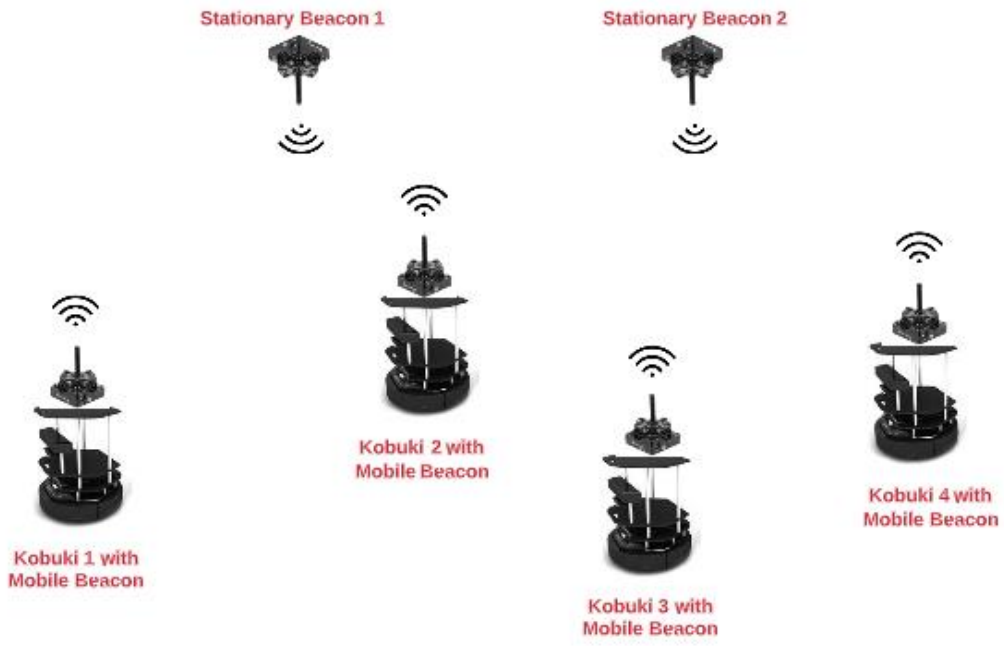


Figure 4-6: Marvelmind beacons set-up for positioning of Kobuki turtlebots

There is a diagnostics message window at the top, where important messages and warnings (in case of wrong setup) appear. These messages/ warnings are color-coded to highlight the severity of the issue: Orange signifies a minor problem and Red shows that there is some major issue. The beacon markers also change their colors to red/orange to highlight any issues. In normal operation, the mobile beacons appear as blue (with green outer ring) while the stationary ones are green.

There are many other advanced Dashboard settings and features which can be explored further by reading the manual and by experimenting with the system. Figure 4-7 shows

a screenshot of the Marvelmind Dashboard application with the mobile beacons in blue tracing a green path, and the stationary beacons in green.

4.2. Robot Operating System- ROS

ROS is an open-source, software framework for programming robots, sensors, actuators etc. ROS makes communication between various system components easier because of its publish/ subscribe model. Each executable piece of code becomes a “node” which can either “publish” or initiate messages to other nodes or “subscribe” to or consume messages being generated by other nodes. The nodes do not have to be a part of the same system nor do they need to be of similar architectures.

All communication between nodes is handled by the ROS Master which assigns IP addresses to all nodes comprising the system. In the beginning, all nodes need to register with the Master in order to make inter-node communication possible. For instance, a camera node could be publishing image data to the other nodes, such as an image-processing node, which has subscribed to this “service”- i.e., the image data being generated by the camera node.

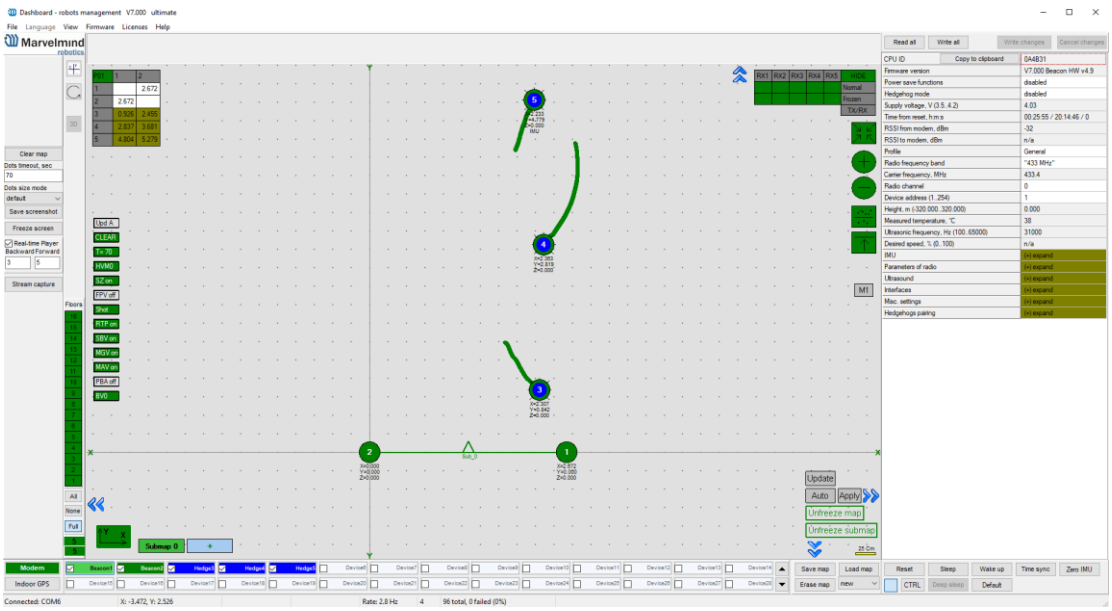


Figure 4-7: Marvelmind Dashboard Application

RQT_GRAPH in ROS represent the structure of how “topics” are being passed around the system. The data being published by a node is packaged as a topic in ROS. The

rosrun rqt_graph rqt_graph command helps the user visualize the nodes and the topics being published and subscribed to by each node.

The pre-built ROS software packages for the devices being used can be installed easily. There is a ROS package available for Kobuki robots, developed by the manufacturer. This package enables position and velocity control of the robot in addition to other features. There is a special topic by the name of /tf which keeps track of all the system-related coordinate frames. For instance, if we are running a ROS package on each of the Kobuki turtlebots, any moving component on a given Kobuki turtlebot (such as the four wheels) will have a corresponding tf which describes the associated coordinate frame. Similarly, there is a ROS package available for the Marvelmind indoor positioning system.

While the standard Debian Raspberry Pi OS which comes as the main installation option in the Raspberry Pi installer is commonly installed on Raspberry Pi Boards, it does not work well with ROS. In order to avoid installation and compatibility issues, Ubuntu Mate 20.04 (arm64) OS was installed on the raspberry pi controllers. The Ubuntu Mate software image was downloaded from the official Ubuntu Mate website and then flashed onto the SD Card using balenaEtcher. It is also important to choose SD cards with sufficient memory and speed.

Since Ubuntu Mate 20.04 (arm64) is the Operating System (OS) installed on the Raspberry Pi 4 controller boards attached to the mobile robots, a compatible version of ROS needed to be installed. ROS Noetic Ninjemys is the 13th ROS LTS distribution which is compatible with the Ubuntu Mate 20.04 OS.

4.2.1. Kobuki ROS package

The Kobuki ROS package can be easily installed on ROS Kinetic by using the sudo apt install command. For ROS Melodic however, the package for Kobuki needs to be built from source. The installation on ROS Noetic is slightly more complicated as the noetic branch of the Kobuki package has not been updated.

The first step is to make a ROS workspace for the kobuki package installation. A src directory is created inside this workspace and then we run the catkin_make command. The next step is to clone the melodic branch of the kobuki repository from https://github.com/yujinrobot/kobuki using the git clone command, into the src

directory. Another repository from https://github.com/yujinrobot/yujin_ocs is cloned into the src folder. The next step is to delete all folders except the yocs_cmd_vel_mux, yocs_controllers and yocs_velocity_smoother from the installed yujin_ocs folder, since these are the only required dependencies of the kobuki package. Next, the liborocos-kdl package is installed by typing command: sudo apt install liborocos-kdl-dev. The final step is to run rosdep which will find and install all dependencies. This is done by going to the root of the kobuki workspace and typing command: rosdep install --from-paths src –ignore-src -r -y. This will ensure that all dependencies of the Kobuki package are installed.

Since the Kobuki package has been built from source in ROS Noetic, the setup file from the kobuki workspace needs to be sourced before running the package. This is done by running the following commands:

```
source ~kobuki_ws/dev/setup.bash

rosrun kobuki_node minimal.launch --screen
```

where minimal.launch is the launch file. A launch file can run a number of nodes simultaneously.

Once the kobuki node is running, we can use the commands rostopic list to see the name of the topics to which we can publish or subscribe. Usually, the sensor data is subscribed to, to get information about the sensor status and the actuators such as motors can be given commands by publishing to their respective topics.

- /diagnostics
- /diagnostics_agg
- /diagnostics_toplevel_state
- /joint_states
- /mobile_base/commands/controller_info
- /mobile_base/commands/digital_output
- /mobile_base/commands/external_power
- /mobile_base/commands/led1
- /mobile_base/commands/led2
- /mobile_base/commands/motor_power

- /mobile_base/commands/reset_odometry
- /mobile_base/commands/sound
- /mobile_base/commands/velocity
- /mobile_base/controller_info
- /mobile_base/debug/raw_control_command
- /mobile_base/debug/raw_data_command
- /mobile_base/debug/raw_data_stream
- /mobile_base/events/bumper
- /mobile_base/events/button
- /mobile_base/events/cliff
- /mobile_base/events/digital_input
- /mobile_base/events/power_system
- /mobile_base/events/robot_state
- /mobile_base/events/wheel_drop
- /mobile_base/sensors/core
- /mobile_base/sensors/dock_ir
- /mobile_base/sensors/imu_data
- /mobile_base/sensors/imu_data_raw
- /mobile_base/version_info
- /mobile_base_nodelet_manager/bond
- /odom
- /tf

4.2.2. Marvelmind ROS package

The ROS package for marvelmind beacons called marvelmind_nav, is available online which can provide beacon location data over ROS. The marvelmind_nav ROS package is available at the link:

https://bitbucket.org/marvelmind_robotics/ros_marvelmind_package

This package is compatible with both ROS Melodic and ROS Noetic.

To install the package, first change directory to the source folder of the catkin workspace by typing the command in terminal:

```
cd ~/catkin_ws/src
```

Then create a directory by the name of marvelmind_nav:

```
mkdir marvelmind_nav
```

The next step is to clone the package from the link given above into the newly created package. Then finally, the catkin_make command is executed from the catkin workspace which installs the package.

The Dashboard app is opened on either Windows or Linux based system to build a map of stationary and mobile beacons. The stationary beacons are switched on first, the map is frozen on Dashboard and then the mobile beacons (hedgehogs) are activated. The mobile beacons are then connected via a USB cable to the Raspberry Pi controller. The virtual serial port used by the mobile beacon can be found by running the command:

```
ls /dev/ttyACM*
```

Usually the beacon connects to /dev/ttyACM0 which is the port used by default by the Marvelmind ROS package. However, if the port cannot be found, the following command can be tried:

```
ls /dev/ttyUSB*
```

The following command can be used to run the package:

```
rosrun marvelmind_nav hedge_rcv_bin
```

where marvelmind_nav is the name of the installed package and hedge_rcv_bin is a node which receives position data from all the hedgehogs. If the serial port is some port other that the ttyACM0 port, the port name needs to be appended to the command. The baud rate can also be specified:

```
rosrun marvelmind_nav hedge_rcv_bin /dev/ttyACM1 115200
```

Sometimes there might be port access issues if you do not have the permission to access it. There will be an error like “unable to open serial connection” even if the port is visible. To change the access setting, the following command can be executed before running the node:

```
sudo chmod 0777 /dev/ttyACM0
```

This might need to be done every time before running the node unless a permanent solution is found. When the node begins to run successfully, the beacon position data will be visible on the terminal.

Table 4-1 shows the list of topics on which the hedge_rcv_bin node publishes different kinds of data. These topics can be seen by running command rostopic list (while the hedge_rcv_bin node is running).

Table 4-1: ROS Topics available with the Marvelmind ROS package marvelmind_nav

	Topic	**Message Field**	**Type**	**Description**
1.	**beacon_raw_distance**	address_hedge	uint8	Address of mobile beacon
		address_beacon	uint8	Address of stationary beacon
		distance_m	float64	Raw distance from mobile to stationary beacon, meters
2.	**beacons_pos_a**	Address	uint8	Address of stationary beacon
		x_m	float64	X coordinate, meters
		y_m	float64	Y coordinate, meters
		z_m	float64	Z coordinate, meters
3.	**hedge_imu_fusion**	timestamp_ms	int64	Timestamp of IMU fusion data, milliseconds
		x_m	float64	(x,y,z) coordinates of mobile beacon by IMU fusion
		y_m	float64	
		z_m	float64	
		Qw	float64	Orientation quaternion of mobile beacon (qw,qx,qy,qz)
		Qx	float64	
		Qy	float64	
		Qz	float64	
		Vx	float64	

		Vy	float64	(vx,xy,xz)- speed vector of mobile beacon calculated by IMU fusion
		Vz	float64	
		Ax	float64	(ax,ay,az)- acceleration of mobile beacon
		Ay	float64	
		Az	float64	
4.	**hedge_imu_raw**	timestamp_ms	int64	Timestamp of raw IMU data, milliseconds
		acc_x	int16	(acc_x,acc_y,acc_z)- raw accelerometer data
		acc_y	int16	
		acc_z	int16	
		gyro_x	int16	(gyro_x,gyro_y,gyro_ z)- raw gyroscope data
		gyro_y	int16	
		gyro_z	int16	
		compass_x	int16	(compass_x, compass_y, compass_z)- raw compass data (only for HW4.9 beacons
		compass_y	int16	
		compass_z	int16	
5.	**hedge_pos**	timestamp_ms	int64	Timestamp
		x_m	float64	Position (x,y,z) of mobile beacon, meters
		y_m	float64	
		z_m	float64	
		Flags	uint8	Flags of location
6.	**hedge_pos_a**	Address	uint8	Address of mobile beacon
		timestamp_ms	int64	Timestamp (milliseconds)
		x_m	float64	(x,y,z) coordinates of mobile beacon
		y_m	float64	

		z_m	float64	
		Flags	uint8	Flags of location
7.	**hedge_pos_ang**	Address	uint8	Address of mobile beacon
		timestamp_ms	int64	Timestamp (milliseconds)
		x_m	float64	(x,y,z) coordinates of mobile beacon
		y_m	float64	
		z_m	float64	
		Flags	uint8	Flags of location
		Angle	float64	Orientation angle of paired beacons, degrees
8.	**hedge_quality**	Address	uint8	Address of mobile beacon
		quality_percents	uint8	Quality of location, percents
9.	**hedge_telemetry**	battery_voltage	float64	Battery voltage of mobile beacon, volts
		rssi_dbm	int8	RSSI (radio signal strength), dBm
10.	**marvelmind_waypoint**	total_items	uint8	Total number of waypoint program items (N)
		item_index	uint8	Index of this waypoint item (0…N-1)
		movement_type	uint8	Type of action (6 = move to specified point)
		param1	int16	Parameter 1

				x coordinate of waypoint, cm (if movement_type = 6)
		param2	int16	Parameter 2 y coordinate of waypoint, cm (if movement_type = 6)
		param3	int16	Parameter 3 z coordinate of waypoint, cm (if movement_type = 6)

Chapter 5. Results

5.1. MATLAB Simulation Results

Figure 5-1 shows the simulation results of Rendezvous using consensus protocol for a group of six (6) agents / robots (simulated in MATLAB). The black dots denote the starting points of robots (chosen randomly). The black dotted lines display the underlying communication network topology. We have a fixed/ static communication graph as the communication links remain fixed during the manoeuvre and are independent of inter-agent distances or any other factors. It is obvious that we have a cyclic and connected communication graph here. The six coloured lines depict the trajectories of the six robots from starting points to end points. The agents converge at the meeting point and are thus able to achieve rendezvous using consensus protocol.

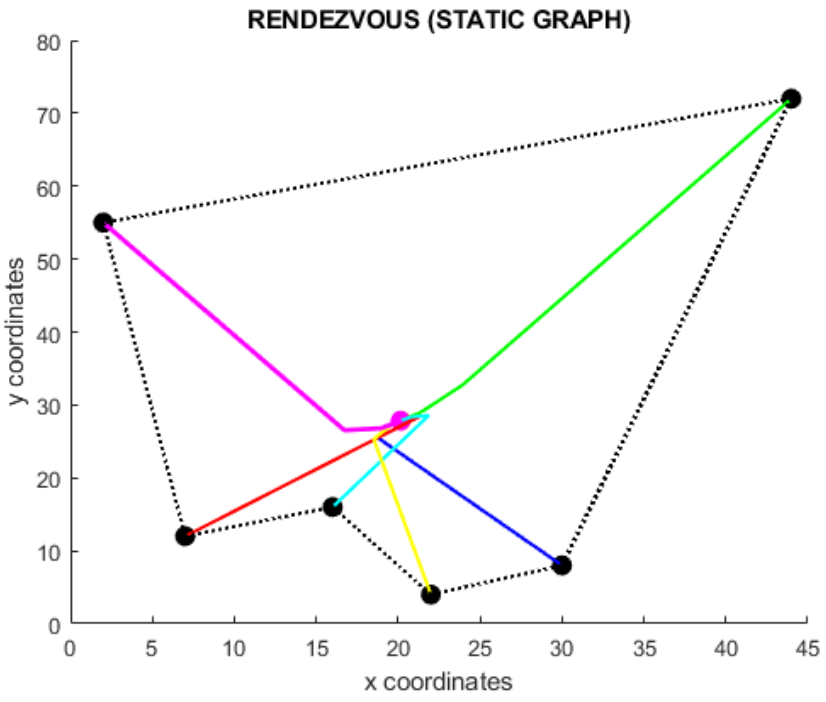


Figure 5-1: Agents converging at the meeting point (MATLAB simulation)

Figure 5-2 shows the simulation results of Formation Control using consensus protocol for a group of six (6) agents / robots (simulated in MATLAB). The black dots denote the starting points of robots (chosen randomly). Here, the desired inter-agent distances are given such that the resulting formation is hexagonal. The black dotted lines display the underlying communication network topology which is a fixed/ static communication graph (independent of changing inter-agent distances, or any other

environmental factors). The graph is cyclic with each node having a degree of 2 (i.e. each agent has 2 neighbours each). The coloured lines represent the individual trajectories whereas the coloured dots represent the endpoints.

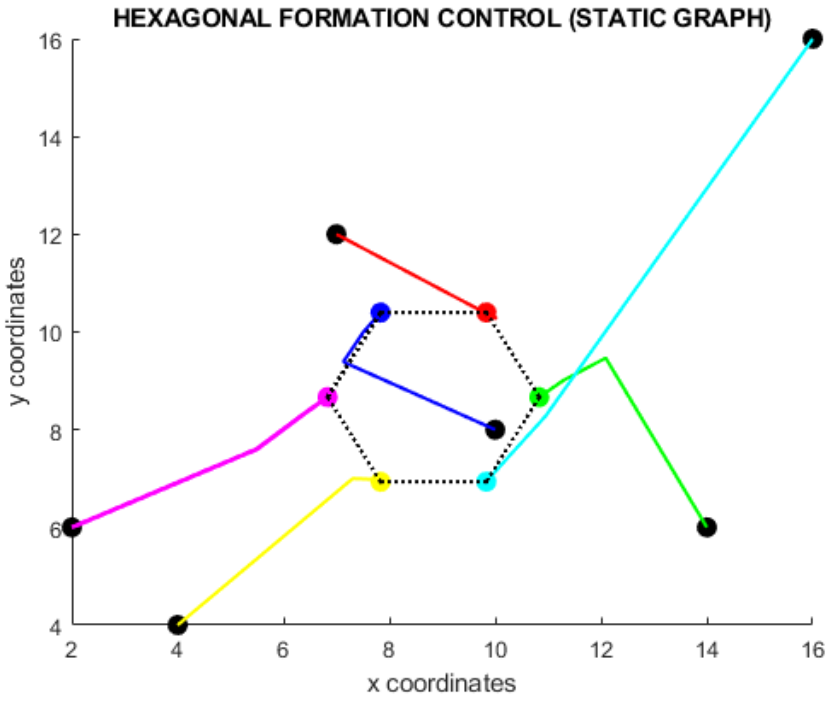


Figure 5-2: Hexagonal Formation (six agents, cyclic graph)

The results and accuracy of the Formation Control and the Gesture Recognition algorithms are discussed separately.

5.2. Hand-Gesture Recognition

The implemented gesture recognition algorithm has advantages in terms of being both scale and rotation-invariant. To test the accuracy and latency of the algorithm, hand images were captured by the Raspberry Pi camera V2 in four orientations, as shown in Figure 5-3. Gestures were made with the back of hand and the palm side facing the camera and the results were calculated to get the detection accuracy percentage.

The algorithm was implemented with the circle and ellipse shapes and the results were calculated for each case. The circle was drawn with radius as given in equation (36). Equations (37) and (38) give the dimensions and orientation values initially used to draw the ellipse- (Ellipse 1). However, the dimensions of the ellipse were further calibrated to achieve improved results (Ellipse 2). These results are tabulated in Table 5-1.

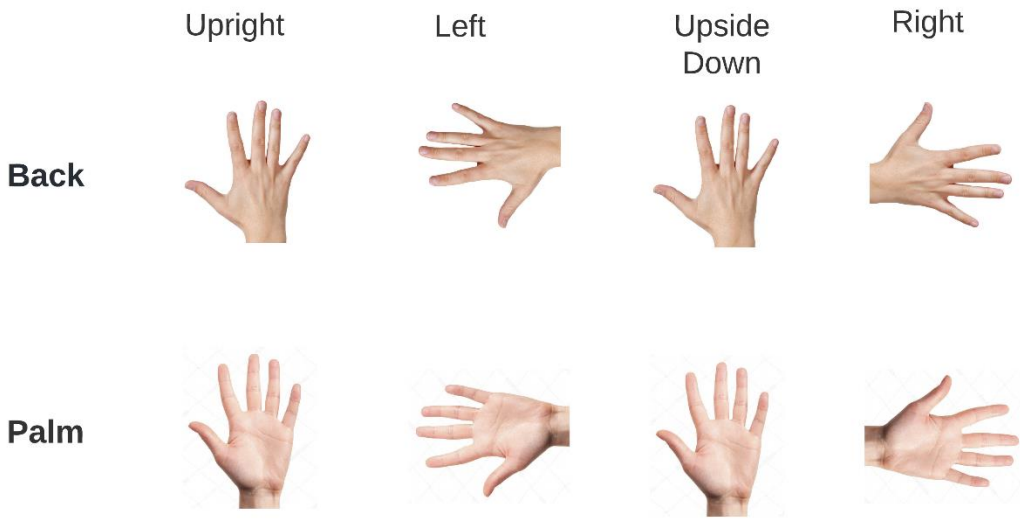


Figure 5-3: Hand orientations used to calculate the accuracy of the algorithm

$$major - axis = 0.68\ l \qquad minor - axis = 1.58(0.32\ l) \tag{39}$$

Table 5-1: Accuracy and Latency calculated for the gesture recognition algorithm

Circle		Ellipse 1		Ellipse 2	
Accuracy	Average Latency	Accuracy	Average Latency	Accuracy	Average Latency
88.75%	2.53 sec	82.5%	2.54 sec	90.8%	2.56 sec

It is apparent that calibrating the dimensions of the ellipse results in improvement in the accuracy of the implemented algorithm. In Figure 5-4, we can see that if a circle is used to intersect the open digits included in the gesture, there is a chance of missing the shortest open digits. A properly aligned ellipse on the other hand, is more suited for this purpose and thus leads to better results.

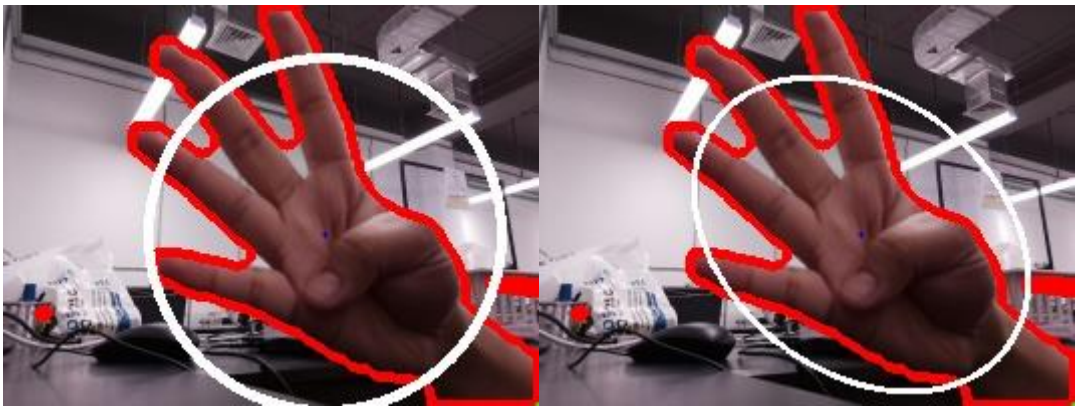


Figure 5-4: Comparison between using a circular and elliptical shape to intersect the digits

5.3. Formation Control

Here, the results of two experiments are discussed. In our implementation of formation control, we are considering complete communication graphs with a Laplacian matrix (4-agent network) given as:

$$L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

While for a 3-agent network, the Laplacian matrix of a complete graph will be:

$$L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

In the first experiment, positions of the three robots forming a triangular formation, as seen in Figure 5-5, were recorded in the Dashboard software and through ROS. For a group of three robots, we can calculate the formation error based on the inter-robot distances. The distance-based error was evaluated as the group of robots begins moving from random positions and forms a triangular formation. The decrease in the formation error with the passage of time is evident from the plots.

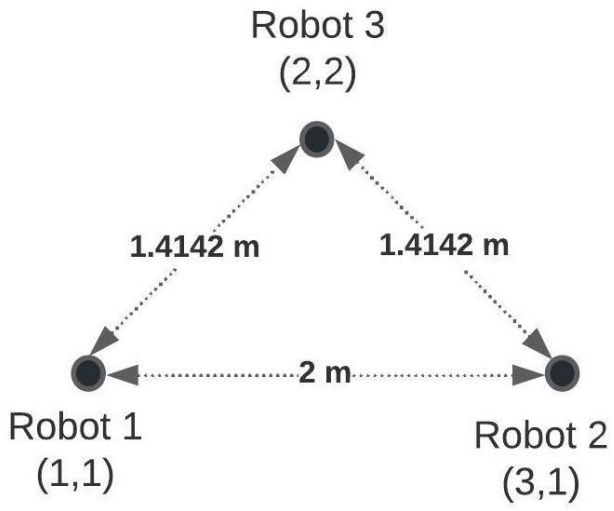


Figure 5-5: Triangular Formation formed by 3 robots

The second formation that was implemented is a linearly horizontal formation comprising four mobile robots as shown in Figure 5-6.

The distance-based error metric for the triangular formation; e_d is given as:

$$e_d = |2 - d_{12}| + |1.4142 - d_{23}| + |1.4142 - d_{13}| \quad (40)$$

Where d_{12} is the distance between robots 1 and 2, d_{23} is the distance between robots 2 and 3 and d_{13} is the distance between robots 1 and 3 in meters.

The actual robot paths for the case of the triangular formation have been plotted in Figure 5-7. The plot of the distance-based error metric for the triangular formation is shown in the Figure 5-8. The error can be normalized by scaling it with the sum of desired inter-agent distances which is plotted in Figure 5-9.

$$e_{norm} = \frac{e_d}{2+1.4142+1.4142} = \frac{e_d}{4.8284} \quad (41)$$

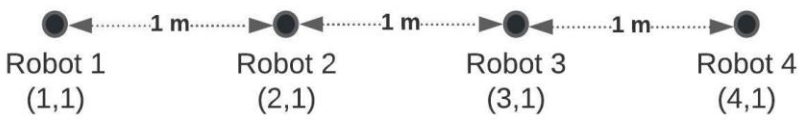


Figure 5-6: Linear Formation formed by four robots

The error metric in the case of a linear formation is calculated similarly. However, with four robots, there can be 6 possible unique pairs of robots and the inter-agent distances need to be calculated for all 6, to ensure that we have a linear formation.

$$e_d = |1 - d_{12}| + |2 - d_{13}| + |3 - d_{14}| + |1 - d_{23}| + |2 - d_{24}| + |1 - d_{34}| \quad (42)$$

The actual robot paths for the case of the linear formation have been plotted in Figure 5-10. The plots of the error metric e_d (versus time) as the group of robots moves from a random configuration to a linear (horizontal) formation is shown in Figure 5-11. It is evident that there is a decrease in error as the robots move into the desired formation. The normalized error in this case, which has been plotted in Figure 5-12, is given as:

$$e_{norm} = \frac{e_d}{1+2+3+1+2+1} = \frac{e_d}{10} \quad (43)$$

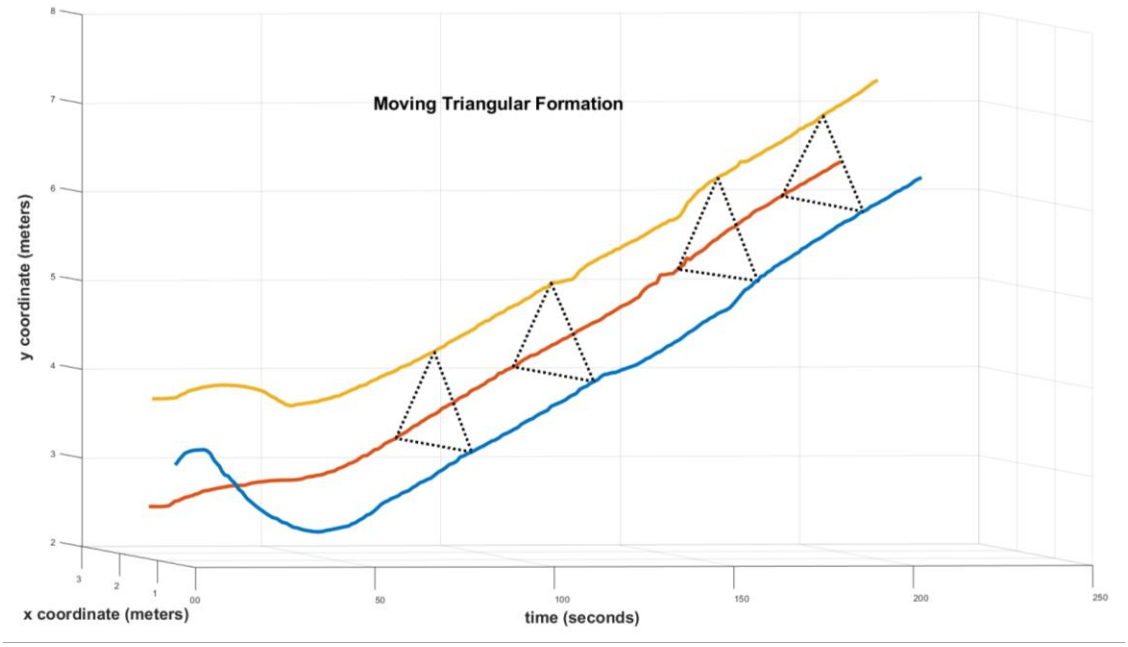


Figure 5-7: Paths of a group of 3 robots assembling into a Triangular Formation, moving along y-axis

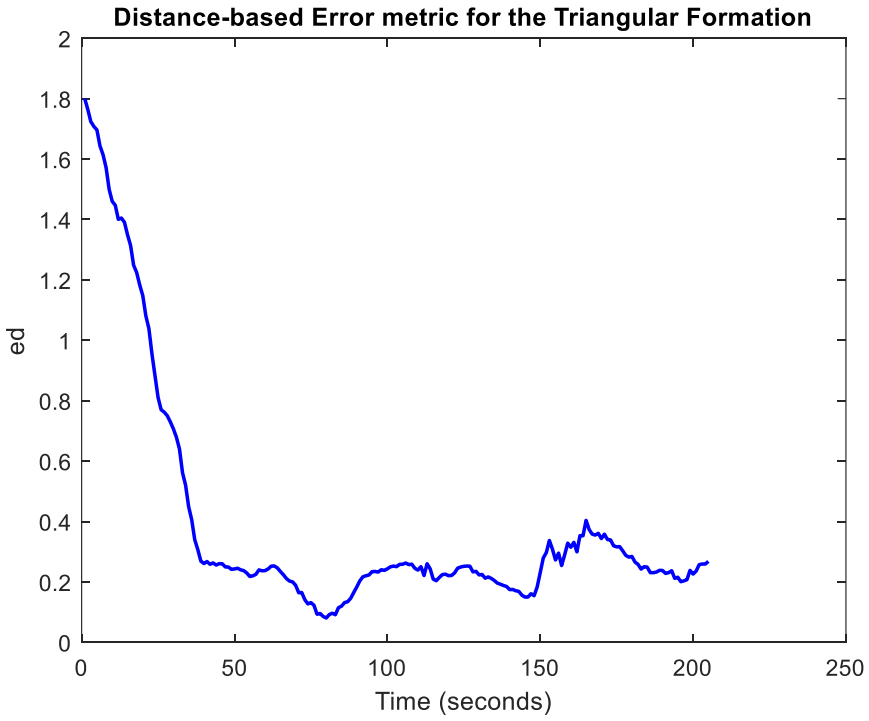


Figure 5-8: Error metric of the Triangular Formation formed by three mobile robots

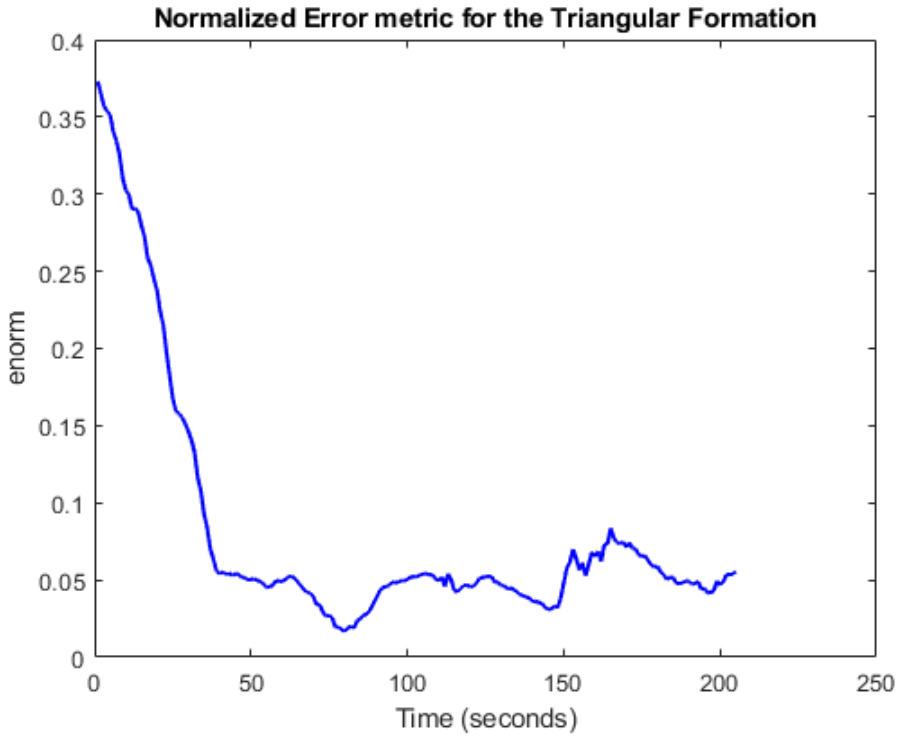


Figure 5-9: Normalized Error of the Triangular Formation

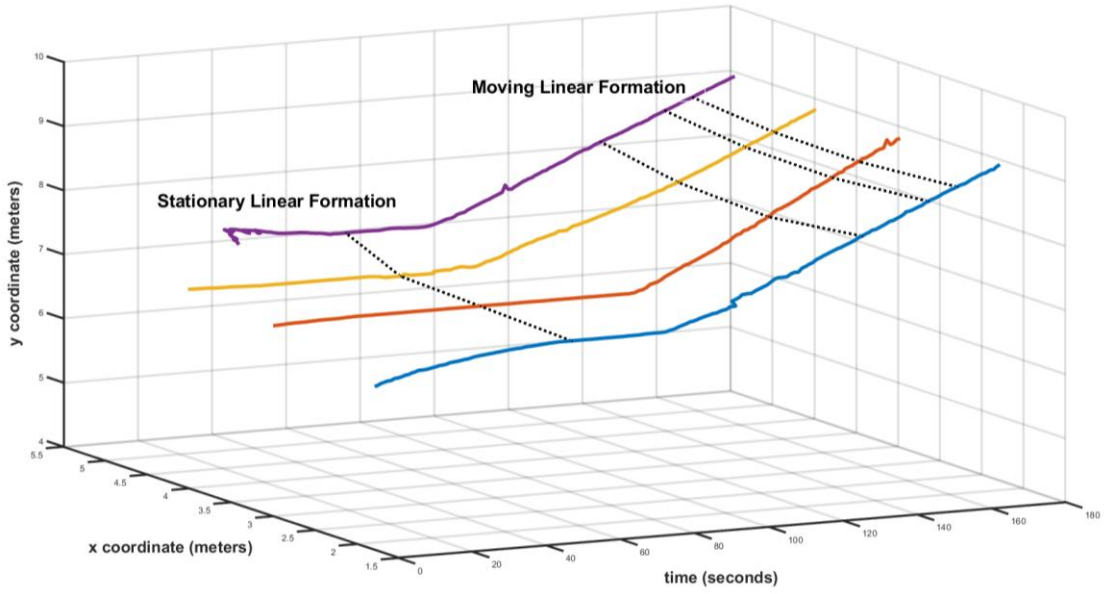


Figure 5-10: Paths of 4 robots forming a linear formation moving along y-axis

Figure 5-13 plots the paths of a group of four Kobuki robots. First, the hand gesture with 1 open finger is recorded by the camera and interpreted correctly, resulting in the

robots assembling into a stationary linear formation along the x-axis. After that, a gesture with 3 open fingers directs the robots to assemble into a stationary linear formation. Then, a hand gesture with 4 open fingers commands the group to form a square formation, which begins to move along the y-axis. Subsequently, a hand gesture with 2 open fingers commands the robots to form a moving linear formation. Finally, a gesture with all 5 open digits gives the halt command to the group of robots.

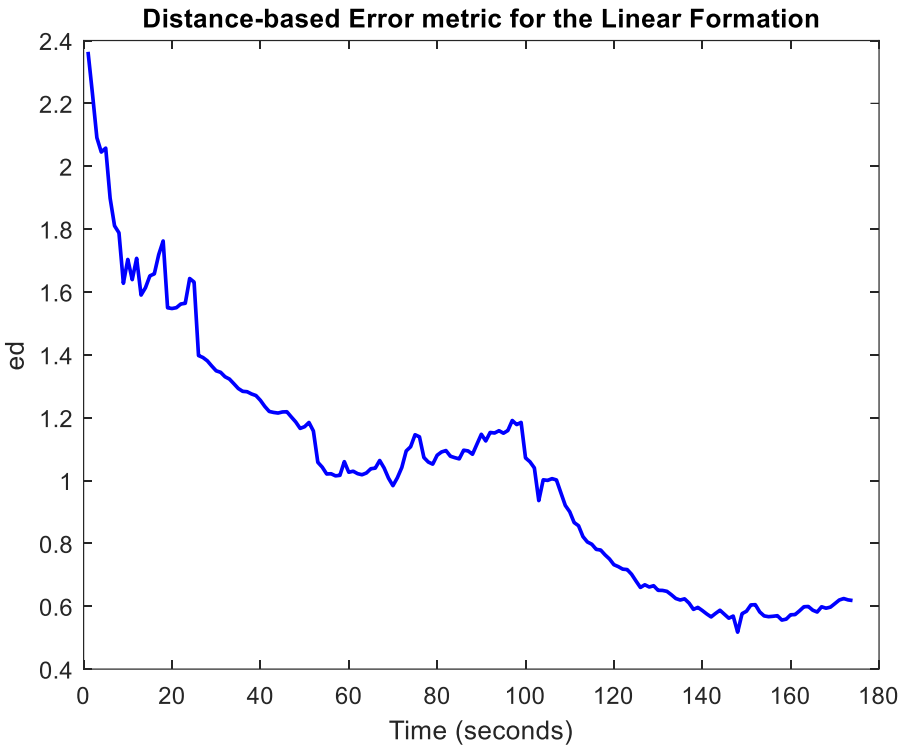


Figure 5-11: Error metrics of the Linear Formation formed by four mobile robots

The sufficiently low latency of the gesture recognition algorithm makes it possible for the group of robots to respond to the given commands in real time. There is a drift observed in the path of the robots over time due to the inherent error in the odometry data received on the Kobuki ROS topic /odom. A technique can be devised to account for the error in the orientation angle of the robot being published by the Kobuki node, so that the formation error is close to zero and to avoid the observed drift in trajectories.

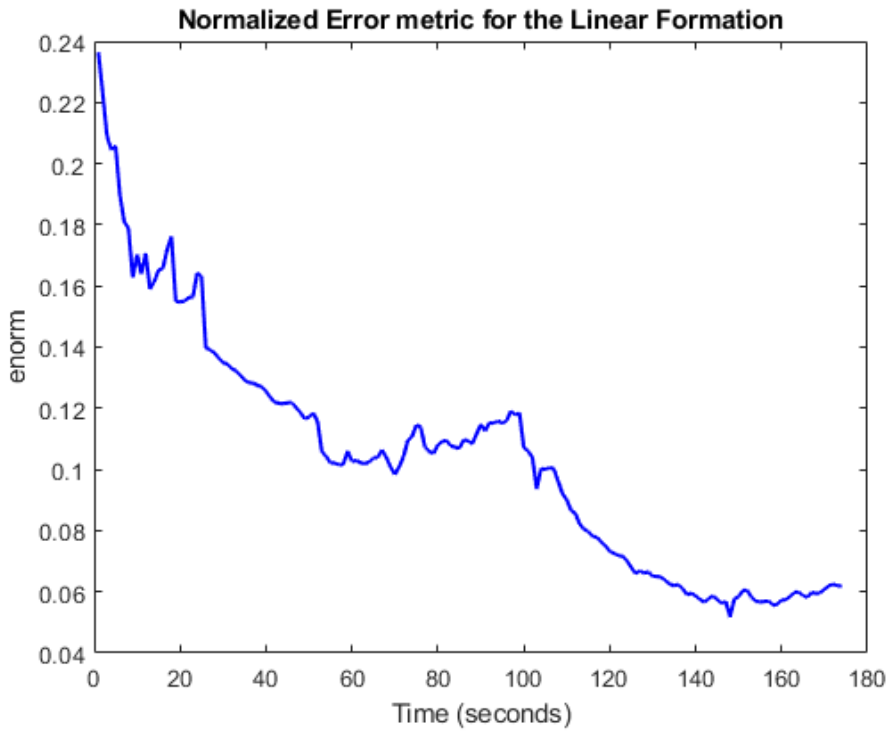


Figure 5-12: Normalized Error of the Linear Formation (4 Robots)

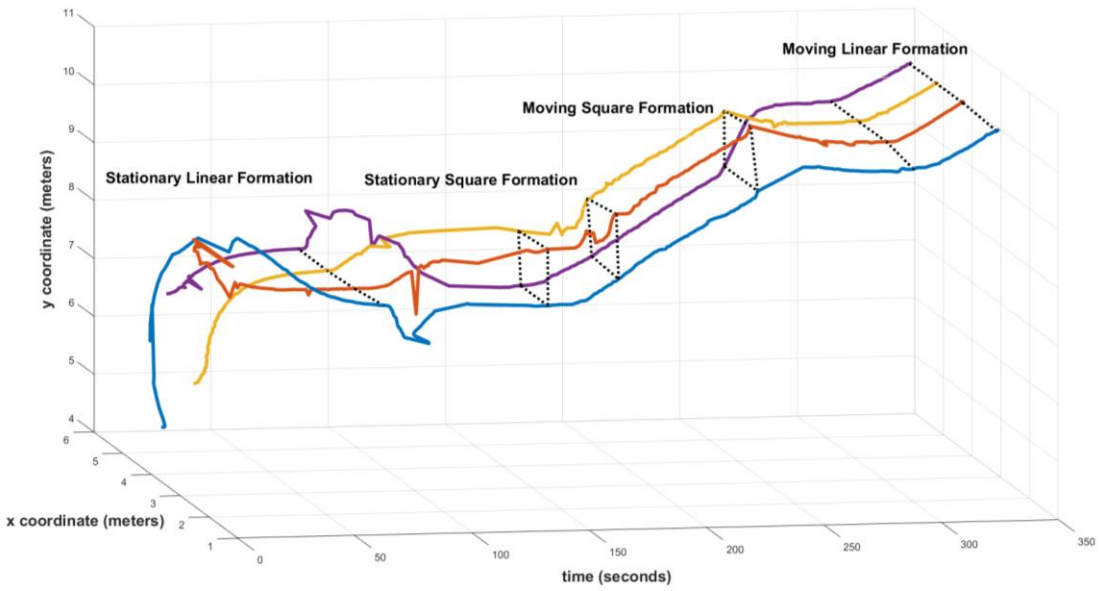


Figure 5-13: Paths of a group of 4 robots controlled by hand gestures and assembling into corresponding formations

Chapter 6. Conclusion and Future Work

Presented work successfully implements gesture-based formation control of a multi-robot system. The Formation control algorithm is distributed and can be easily scaled up to control a large number of robots. The gestures were recognized and interpreted in 2.5 seconds (on average), which allowed for a real-time implementation in ROS as the beacon positions were transmitted over the network with minimal delay. The accuracy was further improved to 90.8% by replacing the circle with an ellipse oriented along the longest open digit present in the hand gesture. In future, further improvements can be made by minimizing beacon fluctuations arising due to noise or disconnect in line-of-sight between the mobile and stationary beacons. This can be achieved by using a suitable filtering technique such as Kalman Filter. Further work can be done on the trajectory and path-planning of the swarm, while keeping the formation intact; or the group of robots can be made more intelligent by adding the capability to react to obstacles in real-time by switching to a different configuration.

The bare-handed gesture recognition was implemented with an accuracy of 90.8% and low enough latency for it to be a useful control technique in real-time. The accuracy can further be improved by combining the digit-counting technique with other feature-extraction techniques to make it more robust. Further improvements are required to counter the deterioration in results caused by sensitivity to change in light conditions and wrong skin-pixel identification.

References

[1] R. Olfati-Saber, J. A. Fax and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," in *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215-233, Jan. 2007.

[2] R. W. Beard, J. Lawton and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," in *IEEE Transactions on Control Systems Technology*, vol. 9, no. 6, pp. 777-790, Nov. 2001.

[3] W. Ren, R. W. Beard and E. M. Atkins, "Information consensus in multivehicle cooperative control," in *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71-82, April 2007.

[4] Y. Zhou, X. Dong, G. Lu and Y. Zhong, "Time-varying formation control for unmanned aerial vehicles with switching interaction topologies," *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 1203-1209.

[5] Z. Lin, L. Wang, Z. Han and M. Fu, "Distributed Formation Control of Multi-Agent Systems Using Complex Laplacian," in *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1765-1777, July 2014.

[6] X. Yi, J. Wei, D. V. Dimarogonas and K. H. Johansson, "Formation Control for Multi-Agent systems with Connectivity Preservation and Event-Triggered Controllers," *International Federation of Automatic Control (IFAC) PapersOnLine*, vol.50, no.1, pp.9367–9373, July 2017.

[7] A. Nikou, C. K. Verginis and D. V. Dimarogonas, "Robust Distance-Based Formation Control of Multiple Rigid Bodies with Orientation Alignment," *International Federation of Automatic Control (IFAC) PapersOnLine*, vol.50, no.1, pp.15458–15463, July 2017.

[8] A. Couture-Beil, R. T. Vaughan and G. Mori, "Selecting and Commanding Individual Robots in a Multi-Robot System," *2010 Canadian Conference on Computer and Robot Vision*, 2010, pp. 159-166.

[9] J. Alonso-Mora, S. Haegeli Lohaus, P. Leemann, R. Siegwart and P. Beardsley, "Gesture based human - Multi-robot swarm interaction and its application to an interactive display," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5948-5953.

[10] W. Kaczmarek, J. Panasiuk and P. Banach, "Industrial Robot Control by Means of Gestures and Voice Commands in Off-Line and On-Line Mode," *Sensors*, vol. 20, (21), pp. 6358, 2020.

[11] A. Suresh and S. Martinez, “Gesture based Human-Swarm Interactions for Formation Control using Interpreters,” *International Federation of Automatic Control (IFAC) PapersOnLine*, vol.51, Issue 34, pp.83-88, 2019.

[12] D. Srivastava, D. M. Lofaro, T. Schuler, D. Sofge and D. W. Aha, “Case-Based Gesture Interface for Multiagent Formation Control,” *International Conference on Case-Based Reasoning (ICCBR 2020)*, pp.295-306.

[13] N. Boonpinon and A. Sudsang, "Formation Control for Multi-Robot Teams Using A Data Glove," *2008 IEEE Conference on Robotics, Automation and Mechatronics*, 2008, pp. 525-531.

[14] J. Nagi, A. Giusti, L. M. Gambardella and G. A. Di Caro, "Human-swarm interaction using spatial gestures," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3834-3841.

[15] H. Han and S. W. Yoon, "Gyroscope-Based Continuous Human Hand Gesture Recognition for Multi-Modal Wearable Input Device for Human Machine Interaction," *Sensors*, vol. 19, (11), 2562, 2019.

[16] B. Sung-Woo and L. Seok-Pil, "Implementation of Hand Gesture Recognition Device Applicable to Smart Watch Based on Flexible Epidermal Tactile Sensor Array," *Micromachines*, vol. 10, (10), pp. 692, 2019.

[17] X. Zhang, Z. Yang, T. Chen, D. Chen and M. Huang, "Cooperative Sensing and Wearable Computing for Sequential Hand Gesture Recognition," in *IEEE Sensors Journal*, vol. 19, no. 14, pp. 5775-5783, July 2019.

[18] T. Müezzinoğlu and M. Karaköse, "Wearable Glove Based Approach for Human-UAV Interaction," *2020 IEEE International Symposium on Systems Engineering (ISSE)*, 2020, pp. 1-6.

[19] D. S. Breland, A. Dayal, A. Jha, P. K. Yalavarthy, O. J. Pandey and L. R. Cenkeramaddi, "Robust Hand Gestures Recognition Using a Deep CNN and Thermal Images," in *IEEE Sensors Journal*, vol. 21, no. 23, pp. 26602-26614, 1 Dec.1, 2021.

[20] G. Tofighi, N. A. Afarin, K. Raahemifar and A. N. Venetsanopoulos, "Hand pointing detection using live histogram template of forehead skin," *2014 19th International Conference on Digital Signal Processing*, 2014, pp. 383-388.

[21] S. Y. Kim, H. G. Han, J. W. Kim, S. Lee and T. W. Kim, "A Hand Gesture Recognition Sensor Using Reflected Impulses," in *IEEE Sensors Journal*, vol. 17, no. 10, pp. 2975-2976, May 2017.

[22] W. K. Wong, F. H. Juwono and B. T. T. Khoo, "Multi-Features Capacitive Hand Gesture Recognition Sensor: A Machine Learning Approach," in *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8441-8450, March 2021.

[23] P. Premaratne, *Human computer interaction using hand gestures*, New York: Springer-Verlag, 2014.

[24] D. Chai and K. N. Ngan, "Face segmentation using skin-color map in videophone applications," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 551-564, June 1999.

[25] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, 2nd ed. Cham, Switzerland: Springer, 2017.

[26] Chin-Chen Chang, I-Yen Chen and Yea-Shuan Huang, "Hand pose recognition using curvature scale space," *International Conference on Pattern Recognition*, vol. 2, pp. 386-389, 2002.

[27] A. Malima, E. Ozgur and M. Cetin, "A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control," *IEEE 14th Signal Processing and Communications Applications*, pp. 1-4, 2006.

[28] J. M. Harris, J. L. Hirst and M. J. Mossinghoff, *Combinatorics and graph theory*, 2nd ed. New York: Springer, 2008.

[29] V. I. Voloshin, *Introduction to Graph Theory*, New York: Nova Science Publishers, 2009.

[30] S. Arumugum, A. Brandstädt, T. Nishizeki and K. Thulasiraman, *Handbook of Graph Theory, Combinatorial Optimization and Algorithms*, Chapman and Hall/ CRC, 2016.

[31] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*, Princeton: Princeton University Press, 2010.

[32] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge University Press, 2012.

[33] R. Merris, “Laplacian matrices of graphs: A survey,” *Linear Algebra and its Applications*, vol. 197–198, pp. 143–176, 1994.

[34] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," in *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465-1476, Sept. 2004.

[35] Yoonsoo Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," in *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 116-120, Jan. 2006.

[36] M. Ji and M. Egerstedt, "Distributed Coordination Control of Multiagent Systems While Preserving Connectedness," in *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693-703, Aug. 2007.

Appendix A

Marvelmind Indoor Navigation System Architectures

There are three architectures in which the beacons can be set up. Each architecture comes with somewhat different features.

1. Non-Inverse Architecture (NIA)

The non-inverse architecture is suitable for applications where the mobile beacons are being installed on a noisy vehicle/ piece of equipment such as drones etc. It comprises 2 or more stationary beacons receiving ultrasonic waves and 1 or more mobile beacons transmitting the waves at the same ultrasonic frequency. The third element in the system is the modem/ router.

2. Inverse Architecture

As opposed to the non-inverse architecture, the Inverse Architecture is suitable for applications where the mobile beacons are installed on quieter machinery/ equipment (not suitable for drones since the mobile beacons are receiving ultrasonic waves). It comprises 2 or more stationary beacons transmitting ultrasonic waves at different frequencies (19 and 25 kHz or 25 and 31 kHz for example).

3. Multi-Frequency NIA

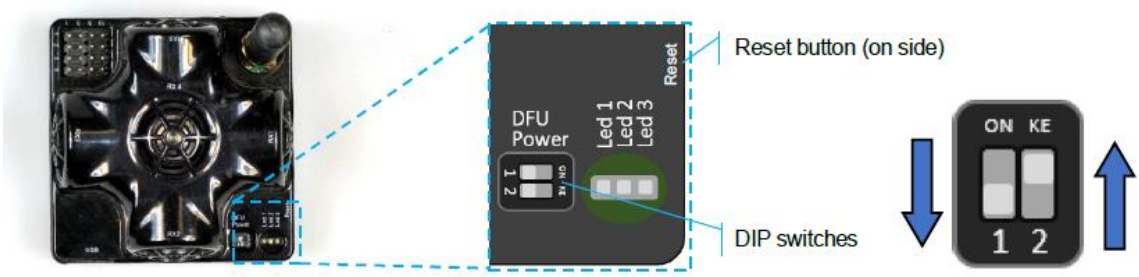
Multi-Frequency NIA is a combination of the NIA and IA architectures, The mobile beacons emit the ultrasonic waves however at different frequencies (unlike NIA).

Marvelmind Beacon Set-up Procedure

The following steps need to be followed while setting up the beacons for the first time:

1. Turn the beacons on by moving the power dip switch (as shown in figure).
2. The updated software pack is available on the marvelmind website. The hex file for the correct type of beacon and architecture needs to be selected for programming the beacons. In this case, since NIA architecture is being implemented with HW v4.9 type of beacons, we selected the file from the folder named “beacon-hw49-nia”. Since the beacons operate at 433 MHz, the file which has the correct frequency needs to be selected.

3. The next step is to install the Dashboard on your OS and run it. The firmware tab on the Dashboard can be used to select the hex file and program the beacon or modem (while connected to the computer via USB).



DIP switches available on the beacons for turning them on/ off

4. Once the latest firmware has been uploaded, the next step is to activate the device by pressing the default button on Dashboard with the beacon or modem still connected to the system.
5. The final step is to press the tiny reset button on the beacons side. Now the beacons and modem are ready to use.

Beacons comparison






	Beacon Mini-RX/Beacon Mini-RX-Outdoor	Beacon Mini-TX/Beacon Mini-TX-Outdoor	Beacon Mini-TX batteryless	Beacon HW v4.9/Beacon HW v4.9 Outdoor	Industrial-TX/Industrial-TX-EX	Industrial-RX/Industrial-RX-EX
Specialty and main use	Universal, multi-frequency and high-sensitivity RX-only beacon	Small TX only beacon	The lightest TX only beacon	Universal dual-use beacon. Support of 433- or 915/868MHz bands	Heavy-duty outdoor/Explosion dangerous environment; RS485 or CAN	Heavy-duty outdoor/Explosion dangerous environment; RS485 or CAN
Mode of operation	RX only	TX only		Dual-use (RX and TX)	TX only	RX only
Range	- Up to 50m[3]	- Up to 30m with DSP RX v5.05 - Up to 20m with Beacon v4.9		- Up to 50m with DSP RX v5.05[3] - Up to 30m with Beacon v4.9[3]	- Up to 30m with Beacon-RX-915-XX	- Up to 30m with Beacon-TX-915-XX
Ultrasonic frequencies	- 19/25/31/45kHz - Several at the same time	- 31/45kHz - Only one HW defined frequency at the time		- 19/25/31/45kHz - Only one frequency at the time	- 19/25/31/45kHz	- 19/25/31/45kHz - Several at the same time
Radio band	915/868MHz	915/868MHz		915/868MHz or 433MHz	915/868MHz or 433MHz	915/868MHz or 433MHz
Power/LiPol battery	USB/750mAh	USB/250mAh	USB/No embedded battery	USB/1000mAh	+5V or 12V or IP67 converter/Optional	+5V or 12V or IP67 converter/Optional
Environmental conditions	- Indoor/Outdoor up to IP67 - t=0..40C[6]	- Indoor/Outdoor[2] - t=0..40C[6]	- Indoor - t=0..40C[6]	- Indoor/Outdoor[2] - t=0..40C[6]	- Outdoor[2]/Intrinsically Safe[5] - t=-20..40C[6]	- Outdoor[2]/Intrinsically Safe[5] - t=-20..40C[6]
Size and weight	47x42x15mm & 25g	35x35x26mm & 19g	35x35x20mm & 12g[9]	55x55x33(64[7])mm & 62/75g	83x58x65mm[8] & 250g	83x58x33mm[8] & 200g
IMU (3D gyro+acc+mag)	Yes (6D)	Yes (6D)	Yes (6D)	No/Yes 9D (for the version with IMU)	Yes (6D)	Yes (6D)
Price	99/129USD	89/129USD	89USD	69/99USD (w/o IMU) 89/119USD (with IMU)	149/189USD	149/189USD

1) Withstand submersion to water on 1m up to 30m (IPx7 requirements)
2) Mild outdoor: occasional rain, dust doesn't kill the device. Performance during this time is no guaranteed
3) 1D mode: RX4 to RX4 sensors; other sensors are disabled
4) Other power options available upon request
5) Exact type of certification shall be discussed separately
6) Temperature range down to -40C is available with external power supply only and upon request
7) With antenna
8) Sizes without mounting holes
9) 6.3g without housing

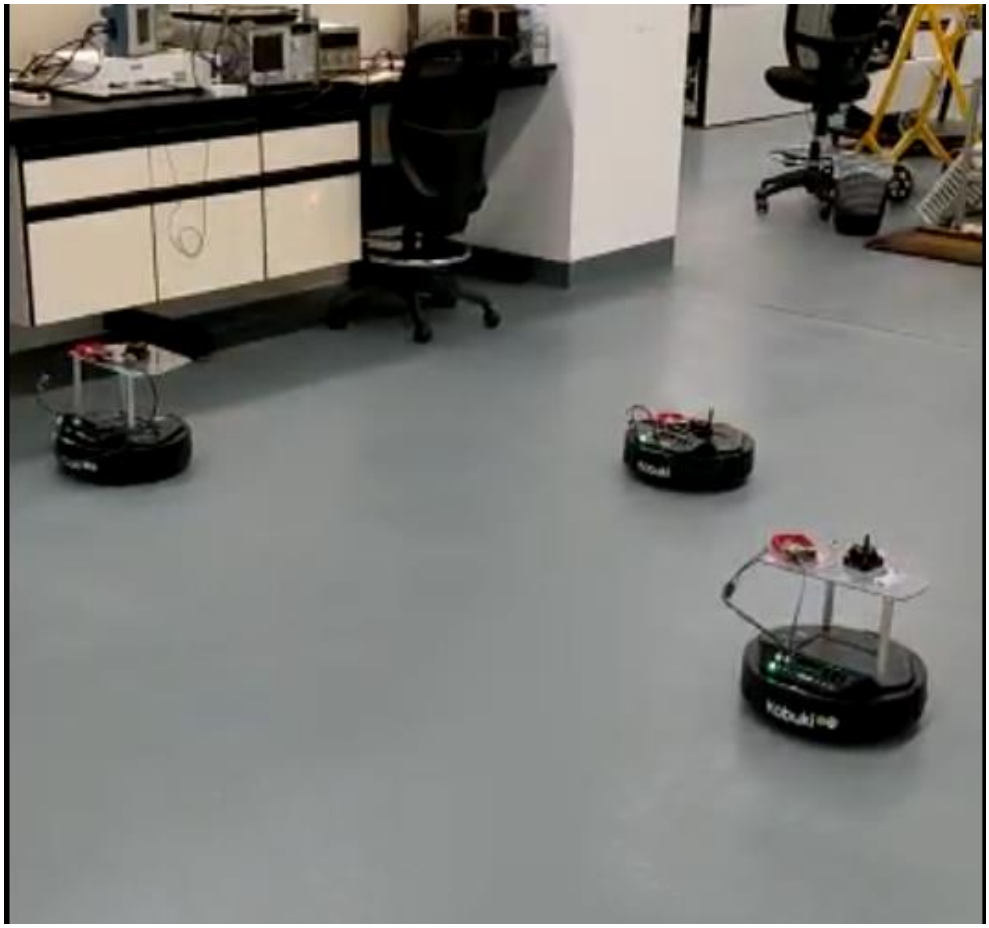
Marvelmind robotics

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	x1	y1	x2	y2	x3	y3	t1	d12	d23	d13	Distance error ed	Coordinate error bw	Coordinate error bw 2 & 3	Coordinate Error bw 1&3	Total Coordinate Error ec	e= ed + ec
2	0.629	3.11	1.328	2.585	1.239	3.808		0.8742	1.226234	0.926987	1.800979206	1.826	1.134	0.692	3.652	5.452979206
3	0.599	3.183	1.328	2.585	1.239	3.808		0.942892	1.226234	0.894553	1.764721138	1.869	1.134	0.735	3.738	5.502721138
4	0.572	3.244	1.328	2.585	1.239	3.809		1.002904	1.227231	0.874136	1.724128135	1.903	1.135	0.768	3.806	5.530128135
5	0.565	3.265	1.332	2.584	1.239	3.809		1.025695	1.228525	0.866148	1.708032188	1.914	1.132	0.782	3.828	5.536032188
6	0.56	3.278	1.334	2.584	1.239	3.81		1.039573	1.229675	0.862592	1.696559826	1.92	1.131	0.789	3.84	5.536559826
7	0.558	3.283	1.381	2.583	1.244	3.813		1.08043	1.237606	0.866889	1.643475146	1.877	1.093	0.784	3.754	5.397475146
8	0.557	3.285	1.404	2.584	1.252	3.815		1.099459	1.240349	0.874028	1.614564269	1.854	1.079	0.775	3.708	5.322564269
9	0.558	3.288	1.445	2.587	1.263	3.818	0:00:15	1.130562	1.244381	0.882001	1.571456253	1.814	1.049	0.765	3.628	5.199456253
10	0.578	3.281	1.513	2.591	1.299	3.832	0:00:17	1.162035	1.259316	0.907437	1.499612012	1.755	1.027	0.728	3.51	5.009612012
11	0.616	3.262	1.581	2.595	1.319	3.848	0:00:18	1.173079	1.280099	0.915208	1.460014694	1.702	0.991	0.711	3.404	4.864014694
12	0.65	3.236	1.63	2.601	1.326	3.859	0:00:19	1.167744	1.29421	0.919296	1.447150339	1.655	0.954	0.701	3.31	4.757150339
13	0.695	3.193	1.699	2.611	1.326	3.887	0:00:21	1.160491	1.3294	0.937975	1.400533538	1.578	0.903	0.675	3.156	4.556533538
14	0.731	3.149	1.722	2.615	1.327	3.897	0:00:22	1.125716	1.341473	0.95641	1.404801131	1.543	0.887	0.656	3.086	4.490801131
15	0.759	3.114	1.75	2.62	1.327	3.907	0:00:22	1.107302	1.354732	0.975435	1.390931842	1.503	0.864	0.639	3.006	4.396931842
16	0.786	3.08	1.798	2.628	1.325	3.918	0:00:23	1.108354	1.372983	0.996376	1.349687436	1.44	0.817	0.623	2.88	4.229687436
17	0.82	3.024	1.835	2.635	1.324	3.929	0:00:24	1.086989	1.391243	1.035877	1.314290687	1.374	0.783	0.591	2.748	4.062290687
18	0.838	2.978	1.884	2.643	1.324	3.935	0:00:25	1.098336	1.408142	1.073334	1.248588811	1.289	0.732	0.557	2.578	3.826588811
19	0.842	2.967	1.901	2.646	1.324	3.941	0:00:26	1.106581	1.417728	1.086738	1.224409004	1.262	0.718	0.544	2.524	3.748409004
20	0.85	2.919	1.934	2.652	1.324	3.948	0:00:26	1.116398	1.432381	1.132924	1.183058915	1.183	0.686	0.555	2.424	3.607058915
21	0.855	2.884	1.961	2.658	1.323	3.95	0:00:27	1.128854	1.44094	1.164208	1.147877799	1.12	0.654	0.598	2.372	3.519877799
22	0.856	2.819	2.002	2.667	1.312	3.954	0:00:28	1.156036	1.460298	1.223177	1.081084639	1.006	0.597	0.679	2.282	3.363084639
23	0.848	2.783	2.018	2.671	1.301	3.955	0:00:29	1.175348	1.470627	1.2565	1.038778866	0.942	0.567	0.719	2.228	3.266778866
24	0.829	2.717	2.053	2.682	1.271	3.955	0:00:30	1.2245	1.494006	1.314537	0.954968202	0.811	0.491	0.796	2.098	3.052968202
25	0.813	2.678	2.088	2.692	1.26	3.953	0:00:31	1.275077	1.508544	1.351086	0.88238091	0.739	0.433	0.828	2	2.88238091
26	0.796	2.638	2.11	2.698	1.255	3.951	0:00:32	1.315369	1.516916	1.390917	0.810629837	0.746	0.398	0.854	1.998	2.808629837
27	0.777	2.602	2.121	2.701	1.242	3.946	0:00:32	1.347641	1.52403	1.422168	0.770155966	0.755	0.366	0.879	2	2.770155966
28	0.756	2.568	2.144	2.707	1.233	3.94	0:00:33	1.394943	1.533039	1.452554	0.762250771	0.751	0.322	0.895	1.968	2.730250771
29	0.73	2.534	2.162	2.712	1.223	3.932	0:00:34	1.44302	1.53952	1.482381	0.750480142	0.746	0.281	0.905	1.932	2.682480142
30	0.707	2.508	2.181	2.719	1.213	3.923	0:00:34	1.489026	1.544875	1.502751	0.730201023	0.737	0.236	0.909	1.882	2.612201023
31	0.695	2.496	2.194	2.723	1.2	3.911	0:00:35	1.51609	1.548993	1.502415	0.706917578	0.728	0.194	0.91	1.832	2.538917578
32	0.649	2.459	2.21	2.728	1.189	3.899	0:00:36	1.584008	1.553603	1.537921	0.679115376	0.708	0.192	0.9	1.8	2.479115376
33	0.635	2.449	2.224	2.732	1.171	3.878	0:00:37	1.614004	1.556318	1.526217	0.640129996	0.694	0.199	0.893	1.786	2.426129996
34	0.561	2.412	2.25	2.737	1.138	3.842	0:00:38	1.719984	1.567664	1.542021	0.561300322	0.636	0.217	0.853	1.706	2.267300322
35	0.534	2.403	2.264	2.739	1.128	3.824	0:00:39	1.762327	1.570898	1.540155	0.520326142	0.606	0.221	0.827	1.654	2.174326142
36	0.482	2.387	2.272	2.742	1.123	3.793	0:00:40	1.824863	1.557178	1.545224	0.449138498	0.565	0.2	0.765	1.53	1.979138498
37	0.45	2.379	2.284	2.743	1.12	3.775	0:00:41	1.869773	1.555609	1.548456	0.405891962	0.53	0.196	0.726	1.452	1.857891962
38	0.402	2.373	2.292	2.742	1.127	3.748	0:00:42	1.925685	1.53924	1.554429	0.339584999	0.479	0.171	0.65	1.3	1.639584999
39	0.373	2.372	2.301	2.743	1.129	3.738	0:00:43	1.963371	1.537403	1.561247	0.306879442	0.443	0.177	0.61	1.23	1.536879442
40	0.349	2.377	2.308	2.753	1.144	3.732	0:00:44	1.994757	1.520966	1.571003	0.268811297	0.417	0.185	0.56	1.162	1.430811297

Sheet1 Sheet2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	x1	y1	x2	y2	x3	y3	x4	y4	t2	d12	d13	d14	d23	d24	d34	Distance error ed	Coordinate Error 1&2	Coord. Er. 1&3	Coord. Er. 1&	Coord Er. 2&	Coord. Er.	Coord Er.	Total ec	e = ed+ec
2	1.828	6.526	3.041	6.947	4.058	7.077	3.45	8.034	1:17	1.2839821	2.297064	2.214712	1.025275	1.1614	1.133805	2.364013653	0.634	0.781	2.886	0.147	2.548	1.349	8.345	10.7090137
3	1.833	6.539	3.044	6.949	4.055	7.075	3.52	8.05	1:18	1.278523	2.285734	2.264749	1.018821	1.19949	1.112138	2.23097621	0.621	0.758	2.824	0.137	2.499	1.44	8.279	10.5099762
4	1.841	6.559	3.045	6.95	4.055	7.073	3.607	8.04	1:19	1.2658977	2.272882	2.304803	1.017462	1.226354	1.065736	2.090820537	0.595	0.728	2.715	0.133	2.405	1.519	8.095	10.1858205
5	1.843	6.573	3.045	6.953	4.06	7.07	3.648	8.03	1:20	1.2606363	2.272025	2.319671	1.021721	1.234317	1.044674	2.045068732	0.582	0.714	2.652	0.132	2.357	1.548	7.985	10.0300687
6	1.846	6.585	3.046	6.958	4.062	7.067	3.65	8.02	1:21	1.256634	2.267814	2.305134	1.02183	1.221745	1.038245	2.057645023	0.573	0.698	2.631	0.125	2.349	1.541	7.917	9.97464502
7	1.848	6.59	3.047	6.959	4.062	7.067	3.76	8.01	1:22	1.2544967	2.264801	2.381626	1.02073	1.270028	0.990178	1.898195508	0.568	0.691	2.508	0.123	2.23	1.641	7.761	9.65919551
8	1.855	6.611	3.047	6.961	4.061	7.064	3.83	8.01	1:23	1.242322	2.252031	2.420295	1.019218	1.309003	0.973795	1.810477961	0.542	0.659	2.424	0.117	2.163	1.715	7.62	9.43047796
9	1.856	6.615	3.046	6.961	4.06	7.061	3.843	8.01	1:24	1.2392804	2.248673	2.427796	1.018919	1.317426	0.973494	1.78815751	0.536	0.65	2.408	0.114	2.152	1.732	7.592	9.38015751
10	1.858	6.628	3.046	6.962	4.06	7.058	3.98	8	1:25	1.2340583	2.243592	2.526909	1.018534	1.396352	0.945391	1.627532251	0.522	0.632	2.25	0.11	2.008	1.862	7.384	9.01153225
11	1.862	6.644	3.046	6.963	4.061	7.056	3.94	7.99	1:26	1.2262206	2.237263	2.475843	1.019252	1.361604	0.941805	1.703482833	0.503	0.611	2.268	0.108	2.04	1.813	7.343	9.04648283
12	1.865	6.65	3.05	6.964	4.06	7.056	3.985	7.99	1:27	1.225896	2.232232	2.507987	1.014181	1.388129	0.937006	1.639187492	0.499	0.601	2.22	0.102	1.999	1.859	7.28	8.91918749
13	1.87	6.667	3.051	6.965	4.061	7.056	4	7.95	1:28	1.2180168	2.225264	2.486562	1.014091	1.367781	0.896079	1.706950751	0.479	0.58	2.153	0.101	1.945	1.833	7.091	8.79795075
14	1.874	6.677	3.05	6.968	4.062	7.054	4.088	7.94	1:29	1.2114689	2.220242	2.548914	1.015648	1.422051	0.886381	1.590011659	0.467	0.565	2.049	0.098	1.848	1.86	6.887	8.47701166
15	1.876	6.696	3.049	6.971	4.062	7.051	4.072	7.94	1:30	1.2048045	2.214638	2.523876	1.016154	1.409074	0.889056	1.613589677	0.448	0.541	2.048	0.093	1.866	1.879	6.875	8.48858968
16	1.878	6.694	3.049	6.972	4.063	7.048	4.06	7.93	1:31	1.2035468	2.213491	2.507752	1.016844	1.392798	0.882005	1.651327033	0.449	0.539	2.054	0.09	1.871	1.879	6.882	8.53332703
17	1.878	6.698	3.049	6.971	4.062	7.046	4.08	7.91	1:32	1.2024018	2.211551	2.513513	1.015773	1.394519	0.864187	1.657506753	0.444	0.532	2.01	0.088	1.833	1.846	6.753	8.41050675
18	1.881	6.706	3.049	6.971	4.06	7.045	4.077	7.88	1:33	1.1976849	2.205212	2.490119	1.013705	1.372248	0.835173	1.71906184	0.433	0.518	1.978	0.085	1.807	1.818	6.639	8.35806184
19	1.884	6.716	3.048	6.972	4.061	7.043	4.075	7.86	1:34	1.1918188	2.201422	2.471683	1.015485	1.357672	0.81712	1.762250821	0.42	0.504	1.953	0.084	1.79	1.803	6.554	8.31625082
20	1.887	6.724	3.047	6.973	4.059	7.041	4.202	7.85	1:35	1.1864236	2.195011	2.574316	1.014282	1.450225	0.821541	1.549634192	0.409	0.489	1.811	0.08	1.654	1.666	6.109	7.65863419
21	1.89	6.736	3.048	6.976	4.054	7.04	2.208	7.84	1:36	1.182609	2.185249	1.148886	1.008034	1.205029	2.011894	4.033869195	0.398	0.468	3.786	0.07	1.96	1.646	8.328	12.3618692
22	1.892	6.739	3.049	6.978	4.052	7.04	2.215	7.83	1:37	1.1814271	2.180872	1.137809	1.004914	1.19225	1.999667	4.036821092	0.396	0.461	3.768	0.065	1.956	1.627	8.273	12.3098211
23	1.896	6.746	3.051	6.978	4.055	7.039	2.22	7.821	1:38	1.17807	2.178791	1.122765	1.005851	1.183727	1.99468	4.050900481	0.387	0.452	3.751	0.065	1.951	1.617	8.223	12.2739005
24	1.903	6.758	3.053	6.98	4.057	7.038	2.23	7.81	1:39	1.1712318	2.172122	1.10165	1.005674	1.168858	1.983409	4.061929521	0.372	0.434	3.725	0.062	1.949	1.599	8.141	12.2029295
25	1.905	6.765	3.053	6.98	4.059	7.037	2.24	7.76	1:40	1.1679593	2.171106	1.049881	1.007614	1.126663	1.957419	4.12755406	0.363	0.426	3.66	0.063	1.91	1.542	7.964	12.0915541
26	1.905	6.77	3.053	6.98	4.061	7.036	4.257	7.75	1:41	1.1670493	2.172347	2.548	1.009554	1.429166	0.740413	1.631371157	0.358	0.422	1.628	0.064	1.51	1.518	5.5	7.13137116
27	1.906	6.777	3.052	6.98	4.06	7.035	4.372	7.742	1:42	1.1638406	2.169396	2.64809	1.009499	1.524154	0.772783	1.397709999	0.349	0.412	1.499	0.063	1.387	1.395	5.105	6.50271
28	1.908	6.788	3.051	6.98	4.061	7.035	4.39	7.723	1:43	1.1590138	2.167122	2.652272	1.011496	1.531329	0.762617	1.391413162	0.335	0.4	1.453	0.065	1.349	1.359	4.961	6.35241316
29	1.909	6.791	3.05	6.98	4.062	7.034	4.409	7.704	1:44	1.1565474	2.16667	2.661498	1.01344	1.539824	0.754526	1.38081005	0.33	0.396	1.413	0.066	1.311	1.323	4.839	6.21981005
30	1.915	6.799	3.051	6.981	4.062	7.034	4.426	7.688	1:45	1.1504869	2.159823	2.663727	1.012388	1.546116	0.748473	1.364381818	0.318	0.382	1.378	0.064	1.279	1.29	4.711	6.07538182
31	1.921	6.809	3.052	6.98	4.062	7.034	4.443	7.67	1:46	1.143854	2.15279	2.664921	1.011443	1.552733	0.741389	1.34904367	0.302	0.366	1.339	0.064	1.245	1.255	4.571	5.92004367
32	1.922	6.816	3.052	6.979	4.063	7.033	4.456	7.654	1:47	1.1416957	2.151969	2.66897	1.012441	1.557832	0.734908	1.3443955	0.293	0.358	1.304	0.065	1.217	1.228	4.465	5.8093955
33	1.925	6.822	3.052	6.979	4.062	7.034	4.47	7.639	1:48	1.1378831	2.14749	2.672922	1.011496	1.564073	0.729718	1.330155771	0.284	0.349	1.272	0.065	1.187	1.197	4.354	5.68415577
34	1.927	6.825	3.053	6.979	4.062	7.034	4.479	7.629	1:49	1.1364823	2.145205	2.675653	1.010498	1.567155	0.726577	1.322800059	0.28	0.344	1.252	0.064	1.169	1.178	4.287	5.60980006
35	1.93	6.831	3.05	6.98	4.064	7.034	4.498	7.608	1:50	1.1298677	2.143634	2.682975	1.015437	1.578318	0.719605	1.308039946	0.269	0.337	1.209	0.068	1.126	1.14	4.149	5.45703995
36	1.929	6.836	3.049	6.98	4.063	7.035	4.513	7.592	1:51	1.1292192	2.143259	2.692321	1.015491	1.58677	0.716065	1.292812044	0.264	0.333	1.172	0.069	1.093	1.107	4.038	5.33081204
37	1.93	6.844	3.05	6.98	4.061	7.035	4.522	7.579	1:52	1.1282269	2.139542	2.694195	1.012495	1.589209	0.713062	1.283798282	0.256	0.322	1.143	0.066	1.072	1.083	3.942	5.22579828
38	1.931	6.848	3.051	6.98	4.062	7.035	4.528	7.571	1:53	1.1277517	2.139189	2.695763	1.012495	1.590852	0.710248	1.282572994	0.252	0.318	1.126	0.066	1.059	1.07	3.891	5.17357299
39	1.934	6.854	3.051	6.98	4.062	7.035	4.533	7.565	1:54	1.1240841	2.135684	2.694498	1.012495	1.593282	0.709042	1.275439499	0.243	0.309	1.112	0.066	1.048	1.059	3.837	5.1124395
40	1.938	6.858	3.052	6.98	4.062	7.035	4.536	7.561	1:55	1.1206605	2.131362	2.691433	1.011496	1.59368	0.708062	1.270343439	0.236	0.301	1.105	0.065	1.042	1.052	3.801	5.07134344

Sheet1 Sheet2



Vita

Mahroo Sajid graduated as an Electrical Engineer from the College of Electrical and Mechanical Engineering, NUST Pakistan in the year 2009. She worked as a Contracts Engineer for a period of three years with Descon Engineering.

She joined the Masters in Mechatronics Engineering program at the College of Engineering, American University of Sharjah in September 2019 as a graduate teaching assistant. She has co-authored a conference paper for the ECCE 2021 conference, held in Vancouver, Canada. Her research interests are in Swarm Robotics, Biorobotics and AI.