Electronics Club IIT Kanpur
Swarm Robotics

by

Muskan Agarwal
Vipul Jain
Sankalp Sharma
Kaustav Sen
Piyush Singh
Sharad Gaikwad
Shourya Jain
Palashdeep Singh
Vyush Agarwal
Debashish Reang

Final Documentation
Swarm Robotics
Summer Project
Electronics Club
The Science and Technology Council
Indian Institute of Technology, Kanpur

Project Mentors:

Naveen Balaji
Soumya Ranjan Dash
Jay Mundra
Mudit Agrawal
Nitish Vikas Deshpande

Summer 2019

Contents

Abstract

Swarm robotics is an approach to the coordination of multiple robots as a system which consists of large numbers of mostly simple physical robots. It is supposed that a desired collective behavior emerges from the interactions between the robots and interactions of robots with the environment. This approach developed in the field of artificial swarm intelligence, from the biological studies of insects, ants and other fields in nature where swarm behaviour occurs [6]. In this project, we tried to simulate that swarm behaviour in that our robots to coordinate with each other and try to form the shape that the user inputs while taking care that the distance covered and the time taken for the formation of the shape is optimal. We also take care that the robots do not collide while following their paths, using the Conflict Based Search (CBS) algorithm [5] and infrared(IR) sensors. We show an example case - how the algorithm works, and how the robots attempt to form the shape optimally. We conclude by discussing how the model can be improved while discussing the problems we faced along the way and what future work can be done.

Acknowledgements

We are grateful to the **Science and Technology Council IIT Kanpur**, and **Electronics Club** for providing us with the resources, opportunity and motivation to carry out this summer project. We would also like to express our gratitude to the coordinators of Electronics club: **Naveen Balaji, Soumya Ranjan Dash, Jay Mundra, Mudit Agrawal, Nitish Vikas Deshpande**, for guiding us throughout this project.

Thanks also to those numerous researchers whose work we've referred to in the citations. It is because of them that the field of **Swarm Robotics** is getting noticed for its applications, and its scope to get rid of many human drudgery.

1 Introduction

"Don't take it all too seriously. If you want to live your life in a creative way, as an artist, you have to not look back too much. You have to be willing to take whatever you've done and whoever you were and throw them away."

– Steve Jobs

1.1 Overview

Swarm robotics is the study of how to coordinate large groups of relatively simple robots through the use of local rules. It takes its inspiration from societies of insects that can perform tasks that are beyond the capabilities of the individuals. Beni [1] describes this kind of robots' coordination as follows:

The group of robots is not just a group. It has some unique characteristics, which are found in swarms of insects, that is, decentralised control, lack of synchronisation, simple and (quasi) identical members.

In this report we realize the project we carried out over the summer in this field of multi-robotic systems. The aim is to give a glimpse of swarm robotics and its applications, correctly form shapes using coordinated movement of robots as specified by the user input. Section 2 describes the problem we attempted to solve formally. In Section 3, we give a brief background on swarm robotics as a sub-field of robotics, while the motivation and inspiration of swarm robotics taken from social insects is also explained. Section 3 continues addressing the main characteristics and classification of swarm robotics. Sections 4 and 5 lays the foundation to our solution approach. A detailed layout of the components used, and the methods devised is given in Section 6. Section 8 discusses the possible applications of swarm robotics in the near future. We discuss the problems we faced in Section 9, while laying down the foundation for future work in Section 10.

Figure 1: **Human Rescue using four robots.** *This image is taken from an actual research paper by Madhav D Patil and Tamer Abukhalil.*

2 Problem Statement

"We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried."

– Alan Turing, *Computing Machinery and Intelligence (1950)*

Given a user input from the GUI, the robots should collectively try to emulate the shape as closely as possible while avoiding collision with each other, at the same time taking the shortest possible path and the optimal time to cover their respective paths.

The following block illustrates the task for 1 robot:

```
def Swarm(GUI_INPUT):
    # GUI_INPUT has start, end co-ordinates
    start, end = GUI_INPUT

    endNotReached = True

    while endNotReached:
        move forward
        avoid collision

        if not endNotReached:
            notReached = False

    return None
```

3 Swarm Robotics and Multi-Robotic Systems Background

3.1 Social Insect Motivation and Inspiration

The collective behaviours of social insects, such as the honeybee's dance, the wasp's nest-building, the construction of the termite mound, or the trail following of ants, were considered for a long time strange and mysterious aspects of biology. Researchers have demonstrated in recent decades that individuals do not need any representation or sophisticated knowledge to produce such complex behaviours [3]. In social insects, the individuals are not informed about the global status of the colony. There exists no leader that guides all the other individuals to accomplish their goals. The knowledge of the swarm is distributed throughout all the agents, where an individual is not able to accomplish its task without the rest of the swarm.

Social insects can exchange information, and for instance, communicate the location of a food source, a favourable foraging zone or the presence of danger to their mates. This interaction between the individuals is based on the concept of locality, where there is no knowledge about the overall situation. The implicit communication through changes made in the environment is called *stigmergy*. Insects modify their behaviours because of the previous changes made by

their mates in the environment. This can be seen in the nest construction of termites, where the structure of the nest determines the changes in the behaviours of the workers.

Organisation emerges from the interactions between the individuals and between individuals and the environment. These interactions are propagated throughout the colony, and therefore, the colony can solve tasks that could not be solved by a sole individual. These collective behaviours are defined as self-organising behaviours. Self-organisation theories, borrowed from physics and chemistry domains, can be used to explain how social insects exhibit complex collective action that emerges from interactions of individuals behaving naturally. Self-organisation relies on the combination of the following four basic rules: positive feedback, negative feedback, randomness, and multiple communications.

3.2 Classification

In this section, we classify and characterise swarm robotics using the most known taxonomies and classifications in the multi-robotic systems' literature. Dudek et al. define a taxonomy [2] in which different axes are used to characterise multi-robotic architectures using their properties. The taxonomy axes are summarised in Table 1, directly extracted from the author. Using this classification, properties are assigned to each one of the axes, for a generic swarm-robotic architecture, although these properties would depend on the concrete architecture. *Collective Size* is *SIZE-INF*, that is, the number of robots $N \gg 1$, in opposition to *SIZE-LIM*, where the number of robots is small compared to the size of the task or environment. This expresses the scalability aimed in swarm-robotic systems. *Communication Range* is *COM-NEAR*, robots can only communicate with robots which are close enough. *Communication Topology* for a swarm system would be generally *Communication Bandwidth* is *BAND-MOTION*, communication costs are of the same magnitude as the cost of moving the robot between locations. *Collective Reconfigurability* is generally *ARR-COMM*, this is, coordinated rearrangement with members that communicate; but it could also be *ARR-DYN*, dynamic arrangement, positions can change arbitrarily. *Process Ability* is *PROC-TME*, where computational model is a turing machine equivalent. Lastly, *Collective Composition* is *CMP-HOM*, meaning that robots are homogeneous [4].

Axis	Description
Collective size	Number of robots in the collective.
Communication range	Maximum communication range.
Communication topology	Of the robots in the communication range, those which can be communicated with.
Communication Bandwidth	How much information the robots can send to each other.
Collective reconfigurability	The rate at which the organisation of the collective can be modified.
Process ability	The computational model used by the robots.
Collective composition	Are the robots homogeneous or heterogeneous.

Figure 2: Classification of Swarm robotics per Dudek et al.

4 Components Used

We will label each bot with a serial number by mounting an `ArUco` marker on top of it and align them accordingly with the help of a camera placed on the top of the arena. Each bot has the following essential components -

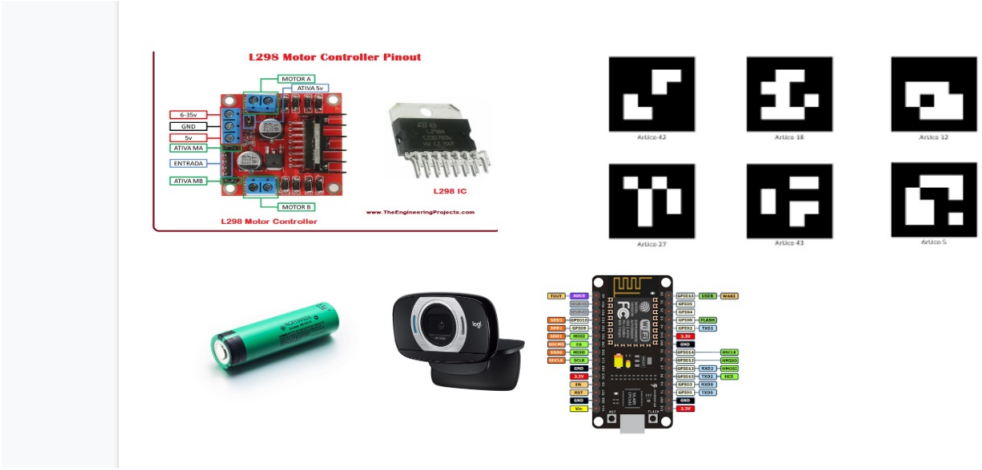


Figure 3: Components Used

- Motor Driver
- 3 wheels(2 wheels and 1 caster wheel)
- A Micro controller or `nodeMCU`
- A battery Pack to drive motors and rest of the components
- 4 IR sensors to avoid obstacle at local level
- Logitech Web-Cam

5 Overview of our Solution Approach

In our solution approach to the problem we are using `ArUco` markers on individual robots to detect and uniquely identify them in the arena. We have an overhead camera mounted to capture the view of the whole arena continuously. But, there is a catch - the image captured can be in different perspectives depending upon the orientation of the camera and the level of arena surface. To tackle this we have mapped the arena so that any perspective can be resolved into a single 2-D flat feed. The mapped feedback that we get from the camera consists of the location of each robot on the arena (*as mapped by us*). The user then inputs any shape into the GUI, which we map to the arena as the final coordinates that the bots must move to. We use CBS (Conflict-Based Search) path planning algorithm which follows A-star path planning for individual robot navigation and a conflict based solution approach to avoid intersecting paths, and hence collision. We have also implemented IR sensor based collision handling which will ensure that there is no collision even in the event of the failure of the algorithm or if the paths get too complex.

6 Description of the Assembly and the Solution Approach

6.1 Hardware

6.1.1 Assembling of chassis

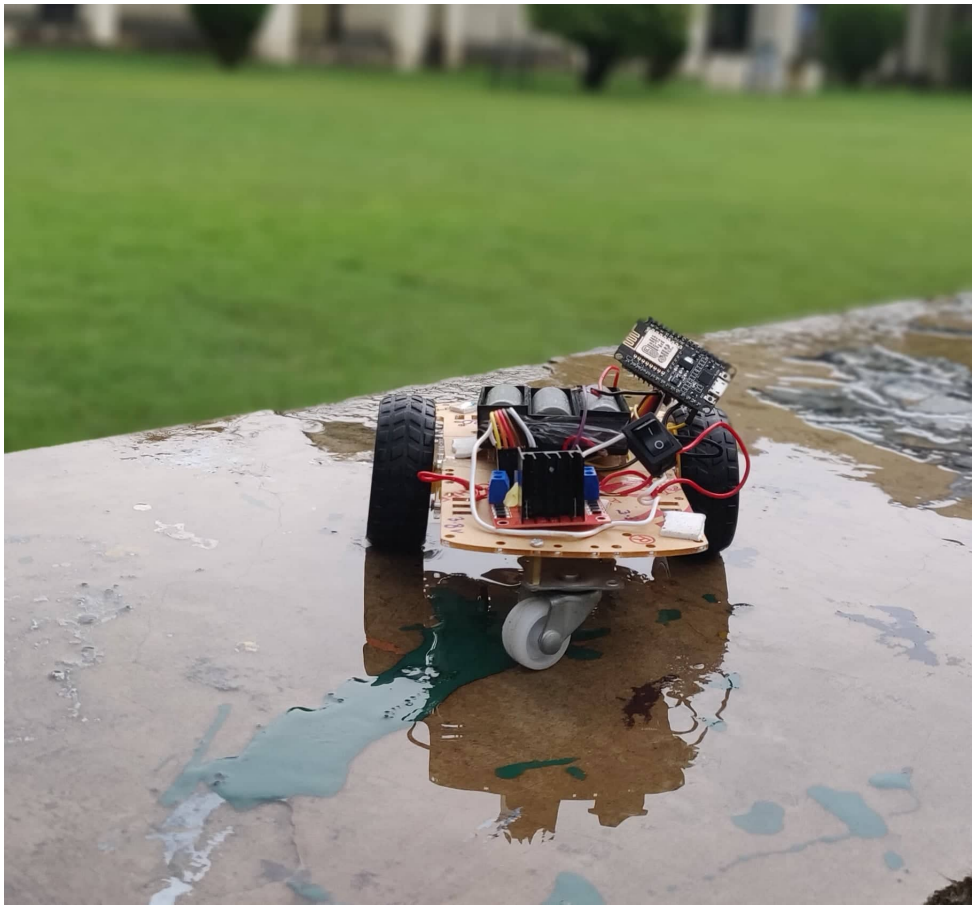


Figure 4: Our Bot

This is the image of the robot that we made using motor drivers, `nodeMCU` and motors. We attach the `nodeMCU` by PC so that we can set up a server, such that all bots will be able to communicate each other. Then we attach both motors and the motor driver on the chassis. Then we upload the code from the PC to the `nodeMCU`. We monitor the bots by the client and server.

6.1.2 Running the robot using Server and Client

For working of robot, we used the robot as client and gave input from the server created on the PC. To give wireless input we used `ESP8266`. For controlling the speed of bot and for its movement in different directions we used motor drivers. Motor driver is connected to the `nodeMCU` which gives the command to motor driver received from `ESP8266`.

6.1.3 Powering `nodeMCU` using battery

Firstly we were powering the `nodeMCU` using power bank, but it created problem due to its size. So now we are powering it through motor driver which gives output of 5V.

6.1.4 Using of IR Sensor

Figure 5: IR Sensor

IR sensors are used to avoid collision between the robots while they are traversing towards the final goal. If each bots moves to its nearest goal then there is rare chance of collision. But due to some errors in hardware bots don't always move in straight line. The code that we implemented either stops the robot or redirects its path if it comes too close to another bot. This way we achieve the best of both worlds.

6.2 Computer Vision

6.2.1 `ArUco` Generation

We used `OpenCV` in Python to generate the code for `ArUco` Marker. An `ArUco` marker is a synthetic square marker composed by a wide black border and a inner binary matrix which determines its identifier (id). The black border facilitates its fast detection in the image and the binary codification allows its identification and the application of error detection and correction techniques. The marker size determines the size of the internal matrix. It must be noted that a marker can be found rotated in the environment, however, the detection process needs to be able to determine its original rotation, so that each corner is identified unequivocally.

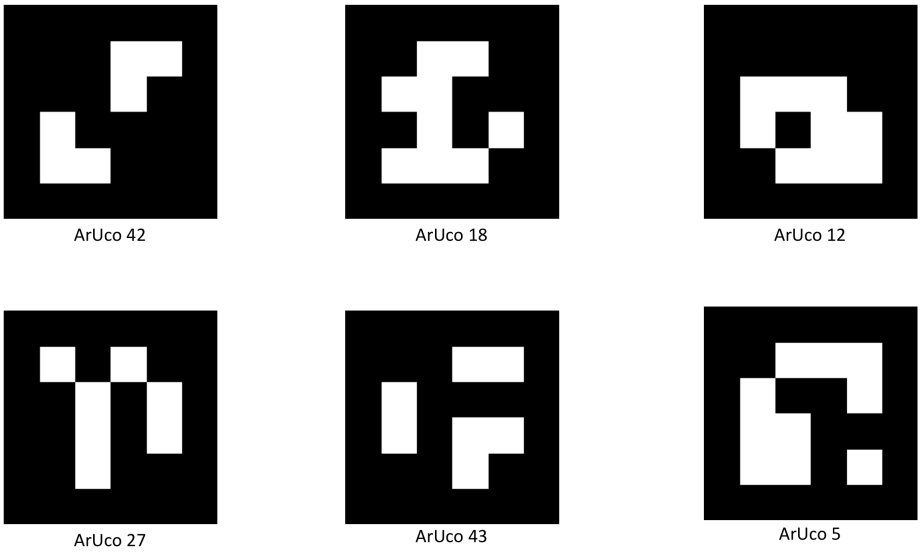


Figure 6: `ArUco` markers

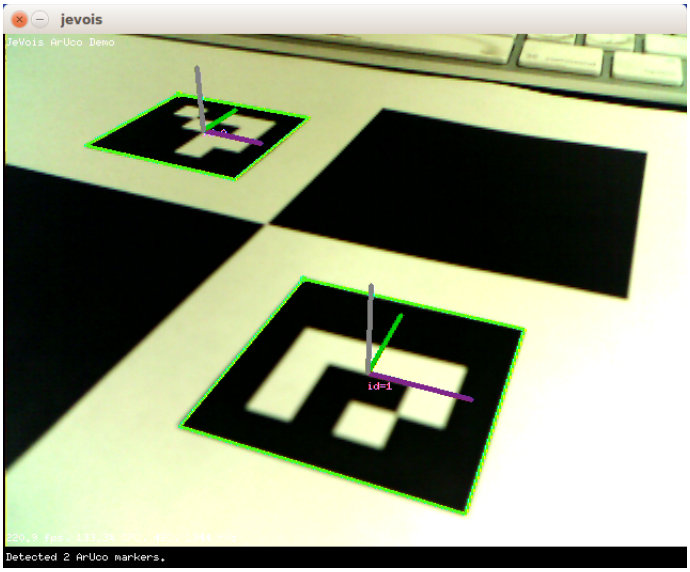


Figure 7: `ArUco` markers

6.2.2 `ArUco` Detection

Given an image where some `ArUco` markers are visible, the detection process has to return a list of detected markers. Each detected marker includes:

- The position of its four corners in the image (in their original order).
- The id of the marker.
- The marker detection process is comprised by two main steps:
  - **Detection of marker co-ordinates.** In this step the image is analyzed in order to find square shapes that are co-ordinates to be markers. It begins with an adaptive threshold to segment the markers, then contours are extracted from the threshold image and those that are not convex or do not approximate to a square shape are discarded. Some extra filtering are also applied (removing too small or too big contours, removing contours too close to each other, etc).
  - **After the co-ordinate detection**, it is necessary to determine if they are actually markers by analyzing their inner codification. This step starts by extracting the marker bits of each marker. To do so, first, perspective transformation is applied to obtain the marker in its canonical form. Then, the canonical image is threshold using `Otsu` to separate white and black bits. The image is divided in different cells according to the marker size and the border size and the amount of black or white pixels on each cell is counted to determine if it is a white or a black bit. Finally, the bits are analyzed to determine if the marker belongs to the specific dictionary and error correction techniques are employed when necessary.

In the `ArUco` module, the detection is performed in the `detectMarkers()` function.

6.2.3 Finding Distance Between 2 `ArUco` markers

Finding the distance between `ArUco` markers is of immense importance for completion of our task, which is, to move one bot from one location to other(goal position). The goal location is a coordinate in the arena (*that we mapped*) as determined by us and the user input. The robots then find a path that is optimal in terms of distance and time, from the start point

to the end. The infrared(IR) sensors mounted on the bots avoid local collision, hence safely getting the bots to their destination and forming the desired shape.

6.2.4 Orienting the bot

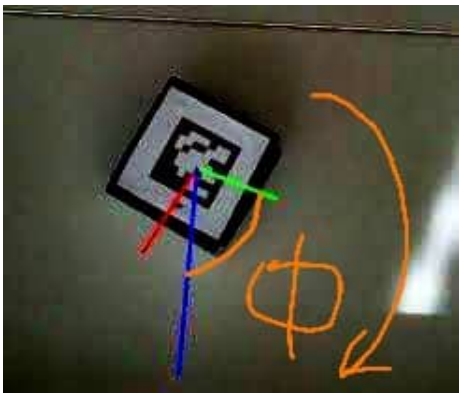


Figure 8: *Orientation of bot*

For this we used a very useful fact that the axes in `ArUco` markers are pre-defined, So we fixed the bot's head with the x-axis of the `ArUco` marker. So when bot turns then the axis of `ArUco` marker also turns. Then we used a inbuilt function to calculate angle of axis of `ArUco` marker with x-axis (*angle 1*). Then we used same function to calculate angle of line joining the centres of two `ArUco` marker with x-axis (*angle 2*). When the two angles mentioned above became equal then the rotation is stopped.

6.3 Graphical User Interface (GUI)

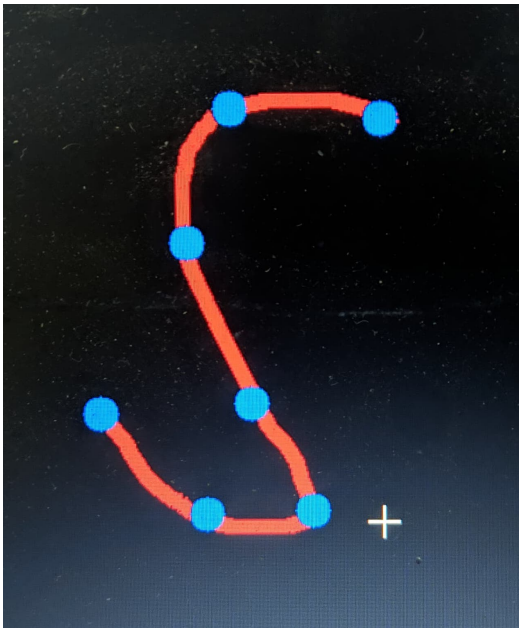


Figure 9: *Our GUI. The* **S** *is input by the user, we select the blue dots which correspond to each of our bot's final goal.*

6.4 The Arena

6.4.1 Setting up of the arena

A rectangular mattress is laid as the arena. The colour of the map is dark so that it can be easily contrasted from the ground on which it is kept. Setting up of arena is important so that we can stop bots from going out of camera vision at any instance. The surface of arena should also be such that wheels of bot don't slip.

6.4.2 Mapping of Arena and camera calibration

Today's cheap pinhole cameras introduces a lot of distortion to images. Two major distortions are radial distortion and tangential distortion. For stereo applications, these distortions need to be corrected first. To find all these parameters, what we have to do is to provide some sample images of a well defined pattern (eg, chess board). We find some specific points in it (*like square corners in chess board*). We know its coordinates in real world space and we know its coordinates in image. With these data, some mathematical problem is solved in background to get the distortion coefficients.

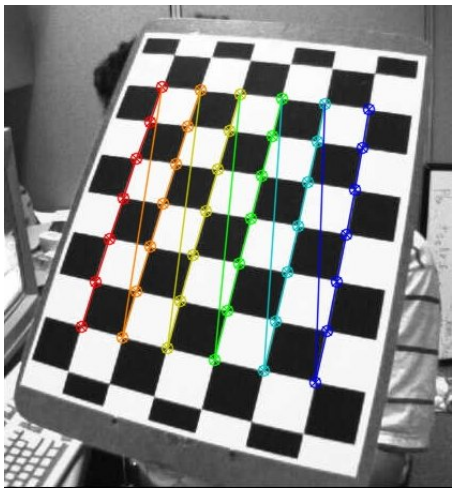


Figure 10: Checker Box

6.4.3 Wrapping of Perspective

As it is clear from the picture below, that 'Wrapping of Perspective' means that whatever be the image viewed through the camera, it must be transformed to normal view and crop out the extra portion to get the output image as shown. This is done to achieve the real pixel co-ordinates of the arena with one corner as (0,0). With this un-distort function is also used

6.4.4 Mapping

After the work of wrapping of perspective is done. The top left corner of the cropped image automatically gets pixel coordinates as (0,0). Now every coordinate in the arena can be easily known with respect to one of the vertex. This will also help in doing path planning tasks. This can also be used in motion code of bots to restrict them inside the arena.

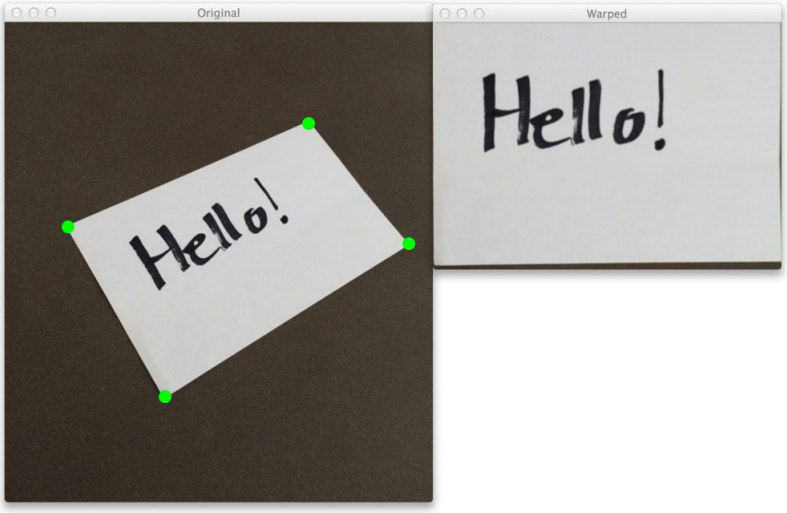
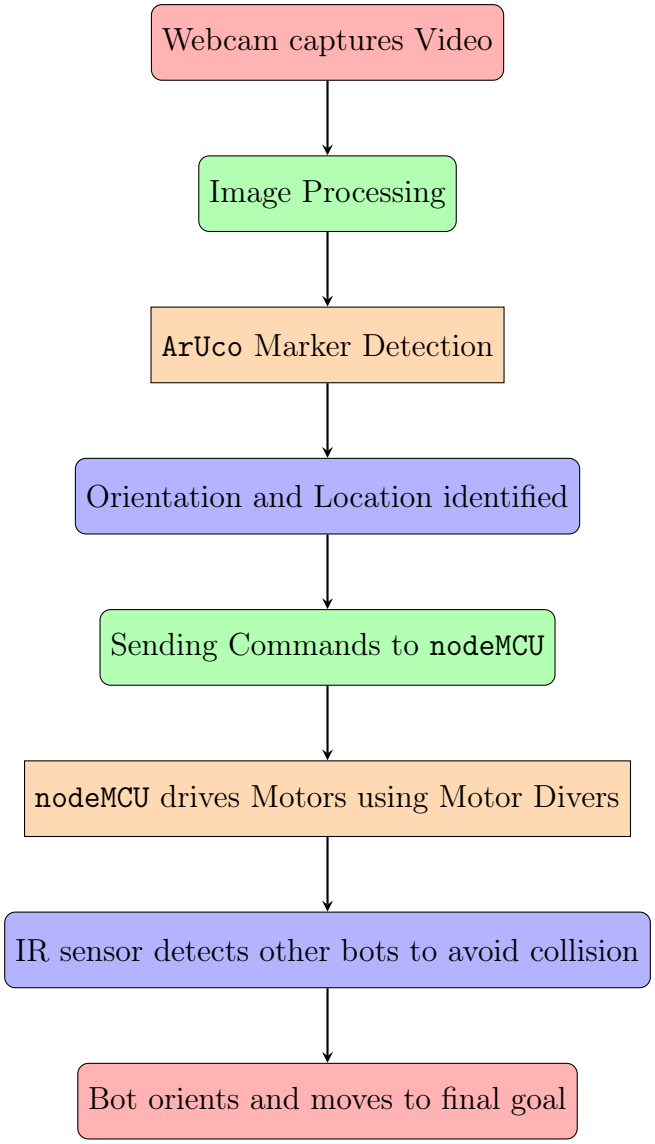


Figure 11: Wrapping of Perspective using `OpenCV`

7 The Recipe



Flowchart: The Recipe - Our approach in a nutshell.

8 Path Planning

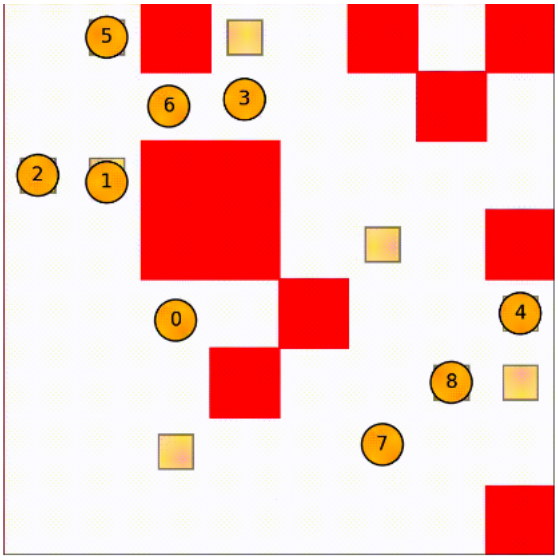


Figure 12: *CBS GUI*

Basically CBS takes all possible path for each and every bot from start to goal by the use of AStar.Then it recursively checks if their is any chances of collision of any bot, then it rejects path having chances of collision and finally gives the path having minimum cost.

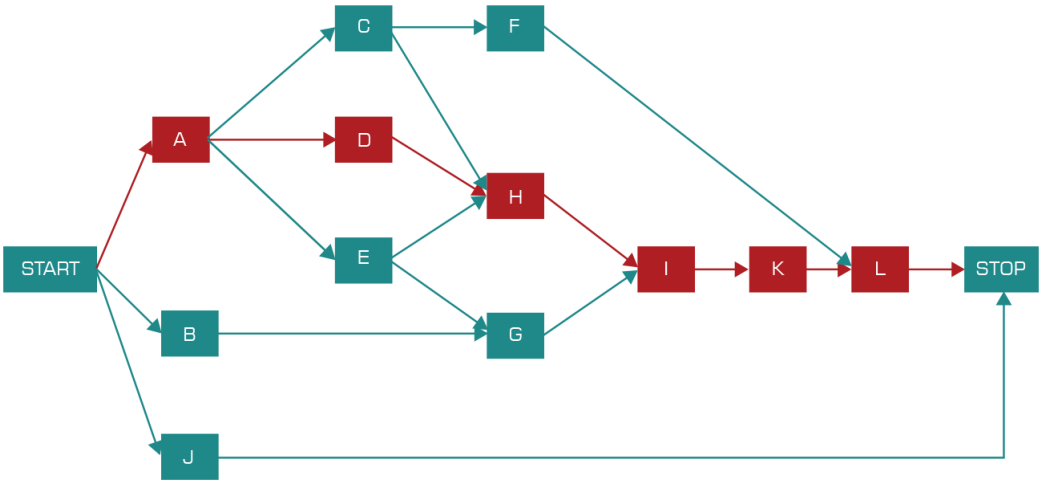


Figure 13: Path Planning

8.0.1 AStar Path Planning

The only difference in AStar and Dijkstra is that heuristic function in Dijkstra is always zero independent of node (Heuristic function calculates the displacement between current node and final goal). That's why AStar is more efficient than Dijkstra.

In AStar bots moves to the node having minimum 'f' value.

f = g + h

g :- Distance between current node and next node.

h :- Displacement between next node and final goal.

9 Applications

- Our bots can be used to do a work which one bot cannot do, like shape formation, moving by maintaining a shape, pushing or lifting a heavier object, moving up the steep incline.
- Disaster rescue missions is one of the most important applications of swarms robots.
- Swarms robots can be used in military to form an autonomous army or IRON MAN kind of suit.
- Swarm robotics, in particular, is considered extremely relevant for precision farming and large-scale agricultural applications.
- Swarms robots can be for Exploration and mapping it would save time.
- In medical fields, a use of nano-robots moving through human veins and arteries (e.g. to fight certain types of cancer)
- Can be used in public places for cleaning, in restaurants for serving purposes.
- The number of possible applications is really promising, but still the technology must firstly be developed both in the algorithmic and modelling part, and also in the miniaturization technologies.

Figure 14: *Applications in Military*

10 Problems Faced

- The camera which we were using earlier was not able to detect `ArUco` marker on the bot continuously . So, our bot was moving with the initial command which it had received from the server.
- Camera was not able to track the movement of `ArUco` so to avoid randomness we had to stop the bot every time the camera was not able to recognise the `ArUco`.
- Because the arena was not entirely black we faced problem while mapping we tried adjusting the threshold but at last we decided to do it manually by choosing the region of interest.

- We had to decrease the fps of camera so that there is no lagging so to that we increased the resolution of the camera.
- To overcome static friction we had to give impulse to our bots.

11 Conclusion and Future Work

- Localization can also be done through WiFi. We can do faster and precise localization of bots using UWB(Ultra Wide-band) sensor and make swarm more robust to use in a room. Also we can change the algorithm of path planing and motion for achieving goal of each bot in a faster way. Also we can use Boids model which is exhibited by birds during their travelling from one location to other.

By the end of this summer project we are able to move bots from any given position and orientation to goal coordinates and form a shape or letter without bots colliding. Finally decentralization should be achieved to complete swarm robotics like the insects, bird or animals. And more future goals are same as application's first point.

References

[1] G. Beni. From swarm intelligence to swarm robotics. *Swarm Robotics Lecture Notes in Computer Science*, page 1–9, 2005.

[2] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4), 1996.

[3] S. Garnier, J. Gautrais, and G. Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1(1):3–31, 2007.

[4] I. Navarro and F. Matía. An introduction to swarm robotics. *ISRN Robotics*, 2013:1–10, 2013.

[5] G. Sharon, R. Stern, A. Felner, and N. Sturtevant. Conflict-based search for optimal multi-agent path finding. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 563–569. AAAI Press, 2012.

[6] Wikipedia contributors. Swarm robotics — Wikipedia, the free encyclopedia, 2019. [Online; accessed 8-July-2019].