SWARM INTELLIGENCE BASED SOLUTION FOR NAVIGATION OF UNMANNED GROUND VEHICLES

Jayasooriya Arachchi Patabedige Isuru

149155L

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

February 2017

SWARM INTELLIGENCE BASED SOLUTION FOR NAVIGATION OF UNMANNED GROUND VEHICLES

Jayasooriya Arachchi Patabedige Isuru

149155L

Thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Artificial Intelligence

Department of Computational Mathematics
University of Moratuwa
Sri Lanka

February 2017

Declaration

I declare that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a Degree or a Diploma in any University and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and summary to be made available to outside organization.

Name of Student	Signature of Student
J.A.P.Isuru	Date:

Supervised by

Name of Supervisor(s)	Signature of Supervisor(s)
Prof. Asoka S. Karunananda	Date:

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Asoka Karunananda for guiding me throughout the project. He has given his fullest corporation to me whenever I sought for advice. Also, Prof. Karunnanda's teaching techniques helped me to consider problems in different avenues.

Since this research deals with hardware implementation, I had to acquire lot of knowledge on that electronics domain. I was able to get support from Shan Chathuranga who is the expert in the domain of electronics. So, I would like to pay my gratitude to Shan Chathuranga for helping me dispute his busy schedule.

I must also thank all members of the lecture panel. During the period of one year and three months, these valuable lectures helped me to think differently.

For this research, I had to refer to many books and research papers as reference. I would like to thank all the authors of those publications.

Also, my batch mates helped me lot in many different ways. I would like to pay my gratitude for them. They made my life enjoyable during the course period.

I would like to place my gratitude to my loving parents and wife for always encouraging for me on higher studies.

Last but not least, I would like to thank all my colleagues and others who are not mentioned, for all the support extended to me. Without their dedication, the project would not have been successful.

Abstract

There are many circumstances where involvement of a human driver to control vehicles is not feasible. The best examples for the above mentioned scenarios are the applications based on Astrology. As a solution to this problem, researchers are trying to create unmanned autonomous vehicles. Most of these researches have been conducted using the power of artificial intelligence. Nevertheless unmanned autonomous vehicle navigation is one of the biggest problems in the current era of artificial intelligence. It is more difficult when vehicles are navigating dynamically changing environment. This thesis presents a swarm intelligent based solution for the navigation of unmanned ground vehicles within dynamically changing environment.

The proposed solution uses only local information around the vehicle, as in reality human driver also getting decisions based on the partially observable environment. System then uses current positions of unmanned vehicles as inputs and provides future positions of unmanned vehicles as the output. Also this proposed system consists of three main modules called data acquisition module, data processing and decision making module and decision execution module. Data acquisition module collects data from other agents and from the environment. Data processing and decision making module acts as the brain of the system and decision execution module executes the output of the decision making module.

Evolutionary computing and machine learning are main techniques which were used behind this proposed system. System initially uses evolutionary computing technique to navigate in an unknown environment. But when system familiarize with the environment, it tries to work with its prior knowledge by using machine learning techniques. Ultimately vehicles will navigate as swarms of vehicles to the targets. Evaluation illustrates swap mutation is more efficient than Gaussian mutation in evolutionary computing approach. Neural network works 98% accurately when we use 25000 training samples in the machine learning module. Final evaluation uses both computer simulated environment and small real toy vehicles to demonstrate the solution. Upon completion of the final system, we can observe a successful target oriented navigation of vehicles in a partially observable environment using swarm intelligence based approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prolegomena

Due to the enhancement of the Artificial intelligence, nowadays unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGV) are used in many domains. Also both types of unmanned vehicles can be used as either as isolated or in teams. In simple terms unmanned autonomous vehicles are small intelligent machines which are controlled and navigated automatically for achieving a target at a given destination either as an individual entity or a team of entities [1].

The path finding, navigation, team working, fault tolerance are some of major problems which can be difficult to achieve in UAV and UGV controlling. Then controlling unmanned vehicles as a team is much harder than controlling a single vehicle [2].

This system proposes swarm like approach for autonomously controlling unmanned vehicles using evolutionary computing and machine learning techniques. In this connection, this chapter presents aim and objectives, background and motivation, problem in brief, novel approach which has been proposed to achieve autonomous navigation of unmanned vehicles and structure of the overall thesis.

1.2 Aim and Objectives

Aim: The aim of this project is to develop swarm intelligence based UGVs controlling system to work in dynamic environment with fault tolerance capabilities.

Objectives:

- Study of the UAVs, UGVs and their controlling techniques
- Study of technologies that can solve the problem (genetic algorithm, machine learning, swarm intelligent techniques)
- Design a system for solving the problem in simulated environments
- Implement a system for software and hardware simulations

- Evaluation of the proposed solution
- Preparation of final documentation

1.3 Background and Motivation

Today people try to create various autonomous machines for replacing human workers. Unmanned autonomous vehicles are the best example for autonomous machines in current era of autonomous machines. There are many situations where human driver cannot be involved in controlling vehicles. Astronomical applications are best example for that kind of situation where human operator cannot be involved. So these autonomous vehicles concept could be an interesting replacement for human driver.

But controlling a vehicle is not an easy task even for a human, because there are many complex scenarios which should be handled by the driver. Randomly changing dynamic environment is the best example for that kind of complex environment. As an example driver should be able to get quick decisions even he hasn’t face similar kind of complex situation in past.

So there are many researchers try various approaches for controlling vehicles autonomously. But still path planning in a dynamic environment can be considered as a difficult problem to achieve. Since artificial intelligence based approaches has proven very success in the area of path planning, we can use power of artificial intelligence to create autonomous unmanned vehicles. So developing a novel approach to control an unmanned vehicle would be a still great target to achieve in current era of artificial intelligence.

1.4 Problem in Brief

Controlling a team of unmanned vehicles in a dynamic environment with fault tolerance capabilities is a one of biggest problem in current UAV and UGV controlling systems. That is the problem which is being gone to solve by this proposed solution. But in this research we mainly focus only on UGV controlling part.

1.5 Novel approach for unmanned vehicle controlling

This proposed solution is mainly based on the swarm intelligent approach. Within that contest system uses technologies like evolutionary computing and machine learning. Also proposed solution suggests evolutionary computing as a robust approach for path finding. Then it uses machine learning as a fault tolerance technique in dynamic environment. Then using above approach, system will predict the most suitable future position of the vehicle by analyzing local information related the vehicle. Most of the time local information may be neighbor vehicles and obstacles near to the vehicle. Finally all these vehicles will communicate each other as a swarm and achieve the goal as a team.

1.6 Structure of the thesis

Rest of the thesis is structured as follows. Chapter 2 critically reviews the domain of autonomous agent navigation by highlighting current solution, practices, technologies, and limitations defining the research problem. Chapter 3 described essentials of swarm intelligence, genetic algorithm and machine learning showing its relevance to implement a solution for unmanned vehicle navigation. Chapter 4 present our Swarm intelligence based approach for controlling unmanned vehicles. Chapter 5 is on the design of proposed solution. Chapter 6 contains details of implementation of the system which predict the next possible position of the unmanned vehicle. Chapter 7 talks about the evaluation part of the thesis. Chapter 8 concludes the outcome of the research with the note on further work.

1.7 Summary

This chapter described the full picture of the whole project showing research problem, objectives, hypothesis and the novel solution. Structure of the thesis was also explained by this chapter. Next chapter will be on literature review of autonomous agent navigation in terms of practices, technologies and issues for defining research problem.

Chapter 2

Development in unmanned vehicle navigation

2.1 Introduction

This chapter is about the development of unmanned vehicle navigation. It will discuss navigation techniques of both unmanned aerial and unmanned ground vehicles. Description includes short introduction to the currently exist techniques and critical evaluation about it. Finally it will describe current problems and limitations in explained techniques.

2.2 Current issues and practices

Here we divided the whole literature research into modules and will explain each topic under a sub topic. Each sub topic critically evaluates existing approaches and their limitations. Evaluation will be done with respect to both quantitative and qualitative properties in existing approaches.

2.2.1 Vision based autonomous vehicle navigation

In recent years there were many researches have been conducted by various universities regarding UAVs controlling. Hanze University of Applied Sciences in Netherland recently publishes a paper regarding the controlling of a UAV using a vision based line following strategy. The system which was proposed by them is intended to guide an autonomous UAV to follow water channel margins, crop lines and other similar patterns, to support automatic monitoring and inspection activities. Since GPS based approach is having accuracy problem, they suggest using vision based system to improve the accuracy [3].

Another paper discusses a vision based approach to navigate the UAV within indoor environment [4]. And some other paper presents a methods for landmark recognition in vision based UAVs [5]. Gianpaolo Conte and Patrick Doherty have presented a paper which uses image matching techniques for UAV navigation. The aim of this paper is to explore the possibility of using geo-referenced satellite or aerial images to augment an Unmanned Aerial Vehicle (UAV) navigation system in case of GPS failure.

Another research papers present the ability track target within an urban environment by fusing data from UAV [6]. Although real mean of UAV is an intelligent machine, above papers talks only about navigation of the UAVs using vision based techniques. But they have found an efficient position tracking method by combining vision based approach and GPS satellite signals.

2.2.2 Kalman filter based autonomous vehicle navigation

Austin M. Jensen used UAV with swarm like behavior for tracking a fish using multiple UAVs. In this research they tag a transmitter with the fish and receivers with UAVs. Then a simplified version of the propagation model and an Extended Kalman Filter were used to estimate the position of the transmitter [7]. Fuzzy/Kalman navigation system for Unmanned Aerial Vehicles (UAV) is another similar kind of research paper presented by F.M. Raimondi in 2010 [8]. Even though these two papers talks about some intelligent approach to estimate the positions, still they are not intelligent as expected for doing intelligent tasks like fault tolerance, learning, etc...

2.2.3 Evolutionary computing based approach for agent navigation

Ioannis K Nikolos, K.P. Valavanis, Nikos Tsourveloudis and A N Kostaras presented a very interesting paper about UAV navigation in 2003. This paper gives two navigation approaches as offline planner and online planner. Offline planner works in known 3-D environment while online planner works in completely unknown 3-D environment. Author uses evolutionary computing for path planning, because of their high robustness compared to other search methods [9]. Although this paper presented an intelligent navigation approach for UAVs, still it consists with lack of learning capabilities for fault tolerance.

2.2.4 Learning based approach for agent navigation

Bond and Gasser published a paper on 1988 regarding Distributed Problem Solving (DPS) and Multi Agent Systems (MAS). Here they defined the use of reinforcement learning approach in the context of MAS. In this paper they explains how huge problem is divided and shared with multiple agents to work with own interest and goals [10].

Another team from Carnegie Mellon University used reinforcement learning to control autonomous helicopter [11].

Todd Hester and Peter Stone presented reinforcement based approach for giving learning ability to robots. They used random forest model as the core algorithm for learning and they empirically evaluate learning accuracy of their concept by using velocity controlling process of an autonomous vehicle in real-time [12]. This paper has given novel approach to the autonomous agent navigation, but their method is lacking with high accuracy within initial stages of learning.

Another team of researchers publish a paper regarding agent learning technique using reinforcement approach. This system is well suited for less resource based computationally poor systems. Their method enables the robot to learn an efficient land-mark selection strategy to compactly model the environment. Also they have done experiments of their concept in both software simulated and real robot environment [13]. And again this system is also not working accurately, at initial stages of the learning process.

B. Bischoff and other researchers published a paper regarding autonomous agent navigation by using hierarchical reinforcement learning. Their research can be done within the scope of two main sections called movement planning and movement execution. Their proposed approach is implemented and evaluated on a mobile robot platform for a navigation task [14].

2.2.4.1 Deep learning based approach

In 1991 A. Pomerleau little experiment to control a vehicle in a simple environment by looking at images which were taken while human driver was driving. In the neural network it takes pixel values of images and processes it through single hidden layer. Then final output layer decide which action to be take in the given scenario [15].

In 1996 A. Stafylopatis, K. Blekas used both genetic learning classifier systems and reinforcement learning to construct efficient rules associated with the steering behavior of the vehicle [16].

Google recently published a paper regarding the unsupervised approach of the deep learning. They trained an autoencoder on a large dataset of images downloaded from the internet. They did it using model parallelism and asynchronous SGD on a cluster with 1,000 machines (16,000 cores) for three days. Then final result is awesome and it reveal that it is possible to train a face detector without having to label images as containing a face or not. Even it detects not only human faces, but also animal faces as well. Finally they improved this model to successfully recognize 22,000 object categories [17].

Three researchers from University of Toronto presented a paper regarding deep learning classification at conference of Neural Information Processing Systems (NIPS). They used convolutional neural network (CNN) approach to classify 1.2 million images into 1000 different classes. Their CNN consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax [18].

Very recently google pulished apaper called “Going deeper with convolutions” which suggest to use existing CNN architectures for local domains by changing few components only. Most often only layer, which user has to change is the final classification layer [19].

2.2.4.2 Deep reinforcement learning based approach

A team from DeepMind Technologies presented the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels in a single frame of Atari game and whose output is a value function estimating future rewards. They have successfully applied same model to multiple Atari games without changing. The results they got are unbelievable, due to its success of beating even human experts [20]. The paper called “Human-level control through deep reinforcement learning” which was published by same authors, explained theoretical parts of the above paper well [21].

Three people from Computer Science and Artificial Intelligence Laboratory at MIT employ a deep reinforcement learning framework to learn state representations and action policies using game rewards as feedback. This framework maps text descriptions into vector level representations that capture the meaning of the game states [22].

2.3 Summary

This chapter explained about current practices in unmanned vehicle navigation and then it critically reviewed those current practices by highlighting issues. Computer vision, kalman filter, evolutionary computing and machine learning are approaches were existing technologies used by people to solve autonomous navigation of agents. So in the approach section, it explains the proposed solutions for above issues.

Chapter 3

AI techniques to model unmanned ground vehicles

3.1 Introduction

This chapter presents the major technologies associated with the research. In proposed system unmanned vehicles act as swarms for achieving multiple goals together. Evolutionary computing approach is the technology which is used by agents for guessing their nearest future position of the path. Also agents will learn and change their behaviors according to their experience.

3.2 Swarm Intelligence

Swarm intelligence (SI) can be described as a theory which works with natural and artificial systems and contains many individuals that coordinate as decentralized manner and control as self-organization. Actually, there are already lots of Artificial Intelligence (AI) techniques which were inspired from biological techniques. As an example artificial neural network inspired from human brain and genetic algorithm inspired from evolution of human. Likewise in here Swarm intelligence is also inspired from biological system called social system which specifically does collective behaviors of simple individuals interacting with the environment and each other. There are two popular swarm inspired methods in AI called Ant colony optimization (ACO) and particle swarm optimization (PSO). ACO inspired from the behavior of collecting foods by ants and PSO is inspired from social behavior of bird flocking around food sources.

In ACO algorithm is searching for optimal path in the graph based on behavior of ants seeking a path between their colony and source of food. In this algorithm each ant moves at random and they deposit pheromone on path which they are travelling. Then other ants are trying to go through those pheromone paths and finally the shortest path is discovered via pheromone trails. In here algorithm assume that more pheromone on path increases probability of path being followed.

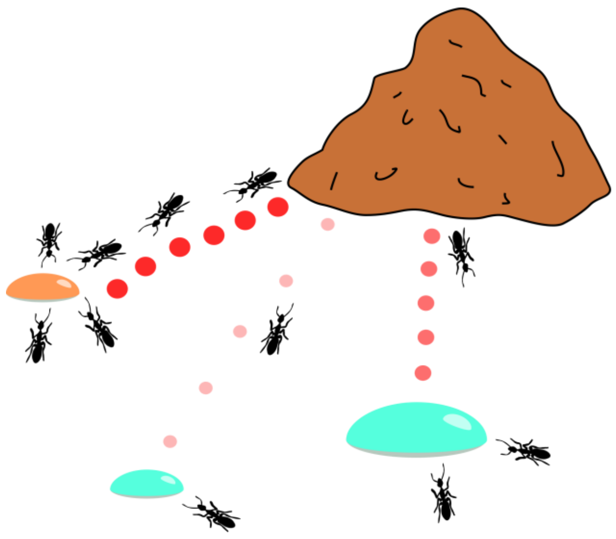


Figure 3.1: Ant colony optimization

PSO has many similarities with Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. But unlike GA, PSO has no evolution operators such as crossover or mutation. In PSO, the potential solutions, called particles, move through the problem space by following the current optimum particles [23].

Figure 3.2: Particle swarm optimization

3.3 Genetic algorithms

Genetic algorithm (GA) is a technique for solving both constrained and unconstrained optimization problems based on a natural selection process which uses biological evolution. The algorithm repeatedly modifies a population of individual solutions until they met the optimal solution or until they come to given number of generations. GA can be apply to solve problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, no differentiable, stochastic, or highly nonlinear. GA is different than older AI systems. Because GA systems don't break easily even if the inputs changed slightly, if there is a reasonable noise. When problem contains a large state-space, a genetic algorithm may give significant benefits over more typical search of optimization techniques.

In genetic algorithm each individual has own chromosome which explains their gene structure. Initially these gene values will be chosen randomly, then algorithm will choose parents and do the crossover and mutation over selected genes. From all parents and children, some of individuals from both parents and children will go to the next generation according to the fitness function [24].

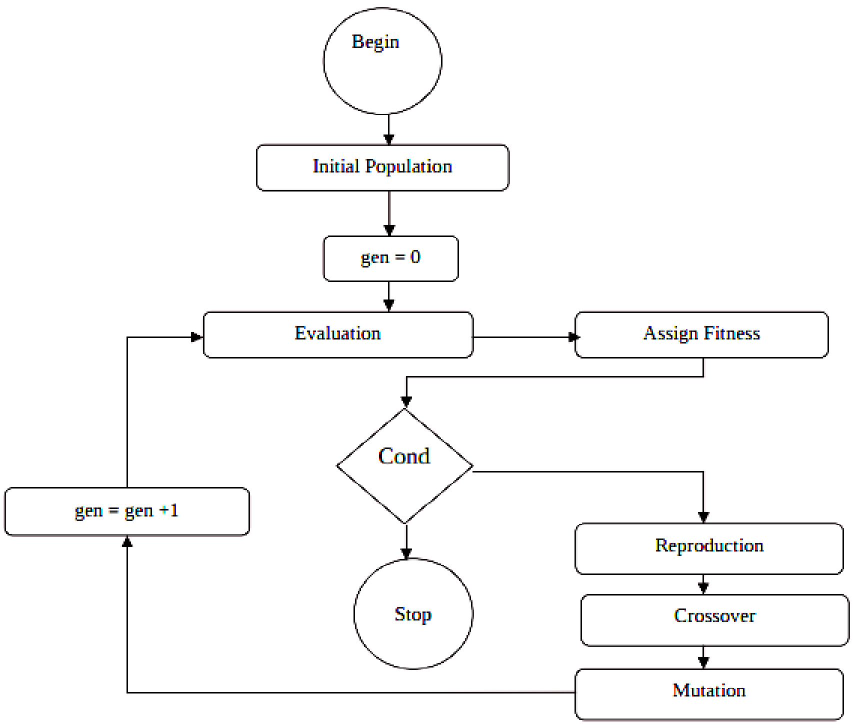


Figure 3.3: Flow chart of genetic algorithm

3.4 Reinforcement learning

Reinforcement Learning is the area of Machine Learning dealing with the actions that software agents like to take in a particular environment in order to maximize rewards. Agents update their behavior in future through trial and error interactions with a dynamic environment and then they try to adapt to the environment automatically. Because of this automated learning features, there is little need for a human expert who knows about the domain of application for creating intelligent agents.

In normal reinforcement learning model, an agent is connected to its environment via perception and action.

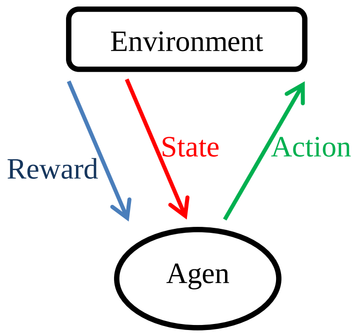


Figure 3.4: Flow chart of reinforcement learning

First an agent applies action from its finite action space by considering its current state. Then agent gets a reward from environment which is associated with the last state transition. By looking at received reward, agent can learn how to apply correct actions at given state. This process is continued as a repeated cycle and agent will learn continuously. In here the problem is to learn a way of controlling the agents so as to maximize the total reward. And also the learning problems differ in the details of how the data is collected and how performance is measured.

3.5 Deep learning

Deep learning is now doing a great role in the era of machine leaning. It has changed the whole machine learning aspects by applying deep structure for neural networks. Deep learning is also called Feature learning, because of its ability to learn features.

Deep learning deals with the machine learning process using an artificial neural net which has arranged number of layers as a hierarchy. It learns something simple at the first layer in the hierarchy and then sends this information to the next layer. The next layer takes this information and creates a bit more complex output. This process continues on each layer in the hierarchy which creates something more complex output from the input it received from the previous layer.

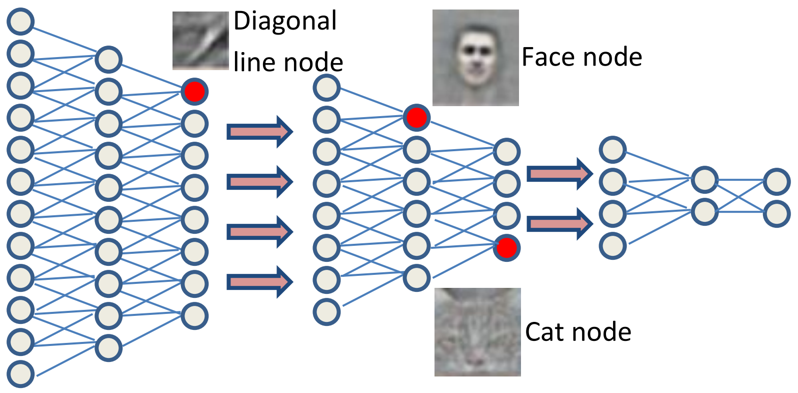


Figure 3.5: Basic deep learning architecture

Figure 3.5 represents an example deep learning architecture for image classification. Input layer contains values at pixel level. As an example it could be RGB information or HSV information of each pixel. Then next few layers can identify very low level features like horizontal lines, vertical lines or small blobs in the image. Those calculated low level information will be then processed by latter layers for generating higher level features such as faces. This process continuous until it reaches to final layer. Final layer is responsible for doing classification of the result from previous layer. As an example final layer classifies given image as a cat or another animal. While it was learning about cats, the network also learned to identify all of the other animals it saw along with the cats. Because network itself was learning about features of animals rather than just classifying image by given hand crafted features [25].

3.6 Deep reinforcement learning

With recent advancement of machine learning techniques, the combination of deep learning and reinforcement learning was proposed. This combination allows a learning agent to control a system based multi-dimensional input using a deep neural network to extract relevant features from the low level information. Recently most of the machine learning researchers applied deep reinforcement learning for famous ATARI games. These proposed algorithms beats even human experts in many different ATARI games.

Deep Q learning which is built on very famous reinforcement algorithm called Q learning algorithm is as example for new generation of deep reinforcement algorithm. The key idea of deep Q learning is to use deep neural networks to represent the Q-network. After that this network will be trained to predict total reward. Previous attempts to combine RL with neural networks had largely failed due to unstable learning.

3.7 Summary

Throughout this chapter we discussed technologies which are used to navigate multiple agents within partially observable environment. Swarm intelligence, genetic algorithm, reinforcement learning, deep learning and deep reinforcement learning were discussed in this chapter. This chapter will be the foundation for the theoretical part of the research.

Chapter 4

Approach - Swarm intelligence for UGV controlling

4.1 Introduction

In the previous two chapters we define the research problem as the inefficiency and incompetency in current approaches of autonomous vehicle controlling in dynamic environment. And also we discussed why swarm intelligence is the potential technology to develop novel method for controlling unmanned vehicles.

This chapter is about the approach which is suggested by the proposed system to solve the navigation problem when there are multiple unmanned vehicles in dynamic environment with multiple targets. System input, system output and the process are discussed with the non-functional requirements of the system.

4.2 Hypothesis

Swarm intelligent based approach for controlling multiple unmanned ground vehicles in a dynamic environment. It means that given multiple unmanned vehicles will navigate to given multiple destinations as swarms of unmanned ground vehicles.

4.3 Inputs to the System

Since agent is able to get only partially observable environment information, agent receives only information near to it. Each agent receives position of their neighbor agents by using a message parsing interface in the software simulation environment. But in hardware simulation it uses a color based object tracking method to get the positions of nearby agents. Also agents receive position of obstacles near to them by using centralized database in software simulation environment. System uses colored object detection approach to capture nearby objects in hardware simulation.

4.4 Output from the System

Output of the process is future positions of each agent, where they can move without any collision with other agents or obstacles. This output position should be in a short distance to the previous position, since agents are working on a dynamically changing environment. Also calculated path will be the most efficient path to achieve the goal with minimum cost.

4.5 Process

Initially system is using evolutionary computing approach for calculating or guessing the most accurate and most efficient future positions for UGVs. Since we are using evolutionary computing approach, we have to find correct fitness function with correct genes. In this system it will use positions of UGVs as genes of the chromosome. Then the fitness function provides most efficient path without any collision between agents or other obstacles. Then system calculates most possible intermediate future positions which can be traversed by agent with their current states.

While agents are travelling to the goal according to the above method, they will also try to learn from their past and try to produce most efficient travelling behavior in future.

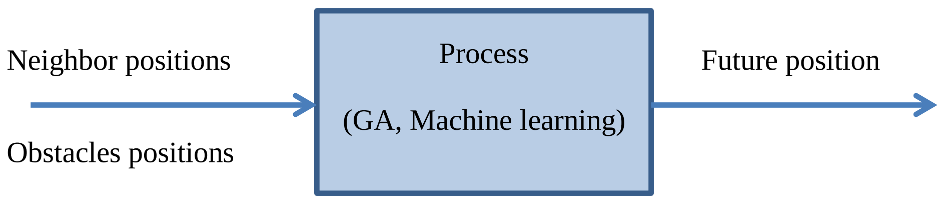


Figure 4.1: High level diagram of the proposed approach

4.6 Features

To control multiple UGVs autonomously in a dynamically changing environment, system facilitates following features.

- Path finding

  Since this is a very common task in any intelligent machine, there are lots of well-established methods which have been confirmed by researchers. This proposed solution will use Evolutionary Computing based approach as a path finding technique in partially observable environment.

- Navigation

  Sometimes distance between two calculated consecutive positions will be large and sometimes vehicles should be able to navigate through curved paths.

It means that it has to calculate intermediate positions for smooth navigation. In this situation system uses a mathematical model for intermediate position calculation.

- Team working

  This is the place what will use swarm intelligence for grouping behavior. Agents will divide the whole work into modules and will act as separate swarms to achieve one single goal. In this approach not only they work separately, but also agents will communicate each other and improve their learning models.

- Fault tolerance

  Fault tolerance in the sense we are using previous knowledge to avoid future failures such as obstacles clashes at earlier as possible. In here it will use a machine learning approach for future failure prediction.

4.7 Users of the system

Generally researchers who are involving in astronomy related things and people who are working in military activities. But people can use this system when they work on any research related to autonomous robot navigation. So there may be various types of users for this system including people who are from variety of domains.

4.8 Summary

This chapter discussed about the approach to design and implement solution for unmanned ground vehicle navigation in dynamic environment. So it gives brief description about the proposed system. Input, outputs, users, features are given along with the process which will be carried to create the system.

Chapter 5

Design of Swarm intelligence based UGV controller

5.1 Introduction

In this chapter we are going to discuss how we carried out the designing of this research. This chapter provides a detailed explanation of all methods and modules used in the analyses and relationship between those modules, as well as the reasons behind using those methods.

In our research agents are moving to targets as swarms or group of agents. Since agents are able to discover the environment partially, each agent communicates only with their neighbors. Also they can only see obstacles which are within short distances.

If we consider the whole process of a one agent, we can divide it into three main modules. Those modules are data acquisition module, data processing and decision making module, decision execution module.

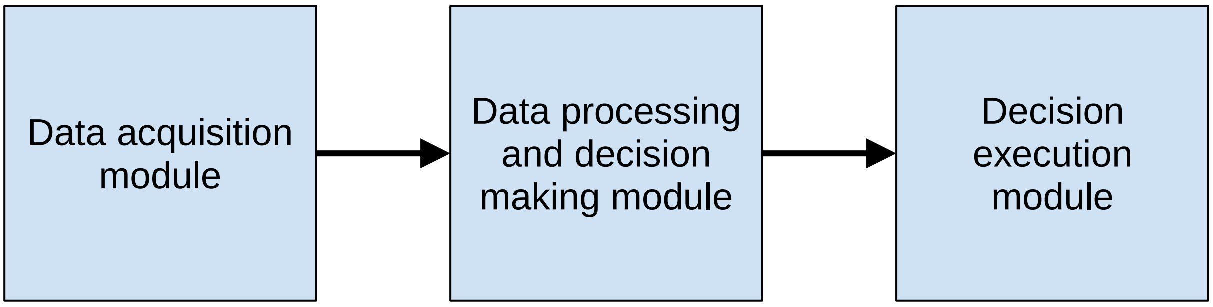

Data acquisition module
Data processing and decision making module
Decision execution module

Figure 5.1: Top level architecture of the proposed system

Other than above modules there is another module called **Indoor Positioning** module for detecting positions of each agent in the indoor environment, since there is no stable positioning solution like GPS for indoor environment. This module detects agent’s positions and facing angle using a computer vision approach. Also it detects obstacle positions and target positions as well.

5.2 Data acquisition module

This module does the process of gathering inputs to the system. There are two types of data acquisition processes are being happened in this module.

- Data acquisition from other agents
    - In this process agent acquires position information and historical records from neighbor agents.
- Data acquisition from the environments
    - In this process agent acquire nearest obstacle position information from the environment.

Finally all information will be sent to the data processing and decision making module for analyzing data.

5.3 Data processing and decision making module

This module acts as the brain of the agent and all the intelligent tasks are happened within this module. In our research we can divide agent's intelligence or agent's brain into two modules called cognitive intelligence module and learning intelligence module.

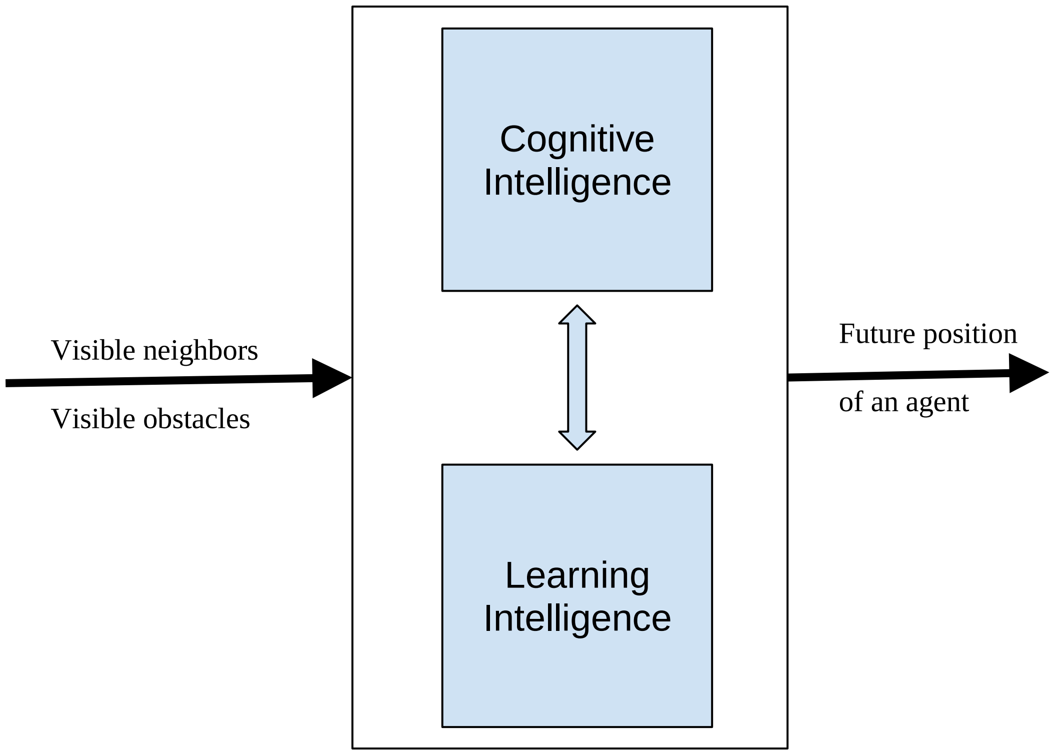


Figure 5.2: Data processing and decision making module

When data is come from the data acquiring module, the data will be processed by cognitive intelligence module and learning intelligence module. If the system doesn’t have enough historical data for prediction or system is in the initial stages, then cognitive module will handle the process of calculating future positions. If the learning module is capable enough to predict the future position, it will handle the prediction of the future positions with high priority. In some case the predicted values from learning module doesn’t have enough accuracy, then again cognitive module will handle the process. Both modules use two separate approaches to calculate future positions of the agents.

- Cognitive module
    - Cognitive module uses a genetic algorithm based approach to calculate the future position of the agent.
- Learning module
    - Learning module uses a reinforcement learning based approach to predict the future position of the agent.

5.4 Decision execution module

This module receives calculated future positions from the previous module called decision making module. Then intermediate positions will be calculated by this module and send those intermediate positions to the simulator. The simulator may be a software based simulator or hardware based simulator.

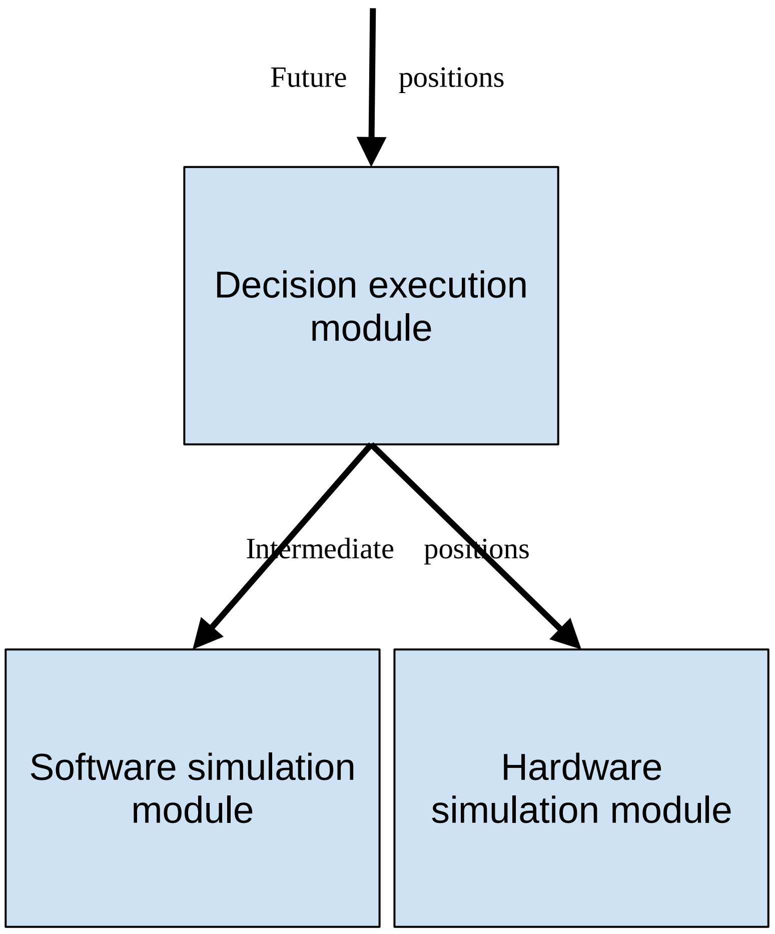

Future positions
Decision execution module
Intermediate positions
Software simulation module
Hardware simulation module

Figure 5.3: Data execution module

- Software simulation module
    - In this module it will simulate the behavior of agents within the computer itself by using a software framework.
- Hardware simulation module
    - In this module it will simulate the behavior of agents using small toy vehicles.

5.5. Indoor positioning module

Objective of this module is to calculate positions of robots (agents) within the indoor environment. This module has two sub modules called Image Capturing module and image processing module. These two modules synchronize each other for calculating positions of agents, positions of obstacle and positions of targets.

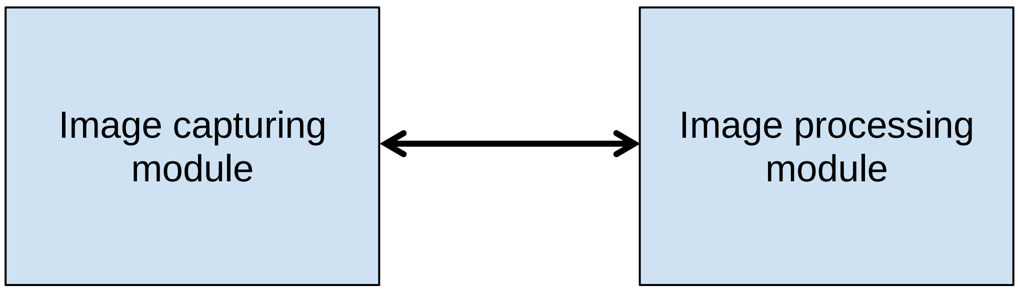


Figure 5.4: Indoor positioning module

- Image capturing module
    - This module runs inside of an android phone with a high quality camera. It captures images of the terrain and vehicles from the top the ground and sends those captured images to the image processing module in the computer.
- Image processing module
    - This module runs inside the computer and receive images from the image capturing module. Then it processes the images and identifies positions of vehicles, obstacles and targets.

5.6 Summary

This chapter explained full details description about the high level architecture and the design of the proposed system. And it has explained relationship between each module and the data flowing between each module. Next chapter will explain the implementation of each module descriptively.

Chapter 6

Implementation of Swarm intelligence based UGV controller

6.1 Introduction

In this chapter we are going to discuss how we implement the application. This chapter discusses each module in detail. Since chapter 5 provides design perspective of the application, this chapter discusses how each module is developed and technologies, methods, algorithms used to develop the application.

In the design phase we have divided the application into three main modules called Data acquisition, Data processing & decision making and decision execution. We will be discussing the implementation of each part detailed here.

6.2 Data acquisition

This is the initial part of the project. In this module agents acquire data not only from other agents but also from the environment. The data which are required from others are position information of them. Also agents gather obstacle information from the environment.

- Data acquisition from other agents
    - The first step of this project is to implement the data acquisition part of the agent. In the software simulation environment agents acquire data through a message parsing interface called JADE. This message parsing interface is implemented to use in multi agent applications. Also framework gives interface for JAVA to implement our own multi agent applications.

      Using JADE framework we can write behavior classes for representing behaviors of agents. The key behavior class which is in this system is the Communication behavior class. This behavior class is used to send position requests to other agents. Then other agents send their current positions to the requested user by using same behavior class.

- In hardware simulation environment we use radio frequency (RF) technology to transfer messages between agents. So that each agent consists with RF transmitter and RF receiver for sending and receiving the data from other agents. But for detecting vehicles system uses a computer vision approach. More details about computer vision approach will be discussed in last section in this chapter.

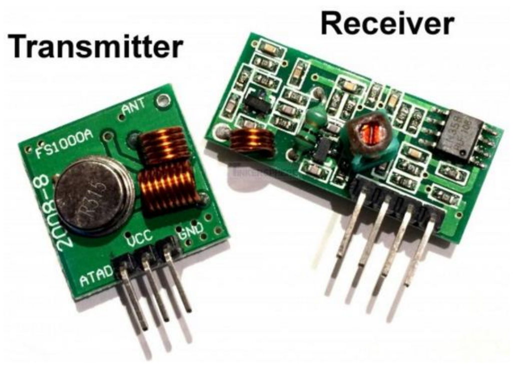


Figure 6.1: RF transmitter and receiver

- Data acquisition from the environment
  - Within software simulations, obstacle points are saved at a central file which can be access by all agents. So agents get obstacle information from that centralized file. Here agents have to submit their current position and then they will receive obstacle position near to them.
  - But in hardware simulations environment, obstacles will be detected by using a computer vision approach. Here we assume that all obstacles are circular objects.

6.3 Data processing and decision making

Whole intelligence tasks are happened at this module and it is worked as the brain of the agent. This module has another two sub modules called GA module and Machine Learning module.

6.3.1 GA approach

This module will calculate future positions of the agents using evolutionary computing approach. Since agents are getting decisions by decentralized manner, all agents have their own genetic algorithm (GA) module within their brain.

6.3.1.1 Structure of the chromosome

The structure of the chromosome is very simple with two gene values to represent X and Y coordinates of future position.

X coordinate	Y coordinate

Figure 6.2: Chromosome structure of each generation

6.3.1.2 Crossover and mutation

Here we used single point crossover as the crossover method. Swap mutations is the method that we have used to do the mutation in this GA model. By experiments we decided to use 0.1 as the mutation probability in the model.

6.3.1.3 Calculating fitness value

We consider following two simple rules to evaluate the fitness value within the fitness function.

1. Distance from current position to the start position should be maximized and distance from current position to target should be minimized.
2. Calculated position should not be colliding with any other agent or obstacle.

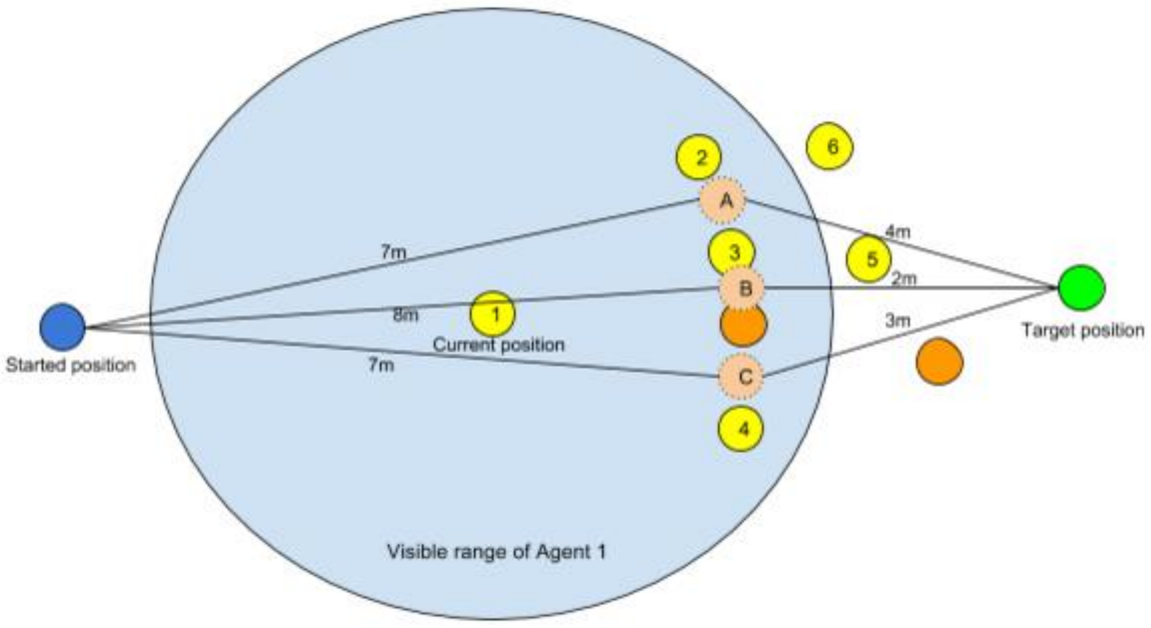


Figure 6.3: Instance of an agent navigation

In above diagram yellow colored circles denote the agents of the system and orange colored circles denotes the obstacles in the environment. Start position of the agent and target of the agent are denoted by blue and green colored circles respectively. The big light blue circle shows the visibility range of agent 1 and pink colored circles denotes next possible positions for the agent 1.

Following figure shows distances from start position to the possible positions and possible position to the target position for agent 1.

Possible position	Distance from the start	Distance to the target
A	7m	4m
B	8m	2m
C	7m	3m

Table 6.1: Distances for possible positions

Position B couldn't be existed according to the rule number 2, since it's colliding with other agents and obstacles. Then A and C are remaining possible positions for 1st agent. Distance from start to remaining positions are same for both A and C. But distance to the target is smaller in position C then A. So most of the time position C will be selected by this module. Likewise this process will be continued repeatedly until agent reaches to the target.

Other than above two rules agents also check whether any obstacle or any agent is on the path to the calculated position. If some object is on the path, agent will return small fitness value and recalculate the path.

6.3.1.4 Check objects on the path

Initially system defines two lines on current location and predicted future position which are perpendicular to the future path of the vehicle. Then system connects those two lines and creates a rectangle between current position and predicted future position. Then system checks for objects which are appeared inside of the calculated rectangle.

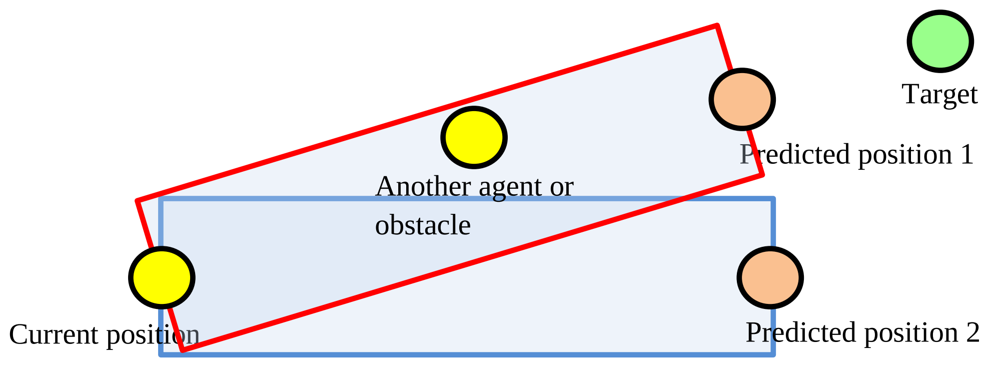


Figure 6.4: Checking disturbance on the path

Even whether above red colored path is the shortest path and there is no any collision at the predicted position, it will be removed by above rule. It means robot will not be navigated to the predicted position 1 even it's the shortest path.

Since blue colored path has no any object between current position and predicted position 2, system will choose it as the optimal path to the target.

6.3.2 Machine learning approach

This module will calculate future positions of the agents using a machine learning approach. Since agents are getting decisions by decentralized manner, all agents have their own machine learning module within their brain.

Here we used deep reinforcement learning approach which gives prominent results in last few years for learning agents. Input vector for the learning model is calculated by considering existence of objects within the visible area of agents. The correct action for the given input vector is found by looking at the output of genetic algorithm based approach.

6.3.2.1 Representation of the input

We divided visible area of an agent into 16 parts. The angle of each area is 22.5 degrees. It means learning system has size of 16 inputs for training the learning model. And also each input is contains a binary value which represents the existence of obstacle or other agents within given region. If some object is in given area, we assign 1 to corresponding input. Otherwise we assign 0 to the corresponding input.

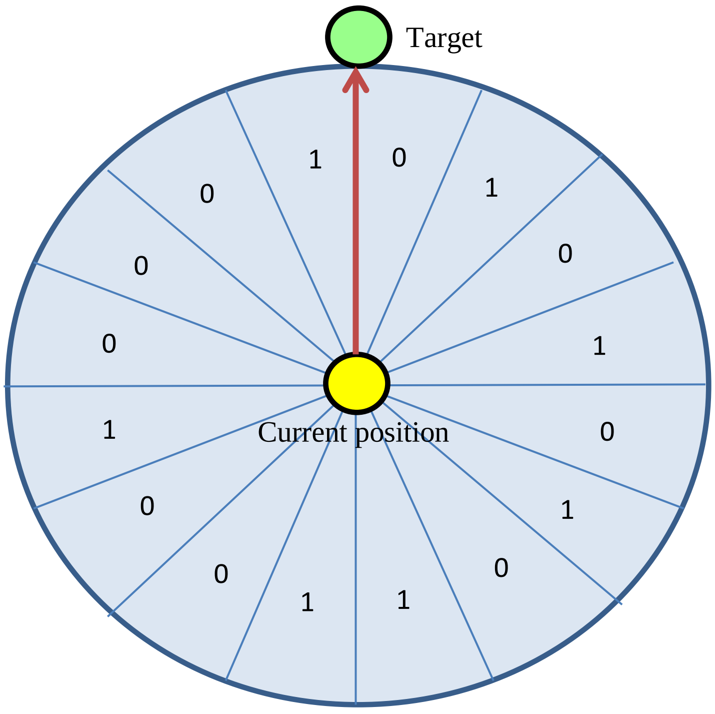


Figure 6.5: Representation of inputs with respect to area

So that final input to the learning model will be like a vector which contains 16 dimensions.

0	1	0	1	0	1	0	1	1	0	0	1	0	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Figure 6.6: Representation of inputs as a vector

6.3.2.2 Structure of the artificial neural network

In this deep Q learning algorithm, we use four layered artificial neural network to represent the Q-function of the reinforcement learning.

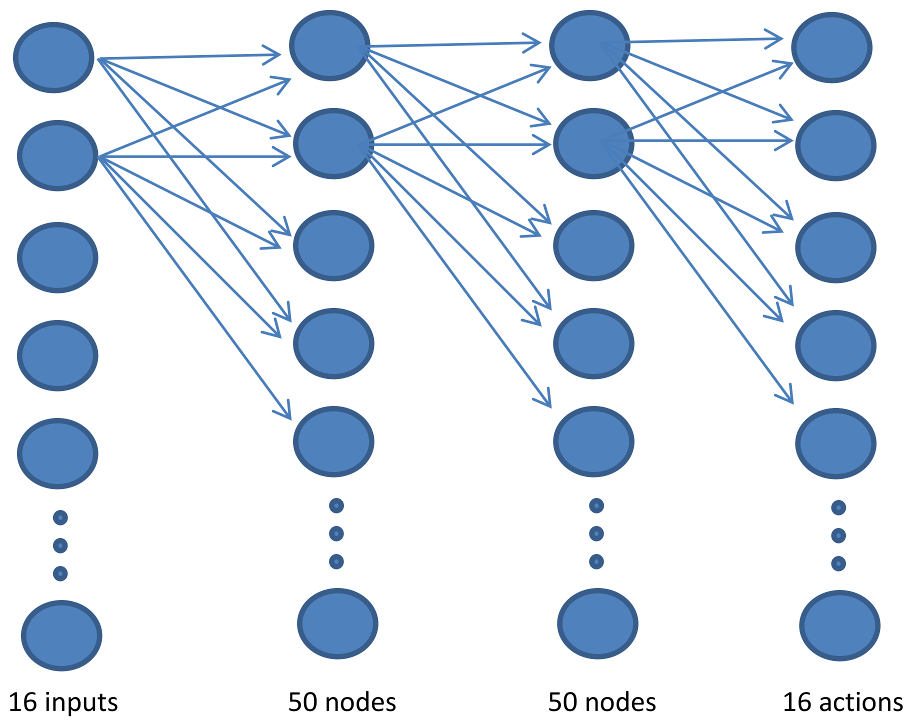


Figure 6.7: Structure of the artificial neural network

In any artificial neural network first layer is the input layer. Here we have 16 inputs to the network which represents the existence of neighbor agents and obstacles in the visible area of an agent.

In this neural network there are two hidden layers have been defined. There are fifty nodes in this each hidden layer and each node uses Rectified Linear Unit (ReLU) as the activation function.

Final layer contains sixteen nodes for representing sixteen actions that an agent can take. A simple regression function is used in this final layer as the activation function. Q-values of each action are returned from this layer as the final output.

6.3.2.3 Finding correct action

Correct action for given input is calculated by looking at output location from genetic algorithm. Here we defined the action as most possible area for next position of the agent.

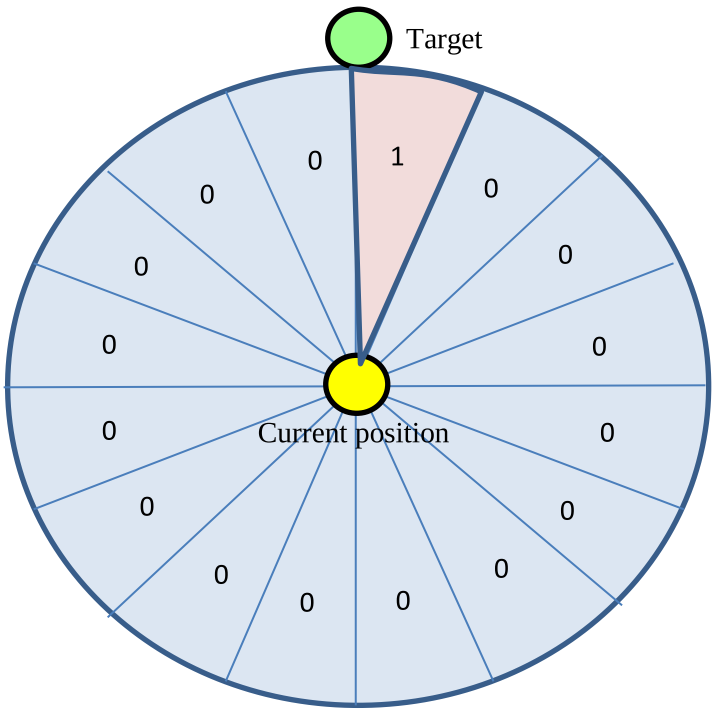


Figure 6.8: Representation of correct action with respect to area

So representation of action space will be like a vector which contains 16 dimensions for representing 16 actions which can be taken by an agent.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

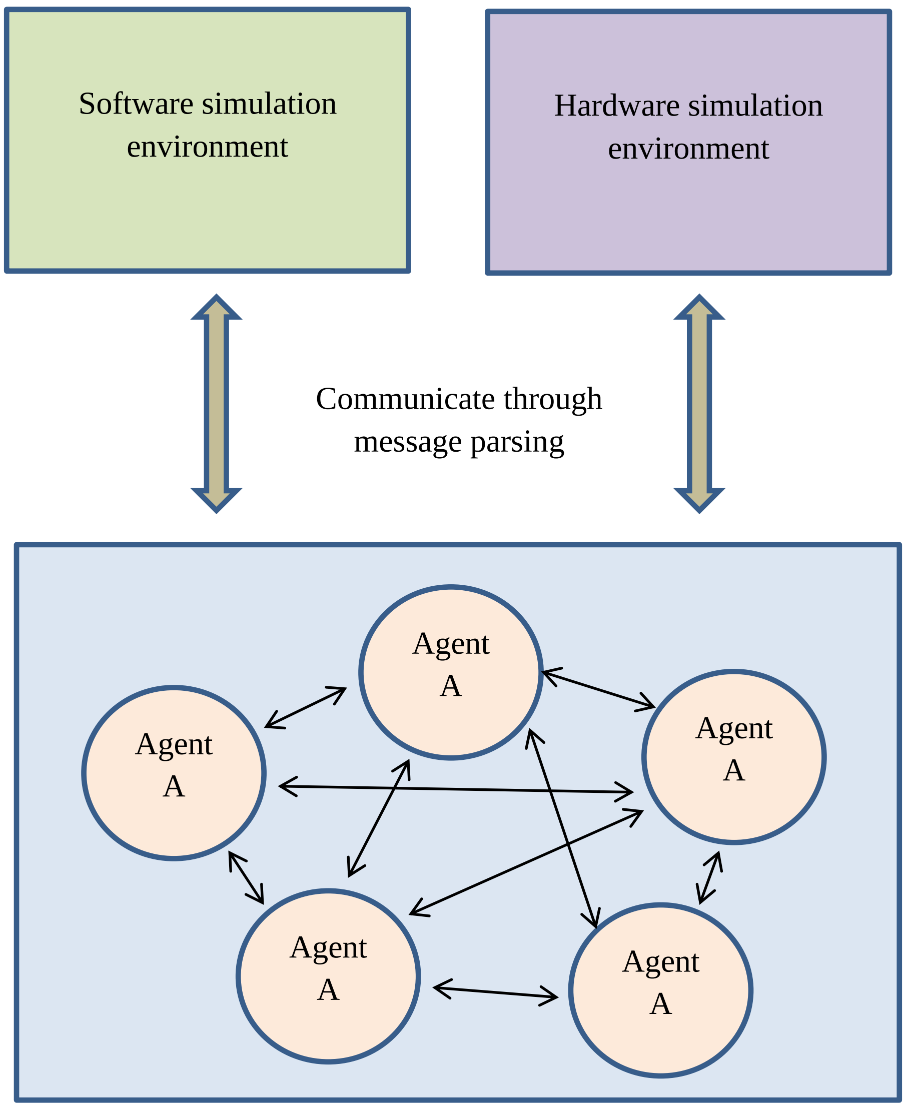
Figure 6.9: Representation of actions as a vector

In above example the selected action is the 1st action from 16 actions.

6.4 Decision execution

In this module we discuss how agents can perform tasks or navigation with calculated positions. We can discuss decision execution under two types of simulation techniques, since we have to simulate the project in both hardware and software environments.

This whole process between simulation environments and agents are separated by using java socket programing environment. Because of that, we can use same agent implementation for both simulation environments with small effort. In both cases agents are run on top of same JADE framework and they communicate with simulation environments by massage parsing.



Software simulation environment
Hardware simulation environment
Communicate through message parsing
Agent A
Agent A
Agent A
Agent A
Agent A

Figure 6.10: Whole process of communication

6.4.1 Decision execution by software simulation

Within software simulation environment we used Java based Processing framework to execute and display the navigation of each agent. It will display all agents, all obstacles, start positions and target positions using an applet form in real time. This software simulation environment acts as a server to all agents running in the jade environment. Those all agents send their positions to this software simulation environment and those positions will be displayed by the server using the Java Processing framework. After those positions have been drawn to the applet, server sends response to the relevant client saying finish.

In software simulation environment agents are denoted by yellow ellipse and its visible range is drawn using a transparent yellow colored ellipse. Usually these two ellipses are starting from same position. But their radiuses are different from each other. Obstacles and targets are denoted by red and green colored ellipse respectively.

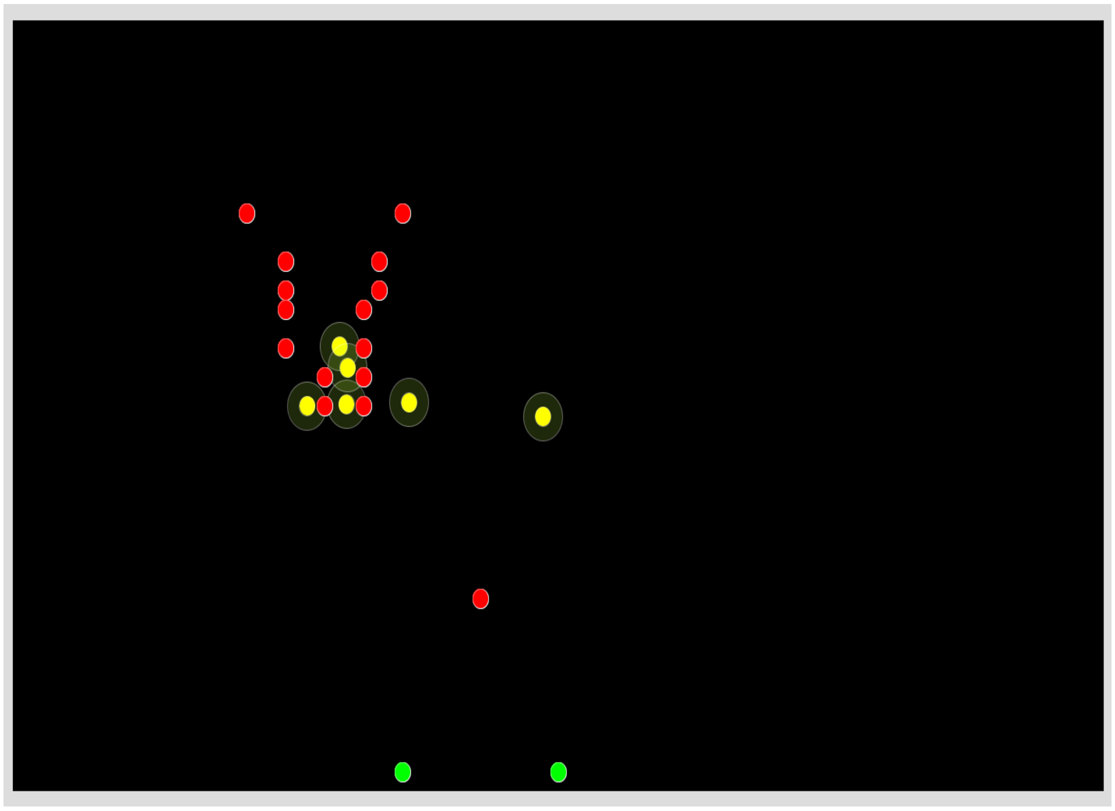


Figure 6.11: Software simulation environment

6.4.2 Decision execution by hardware simulation

For simulating robot navigation in hardware environment, we have created an Arduino based robot cars.

6.4.2.2 Robot chassis implementation

We design our robot car to work with four wheels, due to easy of controlling. Since this vehicle has separate motors per each wheel, each of vehicle acts as a four wheel drive vehicle. Most of the parts in this chassis have been created by using fiber, plastic and rubber.

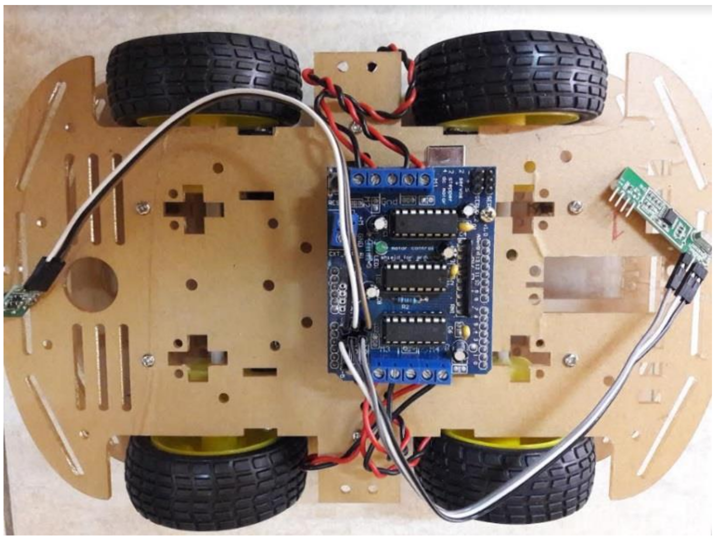


Figure 6.12: Robot car

As previously explained we used Arduino environment for programing the robots. Here we used UNO version of Arduino board for programing. Arduino UNO is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.

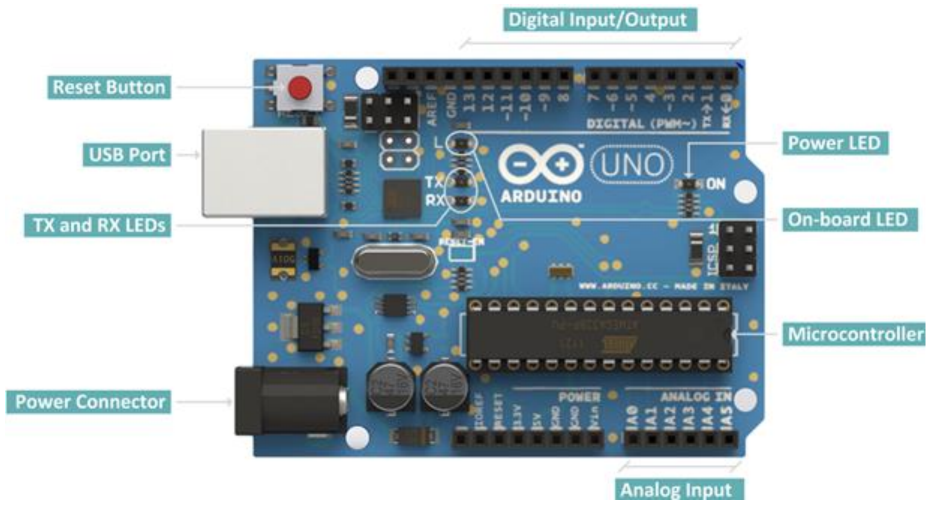


Figure 6.13: Arduino UNO board

6.4.2.2 Robot communication

Arduino UNO has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers. Here these robot cars communicate with the computer through radio frequency (RF). So every robot has RF module on top of Arduino board. Computer also has connected to another Arduino board for communicating with robot cars. The ATmega328 provides UART TTL (5V) serial communication through 0 (RX) and 1 (TX) digital pins. Then this Arduino board is connected to the computer via USB port.

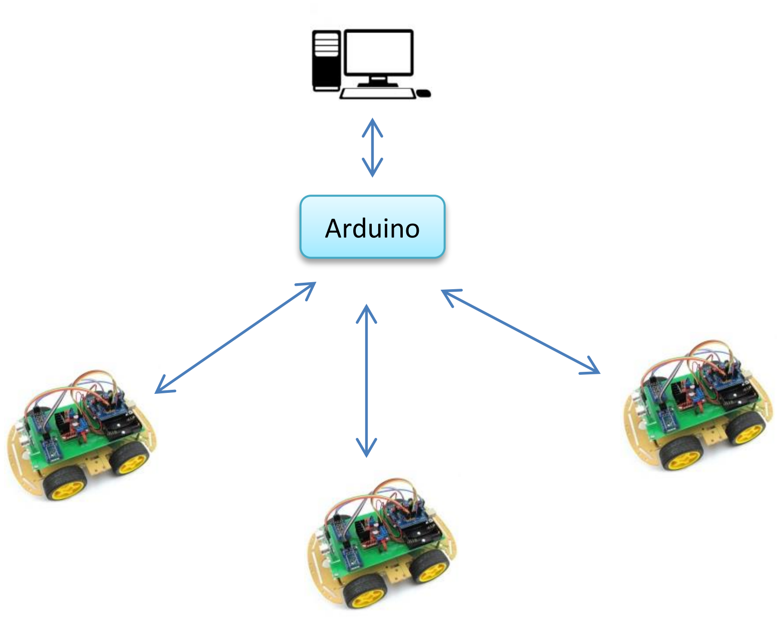


Figure 6.14: Agent communication architecture

6.4.2.3 Robot moving strategy

Previous module called decision making module sends calculated immediate next positions to each robots. Then each robot should be able move to that position with minimum error and minimum time duration. But robot cars don't have accurate rotating angle calculation or travelling distance calculation methods. It means robot cars should always calculate the error when they are moving and adjust moving angle and distance accordingly.

So if we have vehicle and target as in Figure 6.4, vehicle should be rotated to anti-clock wise until $\Theta < \alpha$, where Θ is angle between vehicle direction and target direction, α is maximum error angle which can be exist between above explained two directions. Incase vehicles is rotated more than what it is expected, then vehicle should be rotate in other direction (clockwise) until $\Theta < \alpha$ in that side. In Figure 6.5 we can see the almost rotated vehicle.

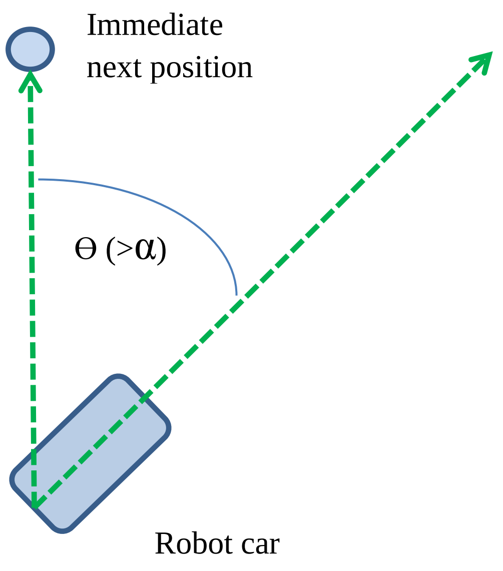


Figure 6.15: Initial stage of vehicle rotation

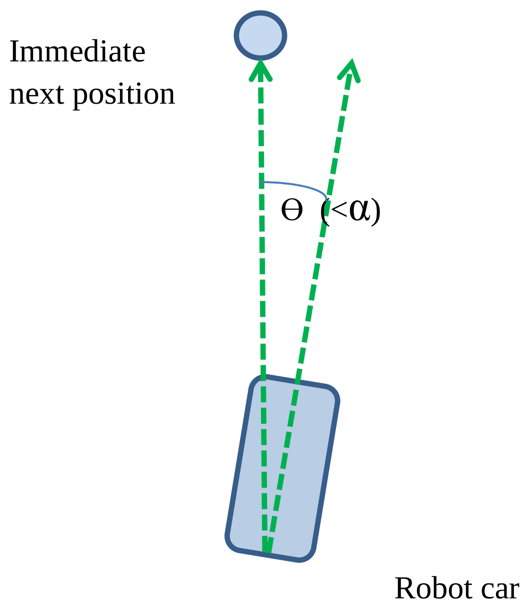


Figure 6.16: After rotating vehicle until given threshold

After the vehicle is in almost same direction with the target, it starts moving to the target. But it at some point angle (ϴ) between the vehicle moving direction and target direction can be larger than α as in Figure 6.6.

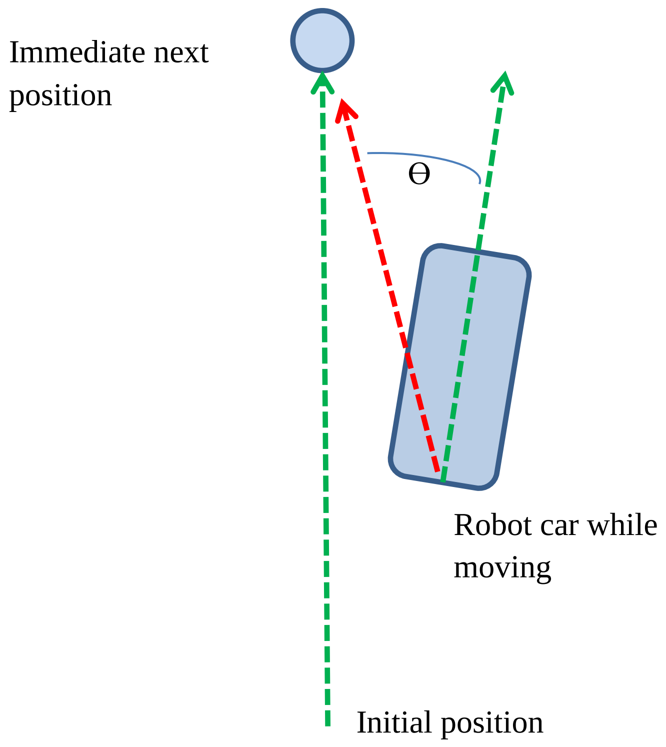


Figure 6.17: Vehicle while moving

So in this situation vehicle should be stopped and it has to be re-rotate until error is minimized. Then again it can start moving towards to the target direction. So this error checking, rotating and moving processes should be done until it reaches to the immediate next target.

6.5 Indoor positioning

In outdoor environment we can use GPS as positioning approach in unmanned autonomous vehicles. But when we navigate unmanned autonomous vehicles within indoor environment, we have to face huge trouble to calculate positions of vehicles. There are various researches are being carried in the current era of robot navigation. But still there is no very established method to do indoor positioning.

In this research we tried to solve above problem by using a computer vision approach.

6.5.1 Image capturing

Image capturing is done by using an android phone which is installed on top of the terrain. Also this android device acts as a server and broadcasts captured video stream to clients. Then computer will act as a client and captures whole video which is being broadcast by android phone.

6.5.2 Image processing

Image processing program runs inside the computer and it receives images from image capturing device. It means this program acts as a client program for the image capturing server.

Then received images are processed by various computer vision algorithms to identify locations of the vehicles, obstacles and targets. First system applies colored object detection approach for detecting vehicles, obstacles and targets. Then detected objects are mapped with the screen to get coordinates of detected objects with respect to screen size.

6.5.2.1 Colored object detection

Here we first convert image to the HSV format and then we use this HSV image for future analysis. Then we capture specific colored areas of the image by applying threshold to HSV values. After that we apply Gaussian smoothing technique to smooth detected regions of the image. Then we do a contour detection method to identify detected pixel values in the image. After that we calculate size of the area of detected contours. Using this area size, we filter largest blobs that we have detected according to our requirements.

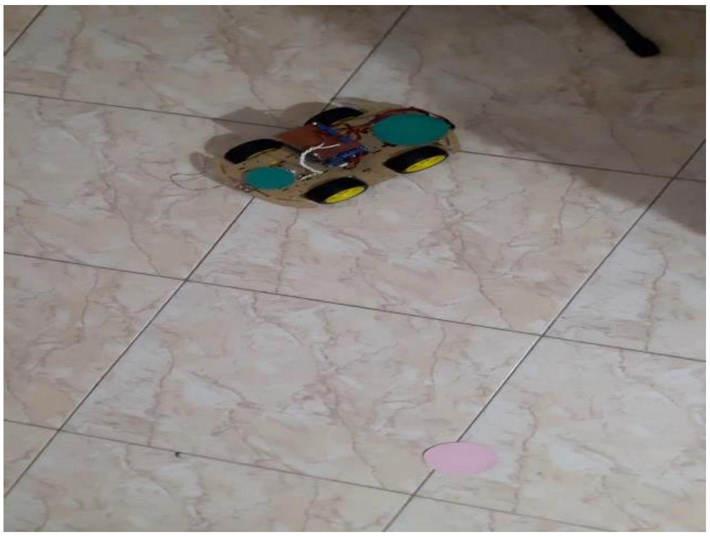


Figure 6.18: Original image for detecting pink colored object

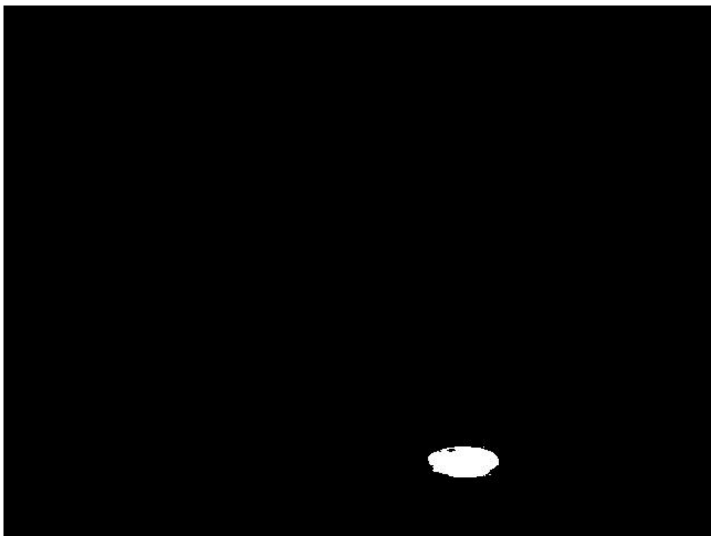


Figure 6.19: Threshold image for detecting pink colored object

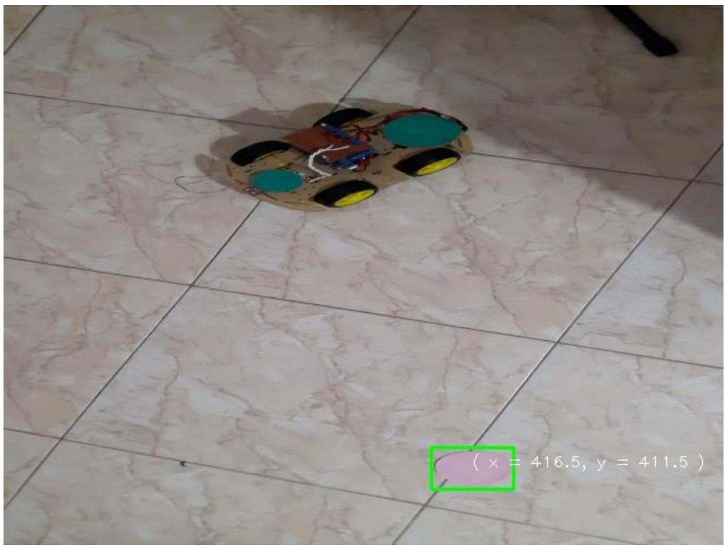


Figure 6.20: Image after detecting pink colored object

- Vehicle detection
    - Each vehicle contains same colored two circles in front of the vehicle and the back side of the vehicle. But front sided circle is smaller than backside circle. Using above explained colored object detection approach system detects those two blobs on the vehicle and calculates direction vector of the vehicle and the center position of the vehicle. Then using detected center position and direction vector, system generate circular boundary for the vehicle.
- Obstacle detection
    - All obstacles are colored in yellow color and system detects those colored objects for calculating positions of obstacles.
- Target detection
    - All targets are colored in pink color and system detects those colored objects for calculating positions of targets.

6.6 Summary

In this chapter first we have discussed data acquisition methods of both hardware and software simulations. Then we discussed data processing and decision making approaches. Under that topic we have discussed genetic algorithm based approach to predict the next most possible position of an agent. Finally we have discussed data processing method within software simulation environment. And next chapter will explain overall conclusion of the research.

Chapter 7

Evaluation

7.1. Introduction

Since this research deals with genetic algorithm, machine learning and mathematical modeling, evaluation is much needed to verify the techniques used in this research. In this chapter, various evaluations are carried out on the various part of the research. Here we mainly evaluated accuracy and performance of the system.

In this system it's very much difficult to think about direct evaluation technique for whole system. So here we do a component wise evaluation of the system

7.2. Evaluation of genetic algorithm model

Since our chromosome contains only two gene values, we don't have to worry about crossover. We used single point crossover as the crossover technique in every evaluation. But we have changed the method of mutation. Swap mutator and Gaussian mutator are two methods that we have tried in our analysis. Also we tried various probability values for the mutation as well.

For the evaluation we used three hundred calculations with different conditions. To obtain different conditions, we navigated agents in a more complex environment with many obstacles.

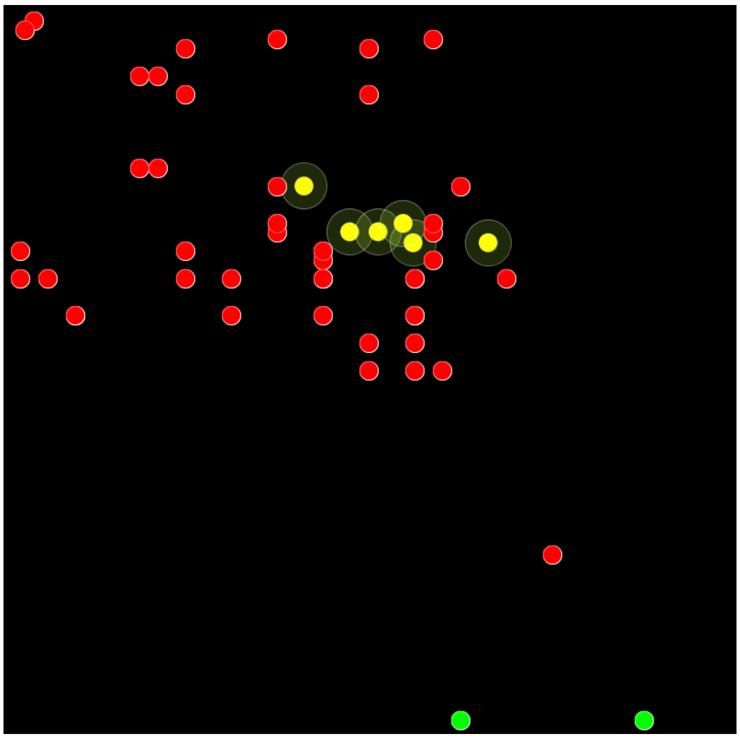
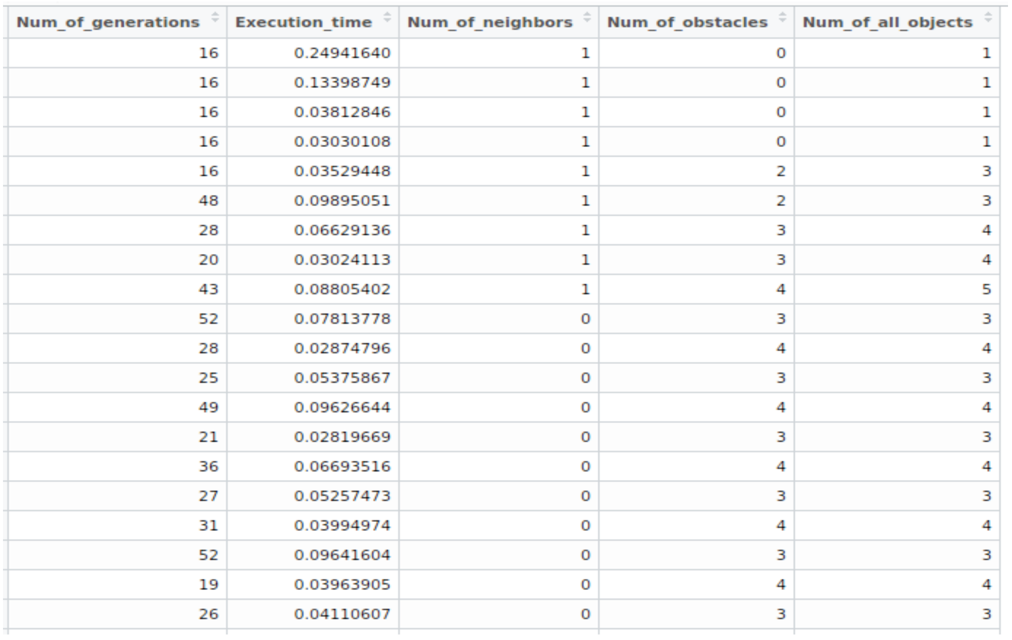


Figure 7.1: Complex environment for gathering data



Num_of_generations	Execution_time	Num_of_neighbors	Num_of_obstacles	Num_of_all_objects
16	0.24941640	1	0	1
16	0.13398749	1	0	1
16	0.03812846	1	0	1
16	0.03030108	1	0	1
16	0.03529448	1	2	3
48	0.09895051	1	2	3
28	0.06629136	1	3	4
20	0.03024113	1	3	4
43	0.08805402	1	4	5
52	0.07813778	0	3	3
28	0.02874796	0	4	4
25	0.05375867	0	3	3
49	0.09626644	0	4	4
21	0.02819669	0	3	3
36	0.06693516	0	4	4
27	0.05257473	0	3	3
31	0.03994974	0	4	4
52	0.09641604	0	3	3
19	0.03963905	0	4	4
26	0.04110607	0	3	3

Figure 7.2: The sample dataset for the evaluation of genetic algorithm model

7.2.1 Comparison of results from swap mutation and Gaussian mutation

	Swap mutation	Gaussian mutation
Mean of number of generation	22.44333	22.49667
Mean execution time	0.0466978	0.05138982

Table 7.1: Comparison swap mutation and Gaussian mutation

Here we have calculated mean value of number of generations in each mutator used. From above results, the mean value of number of generation and mean value of execution time of Gaussian mutation is higher than swap mutation. But in both methods accuracy we have obtained is equal. So that we can say that swap mutation is efficient for this problem than Gaussian mutation.

After selecting swap mutation as the mutation technique, we had to select the best probability for the mutation. In here also we did experiments and figured out best possible probability for the mutation.

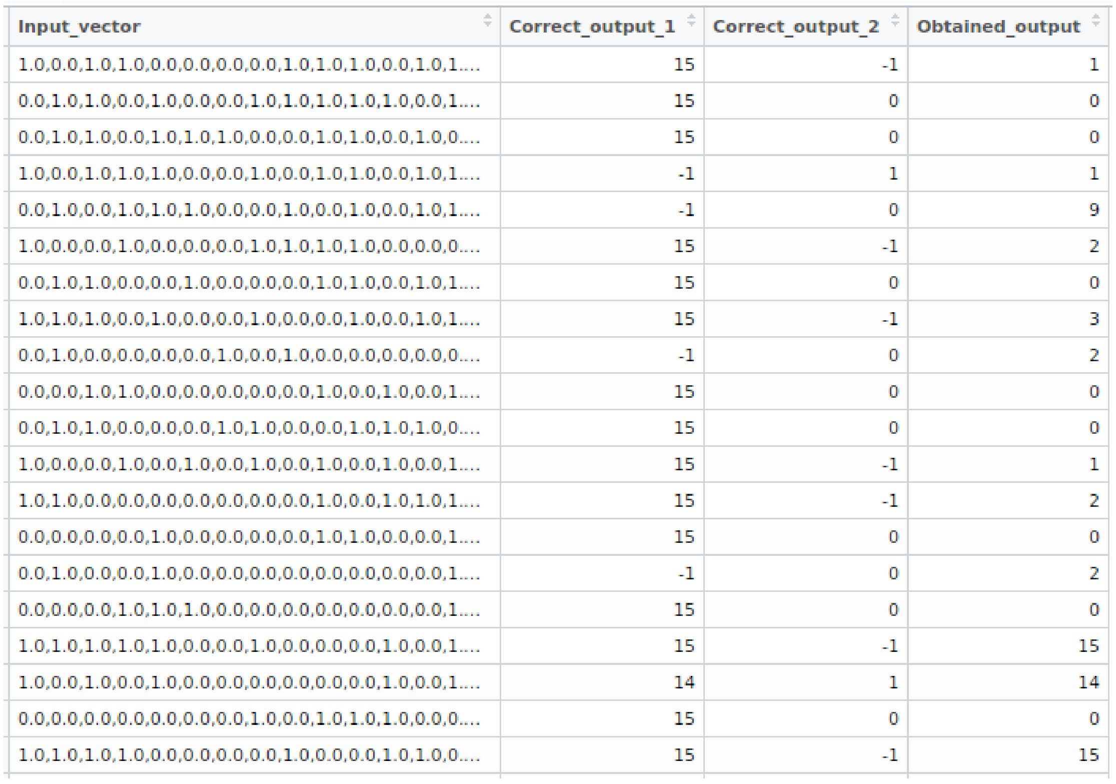
Probability	Mean of number of generation	Mean execution time
0.01	23.54456	0.03726065
0.10	22.43333	0.04247326
0.25	22.44333	0.04669787
0.50	22.42667	0.05267275
0.99	22.78333	0.07611611

Table 7.2: Comparison of probabilities of mutation with respect to number of generations and execution time

By looking at above table, we can choose 0.10 as the mutation probability for our research.

7.3. Evaluation of machine learning model

We first evaluated the accuracy of the results with respect to size of training sample. We tried it with three different sizes of training samples as follows.



Input_vector	Correct_output_1	Correct_output_2	Obtained_output
1.0,0.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,1.0,1.0,0.0,1.0,1....	15	-1	1
0.0,1.0,1.0,0.0,1.0,0.0,0.0,1.0,1.0,1.0,1.0,1.0,0.0,1....	15	0	0
0.0,1.0,1.0,0.0,1.0,1.0,1.0,0.0,0.0,1.0,1.0,0.0,1.0,0....	15	0	0
1.0,0.0,1.0,1.0,1.0,0.0,0.0,1.0,0.0,1.0,1.0,0.0,1.0,1....	-1	1	1
0.0,1.0,0.0,1.0,1.0,1.0,0.0,0.0,1.0,0.0,1.0,0.0,1.0,1....	-1	0	9
1.0,0.0,0.0,1.0,0.0,0.0,0.0,1.0,1.0,1.0,1.0,0.0,0.0,0....	15	-1	2
0.0,1.0,1.0,0.0,0.0,1.0,0.0,0.0,0.0,1.0,1.0,0.0,1.0,1....	15	0	0
1.0,1.0,1.0,0.0,1.0,0.0,0.0,1.0,0.0,0.0,1.0,0.0,1.0,1....	15	-1	3
0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0....	-1	0	2
0.0,0.0,1.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,1....	15	0	0
0.0,1.0,1.0,0.0,0.0,0.0,1.0,1.0,0.0,0.0,1.0,1.0,1.0,0....	15	0	0
1.0,0.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1....	15	-1	1
1.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,1.0,1....	15	-1	2
0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,1.0,0.0,0.0,1....	15	0	0
0.0,1.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1....	-1	0	2
0.0,0.0,0.0,1.0,1.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1....	15	0	0
1.0,1.0,1.0,1.0,1.0,0.0,0.0,1.0,0.0,0.0,0.0,1.0,0.0,1....	15	-1	15
1.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1....	14	1	14
0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,1.0,1.0,0.0,0....	15	0	0
1.0,1.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,1.0,1.0,0....	15	-1	15

Figure 7.3: The sample dataset for the evaluation of machine learning model

After the training of the model, we created hundred random inputs. Then those hundred inputs were applied to trained models. Finally we check the output with the actual result and counted all correct attempts. Following table illustrates the comparison between trained models.

Trained models	Percentage of correct outputs	Percentage of incorrect outputs
Trained model with 5000 samples	41	49
Trained model with 10000 samples	57	43
Trained model with 25000 samples	98	2

Table 7.3: Comparison of trained machine learning models with respect to accuracy

7.4 Summary

After evaluating the system with different aspects, it can be concluded that designed system can be used at acceptable level with respect to accuracy and performance. System not only evaluates the final system, but also evaluates theoretical parts of the proposed approach.

Chapter 8

Conclusion & Further work

8.1 Introduction

This chapter explains the small conclusion of the proposed system. Also then it will explain the future work of the research. As in every research we have many problems while we were doing the research. Time and resources were huge problems in this research even from the beginning.

So this chapter explains limitations of the research, future works of the research and the final conclusion of the research.

8.2 Conclusion

The aim of this project was to develop swarm intelligence based unmanned ground vehicles (UGV) controlling system to work in dynamic and partially observable environment. Research was divided into multiple sections and accomplished each tasks as in each chapter.

Final system has been implemented to navigate unmanned vehicles in both software and hardware simulation environments by using genetic algorithm and machine leaning approaches. Here we implemented the system only by using partially observable information, since in real life also human driver gets decisions from the partially observable environment.

Learning and cognitive thinking are the base concepts behind this whole project. It is also reflects the actual way of thinking by humans when they do any kinds of work. If a human does not have any background or knowledge about any given task, first he tries to solve that given task by explicitly thinking about the solution. But if he has enough knowledge in given problem or task, he tries to solve it using his past knowledge about similar kinds of problems. If it doesn’t work, then only he goes to the cognitive approach. We thoroughly believed this natural way of thinking by humans, also gives intelligent solutions for machines as well. Final results of the system proved the success of the research.

As we explained as the cognitive approach we suggested to use genetic algorithm based model. And we used deep Q learning model as the Learning model in the proposed system. Though this research has provided its main objectives, still there are lots of areas to improve by technically and well as content wise.

8.3 Limitations and Further Work

In this system we assumed all obstacles as ellipses for ease of implementation. So this research can be extended to work with various types of obstacles. Also agents stuck on the terrain, when obstacles are appeared as a non-convex polygon. It can also be addressed in future researches.

8.4 Summary

In this chapter we have discussed the conclusions that we can finally derived from current state of the research. We consider many aspects of the project like objectives, design, and implementation. Based on all the facts, we made some conclusions here. Limitations of the solution and further work are also discussed here.

References

[1] HaiYang Chao, YongCan Cao, and YangQuan Chen, Utah State University, (2010), Autopilots for Small Unmanned Aerial Vehicles: A Survey

[2] Ollero, Aníbal, Maza, Iván, (2007), Multiple Heterogeneous Unmanned Aerial Vehicles

[3] Brandão, A., Martins, F. and Soneguetti, H. (2015) ‘A vision-based line following strategy for an autonomous UAV’, Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics

[4] Paweł Burdziakowski, Marek Przyborski, Jakub Szulwic,(2015), A vision-based unmanned aerial vehicle navigation method, 1st International Conference on Innovative Research and Maritime Applications of Space Technology

[5] Aakash Dawadee, Javaan Chahl, D(Nanda) Nandagopal and Zorica Nedic(University of South Australia), (2013), Landmark Feature Signatures for UAV Navigation, IEEE Conference on Control, Systems & Industrial Informatics

[6] J.-P. Ramirez-Paredes, E. A. Doucette, J. W. Curtis, N. R. Gans, (2015) Urban Target Search and Tracking Using a UAV and Unattended Ground Sensors, American Control Conference (ACC).

[7] Austin M. Jensen, David K. Geller (Utah State University), YangQuan Chen(University of California, Merced), (2013), Monte Carlo Simulation Analysis of Tagged Fish Radio Tracking Performance by Swarming Unmanned Aerial Vehicles in Fractional Order Potential Fields

[8] F.M. Raimondi, Maurizio Melluso,(2013), Stability and Noises Evaluation of Fuzzy/Kalman UAV Navigation System

[9] Ioannis K Nikolos, K.P. Valavanis, Nikos Tsourveloudis, A N Kostaras (Dept. of Production Eng. & Manage., (2003), Tech. Univ. of Crete, Chania, Greece),

Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)

[10] G e r h a r d WeiB, (1993), Learning to Coordinate Actions in M u l t i - A g e n t Systems, Proceedings of the 13th international joint conference on Artifical intelligence

[11] J. Andrew Bagnell, Jeff G. Schneider, (2001), Autonomous Helicopter Control Using Reinforcement Learning Policy Search Methods, Proceedings of the International Conference on Robotics and Automation

[12] Todd Hester, Peter Stone, (2013), TEXPLORE: Real-Time Sample-Efficient Reinforcement Learning for Robots, AAAI Technical Report SS-12-02, Designing Intelligent Robots: Reintegrating AI

[13] Armin Hornung, Maren Bennewitz, Cyrill Stachniss, Hauke Strasdat, Stefan Oßwald, Wolfram Burgard, (2010), Learning Adaptive Navigation Strategies for Resource-constrained Systems

[14] B. Bischoff1, D. Nguyen-Tuong1,I-H.Lee1, F. Streichert1and A. Knoll, (2013), Hierarchical Reinforcement Learning for Robot Navigation, ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium)

[15] Dean A. Pomerleau, (2008), Efficient Training of Artificial Neural Networks for Autonomous Navigation

[16] A. Stafylopatis, K. Blekas, (1998), Autonomous vehicle navigation using evolutionary reinforcement learning, European Journal of Operational Research

[17] Quoc V. Le (Google Inc., USA), (2012), Building high-level features using large scale unsupervised learning, International Conference on Machine Learning, Edinburgh

[18] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, (2012), ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems 25 (NIPS 2012)

[19] Karthik Narasimhan, Tejas D Kulkarni, Regina Barzilay, (2015), Language Understanding for Text-based Games using Deep Reinforcement Learning, Conference on Empirical Methods in Natural Language Processing

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller (DeepMind Technologies), (2013), Playing Atari with Deep Reinforcement Learning

[21] Volodymyr Mnih, Koray Kavukcuoglu1, (2015), Human-level control through deep reinforcement learning

[22] Karthik Narasimhan, Tejas D Kulkarni, Regina Barzilay, (2015), Language Understanding for Text-based Games using Deep Reinforcement Learning, Conference on empirical methods in natural language processing

[23] Shafiq Alam, Gillian Dobbie, Yun Sing Koh, Patricia Riddle, Saeed Ur Rehman, (2014), Research on particle swarm optimization based clustering: A systematic review of literature and techniques

[24] Ulrich Bodenhofer, (2003), Genetic Algorithms: Theory and Applications

[25] Michael Nielsen, (2015), Neural Networks and Deep Learning

Appendix A

Software simulation of the proposed system

A.1 Introduction

This will explain the software simulated output of the proposed system. Explanation will be done through some screenshots of special scenarios of the system.

A.2 Special movements of the proposed system

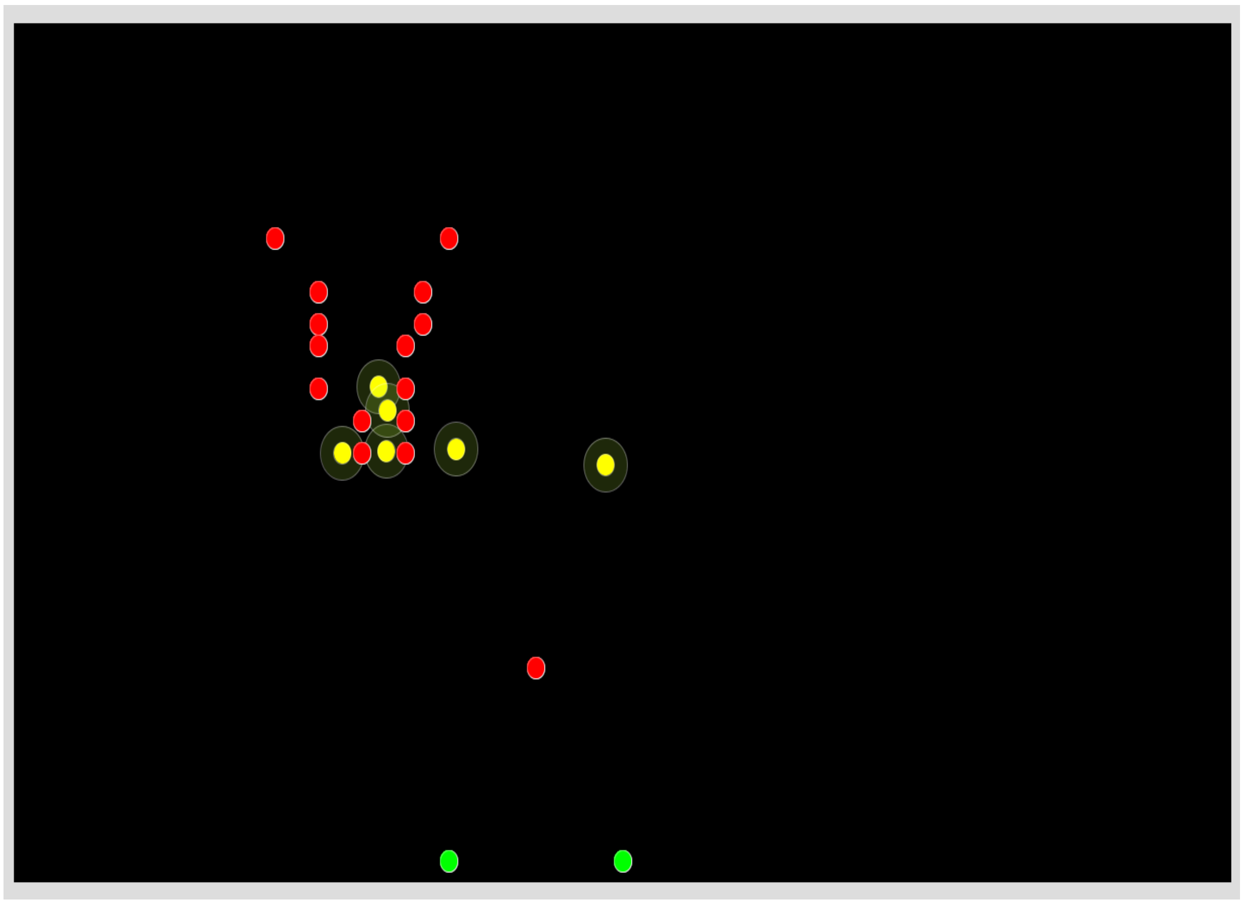


Figure A.1: Multiple agents are avoiding multiple obstacles

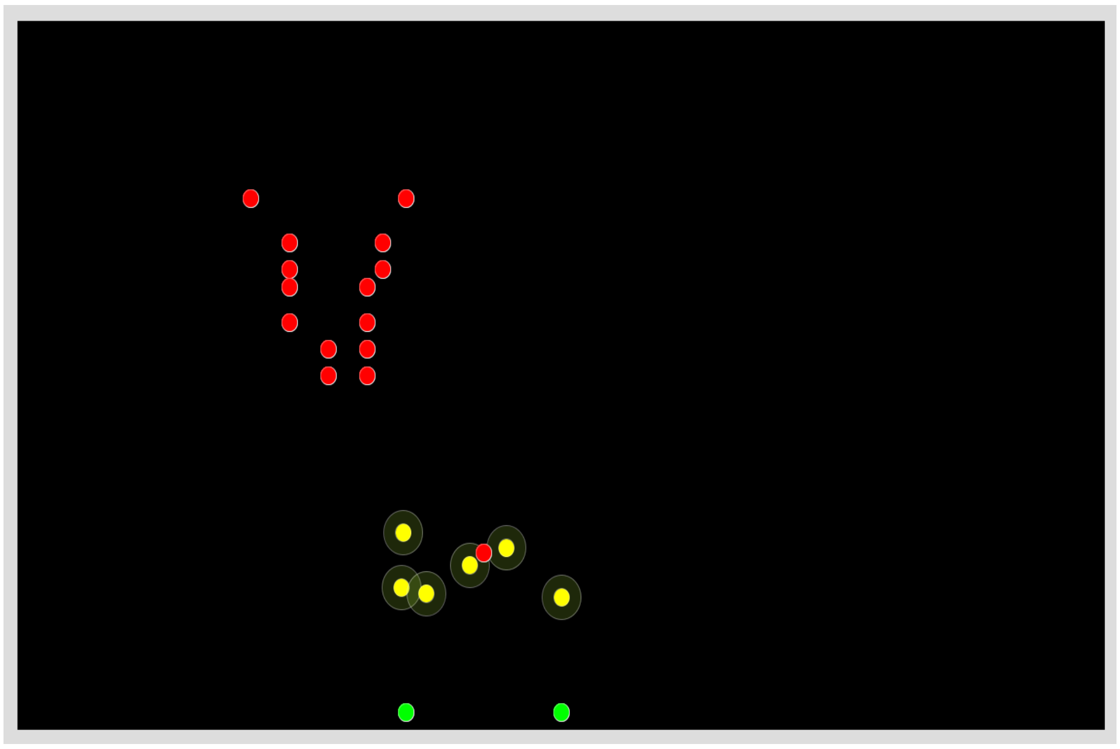


Figure A.2: Multiple agents are avoiding a single obstacle

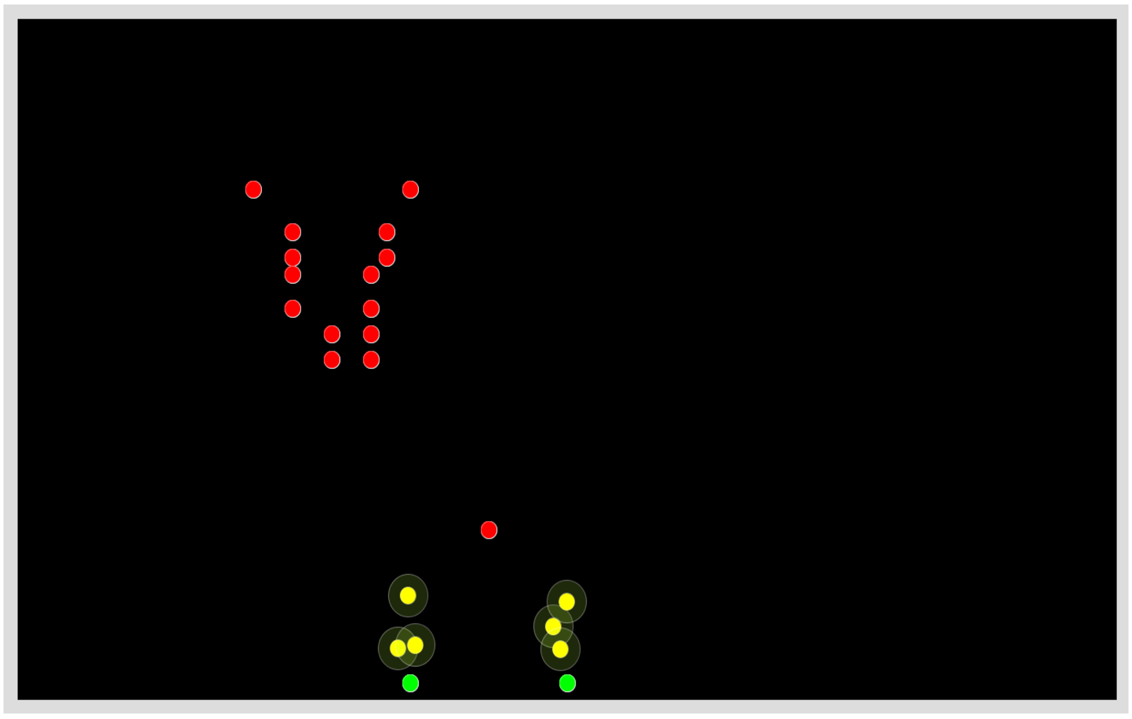


Figure A.3: Multiple agents are reaching to their targets as swarms

Appendix B

Source code of the current system

B.1 Introduction

This section will include source code which was used to develop the proposed system.

B.2 Source code of the proposed system

B.2.1 Source code for software simulator

```
package processing;
import math.MathUtil;
import terrain.Position;
import terrain.PositionUtil;
import util.Info;
import processing.core.*;
import shared.CurrentPositions;
public class MyProcessingSketch extends PApplet {
  float x = 100;
  float y = 100;
  float angle1 = (float) 0.0;
  float segLength = 50;
  public void settings()
  {
    size(1400, 800);
  }
  public void setup()
{
    strokeWeight((float) 1.0);
    stroke(255, 100);
    CurrentPositions.init();
```

```
}
public void draw() {
  background(0);
  if (!CurrentPositions.concurrentMap.isEmpty()) {
    for (int i = 0; i < CurrentPositions.concurrentMap.size(); i++) {
      x = (float) ((Position) (CurrentPositions.concurrentMap.get(i))).getX();
      y = (float) ((Position) (CurrentPositions.concurrentMap.get(i))).getY();
      ellipseMode(CENTER);
      fill(192, 255, 62, 40);
      ellipse(x, y, 50, 50);


      fill(255,255,0);
      ellipse(x, y, 20, 20);
      fill(0,255,0);
      for(Position pos : Info.targetPositions)
      {
        ellipse((float) pos.getX(), (float) pos.getY(), 20, 20);
      }
      fill(255,0,0);
      for(Position pos : Info.obstaclePositions)
      {
        ellipse((float) pos.getX(), (float) pos.getY(), 20, 20);
      }
    }
  }
}
void segment(float x, float y, float a) {
  pushMatrix();
  translate(x, y);
```

```
    rotate(a);
    line(0, 0, segLength, 0);
    popMatrix();
  }
}
```

B.2.2 Source code for GA module

```
package genetic;

import java.io.Serializable;

import java.util.function.Function;

import org.jenetics.Chromosome;

import org.jenetics.DoubleGene;

import org.jenetics.Genotype;

import terrain.Position;

import terrain.PositionUtil;


final class MyFitnessFunction implements Function<Genotype<DoubleGene>,
Double>, Serializable {

 private static final long serialVersionUID = 1L;

 private Position[] neighborPositions;

 Position initialPosition;

 Position targetPosition;

 double vehicleRadius = 10.0;

 double maxNavDisPerStep = 10.0;

 int dimensions = 2;

 int numOfConstrains = 3;

 Position[] obstaclePositions;

 Position currentPosition;

 public MyFitnessFunction(Position currentPosition, Position[] neighborPositions,
Position initialPosition,

   Position targetPosition, Position[] obstaclePositions) {

  this.neighborPositions = neighborPositions;

  this.initialPosition = initialPosition;

  this.targetPosition = targetPosition;

  this.currentPosition = currentPosition;

  this.obstaclePositions = obstaclePositions;
```

```
}
@Override
public Double apply(final Genotype<DoubleGene> genotype) {
 Double retVal = 1000.0;
 final Chromosome<DoubleGene> chromosome = genotype.getChromosome();
 int length = chromosome.length();
 Position position = new Position(currentPosition.getX() + chromosome.getGene(0).getAllele(),
   currentPosition.getY() + chromosome.getGene(1).getAllele(), 0);
 // check inter agent collision
 for (int i = 0; i < neighborPositions.length; i++) {
  if (neighborPositions[i].getDistance(position) < vehicleRadius * 2.2) {
   return 0.0;
  } else {
   retVal = retVal + 1 / (((neighborPositions[i].getDistance(position))
     / (numOfConstrains * 10 * (neighborPositions.length - 1))));
  }
 }
 // check obstacle collision
 for (int i = 0; i < obstaclePositions.length; i++) {
  if (obstaclePositions[i].getDistance(position) < vehicleRadius * 2.2) {
   return 0.0;
  }
 }
 // check moving out from initial position
 retVal = retVal + (position.getDistance(initialPosition) / (numOfConstrains));


 // check check moving in to target position
 retVal = retVal - (position.getDistance(targetPosition) / (numOfConstrains));
 return retVal;
 }
}
```

B.2.3 Source code for multi agent communication

```
package behaviors;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import brain.RobotKnowledgeBase;
import genetic.PathPlanner;
import jade.core.AID;
import jade.core.behaviours.SimpleBehaviour;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import multiagent.RobotAgent;
import terrain.Position;
import util.Info;
import util.MyLog;
public class CommunicationBehavior extends SimpleBehaviour {
 private int agentId;
 private static final MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.INFORM);
 RobotAgent agent;
 RobotKnowledgeBase rkb;
 MyLog mylog;
 PathPlanner pathPlanner;
 int collectedCount;
 BufferedReader in = null;
 BufferedWriter brout = null;

 public CommunicationBehavior(MyLog myLog, RobotAgent agent, int agentId,
RobotKnowledgeBase rkb)
   throws UnknownHostException, IOException {
  super(agent);
  this.mylog = myLog;
  this.agent = agent;
  this.agentId = agentId;
  this.rkb = rkb;
  this.pathPlanner = new PathPlanner();
  this.collectedCount = 0;
  init();
  mylog.log("Inialized");
 }

 public void init() throws UnknownHostException, IOException {

  new Thread() {
   public void run() {
    Socket socket;
    try {
     socket = new Socket(Info.serverHost, Info.serverPort);
     in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
     brout = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
     String line;
     while (true) {
      try {
       line = in.readLine();

       if (line != null) {
        String[] spt1 = line.split("_");
        sleep(1000);
        System.out.println(
          agentId + " --> " + "CommunicationBehavior --> " + spt1[0] + " received");

        if (spt1[0] == "Obstacle") {
         mylog.log("Received Obstacle from sever");
         mylog.testLog("Received Obstacle from sever");
         updateObstacles(spt1[1]);
         //calculateNextPosition();
        }
        if (spt1[0] == "Completed")
        {
         mylog.log("Received Completed from server");
         mylog.testLog("Received Completed from server");
        }

        calculateNextPosition();
       }

      } catch (IOException e) {
       e.printStackTrace();
      } catch (InterruptedException e) {
       e.printStackTrace();
      }
     }
    } catch (IOException e1) {
     e1.printStackTrace();
    }
   }
  }.start();
 }

 private void updateObstacles(String msg) {
  String[] spt2 = msg.split(",");
  Position newObstacle = new Position(Double.parseDouble(spt2[0]),
Double.parseDouble(spt2[1]),
    Double.parseDouble(spt2[2]));
  rkb.addObstacle(newObstacle);
  mylog.log("Updated obstacles");
```

```
}

void calculateNextPosition() {
 rkb.setTargetPosition(rkb.getTargetPositions().get(agentId % 2));
 ACLMessage reqMessage = new ACLMessage(ACLMessage.INFORM);
 reqMessage.setContent("PositionRequest_" + agentId + "_Empty");
 for (int i = 0; i < Info.numOfRobotAgents; i++) {
  if (i != agentId) {
    AID driver = new AID("agent" + i + "@" + myAgent.getHap(), AID.ISGUID);
    reqMessage.addReceiver(driver);
    mylog.log("Sent position request to " + i);
  }
 }
 mylog.log("Sent position request to all others");
 this.rkb.removeAllNeighbors();
 myAgent.send(reqMessage);
}

public void sendPositionToSimulator(Position position) {
 try {
  brout.write("SimulateAgent" + "_" + agentId + "_" + position.getX() + "," +
position.getY() + ","
    + position.getZ());
  brout.newLine();
       brout.flush();
       mylog.log("Sent new position to simulator: " + position.toString());
  mylog.testLog("Sent new position to simulator: " + position.toString());
 } catch (IOException e) {
  e.printStackTrace();
 }
}

public void sendPositionToOthers(Position position) {
 ACLMessage aclMessage = new ACLMessage(ACLMessage.INFORM);
```

```
  aclMessage.setContent("PositionUpdate" + "_" + agentId + "_" + position.getX() +
"," + position.getY() + ","
    + position.getZ());

  for (int i = 0; i < Info.numOfRobotAgents; i++)
   if (i != agentId)
    aclMessage.addReceiver(new AID("agent" + i, AID.ISLOCALNAME));
  myAgent.send(aclMessage);
  mylog.log("Sent new position to other agents");
 }

 @Override
 public void action() {
  ACLMessage aclMessage = myAgent.receive(mt);
  if (aclMessage != null) {
   String data[] = aclMessage.getContent().split("_");
   String event = data[0];
   int sender = Integer.parseInt(data[1]);
   String content = data[2];
   mylog.log("ACTION" + " --> " + event + " _ " + sender  + " _ " + content);
   if (event.compareTo("PositionResponse") == 0)
   {
    mylog.log("Received PositionResponse from " + sender);
    this.rkb.addNeighbor(new Position(content));
    collectedCount++;
    if (collectedCount % (Info.numOfRobotAgents - 1) == 0)
    {
     pathPlanner.calculateNextPosition(rkb);
     mylog.testLog("Calculated next position");
     sendPositionToSimulator(rkb.getCurrentPosition());
     mylog.log("Calculated next position");
    }
   }
   if (event.compareTo("PositionRequest") == 0)
```

```
    {
     mylog.log("Recieved PositionRequest from " + sender);
     ACLMessage reply = new ACLMessage(ACLMessage.INFORM);
     reply.setContent("PositionResponse_" + agentId + "_" +
rkb.getCurrentPosition().toString());
     reply.addReceiver(new AID("agent" + sender + "@" + myAgent.getHap(),
AID.ISGUID));
     myAgent.send(reply);
     mylog.log("Sent Position response to " + sender);
    }
   }
  }
 @Override
 public boolean done() {
  // TODO Auto-generated method stub
  return false;
 }
}
```

B.2.4 Source code for learning module

```
package machinelearning;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.util.ArrayList;

import javax.script.Invocable;

import javax.script.ScriptEngine;

import javax.script.ScriptEngineManager;

import javax.script.ScriptException;

import org.apache.commons.math3.util.FastMath;

import math.Vector2D;

import terrain.Position;

public class GARuleLearner {

 Invocable inv;

 Object myObject;

 String model;

 String inputExpe = "";

 String outputExpe = "";

 public GARuleLearner(int numOfInputs, int numOfActions) {

  try {

   ScriptEngineManager manager = new ScriptEngineManager();

   ScriptEngine engine = manager.getEngineByName("JavaScript");

   inv = (Invocable) engine;

   String scriptPath = "/home/isuru/MSC-
AI/Project/WorkSpace/for_simulation_only/source/ugv-controller-
ai/Client/src/MyDQN.js";

   engine.eval("load('" + scriptPath + "')");

   myObject = engine.get("myObject");
   inv.invokeMethod(myObject, "init", numOfInputs, numOfActions);
  } catch (ScriptException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  } catch (NoSuchMethodException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }
 public void test(double x, double y) {
  Vector2D toTarget = new Vector2D(0.0, 10.0);
  Vector2D toPos = new Vector2D(x, y);
  toTarget = toTarget.perpendicularClock();
  double angleToPos = FastMath.toDegrees(
    FastMath.atan2(toTarget.getY(), toTarget.getX()) - FastMath.atan2(toPos.getY(),
toPos.getX()));
  System.out.println(angleToPos);
 }
 public void train(Position currentPosition, Position futurePosition, Position
initialPosition,
   Position targetPosition, ArrayList<Position> neighborPositions,
ArrayList<Position> obstaclePositions) {
  try {
   Vector2D toTarget = new Vector2D(targetPosition.getX() - currentPosition.getX(),
     targetPosition.getY() - currentPosition.getY());
   toTarget = toTarget.perpendicularClock();
   double neighbors[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0 };
   double obstacles[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0 };
   double angleToPos;
   Vector2D toPos;

   for (Position pos : neighborPositions) {

    toPos = new Vector2D(pos.getX() - currentPosition.getX(), pos.getY() -
currentPosition.getY());

    angleToPos = FastMath.toDegrees(

      FastMath.atan2(toTarget.getY(), toTarget.getX()) - FastMath.atan2(toPos.getY(),
toPos.getX()));

    //System.out.println("neighbors index " + (int) Math.floor(angleToPos / 22.5));

    if (angleToPos < 0) {

     angleToPos = 360 + angleToPos;

    }

    neighbors[(int) Math.floor(angleToPos / 22.5)] = currentPosition.getDistance(pos);

   }

   for (Position pos : obstaclePositions) {

    toPos = new Vector2D(pos.getX() - currentPosition.getX(), pos.getY() -
currentPosition.getY());

    angleToPos = FastMath.toDegrees(

      FastMath.atan2(toTarget.getY(), toTarget.getX()) - FastMath.atan2(toPos.getY(),
toPos.getX()));

    //System.out.println("obstacles index " + (int) Math.floor(angleToPos / 22.5));

    if (angleToPos < 0) {

     angleToPos = 360 + angleToPos;

    }

    obstacles[(int) Math.floor(angleToPos / 22.5)] = currentPosition.getDistance(pos);

   }

   String input = "";

   for (int i = 0; i < 16; i++) {

    if (i == 0) {

     input = input + neighbors[i] + "," + obstacles[i];

    } else {

     input = input + "," + neighbors[i] + "," + obstacles[i];
    }
   }
   double result;
   angleToPos = FastMath.toDegrees(FastMath.atan2(toTarget.getY(),
toTarget.getX())
     - FastMath.atan2(futurePosition.getY(), futurePosition.getX()));
   double expected = Math.floor(angleToPos / 22.5);
   double reward = 0;
   for (int i = 0; i < 50; i++) {
    result = (double) inv.invokeMethod(myObject, "forward", input);
    System.out.println("result " + result + " expected " + expected);
    inputExpe = inputExpe + input + "-" + expected + "-" + result + "\n";
    if (result == expected)
    {
     reward = 1000 + 1000;
    }
    else
    {
     reward = 1000 / Math.abs(result - expected);
    }

    if(obstacles[(int) result] == 1 || neighbors[(int) result]== 1)
    {
     reward = 0;
    }
    inv.invokeMethod(myObject, "backward", reward);
   }
  } catch (NoSuchMethodException | ScriptException e) {
   // TODO Auto-generated catch block
   System.out.println("NoSuchMethodException | ScriptException e");
   e.printStackTrace();
  }
 }
 public void saveInput()
 {
  try {
   Files.write(Paths.get("/home/isuru/input.txt"), inputExpe.getBytes(),
StandardOpenOption.CREATE);
  } catch (IOException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }
 public void saveOutput()
 {
  try {
   Files.write(Paths.get("/home/isuru/output.txt"), outputExpe.getBytes(),
StandardOpenOption.CREATE);
  } catch (IOException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }
 }
 public double predict(Position currentPosition, Position futurePosition, Position
initialPosition,
   Position targetPosition, ArrayList<Position> neighborPositions,
ArrayList<Position> obstaclePositions) {
  try {
   inv.invokeMethod(myObject, "finish");
   Vector2D toTarget = new Vector2D(targetPosition.getX() - currentPosition.getX(),
     targetPosition.getY() - currentPosition.getY());
   toTarget = toTarget.perpendicularClock();

   double neighbors[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0 };

   double obstacles[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0 };

   double angleToPos;

   Vector2D toPos;

   for (Position pos : neighborPositions) {

    toPos = new Vector2D(pos.getX() - currentPosition.getX(), pos.getY() -
currentPosition.getY());

    angleToPos = FastMath.toDegrees(

      FastMath.atan2(toTarget.getY(), toTarget.getX()) - FastMath.atan2(toPos.getY(),
toPos.getX()));

    if (angleToPos < 0) {

     angleToPos = 360 + angleToPos;

    }

    neighbors[(int) Math.floor(angleToPos / 22.5)] = currentPosition.getDistance(pos);

   }

   for (Position pos : obstaclePositions) {

    toPos = new Vector2D(pos.getX() - currentPosition.getX(), pos.getY() -
currentPosition.getY());

    angleToPos = FastMath.toDegrees(

      FastMath.atan2(toTarget.getY(), toTarget.getX()) - FastMath.atan2(toPos.getY(),
toPos.getX()));

    if (angleToPos < 0) {

     angleToPos = 360 + angleToPos;

    }

    obstacles[(int) Math.floor(angleToPos / 22.5)] = currentPosition.getDistance(pos);

   }

   String input = "";

   for (int i = 0; i < 16; i++) {

    if (i == 0) {
     input = input + neighbors[i] + "," + obstacles[i];
    } else {
     input = input + "," + neighbors[i] + "," + obstacles[i];
    }
   }
   double result = (double) inv.invokeMethod(myObject, "predict", input);
   angleToPos = FastMath.toDegrees(FastMath.atan2(toTarget.getY(),
toTarget.getX())
     - FastMath.atan2(futurePosition.getY(), futurePosition.getX()));
   double expected = Math.floor(angleToPos / 22.5);
   outputExpe = outputExpe + input + "-" + expected + "-" + result + "\n";
   double retVal = -1;
   if(result == expected)
   {
    retVal = 0;
   }
   else
   {
    retVal = Math.abs(result - expected);
   }
   if(obstacles[(int) result] == 1 || neighbors[(int) result]== 1)
   {
    retVal = 1000;
   }
   return retVal;
  } catch (NoSuchMethodException | ScriptException e) {
   System.out.println("NoSuchMethodException | ScriptException e");
   e.printStackTrace();
   return -1;
  }
 }
 public void save() {
  try {
   inv.invokeMethod(myObject, "finish");
   model = (String) inv.invokeMethod(myObject, "savenet");
   Files.write(Paths.get("/home/isuru/model.txt"), model.getBytes(),
StandardOpenOption.CREATE);
  } catch (NoSuchMethodException | ScriptException e) {
   e.printStackTrace();
  } catch (IOException e) {
   e.printStackTrace();
  }
 }
 public void load() {
  try {
   model = new String(Files.readAllBytes(Paths.get("/home/isuru/model.txt")));
   inv.invokeMethod(myObject, "loadnet", model);
   inv.invokeMethod(myObject, "finish");
  } catch (NoSuchMethodException | ScriptException e) {
   e.printStackTrace();
  } catch (IOException e) {
   e.printStackTrace();
  }
 }
}
```