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

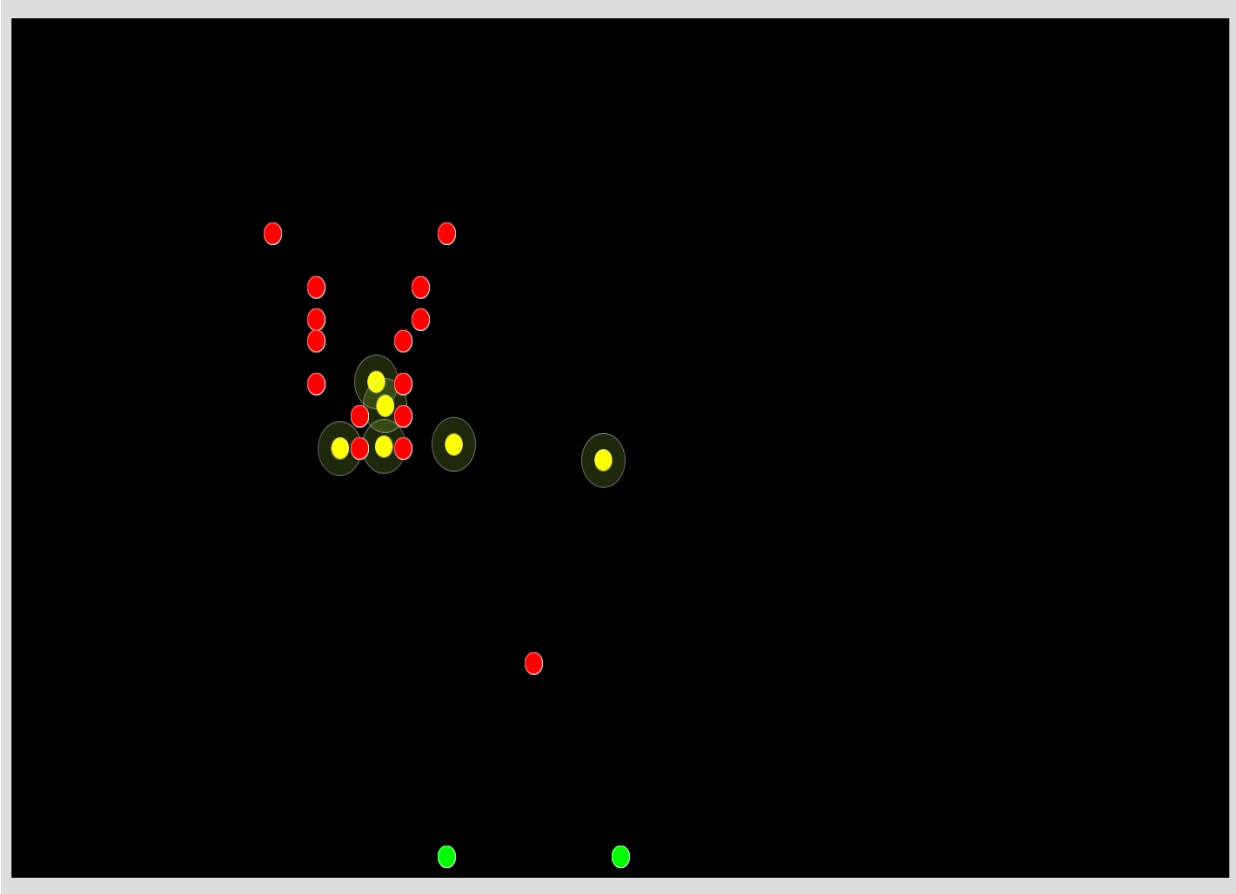

Figure A.1: Multiple agents are avoiding multiple obstacles

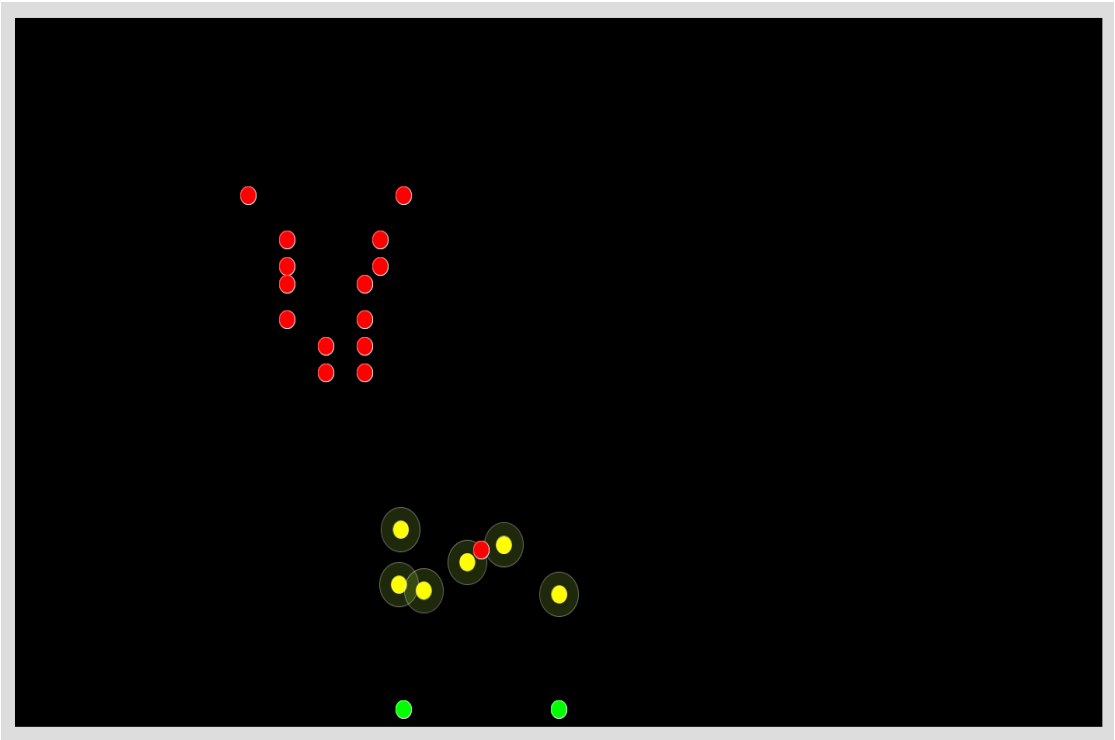

Figure A.2: Multiple agents are avoiding a single obstacle

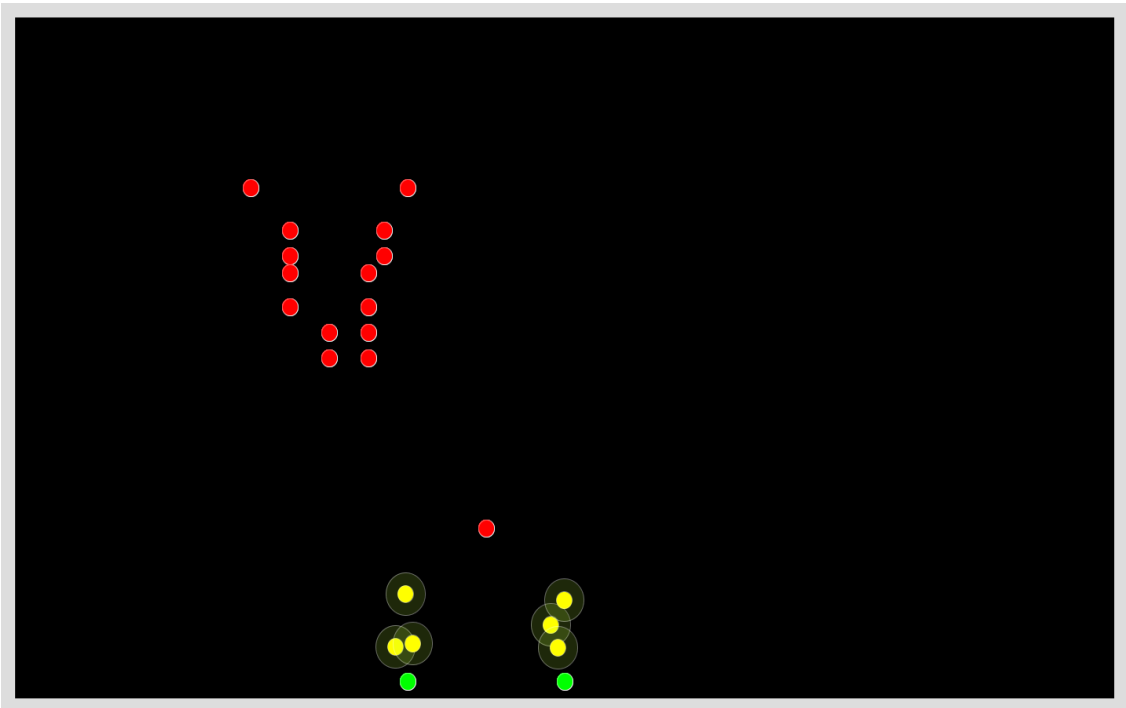

Figure A.3: Multiple agents are reaching to their targets as swarms

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();
  }
 }
}
```