UNIVERSITY OF CINCINNATI

Date: 9-Nov-2010

I, Mohit Ahuja ,

hereby submit this original work as part of the requirements for the degree of:

Master of Science

in Aerospace Engineering

It is entitled:

Fuzzy counter Ant Algorithm for Maze Problem

Student Signature: Mohit Ahuja

This work and its defense approved by:

Committee Chair: Kelly Cohen, PhD
Kelly Cohen, PhD

Bruce Walker, ScD
Bruce Walker, ScD

Manish Kumar, PhD
Manish Kumar, PhD

Fuzzy Counter Ant Algorithm for Maze Problem

A thesis submitted to the Graduate School of the
University of Cincinnati
in partial fulfillment of the requirements for the degree of
Masters of Science

In the Department of Aerospace Engineering & Engineering Mechanics
of College of Engineering & Applied Sciences

By

Mohit Ahuja

November 09, 2010

B.E.- Mechanical Engg. Thapar University, India

June 2008

Committee Chair: Kelly Cohen, Ph.D.

Abstract

AUTONOMOUS intelligent multi-agents have a widespread application in different fields including homeland security. Several applications of such autonomous robots include exploration, mine detection, border patrol, ISR missions etc. It is desired that the autonomous robot should have some adaptive and intelligent decision making capability for navigation in unknown obstacle rich terrains. Researchers in the field of AI and neural networks have considered the generalized maze problem an interesting test bed. Over the last couple of years, scientists and researchers have done a lot of work in this area. Using a cooperative multi agent robot system reduces the convergence time considerably as compared to using a single agent. Many algorithms, till date, have been proposed to develop a common decision support system among multi agents. However, most of the decision support systems are either time consuming or unscalable. Presented here is a fuzzy inference system used along with counter ant algorithm (modified ant colony optimization algorithm) that aids the agents in choosing paths along the unexplored regions, and hence, solves the maze problem in reduced time. MATLAB simulations results have been presented to demonstrate the effectiveness and scalability of the proposed algorithm. Application of the developed algorithm onto an actual mobile robot has also been shown to solve a lab based maze without the repetition of the already explored paths.

Acknowledgement

I would like to gratefully acknowledge the enthusiastic supervision of Dr. Kelly Cohen during this work. I thank Dr. Manish Kumar for the technical discussions on the maze problem and Elad Kivelevitch for the numerous technical discussions regarding counter ant model. I would also like to thank Baisravan HomChaudhuri for numerous stimulating discussions, help with the experimental setup and general advice. I would also like to acknowledge the help of Curtis Fox for his support with all types of technical problems - at all times.

I am grateful to all my friends from University of Cincinnati, for being the surrogate family during these two years and for their continued moral support there after.

Finally, I am forever indebted to my parents and Neena, my wife, for their understanding, endless patience and encouragement when it was most required.

Mohit Ahuja

Contents

1. Introduction.....	8
1.1 Motivation.....	8
1.2 Research Summary.....	15
2 Literature Review.....	19
3 Methodology.....	30
3.1 Carrot Or Stick Approach.....	34
3.2 Counter Ant Algorithm with Fuzzy Logic Approach.....	36
4 Problem Formulation and Scalability.....	40
5 Simulation Results & Discussions.....	50
5.1 Carrot or Stick Approach.....	50
5.2 Counter Ant Algorithm with Fuzzy Logic.....	57
5.3 Scalability	60
5.4 Figures of Merit.....	67
6 Experimental Validation.....	73
6.1 Experimental Setup.....	73
6.2 Khepera III Robot Dynamics.....	76
6.3 Swistrack Tracking Tool.....	79
6.4 Algorithm Flowchart.....	82
7 Conclusion and Recommendations.....	85

References

List of Figures

2.1	Follower using right hand rule along with Pledge Algorithm.....	21
3.1	Experimental Setup for Ant Algorithm with two paths.....	37
4.1	The maze problem.....	40
4.2	Simulation result of movement of Single agent using “Carrot or Stick” Approach.....	42
4.3	Simulation result showing one of the odd cases.....	44
4.4	Choices available for an agent at each move	46
4.5	Fuzzy Inference System.....	47
4.6	Input Membership Functions.....	48
5.1	Movement of one agent through cross check pattern using “Carrot or Stick” approach.....	51
5.2	Movement of one agent through horizontal pattern using “Carrot or Stick” approach.....	52
5.3	Movement of one agent through vertical pattern using “Carrot or Stick” approach.....	53
5.4	Demonstrations of Problematic Scenarios.....	55
5.5	Demonstration of Fuzzy Counter Ant Algorithm.....	58
5.6	Demonstration of final solution to maze problem.....	58
5.7	Algorithm solving a 20 X 20 maze using 4 agents.....	61
5.8	Algorithm solving a 30 X 30 maze using 6 agents.....	63

5.9	Algorithm Solving a 15 X 20 maze using 3 agents.....	64
5.10	Algorithm Solving a 15 X 25 maze using 5 agents.....	66
5.11	Plot of Number of steps and Maze sizes for two agents.....	68
5.12	Plot of Average Steps to Number of agents.....	69
5.13	Figure of Merit demonstrating the scalability of the algorithm.....	71
6.1	Actual Experimental Lab setup	74
6.2	Grouped Robot Sensors	77
6.3	Processing Pipeline for Maze Experiment	80
6.4	ID Ring used in Maze Experiment	81
6.5	Main Display window for Swistrack	81
6.6	Integration of Different Components in lab	82
6.7	Algorithm Flowchart for the lab model	83

Chapter -1

Introduction

1.1 Motivation

Developing intelligent multi-agent system is a problem that has captivated the scientific community for several years. Its evolution started in the decade of 1970 after which it became an established and promising research field that brings together concepts and ideas from many disciplines, including artificial intelligence, computer science, management science etc. An agent is a computational entity such as a software program or a robot that can be viewed as perceiving and acting upon its environment in an autonomous manner. As an intelligent entity, an agent operates flexibly and rationally in a variety of environmental circumstances given its perceptual and effectual equipment [1]. These Autonomous Agents have a widespread application in different fields including homeland security. According to different applications, multi-agents can have different contribution in different areas. Several applications of such autonomous agents include exploration, mine detection, border patrol, ISR missions etc.

One of the challenging issues in multi-agent system research is the cooperation, coordination, and negotiation among distributed agents in a multi-agent system [2]. A major focus of these agents is their ability to plan their path and execute the task assigned to them in a cooperative manner. The movement from one place to another from an initial to a final point, with only the information of where we want to reach at the end, is a complicated task [3]. It is desired that the autonomous robot should have some adaptive and intelligent decision making capability for navigation in unknown terrains. These agents, in particular, rely on their self organization phenomena in order to produce a complex collective behavior. The main difficulty comes from the fact that self-organization is a process that is not well understood and is not easy to control. Thus, the only solution for the designer is often to undergo a tedious task of tuning the parameters of the reactive behaviors to obtain the desired collective behavior [4]. Intelligent behavior of a robot may be achieved as a result of its capability to learn and adapt [20].

Exploration in an unknown environment in a cooperative manner is an important task for an autonomous multi-agent system. Exploration is the act of searching or traveling a terrain for the purpose of discovery. The problem deals with the use of an agent to maximize the knowledge over a particular area. The

exploration problem arises in mapping, and search and rescue situations, where an environment might be dangerous or inaccessible to humans. Apart from these obvious applications, such exploration problems are also inspired from daily human behavior. In normal daily life, we are capable of finding the optimal paths through the places, which we have not visited before, without bumping or colliding with obstacles. This adaptive human behavior generates curiosity among researchers and scientists to have robots which are capable to adapt and operate in unknown or changing environments with a higher degree of intelligence [5]. However, present obstacle avoidance algorithms have certain limitations. For example, knowledge of an environment to be navigated is often required to be known in advance. The designer is required to revise the previous knowledge of features or map so as to correspond to the new changes in the environment [6]. Such an approach is infeasible for most real industrial applications because the result of a robot colliding with its environment can be catastrophic for the robot itself, and damaging for the environment [5].

A maze can be described as a puzzle in the form of a complex environment through which the route has to be determined by the solver. The generalized maze i.e. maze which can be scaled randomly, has been considered an interesting test bed by various researchers in the field of Artificial Intelligence. In order to solve

the maze using a single robot, the robot should search the whole maze to find a solution. This is because the environment is unknown with no prior knowledge of the solution. However, Multi agent robot system can be used to solve the maze problem to increase efficiency. During the exploration, each robot should ideally search different locations so that the environment as a whole is mapped at a faster rate. Exploration of an agent in a region already being explored by another will result in overlap of the region's map and hence, decreased efficiency and long elapsed time. After an agent reaches the end point (the solution) of the maze, information exchange should enable the other agents to reach the maze exit in near minimum time.

Moreover, Multi-agent systems are sometimes easier to understand and easier to develop, especially when the problem being solved is itself distributed. In fact, Multi-agent systems are the best way to characterize or design distributed computing systems where information processing is ubiquitous. These agents could function as intelligent application programs, which would be knowledgeable about information resources that are local to them, and cooperate to provide global access and better management of the information [1]. For problems like generalized maze, where the environment is large for a global solution to be

formulated and implemented, such agents need to execute independently, and cooperate in an autonomous fashion so as to :

- Share Information & Expertise
- Work parallel on solving common problems
- Represent multiple viewpoints

For Example: an Unmanned Aerial Vehicle (UAV) is a powered aerial vehicle without a human operator that uses aerodynamic forces to provide lift and can fly autonomously carrying lethal or nonlethal payloads. These days the use of UAVs to improve border security is a technique that has garnered congressional attention. For a long time Congress has recognized Border Security as one of the top priorities. According to the CRS report for Congress, released in Jan 2004, the Northern border separating the mainland United States and Canada is 4,121 miles long, and consists of 430 official and unofficial ports of entry. The expansive nature and the possibility of entry through unpopulated regions make the border difficult to patrol. The southern border separating the United States and Mexico is 2,062 miles long and consists of thirty ports of entry and “innumerable unofficial crossings”. Despite of having larger number of border patrol agents, illegal border crossings and significant drug smuggling activities could not be checked frequently. Past difficulties in securing the borders in conjunction with fears that

terrorists could exploit existing security vulnerabilities by surreptitiously crossing the borders has prompted Congress to call on the Department of Homeland Security (DHS) to examine the potential use of UAVs [7].

Present UAVs, Drones, are intelligent enough to be considered as agents for the purpose of border patrolling. Like Maze Environment, border patrolling also represents an unknown dynamically changing environment. The diverse nature of threats ranging from terrorists to drug smugglers, arms dealers, and human traffickers has possessed a great challenge to the present U.S Border defense. In addition to border patrol agents the borders are also monitored and protected by video cameras, ground sensors and land vehicles. When these systems i.e. ground systems and UAVs, are interconnected so they can interact, then they become more than just a collection of processors – a distributed computing system – it becomes a multi-agent system. Coverage along the remote sections of the U.S. borders can be improved by utilizing the technical capabilities of UAVs. Electro-Optical (E-O) identification technology is advanced enough to identify a potentially hostile target of the size of a milk carton at an altitude of 60,000 feet. The ground control operators can utilize UAVs to receive a precise and real-time imagery, based on which, they can then disseminate that information so that informed decisions regarding the deployment of border patrol agents can be made quickly [7].

Much of traditional AI has been used to make these UAVs intelligent, but intelligent systems do not function in isolation – they are very small part of the environment in which they operate, and the environment typically contains other such intelligent systems. Thus it makes sense to view such systems in social contexts.

1.2 Research Summary

Exploration of an unknown terrain, like maze, by an autonomous agent is a very challenging task primarily because of the complexity and un-certainties involved in the task. Complexity arises due to the fact that a maze problem is an obstacle rich environment. No matter what the maze looks like, what we have to attribute to the explorer is an instinct to avoid the path it has already explored, and to share information with the other agent/explorer.

A typical multi-agent system which is solving a maze problem requires two separate modes. First, the agents need to explore the whole region for solution in the most optimized way. Second, after any agent finds the solution, efficient algorithm is required to make the phase transition of other agents from exploration to follow. The agents being used for exploration are desired to be autonomous in nature, and should be capable of taking decisions based on local information. Therefore, these agents are not employed with the models of their environment. However, sufficient information about the dynamic local environment has to be made available to these agents instantaneously. In other words, we can say that these agents have explicit knowledge about the environment and employ a mechanism for operation, based on which, they draw inferences from their knowledge. Apart from this, it is highly desired that these agents should also have

the ability to communicate in order to achieve the goals in an efficient manner. Communication can enable the agents to coordinate their actions and behavior, resulting in systems that are more coherent. The ability to communicate can be divided into two parts i.e. perception, the capability to receive the messages, and action, the capability to send the messages. The degree of coordination is the extent to which agents can avoid extraneous activity by reducing resource or avoiding deadlocks [1]. In order to cooperate successfully, each agent maintains a log of the previous movements of the other agents through a mechanism and develops a model for future interaction by updating its own movements for the log of other agents.

In this research, an effort has been made to solve the exploration problem in a maze using two different techniques i.e. carrot or stick approach and modified ant colony optimization algorithm (counter ant algorithm) with fuzzy logic modeling. The carrot or stick approach is an unsupervised learning concept offering a combination of rewards and punishment to induce intelligent behavior. This approach is motivated from old times when people use to train their animals by offering them carrots for right behavior and sticks for any notorious activity. In this research, the same concept has been used by awarding and punishing the robot, based on the results of its movements.

The ant colony optimization algorithm is a probabilistic technique for solving computational problems which can be reduced to finding potential paths through graphs. In the ant colony optimization technique, the probability of an ant to follow a particular path increases with the increase of deposited pheromone concentration (a chemical substance excreted by the ants). More pheromone concentration indicates that more ants have followed the path. In an exploration problem, it is desired not to explore a region that has already been explored by any other agent. A modified version of the ant colony optimization, Counter Ant Algorithm (CAA), deals with the robot's collaborative behavior based on repulsion than attraction to pheromone concentration i.e. the more the concentration of pheromone, the less the probability of choosing that particular path. The robot's reaction consists henceforth in avoiding paths more covered by this chemical substance. Fuzzy logic can be used in modeling this type of ant's behavior. This is because most of the animal's or insect's actions can be stated as a collection of If-Then rules expressed using natural language. Studies of this behavior often provide a verbal description of both field observation and interpretation.

The importance of developing a common decision support system among multi UAVs for the purpose of efficient border patrolling provides the major motivation for this research. For the purpose of lab research, we have equipped the

Khepera III mobile robots as agents. These robots have a high end embedded computing power feature along with multiple sensor arrays for both long range and short range object detection that can match the performance of much bigger robots. A distributed system among these mobile robots provides a framework for multiple sensors to search a target and localize its position precisely on the maze. Another contribution of this research is that it also develops an effective communication strategy among agents so that they can keep a track of other agent's movement. The incorporation of this property enables the agents to determine common tasks, avoid unnecessary collisions, and pool knowledge and evidence.

The thesis is organized as follows: First, a background survey on related works are shown followed by a brief discussion on carrot and stick approach, and counter ant algorithm. The problem formulation is discussed in the following chapter and the scalability of the approach is also investigated. Simulation results of both algorithms are discussed along with significant figures of merit. Finally, the experimental validation is shown in the following chapter followed by conclusions and future works.

Chapter 2

Literature Survey

Intelligent Multi-agent systems have been the subject of intensive research over the past few years. Researchers in the field of AI and neural network have performed a good amount of research in different areas of intelligent agent systems. There is a wide range of algorithms for navigation and tracking intelligent agents which can easily formulate the goal of finding the maze solution through cooperation and collaboration. In this chapter, a survey of such related works is presented.

As mentioned in Chapter 1, a typical maze problem requires two separate modes. First is to navigate the agents through maze so as to explore the whole region for solution, and second is to make a phase transition of other agents from exploration to follow. Some algorithms to solve a maze problem will include random walk [8], follow the wall algorithm [9], Pledge Algorithm, Tremaux's algorithm, etc. Random walk algorithm is one of those notions that come to every mind who think about navigation of an intelligent system. In random walk, the agent will have equally likely probability to move to any path sensed by it. This method is based on the theory of Markov Chains. Given a graph and a starting

point, this algorithm will select a neighbor randomly and move towards it. Random walks can be used to reach “obscure” parts of large sets but this method is very inefficient, and results in exploration of the same region over and over again [8].

For a simply connected maze, using “follow the wall” algorithm, often referred to as the *left hand rule* or *right hand rule*, the agent follows a path where the same wall is always on one of its sides, and it either finds the exit or returns to the start. The algorithm is based on Moore’s algorithm of finding the lowest cost path through a maze which has a cost assigned to each passageway. This method reduces large open areas containing many possible paths to a small set of shortest paths. This method works well when the beginning is the entry point of the maze but is not suitable when the starting position of the agent is any point within the maze [9]. In such cases the wall follower will continually go around its ring. “The pledge algorithm” (named after Jon Pledge of Exeter) is another algorithm designed to solve this problem. This algorithm requires an arbitrary chosen direction to go forward. When an obstacle is met, one hand is kept along with obstacle and the angles turned are counted. The solver leaves the obstacle, when it is facing the original direction again and the angular sum of the turns is 0, and continues moving in its original direction. This algorithm does not leave the rightmost wall due to the “sum of turns made” not being zero at that point. It

follows the wall all the way around, finally leaving it heading left outside and underneath the letter shape [24]. Figure 2.1 shows the follower using the right hand rule followed by the pledge algorithm to get to the solution of the maze. Though the pledge algorithm is used to find a solution out of the maze, but it does not help in exploring the maze. Also to use the pledge algorithm, an exploration algorithm is required that will be followed by pledge algorithm and a well defined mode change is required.

Figure 2.1 Follower using *right-hand rule* along with Pledge Algorithm

On the other hand, Tremaux's algorithm is an efficient method for finding the way out of a maze. It requires marking a path that it has already followed, and is guaranteed to work for all mazes with well defined passages. When the agent arrives at a marked junction, the algorithm enables the agent to take up an unmarked path if possible. If there is no unmarked passage then the agent should take a marked path, marking it again. The agent shouldn't pick a path marked twice. If there is no exit, this method will take the agent back to the start. Tremaux's algorithm implements a depth-first search very effectively [10]. This

method works well for single robot cases but is problematic in a multi-robot group. Navigation and obstacle avoidance tasks may also generally be solved by using artificial neural network based strategies [23]. Neural networks have learning properties which make them ideal candidates for multi-robot motion planning. A multi-layer perceptron neural network application with standard feed forward back propagation algorithm is proposed to predict the next direction in which the robot should move at the next step. Given the current position and target position of the robot as input, the neural networks are trained to determine the output node based on training sets obtained from sufficient number of experiments. But such networks suffer from the problem of local minima; also a global minimum based on the total error of the training data does not guarantee or imply that the generalization of the network will be better [5]. Such problems make the neural network unsuccessful for a class of problems representing “maze-like” features [17,22] , an argument also derived from Minsky et al [21].

Werbos and Pang [17] showed that this problem may be solved by a cellular simultaneous recurrent neural network (SRN) structure, of fixed architecture, repeated at each node. In this method the generalized maze problem can be simply specified, in matrix or graph notation. Each node in the mesh is like target node, a pathway node, or a barrier. Each edge has a cost associated with it -

infinite for a barrier, or one, otherwise. The J function of the critic is the expected total optimal cost-to-go from a specific node. Any currently occupied node may be considered the starting point. The weights, once learned may also be repeated at each node. Convergence to a correct solution is assured by designing these networks accordingly. However, in these applications, the architecture used is fixed, regardless of maze size or complexity. The price of this, of course, is the repetition of the entire network at each node of the maze [11] that will lead to a high convergence time.

Apart from working on the traditional multi-layer perceptron neural networks, researchers have also done a lot of work using other approaches in the field of artificial intelligence like learning algorithms, evolutionary algorithms etc. The evolutionary algorithms (EA) [25, 26] represent a stochastic search technique used to find approximate solutions to optimization and search problems. They use techniques inspired by evolutionary biology such as mutation, selection, and crossover. The EA typically works with a population of individuals representing abstract representations of feasible solutions. Each individual is assigned a fitness that is a measure of how good is the solution it represents. The better the solution, the higher the fitness value it gets. The population evolves toward better solutions. The evolution starts from a population of completely random individuals, and

iterates in generations. In each generation, the fitness of each individual is evaluated. Individuals are stochastically selected from the current population (based on their fitness), and modified by means of operator mutation and crossover to form a new population. The new population is then used in the next iteration of the algorithm [27]. Although, the rate of convergence of such algorithms is big which ensures their exploration capabilities, the lack of theoretical insight into such evolutionary algorithms prevents their use. Moreover, the Evolutionary Algorithms are more suitable for optimization problem, where as the maze problem is more concerned with search and exploration of the unknown environment rather than optimizing the steps or moves. Apart from this, for a problem like the maze, the computational complexity involved is also a prohibiting factor.

On the other hand, the Reinforcement Learning algorithm is based on dynamic programming, which has been studied for more than 50 years already. It has solid theoretical backgrounds built around Markov chains and several proved fundamental results [28]. The general model of agent-environment interaction is modeled through the notion of rewards. It is assumed in RL states that the agent is able to sense rewards coming from the environment. Agent's task is to take the action so as to maximize the rewards. Another assumption of RL algorithms is the Markov property, which states, that agent does not need the history of previous

states to make decision. The decision of the agent is based on the last state only. The Reinforcement learning algorithm has strong mathematical background but in real experiments, some of the assumptions are not realistic.

Moreover, learning in physically embedded robots is known to be a difficult problem, due to sensors and effectors uncertainty. The level of the difficulty increases due to the presence of other robots in the environment. In addition to traditional learning algorithms discussed above, research work is also being done in certain control based learning approach which has served as an effective methodology for multi-robot learning. Till date, there are four basic classes of control which can be summarized as following [31]:

- Reactive control: don't think, react.
- Deliberative control: think hard, and then act.
- Hybrid Control: think and act independently, in parallel.
- Behavior based control: think the way you act.

Reactive control tightly couples sensory inputs and effectors outputs, to allow the robot to quickly respond to changing and unstructured environments [34]. The reactive control is a powerful method largely used by animals. These systems make trade off in favor of fast reaction time against the complexity of reasoning. Reactive controls can demonstrate highly effective performance in case of a priori tasks [31]. However, its limitations include the robot's inability to have

memory, internal representations of the world, or the ability to learn over time [32].

In *Deliberative control*, the robot is provided with all the available sensory information and all the internally stored knowledge so as to think of a reason for taking an action. Reasoning is typically in the form of planning, requiring a search of possible state-action sequences and their outcomes, a computationally complex problem if the state space is large or the state is partially observable, both of which are physically situated, embodied systems [31]. This type of control is much suitable for multi-robot systems because in multi-agent or multi-robot systems, planning involves the ability to predict the actions of others, which in turn requires a complete knowledge of other's model as well. Since sufficient information (including the information about other models) and sufficient time is available, this method is highly effective for generating strategic actions. However, the environments of multi-robot systems are noisy and uncertain because of which the model maintenance becomes extremely difficult and computationally prohibitive for the use of purely deliberative approach to control.

Hybrid control approach makes use of the best aspects of reactive and deliberative controls, i.e. it attempts to combine the real time response of reactivity with the rationality and the optimality of deliberation. For this purpose, such control systems contains two different components, one for reactive and other for

deliberative ones, which interact with each other in order to produce a coherent output. However, this is a great challenge in itself, because the reactive component deals with the robot's immediate need, such as avoiding obstacles, and thus operates on a short timescale, and largely at the level of sensory signals. In contrast, the deliberative component uses highly abstract, symbolic, internal representations of the world, and operates on them on a longer time-scale [31]. A number of researchers have done a lot of work in designing such hybrid systems, few among them are Girat, Chatila, & Vaisset [35], Arkin [36], Conell [37], but till date, these controls are particularly popular for single robot control only.

Behavior-based control takes into consideration, the biological aspects for designing the control systems. As the name suggests, these controls are based on behaviors of the robot while it interacts with its environment. These control systems are constructed in bottoms-up fashion, starting with a set of survival behaviors, such as obstacle-avoidance, which couple sensory inputs with robot actions. Next, behaviors are added to provide more complex capabilities such as target chasing, exploration etc. This addition of behaviors continues till the desired capabilities of the robot are not obtained. Like the hybrid systems, these systems can also have different layers, and importantly, behavior-based systems can also store representations, but in a distributed fashion. Thus, if a robot needs to plan, it does so in a network of communicating behaviors, rather than a centralized

planner, and this representational difference carries with it significant computational and performance consequences [31]. Because of these properties, behavior based control often makes mistakes, repeats actions and can confuse the robot.

In a recent paper, Ahuja et al. [38] proposed a fuzzy inference system used along with counter ant algorithm (modified ant colony optimization algorithm) that aids the agents in choosing paths along the unexplored regions, and hence, solve the maze problem in a reduced time. Counter Ant Algorithm (CAA), deals with robot's collaborative behavior based on repulsion (rather than attraction) to pheromone concentration i.e. the more the concentration of pheromone, the less the probability of choosing that particular path. The robot's reaction consists henceforth in avoiding paths more covered by this chemical substance. Fuzzy logic can be used in modeling this type of ant's behavior. Rozen and Margaliot et al. [20] provide enough insight into fuzzy modeling to develop a mathematical model for the foraging behavior of the ants.

It is clear from this Chapter that considerable research has been performed in different directions, considering different aspects of artificial intelligence, for developing autonomous intelligent agents. As explained in earlier section, our research focuses on utilizing such intelligent agents for developing a multi-agent

decision support system for exploration and exploitation of an unknown obstacle rich environment represented by a 2D maze problem. Such research problem is an essential component for robot path planning according to best of our knowledge.

Chapter 3

Methodology

One of the major components, as discussed in first chapter, for the creation of an intelligent agent is its ability to plan its path. In a known or carefully engineered environment, it is possible to program the robot for all possible combinations of motions in order to accomplish specific tasks [29]. However, it is impossible to pre-program a robot for an experiment, like ours, where the environment is unknown, due to the fact that the number of motion combinations can be large or infinite. Moreover, it is highly desirable to have robots with higher degree of intelligence, so that they are capable to adapt and operate in an unknown environment. Such intelligence can only be achieved through learning algorithms [30].

As seen in previous chapter, researchers have done a lot of work in developing various learning algorithms for robot path planning. The classical goal of the machine learning algorithms is to develop mechanisms to recognize complex patterns, and to make intelligent decisions based on data which would lead to autonomous intelligent agents and optimize system performance over its lifetime.

In case of multi-robot systems, where the robot faces an uncertain and changing environment, the aim is to improve the efficiency over a shorter time span rather than attaining the optimality in asymptotic fashion [31]. Machine learning algorithms can be organized into following categories, based on the desired outcome of the algorithm

- Supervised Learning Algorithms
- Unsupervised Learning Algorithms
- Reinforcement Learning Algorithms.

Supervised Learning Algorithm is good for classification problems where the goal is to get the agent a classification system being created by the designer. A common example of such classification learning is Digit recognition. It is basically a machine learning technique for deducing a function from training data. The training data consists of pairs of input objects and their corresponding outputs. The task of supervised learning algorithm is to predict the value of the function for any valid input object having seen a number of training examples [39]. Supervised learning is a common technique for training neural networks and decision trees. However, these algorithms are highly dependent on pre-determined classifications for the information. Since not all the training sets have inputs classified correctly,

problems may arise if the algorithm used is powerful enough to memorize even those cases that do not fit the more general principles [40].

Unsupervised Learning Algorithms lay emphasis on determining the organization of data. These algorithms are harder as compared to the supervised learning algorithms since the goal here is to let the agent learn how to do something without telling it so. There are two approaches to unsupervised learning. The first approach is to teach the agent by using some sort of reward system to indicate success instead of giving any explicit categorizations. This type of training will fit well for those decision problems where the goal is to maximize the rewards rather than to produce a classification. A second type of approach is called clustering, where the goal is not to maximize a utility function but simply to find similarities in the training data. This is a data driven approach that can work well when there is sufficient data. This approach often assumes that the clusters discovered will be reasonably well with an intuitive classification [40].

Reinforcement Learning Algorithm is a type of machine learning that allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behavior. Reinforcement learning algorithms can easily map a large number of AI problems to a decision process. However,

there are many challenges in current Reinforcement Learning algorithms. Firstly, the problems can be pretty complex because of reason that the reinforcement algorithms are often too memory expensive to store values of each state. Solving such complexity might involve value approximation techniques, such as Decision Trees or Neural Networks. There are many consequences of introducing these imperfect value estimations, and research tries to minimize their impact on the quality of the solution. Secondly, due to limited perception, it is often impossible to fully determine the current state. This also affects the performance of the algorithm, and much work has been done to compensate this Perceptual Aliasing [41].

To summarize the whole discussion, learning algorithms can be valuable, and the selection of the algorithm depends on the type of problem and the time allotted for solving it. However, unsupervised learning has produced many successful results as compared to the other learning algorithms. It can be a powerful technique when there is an easy way to assign actions. Moreover, the maze problem can be defined more likely as a clustering problem rather than a classification one. In this chapter, two such approaches of the unsupervised learning are discussed.

3.1 Carrot or Stick Approach

Carrot or Stick Approach is a form of reinforcement learning that can be used for unsupervised learning, where the agent bases its actions on the previous rewards and punishments. The carrot or stick is an idiom that refers to a practice of giving rewards and punishment to induce the desired behavior. In old days, the carrot or stick approach was first used by owners of animals in order to keep them moving. Whenever the animal did something they wanted it to do, then, the animal was rewarded with a carrot but if it failed, the owners punished it by beating it with stick. The “Carrot or Stick” method works so well because it takes advantage of two of the most basic emotions -‘desire’ and ‘fear’. To be more explicit, ‘carrot’ scores a direct hit on ‘desire’ and ‘stick’ does the same on ‘fear’. It can be argued that if we use the terms ‘desire’ and ‘fear’ in a broad sense, most of the human behavior can be ‘modeled’ in terms of these two (and the human responses to them).

In fact, the carrot or stick approach is one of the motivation strategies, commonly used by managers and employers in industry. If we consider motivation as a 'state of mind', 'carrot and stick' can't directly cause motivation to occur i.e. it can create a situation where motivation is likely to be 'triggered'. In scientific terms, this approach motivates agent to learn its behavior based on feedback from

the environment. It generalizes the problem to the real world, where the agent might be rewarded for doing certain actions and punished for doing others. All other information is unnecessary because by learning a reward function, the agent simply knows what to do without any processing as it knows the exact reward it expects to achieve for each action it could take. Such an approach is very efficient for robot navigation where collision avoidance can be learnt by negative feedback from bumping into obstacles.

The Carrot or Stick approach takes less time for designing a solution, since there is no need to hand-craft complex sets of rules. Moreover, this can be extremely beneficial in the cases where calculating every possibility is very time consuming, or in the cases where learning has to be done by trial and error [40, 41].

3.2 Counter Ant Algorithm with Fuzzy Logic Approach

Counter ant algorithm is a probabilistic technique for solving computational problems which can be reduced to finding potential paths through graphs. The counter ant algorithm is an opposite technique developed by reversing the concept of well-known ant colony optimization algorithm, which is a part of swarm intelligence methods. In this algorithm ant (initially) wander randomly, and upon finding food, return to their colony while laying down *pheromone* trails. Following ants are able to smell this trail and tend to choose the path that has pheromone concentration. This property of ants can be used to simulate a system on computer, and to provide predictions for future behavior. Furthermore, for multi foragers, such as ants, it is also important as it helps them in reaching a consensus when faced with a choice of paths.

To explain this method we consider an experiment, designed by Goss et al. [13], where ants have two possible and unequal paths to their food as shown in figure 3.1. Ants must choose one of the paths to move from their nest to the food and come back through the same path. After making the choice, ants mark the chosen path by laying an evaporative substance called *pheromone*. Ants that take the shorter path return sooner as compared to those using the longer path. Thus eventually with passage of time the shorter path receives more pheromone

deposition than the longer path. This difference in the pheromone concentrations is amplified by the positive feedback process [14]. The process gets repeated until nearly all the ants take the same branch, neglecting the other one. Thus, the ants do not communicate directly. Rather, they affect the behavior of other ants by laying down pheromone trails. In addition, the evaporation property of the pheromone also plays an important role, as it enables the ants to deal with dynamic situations i.e. with the evaporation of the deposited pheromone, the obsolete paths, that lead to depleted food sources, get disappeared thus increasing the chances of detecting new and more relevant paths.

Figure 3.1 Experimental Setup for Ant Algorithm with two paths

The reason for choosing this approach is that it is easier to transform these verbal descriptions into mathematical models. Fuzzy logic is the most suitable tool for

studying such behaviors. This is due to the fact that many of the animal actions are fuzzy i.e. they can easily be categorized as a verbal description of both field observations and interpretations. We then apply fuzzy modeling to transform these verbal descriptions in the rules using suitable membership functions, and interfacing the rule base, to develop a well defined mathematical model. For an experiment suggested by Goss et al. [13] and Deneuborg et al. [14], given two paths, the function for choosing one path over another can be described as :

$$P_{n,k}(L,R) = \frac{(k+L)^n}{(k+L)^n + (k+R)^n}$$

Where, $P_{n,k}(L,R)$ is the probability of choosing the left branch instead of the right and “ n ” and “ k ” are the degree of non linearity and the degree of attraction attributed to an unmarked branch respectively. L and R represent the concentration of pheromone in left and right branch respectively.

For our maze problem, this concept can produce overwhelming success because many of the core problems that we face during the research can be easily tackled by mimicking the sophisticated mechanisms developed by various biological agents. The core idea is that over the course of evolution, living systems can develop efficient and robust solutions to various problems. Some of these problems are same that we encounter at engineering applications [14]. Thus, the

study of such biological behaviors can greatly benefit the researchers to model these verbal descriptions for the purpose of dealing with complex problems.

The next chapter will present a strategy to formulate the problem according to both of the above discussed approaches.

Chapter 4

Problem Formulation and Scalability

The task is defined as follows: Given a maze of the type shown in Fig. 4.1(which may vary in size, number of dimensions or the configuration of the obstacles), find an appropriate algorithm which provides a common decision support system between multi-agents, and moves agents from their initial position (start point) to a target position (end point), while minimizing the convergence time, which the robot takes to find the solution.

Fig. 4.1 The maze problem: Find an appropriate path that will lead the agent 1 and 2 from their present positions to end position.

The problem has been formulated in two different stages. Preliminary work on the problem is done by simulating the movement of one agent through the maze using

a technique which was similar to the “Carrot or Stick” approach of reinforcement learning. As discussed in Chapter 3, this technique can be explained as assigning rewards and punishment for the outcome of a task. This approach has been used by various researchers for developing algorithms in the field of evolutionary robotics [12]. A similar approach was used for the maze problem where the reinforcement mechanism is based on the numerals “1 or -1”. The area of maze is being divided into a number of equal square cells with each cell having an initial value “0”. The number of these cells depends on the size of maze. Whenever the robot makes an advancement for exploration, the value of each cell being advanced is replaced with the number 1, whereas if the same move leads to a blocked path, the numeral value of the all the cells, which are the part of that path, are replaced by the number -1. These values are stored in the form of a matrix having size equal to the number of cells. This matrix is called Agent Value Matrix, and it acts as a memory for the further exploration of the maze in a way that if the robot or agent happens to come to the same cell again, then the value -1 would prevent the robot repeating the same “mistake” again, and hence reduces the convergence time.

Figure 4.2(a) shows the simulation result for a single agent exploring the maze. As the agent starts moving from cell to cell, the value of matrix gets updated with the value of corresponding cell in the maze so that at every junction, where the agent has to take a decision among the directions available, it can consult

values of immediate cells in those directions. Preference is always given to direction having cell value '0'. If any of the direction doesn't have zero value, then the agent moves in the direction where the value of cell is '1', which denotes that the way ahead is blocked and updates the values of those cells as '-1'. Here the value '-1' denotes the punishment so that if the agent happens to come to same cell again it should not repeat advancing to same direction again. Fig 4.2 (b) shows the final agent value matrix representing the values of each cell of the same maze. It can be easily seen that matrix cells having value '1' in figure 2 (b) denotes the solution of the maze from the initial point of agent 1 to the final point.

Figure 4.2 (a) Simulation result of movement of Single agent using “Carrot or Stick” Approach

1	1	0	0	0	0	0	0	0	0
-1	1	1	0	0	0	0	0	0	0
-1	-1	1	1	1	0	0	0	0	0
-1	-1	-1	-1	1	0	0	0	0	0
-1	-1	-1	-1	1	0	0	0	0	0
-1	-1	-1	-1	1	0	1	1	0	0
-1	-1	-1	-1	1	1	1	1	0	0
-1	-1	-1	-1	-1	0	0	1	0	0
-1	-1	-1	-1	-1	0	0	1	1	0
-1	-1	-1	-1	-1	-1	0	-1	1	1

Figure 4.2(b) Agent Value matrix.

Once the result for a single agent has been obtained, next stage was to scale this concept to two agents so as to develop a common decision support system between them, and also to reduce the convergence time. Same concept was scaled by adding another agent in the matrix so that both agents can explore the maze in the most optimized manner by avoiding the repetition of the already explored path. But there arose some cases when both the agents happened to come across each other. In such cases the agents were unable to find the immediate cells having cell value '0' because of which they had to choose the cells having value '1' in either directions, and hence, the value of those cells were updated to '- 1', therefore, closing that path for future advances as well. Fig 4.3 shows one of such cases. In this figure, it can be seen that even if the agent '1' or agent 'Red' goes straight, due to the cell value '1', the '2' agent or the 'blue' agent will have to choose a path having cell value '1', thus updating the value of that particular cell to '-1'. In any

of the cases, one of the two agents will get confined in that particular region only and hence, will not be able to explore the maze further.

Figure 4.3(a) Simulation result showing one of the odd cases

1	1	1	1	1	1	1	0	2	2
1	1	-1	0	0	0	1	2	2	2
1	1	0	0	0	0	1	2	-2	-2
-1	0	0	0	0	0	0	-2	-2	-2
-1	-1	0	0	0	0	0	-2	-2	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	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 4.3(b). Agent Value Matrix of the above maze

Thus, for the purpose of developing an effective decision support system, we need a better, well organized and more structured algorithm that could overcome the problems faced in initial algorithm. Therefore, we decided to incorporate a hybrid algorithm which has the key points of old approach and the “Counter Ant Algorithm” [18]. As Discussed in Chapter 3, ant colonies, and more generally social insect societies, are distributed systems that in spite of the simplicity of their individuals, present a highly structured social organization. For our exploration strategy, we use the Counter Ant Algorithm where the agent chooses its path according to the less pheromone concentration i.e. probability of choosing a path with lesser pheromone concentration is higher than the other paths. This model enables the agents not to explore a region that has already been explored by some other agent and thus, reduces the convergence time.

For solving the maze problem, we use fuzzy logic technique to mathematically model the above discussed hybrid algorithm. Similar to the “Carrots or Stick” approach, the pheromone concentration has been replaced by numeral value ‘1’ i.e. as the agent moves from one cell of maze to another, instead of depositing a chemical, it will update the value of cell with value ‘1’. However, unlike the old approach, the concept of ‘-1’ is being replaced by numeral ‘3’ which depicts the action of ants depositing more pheromone on the blocked path so as to avoid that path in future. Preference will now be given to the path having less pheromone

concentration. Now, similar to the mathematical model discussed in previous chapter, given two branches with pheromone concentration L and R, the input to the fuzzy inference system (FIS) is the difference between the pheromone concentrations of the two branches,

$$D \text{ (the input)} = L - R$$

The output has been modeled in the form of “IF – THEN” rules as:

- If D is positive (i.e. concentration L is higher than R) Then $P(L,R) = 0$
- If D is negative (i.e. concentration R is higher than L) Then $P(L,R) = 1$

Here, $P(L, R)$ is the probability of choosing “L” (left) over “R” (right).

In a maze, at most, there can be four choices for an agent to move as shown in the fig 4.4.

Figure 4.4 Choices available for an agent after each move

As a result, we have six FIS (Fuzzy Inference System) functions i.e. FIS₁₂, FIS₁₃, FIS₁₄, FIS₂₃, FIS₂₄ and FIS₃₄. (Where subscript 12 etc. denotes Fuzzy Inference System for choosing path 1 or 2). All the FIS are similar, and have one input i.e. the difference of concentration between the two paths, and one output, the probability of choosing a path over the other. As an example, for FIS₁₂, the input is the difference in immediate cell values for paths 1 and 2. The output will be the probability of choosing 1 over 2. Since the input can be negative or positive, the membership functions should monotonically increase from zero and reach the saturation level. For our case we use sigmoid functions. The output used is Sugeno i.e. either one or zero. Below shown are the FIS and the membership functions.

Figure 4.5 Fuzzy Inference System (MATLAB – Fuzzy Logic Toolbox)

Figure 4.6 Input membership functions

According to the model formed above, choosing a path from the available choices can be decided as:

Path 1 = mean $[P_{12}, P_{13}$ and $P_{14}]$

Path 2 = mean $[(1 - P_{12}), P_{23}$ and $P_{24}]$

Path 3 = mean $[(1-P_{13}), (1-P_{23})$ and $P_{34}]$

Path 4 = mean $[(1-P_{14}), (1-P_{24})$ and $(1-P_{34})]$

Output =Max [Path 1, Path 2, Path 3, Path 4]

Stagnation condition is a common case that arises when all the paths, surrounding the agent, have already been explored by other agents. The pheromone concentration in all those paths will be high and equal. Such conditions can be

avoided by utilizing the property of pheromone evaporation, because of which the concentration of pheromone on all paths will decrease with each time step. This property, as discussed in previous chapter, would also help us in dealing with dynamic situations like changing environment or in case when the number of agents is much higher than the optimal number required for solving a particular maze. In such cases, the agents would come across the same explored path repeatedly till one of them finds a solution. Thus, it is highly desirable to have such provisions where agent can make choices among the available paths, despite knowing that they have already been explored. The pheromone evaporation model, used in the algorithm, can be modeled as,

$$\text{Pheromone Concentration (t+1)} = \text{Pheromone Concentration (t)} - s * (\text{Time step})$$

Where, parameter “t” is the time step and parameter “s” determines the pheromone evaporation rate which depends upon the size of the maze. Once the problem has been formulated correctly, the next step is to check the scalability and robustness of the solution.

Thus, in next chapter we will discuss in detail about the simulation results, obtained from the formulated algorithm, and the scalability and robustness of the solution by running the developed algorithm for different maze sizes and for different number of agents.

Chapter 5

Simulation Results & Discussions

In previous two Chapters, the problem to be solved and the two approaches used to solve it have been discussed. This chapter presents the simulation result of both the approaches mentioned earlier.

5.1 Carrot or Stick Approach

As mentioned in Chapter 3, the first approach we used is the carrot or stick approach, in which learning by the agent is done by assigning rewards and punishment for the outcome of its actions. This is just a preliminary approach showing how an agent can be used for solving a maze autonomously. In this strategy, an award, i.e. “1”, is being rewarded for choosing a right path where as choosing a wrong path results in a punishment i.e. “-1”. For the purpose of analyzing the robustness of the solution, the algorithm is run for three different types of maze patterns i.e. cross check pattern, horizontal pattern and vertical pattern. Figures 5.1(a), 5.2(a) and 5.3(a) below respectively show how a single agent is capable of solving these different types of mazes using carrot or stick approach. Figures 5.1(b), 5.2(b) and 5.3(b) below show the corresponding agent value matrices demonstrating the end values of all the maze cells.

Figure 5.1 (a) Movement of one agent through cross check pattern maze using “Carrot or Stick” approach

1	1	1	0	0	0	0	0
-1	-1	1	1	1	0	0	0
-1	-1	-1	-1	1	0	0	0
-1	-1	-1	0	1	1	1	0
-1	-1	-1	-1	-1	0	1	0
-1	-1	-1	-1	-1	-1	1	0
-1	-1	-1	-1	-1	-1	1	0
-1	-1	-1	-1	-1	-1	1	1

Fig 5.1 (b) Agent Value Matrix for Cross check pattern maze.

Figure 5.2 (a) Movement of one agent through horizontal pattern maze using “Carrot or Stick” approach

1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	-1	-1	-1	-1	0
1	0	0	0	-1	-1	-1	-1
1	1	0	0	0	0	-1	-1
-1	1	1	0	0	0	0	0
0	1	1	-1	-1	0	0	0
0	1	1	1	1	1	1	1

Fig 5.2 (b) Agent Value Matrix for horizontal pattern maze.

The above figures were the results of the preliminary approach, and as discussed in previous chapter, when this approach was scaled to two agents, there arose some cases in which the preliminary approach failed to give the desired results and lead to other problems. The main three problems that came into picture due to those instances were collision, direction decision and optimization of the steps. We have already discussed the collision problem in last chapter. Collision of two agents happens when both the agents come across each other leading to confinement of either of the agent in a particular area of the maze, thus making it unable to solve the maze further. Direction decision is another problem which arose due to the inability of the preliminary approach to make the agent autonomous enough so that they can decide the right direction based on the information available from the other agent. Figure 5.4 below demonstrates the problematic scenario caused due to wrong direction decision taken by the algorithm, based on the carrot or stick approach. From the figure, it is clear that the blue agent should have returned back from the direction decision point considering the fact that the red agent is already exploring that particular area of the maze. But instead it took a wrong decision and passed the point, thus making the particular cell value “-1”. Because of this decision both the agents are now confined into that area of maze, thus resulting into an infinite loop of algorithm, and hence, no solution for the maze.

Figure 5.4 Demonstrations of Problematic Scenarios

Another problem that came into picture due to the algorithm inefficiency is the moves optimization i.e. once either of the agents is able to find the solution, the other agent is required to exit the exploration phase and follow the information shared by the first agent, so as to minimize the number of steps involved in finding the solution by the two agents. In the present algorithm, the agents are unable to share the information regarding their position history, and thus, it becomes impossible for them to optimize the number of moves.

As discussed in the last chapter, considering the above problems that arose due to carrot or stick approach, we decided to incorporate another hybrid approach to form an efficient algorithm that could repeatedly give a robust solution for any kind of maze problem. The next section of this chapter, thus, describes the simulation results obtained from the hybrid approach.

5.2 Counter Ant Algorithm with Fuzzy Logic

Final simulation results using the hybrid approach have been shown in the figures 5.5 & 5.6. Figure 5.5 gives a demonstration of the effectiveness of the decision making algorithm developed in the previous chapter. At the decision point, marked in the maze below, the 'Blue' agent had two ways to go i.e. it could have followed the 'Red' agent but as per the algorithm developed, it decided to avoid the already explored path and chose to explore the other path so as to minimize the convergence time. Now considering the stagnation case when the 'Red' agent would come back to the marked decision point, due to pheromone evaporation process, the cell value on marked point will be less as compared to the present cell of the agent due to which probability of choosing the marked cell will be more as compared to going back. Figure 5.6(a) provides the demonstration of the final simulated solution to the desired maze problem. In the figure 5.6(b), the red line corresponding to the numeral values less than '1', indicates the solution from initial position of agent 'Red' to the final exit point. On the other hand, the blue line, corresponding to numeral values less than '1', indicates the solution for agent 'Blue' from its initial position to the final position.

Figure 5.5 .Demonstration of Fuzzy Counter Ant Algorithm

Figure 5.6(a) Demonstration of final solution to maze problem

0.9737	2.9236	0	2.9821	2.9861	0.9962	0.9965	0.9968	0.9737	0.9737
0.9737	0.9741	0	0.9936	0.9950	0.9959	0	2.9909	0.9738	0
2.9222	0.9747	0.9911	0.9933	0.9955	0.9957	0	0	0.9740	0.9743
2.9219	2.9727	0.9915	0.9923	0	0	0	0	2.9223	0.9743
2.9217	2.9709	2.9754	2.9297	0	0	0	0	0.9747	0.9744
2.9678	2.9687	2.9525	2.9300	0	2.9336	0	0	0.9748	2.9236
2.9665	2.9640	2.9517	2.9497	2.9474	2.9470	0	0	0.9749	2.9254
2.9573	2.9635	2.9277	2.9316	2.9329	2.9466	0	0	0.9750	0.9750
2.9568	2.9623	2.9273	2.9312	2.9453	2.9463	2.9370	2.9388	0	0.9753
2.9553	2.9618	2.9270	2.9309	2.9440	2.9420	2.9408	2.9405	2.9401	0.9753

Figure 5.6(b) Final Agent Value Matrix

Thus from the above figures, it is clear that the problems discussed in last section i.e. collision and direction decision, can be avoided by incorporating this hybrid approach. But a solution is termed as stable, when it can solve all the related problems irrespective of the maze size or the number of agents. The next section would thus analyze the scalability of the solution.

5.3 Scalability

Scalability of a solution is the property which indicates its capability to handle changing amount of work in a graceful manner. In simple words, an algorithm is said to be scalable if it is suitably efficient and consistent for large situations. The generalized maze problem, being discussed in the last few chapters, has a tendency to grow in terms of maze size or the number of agents. This scalability property, thus, provides an economical means of expanding the existing algorithm to meet the future changes, without interrupting its functionality and response speed.

The scalability analysis of the above developed hybrid algorithm is necessary for determining the stability and robustness of the solution. In the case of maze problem, a solution would be stable if it is capable of solving all the similar problems irrespective of maze size, maze type or the number of agents. For this reason, an initial test was conducted in which the developed algorithm was run for different mazes with varying number of agents. Figures 5.7, 5.8, 5.9 and 5.10 below show the simulation results for the algorithm respectively solving a 20 x 20 maze using 4 agents, a 30 X 30 maze using 6 agents, a 15 X 20 maze using 3 agents and a 15 X 25 maze using 5 agents.

The scalability of the solution can be observed from two different perspectives i.e. scalability in terms of maze size and scalability in terms of group

size. It can be seen from figures below that all the groups are capable of finding the solution for any maze size.

(a)

(b)

(c)

(d)

Figure 5.7 Algorithm solving a 20 x 20 maze using 4 agents

(a)

(b)

(b)

(d)

(e)

Figure 5.8 Algorithm solving a 30 x 30 maze using 6 agents

(a)

(b)

(C)

(d)

(e)

Figure 5.9 Algorithm solving a 15 x 20 maze using 3 agents

(a)

(b)

(c)

(d)

Figure 5.10 Algorithm solving a 15 x 25 maze using 5 agents

Thus, from the above figures we can successfully conclude that the developed algorithm can be scaled for any maze size and for any number of agents. However, it is very important to determine the various performance characteristics of the developed algorithm. The next section of this chapter would now discuss about various performance trends by plotting different figures of merit.

5.4 Figures of Merit

Figures of Merit are a means to characterize the performance of a solution or method. They can be used to evaluate a potential method or to optimize a method that is already in use. In order to analyze the successful behavior for a cooperative decision support system between multi-agents, the algorithm being developed was tested by means of plotting different figures of merit. It is interesting to see the performance of the developed algorithm for different maze sizes and different number of agents in a group. Ordered pairs were formed by running each group of agents for each maze size. Since these results have been obtained using mat lab simulation, considering the fact that the simulation generates a different maze pattern every time we run the algorithm, therefore for each pair the algorithm was repeated for 250 times. The total number of moves taken by the pioneer agent to find the solution was counted for each run, and the mean of those iterations was taken as an output number. Such results for the ordered pairs were then plotted against number of agents for each maze size.

In the first figure of merit, the algorithm was simulated for different maze sizes, keeping the number of agents constant at two. Fig. 5.11 shows the final plot of number of moves and maze sizes for two agents. It can be easily observed from

the plot that as the maze size increases from 5x5 to 50x50, the total number of moves to find the convergence increase sharply.

Figure 5.11 Plot of Number of steps and Maze sizes for two agents

In second Figure of Merit, the algorithm is being repeated for different maze sizes and increasing number of agents. The number of moves involved in finding the solution for the pioneer agent are counted. The mean of the 250 iterations is then plotted against the number of agents for each maze size. Figure 5.12 below provides the insight of the algorithm's performance by providing the trend for energy savings with increase in the number of agents for a particular maze size.

Figure 5.12 Plot Demonstrating the energy savings with increasing number of agents

In this figure of merit, the simulation is being run for ten group sizes varying from a single agent to ten agents, and the maze size varying for 15x15 to 50x 50. It is interesting to see from the figure that, as the number of agents increases for every particular maze size, number of moves made by a single agent to find the solution decrease sharply i.e. with the increase in number of agents in a group, the contribution of an agent decreases sharply, thus saving an enormous amount of energy for each agent.

Last Figure of Merit gives us an insight into the scalability of the algorithm by plotting the number of moves against group sizes for different maze sizes, where the number of agents in a group ranges from 2 to 10, and the maze sizes varies from 8 x 8 to 40 x 40. Looking at these plots, we can easily derive that for each maze size the total number of steps decreases sharply with increase in number of agents, which reaches a minimum value, and then, increases with an increase in number of agents.

Figure 5.13 Figure of Merit demonstrating the scalability of the algorithm

Thus, adding an agent to a group increases its efficiency to explore the maze, by enabling it to explore the larger area of the maze and decreasing the convergence time for the pioneer agent. However, for a particular maze size as the number of agents increases, the total number of moves involved in getting them all to the solution also increases because more the number of agents, greater will be the repetition of the already explored paths in order to avoid collisions, and also because of the direction decisions of each agent.

It is also interesting to see that the graphs in figure 5.13 gives an insight into the trends developed by increasing number of agents for each maze size. Note that for each maze size, the number of moves decreases significantly to a minimum value, and then increases or becomes consistent. This minimum value defines the

preferred number of agents for that particular maze size. As seen from the figure, for a 8 x 8 maze the minimum steps are given by a group of 5 agents, whereas for a 10 x 10 maze, although the group having 7 agents gives the minimum number of steps but since the difference between the number of steps for 6 agents and those for 7 agents is not that much significant, therefore, instead of going for 7 agents, the solution can be reached by 6 agents. Furthermore, for 15 x 15 and 20 x 20 maze, this number increases to 7 agents whereas for 30 x 30 and 40 x 40 maze size, the solution is given by 8 agents.

Thus, the above figure of merit successfully defines the performance and the stability of the developed algorithm along with its capability to handle any maze size and any number of agents. It would now be interesting to test the algorithm for actual real world conditions, which always differ from the simulation condition. The next chapter would, therefore, discuss the implementation of this algorithm to a lab setup where mobile robots will try to solve an actual maze.

Chapter 6

Experimental Validation

6.1 Experimental Setup

The fuzzy ant maze exploration develops a decision support system among multi-agents so as to make them efficient for exploring an unknown terrain in an unsupervised way. In the last Chapter, we have discussed the results obtained by simulating the developed algorithm. The current chapter would now discuss the implementation of the above developed algorithm to the mobile robots, subjected to an actual dynamic maze environment in the lab. While designing this setup, major consideration was given to the fact that the lab experimental conditions should match the simulation conditions so that the results obtained by plotting various figures of merit should remain consistent.

In an actual experimental setup, the Khepera III mobile robots are being used as agents to explore a generalize maze test bed. The maze being setup in the lab is constructed with the help of movable plastic walls which can change its pattern so as to match the simulation conditions, where the maze pattern changes in a random order. Similar to the simulation conditions, lab maze is also divided into a number

of 121 square cells, having 11 rows and 11 columns, and each cell has a side of 180 cm i.e. size of each movable wall. The robots being used in the lab can match the performances of the simulated agents by using multiple sensor arrays for both long and short range object detection. Figure 6.1 below demonstrates the lab maze setup where three mobile robots are solving the maze.

Figure 6.1 Actual Experimental Lab setup

The algorithm runs at a remote workstation, from where commands are sent to all the robots using wireless communication. A linux based interface, named “player”, is used to collect data from the robots and send it back to the remote workstation.

The remote workstation then interprets the data and utilizes the same to take the required decision for the next move of the robot, from which it came. The decision is then directed back to the robot using the same interface. The position of a robot is traced using an overhead Cognex 5600 image processing camera, using another linux based interface named “SwisTrack”. Tracking the robot positions helps in updating the values of agent matrix, as discussed in the algorithm in previous chapter.

Thus, a considerable effort has been done in duplicating the simulation environment in lab. The later sections of this chapter would now respectively discuss the robot dynamics along with two interfaces used in the lab experiment as discussed above. The last section of this chapter gives a detailed explanation of the application of the algorithm with the help of a flowchart.

6.2 KheperaIII Robot Dynamics

The Khepera III mobile robots are one of the high level robots that have multiple sensor arrays for long and short range object detection and can match the performance of much bigger robots. These mobile robots have a standard Linux Operating system and can be operated remotely using serial port communication or WIFI. The robot has a base of diameter 180 mm and includes an array of 9 infrared sensors for short range obstacle detection, up to 30 cm, as well as 5 ultrasonic sensors for long range object detection, from 20 cm to 4 meters [15]. Apart from that, these mobile robots also have an optional pair of ground infrared sensors for the line following and table edge detection. There are two DC motors attached to the wheels, and power is supplied from a swappable battery pack system.

The Khepera III robots are used in combination with Korebot II extention, which provides a standard C/C++ environment for application development. The KoreBot II GNU C/C++ cross-compiler provides a powerful standard tool for complex code compilation and supports all the standard C library functions [15]. As discussed in the last section of this chapter, the algorithm runs on a remote desktop on a linux based C environment. The commands are sent to the KoreBot using a linux based interface named “Player”. Player is a network server for robot control which provides a clean and simple interface to the robot’s sensors and

actuators using the TCP /IP network. It is an open source interface which is independent of the language platform and allows multiple devices to present the same interface. It is designed to support any number of clients [42].

For the lab experiment, the 9 infrared sensors, present on the outer periphery of the robot, are used to collect the data about the present environment. Data obtained from these sensors is then sent back to the workstation through player interface using WIFI communication. This data is then coupled into four groups to determine the presence of obstacles in all the four directions i.e. front, back, left and right. Figure 6.2 below shows the groupings of all the 9 sensors.

Figure 6.2 Grouped Robot sensors.

These grouped data are then analyzed by the program for deciding the next move of the robot in the maze and based on that, commands are sent back to the robot i.e. for its next move, the algorithm first analyzes all the directions for presence of obstacles and then suitable move is decided based on the algorithm.

Thus, input to the system is the data from the sensors, where as the output of the algorithm are the commands for robot. These commands are in the form of parameters i.e. speed and turn rate, that controls the movement of robot in the maze.

6.3 SwisTrack Tracking Tool

Tracking moving robots with an overhead camera is a technique used in maze experiment to keep record of their trajectories. This record helps in updating the cell's value in agent value matrix. As discussed in previous chapter, the algorithm analyzes these cell values along with data obtained from the sensors to define the next move of the robot. Thus, tracking of the trajectories plays an important role in optimizing the robot's movement in the maze.

SwisTrack is a generic and flexible tool for tracking robots which was developed by Corell et al. [43] in 2006. It is a stable, production quality package for both marked and markerless agents and can support both Windows and Linux systems. SwisTrack package has several inbuilt components available for various camera modules and can be extended easily by implementing new modular components. SwisTrack interface provides a platform to setup a processing pipeline, an ordered set of components or processing steps [44]. Figure 6.3 shows an existing pipeline for the maze experiment where each component in this pipeline has a series of parameters that are applied to the acquired frame with their current configuration. The parameters of each component can then be changed at runtime, and component output can be visualized in real-time on the screen.

Figure 6.3 Processing Pipeline for Maze Experiment

The process above, first, acquires the image from the Cognex 5600 camera and transforms that into a binary image, after converting it into a grayscale image and preprocessing it by applying a background subtraction of an empty maze area. A binary mask, consisting of the maze shape, is then applied on the resulting binary image to remove all the objects outside the maze. Robots to be tracked carries some pre defined ID rings one of which is shown in Figure 6.4. After obtaining a binary image, these ID's are detected as blobs (i.e. set of connected pixels with same value in a binary image), which are then converted into particles after being filtered based on their size so as to remove unwanted blobs from the list. The particles of successive frames are then tracked by using nearest neighbor tracking

method and sent to an output component file. Figure 6.5 shows the main window with the display containing maze and robot.

Figure 6.4 ID Ring used in Maze Experiment

Figure 6.5 Main Display Window for Swistrack

6.4 Algorithm Flowchart

Now since we have understood the construction and functionality of each and every component of the experimental setup, it would be appropriate to combine them all into one system and analyze how these components were used to implement the developed algorithm in actual lab conditions.

Figure 6.6 below demonstrates the integration of various components to form a system. As shown in figure, the robot carries an ID ring over its head and has a WIFI card enabled through which it can send the sensor readings back to the

Figure 6.6 Integration of Different Components in lab

remote workstation, which is also connected to a WIFI router. The ID ring helps the overhead cameras to recognize a particular robot and track its position, which in turn is transmitted back to remote workstation through the WIFI router and updates the present agent matrix cell values. The remote workstation, which has the algorithm activated, then analyzes the data from the robot sensors and the agent value matrix to determine the next move of the robot. The decision is then transmitted back to the robot in the form of control parameters and the cycle is repeated again for next move. Figure 6.7 explains the above discussed working of the model with the help of a flowchart.

Figure 6.7 Algorithm Flowchart for the lab model

Based on the above developed experimental setup and the algorithm flowchart, so far one robot has been successful in autonomously solving the lab maze. However, there are still some issues that need to be resolved for obtaining the desired results. The robot still repeats some of its explored paths because of the uncertainties of the data obtained from the sensors. Moreover, when the robot rotates in a particular direction, different from its initial one, the sensors are not able to adapt the direction change immediately i.e. after the robot rotates, the sensors continue to provide data based on their initial direction for next few moves. As we know that the algorithm analyzes this data for obtaining the next move, it is more likely to take wrong decision which ultimately leads to repetition of the already explored paths. It is recommended to develop a proper co-ordinate rotation system for the sensors that could update their positions immediately after the rotation.

In the event of two robots, the algorithm still needs to be modified in order to make it capable of obtaining data from both the robots simultaneously and to update their moves in agent matrix. This modification requires a lot of analysis which is yet to be worked upon.

Chapter 7

Conclusions and Recommendations

In this thesis, an intelligent decision based algorithm has been developed for a group of swarming multi agents for exploration and exploitation of an unknown obstacle rich environment represented by a 2D maze problem. Two different approaches i.e. “Carrot or Stick” approach and “Counter ant Algorithm” have been applied respectively and analyzed to the core.

The initial approach was successful, enabling one agent to solve any kind of maze, but when the system was scaled to two or more agents there arose some issues which made the carrot or stick approach inappropriate for this particular problem. A hybrid approach, incorporating counter ant algorithm along with the initial approach, was then developed to deal with such problems. Unlike the conventional ant algorithm, the counter ant algorithm gives preference to an unexplored path or the less explored path so as to reach the target in an efficient and optimized manner. Ability to handle dynamically changing environment becomes very important while exploring an unknown terrain. For this purpose a layer of fuzzy logic has been added to the counter ant algorithm. Moreover, the developed algorithm also incorporates a phenomenon of pheromone evaporation,

which provides a means for correction of a wrong decision. Results, based on MATLAB simulation, have been presented to discuss both the initial and final approach outcomes and the problems associated with them. Different figures of merit have been plotted to get an insight of the algorithm's performance for different scenarios and to find the preferred number of the agents for a particular maze size. Furthermore, the scalability of the algorithm has been shown by running the same algorithm, successfully, for different maze sizes as well as different agent groups. Another major achievement of this research is that, by applying the developed algorithm, one mobile robot has been made autonomous for solving a lab based maze environment. A detailed presentation about the experimental setup along with an in-depth explanation about its components has also been presented.

Drawback of this hybrid approach is that as the maze size increases the number of cell also increases, because of which the complexity as well as combined time taken by all the agents to get out of the maze also increases. Thus after a particular time period, the pheromone concentration gets totally evaporated leaving no trailing edge behind and thus increasing the total number of steps involved. Since we already know that the pheromone concentration plays an important role in taking direction decisions, therefore it is recommended that the algorithm should be capable of deciding the evaporation constant based on the size

of the maze. A layer of fuzzy logic having maze size and number of agents as inputs, along with certain set of rules, would be sufficient enough to make the intelligent decisions in this regard. Also with increase in number of agents, the probability of repetition of already explored paths also increases, which also contributes to increase in total number of moves as well as time taken. Thus the algorithm is also required to be capable of taking intelligent decisions based on the group size which is yet to be worked upon.

For the lab experiment, the data from the robot sensors is still quite unstable because of which the robot sometimes takes wrong decisions. Such wrong decisions tend to move the robot either towards the direction of already explored paths or towards the dead ends, which ultimately results in slow convergence. To avoid such situations, the interface “player” needs to be reconfigured with respect to the sensors responsiveness, which might involve writing a new code for the Khepera robots or making changes to the present code. Moreover, since the data from the sensors defines the presence of obstacles in a particular direction, the positioning of the sensors is of at most importance. For this reason a more appropriate co-ordinate rotation system is recommended for the sensors, so as to make the robot updated about the change of direction of the sensors. Presently the ID rings equipped by the robots are capable of determining the orientation of the

robot with respect to the walls. This property can be utilized to determine the correct position of the sensors after rotation. Furthermore, the frame rate of the camera has to be synchronized with the robot speed so that the tracking file generated by the swistrack tool can give a better estimate about the robot's position.

Future work would include the improvise version of the algorithm for the scalability purposes and to apply the improvised algorithm to two or more actual mobile robots. Moreover, it would be interesting to see how the current algorithm would behave for different cell shapes and sizes. A cell is an elementary maze item having a bounded space. The number of cells in a maze defines the complexity of the problem. Currently we have divided the whole maze area into a number of square cells, each having same cell size. These square cells helps the agents to maintain a constant step size while exploration. However, in real world applications the environment may contain different obstacles of varying shapes and the robots may be required to change their step sizes in order to adapt those environments. Being capable of handling different cell shapes and sizes would make the algorithm efficient to handle such dynamic situation.

Furthermore, it would also be interesting to see the application of the developed algorithm on to 3-D maze. Currently the developed algorithm is capable

of solving 2-D Mazes, utilizing the sensors present on the outer periphery of the robot. For the purpose of 3-D mazes, it would be required to integrate the sensors present at the bottom of the robot for the purpose of determining obstacle in third direction. Also it would be interesting to expand the current problem for multiple entrances or exits. Being capable of handling such opening, the algorithm can then be utilized for more real world applications like autonomous material handling on production plants, resource allocations for forest fires etc.

References

- [1] Gerhard Weiss, “*Multi-agent systems: a modern approach to distributed artificial intelligence*”, MIT Press, 1999 ISBN 0-262-23203-0
- [2] S. L. Wu, K. C. Tung, Y. R. Liou, W. H. Lin, M. H. Wu, “A Multi-agent Algorithm for Robot Soccer Games in Fira Simulation League,” *FIRA RoboWorld Congress, USA*, 2007
- [3] Cinthya Solano-Aragón¹ and Arnulfo Alanis, “A Multi-agent Architecture for Controlling Autonomous Mobile Robots Using Fuzzy Logic and Obstacle Avoidance with Computer Vision”, *Journal : Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition, Studies in Computational Intelligence*, Vol 256/2009, pp. 215-246
- [4] Olivier Buffet, Alain Dutech and François Charpillet, “Shaping multi-agent systems with gradient reinforcement learning” , *Journal : Autonomous Agents and Multi-Agent Systems*, Vol 15, No. 2 / October, 2007, pp.197-220
- [5] D.C. Dracopoulos, “Neural Robot Path Planning: The Maze Problem”, *Journal: Neural Computing & Applications*, Vol 07, No. 2, pp. 115-120
- [6] Fang Wang and Eric Mckenzie, “A Multi-agent based Evolutionary Artificial Neural Network for General Navigation in Unknown Environments”, *Proceedings Of 3rd International Conference on Autonomous Agents* , Seattle, May 1999, pp. 154-159
- [7] Jason Blazakis, “Border Security and Unmanned Aerial Vehicles”, *CRS report for congress*, Jan 2004
- [8] L. Lov’asz, “Random Walks on Graphs: A Survey,” *Bolyai Society Mathematical Studies*, pp 1-46,1993

- [9] Ivan E. Sutherland, "A Method for Solving Arbitrary-Wall Mazes by Computer," *IEEE Transactions on Computers*, Vol. C-18, No. 12, December 1969
- [10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, "Introduction to Algorithms," Second Edition. MIT Press and McGraw-Hill 2001, pp.540– 549
- [11] Donald Wunsch, "The Cellular Simultaneous Recurrent Network Adaptive Critic Design for the Generalized maze problem has a simple closed-form solution", *IEEE 0-7695-0619-4/00*, 2000, pp. 79-82
- [12] Nolfi, S. and Floreano, D. "Evolutionary Robotics- The Biology, Intelligence and Technology of Self - Organizing Machines", MIT Press.
- [13] S. Goss, S. Aron, J.L. Deneubourg, and J.M. Pasteels, "Self-organized shortcuts in Argentine Ant," *Naturwissenschaften*, vol 76, pp. 579-581, 1989
- [14] J.L. Deneubourg, S. Aron, S. Goss, and J.M. Pasteels, "The self-organized exploratory pattern of the Argentine ant," *Journal: Insect Behavior*, vol. 3, pp. 159-168, 1990
- [15] <http://www.kteam.com/kteam/index.php?site=1&rub=22&page=197&version=EN>
- [16] Elad Kivelevitch, Kelly Cohen, "Multi Agent Maze Exploration", *AIAA/ASM Conference*, Orlando, FL, January 2010.
- [17] Werbos PJ, Pang X., "Generalized maze navigation: SRN critics solve what feedforward or hebbian nets cannot", *World Congress on Neural Networks*, San Diego, CA, 1996, pp. 88-93.
- [18] Ilhem Kallel, Abdelhak Chatty and Adel M. Alimi, "Self-Organizing Multi-Robot Exploration Through Counter-Ant Algorithm," *IWSOS, LNCS 5343*, 2008, pp. 133-144
- [19] L. A. Zadeh, "Fuzzy Sets", *Information and Control*, Vol. 8, 1965, pp. 338-353

- [20] Valeri Rozin and Michael Margaliot, “The Fuzzy Ant,” *IEEE Computational Intelligence Magazine*, November 2007, pp. 18-28
- [21] Minsky, Marvin, Papert, and Seymour, “Perceptrons: An Introduction to Computational Geometry”, MIT Press, 1969
- [22] Houillon P, Caron A., “Planar robot control in cluttered space by artificial neural network”, *Journal: Math Modeling and Computing*, 1993, pp. 498-502
- [23] Floreano, D. and Mondalo, F., “Automatic creation of an autonomous agent: Genetic evolution of neural-network drive robot”, *Proceedings of Third conference on Simulation of Adaptive Behaviour*, 1994
- [24] http://en.wikipedia.org/wiki/Maze_solving_algorithm#Pledge_algorithm
- [25] D. B. Fogel, “*Evolutionary Computation: The Fossil Record* ” MIT-IEEE Press, 1998
- [26]. J. Holland, “*Adaptation In Natural and Artificial Systems*”, MIT Press, reprinted edition, 1992
- [27] Roman Neruda , Petra Vidnerová, “Learning Algorithms for Small Mobile Robots: Case Study on Maze Exploration”
- [28] Bellman, Richard (1957), “*Dynamic Programming*”, Princeton University Press. Dover paperback edition (2003), ISBN 0486428095
- [29] Jean-Claude Latombe, “*Robot Motion Planning*”, Kluwer Academic Publishers, Norwell, MA, 1991
- [30] Dimitris C. Dracopoulos “Robot Path Planning for Maze Navigation", in *Proceedings of the 1998 World Congress on Computational Intelligence-International Joint Conference on Neural Networks (IJCNN'98)*, IEEE Press, May 4-9, 1998, Anchorage, Alaska, USA
- [31] M. J. Matarić, “Learning in behavior-based multi-robot systems: policies, models, and other agents,” *Journal of Cognitive Systems Research*, vol. 2, 2001, pp. 81–93

- [32] Brooks, R.A. (1991). “Intelligence without reason” *In:Proceedings, IJCAI-91*, Morgan Kaufman, Sydney, Australia, pp. 569-595.
- [33] Brooks, R. A., & Stein, L. A. (1994). “Building brains for bodies” *Autonomous Robots* 1, pp.7–25
- [34] Brooks, R. A. (1986). “A robust layered control system for a mobile robot”, *Journal of Robotics and Automation RA-2*, pp. 14– 23
- [36] Arkin, R. C., “Towards the unification of navigational planning and reactive control” *AAAI spring symposium on robot navigation*, pp. 1–5.
- [37] Connell, J. H. (1991), “A hybrid architecture applied to robot navigation” *In: IEEE international conference on robotics and automation*, Nice, France, pp. 2719–2724
- [38] Mohit Ahuja, Baisravan HomChaudhuri, Kelly Cohen, Manish Kumar, “Fuzzy Counter Ant Algorithm for Maze Problem”, *AIAA/ASM Conference*, Orlando, FL, January 2010.
- [39] http://en.wikipedia.org/wiki/Supervised_learning
- [40]http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm
- [41] <http://reinforcementlearning.ai-depot.com/Intro.html>
- [42] <http://playerstage.sourceforge.net/index.php?src=player>
- [43] N. Correll, G. Sempo, Y. L. de Meneses, J. Halloy, J.-L. Deneubourg, and A. Martinoli, “Swistrack: A tracking tool for multi-unit robotic and biological systems,” in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*. IEEE/RSJ, 2006, pp. 2185—2191
- [44] Thomas Lochmatter, Pierre Roduit, Chris Cianci, Nikolaus Correll, Jacques Jacot, Alcherio Martinoli, “SwisTrack - A Flexible Open Source Tracking Software for Multi-Agent Systems”, *IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS 2008)*, September 2008.