Multi-Robot Cooperation in RoboCup-SPL

By
Zain Murtaza
Reg No: NUST201362039MSMME62113F

Supervised By
Dr. Yasar Ayaz

A Report Submitted to the
Department of Robotics and Artificial Intelligence
In Partial Fulfillment of the Requirements for the Degree of
Masters in
ROBOTICS AND INTELLIGENT MACHINE ENGINEERING

School of Mechanical and Manufacturing Engineering (SMME),
National University of Sciences and Technology (NUST),
Islamabad, Pakistan
October, 2016

FORM TH-4

National University of Sciences & Technology

MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: **Zain Murtaza (NUST201362039MSMME62113F)** Titled: **Multi-Robot Cooperation in RoboCup-SPL** be accepted in partial fulfillment of the requirements for the award of **Masters in Robotics and Intelligent Machines Engineering** degree with grade _____ .

Examination Committee Members

1. Name: **Dr. Muhammad Naveed** Signature:_____

2. Name: **AP Sara Babar Sial** Signature:_____

3. Name: **Dr. Hasan Sajid** Signature:_____

Supervisor's name: **Dr. Yasar Ayaz** Signature:_____

Date:_____

_____ _____

Head of Department Date

COUNTERSINGED

Date:_____

_____

Dean/Principal

Declaration

I certify that this research work titled "*Multi-Robot Cooperation in RoboCup-SPL"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Zain Murtaza

NUST201362039MSMME62113F

Signature of Student

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Zain Murtaza

Signature of Student

Registration Number

NUST201362039MSMME62113F

Signature of Supervisor

Acknowledgements

I am thankful to Allah who have all the powers for considering me worthy of this educational honor.

I am thankful to my parents and brothers who supported me both financially, morally and for providing me motivation whenever I needed.

I am thankful to my friends especially Muhammad Rizwan, Husnain Ahmed and Zaid Ahsan Shah for their guidance and cooperation. Staying with them I gained more than just an educational degree.

I would like to thanks Dr Yasar Ayaz whose support and well wishes were always behind me. Dr Yasar Ayaz made extra efforts for Team-NUST to compete in RoboCup 2016 in Germany. Without his efforts and guidance RoboCup project couldn't be possible.

I would like to thank my supervisor Dr. Yasar Ayaz, along with other committee members for evaluating this dissertation and for providing me with the requisite guidance and support, in order to successfully complete this work.

Finally, I would like to thank all students in RISE Lab for helping me throughout my research phase.

Dedicated to my Mother

Abstract

RoboCup Standard Platform League (SPL) is an international robotics soccer competition with an aim to foster research in artificial intelligence and its application in advance robotics. It involves multi-robot cooperation to achieve a common team objective. Hence the need of dynamic task allocation in real time, under a partially known and dynamic environment, arises to achieve a global objective through cooperation or task decomposition. Multi-robot cooperation has gained some attention in the recent past, but to-date only a handful of architectures exist. However, most of the research is ad-hoc and focused on developing proof of concept of available architecture using simulation based techniques, while rarely being employed on actual hardware platforms. This research proposes a robust hybrid architecture for multi-robot cooperation that exploits the advantages of both market based and behavior based architectures, which was employed for RoboCup-SPL 2016 by Team-NUST.

Key Words: *RoboCup, Team-NUST, Multi-robot coordination, Task allocation, Hybrid architecture*

Table of Contents

Contents

Declaration ..... **i**
Language Correctness Certificate ..... **ii**
Acknowledgements ..... **iii**
Abstract ..... **v**
Table of Contents ..... **vi**
List of Figures ..... **ix**
List of Tables ..... **x**
Chapter 1: INTRODUCTION **1**
1.1 ROBOCUP 1
1.1.1 The RoboCup Leagues 3
1.2 RoboCup Standard Platform League (SPL) 4
1.3 Nao Robot 5
1.3.1 Choregraphe 6
1.3.2 Development Environment 7
Chapter 2: TEAM-NUST **8**
2.1 Team Participation in RoboCup 2016 8
2.2 Brief description of work 10
2.2.1 KICK 10
2.2.2 Vision 12
2.2.3 Localization 14
2.2.4 Motion Planning 14
2.2.5 Goal Keeper 15
2.3 Software Architecture 16
2.4 Challenges 18
2.4.1 Fully Autonomous Robots 18
2.4.2 Stability 19
2.4.3 Robot Safety 19

2.4.4 No overhead camera 19
2.4.5 Monocular cameras on robot 19
2.4.6 Constrained communication 19
2.4.7 Cooperative and competitive environment 20
Chapter 3: Multi-Robot Cooperation 21
3.1 Task Allocation & Task Decomposition 21
3.2 Multi-Robot Architectures 22
3.2.1 Centralized Architectures 22
3.2.2 Decentralized Architectures 23
3.2.3 Hierarchical Architectures 23
3.2.4 Hybrid Architectures 23
3.3 Communication 24
3.3.1 Cooperation-without-communication 24
3.3.2 Kin-recognition 24
3.3.3 Explicit-communication 24
3.4 Robot classification 25
3.4.1 Homogenous Robot Network 25
3.4.2 Heterogeneous Robot Network 25
3.5 Cooperation 26
3.5.1 Intentional cooperation 26
3.5.2 Non-intentional cooperation 26
3.6 Task Allocation Approaches 26
3.6.1 Behavior Based Approach 27
3.6.2 Market Based Approach 28
Chapter 4: Research Methodology and Implementation 30
4.1 Problem Statement 30
4.2 Proposed Architecture 30
4.3 Analytical Hierarchical Process (AHP) 31
4.3.1 Checking the consistency 33
4.4 Example Problem 34
4.5 Setting the priorities 38

4.6 Pseudo Code for the bidder and auctioneer 39
4.7 Capability Matrices 40
4.8 Behavior Base Model Integration 41
4.8.1 Inter-robot communication 42
4.8.2 Task Suppression 42
4.8.3 Impatience 42
4.8.4 Subsumption 43
4.8.5 Subsumption reset function 43
4.9 Architecture Overview 44
Chapter 5: Results and Conclusion 46
5.1 Simulation 1 46
5.2 Simulation 2 49
5.3 Conclusion 52
5.4 Future Work Suggestions 53
REFERENCES 55

List of Figures

Figure 1: RoboCup Standard Platform League (SPL) 4
Figure 2: Nao Humanoid Robot 6
Figure 3: Choregraphe for Nao Robot 7
Figure 4: Development Environment 7
Figure 5: Team-NUST logo 9
Figure 6: Team-NUST shirt design 9
Figure 7: Foot Planning 10
Figure 8: Approximated foot contour with Bezier curve 10
Figure 9: Velocity Results 11
Figure 10: Desired distance results 12
Figure 11: Field, Line, Corner detection 13
Figure 12: Robot detection 13
Figure 13: Ball Detection 13
Figure 14: Localization in field 14
Figure 15: Motion planning (potential field) 15
Figure 16: Software Architecture 16
Figure 17: Architecture flow, Block Diagram 17
Figure 18: Augmented finite state machine 28
Figure 19: Example Problem 34
Figure 20: Propose Architecture overview 44
Figure 21: Simulation 1 - Each robot completing the task 47
Figure 22: Simulation 1 - Parameters after simulation was terminated at task completion 47
Figure 23: Simulation 1 - Robot B (green) assigned the task 48
Figure 24: Simulation 1 - Robot A (Red) assigned the task when Robot B (Green) failed 49
Figure 25:: Simulation 2 - Each robot completing the task 50
Figure 26: Simulation 2 - Parameters after simulation was terminated at task completion 51
Figure 27: Simulation 2 - Robot C (blue) assigned the task 52

List of Tables

Table 1: Value of random index (RI) for small problems 34
Table 2: Pair-wise comparison matrix 35
Table 3: Option score matrix relative to time parameter 36
Table 4: Option score matrix relative to Energy parameter 36
Table 5: Option score matrix relative to distance parameter 37
Table 6: Lookup Table for priorities 39

Chapter 1: INTRODUCTION

The research work proposed in this thesis is divided into two main parts. First part addresses the proposed strategy and novel methods for the research and technical challenges in RoboCup SPL by Team-NUST. Second part proposes a more fault tolerant and robust hybrid architecture for the multi-robot collaboration employed initially for RoboCup SPL by Team-NUST.

1.1 ROBOCUP

RoboCup is an international competition, kind of like world-cup, it aims to fuel up the developments in advance robotics, artificial intelligence by providing a common ground for international researchers and providing educational initiatives. Hence RoboCup is a robotic world-cup and it's one of the biggest competition of robotics in world. For the competition purpose robot soccer was chosen as the primary game.

Hence RoboCup is a task for multiple robots acting in coordination for a common global objective. These robots have to respond quickly in a fast changing environment and for a team of robots to play soccer a lot of technologies must be considered. For instance the team of

- Robots must be able to play autonomously without external human intervention
- Robots should be able to respond effectively to the environment in real time.
- Robots should adopt strategies as a team and also consider local control.
- Robots should act together in coordination toward a common global objective.

The official goal of the RoboCup project is that, a team of autonomous life size humanoid robots will win a soccer game against the winning team of most recent human soccer world-cup, complying with the official rules of FIFA by 2050.

RoboCup is also adopting its reputation as a new standard problem in the robotics research community. A standard problem is a challenge which is defined clearly and is used in research community by the researchers across the globe to evaluate, analyze and compare their

algorithms and research work with each other. In the days of classical artificial intelligence turning proposed that “Chess” should be the standard problem because chess is clearly defined and a difficult task, but with the advancements in research and computing chess slowly became less attractive as it was easy for new algorithms and research to solve this problem and became too simple to provide the room for more innovation and research. Deep blue a computer system designed by IBM defeated Garry Kasparov, the human grand chess master (although the match results are controversial). On the other hand RoboCup is cutting edge technology and spurs innovation and research especially in robotics community the reason and a brief comparison is as follow.

- RoboCup offers a dynamic environment which means that environment is continuously changing hence it becomes very challenging for the robot to respond in fast moving dynamic environment for instance in RoboCup-SPL all the robots are moving around and changing position in fields and attacking, the opponent team must be able to quickly respond against fast moving team of robots. On the other hand the chess offers a static environment which can also be divided into discrete states hence the computation becomes simpler.
- RoboCup offers a platform in which the robot must be able to respond to the continuous environment in the real time which means that robot must be able to do all its perception, cognition and action in the real time and within tight time limits, for instance in RoboCup-SPL all the positions of robots and the ball are changing continuously and the robots should respond in real time for the proper team play. While the chess depends on turn taking and each discrete state is changed after each turn one by one.
- RoboCup offers an environment where information accessible to the robot is complete, a robot facing a specific direction can only observe that part of the environment, at any time instance robot is exposed to a specific information only hence it becomes challenging for the robots to take decision in partially observable environment. In RoboCup-SPL Nao robots are used and their camera are mounted in position of lips and forehead hence head is moved accordingly for effective perception. On the other hand in chess the environment is fully

observable and at every time instant the whole state of the problem is known making it easier to solve.

- Sensor reading in the RoboCup are continuous and depending on the robot the data acquisition and the data processing for the usable or relevant information can be very challenging for instance in RoboCup-SPL use of camera as the only sensor for the perception makes data handling and image processing more complex. While on the other hand in chess the sensor readings can be symbolic the state of the board can be easily be interpreted even the force/pressure sensors are enough to interpret the whole state of the game.
- RoboCup offers distributed control, each team member has its own local control law and local performance measures and on the global level they have common team objectives each robot as a separate entity participate toward a common team objective in a distributed fashion researchers classify this as decentralized architecture. In the RoboCup-SPL each robot makes its own decision and then also execute it using local control while acting in coordination with other team members all together. While in the chess the control is central all the information is dumped in a central processor and all cognition occurs in a single standalone central processor.

1.1.1 The RoboCup Leagues

RoboCup environment offers a lot of leagues to attract the multidisciplinary researchers across the globe. Each of this leagues is focused toward a specific domain and offers the challenging problems from a wide spectrum of research domains. Some of the leagues in RoboCup are mentioned below.

- Standard Platform League
- Small Size League
- Middle Size League
- Rescue Robot League
- Simulation League
- RoboCup @ Work

- RoboCup @ Home
- RoboCup Junior League
- RoboCup Logistics

1.2 RoboCup Standard Platform League (SPL)

RoboCup Standard Platform League (SPL) is one of the most famous league in RoboCup. As the name suggests in this league standard humanoid platforms are used. The purpose of using standard platforms is that to provide a platform where the researchers don't have to take account of hardware complexity and low level hardware problem. The researchers focus on developing the algorithms and architectures for the problem only excluding hardware problems altogether.

Standard platform league (SPL) is a soccer league in which standard NAO robots manufactured by Aldebaran are used by each team. These NAO robots are humanoid robots and in RoboCup SPL each team utilized them to play soccer with each other. Standard platform league has various sub competitions and technical challenges which are upgraded and modified each year. The competition initially used Sony Aibo robots as standard platforms which is a quadrupedal robot. The league is continuously evolving and progressing at very fast pace. The researchers focus on research domains like local control of humanoid, enhanced stability, efficient walk and multi-robot coordination etc. A brief representation of RoboCup SPL is given in Figure. 1 in which NAO humanoid robots are playing each other on a standard ground as specified by the RoboCup SPL rules.

Figure 1: RoboCup Standard Platform League (SPL)

RoboCup SPL is divided in three main events

1. Indoor main competition: This is the main event of the RoboCup SPL. In this event two teams play against each other, each team comprises of 6 Nao robots. 1 robot is goalie and other 4 play in field as defender or attacker. Other than these 5 robots there is another coach robot which doesn't physically participate in match but is located outside field, the coach robot can only observe the game and guide the team players.
2. Drop-in player competition: In this completion two teams play soccer against each other, each team comprises of 5 Nao robots. Each Nao robot in the team is contributed by a different team hence 10 teams play soccer in each round of this competition. The purpose of drop-in player competition is for robots to coordinate with each other without explicitly being hard programmed before the game. Distributed architecture and flexible solution are the key features of this competition.
3. Technical Challenges: Technical challenges change every year, and technical challenges are then integrated in the game play of following year. In RoboCup SPL 2016 the technical challenges were, outdoor team competition and no-wifi challenge. In outdoor team competition each team have to play a match in outdoor conditions where perception becomes really challenging and the field used for these matches were also different from previous years hence stability was also an issue. In no-wifi challenge two robots located far away were supposed to communicate with each other without means of wifi or ethernet medium.

1.3 Nao Robot

Nao is a standard programmable humanoid robot developed by Aldebaran for the research purposes. Nao robot is extensively being used in research and educational purposes hence the support community is also active. The robot has 25 degrees of freedom and all the inverse kinematic solutions for this robot are easily available. Aldebaran provides hardware and software support for the platform. A brief overview of Nao robot is given in Figure. 2.

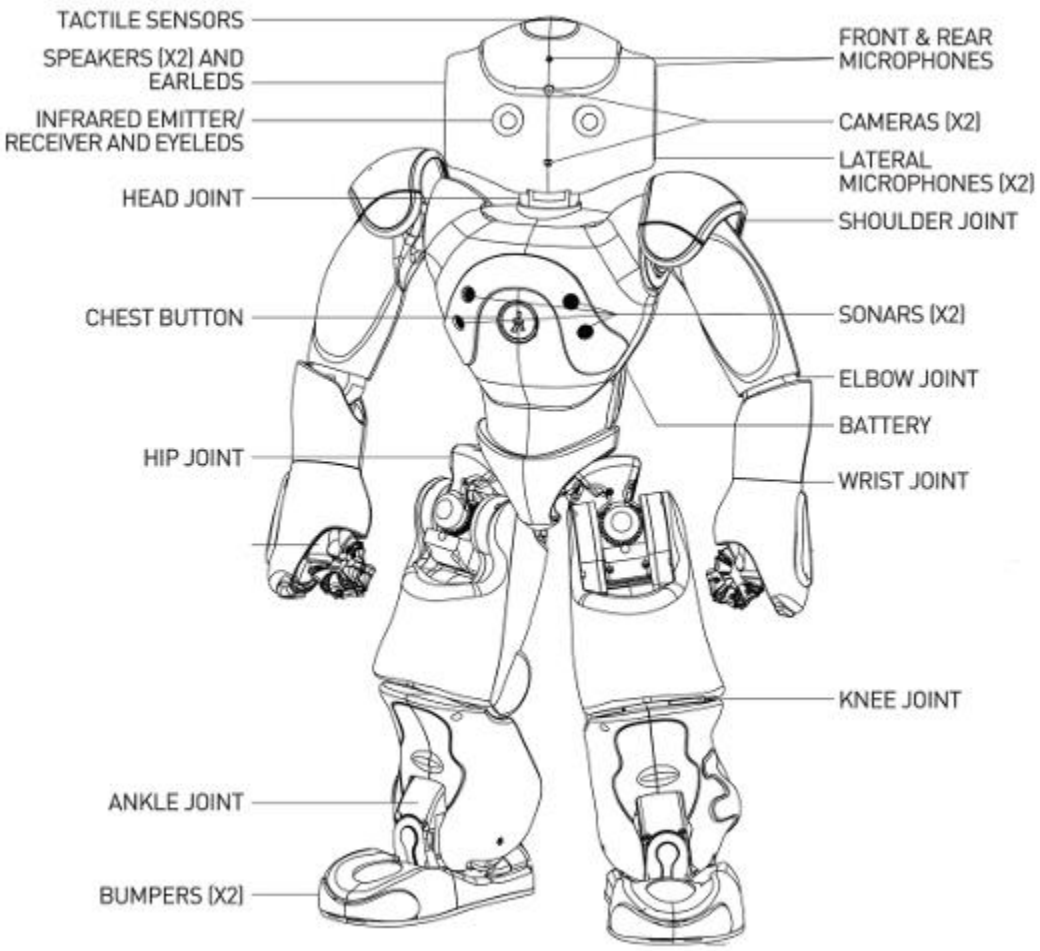

Figure 2: Nao Humanoid Robot

1.3.1 Choregraphe

Choregraphe is a software developed by Aldebaran especially for the Nao humanoid robot. The software can run on multiple platforms like Windows, Linux or Mac. It is very easy to use and is very suitable for beginners it allows the user to interact with robot using graphical user interface and without writing any code. It also allows to create and run certain behaviors on the robot and even support writing python scripts within the behaviors. It also allows the user to test the behaviors on the simulated robot as well as on real hardware. A brief outlook of the choregraphe software with available behavior sets (on the left side) and workspace with chosen behavior set (in the middle) is given in Figure 3. The choregraphe even allows creating complex behavior like human-robot interaction.

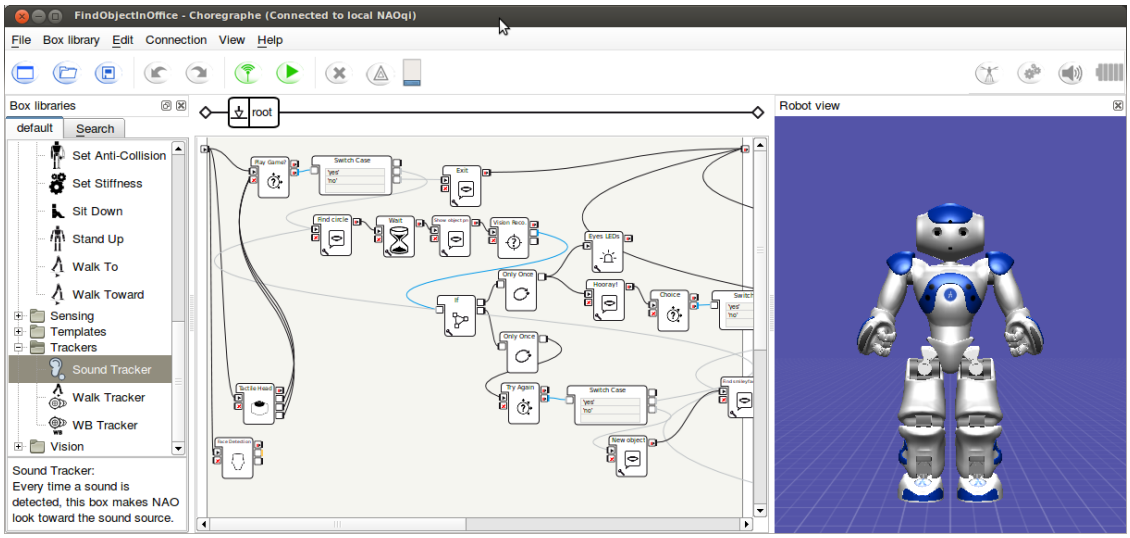

Figure 3: Choregraphe for Nao Robot

1.3.2 Development Environment

The programming framework which is used to program the Nao is called Naoqi. Naoqi is the main framework of Nao robot it runs on the Nao on board computer and controls the whole platform. Naoqi is cross-platform framework which means it is supported with multiple platforms and is also cross-language supported it can be developed using python and C++, moreover it also offers introspections. A brief overview of development environment for the Nao is given in Figure. 4. The code can be written in any IDE of choosing and on any platform of choosing, but for the cross compilation only Linux is support. After the cross compilation using the toolchain developed by Aldebaran the binaries also called local module is transferred inside the Nao memory using SSH connection.

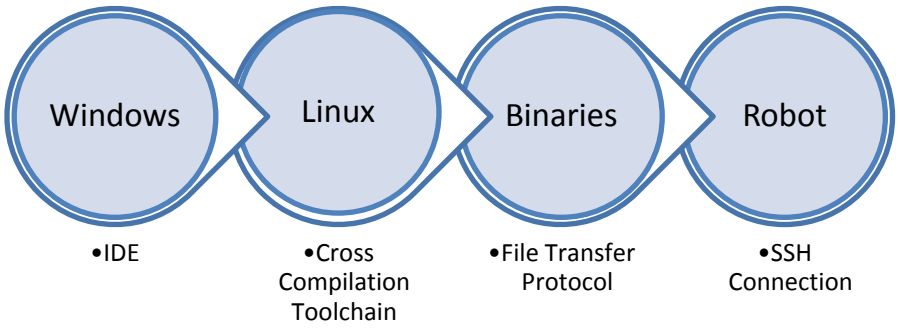

Figure 4: Development Environment

Chapter 2: TEAM-NUST

Team-NUST is a RoboCup SPL team from the National University of Sciences and Technologies (NUST) - SMME - Robotics & Intelligent Systems Engineering (RISE) lab. Team-NUST has consecutively qualified for the RoboCup SPL from 2014-2016 and also participated in RoboCup SPL 2016 held in Leipzig Germany.

2.1 Team Participation in RoboCup 2016

Team-NUST is the first and only team from South-Asia which has qualified and participated in RoboCup SPL. Participation in RoboCup has helped Team-NUST in technical knowledge advancement, as we have learned from other teams as well, approaching similar challenges as us. It has provided an opportunity to better contrast our research and ideas with fellow researchers across the globe. Participation of Team-NUST in RoboCup SPL 2016 has also provided other teams participating in the competition a means to get acquainted with the state of the art RoboCup research being conducted at RISE research center. Our participation in SPL has inspired other teams participating in SPL to focus on high level multi-robot task allocation and decomposition architectures.

As the only team from Pakistan, our participation has increased the diversity of the participating teams and help form new international collaboration avenues for future research. RISE Research center at NUST is currently the most advanced robotics facility in the country that houses the RoboCup Soccer Team. RoboCup is one of the largest research projects at RISE and responsible for sparking interest in a lot of tangent areas. Demonstrations of robots playing soccer by the RoboCup SPL Team at RISE positively contributes toward directing focus of researchers, students, faculty, and industry towards robotics and NUST. The RoboCup SPL Soccer team's work has been covered by several national TV channels and newspapers and was appreciated throughout the country. Our participating in RoboCup SPL 2016 also contributes positively toward acceleration of Robotics awareness in Pakistan. As no team other than team-NUST qualified from South Asia, it is an excellent mean to encourage robotics research at a national level and to encourage student's interest.

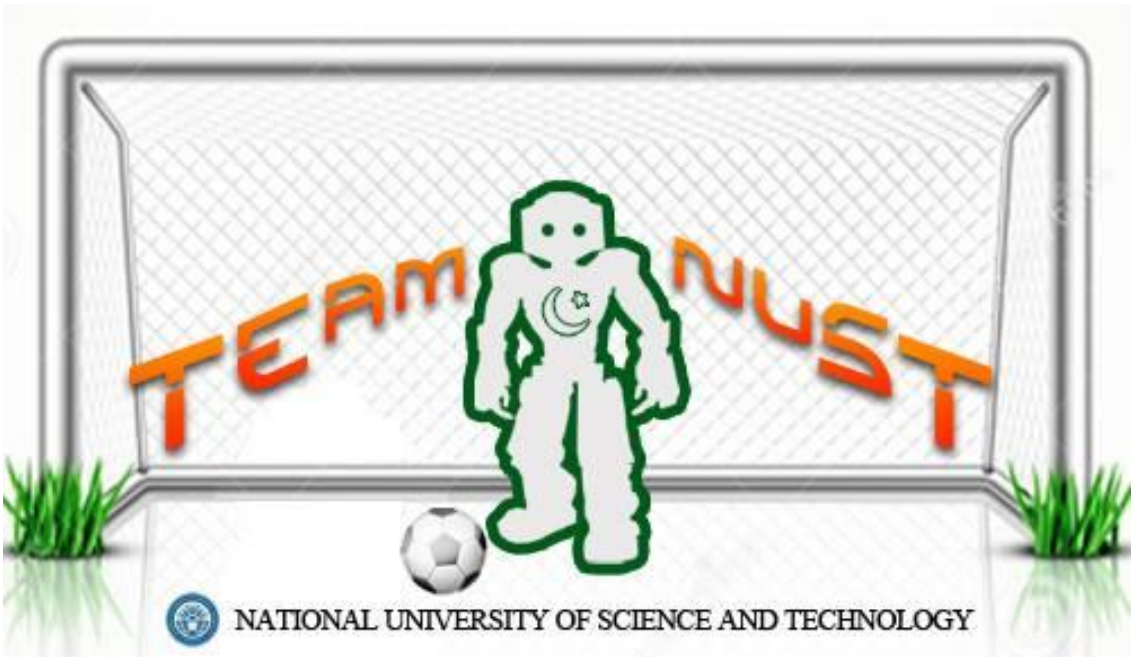

Figure 5: Team-NUST logo

Team-NUST official logo for RoboCup is given in Fig.5 while the team shirt design use by Team-NUST in RoboCup 2016 held in Germany is given in Fig.6.

Figure 6: Team-NUST shirt design

2.2 Brief description of work

This section addresses the brief description of work by Team-NUST for RoboCup 2016

2.2.1 KICK

For the kick, a new kick engine is developed with the capability of producing multi-directional, dynamic and impact controlled kicks. The engine only takes two inputs, the local coordinates of the ball and the target. Based on the target direction and distance, the ball dynamics and friction model is used to find the desired initial velocity of the ball.

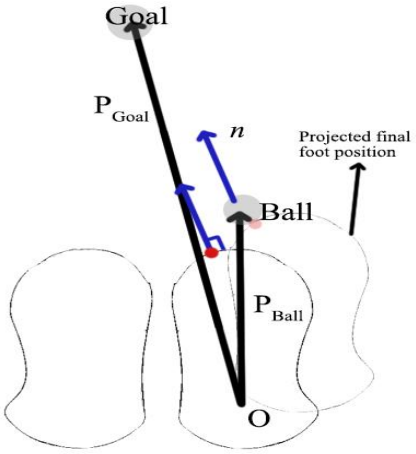

Figure 7: Foot Planning

Then a point on the foot contour is found which makes a perpendicular with the target direction and chosen as the contact point, where the foot contour is based on two approximate Bezier curves.

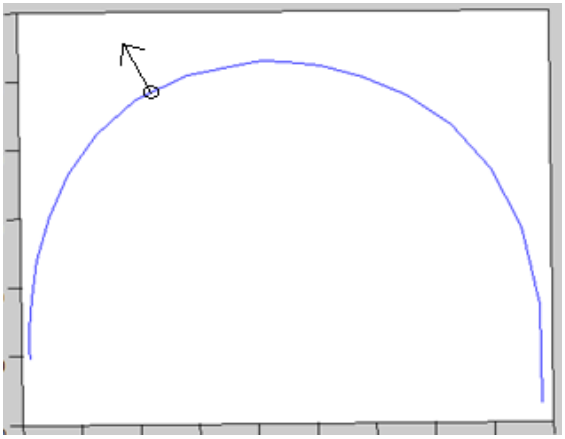

Figure 8: Approximated foot contour with Bezier curve

The direction is projected back onto a circle of radius r, to find the retraction point, where r is adjusted depending upon the desired speed needed for the kick. Using the inverse kinematics, the joint configurations at the retraction point and the final ball hitting point are calculated. The dynamic model of the kicking leg is then used to find the mass matrix at the final configuration and further it is used to find the virtual mass in the desired kicking direction. Using the virtual mass, the ball mass and the desired ball velocity, a linear elastic collision model is constructed to find the velocity needed for the foot contact point. The resultant velocities are then converted into joint velocities and finally fed into the trajectory generator. The trajectory generator takes the retraction point and the ball contact point as the knot points. It utilizes the joint configurations (acceleration, velocity, position) at knot points and constructs a quintic spline to provide smooth and continuous position, velocity and acceleration profiles. The trajectory is then executed.

2.2.1.1 Results

To find out the results of impacts, an overhead firefly MV 60 fps camera was used. The frames just before and after the impact were taken and the ball pixels displacement was detected. Matlab was used for calculating ball displacements and doing further calculation. The graph in Fig.9 shows the difference between the desired velocity and actual measured velocity using the experiment

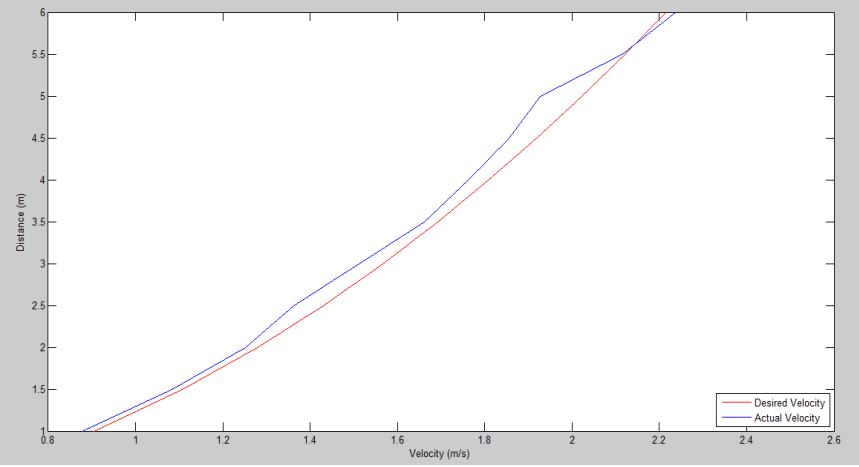

Figure 9: Velocity Results

The graph in Fig.10 shows the results of different straight kicks with desired distances (1m, 2m, 3m, 4m, 5m, and 6m). The standard error mean (SEM) in each case is: SEM_1 = 6.8%, SEM2 = 6.1%, SEM_3 = 2.01%, SEM_4 = 5.39%, SEM_5 = 3.57%, SEM_6 = 4.73%

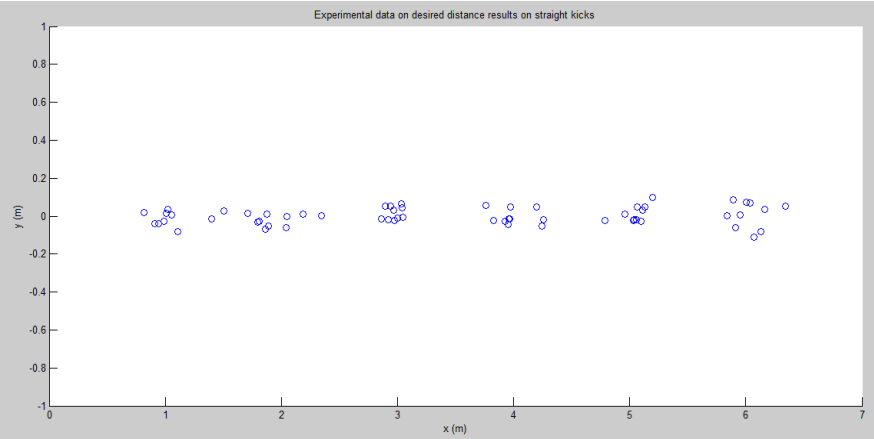

Figure 10: Desired distance results

Consideration is given to the allowed range of feet, and this is adjusted in the path planning stage before kick where the robot is aligned in an orientation such that it can perform the kick.

2.2.2 Vision

Vision is one of the most crucial factor in RoboCup SPL because the cognition and decision making mostly rely on this module. No matter how stable the robot is how efficiently it walk or kicks if the vision module is not working efficiently it results in poor performance of whole robot. Localization, kick, walk rely on data from the vision for instance field extraction then corner and line detection, goal post detection, robot detection and ball detection. The algorithm and solution for the vision problem were mostly focused maintaining the robustness and variation of environment at the same time.

Color features were taken as starting point for detecting different field features like field bounding edges, field lines, corners and objects such as robots and goal post. The camera model is used along with forward kinematics to determine distance information of landmarks in scene. Field area is extracted from each frame using an adaptive color range, and inverse perspective transform which is derived from forward kinematics and camera model, is applied on each frame

to get a bird's eye view of the visible field area. This bird's eye view makes it easier to detect field corners and to distinguish goal post from field lines since both of them have white color.

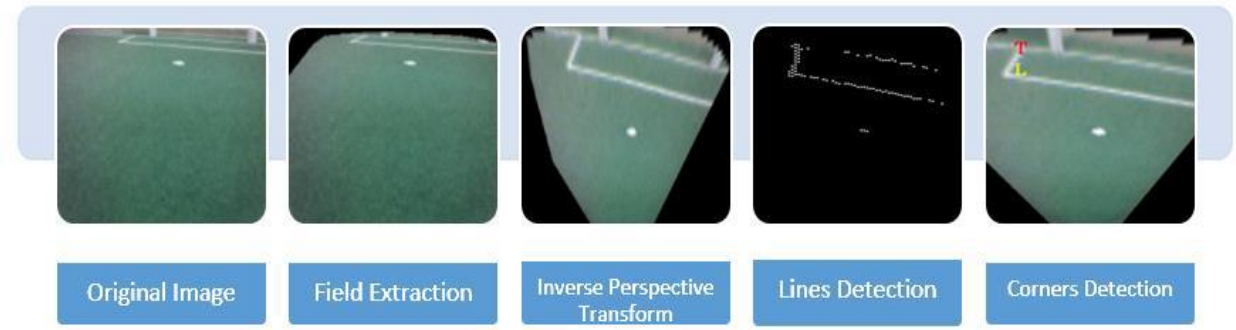

Figure 11: Field, Line, Corner detection

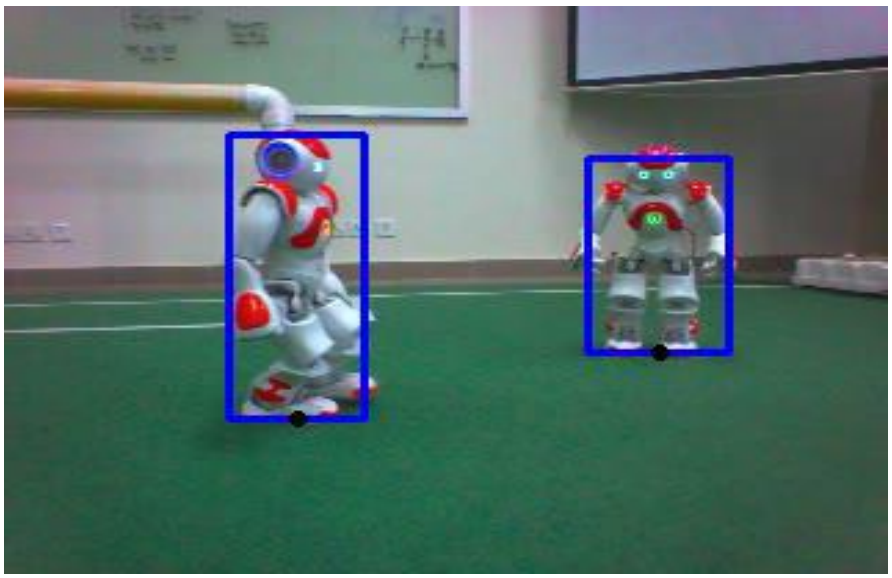

Figure 12: Robot detection

Soccer ball detection is done by combining different methods used in object detection. These include color based detection and shape/polygons based detection etc. A simple illustration is shown in Fig. 13. Once detected, the ball is tracked using histogram comparison algorithm and position and size is updated after regular intervals to minimize error while tracking.

Figure 13: Ball Detection

2.2.3 Localization

Landmarks like field, lines and corners are used by Kalman and Particle filters for localization. An odometric model is used for both filters. Particle filter is used to solve kidnap problem. Once a unimodal distribution is established Kalman filter is used to track the estimate with extra states to estimate slippage and odometric errors. We are working on localizing dynamic objects and adding them to world belief which is shared among all players at run time. Fig.14 below shows the successful localization of the robot in field.

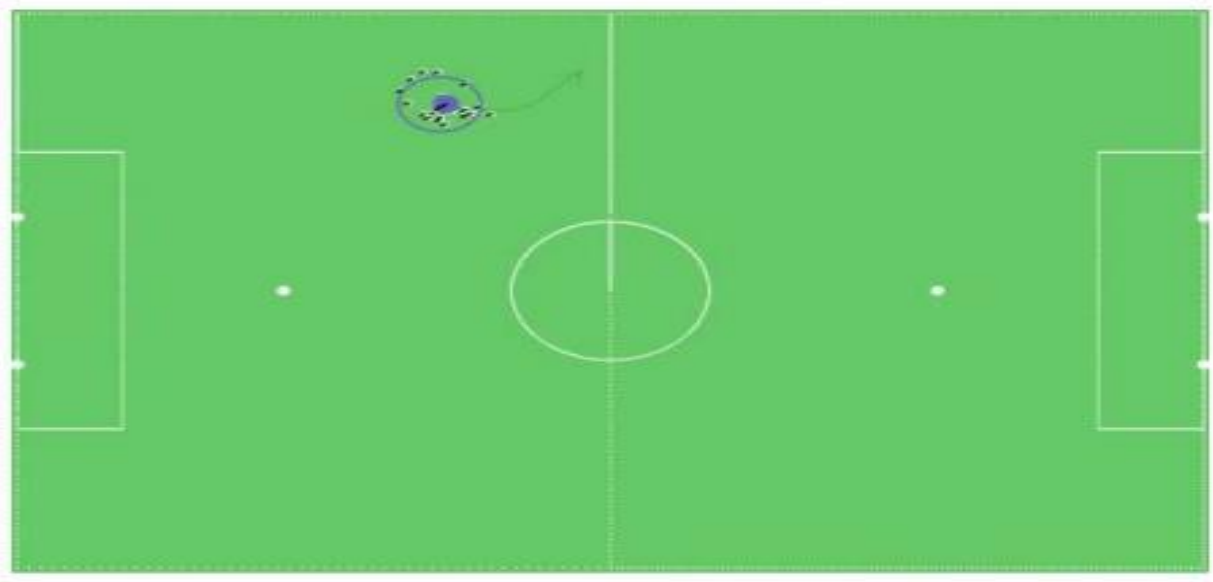

Figure 14: Localization in field

2.2.4 Motion Planning

Robot motion in the field was planned using different methods. Trajectory generator plans an estimated path for longer distances. Motion of the robot while approaching the ball at close proximity has been experimented using potential fields as well as footstep planning. The approach angle toward the ball and the robot position and orientation is also an important variable used in ball kicking. A Bezier curve is generated towards the ball, considering robot direction towards the target and to generate the trajectory of kicking foot. The practical result from potential field based motion planner is given in Fig. 15.

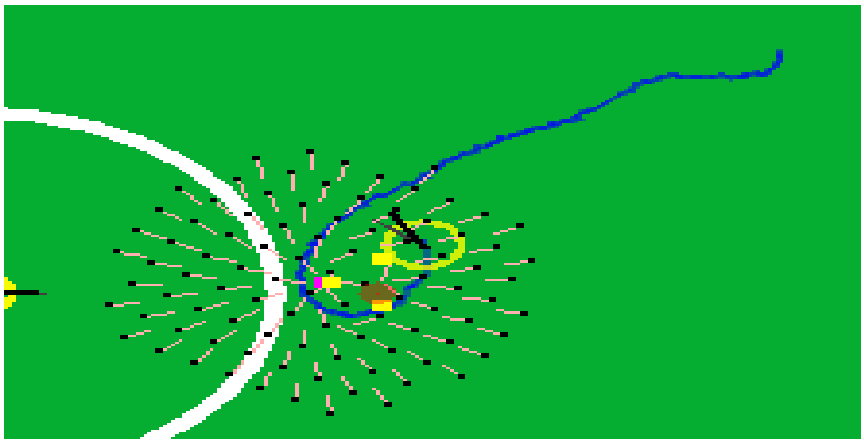

Figure 15: Motion planning (potential field)

2.2.5 Goal Keeper

Taking right decision, in limited time, for a goalkeeper is of significant importance as it plays a vital role in winning a game. Heuristic based decision-making is used for goalkeeper. The goalkeeper's behavior is implemented using different components, each one providing different feature of a goalkeeper. Components are organized into different level or hierarchies to implement behaviors of a goalkeeper. Tasks that require several components are triggered in a hierarchical tree. Each component is made up of states and transitions that help the states to switch between each other. Pseudo code for state transitions is shown below.

- $[\quad - \quad , \ start \quad] \Rightarrow$ [SeekingGoal]
- $[SeekingGoal\,,\ goalFound] \Rightarrow$ [Positioning]
- $[Positioning,\ inPosition] \Rightarrow$ [SeekingBall]
- $[Positioning\,, goalLost] \Rightarrow$ [SeekingGoal]
- $[SeekingBall,\ ballFound] \Rightarrow$ [Defending]
- $[Defending\,,\ ballAtFeet] \Rightarrow$ [ClearingBall]
- $[Defending\,,\ ballNear] \Rightarrow$ [MovingToTheBall]
- [MovingToTheBall$\,,\ \ ballNear] \Rightarrow$ [ClearingBall]
- [MovingToTheBall$\,, ballost] \Rightarrow$ [SeekingBall]
- [ClearingBall$\,,\ ballCleared] \Rightarrow$ [SeekingGoal]

2.3 Software Architecture

The software architecture used by Team-NUST is briefly given in Fig. 16 below.

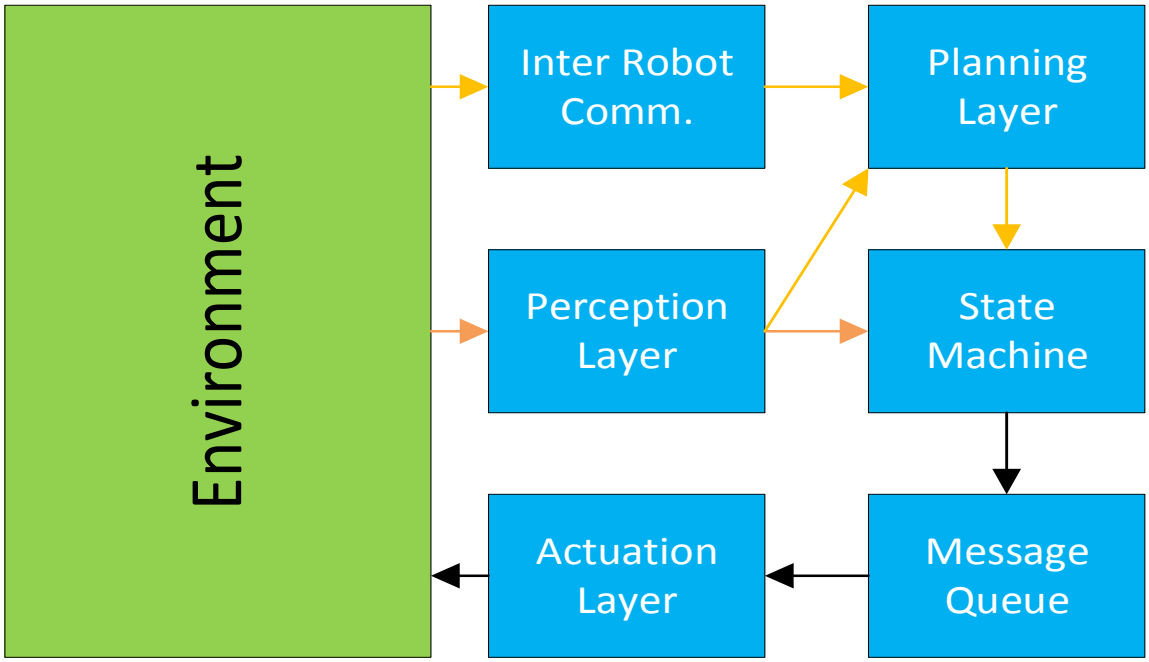

Figure 16: Software Architecture

The perception layer uses the sensors on robot to collect the data from the environment which include information like field, ball other robots etc. Inter-robot communication layer collect the data from the other team players which include data like current role of team member and ball location etc. This data is then fed into planning layer for cognition and proper plans are selected accordingly. These plans are then fed into state machines which has high level state functions. State machines generate the messages in a specific sequence for any particular behavior from the planning layer and queen them. These message queues are then executed and low level function are performed at actuation which finally in turn effect the environment.

Sometimes during the game play the robots don't need to make team strategies and plans to encounter the situation. For instance if a robot has a ball lying right in front of it and it is clear to shoot the ball inside the goal. The robot in this situation need to generate a quick reactive response and kick the ball instantaneously hence to consider this an input is introduced into state machine directly from perception which is responsible for reactive responses and also adds more flexibility in team play.

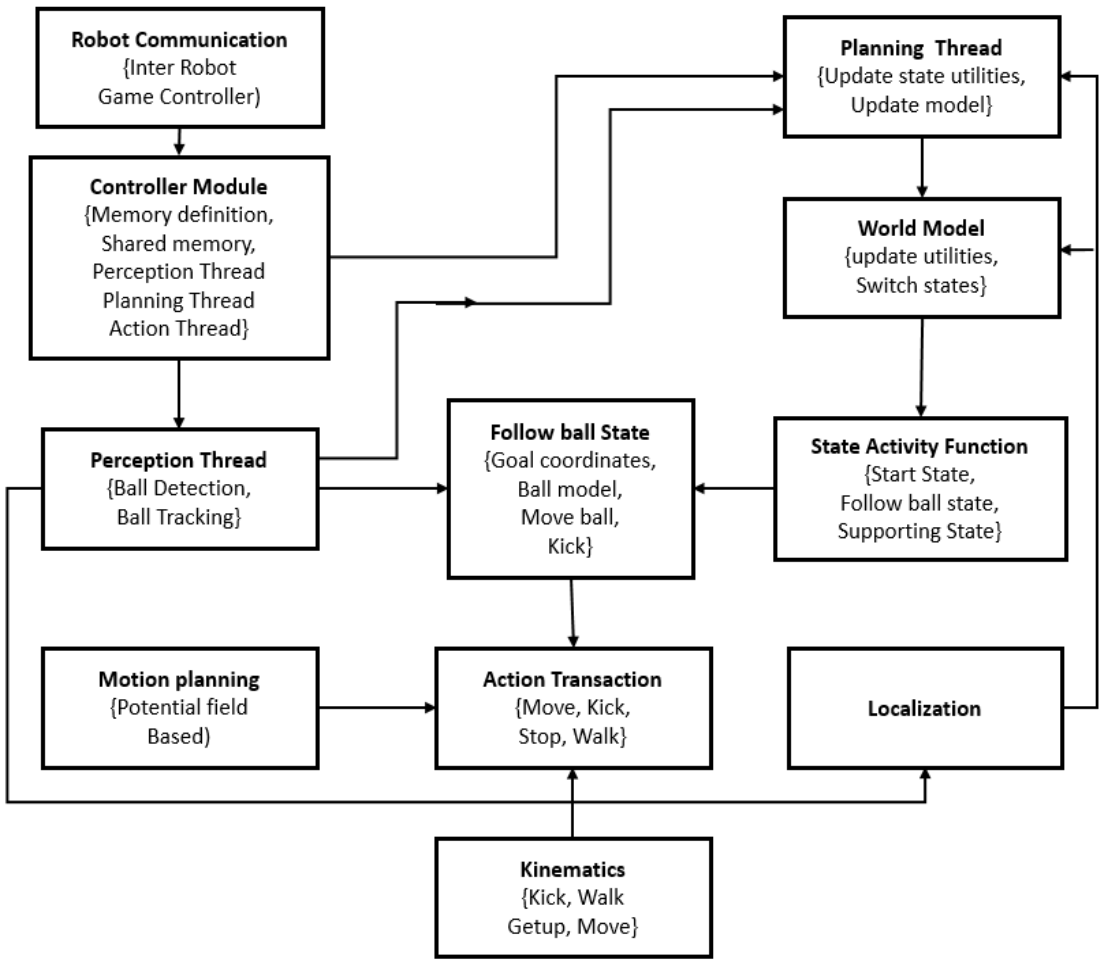

Figure 17: Architecture flow, Block Diagram

An overview of the flow of information in the software architecture is given in Fig. 17. This block diagram also briefly depicts the role of each layer and dependencies of modules with each other.

First of all controller module is responsible for initiating various threads like planning thread, perception thread, action thread. In each thread variables are defined, input and output connector for each is defined and memory definitions are also created. This module is also responsible for creating main broker. Perception thread also processes the data from cameras and produces useful information. This thread doesn't provide raw data for processing but process the data and provides the data to the other modules like planning and localization. Perception thread and all other threads are dependent on the controller module.

Planning thread takes the input from controller module which means it also accesses the shared memory and perception layer, localization layer and update the state utilities which are high level task achieving functions. Different high level states are already defined and world model module helps updating the expected world model taking input from planning layer by switching between these states.

The functionality high level states like follow ball state (robot approaches the ball and kicks it), supporting state (robot let other team member take the ball) are defined in state activity function. These states when activity are responsible for specific output behavior.

Each state contains low level task achieving function like kick, walk, stop, stand which are invoked from the module called action transaction. Each state using primitives from these action transactions achieve a high level task. Robot inverse kinematic solutions are used by these actions transactions to perform actions like kicking. For walk and move robot kinematics is used along with another high level module which is responsible for generating path trajectory using potential field and footstep planning.

Consideration is given to the variable handling since variable circulating in various modules is not very efficient, requires a lot of memory and makes the software more complex. Hence variables are written on shared memory and then accessed using the address but this method can also cause problem in case the variable is not written in the memory first and some other module tries to access it, this will cause naoqi to crash and the robot will become unresponsive dumping its main core.

2.4 Challenges

Some of the common challenges faced in RoboCup SPL which makes this problem more interesting are as follow.

2.4.1 Fully Autonomous Robots

The robots used in this competition are fully autonomous i-e- they make decision on their own without any outside human intervention. During the game play all communication between robots is also monitored. Robots make the team strategies and implement it in real time without any outside help.

2.4.2 Stability

Stability is an another concern especially during walk and kicking because available walk and movement control in naoqi are not stable enough so mostly teams have to come up with their own walk and kick engines . Moreover trend in SPL is also shifting toward Astroturf grounds hence stability is becoming more challenging.

2.4.3 Robot Safety

If robot stability is concern then so is robot safety. Nao robots are expensive and are only humanoid robots available in Pakistan. Because of the stability issues, errors and real time problems it is very possible that robot may fall in field, even though for safety naoqi provides functionality like fall manager but still over times hardware issues start arising if proper maintenance is ignored.

2.4.4 No overhead camera

Overhead camera is not available, each robot must use its local camera for perception. Hence the localization and cognition becomes more challenging because the environment is partially observable, at every time instant only specific part of information is available for the robots.

2.4.5 Monocular cameras on robot

The camera available on robots are monocular and does not provide stereo vision hence the depth perception is another problem. The information like distance of ball from robot local frame and distance of other landmarks and robots from local frame become more challenging to find out.

2.4.6 Constrained communication

The only communication allowed between the robots is through the central game controller which monitors all the communication. The communication is constrained and only allows limited data exchange under specific constraints. Each data packet is also logged and available to other teams after the match.

2.4.7 Cooperative and competitive environment

RoboCup SPL offers an environments in which a robot has to play in coordination with its other team players toward a common team objective and at the same time play against another team of robots. Hence the environment is cooperative and competitive at the same time which poses more rich and complex problems.

Chapter 3: Multi-Robot Cooperation

Multi-Robot cooperation implies multiple robots cooperating with each other for a common objective. Because of their wide range of applications and advantages robots are now being utilized widely and demand for the robotic solutions is increasing day by day. But with increase in problem complexity and tight application requirements it is now slowly becoming inefficient and even in some cases impossible for a single robot to meet the required criteria. Hence the trend starts shifting towards multiple robots collaborating with each other in order achieve the desired objective. Sometime the problem becomes too complex for a single robot to solve efficiently, multi-robot offer easy, fault tolerant and more flexible solution for the same problem and in some cases even more cost effective .Over the past decade research focused on multi-robot coordination is accelerated and many solutions and architecture are proposed for the problems and many new aspects of this fields are explored as well.

3.1 Task Allocation & Task Decomposition

The first question which arises in multi-robot collaboration is that how multiple robots decompose a global objective and divide the sub-objects within each other? And secondly in case of multiple objectives how they divide the individual between each other? The first question is the task decomposition problem which allows breakdown of the tasks into subtasks and then allocating each subtask to the specific robot or even require further dividing the task. The second question is the task allocation problem which dictates which task should be assigned to which robot and based on what performance measures. The task decomposition problem is more explored problem then task allocation and many solution form the field of computer science and artificial intelligence exist for this problem. The field of task allocation is less explored and more interesting. This thesis is focused on task allocation problem.

In the multi-robot cooperation planning literature the researchers have tried mainly two approaches for planning [1]

- *decompose-then-allocate* in which a single robot decomposes and break down the high level tasks in low level tasks and then distribute the low level doable tasks to all other robots in the system [2]
- *allocate-then-decompose* in which high level tasks are directly given to multiple robots and then each robot breaks down its task in low level doable tasks [3].

However the optimal planning in multi-robot systems is NP-hard problem [4]. NP-hard refers to non-deterministic polynomial time hard problems, these problems are at least as hard as the hardest problem in NP. NP problems are problems which cannot be solved using deterministic Turing machine in polynomial time but can be verified in polynomial time by a deterministic Turing machine. These problems are solved using non-deterministic Turing machine. All problems in NP can be reduced to NP-hard problem.

3.2 Multi-Robot Architectures

Multi-robot architecture is core and most important factor in multi-robot networks. These architectures are mainly divided in four categories [10].

3.2.1 Centralized Architectures

Centralized architectures in which one robot [11] is chosen as leader and is responsible for the communication between all other robots and also is responsible for task assigning and planning in whole network. The main features of these type of architectures are as follow

- Coordinated the entire team from a single point
- Application oriented
- Practically unrealistic
- Vulnerability to a single point failure
- Single point capability debate/ Computational constraints etc

3.2.2 Decentralized Architectures

Decentralized architectures in which each robot makes use of local information and makes its own decisions and actions Some example include [6], [7], [8], [9]. The main features of decentralized architectures are as follow.

- These architectures require robots to take action based only on knowledge local to their situation.
- Highly robust to failure.
- RoboCup SPL employs a decentralized architectures where each robot is expected to take actions based on their local information.

3.2.3 Hierarchical Architectures

Hierarchical architectures [5] in multi robots network are like hierarchical management in the organizations each robot manages are oversees its duties and delegate tasks accordingly. The main features of these type of architectures are as follow

- Every robot manages the actions of a group of robots and each robot in that group manages yet another group of robots and down to last robot in network which is simply doing its task.
- Weakness of these types of architectures is recovering from failure if robot in high control tree fails.

3.2.4 Hybrid Architectures

Hybrid architectures which is the mixture of high level control in the architectures and each robot's individual local control. The main features of this type of architecture can be defined as

- Local control + high level control
- Robustness + ability to influence entire team action through global goal
- Application oriented

3.3 Communication

Another important aspect in a group is robot is the means of communication, how the robots interact with each other. A taxonomy of communication structures in multi-robot systems is given in [5]. Inter robot interaction can be done through various techniques with their own advantages and disadvantages [12],[13] some of which are given in this section.

3.3.1 Cooperation-without-communication

In cooperation-without-communication technique the environment itself is a medium and there is no direct communication between the robots. This type of communication technique is also known as “stigmergy” and is mostly employed in centralized architectures [14], [15], [16]. SWARMS and other centralized architectures for homogeneous robots usually use this type of communication. Many researchers have worked on systems depending on this type of interaction which include [17], [18], [19].

3.3.2 Kin-recognition

In kin-recognition technique for communication robots can distinguish other robots in the environment and the other objects as well and interaction is accomplished after modelling many vision based solution exist in this domain [20], [21], [22]. This type of technique for communication come at the price of computation complexity and delays but is more robust than others [23].

3.3.3 Explicit-communication

In explicit-communication robots directly communicate with each through some communication medium [24], [25] this is the most widely used technique used so far and a lot of literature exist in this domain [26], [27], [28] but the challenges like communication delay, range and bandwidth has raised many interesting challenges in this domain [5]. This technique for communication is simplest and RoboCup SPL also uses this technique for the communication among robots.

3.4 Robot classification

In multi-robot coordination research domain robots are divided into two main categories, both of them are briefly discussed in this section.

3.4.1 Homogenous Robot Network

Homogenous robot network, which means all the robots in the system have the same capabilities like in RoboCup SPL in which all the robots are same, these system exhibit and emergent or non-intentional cooperation and are mostly used in swarm robotics. Hence in homogeneous robot network all robots in the network are identical, this could be because of the specific objective criteria or flexibility parameters because in these type of network each robot is capable of performing the task of any other robot and robots can also switch, swap tasks and also can overtake in case of failures. Hence these type of robot network are redundant and more fault tolerant but are expensive and more resource consuming.

3.4.2 Heterogeneous Robot Network

Heterogeneous, which includes different robots having different capabilities and each robot is allocated task that maximizes the performance measure. In these type of robot network each robot is different from others or at least some robots are different from the rest in term of capabilities and performance. The tasks in these networks are allocated based on the capabilities of these robots or according the objective or nature of the task. These type of networks are less fault tolerant than homogeneous robot network because some robots could be unique or some tasks could only be performed through specific robots in the network. The advantage of this type of network is that these networks are cheaper and consume less resources than homogenous robot network as redundancy is reduced. These type of robot networks are also more optimal comparing the resource utilization verses performance. These type of robot networks are more commonly used then homogenous robot networks. The networks employed for multi-robot coordination is also a dominant factor for the selection of multi-robot architectures. Some architectures are specific and can only be used by a specific robot network and some architecture are more flexible and incorporate both heterogeneous and homogenous robot networks.

The comparison of these architectures and systems are studies by many researchers and a lot of literature exist on these systems [29], [30].

3.5 Cooperation

Based on the robot network classification in the previous section the cooperation itself can be divided into two main categories [5].

3.5.1 Intentional cooperation

In this type of cooperation each robot in the network has the knowledge of presence of other robots in the network, each robot is also aware about the state of other robots and the capabilities and tasks they can perform. Hence all the robots in the network having this information about other robots act together to achieve a common global objective.

RoboCup SPL employs intentional cooperation in which each robot is aware of the presence of other robot, each robot know the current state and role of its teammate and having all this information they communicate explicitly for cooperation.

3.5.2 Non-intentional cooperation

In this type of cooperation each robot is not aware of the presence and capabilities of other robot in the network. This type of behaviors is known as emergent behavior and is mostly used in homogenous robot networks. Each robot uses its own local law to generate a global team plan then this plan is executed mostly using stigmergy. Collective swarm [32] is a good example of this type of cooperation. In nature, insects and bees mostly make use of this type of cooperation for the cooperation [31].

3.6 Task Allocation Approaches

A group of robots in which each robot have some specific capabilities or behaviors sets to achieve a certain task, of course each robot could have different way of performing same task. Now for the successful cooperation or coordination among these robot whenever a task is

introduced or generated, which robot should take this task and how this task will be assigned to some particular robot and who will assign some specific task to some specific robot is the problem of task allocation. There are two main task allocation approaches which are used for these type of problem in literature.

- Behavior Based Approach
- Market Based Approach

3.6.1 Behavior Based Approach

In classical artificial intelligence the robot brain is considered a serial machines. Data is collected from the environment using some sensors then it is processed and the cognition module make use of data and decide an actions from set of actions and give some output to the environment.

In contrast to this approach behavior based robotics small reactive responses are considered against the immediate states and then these small responses when combined exhibit the complex behavior of the robot [33]. The concept was coined by Rodney brooks in 1986. An augmented finite state machine proposed by Rodney brooks is given in Fig 18. The robot is then considered the combination of these atom finite state machines. It perform local computation and is easily mapped onto hardware. It does not require any central models, global clocks or memory bus. Each input of this finite state machine can be inhibited by another input. The finite state machine can be reset through reset port. The output of the finite state machine can be suppressed by another input. Hence the finite state machine takes input or this input can be override any time if provided through inhabitation. This machine simply takes input and maps an output accordingly just like a deterministic Turing machines. These finite state machines are then used in combination and their input and output connected in such a fashion that they exhibit even complex behaviors.

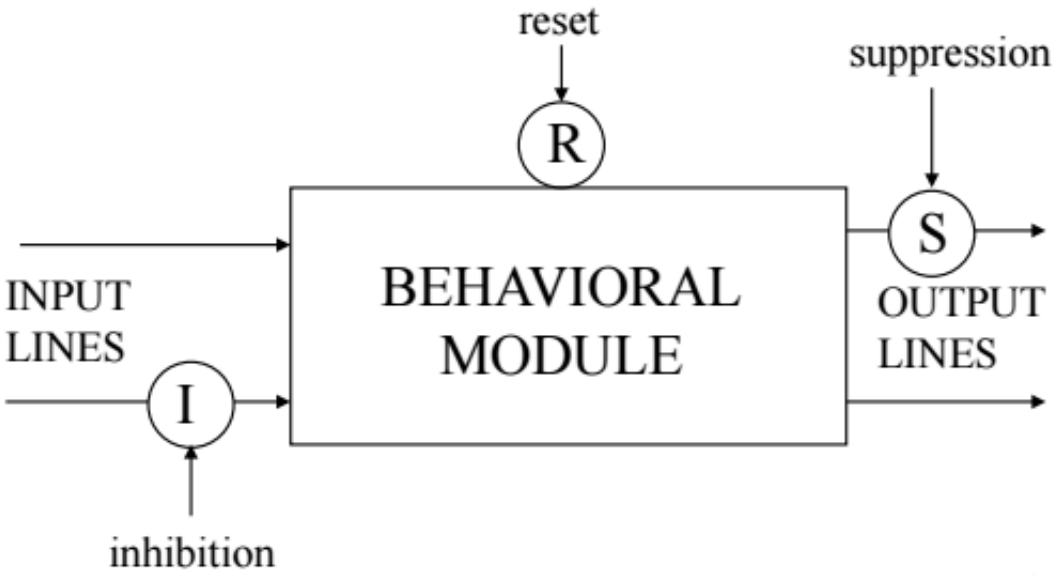

Figure 18: Augmented finite state machine

The behavior based approach for the task allocation allow the robots to adopt to a certain task without explicitly negotiating each task. This approach is more robust and provides the highest level of fault tolerance. The most well-known architecture which uses behavior-based approach is ALLIANCE [24] other architecture include [36], [37], [38]. This approach is although more fault tolerant and flexible but also difficult to implement and increases the computational cost.

3.6.2 Market Based Approach

Market Based approach for task allocation uses the concept of virtual economy. Just like a market where good are sold and every trader tries to maximize its profit by selling good. This concept is used to solve the task allocation problem and for a group of robots to coordinate together to complete a task, first introduced by Stentz and Diaz [39] using the contract net protocol by Smith [34].

This approach is a negotiation process unlike behavior based approach. The process begins when a task is generated or introduced in the system. This task is then negotiated among all the robot and the robot that best fit the task is assigned the job. First of all task is announced by a robot which is also called auctioneer. This robot tries to sell the task for the maximum

profit. As soon as the task is announced all the robots compute their cost for the given cost based on some metric. All the costs for the individual robots for the particular task including the robot auctioning the task submit their bid to the auctioneer. The auction is then closed and the results the evaluated. The robot with the lowest cost is expected to yield the maximum profit in the overall system hence the robot with lowest cost is then awarded the task. Some example of these architecture include [40], [41], [42].

This approach also corresponds to the bidding system where each party submit a bid and based on some heuristic a party wins the contract. This type of approach for coordination for task allocation is easy to implement but can take more time. This technique is also not very robust and fault tolerant. As if the robot awarded with the task fails in the middle of doing a task then other robots will not be aware of the situation for a certain time. Task renewal and task monitoring can be used to add some fault tolerance in this approach but these techniques doesn't respond fast when the fault is introduced in system. Moreover the cost computation itself is a complex problem as which parameters and how they will be incorporated according to the nature of the task.

Chapter 4: Research Methodology and Implementation

This chapter explains the research methodology for the design of a multi-robot coordination architecture on the task level and implementation of the proposed architecture for the multi-robot systems.

4.1 Problem Statement

Considering the debate from previous chapters and literature review it is evident that the domain of task allocation in multi-robot networks is not very well explored. Only a handful of architectures exist for this problem which address only a specific dimension of this problem. Hence need for a more robust, fault tolerant and efficient architecture exists. This research also addresses this need in the domain of multi-robot coordination by providing a more novel approach for the efficient solution for this problem.

4.2 Proposed Architecture

Proposed architecture exploits the advantages of both market based and behavior based architectures and proposes a hybrid architectures. The behavior based although more robust and fault tolerant but are very complicated and difficult to implement these architectures consume a lot of resources and increase the computational complexity. The marked based architecture although easy to implement and have less complexity then behavior based architectures but they are not robust and fault tolerant and also introduce unnecessary delays.

More over factors like cost calculation for the tasks for each robot cannot be standardized since nature of task varies from each other. The cost calculation itself can be a complex problem and also introduces delays in decision making. This problem of calculating cost for a specific task for each task is also referred as metric evaluation [40]. Mostly researchers use single parameter for the metric evaluation for instance euclidean distance, time etc when proposing an architecture like in [41], [42]. But a single parameter for metric evaluation can only address this problem in very limited way. Even if multiple parameters are used how each parameter should be assigned weight according to the nature of task and which parameters to use in first place, is a still a research problem. Moreover the metric is calculated by individual robot based on the capabilities of that robot hence the metric evaluation is robot capability oriented but this research

also takes account of the nature of task. Hence to address these problems instead of using a single parameter for metric evaluation a vector of cost is used each component of this vector addressing a parameter for the metric evaluation and then overall cost relative to a specific task is calculated using Analytical Hierarchical Process (AHP) in which (1) First according to the nature of task a criteria vector is obtained which is task oriented, each component of criteria vector corresponds to the task or criteria weight which is calculated based on the nature of the task. (2) Secondly each option/robot in the network is evaluated based on the defined criterions and a score matrix is calculated against all the options in the network (3) in the end options are ranked considering both criteria vector and score matrix of all options.

4.3 Analytical Hierarchical Process (AHP)

In the proposed architecture AHP [43] is used for selecting the best candidate for the task which considers both the nature of the given task and the capabilities of each robot in network and the best robot is assigned the task. The evaluation criteria are the mission objectives which are output of task planning or can be injected in the system.

Analytical Hierarchical Process (AHP) starts with creating a pairwise comparison matrix. In pairwise comparison matrix the weight for the each criterion is calculated based on the importance of each criterion according to the nature of the task.

- Let **A** be the pairwise comparison matrix.
- This matrix **A** is the $\boldsymbol{m \times m}$ real matrix, where m is the number of evaluation criteria which can be adjusted according the task requirements.
- The entry $\boldsymbol{a_{jk}}$ of the matrix **A** represents the importance of the *j*th criterion relative to the *k*th criterion.
- If $\boldsymbol{a_{jk} > 1}$, then it means that *j*th criterion is more important than the *k*th criterion and vice versa.
- If two evaluation criteria have the equal importance which is possibly considering the nature of task, then the entry against them $\boldsymbol{a_{jk}}$ is 1.
- The entries $\boldsymbol{a_{jk}}$ and $\boldsymbol{a_{kj}}$ of the matrix **A** satisfy the following constraints:

$$a_{jk}.a_{kj} = 1$$
$$a_{jj} = 1$$

Computing vector of criteria weights

The pairwise comparison matrix **A**

$$A = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mm} \end{pmatrix}$$

First normalizing the matrix, each entry is normalized column wise in **A**

$$\bar{a}_{ij} = \frac{a_{ij}}{\sum_{i=1}^{m} a_{ij}}$$

Row wise weighted sum is calculated from this normalized matrix

$$w_i = \frac{\sum_{j=1}^{m} \bar{a}_{ij}}{m}$$

Criteria vector w is calculated

$$w = [w_1 \, w_2 \, \dots w_m]$$

This criteria vector dictates the weight against the each criteria of the task, in other words this is the task definition which needs assignment from a robot.

Next we calculate the matrix of option scores or in the other words the fitness of each robot according to the defined parameters.

Computing the matrix of option scores

$\boldsymbol{B_j}$ is a $\boldsymbol{n \times n}$ real matrix, where n is the total number of option or robots being evaluated

$$B^j = \begin{pmatrix} b_{11}^j & \dots & b_{1n}^j \\ \vdots & \ddots & \vdots \\ b_{n1}^j & \cdots & b_{nn}^j \end{pmatrix}$$

$\otimes$ *b_{ih}^j represents the evaluation of i_{th} option relative to h_{th} option with respect to j_{th} criterion*

$\otimes$ *each entry in B^j satisfy all the constraints as in pairwise comparison matrix A*

$\otimes$ *similarly as in computing pairwise comparison matrix A, we first normalize each entry of B^j then compute the weighted sum.*

$\otimes$ *let wt^j be the weight vector of options scores relative to the j_{th} criterion*

finally the score matrix $\boldsymbol{S}$ *is obtained as*

$$S = [wt^1 \, wt^2 \, \dots wt^m]$$

Ranking the options

Finally each option/robot is ranked according to the fitness for the given task by multiplying the vector of criterion weights with the matrix of options scores. The output matrix v represents the percentage fitness of each robot for a given task

$$v = S \,.\, w$$

The i_{th} entry v_i of v represents the global score assigned by the AHP to the i_{th} option.

4.3.1 Checking the consistency

When assigning the priorities of each option for the evaluation against any given parameter or assigning the priorities of parameters against a specific task the inconsistency may arise. The inconsistency arise due to the illogical weight assignment. For instance let's consider, Ali says he likes banana 5 times more than orange and he likes orange 2 times more than apple. A person can conclude Ali likes banana 10 times more than apple. Ali's statement that he like apple 3 times more than banana is highly inconsistent. While the statement that Ali likes banana 10 times more than apple is perfectly consistent.

Hence against every pairwise comparison matrix consistency index is obtained if the consistency is within limits then the matrix is considered acceptable otherwise it is discarded because the inconsistent comparisons can yield into to false decision. Inconsistency is checked for both pairwise comparison matrix and the matrix of options scores. A task with inconsistent criteria is discarded before any further actions.

The Consistency Index (CI) is obtained by first computing the scalar x, which is calculated first by taking the column wise sum of **A** then by multiplying this row matrix with column matrix **w** and taking the average, Then

$$CI = \frac{x - m}{m - 1}$$

$$\frac{CI}{RI} \leq 0.1$$

RI random index is standard and is defined against number of parameters

Table 1: Value of random index (RI) for small problems

m	2	3	4	5	6	7	8	9	10
RI	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.51

4.4 Example Problem

Let's consider a problem and evaluate the options as an example for the AHP for the elaboration of the effectiveness of the process. Consider three robots at random places and the task is introduced for any robot to reach a certain point called Goal as in Fig 19.

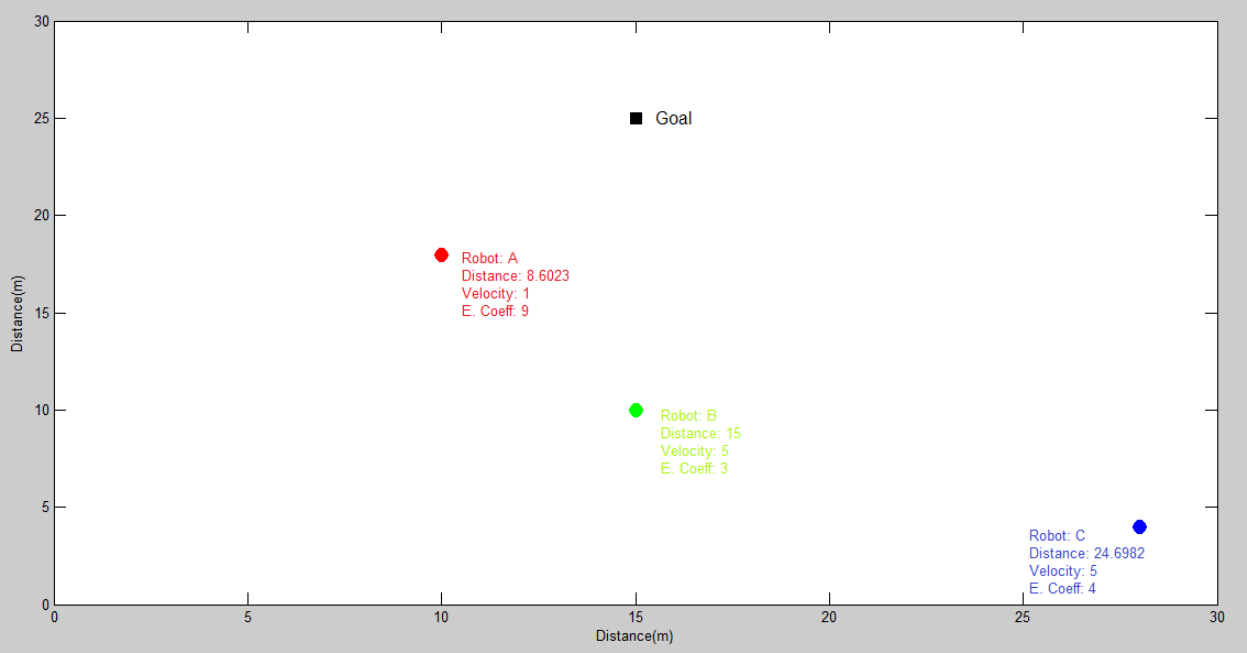

Figure 19: Example Problem

For this particular task to reach a certain destination we define the nature of task. Three parameters are define for this problem but any number of parameters can be selected based on the application and complexity of task only the dimension of matrix will increase and this will of course increase the computational complexity as well.

The three parameters which are selected are Time, Energy and Distance. For this given task we are more concerned with time. Hence we define the priority of time 5 times more than energy and 9 times more than distance. We are least concerned with distance parameter and after

time we are more concerned with the energy so we set the priority of energy 3 times more than distance. These priorities are the nature of the task and vary according to the task. These priority can be injected by a human or can be output from the task planning layer.

We first generate the pair-wise comparison matrix **A** which is associated with the nature of the task. As we can see in Table 2 according to the definition of task the calculated weightage of time parameter is 74.82% and then Energy 18.04% and Distance 7.14%. These weights are calculated by AHP according the priorities defined ahead. The pairwise comparison matrix A is also consistence with index ratio less than 0.1.

Table 2: Pair-wise comparison matrix

Pair-wise	Time	Energy	Distance	**wi**
Time	1	5	9	0.7482
Energy	0.2	1	3	0.1804
Distance	0.111	0.333	1	0.0714
Sum	1.311	6.333	13	x

$$A = \begin{pmatrix} 1 & 2 & 8 \\ 1/2 & 1 & 4 \\ 1/8 & 1/4 & 1 \end{pmatrix}$$

Secondly the matrix of option scores B_j are generated against with respect to each parameter. Each robot just respond to the robot auctioning the task with the cost of vectors. The auctioneer robot receives these vector of costs and evaluate the each option relative to the each parameter and finds out the most suitable candidate for the task.

For our proposed problem let's consider Robot C is 9 times faster than Robot A and 7 times faster than Robot B. And Robot B is assumed 3 times faster than Robot A. Now these priorities depend on the capability of each robot the fitness of each option is evaluated with respect to the given parameter. In Table 3 options are evaluated relative to the Time parameter. The matrix is also consistent.

Table 3: Option score matrix relative to time parameter

Time	Robot A	Robot B	Robot C	**wt i**
Robot A	1	0.333	0.111	0.0685
Robot B	3	1	0.143	0.1549
Robot C	9	7	1	0.7766
Sum	13	8.333	11	x

The above matrix tells us that relative to the time parameter the fitness of robot A is 6.85% and the fitness of Robot B is 15.49%. The Robot C is fastest and consumes the minimum time has the fitness of 77.66%. These fitness levels strictly depend on the capability of each robot and are fixed.

Now each option is evaluated again with respect to the Energy parameter given in Table. 4. The matrix is also consistent. Here it is stated again that each robot only broadcasts its cost against each parameter the evaluation of all the options against any parameter is performed by auctioneer robot only.

Table 4: Option score matrix relative to Energy parameter

Energy	Robot A	Robot B	Robot C	**wt i**
Robot A	1	0.333	5	0.2828
Robot B	3	1	7	0.6434
Robot C	0.2	0.148	1	0.0738
Sum	7	1.533	5	x

Here it can be observed that Robot C is most energy efficient and Robot C is least energy efficient these parameters are assumed for the elaboration of the process.

All robots are evaluated again relative to the third parameter, distance in Table 5. The Robot A is closest to the goal and Robot C is considered farthest from the goal. The option score matrix in Table. 5 is also consistent.

Table 5: Option score matrix relative to distance parameter

Distance	Robot A	Robot B	Robot C	**wt i**
Robot A	1	5	9	0.7482
Robot B	0.2	1	3	0.1804
Robot C	0.111	0.333	1	0.0714
Sum	1.311	6.333	13	x

Now score matrix is calculated after evaluating all the options relative to all the parameters as defined in nature of task. The score matrix S in our case turn to be

$$S = \begin{pmatrix} 0.0685 & 0.2828 & 0.7482 \\ 0.1549 & 0.6434 & 0.1804 \\ 0.7766 & 0.0738 & 0.0714 \end{pmatrix}$$

In the end obtaining the global score of each robot relative to task assumed in the example.

$$v = S.w$$

$$v = \begin{bmatrix} 0.1557 \\ 0.2448 \\ 0.6453 \end{bmatrix}$$

From the final rank matrix we observe that for our assumed example the best candidate is Robot C with maximum task fitness of 64.53% and Robot A is least fit for this task with task fitness of 15.57%.

In our proposed example using AHP Robot C was most fit for the task. Robot C is even though farthest robot from the goal with respect to distance and also Robot C is the least efficient robot but Robot C is the fastest robot and according to the nature of task the time had maximum priority hence Robot C yielded maximum score through AHP. In the same problem if had the flexibility in time and then wanted to conserve the energy of the system we see Robot B is the best candidate through AHP. Hence this approach using vector of cost to calculate fitness of each robot according to those parameter, then selecting the robot according to its capabilities and nature of the task yields better output.

Using single parameter for cost or changing the parameters again and again according to the nature of task is not feasible as practiced in earlier literature. For instance if we only use Euclidean distance as a parameter then the robot closes to the goal will always be selected for the task. In the real world application and complex environments the nature of task is dynamic and the architecture should be flexible enough to incorporate these changes. Moreover even if cost vectors are used how the architecture should evaluate each option based on the task definition, AHP provides an efficient answer to this question

4.5 Setting the priorities

The importance of parameters with respect to each other are mission objectives this can be a part of task planning or manually injected. The importance of evaluated options with respect to a given parameter is calculated based on cost of all the individual option for that particular parameter. These priorities are adjusted by the auctioneer based on the cost vector received from each robot.

The pseudo code for the setting these priorities is given below. The lookup table used in this pseudo code is given in Table 6. The priorities are set in odd number from 1 to 9, even number can also be used for the fuzzy logic. Lookup table is not the best solution for setting the priorities, a learning algorithm looks like a more attractive option but not used and is recommended in future works.

Setting the Priorities

Let C_{ij} be the cost jth parameter of option i
for all j
    for i = 1 to n-1
        score = $\frac{C_{ij}}{C_{(i+1)j}}$
        if score is more then 1 then goto lookup, cont;
        else swap i,j and goto lookup, continue;
        end
        goto lookup
    end for
end for

Lookup Table

Table 6: Lookup Table for priorities

Entry	C_{ij}	b_{ih}^{j}
1	1	1
2	1-3	3
3	3-5	5
4	5-7	7
5	>7	9

$h = i + 1$

4.6 Pseudo Code for the bidder and auctioneer

This section includes the Pseudo code for the bidder and auctioneer in the architecture. The marginal costs are used for stacking the tasks for individual robot.

Pseudo Code for Auctioneer

```
if there is task in task list then announce task
     while time is running do
          receive vector of cost bids
     end while
     calculate the best option based on priorities using AHP
     if the best option is bigger then auctioneer then
          send task to bidder
     end if
     delete task from announcement list, add in monitoring list
end if
if there is a task in monitoring list then get acknowledgement
     if acknowledgement not received against task in time t
          add task in announcement list, delete from monitoring
          list
     if task completion flag received against task
          delete that task from monitoring list
end if
```

Pseudo Code for Bidder

```
if message is task announcement then
     calculate optimal position of task in local plan
     calculate vector of cost bids (marginal costs)
     send bid to auctioneer
else if message is task award then
     insert task in local plan in position calculated before
     send acknowledge to auctioneer periodically.
     introduce task in announcement list
end if
```

4.7 Capability Matrices

Capability matrices are introduced to improve decision process and makes the process more fault tolerant. It provides the status of behavior set available during any time instance t. This process is used as pre filtering in the bidding process and improves the performance of the architecture. For instance consider a robot ‘i’ due to some physical fault it is unable to move but being the part of the network this robot always participates in bids and also have the possibility of winning the task. The task can later be opted by some other robot but these small real challenges introduce a lot of delay. Hence before the bidding a pre filtering process for each robot which provide them with the information weather they have require behavior set for achieving the task at any time increases the efficiency of the architecture.

$$\begin{pmatrix} C_{i1} & 0 & \cdots & 0 \\ 0 & C_{i2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_{im} \end{pmatrix} \begin{pmatrix} f_{i1} \\ f_{i2} \\ \vdots \\ f_{im} \end{pmatrix} = \begin{pmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{im} \end{pmatrix}$$

- C_{im} *is the* m_{th} *capability of* i_{th} *robot*
- f_{im} *is the feedback of* m_{th} *capability for* i_{th} *robot*
- b_{im} *is the availability of* m_{th} *capability for* i_{th} *robot*
- b_{im} provides the status of available behavior sets.
- f_{im} could be a threshold function giving status 1 or 0 of b_{im} , if implemented on hardware level it could be a continuous function giving the current status level of capability.

4.8 Behavior Base Model Integration

A task oriented behavior based model in integrated in the architecture to add another layer for fault tolerance since real world applications can pose many challenging scenarios for the architecture hence the multi-robot architecture should be flexible and more tolerant to incorporate different real world applications like RoboCup.

In real world application like RoboCup, consider a robot is awarded a task and during the execution of the task the robot falls or fails or due to some other problem the task is being delayed. One solution for this problem that exist in literature is task monitoring but task monitoring also introduces delay in case the fault occurs and the bidding process starts again. AHP although announces the best candidate for the task but a flexible system is still required in case the robot with task award fails to perform its task which is very common in applications like RoboCup.

Consider ‘n’ number of robots

$$R = \{r_1, r_2 \cdots r_n\}$$

Consider ‘m’ number of independent task for robot ‘r_i’

$$T_i = \{t_1, t_2 \cdots t_m\}$$

Let T_{ij} *be robot* $'r_i'$ *working on task* $'t_j'$

The behavior based model is task oriented and is associated with each task T_{ij}

4.8.1 Inter-robot communication

While monitoring the communication messages, robots i must also incorporate when another team member is pursing task ti

$$\text{inter\_com}(i,k,j,t_1,t_2)=\begin{cases}1 & \text{if robot } r_i \text{ has recieved message from robot } r_k \\ & \text{concerning task } t_i \text{ in the time span}(t_1,t_2);\ t_1 < t_2 \\ 0 & \text{otherwise}\end{cases}$$

4.8.2 Task Suppression

Task tj is being suppressed at time t on robot ri if some other task tk is currently active on robot ri at time t

$$task\_\sup_{ij}(t)=\begin{cases}0 & \text{if another task } t_k \text{ is active for robot } r_i \text{ at time } t \\ & \text{where } k \neq j \\ 1 & \text{otherwise}\end{cases}$$

4.8.3 Impatience

Three factors are considered in when calculating impatience. The impatience is the motivation of a robot at a certain time to quit a task. The fast and slow increase in impatience could be a linear function. The impatience is fast if a certain task active more than the time robot wants to maintain it and some other robot wants to purse it and haven't ceased to function or the robot wants to give up the task and try to possibility of another.

α_{ij} = *time that robot* r_i *wants to maintain task* t_j *before yielding this task to another robot*

β_{ij} = *time that robot* r_i *wants to maintain task* t_j *before giving up possibility to try another task*

γ_i = *time that robot* r_i *allows to pass without recieving a communication message from a specific team memeber before deciding that the team - mate has ceased to function*

$$impatience_{ij}(t) = \begin{cases} fast_{ij}(t) & \text{if task } t_j \text{ is active for more than } \alpha_{ij} \text{ time units and} \\ & \exists x.inter\_com(i,x,j,t-\gamma_i,t) = 1; \\ & \text{OR} \\ & \text{task } t \text{ has been active for more than } \beta_{ij} \text{ time unit at } t \\ slow_{ij}(t) & \text{otherwise} \end{cases}$$

4.8.4 Subsumption

$$S_{ij}(t) = \text{subsumption of task } j \text{ on robot } i \text{ at time } t$$
$$S_{ij}(0) = 0$$
$$S_{ij}(t) = \left[S_{ij}(t-1) + impatience_{ij}(t) \right] \times task\_\sup_{ij}(t)$$

- Subsumption at any time t subsumes or overcomes the task rendering it to cease functioning if the value of Subsumption either increases the threshold value or is equal to zero, each task has its own subsumption level.
- Subsumption keeps increasing using linear function and can be reset through reset_subsumption function
- Subsumption at time 0 or after reset is 0 and starts incrementing with impatience. If some other task is currently active on robot then robot subsumes all task other than that particular task by forcing subsumption to zero. When the particular task is active its suppression becomes 1. The task subsumption now depends on impatience only.

4.8.5 Subsumption reset function

It causes the impatience to be set to zero if robot ri has just received its first message from robot rk indicating that rk is performing task tj.

$$subsumption\_reset_{ij}(t) = \begin{cases} S_{ij}(t) = 0 & \text{if } \exists k.inter\_com(i,k,j,t-\lambda,t) = 1 \text{ and} \\ & inter\_com(i,k,j,0,t-\lambda) = 0; \\ & \text{where } \lambda = \text{time since last comm check} \\ S_{ij}(t) = S_{ij}(t) & \text{otherwise} \end{cases}$$

4.9 Architecture Overview

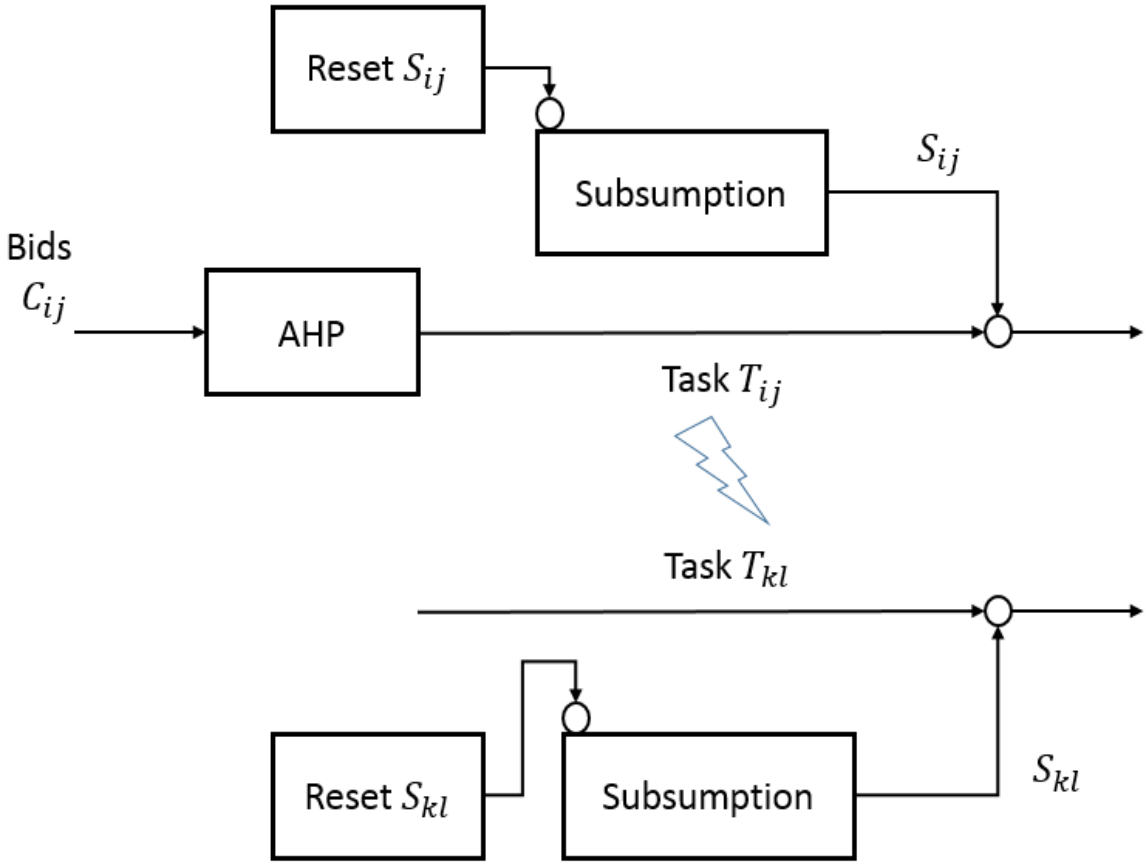

Figure 20: Propose Architecture overview

An overview of the proposed architecture is given in Fig. 20. Each task active on any robot in the system is first evaluated by auctioneer using AHP then assigned to the robot and monitored as well. At any time t a robot could have multiple task executing them in specific

order. Each task on every robot can be subsumed at any time if the subsumption level increases the threshold value or is equal to 0. At any time a single robot cannot have more than one active task.

When a task is active on a robot the impatience associated with that particular task starts increasing, this impatience also keep increasing the subsumption. If the value of subsumption increases a threshold the task is killed. The submsumption is reset if some new robot announces that it is doing that task. The task is also killed if subsumption is zero because zero subsumption means another task currently active on the robot.

Chapter 5: Results and Conclusion

The proposed architecture is also proved using simulations with different scenarios and parameters. This chapter contains the results from those simulations, conclusion of the research and the future work suggestions.

5.1 Simulation 1

Consider the scenario as in Fig. 19. Three Robot A, B & C at fixed distance from goal have fixed parameter like robot velocity and energy loss coefficients. The parameters distance from goal, robot maximum velocity and energy loss factor is given in Fig 19. These parameters are fixed for each robot. A task is introduced to reach at the goal. Task is introduced manually and time is given maximum preference because we want task to be accomplish in minimum time after time, energy is preferred over distance but time is not given too much preference over energy and we are not very concerned with the distance factor. The task demands the completion in which time is given double importance over energy and is considered 8 times more important than distance factor. Energy has significant importance and is preferred 4 times more important than distance.

Analyzing the given situation first less analyze this situation by evaluating the parameters for all robots simultaneously. Fig. 21 shows each robot trying to complete the task and is terminated when the task is achieved i-e a robot has reached the distance. Since robot A has the minimum velocity hence even being closest to the goal it haven't arrived the goal yet. Robot B and robot C have the same velocity but since robot B is closer to the goal hence it arrived the goal first.

Fig. 22 shows the each parameter Time left, Energy loss and Distance after the simulation was terminated. Robot B have arrived the destination hence its time left and distance has converged to zero. The Robot A has minimum energy loss but it is still far from goal and might consume more energy in total. Robot C is not very energy efficient and has already spend a lot of energy trying to achieve the task. According to the nature of task as defined earlier Robot B was selected for this task with the task fitness of 45.6%, Robot A has the task fitness of 41.8% which is very close to task and will not consume more than Robot C in total so it was preferred over

Robot C considering all the factors as in the simulation. Robot A and Robot B doesn't have very significant difference in fitness level with respect to task.

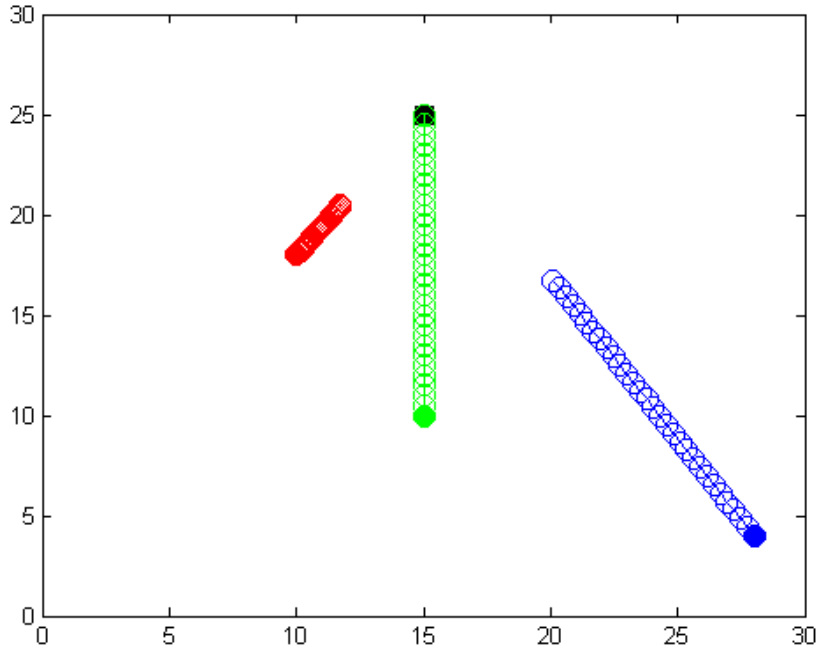

Figure 21: Simulation 1 - Each robot completing the task

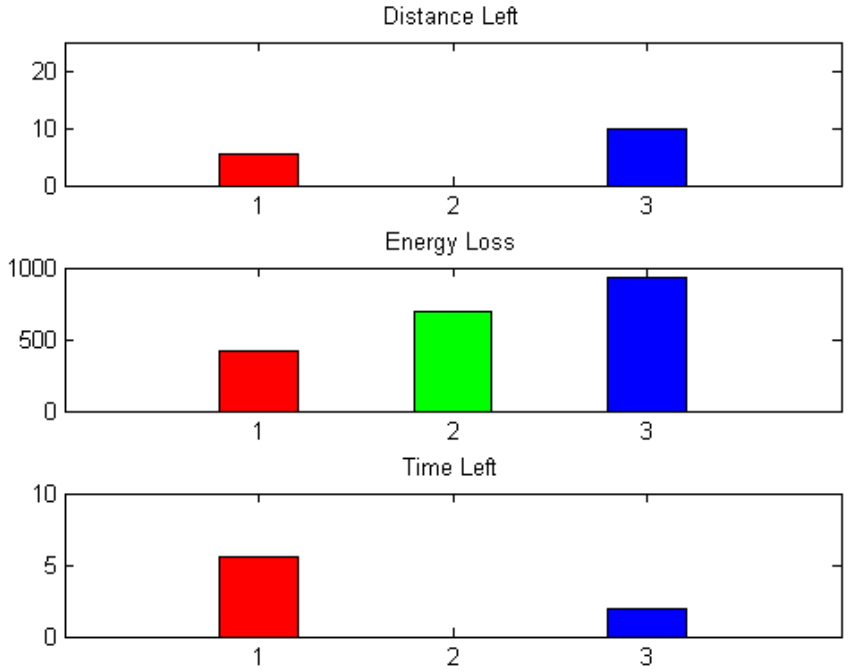

Figure 22: Simulation 1 - Parameters after simulation was terminated at task completion

The cost vector of each robot was given input to the AHP and according the nature of task and evaluating all the cost for each robot against each parameter Robot B (Green) was assigned the task and as seen in Fig. 23 Robot B (Green) is performing the task of going to goal position. This is a simple example which is designed to elaborate and validate the working of the architecture. Behavior based model is also integrated in this simulation although it is more effective on the real applications. An error was introduced in Robot B after some time and as expected Robot A (Red) adopted the task after some time as shown in Fig. 24 as it was the second best candidate according to the AHP evaluation.

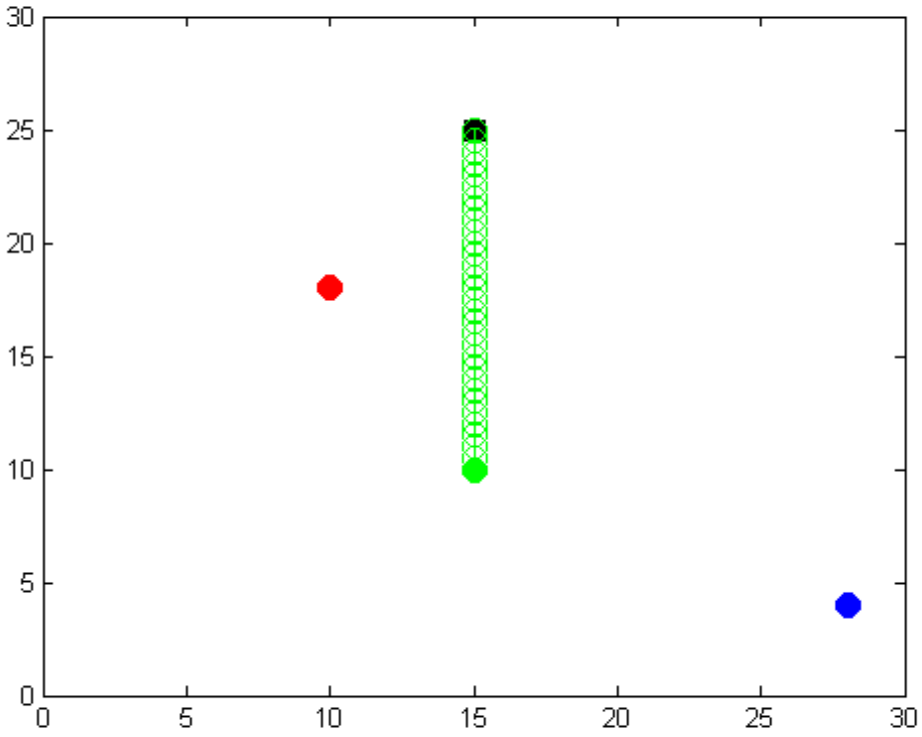

Figure 23: Simulation 1 - Robot B (green) assigned the task

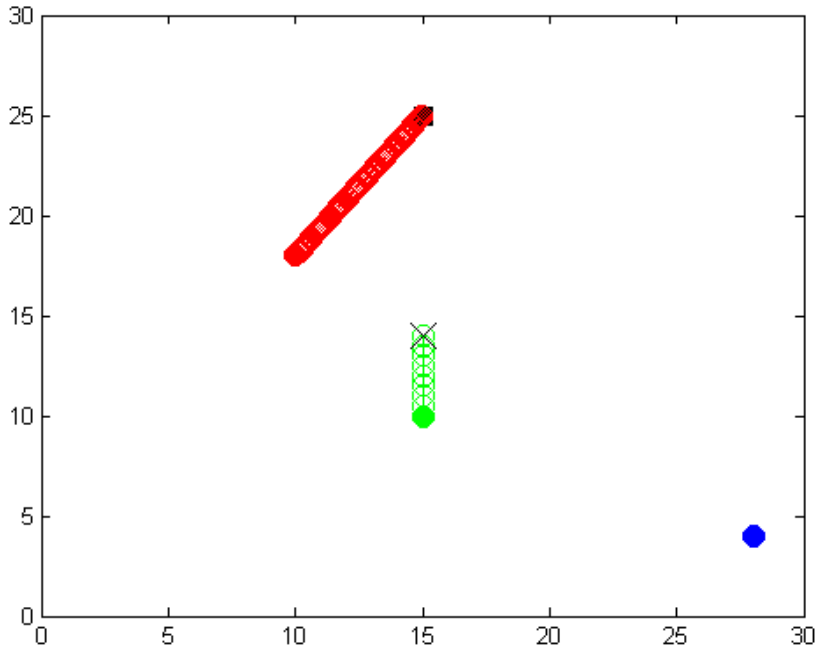

Figure 24: Simulation 1 - Robot A (Red) assigned the task when Robot B (Green) failed

5.2 Simulation 2

Another situation was simulated with same scenario but nature of task and capabilities of robots were changed. For this simulation the parameter for each robot were as follow.

Robot A (Red)

Distance	8.6023
Loss-factor	9
Velocity	1

Robot B (Green)

Distance	15
Loss-factor	3
Velocity	5

Robot C (Blue)

Distance	24.6982
Loss-factor	4
Velocity	15

Nature of Task

The nature of task introduced dictates that time is given extreme importance it is preferred 5 times over energy and 9 times over distance. Energy is preferred 3 times over distance. Distance is has the least importance because the task doesn't care about the distance as long as it is completed in minimal time.

From the robot capabilities it can be seen the Robot C is the fastest robot with the velocity of 15m/s but the Robot B is more efficient as its loss factor over time is 3. This loss factor is constant. Robot A is closest to the goal with the minimum distance. Under these capabilities and nature of task we first analyze the situation by evaluating all the parameters of robots with respect to the task which is given in Fig. 25 shows all the robots trying to complete the task and the simulation was terminated as the task was achieved. Robot C finished the task in minimum time as it has the maximum velocity.

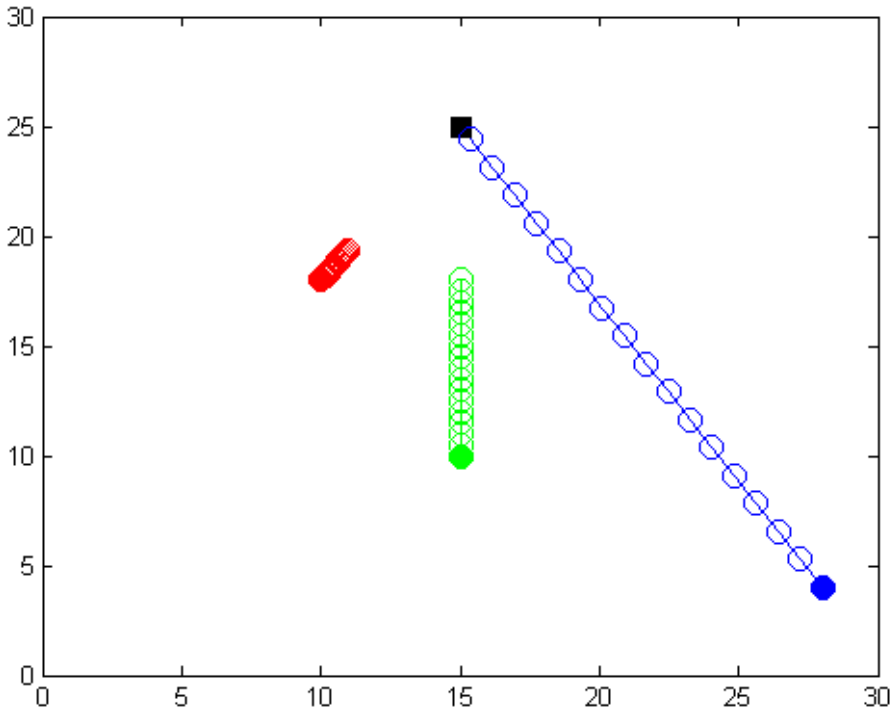

Figure 25:: Simulation 2 - Each robot completing the task

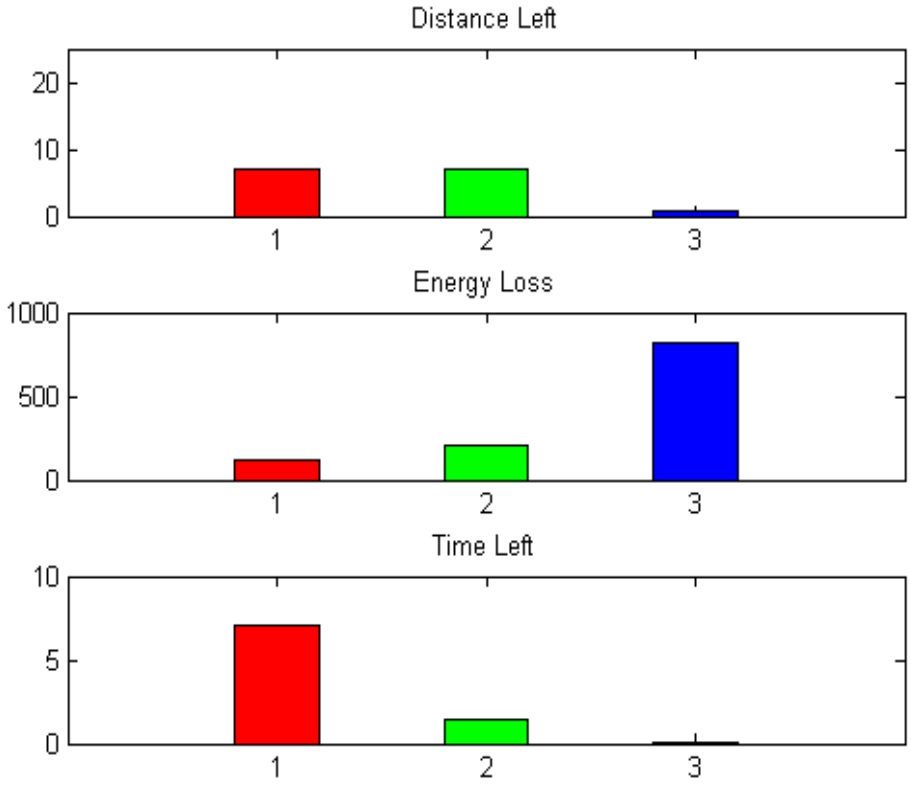

Figure 26: Simulation 2 - Parameters after simulation was terminated at task completion

Fig. 26 shows the each parameter Time left, Energy loss and Distance after the simulation was terminated of the simulation 2. Robot C have arrived the destination hence its time left and distance has converged to zero. The Robot A is slowest hence the most time is left for this robot. According to the nature of task and the capabilities of robots as defined earlier Robot C was selected for this task with the task fitness of 59.74%, Robot B has the task fitness of 20.86% and Robot C has task fitness of 13.46%. These result vary significantly from previous results with Robot A at one side of spectrum and Robot C at the other. All the options have the significant difference in fitness scores from each other. In Fig. 27 it can be seen that Robot C has been awarded with the task.

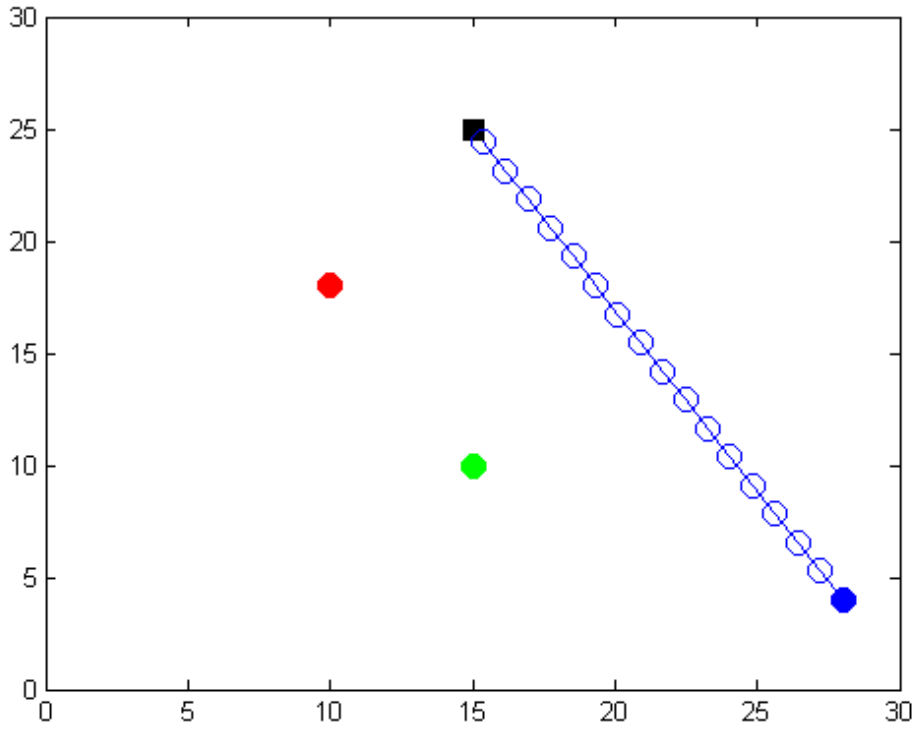

Figure 27: Simulation 2 - Robot C (blue) assigned the task

Each robot can be auctioneer or bidder at the same with the pseudo code as mentioned in previous chapter. The behavioral model is integrated with each robot but we cannot exploit all of its advantage in simulation results. The architecture proposed in this research proved to be efficient and more robust compared to the other architectures which either used single metric or doesn't cater fall the nature of task and each capability of the robot. The designer doesn't have to redefine the metric again and again according to each scenario or redefine the criteria's.

5.3 Conclusion

- Team-NUST was the first team from South Asia to qualify for RoboCup SPL. The team competed in RoboCup SPL 2016 in Leipzig Germany.
- RoboCup project is a continuous improving process and is refined and upgraded a lot in previous year as explained in the first and second chapter. Each module is upgraded and

enhanced, data passing among the functions is optimized and a reasonable gameplay was introduced for RoboCup SPL 2016 by Team-NUST.

- This research proposes a robust hybrid architecture for multi-robot cooperation that exploits the advantages of both market based and behavior based architectures. The architecture inherits the simplicity of implementation and reduced computation overhead from market based architectures, increases the robustness and adds more flexibility at same time by integrating a behavioral layer at individual task level.
- Instead of using a single parameter for metric evaluation each robot broadcasts a vector of cost for a given task, each component of this vector addressing a parameter for the metric evaluation and then overall cost relative to a specific task is calculated using Analytical Hierarchical Process (AHP). The process not only incorporate each capability of robot but also incorporate the nature of task as well. Multiple cost evaluation through AHP for task allocation problem as proposed in this research provides a new way of looking at this problem.
- A task oriented behavior based model in integrated in the architecture to add another layer for fault tolerance since real world applications can pose many challenging scenarios for the architecture like robot failure or failing to accomplish a task due to uncertainty in environment hence the multi-robot architecture should be flexible and more tolerant to incorporate different real world applications like RoboCup.
- Proposed model is proved mathematically, elaborated through examples and validated through simulations. The architecture is not fully deployed for RoboCup SPL yet by Team-NUST.

5.4 Future Work Suggestions

- The architecture proposed in this research although validated through simulations and proved mathematically but is aimed for practical application of multi-robot coordination. The implementation of architecture with actual hardware platforms is still under process hence the implementation of this architecture on various applications for performance

evaluation under different challenging real life application is a good step to take this research forward.

- The importance of evaluated options with respect to a given parameter is calculated based on cost of all the individual option for that particular parameter and the nature of the task. Setting the priorities as in Section 4.5 is an acceptable method. But setting the priorities and setting the parameters for behavior based layer assisted with machine learning can yield in more efficient output.
- The integration of behavior based layer at individual task although introduces more flexibility and fault tolerance in architecture but at the same time adds the computational complexity hence instead of task oriented behavior based layer another layer maybe considered addressing only individual options.
- This research addresses the task allocation problem by providing a standard multi-robot coordination architecture but a more rich architecture which also includes task planning and definition of nature of each task may also be considered to move this research further for the hunt of a more diverse architecture.

REFERENCES

[1] Mosteo, Alejandro R., and Luis Montano. "A survey of multi-robot task allocation." *Instituto de Investigacin en Ingenierła de Aragn (I3A), Tech. Rep* (2010).

[2] Caloud, Philippe, et al. "Indoor automation with many mobile robots." *Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on*. IEEE, 1990.

[3] Botelho, Sylvia C., and Rachid Alami. "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement." *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 2. IEEE, 1999.

[4] Gerkey, Brian P., and Maja J. Matarić. "A formal analysis and taxonomy of task allocation in multi-robot systems." *The International Journal of Robotics Research* 23.9 (2004): 939-954.

[5] Cao, Y. Uny, Alex S. Fukunaga, and Andrew Kahng. "Cooperative mobile robotics: Antecedents and directions." *Autonomous robots* 4.1 (1997): 7-27

[6] Arkin, Ronald C. "Cooperation without communication: Multiagent schema-based robot navigation." *Journal of Robotic Systems* 9.3 (1992): 351-364.

[7] Beckers, R., O. E. Holland, and Jean-Louis Deneubourg. "From local actions to global tasks: Stigmergy and collective robotics." *Artificial life IV*. Vol. 181. 1994.

[8] Maja, J. *Mat aric. Interaction and Intelligent Behavior*. Diss. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1994.

[9] Steels, Luc. "A case study in the behavior-oriented design of autonomous agents." (1994).

[10] Parker, Lynne E. "Multiple mobile robot systems." *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008. 921-941.

[11] Milutinovi, Dejan, and Pedro Lima. "Modeling and optimal centralized control of a large-size robotic population." *IEEE Transactions on Robotics* 22.6 (2006): 1280-1285.

[12] Parker, Lynne E. "The effect of action recognition and robot awareness in cooperative robotic teams." *Intelligent Robots and Systems 95.'Human Robot Interaction and*

Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on. Vol. 1. IEEE, 1995.

[13] Dudek, Gregory, et al. "A taxonomy for swarm robots." *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*. Vol. 1. IEEE, 1993.

[14] Matarić, Maja J. "Issues and approaches in the design of collective autonomous agents." *Robotics and autonomous systems* 16.2 (1995): 321-331.

[15] Deneubourg, J., et al. "Self-organizing collection and transport of objects in unpredictable environments." *Japan-USA Symposium on Flexible Automation*. 1990.

[16] Kube, C. Ronald, and Hong Zhang. "Collective robotics: From social insects to robots." *Adaptive behavior* 2.2 (1993): 189-218.

[17] Beckers, R., O. E. Holland, and Jean-Louis Deneubourg. "From local actions to global tasks: Stigmergy and collective robotics." *Artificial life IV*. Vol. 181. 1994.

[18] Onn, Shmuel, and Moshe Tennenholtz. "Determination of social laws for multi-agent mobilization." *Artificial Intelligence* 95.1 (1997): 155-167

[19] Werger, Barry Brian. "Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams." *Artificial Intelligence* 110.2 (1999): 293-320.

[20] Kuniyoshi, Yasuo, et al. "Cooperation by observation: the framework and basic task patterns." *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994.

[21] Kuniyoshi, Yasuo, et al. "Vision-based behaviors for multi-robot cooperation." *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*. Vol. 2. IEEE, 1994.

[22] Sugie, Hiroshi, et al. "Placing objects with multiple mobile robots-mutual help using intention inference." *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*. Vol. 2. IEEE, 1995.

[23] Huber, Marcus J., and Edmund H. Durfee. "Deciding When to Commit to Action During Observation-Based Coordination." *ICMAS*. Vol. 95. 1995.

[24] Parker, Lynne E. "ALLIANCE: An architecture for fault tolerant multirobot cooperation." *IEEE transactions on robotics and automation* 14.2 (1998): 220-240.

[25] Asama, H., et al. "Development of task assignment system using communication for multiple autonomous robots." *Journal of Robotics and Mechatronics* 4.2 (1992): 122-127.

[26] Jennings, Nicholas R. "Controlling cooperative problem solving in industrial multi-agent systems using joint intentions." *Artificial intelligence* 75.2 (1995): 195-240.

[27] Tambe, Milind. "Towards flexible teamwork." *Journal of artificial intelligence research* 7 (1997): 83-124.

[28] Wang, Jing. "On sign-board based inter-robot communication in distributed robotic systems." *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994.

[29] Asama, Hajime, et al. "Communication system between multiple robotic agents." *the 1992 Japan- USA Symposium on Flexible Automation Part 1(of 2)*. 1992.

[30] Asama, Hajime, Akihiro Matsumoto, and Yoshiki Ishida. "Design Of An Autonomous And Distributed Robot System: Actress." *IROS*. Vol. 89. 1989

[31] Werner, Gregory M., and Michael G. Dyer. "Evolution of herding behavior in artificial animals." *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. Vol. 2. MIT Press, 1993.

[32] Steels, Luc. "A case study in the behavior-oriented design of autonomous agents." (1994).

[33] Mondada, Francesco, et al. "The cooperation of swarm-bots: Physical interactions in collective robotics." *IEEE Robotics & Automation Magazine* 12.2 (2005): 21-28.

[34] Arkin, Ronald C. *Behavior-based robotics*. MIT press, 1998.

[35] Smith, R. "Communication and control in problem solver." *IEEE Transactions on computers* 29.12 (1980): 1104-1113.

[36] Parker, Lynne E. "L-ALLIANCE: Task-oriented multi-robot learning in behavior-based systems." *Advanced Robotics* 11.4 (1996): 305-322.

[37] Werger, Barry Brian, and Maja J. Matarić. "Broadcast of local eligibility for multi-target observation." *Distributed autonomous robotic systems 4*. Springer Japan, 2000. 347-356.

[38] Botelho, Sylvia C., and Rachid Alami. "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement." *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 2. IEEE, 1999.

[39] Dias, M. Bernardine, and Anthony Stentz. "Opportunistic optimization for market-based multirobot control." *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. Vol. 3. IEEE, 2002.

[40] Gerkey, Brian P., and Maja J. Mataric. "Sold!: Auction methods for multirobot coordination." *IEEE transactions on robotics and automation* 18.5 (2002): 758-768.

[41] Viguria, Antidio, and Ayanna M. Howard. "An Integrated Task Allocation Approach for Multi-Robot Navigation in Realistic Scenarios."

[42] Viguria, Antidio, Ivan Maza, and Anibal Ollero. "SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets." *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007.

[43] Saaty, Rozanne W. "The analytic hierarchy process—what it is and how it is used." *Mathematical modelling* 9.3 (1987): 161-176.