DEVELOPING A NOVEL ROBUST MULTI-AGENT TASK ALLOCATION ALGORITHM FOR FOUR-LEGGED ROBOT SOCCER DOMAIN

by

Çetin Meriçli

B.S. in Computer Engineering, Marmara University, 2002

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2005

DEVELOPING A NOVEL ROBUST MULTI-AGENT TASK ALLOCATION ALGORITHM FOR FOUR-LEGGED ROBOT SOCCER DOMAIN

APPROVED BY:

Prof. H. Levent Akın (Thesis Supervisor)	.....
Asst. Prof. Pınar Yolum Birbil	.....
Prof. H. Işıl Bozma	.....

DATE OF APPROVAL: 08.07.2005

ACKNOWLEDGMENTS

First of all, I would like to thank to my supervisor Prof. H. Levent Akın for his encouragement, guidance and enthusiastic support. This thesis would not be possible without his contributions.

People of AILab and Cerberus Team has taken place in every single moment of this work with their valuable friendship, discussions, brilliant ideas and support so I would like to thank all of them so much.

My appreciations goes to my sincere jury members Asst. Prof. Pınar Yolum Birbil and Prof. H. Işıl Bozma.

I am so grateful to Hüseyin Öner, who is a tolerant boss. This thesis would not be completed without his tolerance.

I owe so much to my family for what they did for me from the very beginning, and for their endless love and support.

In addition, I am so grateful to my younger brother Tekin Meriçli for his continuous motivation, constructive criticism and support.

Bahar Karaoğlu receives special thanks for her morale support, friendship and remarkable discussions about the life, universe and everything.

Thanks to Rodney Brooks for poisoning a thirteen year old child with his robots Genghis and Attila and indirectly causing this thesis to be written.

Finally, thanks to Donald Knuth for making the development of undoubtedly the greatest word processing system ever, LaTeX, possible.

ABSTRACT

DEVELOPING A NOVEL ROBUST MULTI-AGENT TASK ALLOCATION ALGORITHM FOR FOUR-LEGGED ROBOT SOCCER DOMAIN

Multi-robot systems become more popular since a team of relatively simple robots may achieve a complex goal more effectively than a single complex robot if a proper design paradigm is used. Two main advantages of multi-robot systems over single robot systems are their robustness and higher performance due to parallel execution. Multi-robot systems have a wide application area from mine sweeping to planetary exploration and from soccer playing to search and rescue operations in disaster areas.

Robot soccer is a good platform to test and develop multi-robot applications because it has some physical limitations such as limited and noisy sensorial information and noisy actuators as in the real life and it also has a highly dynamic environment.

The goal of winning the game should be decomposed into a sequence of subgoals and proper sequences of actions for achieving the subgoals should be selected and refined through execution. In order to be able to select proper actions at a time, it should be able to evaluate the current situation of the environment so we have to have some metrics that gives quantitative information about the environment.

In this work, we first propose some metrics calculated from positions of robots and ball on the field and select a subset of these metrics that are statistically proved to be informative. Then, a task allocation algorithm is built on top of those metrics. Experimental study on both metric selection and evaluation of the designed algorithm are given.

ÖZET

DÖRT BACAKLI ROBOT FUTBOLU ORTAMI İÇİN YENİ VE DAYANIKLI BİR GÖREV ATAMA YÖNTEMİ GELİŞTİRİLMESİ

Uygun bir tasarım paradigması kullanıldığında, göreli olarak basit robotlardan oluşan bir takım, karmaşık bir görevi tek bir karmaşık robottan daha etkin bir şekilde yerine getirebileceğinden çoklu-robot takımları giderek popülaritelerini arttırmaktalar. Çoklu-robot sistemlerinin tekil-robot sistemlerine göre ana avantajları sistemin hatalara karşı dayanıklı olması ve paralel işlemeden ötürü yüksek performans sağlanmasıdır. Çoklu-robot sistemleri mayın temizlemeden gezegen keşiflerine, futbol oynamadan felaket sonrası arama-kurtarma çalışmalarına kadar pek çok uygulama alanına sahiptir.

Robot futbolu, sınırlı ve gürültülü algılayıcı bilgisi ve gürültülü eyleyiciler gibi fiziksel kısıtlara ve son derece dinamik değişen bir ortama sahip olduğundan, çoklu-robot uygulamalarını geliştirmek ve test etmek için çok uygun bir ortamdır.

Oyunu kazanma amacı bir dizi alt-amaca çevrilmeli, bu alt-amaçlara ulaşabilmek için gereken uygun eylemler seçilmeli ve oyun süresince iyileştirilmelidir. Uygun eylemleri seçebilmek için ortamin o anki durumunu değerlendirebiliyor olmamız, dolayısı ile ortam hakkında nitel bilgi sağlayabilen bazı ölçütlerimiz olması gerekir.

Bu çalışmada, önce robotların ve topun sahadaki pozisyonlarından hesaplanan bazı ölçütler tanımlayarak bu ölçütler arasından bilgilendirici olduğu istatistiksel olarak kanıtlanmış bir altkümeyi seçeceğiz. Sonra, seçilen ölçütlerin üzerine dayanıklı bir görev paylaşımı yöntemi yerleştireceğiz. Gerek ölçüt seçimi, gerek tasarlanan yöntemin sınanması konularındaki deneysel çalışmalar detaylı olarak verilecektir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADR	Attack/Defense Ratio
BP	Ball Possession
FIRA	Federation of International Robot-Soccer Association
HP	Hodrick-Prescott
MIT	Massachusetts Institute of Technology
RF	Radio Frequency
RIY	Robot Idman Yurdu
RL	Reinforcement Learning
SD	Score Difference
VRML	Virtual Reality Modeling Language

1. INTRODUCTION

Multi-robot systems are getting more popular since a team of relatively simple robots may achieve a complex goal more effectively than a single complex robot if a proper design paradigm is used. Two main advantages of multi-robot systems over single robot systems are their robustness to the failures (if a decentralized architecture is used, the system is free from single-node failure) and higher performance due to parallel execution [1].

Multi-robot systems have a wide application area from mine sweeping to planetary exploration and from soccer playing to search and rescue operations in disaster areas and collapsed buildings.

Robot soccer is a good platform to test and develop multi-agent applications because it has some physical limitations such as limited and noisy sensorial information and limited moving capability as in the real life and it also has a highly dynamic, real-time environment.

Multi-agent coordination is a very active research area in mobile robotics and many different approaches have been proposed for performing task allocation in different environments. In general, the size of the team, the overall goal and properties of the environment play important roles on the selection and implementation of the task allocation algorithm. For example, in a search and rescue in a disaster environment task, time is the main criterion. On the other hand, in a soccer game, only the score is important. An algorithm with high communication burden may not be a problem in a team of three robots working in the same room, cleaning the floor but it does matter if a team of a hundred unmanned trucks spread over a hundred mile-square area are on a scout mission. Since we are dealing with the robot soccer case, we will briefly examine the task allocation algorithms in general multi-robot problems and we will focus on research in robot soccer domain.

Since it is hard to develop a controller for a single mobile autonomous robot, it is even harder to develop a controlling and task allocation system for a team of mobile robots. Zlot *et al.* [2], presented a free-market driven exploration algorithm for mapping with multi-robot teams. Kaplan has proposed a fixed role assignment schema for small sized robot soccer teams [3]. CMPack'02 Legged Robot Soccer Team from Carnegie-Mellon University used a bidding mechanism for dynamic role assignment in a game [4]. Kose *et al.* presented a task allocation algorithm based on free-market approach [5]. In this work, we raised the question whether it is possible to develop a robust, immune to single node failures and quite noisy sensory information and computationally and communicationally cheap task allocation algorithm for a highly dynamic environment like robot soccer domain and we have tried to find an answer to this question.

The motivation behind this thesis is to explore the possibility of implementing a task allocation algorithm for a robot soccer team that is robust, flexible, allowing both low level reactive behaviors and high level (game level) strategies to be defined. Also, exploration of possibility to define a set of metrics that can evaluate the situation of soccer game in different hierarchical resolutions and exploration of the possibility to propose a method for statistical validation of metric informativeness.

A very brief information about robots, mobile robots and multi-robot systems is given in the Chapter 2. Basics of behavior-based (or, bottom-up approach) robotics, motor schemas and use of potential fields in mobile robot navigation are discussed and brief information about multi-robot teams are presented.

In Chapter 3, first the metric definition and verification processes are explained in high detail. Then, proposed three-layered approach to soccer strategy and low level behavior definition is presented. The defined roles for players and objective functions measuring the fitness degree of a player to a certain role are discussed. Finally, the potential field definitions for low level behaviors are given.

Chapter 4 contains information about the setting and performing experiments

both in simulator and on physical robots.

Finally, we conclude in Chapter 5 by summarizing the work done and pointing to possible future work.

2. BACKGROUND

Mobile robot control and multi-agent task allocation problems are very active research areas and many different approaches have been proposed for both single-agent and multi-agent control of a robot / robot team. In this section, first traditional single-agent robot control paradigms are examined. Then a literature survey about multi-robot systems is presented. Finally, detailed information about robot soccer domain is presented and different robot soccer organizations and leagues are examined.

2.1. Robot Control Paradigms

Robot control paradigms are often described by the relationship among the three commonly accepted primitives of robotics:

- Sense
- Plan
- Act

Functions taking in information from robot's sensors are considered in the *Sense* category. Functions taking in information from either the sensors or internal knowledge of the robot and produces one or more tasks to perform falls in the *Plan* category. Finally, functions producing output commands to robot's actuators are in the *Act* category [6].

Table 2.1. Robot primitives defined in terms of inputs and outputs

Robot Primitives	**Input**	**Output**
Sense	Sensor Data	Sensed Information
Plan	Information (Sensed and/or cognitive)	Directives
Act	Sensed information or directives	Actuator commands

2.1.1. Reactive Paradigm

Reactive architectures [6] simply ignore the *Plan* phase and use the outputs of *Sense* layer as the inputs of *Act* layer (Table 2.2). Connecting sensory input to actuation layer with a very low computational overhead allows the robot to respond the changes in its environment very quickly. Although the output of reactive controllers are impressive, it was quickly realized that completely ignoring the planning part is not suitable for general purpose robots. Reactive architectures have many advantages such as fast response time, low memory usage and using the world as its own model (no abstract internal world modelling is required).

Table 2.2. Relationships among robotic primitives in Reactive Paradigm

Robot Primitives	Input	Output
Sense	Sensor Data	Sensed Information
Plan	Information (sensed and/or cognitive)	Directives
Act	Sensed information or directives	Actuator Commands

2.1.2. Deliberative Paradigm

Deliberative Architectures are also known as "Hierarchical Paradigm" or "Top-down Architectures" [1]. In a deliberative architecture, robot senses the world through *Sense* layer, constructs an internal model of the worlds by using both sensory information and its knowledge base through *Plan* layer, and finally acts upon the directions generated by planning layer through *Act* layer. The major advantage of the deliberative paradigm is allowing the robot to act strategically based upon a specified goal. The major disadvantages are high computational and memory requirements, responsing slowly and infrequently, and possibility of failure in case of losing the synchronization between internal world model and real world [7]. Also, deliberative architectures suffers from the *Closed World Assumption* (Internal world model of the robot contains everything the robot needs to know, so there can be no surprises).

Table 2.3. Relationships among robotic primitives in Deliberative Paradigm

Robot Primitives	Input	Output
Sense	Sensor Data →	→ Sensed Information
Plan	Information (sensed and/or cognitive) →	→ Directives
Act	~~Sensed Information or directives~~ →	→ Actuator Commands

2.1.3. Hybrid Paradigm

The Hybrid Paradigm [1] combines desirable properties of both reactive and deliberative paradigms in a way that the robot uses planning to decompose the task into proper subtasks and determines the appropriate behaviors for accomplishing them. Then, behaviors start executing according to reactive paradigm. This paradigm is often referred as *Plan, Sense-Act*, meaning that planning is done as one step and then sense-act coupling is executed at once. Sense layer in the hybrid paradigm is a mixture of reactive and deliberative paradigms. Sensor data are used by behaviors, but are also available for the planning layer. Combination of deliberative and reactive paradigms requires a third layer. This third layer is named according to the state organization algorithm it uses. The ones containing a memory about the past are called *Sequencer* while the ones predicting the future are called *Deliberators* [8, 9]. Although the hybrid approach solves the major pitfalls of deliberative and reactive paradigms, it is quite challenging to design an appropriate middle layer.

Table 2.4. Relationships among robotic primitives in Hybrid Paradigm

Robot Primitives	Input	Output
Plan	Information (sensed and/or cognitive) →	→ Directives
Sense - Act (Behaviors)	Sensor Data →	→ Actuator Commands

2.1.4. Behavior-based Architectures

In Behavior-based systems [1, 6], there is a set of sensor-actuator couplings called *Behaviors* and computation is distributed over these behaviors. Output action is determined according to a behavior combination algorithm. There are many different behavior combination algorithms such as cooperation based algorithms, competition based algorithms and subsumption algorithm [1, 10]. Behavior-based architectures performs best when the detailed and accurate model of the real world is not available.

2.1.4.1. Subsumption Architecture. The Subsumption Architecture was developed by Brooks at MIT Artificial Intelligence Laboratory in mid-80's [10]. In the subsumption architecture, there is a hierarchical organization of behaviors in which the outcome of a behavior at a higher level can *subsume* the outcomes of the behaviors in lower hierarchical levels. Higher level behaviors may access the lower level behaviors but lower level behaviors are not aware of the higher levels. Behaviors decide when to become active according to the sensor readings. Since the philosophy of reactive architectures is "The world is its own model", there is no internal world model or any kind of abstraction based on sensory information.

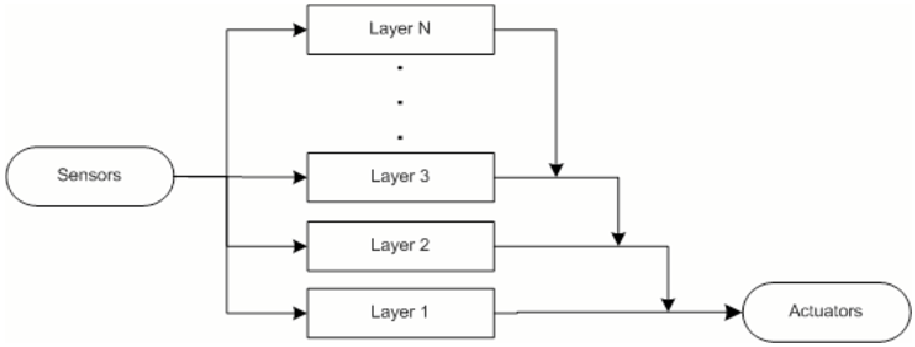

Figure 2.1. Subsumption Architecture

2.1.4.2. Schema-Based Behaviors. Use of schema theory is another widely used approach in behavior-based architectures [1]. In the schema-based approach, the behaviors consist of perceptual and motor schemas and a perceptual schema is embedded within each motor schema. In perceptual schemas, the way of processing the sensor data is encoded and in motor schemas, response is represented in a uniform format.

The final action of the robot is determined by performing a weighted vector sum of individual behaviors.

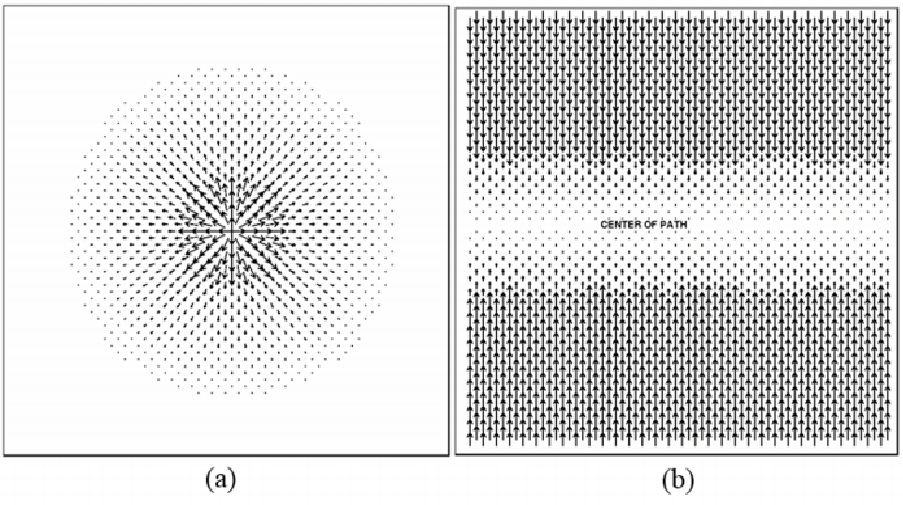

Figure 2.2. Example motor schemas: (a) Obstacle avoidance and (b) Path following

2.1.4.3. Potential Fields. The major drawback of motor schemas is that the entire schema for the environment must be created at once. Since in real world applications, the environment is very dynamically changing, the motor schemas have to be recalculated continuously. For large environments, the computational requirement of this recalculation is usually not affordable for the robots. In the potential field approach [12], the resulting vector is obtained for only the point at which the robot is located on. As in motor schemas, attractive and repulsive points are used for directing the robot. Potential field theory from electromagnetics is borrowed for formulating the calculation of the result vector for a given point with respect to given attractive and repulsive points. Although the force is inversely proportional to the square of the distance, the exact function describing the relation should be selected carefully since the result may largely be affected.

Figure 2.3 shows the 3D plot of the function for a repulsive potential field. The function is given in Equation 2.1.

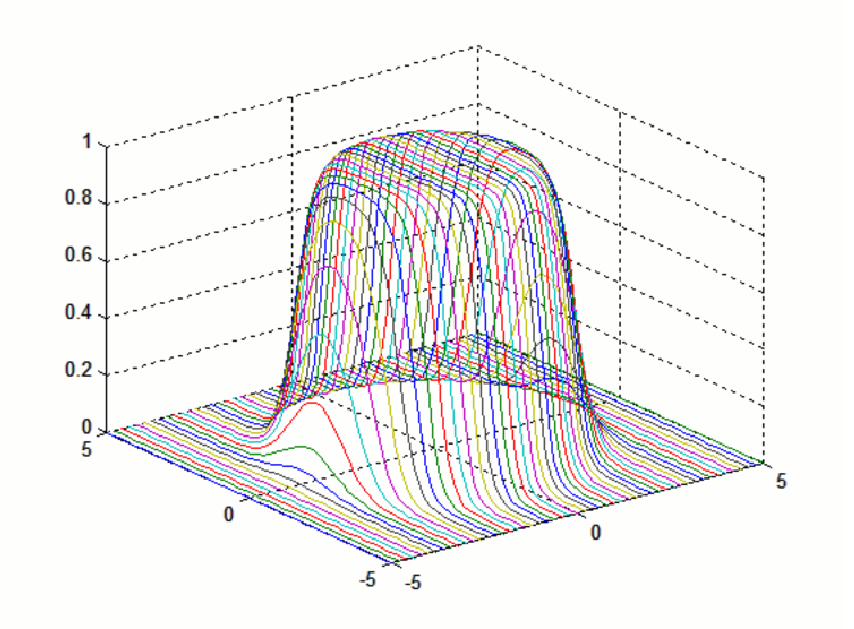

Figure 2.3. An example repulsive potential field

$$F = \frac{1}{1 + e^{k \text{ x } distance}} \tag{2.1}$$

where, k is a coefficient determining the sharpness of the function and *distance* is the euclidean distance to the center of the repulsive field [3]. Value of the k is 5 for Figure 2.3.

2.2. Multi-Robot Systems

The study of multiagent systems focuses on systems in which many intelligent agents interact with each other. The agents are considered to be autonomous entities, such as software programs or robots. There are many different issues to be considered in order to characterize a team of agents. These considerations can be listed as reliability, organization, communication, spatial distribution, congregation, and performance. Arkin discussed the different properties of multi-robot teams varying from social behavior to inter-robot communication and from performance issues to etholog-

ical considerations [1].

- **Reliability:** If the system can act correctly in a given situation over time, then the system can be considered to be reliable.
- **Social Organization:** Heterogeneous societies should be developed if there is a demand for specialized skills. Some examples can be multilevel hierarchical structures, loosely structured mobs, and dominance systems.
- **Communication:** There are two major aspects in communication. One of them is information content, which can be described as the limit of the messages in societies. For example, for ants, there are ten to twenty different chemical signals. The other aspect of communication is mode. In different animal societies different range of communication mechanisms including chemical, tactile, infrared, and electric communication are used.
- **Spatial Distribution:** It is very important for activities such as foraging for food. If a resource is evenly distributed, it is better for the agents to form individual, non-overlapping foraging ranges instead of roosting and foraging together (Horn's Principle of Group Foraging [11]).
- **Congregation:** There are several strategies to keep the society remain together, such as defining a colony location as a predefined meeting point, generation of a loud noise by a number of similar agents (lekking), distinctive calls, and specific assembly calls by a single agent.
- **Performance:** Specific metrics are required to effectively evaluate societal system performance. Speedup is one of the useful metrics that measure the performance of a team of N robots relative to N times the performance of a single robot.

A complete taxonomy capable of categorizing the variety of multiagent robotics systems is proposed in [13]. Team size (number of robots), communication range (each robot's ability to communicate directly with other team members), communication topology (the pathways by which communication can occur), communication bandwidth (amount of communication available), team reconfigurability (flexibility regarding the structure and organization of the team), team unit processing ability

(underlying computational model used), and team composition (composition of agents themselves, i.e. homogeneous or heterogeneous) are the proposed lines for characterizing teaming.

2.3. Robot Soccer

In robot soccer, teams of robots, that are capable of seeing and moving play matches against each other, and the team with the highest goal score wins the match as in real soccer. In order to do this, the player robots must detect their location, the goals, the ball, the members of their team and the opponent team members (optional for high level planning), and place the ball in the opponent team's goal to score. A robot is typically expected to find its own location using the landmarks (artificial or natural) in the field, and then use this information to find the location of the ball and goal.

Federation of International Robot-soccer Association (FIRA) is an association for international robot soccer [14] where wheeled and legged robots compete in the official games. MIROSOT is one of the categories for these robots. In MIROSOT, robots are controlled by a central system via RF signals. Since the environment is very dynamic, a complex controller design is required to perform sophisticated tasks like team organization.

MIROSOT small-size league is a centralized system. Configurations consists of a ceiling-mounted camera having the ability to cover entire field, a host computer and an RF interface for sending commands to the robots. After processing the image obtained from the camera, the controller computes the proper actions (in wheeled case, velocities for left and right wheels) for each robot. The produced commands are then transmitted to the robots via RF. The setup configuration for MIROSOT game can be seen in Figure 2.4.

The RoboCup organization is an international initiative which aims to build a soccer team of fully autonomous humanoid robots beating the last human world soccer

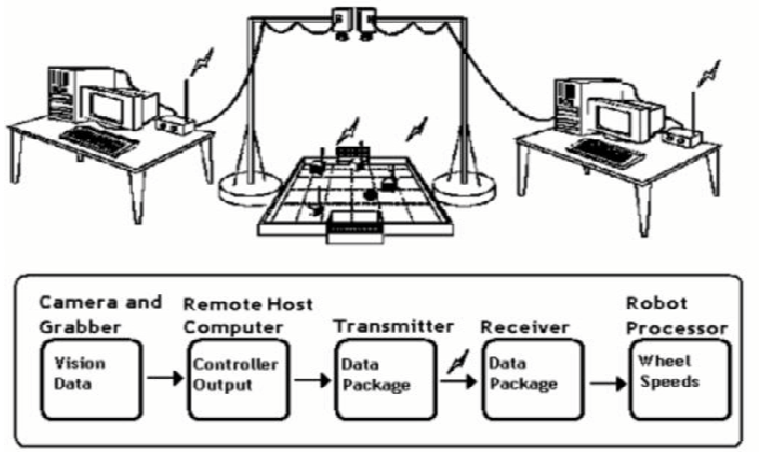

Figure 2.4. Setup configuration for MIROSOT Game

champion by the year 2050 [15].

Sony four-legged league is one of the subdivisions of the RoboCup in which two teams each consisting of four Sony AIBO robotic dogs compete against each other. The game area is 6m by 4m and four unique bi-colored beacons are placed in order to provide information for localization. Robots are fully autonomous so any human intervention other than placing the robots on the field, any off-board computation and providing external data such as image from overhead camera is prohibited. Field setup can be seen in the Figure 2.5. This thesis is a part of the Cerberus Team [16] competing in this league.

2.4. Task Allocation in Multi-Robot Systems

Zlot *et al.* [2], presented a free-market driven exploration algorithm for mapping with multi-robot teams. In their work, robots continuously communicate among themselves for exchanging the calculated cost values for exploration of certain portions of the area. The robot with the lowest exploration cost is assigned to that portion of the area. If a robot proposes a lower cost for exploration of a portion of the area that is already assigned to a robot, the owner *sells* the task to that robot. In market-driven approach, the overall goal is to maximize profit by minimimizing the overall cost since the profit is calculated as $profit = payoff - cost$ and it is assumed that a fixed payoff

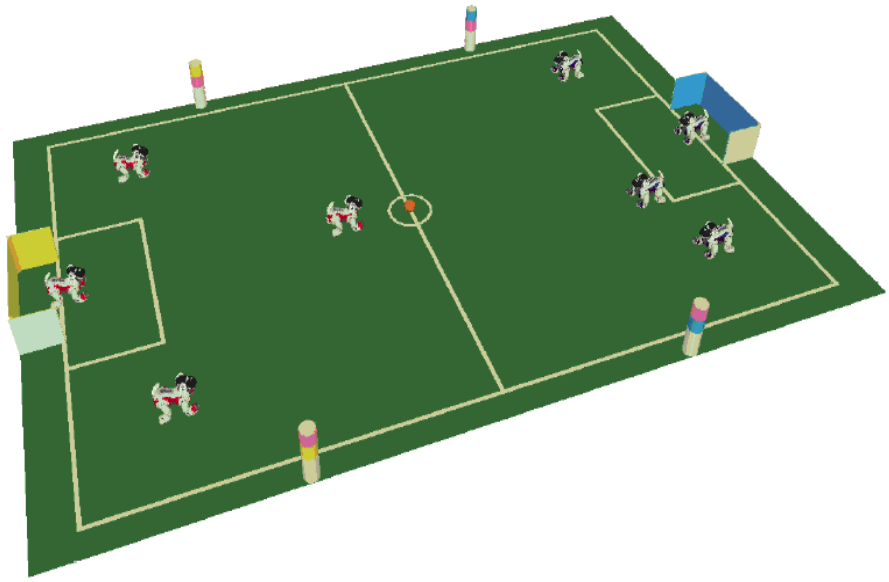

Figure 2.5. Field setup for RoboCup Legged League

is offered for achieving the overall goal of the team.

Kaplan has proposed a fixed role assignment schema for small sized robot soccer teams [3]. Kaplan used potential fields for behavior encoding. Field borders and own goal area are represented with repulsive fields where the ball and opponent goal are represented with attractive potential fields. Then, the output vectors of corresponding potential fields are summed to determine final movement direction and speed of the robot. The metric they used for evaluating the objective functions to assign roles was the distance to the ball. The major drawbacks of their approach are use of primitive metrics and not considering another action than shooting to the opponent goal.

CMPack'02 Legged Robot Soccer Team from Carnegie-Mellon University used a bidding mechanism for dynamic role assignment in a game [4]. They used a set of objective functions for calculating the fitness of players to the roles and then the fittest player is assigned to the role. They used a time threshold for preventing oscillations in role assignments which often occurs due to misreadings obtained from sensors and partial observability of the environment. When a player is assigned to a role, it keeps that role at least for a certain amount of time.

Kose *et al.* presented a task allocation algorithm based on free-market approach [5]. In their work, they defined a set of roles that a player can be assigned in the game. They defined five roles:

- Goalie
- Defender
- Primary Attacker
- Secondary Attacker
- Tertiary Attacker

The *Goalie* role is assigned to one player at the beginning of the game and is not changed throughout the game since game rules strictly separates the goalie from the other players. Then, each player evaluates its fitness to each dynamic role by calculating a cost function for accomplishing the main task of that role. Calculated costs are then exchanged with the teammates and sorted. Since each player has the cost list for the entire team, they *select* the appropriate role rather than being *assigned* to that role. For example, the main task of the primary attacker is to score so the player with the lowest attacking cost (hence, the highest fitness) is assigned to the *Primary Attacker* role. Once a player is assigned to a role, there are two alternatives: The robot may open an auction to sell its role, or the robot may perform an internal auction to select the best behavior to be performed. If another player announces a lower cost for a task, then the current owner of the task sells it to the player with the cheaper cost. In this work, two behaviors for the players are defined: *Shoot* or *Pass*. But, the objective functions for these behaviors are different for each role. They used the low level potential field definitions of Kaplan [3] for solving the motion planning problem.

In both role assignment and behavior selection phases, objective functions consisting of some *metrics* are used to evaluate the fitness of a robot to either a role or to a behavior. Kose and her colleagues used the following metrics:

- Distance to the ball

- Orientation with respect to ball
- Clearance to the ball (i.e. the level of reachability of a player to the ball)
- Distances to the own and opponent goals
- Orientations to the own and opponent goals
- Clearances to the own and opponent goals (i.e. the level of reachability of a player to the own and opponent goals)

The cost functions they used to evaluate fitness values are weighted sums of the metrics above. For example, attacking cost consist of distance to the ball, clearance to the ball and orientation to the ball. The weights of the metrics are handcrafted.

Kose *et al.* have discussed an evolutionary optimization of the weights of the metrics in cost functions in their next paper [17]. Also, they allowed that more than one player can be assigned to same role. They argued that the use of genetic algorithms to determine contributions of metrics to the cost functions resulted in a better performance and the optimized team has outperformed it's handcrafted predecessor.

In Kose *et al.* [18], a Q(λ) learning algorithm replaces the role assignment algorithm in it's predecessors. They used a total of 33 parameters for state representation. They also mentioned the importance of opponent selection in training of adaptive algorithms like RL. They used moderate opponents so that both attack and defense skills can be learned in equal opportunity.

In Tatlidede *et al.* [19], Q(λ) learning is used for policy learning and Cerebellar Model Articulation Controller (CMAC) is used for function approximation and state generalization. For the low level behaviors, Kaplan's potential field definitions for robot soccer [3] are used.

3. PROPOSED APPROACH

The proposed approach for role assignment and task allocation is presented in this section. The section starts with the explanation of the proposed three-layered architecture followed by the metric definition for evaluating the game in different time resolutions. After discussing the proposed metrics for three different time resolutions, proposed *Plays* and methods for game level strategy determination are presented.

3.1. Team Architecture

Due to the highly dynamic nature of the robot soccer and the necessity of very short response times, the researchers are forced to use reactive architectures rather than deliberative architectures in robot soccer players and teams. On the other hand, game strategies and soccer tactics play key roles in the results of the games and neither high level game strategies like changing the distribution of the players on the field according to the strength of the opponent team nor according to the current score in the game can be handled in a reactive system. Both tactics and game strategies are distributed over time and requires more deliberation. In order to be able to use high level soccer tactics and game level stragegies without sacrificing from the quick response time of the reactive architectures, we divided the architecture into three layers in which the game is evaluated in three different time resolutions.

- **Reactive Layer :** Works with the shortest time intervals (in each camera frame or in each time-step) at the bottom of the system. Reactive layer deals with instantaneous information and does not use a history of sensory information. This is the layer where sensory data is obtained and motion plan is generated. The layer on top of the reactive layer is the play layer.
- **Play Layer :** Is the deliberative part of the system and allows us to define tactics that can be performed in a long amount of time (i.e. performing counter-attack, attack, etc.).
- **Game Layer :** Game layer is on top of the other layers for defining team and

game level offensive / defensive level. The decided game strategy is applied by modifying the motion plan generated by the play layer or the reactive layer.

Each layer uses the information available in its time resolution. For example, the reactive layer deals only with the information in a snapshot of the environment, while the game layer uses the information over a long time period in order to make decision about the game strategy.

3.2. Metrics for Game Evaluation

Both the individual robots and the entire team should make decisions about the game strategy, selecting appropriate play or selecting roles and behaviors. In order to evaluate the situation and compute the fitness for roles / behaviors / game strategies to be selected, we need to measure some metrics in the environment that can be quantitatively expressed and having the same trends in the same situations so that we can build objective functions out of these metrics.

The most primitive information we can estimate in the environment is the position information of players and the ball so we defined a set of metrics calculated from the position information for different time resolutions.

An example soccer metric would be the distance of the ball to the opponent goal area. One can say that a team should try to keep this metric as small as possible in order to be able to score and avoid possible opponent scores. The metric is given in Equation 3.1.

$$Dist_{ball}(goal) = \sqrt{(ball_x - goal_x)^2 + (ball_y - goal_y)^2} \tag{3.1}$$

where, $ball_x$ and $ball_y$ are coordinates of the ball and $goal_x$ and $goal_y$ are coordinates of the goal.

We propose metrics for three different time resolutions in game:

- Instantaneous Metrics
- Play Level Metrics
- Game Level Metrics

Table 3.1 shows a list of the defined metrics for each time resolution.

Table 3.1. List of metrics for each time resolution

Time Resolution	**Metric Name**
Instantaneous	Density of the Convex Hull for Own Team
	Density of the Convex Hull for Opponent Team
	Area of the Convex Hull for Own Team
	Area of the Convex Hull for Opponent Team
	Vicinity Occupancy for the Ball
	Vicinity Occupancy for the Own Goal Area
	Vicinity Occupancy for the Opponent Goal Area
	Pairwise Separation for the Ball
	Clearance to the Ball
	Clearance of the Ball to the Opponent Goal
	Clearance to the Ball to the Teammates
	Distance between two points
Play Level	isReachable
	hasBall
Game Level	Attack / Defense Ratio
	Ball Possession
	Score Difference

3.2.1. Instantaneous Metrics

Instantaneous metrics are calculated from one-step position information. These metrics are used for the evaluation of the objective functions for role assignment. The

ball is the most important object in the soccer game and getting control of the ball is the most important sub-task. So, most of the metrics are proposed for evaluating the chance of getting control of the ball.

3.2.1.1. Convex Hull Metrics. *Convex Hull* of a set of points is defined as the smallest convex polygon in which all of the points in the set lies. In an analogous manner, by subsituting points with the players in the soccer, we obtain a new concept: *Convex Hull of a Team*. We propose two metrics involving the convex hull of a team:

- *The Area of Convex Hull* tends to measure the degree of spread of the team over the field. The value of this metric increases as the team members are scattered across the field.
- *Density of Convex Hull* is applied only if the ball falls within the convex hull. The formal definition of the density is given in Equation 3.2.

$$Density = \frac{\sum_{i=1}^{N} \sqrt{(X_{player(i)} - X_{ball})^2 + (Y_{player(i)} - Y_{ball})^2}}{N} \tag{3.2}$$

where, N is the number of players on the corners of the convex hull.

The density value is calculated only if the ball falls within the convex hull. If the ball falls outside of the convex hull, then the value of the metric is 0. The probability of the ball falling within the convex hull increases as the area of convex hull increases and it is expected that if the ball falls within the convex hull, the probability of getting the control of the ball increases. On the other hand, it is expected that the probability of getting the control of the ball increases as the density of the convex hull increases.

But, since the values of the density and the area are negatively correlated (i.e., density decrease as the area increase), either one of the metrics is not useful, or there is an equilibrium point where both metrics are increasing the probability of getting the

control of the ball.

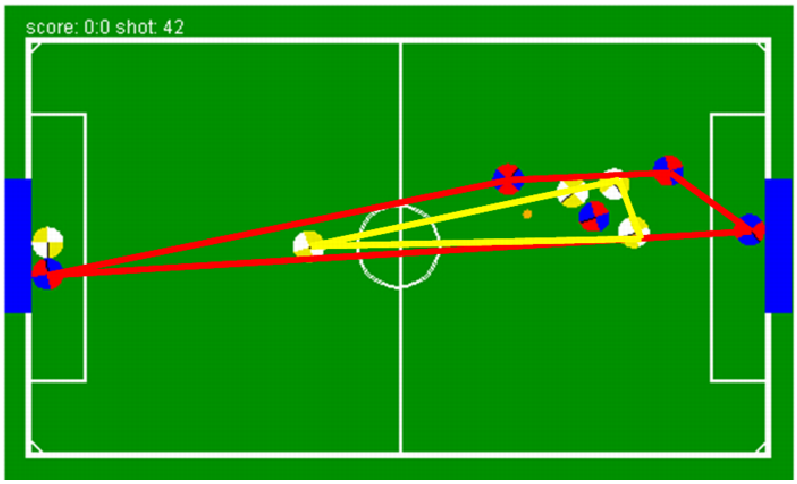

Figure 3.1. Convex Hulls of two teams at a time point

3.2.1.2. Vicinity Occupancy. *Vicinity Occupancy* measures the ratio of the teams players to the opponent players within a vicinity of the object of interest. The formal definition of the vicinity occupancy is given in Equation 3.3.

$$Occupancy = \frac{P_{own} - P_{opp}}{P_{own} + P_{opp}} \tag{3.3}$$

where, P_{own} is the number of own players in the vicinity of the object of interest, P_{opp} is the number of opponent players in the vicinity, and P is the total number of players in the vicinity. The result is a real number in the interval $[-1, 1]$ where -1 means that the vicinity is dominated by the opponent players, 0 means that there is no dominance and finally, 1 means that the vicinity is dominated by our players. Vicinity Occupancy is calculated for three objects of interest:

- Ball
- Own Goal Area
- Opponent Goal Area

The ball is the most important object in the game. Dominating the vicinity of

the ball can be interpreted as having the control of the ball since the probability of controlling the ball increases as the number of own players in the vicinity increases and decreases as the number of opponents in the vicinity increases.

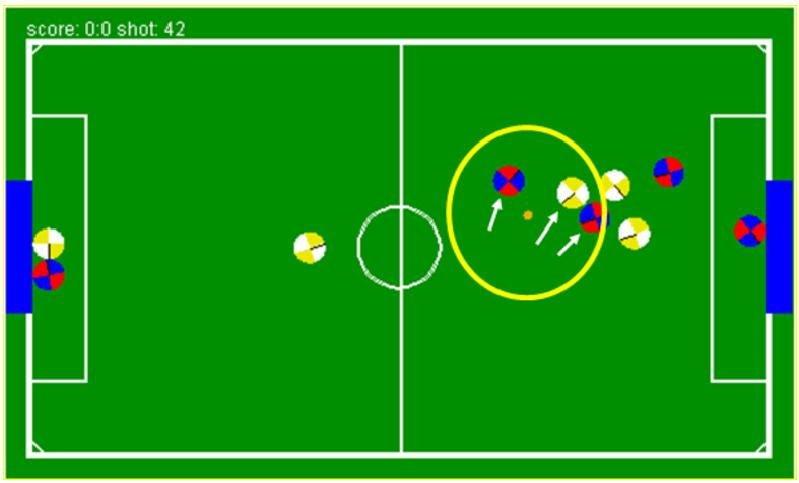

Figure 3.2. Occupancy in the vicinity of Ball

Dominating the vicinity of own goal is desired since it can be interpreted as a good defensive tactic. If the opponent players can be kept away from our goal, their chance of scoring will be decreased.

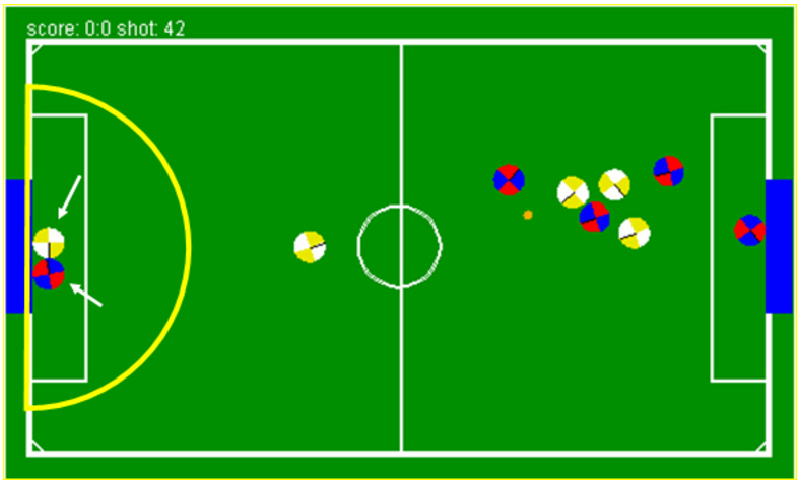

Figure 3.3. Occupancy in the vicinity of Own Goal

Dominating the vicinity of opponent goal is basically the opposite situation of occupancy of own goal case. Domination is desired in this case since if the number of our players are greater than the number of opponent players in the vicinity of opponent goal, our chance to score is increased.

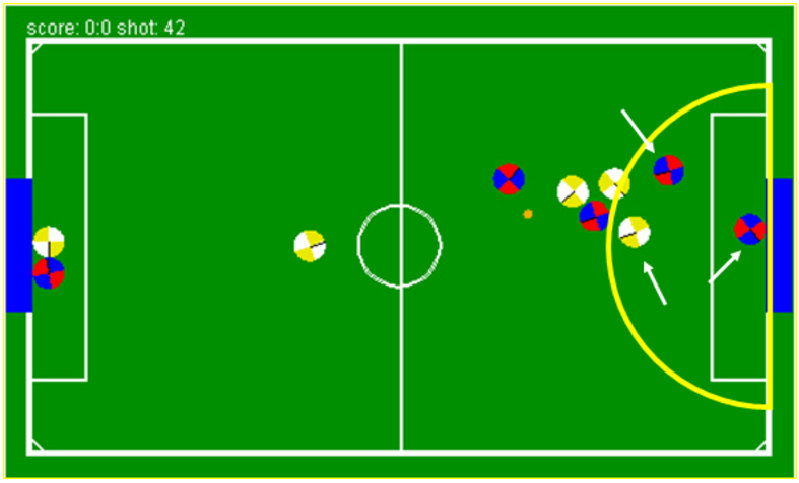

Figure 3.4. Occupancy in the vicinity of Opponent Goal

As a result, in the ideal case, it is desired to dominate vicinities of both ball and goals but dominating the ball vicinity is the most important issue.

3.2.1.3. Pairwise separation Metrics. Pairwise separation is aimed to measure the degree of separation of an object of interest with opponent team. Equation for calculation of pairwise separation is given in Equation 3.4.

$$S_{Object} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} separates(P^i_{own}, P^j_{own}, P^k_{opp}, Object)}{2} \tag{3.4}$$

$$separates(P_1, P_2, P_3, Object) = \begin{cases} 1 & \text{if } Line(P_1, P_2) \text{ intersects } Line(P_3, Object), \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

Pairwise separation depends on the assumption that if an opponent player is *separated* from the object of interest, it is more likely for us to prevent it from accessing the object of interest. For example, if the pairwise separation value for ball is high, our chance to control the ball will also be high. Since separation test is performed for each player and with each teammate, each tuple is counted twice. So, the calculated

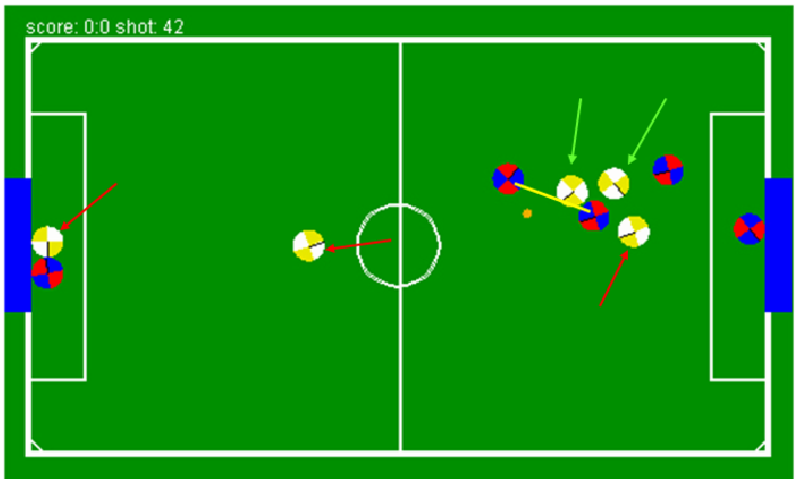

Figure 3.5. Pairwise separation of Ball from Opponent Team: Robots pointed with green arrows are separated from the ball

separation value is divided by 2 to eliminate this double count.

3.2.1.4. Clearance of the path between two points. Clearance metric measures the accessibility of one point from another point. Clearance depends on the existence and positions of players and their movement capability. It is assumed that a player has control over an area called *Area of Impact*. The size and shape of area of impact depends on locomotional abilities of the robot. For a robot with omnidirectional movement and shooting ability with any side (for example Teambots robots or MIROSOT robots), shape of the area of impact will be a circle. Two example areas of impact can be seen in Figure 3.6.

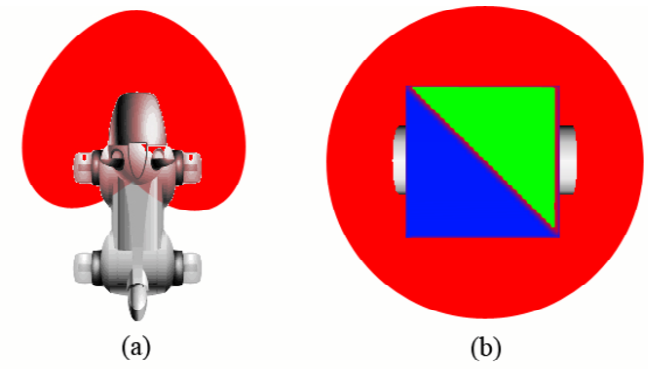

Figure 3.6. Example Areas of Impact: a) Area of Impact for an AIBO robot, b) Area of Impact for a MIROSOT robot

The size of the area of impact depends on the speed of the robot. A fast robot would have a larger area of impact than a slower robot. The area of impact of a robot is considered as a physical obstacle when calculating the clearance. If the path between two points is occluded by the area of impact of at least one opponent player, it is considered that the way between the two points is not clear.

We calculate three clearance metrics:

- Clearance to the ball
- Clearance of ball to the opponent goal
- Clearance of ball to the teammates

Clearance to the ball is related with the accessibility of the ball without an opponent interrupt. If the path to the ball is clear, it means that there is no opponent player nearby to prevent us from reaching the ball so it is highly probable that our player will get the control of the ball.

Once a player reaches to the ball, there are three actions it can take:

- Shooting to the goal
- Dribbling with the ball
- Passing the ball to a teammate

Since it is assumed that the player must reach the ball before shooting or passing, only the clearance of the ball to the opponent goal and to the teammates are important. Sample clearance situations are shown in Figures 3.7 - 3.9.

3.2.1.5. Distance with Respect to a Point. Considering the distance of a point to another point is one of the commonly used metrics in task allocation algorithms. We use distances of players to ball and ball to goals. $distance(Pos_1, Pos_2)$ is calculated as in

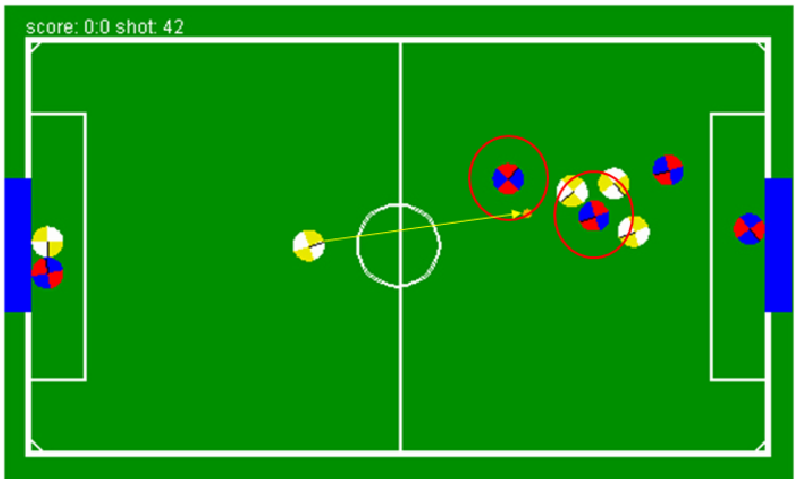

Figure 3.7. Clearance to the Ball

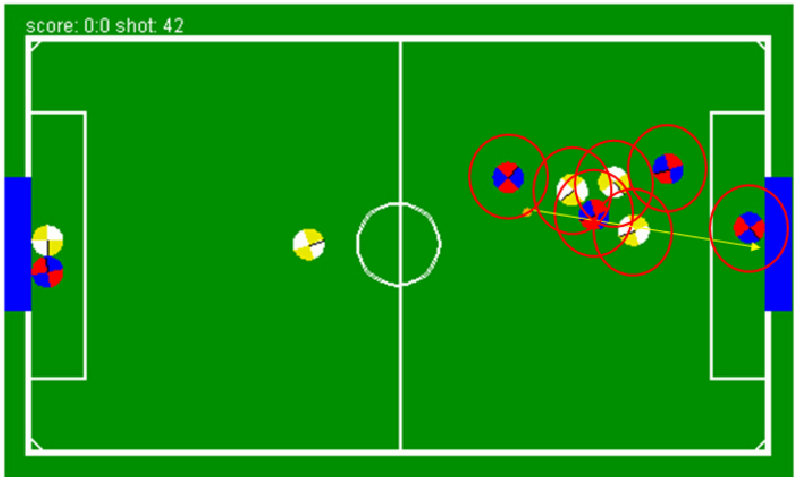

Figure 3.8. Clearance to the Opponent Goal for the Ball

Equation 3.6.

$$distance(Pos_1, Pos_2) = \sqrt{(x_{Pos_1} - x_{Pos_2})^2 + (y_{Pos_1} - y_{Pos_2})^2} \tag{3.6}$$

3.2.2. Play Level Metrics

Play level metrics tend to measure the two important issues in the soccer game: Reachability of a position from another position and ball possession. The proposed predicates $isReachable(Position_{from}, Position_{to})$ and *hasBall(Player)* are calculated

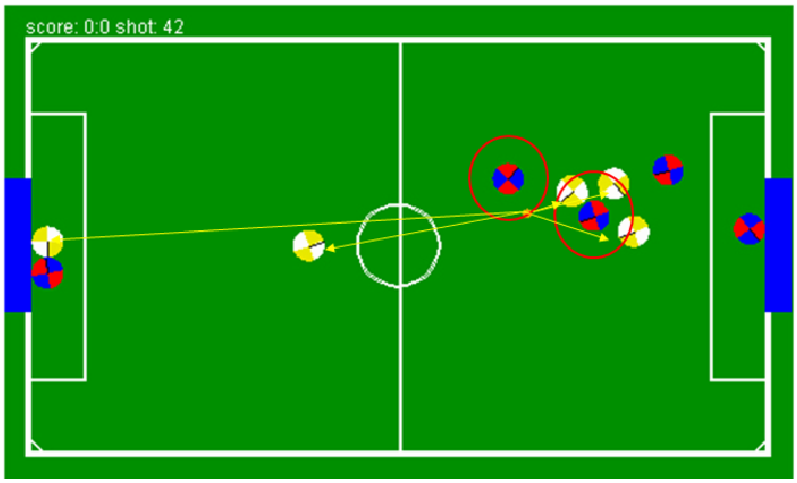

Figure 3.9. Clearance of the Ball to the Teammates

by using instantaneous metrics over a time period. Both *isReachable* and *hasBall* are boolean metrics so we need to map the output of the metric combination from a real number to a boolean value.

3.2.2.1. isReachable Predicate. *isReachable* predicate uses the clearance metrics as the input. $isReachable(Position_{from}, Position_{to})$ returns *True* if the path between $Position_{from}$ and $Position_{to}$ is clear from obstacles (Sec. 3.2.1.4). Since clearance metrics are instantaneous metrics and can be quite noisy, clearance is calculated by examining the consecutive values of the clearance metrics. If the path between two positions is *Clear* for consecutive N time-steps, *isReachable* is set to *True*. Contrarily, if the path between two positions is *Occluded* for consecutive N time-steps, *isReachable* is set to *False*. Determining the number N is another optimization problem. Since such an optimization is beyond the scope of this work, we arbitrarily select $N = 10$ and leave finding the optimal value of N as a future work.

3.2.2.2. hasBall Predicate. *hasBall* is used to check whether a certain player has the ball possession or not. $hasBall(Player)$ returns *True* if the *Player* has the ball possession or not. As in the *isReachable* predicate, output is calculated from the values of the metrics developed for measuring the ball possession over a number of consecutive time-steps. We used the same value of 10 for the window size variable N for calculating

the value of the *hasBall*.

3.2.3. Game Level Metrics

Game level metrics are proposed for measuring the statistics about the game over a long time period. All game level metrics try to measure the dominance of the game. Three metrics are calculated in game level:

- Attack/Defense Ratio
- Ball Possession
- Score Difference

3.2.3.1. Attack/Defense Ratio. Attack/Defense Ratio (ADR) tends to measure the dominance of the game by comparing the longest time the ball spends in our possession area in the game field with the longest time the ball spends in opponent possession area. Possession areas are defined as goal-centered semi-circles. The radius of the circle is a hyper-parameter and it needs some machine-learning and optimization techniques for finding the optimal value of the radius, but we simply select the radius of the circle as half of the field height.

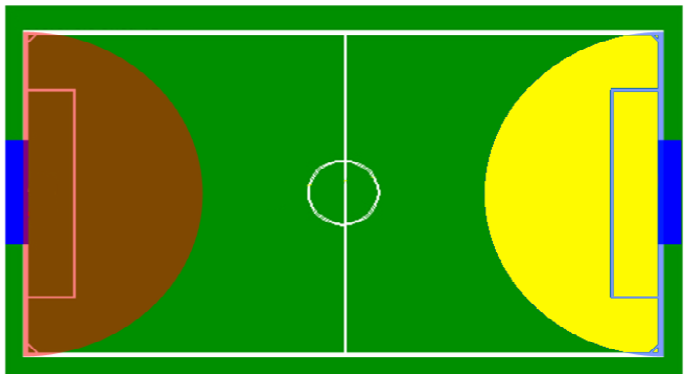

Figure 3.10. Field division for Attack/Defence ratio calculation

The Attack/Defense Ratio is the difference of largest consecutive time-steps that the ball is in opponent possession area and the largest consecutive number of time-steps that the ball is in our own possession area divided by the sum of them. Formula for

Attack/Defense Ratio is given in Equation 3.7.

$$ADR = \frac{Pos_{own} - Pos_{opp}}{Pos_{own} + Pos_{opp}} \tag{3.7}$$

This value is a real number in the interval $[-1, 1]$. A positive value of this metric indicates that the ball is spending more time in the opponent possession field than it spends in own possession field meaning that our team is more aggressive and dominating the game.

3.2.3.2. Ball Possession. Ball Possession (BP) measures the dominance of the game by comparing the longest time our team has the ball possession with the longest time opponent team has the ball possession. Play Level predicate *hasBall* is used in the calculation of ball possession. Formula for Ball Possession is given in Equation 3.8.

$$BP = \frac{Ball_{own} - Ball_{opp}}{Ball_{own} + Ball_{opp}} \tag{3.8}$$

where $Ball_{own}$ is the number of consecutive time steps that $hasBall(Player_{own})$ is *True* for one of our players. $Ball_{opp}$ is the number of consecutive time steps that $hasBall(Player_{opp})$ is *True* for one of the opponent players. The result is a real number in the interval $[-1, 1]$. A positive value of the metric indicates that our team has the control of the ball more than the opponent team. This can be interpreted as our team is dominating the game.

3.2.3.3. Score Difference. Score difference (SD) is calculated from the scores of the teams. Since the aim is to win the game (having scored more than opponents and preventing opponents from scoring against us), the difference of scores gives us the information about the current situation of the game. Score difference is given in (3.9).

$$SD = Score_{own} - Score_{opp} \tag{3.9}$$

The result is an integer indicating the dominion over game so far. A positive value indicates that our team is closer to win the game than the opponent where a negative value means the opposite.

3.3. Role Assignment

The role assignment is done in the reactive layer according to the objective functions of instantaneous metrics. Seven roles are defined:

- ***Goalie :*** Goalie role is assigned to a player at the beginning of the game and is not changed throughout the game.
- ***Passive Defender :*** The player which is both the nearest to the line between ball and own goal and near to own goal is assigned to this role. The passive defender gets a position on the line between the ball and center of the own goal area, on the center of that line.
- ***Active Defender :*** Aims to challenge the opponent player possessing the ball and try to get the control of the ball. The player which is both nearest to the line between ball and own goal, and nearest to the ball is assigned to this role.
- ***Supportive Defender :*** The aim of this role is to estimate the possible opponent player that the ball owning opponent player would pass the ball and hence, it tries to intercept the ball.
- ***Attacker :*** This is the player with the ball. It tries to dribble the ball to a position suitable for shooting, shoots the ball or passes the ball to a teammate at a better location.
- ***Supporter :*** The player in this position tries to get a position where an unsuccessful shoot / pass attempt would causes the ball to move to.
- ***Midfielder :*** The player in this position tries to get a position in a way that

increases the chance of keeping the control of the ball if primary attacker loses the ball due to an encounter with opponent team, due to an unsuccessful action (dribble/pass/shoot).

Since there are seven defined roles but only four players, some sort of mutual exclusion should be applied among the roles. Each team should have one goalie so we consider the goalie role as statically assigned to a player. For the remaining six roles, following criterion is checked:

- If our team has the ball (checked using *hasBall* predicate), players are evaluated for *Attacker*, *Supporter* and *Midfielder* roles.
- If opponent team has the ball, players are evaluated for *Passive Defender*, *Active Defender* and *Supportive Defender*.

3.3.1. Objective Functions for Roles

3.3.1.1. Goalie. Since Goalie role is assigned statically at the beginning of the game and do not change throughout the game, there is no objective function for measuring the fitness of player to that role.

3.3.1.2. Active Defender. If the own team does not have the ball possession, The player which is nearest to the ball is assigned to Active Defender role.

$$Obj_{ActiveDefender} = \sqrt{(x_{Player} - x_{Ball})^2 + (y_{Player} - y_{Ball})^2} \tag{3.10}$$

3.3.1.3. Passive Defender. If the own team does not have the ball possession, The player which is nearest to the center of own goal is assigned to the Passive Defender role.

$$Obj_{PassiveDefender} = \sqrt{(x_{Player} - x_{OwnGoal})^2 + (y_{Player} - y_{OwnGoal})^2} \tag{3.11}$$

3.3.1.4. Supportive Defender. Assignment of the supportive defender also depends on a set of criteria. After the active and passive defenders are assigned, if the remaining player is in its own half of the field, it is directly assigned to the supportive defender role. If the remaining player is in the opponent half and it has a high chance of score (according to the objective function of the shooting action), it is assigned to midfielder position for a possible counter-attack case.

3.3.1.5. Attacker. If our team has the ball possession, the player nearest to the ball is assigned to Attacker role.

$$Obj_{Attacker} = \sqrt{(x_{Player} - x_{Ball})^2 + (y_{Player} - y_{Ball})^2} \tag{3.12}$$

3.3.1.6. Supporter. Since the supporter gets a position symmetrical to the attacker with respect to the ball, the player nearest to the supporter point is assigned to the Supporter role.

$$x_s = x_a \tag{3.13}$$

$$y_s = y_b \pm |y_b - y_a| \tag{3.14}$$

$$Obj_{Supporter} = \sqrt{(x_p - x_s)^2 + (y_p - y_s)^2} \tag{3.15}$$

where, x_a and y_a are the coordinates of the Attacker, x_s and y_s are the coordinates of the supporter point, x_p and y_p are the coordinates of the player being evaluated, and

y_b is the Y coordinate of the ball.

3.3.1.7. Midfielder. After the Attacker and Supporter roles are assigned, the remaining player is directly assigned to the Midfielder role

3.4. Potential Fields for Motion Planning

In our approach, we used potential fields for low level motion planning. Each role and important objects have potential functions based on their properties. For example, the desired positions for the robots in different roles are represented with an attractive field located on the desired point. The obstacles are represented with repulsive fields. In our case, obstacles are field borders and own goal area (since it is prohibited for the players to spend more than three seconds in their goal area).

3.4.1. Goalie Field

Goalie tends to stay on the straight line between the middle of the goal and the ball and in front of the goal line. Formulation of goalie field is given in Figures 3.11 and 3.16.

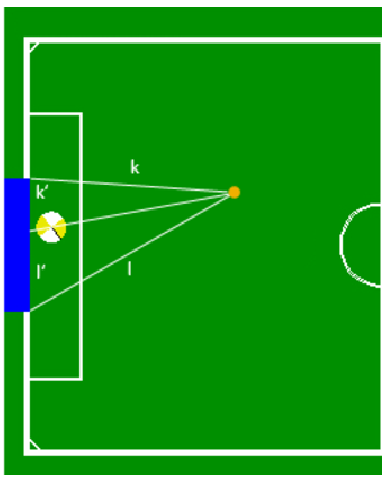

Figure 3.11. Potential Field definition for Goalie

$$\frac{k}{l} = \frac{k'}{l'} \tag{3.16}$$

$$y = y_{GoalTop} + k' \tag{3.17}$$

$$x = RobotRadius \tag{3.18}$$

where k is the distance from ball to the top side of own goal, l is the distance from the ball to the bottom side of the own goal, k' is the distance between top side of own goal and the intersection point of the straight line between the ball and own goal line which passes through the center of the goalie.

3.4.2. Passive Defender Field

Passive defender field is similar to goalie field in some aspects. The location of the attractive point is on the line between the ball and the center of own goal and at the middle point of the line. Formulation for passive defender field is given in Figure 3.12 and Equation 3.19.

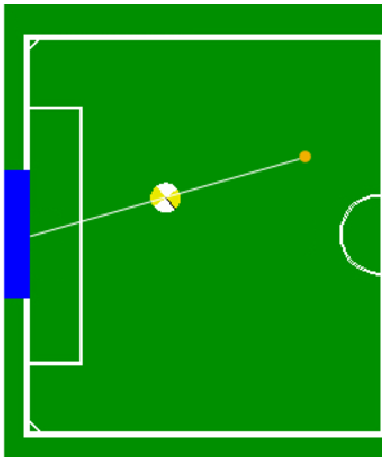

Figure 3.12. Potential Field definition for Passive Defender

$$x = \frac{(x_{Ball} + x_{GoalCenter})}{2} \tag{3.19}$$

$$y = \frac{(y_{Ball} + y_{GoalCenter})}{2} \tag{3.20}$$

$$\tag{3.21}$$

where x_{Ball} and y_{Ball} are coordinates of the ball and $x_{GoalCenter}$ and $y_{GoalCenter}$ are coordinates of the center point of own goal.

3.4.3. Active Defender Field

Active defender tries to gain the ball possession by challenging aggressively for the ball on the opponent attacker. Active defender field consists of an attractive potential field in which the center is the ball position. Since the opponent attacker will probably try to shoot the ball to our goal, a circular alignment field directed towards the opponent attacker is used to increase the possibility of interception. Definition of active defender field is given in Figure 3.13.

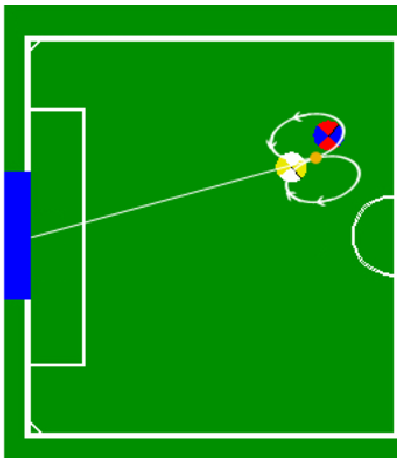

Figure 3.13. Potential Field definition for Active Defender

3.4.4. Supportive Defender Field

Supportive Defender aims to intercept the ball by estimating the possible opponent player that the opponent attacker would pass the ball and taking position accordingly on the line between the ball and candidate pass receiver and on the center of that line. The second nearest opponent player in our half of the field is considered as the candidate player. Formulation for supportive defender field is given in Figure 3.14 and Equation 3.22.

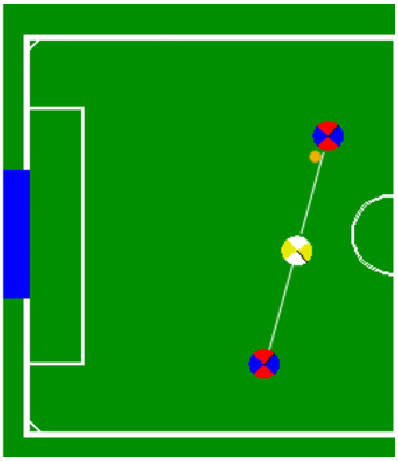

Figure 3.14. Potential Field definition for Supportive Defender

$$x = \frac{(x_{Ball} - x_{Candidate})}{2} \tag{3.22}$$

$$y = \frac{(y_{Ball} - y_{Candidate})}{2} \tag{3.23}$$

where x_{Ball} and y_{Ball} are coordinates of the ball and $x_{Candidate}$ and $y_{Candidate}$ are coordinates of the candidate opponent player.

3.4.5. Attacker Field

The attacker always moves towards the ball and tries to get aligned in a way that the accuracy of dribbling / passing / shooting to a certain position is increased. The

attacker field consists of an attractive potential field located on the ball and a circular ball field that has a direction pointing to the destination position. The definition of attacker field is given in Figure 3.15 and Equation 3.24.

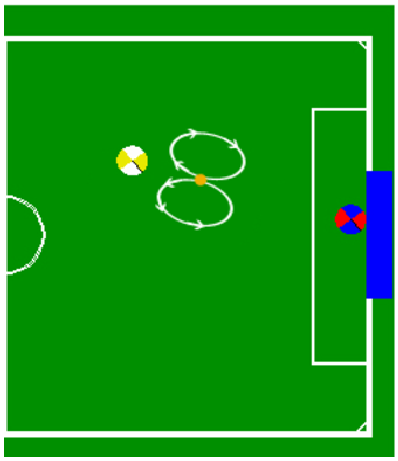

Figure 3.15. Potential Field definition for Attacker

$$x = x_{Ball} \tag{3.24}$$

$$y = y_{Ball} \tag{3.25}$$

where, x_{Ball} and y_{Ball} are the coordinates of the ball.

3.4.6. Supporter Field

The supporter takes the position on the symmetrical point of the Attacker with respect to the horizontal line that divides the field into two equal halves. Formulation for supporter field is given in Figure 3.16 and Equation 3.26.

$$x = x_{Attacker} \tag{3.26}$$

$$y = -y_{Attacker} \tag{3.27}$$

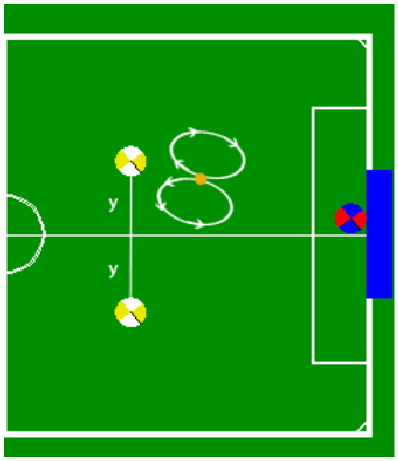

Figure 3.16. Potential Field definition for Supporter

where, $x_{Attacker}$ and $y_{Attacker}$ are the coordinates of the attacker.

3.4.7. Midfielder Field

The midfielder tries to get a position on the y position of the ball and having a distance to the ball equal to the average of the distances of the attacker and the supporter to the ball.

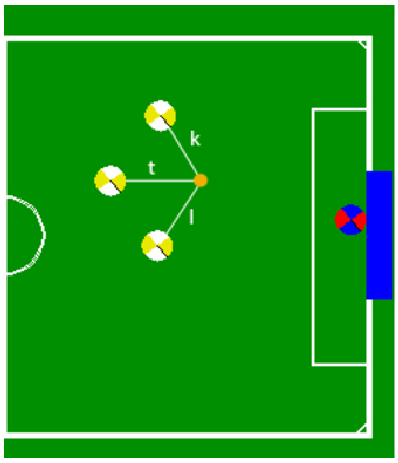

Figure 3.17. Potential Field definition for Midfielder

$$k = \sqrt{(x_{Attacker} - x_{Ball})^2 + (y_{Attacker} - y_{Ball})} \tag{3.28}$$

$$l = \sqrt{(x_{Supporter} - x_{Ball})^2 + (y_{Supporter} - y_{Ball})} \tag{3.29}$$

$$t = \frac{k + l}{2} \tag{3.30}$$

$$x = x_{Ball} \pm t \tag{3.31}$$

$$y = y_{Ball} \tag{3.32}$$

where, $x_{Attacker}$ and $y_{Attacker}$ are the coordinates of the attacker, $x_{Supporter}$ and $y_{Supporter}$ are the coordinates of the supporter, and x_{Ball} and y_{Ball} are the coordinates of the ball.

3.4.8. Ball Field

The Ball Field consists of an attractive field located at the position of the ball and two circular fields right above and below the line between the ball and the target position. The circular fields are similar to limit cycle fields [20]. The direction of the circular fields is towards to the target position.

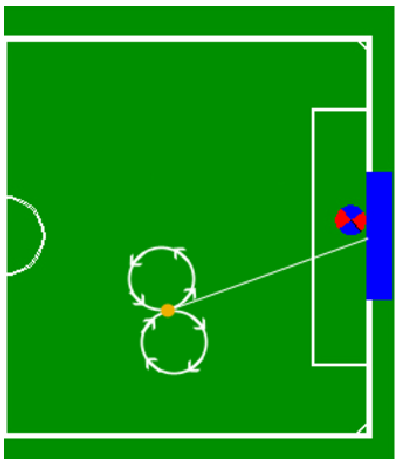

Figure 3.18. Potential Field definition for Ball

$$x = x_{Ball} \tag{3.33}$$

$$y = y_{Ball} \tag{3.34}$$

where, x_{Ball} and y_{Ball} are the coordinates of the ball.

3.4.9. Constraint Fields

Constraint Fields are a set of repulsive fields for representing the obstacles and borders of the fields. Field borders are modelled as line segment fields (Figure 3.19). Line segment fields are combinations of infinite number of field generator points in a line segment shape where only the nearest point affects the robot. Four types of line segment fields are used to represent constraints in the game:

- Attractive Line Segment
- Repulsive Line Segment
- Left Through Right Line Segment
- Right Through Left Line Segment

An *Attractive Line Segment* consists of infinite number of attractive points along the specified line. A *Repulsive Line Segment* consists of infinite number of repulsive points along the specified line. A *Left Through Right Line Segment* behaves like an attractive line segment, if the player is at the left side of the segment and behaves like a repulsive line segment, if the player is at the right side of the segment. A *Right Through Left Line Segment* is the opposite of a left through right line segment.

In our approach, there are two constraints related with the field definition and game rules, and one constraint for game level strategy determination. The field borders and the borders of home goal area are represented with repulsive line segments. For the game strategy determination, one left through right line segment is located at the

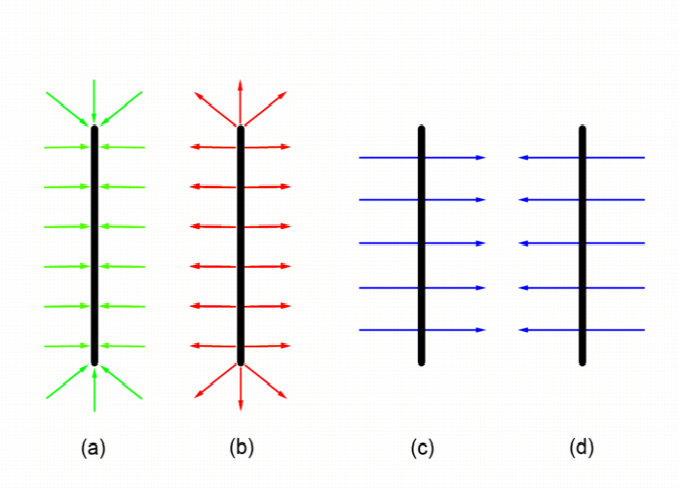

Figure 3.19. Line Segment Fields: (a) Attractive, (b) Repulsive, (c) Left → Right, (d) Right → Left

middle of the home half of the field and one right to left line segment is located at the middle of the opponent half of the field. Magnitude of this line segments determines the aggression level of our game strategy. For example, if the left half of the field is our home half and the magnitude of left through right line segment is smaller than the right through left line segment, our game strategy will be less aggressive, in other words, more defensive.

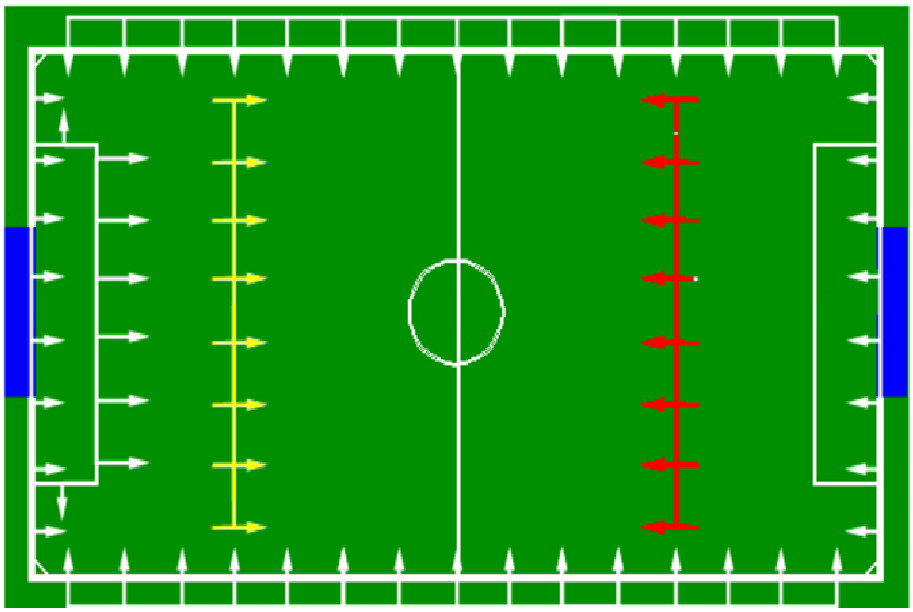

Figure 3.20. Potential Field definition for constraints

3.5. Reactive Layer

The reactive Layer is responsible for low level actions. Instantaneous metrics are calculated in this layer and objective functions consisting of instantaneous metrics for roles are evaluated. The roles are then assigned to the robots according to the values of objective functions. The play layer uses role and position information as well as play level metrics for the robots to build up short-term motion planning for each player. The result of this planning is a vector indicating the direction and speed of the movement. Resultant vector is combined with constraint potential fields for the game field and game rules (teammates, opponent players, field borders, own goal area, etc.) and then modify the combined vector according to the game layer metrics. The motion command is then sent to the actuators.

3.6. Play Layer

After all the players are assigned with proper roles, they should act in a way that they get the possession of the ball if currently the opponent team has the ball or if our team currently has the possession of the ball, the players should act in order to move the ball to a proper position for scoring. The game can be divided into two main phases:

- Defensive Phase
- Offensive Phase

Defensive Phase simply can be summarized as getting the ball possession. In defensive phase, players act upon their role definitions only. In other words, passive defender maintains a position between the ball and own goal while the active defender tries to get the possession of the ball. *Offensive Phase* starts with the gaining of ball possession. Steps of offensive phase is as follows:

- Gaining the ball possession
- Building up the play

- Final touch / shoot

Final touch/shoot to the opponent goal has a set of prerequisite conditions to be satisfied such as the clearance between the ball and the opponent goal and proper distance and orientation of the player having ball possession to the opponent goal. In order to satisfy prerequisite conditions, a sequence of behaviors should be executed. Although we have some metrics for evaluating the possible performance of shooting to the goal or passing to a teammate, all of these metrics are instantaneous metrics so they are calculated by using the snapshots for a time-step. Finding the optimal values of the metrics requires a greedy search. However, it can be said from the experiences in real soccer that the team may have to act towards improper values of the instantaneous metrics. For dealing with such a problem, we need some heuristics. A *Play* can be defined as performing a sequence of predefined behaviors according to defined conditions. Knowledge-based approaches are being used in real soccer based on human expertise. Dylla *et al.* [21] have initiated a qualitative soccer formalism for robot soccer. They proposed a top-down approach to the soccer knowledge, following the classical soccer theory. In this point of view, play layer can be considered as the deliberative part of the system.

Since each player is autonomous and performing a *Play* requires cooperation, each player should agree on the play to be performed. In our approach, the player with the ball (attacker) decides which play to be performed and broadcasts the ID of the play to the teammates. Each player has the definitions of plays so a player can easily perform motion planning since the role of the player and the required motion of that role in the specified play is known. We followed Dylla *et al.* [21] and defined two predicates as discussed in (3.2.2.1) and (3.2.2.2) and four actions for specifying the plays:

- $hasBall(Player)$
- $isReachable(Position_1, Position_2)$

where predicate $hasBall(Player)$ indicates whether *Player* has the ball or not and predicate $isReachable(Position_1, Position_2)$ denotes whether $Position_2$ is reachable from $Position_1$ (i.e., the path between $Position_1$ and $Position_2$ is clear). Defined four actions are:

- $moveTo(Player, Position)$
- $dribbleTo(Position)$
- $shoot(Position)$
- $pass(Player)$

moveTo(Player, Position) simply replaces the potential field of *Player* with an attractive potential field on *Position* making the player move towards the *Position*. *dribbleTo(Position)* makes the *Player* to dribble with the ball to *Position*. *dribbleTo* is applied to the player with the ball.

In *shoot(Position)*, *Player* with the ball shoots the ball to the given *Position*. When $pass(Player)$ action is set, player with the ball kicks the ball to $Player$.

In actions involved with the ball (dribbling, kicking and passing), ball fields are used for maintaining the proper alignment of the player with ball.

The position space is quantized into nine regions in order to keep the position definitions simple. The field is quantized as in Figure 3.21.

The defined predicates and actions allow us to formalize the soccer tactics and to define the plays. The following offensive plays are defined:

- Counter-Attack
- Change Wings
- Attack
- Prepare Attack

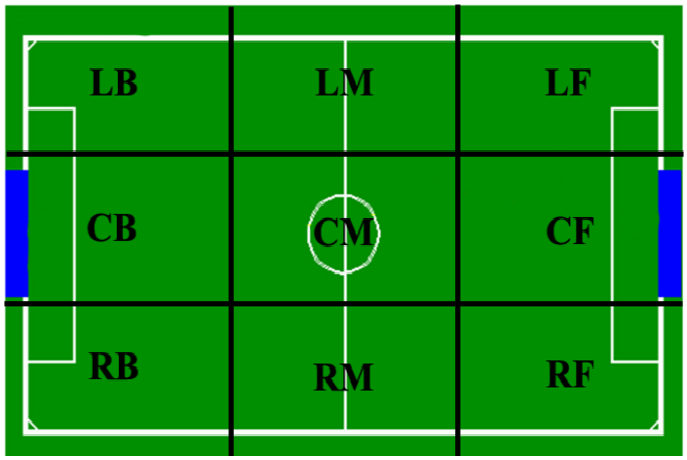

Figure 3.21. Quantization of the game field into regions

Attack play tries to form a position combination for our players in such a way that our attacker is located on a suitable position that maximizes its chance to score and our supporter and midfielder are located on proper positions defined in their roles. It is clear that it would not be so easy to obtain a suitable formation since opponent players try to get the ball and act as obstacles.

Change Wings play aims to change the location of the ball from one side of the field to the other side. It is used for finding an empty area on the field to move on to. *Attack* play employs the *Change Wings* play to move with the ball to the opposite side of the field if it is less occupied by the opponents.

Prepare Attack is used when offensive phase begins (i.e. when our defenders gained the possession of the ball). The aim of this play is to move the ball to the opponent side of the field as quickly as possible without losing any advantage on the control of the ball. Depending on the distances and distribution of the opponents, one or two remaining players (other than the attacker) crosses the half field line at the opposite sides. The attacker then passes the ball to the player with the most advantageous position such as reachability and clearance from the attacker and its clearance and reachability to the opponent goal.

```
startDribble(region[CF]);
waitFor(isReachable(Supporter, region[LF])  ||
        isReachable(Midfielder, region[RF]) ||
           isReachable(Attacker, region[CF]);
if isReachable(Supporter, region[LF]) then
moveTo(Supporter, region[LF]);
pass(Supporter);
else if isReachable(Midfielder, region[RF]) then
  moveTo(Midfielder, region[RF]);
pass(Midfielder);
else if isReachable(Attacker, region[CF]) then
shoot(region[CF]);
end if;
```

Figure 3.22. Specification of Counter-Attack Play

Counter-Attack takes place when we gained the possession of the ball and one or more of our players are in the opponent half. The ball owner (or attacker) tries to pass the ball to one of the players in the opponent half immediately. Players in the opponent half change their positions to increase reachability and clearance for the attacker to themselves.

If there is no appropriate *"Play"* to execute, players act according to their potential field definitions.

waitFor statement produces no output until the specified condition is true. If no action is set for a player in a play specification, player moves according to its role definition. In other words, an action from the play layer *subsumes* the action from the role definition.

3.7. Game Layer

Game level strategy is controlled by the *Game Layer*. Since there is no widely accepted game level optimal strategies for soccer, many different strategies may be used. In our approach, a game level strategy is a mapping from game level metrics to the defined constraint fields. The attack / defense schema is selected based on *hasBall* predicate for team and appropriate roles are assigned depending on the ball possession. *hasBall* for the team is determined as in Equation (3.35):

$$hasBall = \begin{cases} 1, \text{if } \exists Player_i^{own},\ hasBall(Player_i^{own}), \\ 0, \text{otherwise} \end{cases} \tag{3.35}$$

Game level strategy takes place when the motion planning for players is done and the desired motion vectors are calculated. The resultant motion vectors are then modified according to the game level strategy by adding more constraint fields. Two line segment fields are used for game level strategy determination. In the left half of the field, a left through right line segment field is placed at the middle of the field and in the right half of the field, a right through left line segment field is placed. According to the game level strategy, the magnitude of these line segment fields are changed so the aggression level of the team can be adjusted. Setting a high magnitude value for the left through right line segment field and setting a low magnitude value for the right through left line segment field yields a more aggressive team in which the players tend to favor attacking over defensing. Contrarily, setting a low magnitude value for the left through right line segment field and a high magnitude value for the right through left line segment field, our team will act in a more defensive manner than attacking. Value setting for magnitude functions can be done in many different ways. In our work, we used the formula in Equation 3.36:

$$\mu_1 + \mu_2 = 1 \tag{3.36}$$

where, μ_1 is the coefficient for the left through right line segment field and μ_2 is the coefficient for the right through left line segment field. Since the sum of the coefficients is fixed, determination of one coefficient is enough.

4. EXPERIMENTAL RESULTS

A set of experiments were performed in order to first evaluate the validity of the proposed metrics and then evaluate the performance of algorithm built on top of that metrics. Teambots [22] simulator is used to perform such experiments. Teambots is a 2D Java-based simulator that allows us to simulate MIROSOT small-size environment. We have used Teambots for gathering metric data, evaluation of metrics and experiments for the new algorithm. In this section, first, brief information about the simulator is presented. Then the details of experiments are given.

4.1. Teambots

Teambots is a Java™based simulator for multi-robot teams developed by Balch [22]. It is also suitable for different multi-robot simulations for tasks like foraging and mapping. Each agent is controlled by a Java™program. The simulation environment and simulation parameters are set through a configuration file. The simulator calls *takeStep()* function in the controller class at each time step.

Since Teambots uses a configuration file for specification of simulation properties, it is possible to run a large number of consecutive experiments with different parameters. However, since physical simulation of the robots cannot reflect the real properties of the robots, the result of the experiments should be tailored for real robots.

4.2. Experiments for Evaluating Metrics

For the evaluation of defined metrics, a total of 200 games were played against four different opponents. MarketTeam [17] is used as the home team in these games. MarketTeam uses a free market-driven role assignment scheme depending on the cost functions for the roles. Each robot evaluates the cost functions for each role and the robot offering the minimum cost is assigned to that role. Cost functions consist of linear combination of a set of metrics in which each metric is weighted with a coefficient. The

proper coefficients for metrics are obtained with genetic algorithms.

In order to reveal the performance of the opponent teams in all aspects and to eliminate ceiling and floor effects in evaluating the performance of our own team, we have tried to use stratification in selecting the opponent teams so we choose both weak, moderate and powerful teams as opponents. The selected opponents are:

- ***AIKHomoG :*** Is the strongest built-in team in Teambots. It uses dynamic role assignment and potential fields for motion planning.
- ***RIY Team :*** Is the predecessor of Market Team and our approach. RIY uses a dynamic role assignment strategy based on simple metrics like distance to the ball. Potential fields are used in low level motion and coefficients obtained from a genetic algorithm training are used in combination of different fields to obtain resultant motion vector.
- ***Kechze :*** Uses dynamic role assignment and uses geometric calculations as the role assignment criteria. We considered Kechze and RIY as moderate teams.
- ***SchemaNewHetero :*** Uses perceptual and motor schemas for different roles. SchemaNewHetero is the weakest opponent in our experiments.

4.2.1. Decomposition of the Game Data

After the games are played and the position data for the players and the ball are recorded, each game is divided into *episodes* which starts with a kick-off and ends with either a score or end of half or end of game whistle. Episodes ending with own scores are marked as positive examples and episodes ending with opponent scores are marked as negative examples. Episodes ending with end of half or end of game whistle are ignored. At the end of 200 games, 81 negative and 1016 positive episodes were recorded. Each episode is then divided into smaller sequences of time-steps that are separated by a touch (or kick) to the ball. These sub-episodes are also marked as positive/negative examples depending on which team has touched the ball at the end of the sub-episode. If the ball is kicked by own team and the previous kick was performed by the opponent team, that sub-episode is marked as a *Positive* example. If the ball

is kicked by opponent players and the previous kick was made by home players, that sub-episode is marked as a *Negative* example. The sub-episodes that are started and ended with the kicks of same team are ignored. Then, the marked sub-episodes are used to evaluate metrics related to the ball possession.

4.2.2. Metric Validation

Proposing metrics is a challenging task but it is even harder to evaluate the performance of a metric. We use metrics to obtain quantitative information about the environment but how can we be sure that the metric we proposed really *measures* the property it is supposed to measure. So we are confronted with another challenging problem: Metric validation. In order to consider a metric as *informative*, the metric should show the same trends in the same situations. For example, we can propose the *distance to the ball* metric for assessing the probability of getting the control of the ball. However, distance might not be the right indicator. So we should check whether the distance metric has the same trends in positions having the same ending (our team got the control of the ball, or opponent team got the control of the ball). Due to noise and sudden changes in positions of ball and other players, recorded metric data contain noise making the observation of trends in metric data difficult. In order to extract trends in recorded noisy data, some smoothing algorithms are applied to the recorded data. We have tried two smoothing algorithms on the recorded metrics:

- 4253h,Twice Smoothing
- Hodrick-Prescott Filter

4.2.3. 4253h, Twice Smoothing

In 4253h, Twice algorithm, running median smoothers with window sizes 4, 2, 5 and 3 are applied consecutively. Then *Hanning* operator is applied. Hanning operator replaces each data point P_i with $\frac{P_{i-1}}{4} + \frac{P_i}{2} + \frac{P_{i+1}}{4}$. Then the entire operation is repeated [23]. Performing two or three consecutive 4253h, Twice resulted in great reduce in noise but the trend extraction is still hard in resultant smoothed data.

4.2.4. Hodrick-Prescott Filter

Hodrick-Prescott filter is proposed for extracting underlying trend in macroeconomic time series [24]. In the Hodrick-Prescott (HP) Filter approach, the observable time series y_t is decomposed as:

$$y_t = g_t + c_t \tag{4.1}$$

where g_t is a non-stationary time trend and c_t is a stationary residual. Both g_t and c_t are unobservable. We think y_t as a noisy signal for the g_t. Hence, the problem is to extract g_t from y_t.

HP Filter solves the following optimization problem:

$$\underset{\{g_t\}_{t=1}^{T}}{Min} \sum_{t=1}^{T}(y_t - g_t)^2 + \lambda \sum_{t=2}^{T}[(g_{t+1} - g_t) - (g_t - g_{t-1})]^2 \tag{4.2}$$

where λ is a weight for a signal against a linear time trend. $\lambda = 0$ means that there is no noise and $y_t = g_t$. As λ gets larger, more weight is allocated for the linear trend. So as $\lambda \to \infty$, g_t approaches to the least squares estimate of y_t's linear time trend. Selecting the value of λ is another design problem. In our work, we used 14400 as the value of the λ which is used to smooth monthly data in original implementation.

4.2.5. Metric Player

A software called *Metric Player* for replaying the games and displaying the metric data is developed for examining the trends in the metrics / metric combinations / objective functions (Figure 4.2). Metric Player is developed with Microsoft Visual Studio .NET 2003. The software allows the user to replay an episode in real time

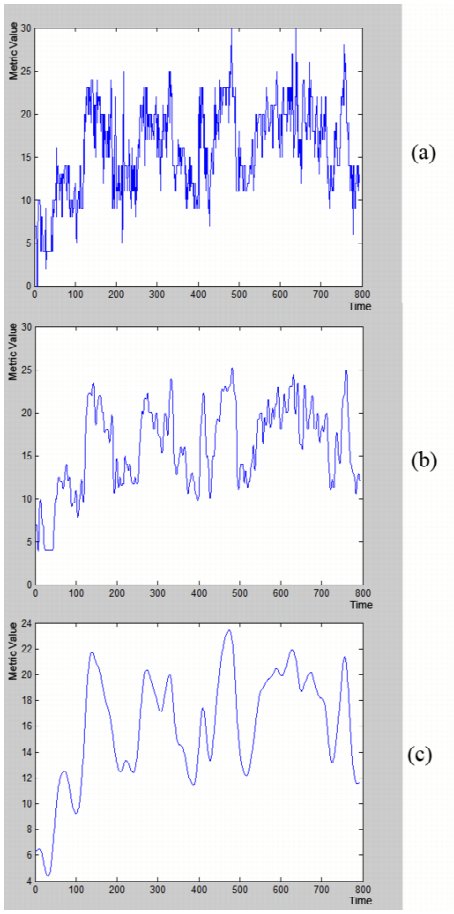

Figure 4.1. Smoothing: a) Raw data, b) 4253h,Twice, c) Hodrick-Prescott Filter

or frame by frame and simultaneously display the selected metric plot for the entire episode so that the user can examine the trends as the episode advances. Also, it has an indicator showing the current value of the metric for that time step in the episode. These features help the user to investigate the consistency of the metrics.

The metric player reduces the metric analysis time dramatically since it allows to determine threshold values of metrics by frame-by-frame playback and lets the user follow the change trends in the metrics. For examining a metric, first the episode that the metric belongs to is browsed and loaded. Then the desired metric is selected and loaded from the list of metrics populated for the loaded episode. The canvas displaying the metric plot is automatically scaled for the minimum and maximum values of the selected metric. Figure 4.2 shows an example episode and pairwise separation metric data smoothed with HP filter.

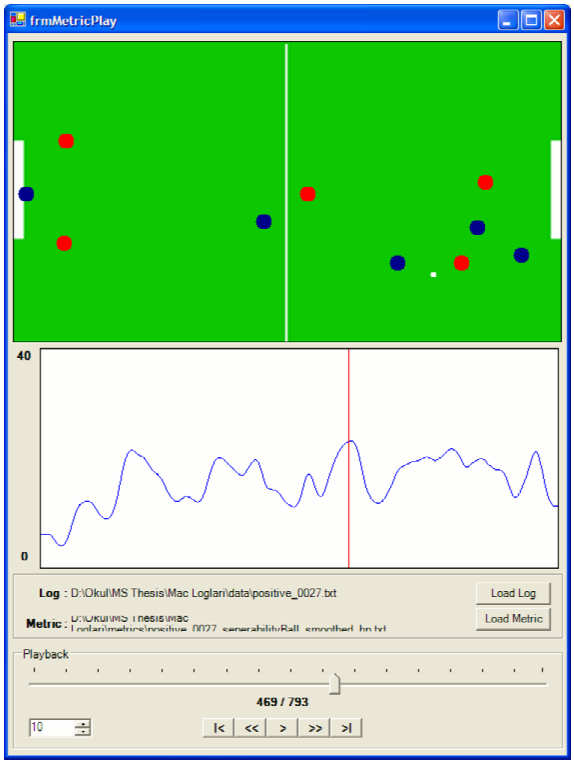

Figure 4.2. A snapshot from Metric Player

The calculated and smoothed metric data for recorded episodes are processed to test the existence of the statistical correlation among the metric data recorded in similar situations. In our work, we used the scored team as the similarity measure among the situations.

In Figure 4.3, all own and opponent kicks are marked on the plot of the metric. Bold spikes denote own kicks and narrow spikes denote opponent kicks. We ignore the consecutive kicks performed by the same team so, Figure 4.4 only shows the kicks in which the team with the ball possession is changed. In Figure 4.4, bold spikes denotes the kicks that are performed by our team and preceding by an opponent kick and, narrow spikes denotes the kicks that are performed by the opponent team and preceding by an own kick. In order to test the correlation among the sub-episodes with the same mark (positive or negative), a straight line is fitted on metric data in the sub-episode by using Least Squares Fitting. Then, the possible correlation between the mark of the sub-episode and the sign of the first derivative of the fitted line (i.e. slope of the line) is investigated. It is expected that the signs of the slopes of fitted lines on

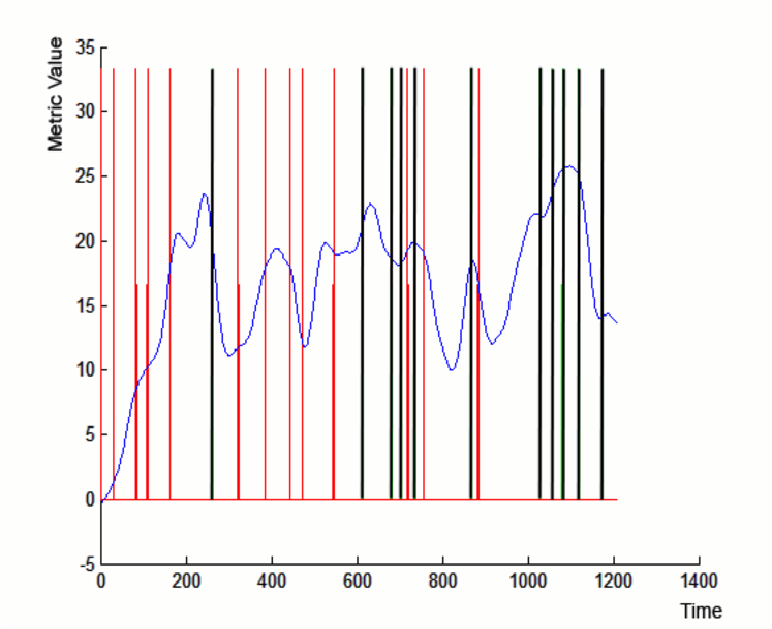

Figure 4.3. An example Pairwise Separation of the Ball metric with all the own and opponent kicks

the metric data in sub-episodes with the same mark are the same. Since sub-episodes separated by the kicks are used to analyze the ball possession, the mentioned test for checking the consistency of a metric is performed for a set of 20 randomly selected sub-episodes and for the following metrics:

- Area of the Convex Hull
- Density of the Convex Hull
- Pairwise separation of the Ball
- Vicinity Occupancy for the Ball

In Figure 4.5, fitted lines on the pairwise separation of the ball metric data between two kicks can be seen. It is seen in the figure that the fitted lines to the positive sub-episodes have positive slopes where the fitted lines to the negative sub-episodes have negative slopes.

Table 4.1 shows that the pairwise separation of the ball metric has a positive correlation with the sub-episode mark. Whenever the metric shows an increasing trend,

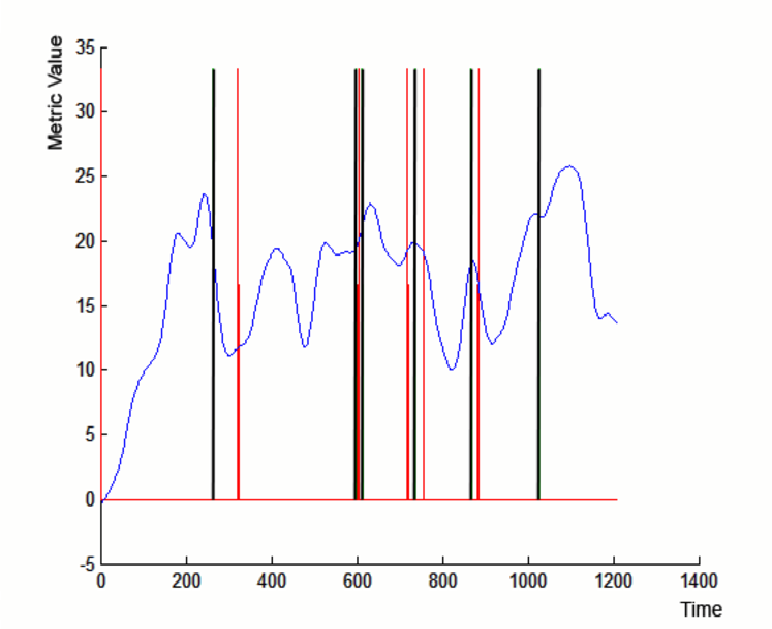

Figure 4.4. An example Pairwise Separation of the Ball Metric with positive and negative kicks

Table 4.1. The Kick-Slope distribution for Pairwise Separation of the Ball

	Own Kick	**Opponent Kick**
Positive Trend	94	27
Negative Trend	19	50

our own team performs a kick and since performing a kick requires the ball possession, it can be said that if the pairwise separation of the ball metric shows an increasing trend, our own team has the ball possession.

The Kick-Slope distribution tables for the remaining metrics are given in Tables 4.2 - 4.6.

Table 4.2. The Kick-Slope distribution for Vicinity Occupancy for the Ball

	Own Kick	**Opponent Kick**
Positive Trend	74	67
Negative Trend	35	48

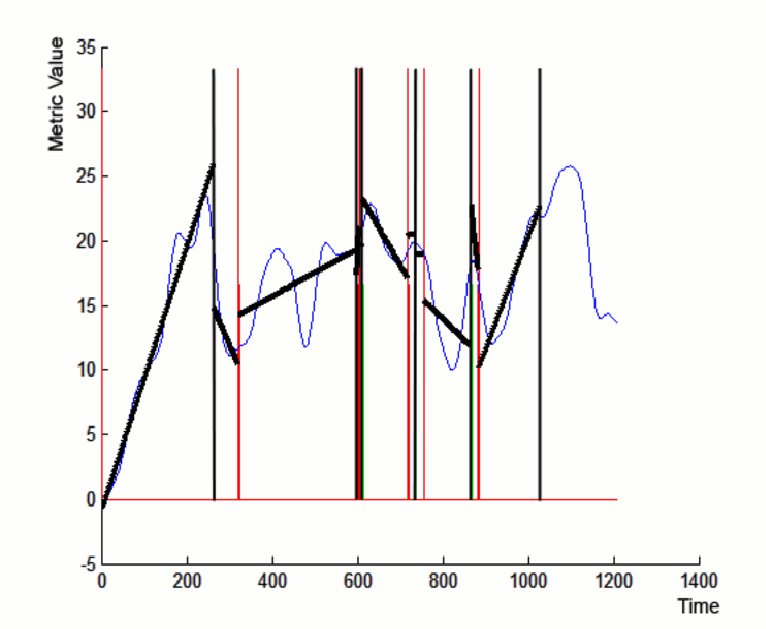

Figure 4.5. After fitting a Least-Squares Line to the metric

Table 4.3. The Kick-Slope distribution for the Density of the Convex Hull for Our Own Team

	Own Kick	**Opponent Kick**
Positive Trend	78	63
Negative Trend	45	38

According to the metric evaluation results, only the pairwise separation of the ball metric is consistent. Other metrics did not show similar trends in similar situations. It should be noted that this evaluation is valid only for the used hyperparameters of the system. Investigation of optimal hyperparameters for the system is beyond the scope of this work, so we conclude that only the pairwise separation of the ball metric can be used to measure the ball possession. Whenever the metric has a positive first derivative, it means that our team will kick the ball. By using this, it can be said that keeping the value of this metric as high as possible is desired.

Table 4.4. The Kick-Slope distribution for the Density of the Convex Hull for the Opponent Team

	Own Kick	**Opponent Kick**
Positive Trend	66	46
Negative Trend	54	51

Table 4.5. The Kick-Slope distribution for the Area of the Convex Hull for Our Own Team

	Own Kick	**Opponent Kick**
Positive Trend	75	66
Negative Trend	38	45

4.3. Experiments for Evaluating the Algorithm

Once the metrics are selected, objective functions for role assignments are built and game level strategy is set, a set of experiments are carried out in order to evaluate the performance of the proposed algorithm. 20 games are played against each of the the following five opponents:

- AIKHomoG
- Kechze
- RIYTeam
- SchemaNewHetero
- MarketTeam

MarketTeam is the free-market driven team in which the coefficients for cost function of roles are optimized by using Genetic Algorithms [17]. The other opponents are described in Section 4.2. Game level metrics are used to compare the performances of the algorithms that both our team and opponent teams uses.

Table 4.6. The Kick-Slope distribution for the Area of the Convex Hull for the Opponent Team

	Own Kick	**Opponent Kick**
Positive Trend	58	59
Negative Trend	52	50

The results of the games are given in two tables. In Table 4.7, the number of wins, losses, draws, for goals and against goals are given. In Table 4.8, the distribution of the time that the ball has spent in own area, center and opponent area (Figure 3.10) in percentage are given.

Table 4.7. Results of the games

Team	**Wins**	**Losses**	**Draws**	**For Scores**	**Against Scores**
AIKHomoG	16	0	4	27	3
Kechze	9	1	10	14	2
RIYTeam	19	0	1	58	1
SchemaNewHetero	19	0	1	80	7
MarketTeam	4	0	16	5	1

Table 4.8. Measured metrics for games

Team	**Own Possession**	**Opponent Possession**	**Center**
AIKHomoG	24.073	49.27	26.655
Kechze	18.427	61.959	19.614
RIYTeam	17.179	62.508	20.312
SchemaNewHetero	25,125	51.526	23.349
MarketTeam	23.623	49.054	27.323

In the experimental games, the MericliTeam using our proposed approach outperformed all of the opponents. In a total of 100 games, our team has won 67 games and lost only one game. 32 games ended with a draw. Score per game ratio for for scores is 1.84 and 0.14 for against scores. The ball has spent at least two times more time in the opponent possession area in all the games. This indicates that our team

is in the attacking phase in the most of the game. Our team has not shown a good performance against its predecessor, the MarketTeam, even though it has not lost a game to the MarketTeam and won four games against the MarketTeam. Based of the experimental results, it can be said that our approach has outperformed the opponent teams including the predecessor MarketTeam.

5. CONCLUSIONS

Multi-robot systems have an increasing popularity since they provide higher performance due to parallelism, are immune to single-node failures due to redundancy and they allow complex tasks to be achieved by many simple (and cheaper) robots instead of a single, complex and expensive robot.

Multi-robot systems are suitable for foraging and exploration tasks (such as planetary exploration and mine sweeping) in which the mission should be accomplished at least for some degree. Also multi-robot systems show good performance in tasks that rely on parallel execution or cooperation by their nature.

Robot soccer is a good testbed for multi-robot systems since it contains many of the problems that a robot team may be confronted with (limited and noisy sensors, limited and noisy actuators, partially observable and very dynamically changing environment, etc).

Since controlling a single robot for performing complex tasks is a very hard and challenging problem, proposing a control architecture for a robot team for cooperating through proper task allocation is even harder.

In this work, a robust task allocation algorithm for four-legged robot soccer is presented. Main contributions of this work are:

- A set of novel metrics in three different time resolutions from measuring the ball possession to analyzing the level of dominance in game are proposed.
- A test for evaluation of *Informativeness* of the metrics is presented.
- A three-layered control approach for different time resolutions is presented.

There is a set of hyperparameters of the system. Finding the optimal values for the hyperparameters is a challenging problem and is beyond the scope of this work.

So in the experiments, we have used arbitrarily selected values for the hyperparameters. *MericliTeam* using the proposed approach outperformed the opponents in the experiments and lost only one game out of the hundred games.

In this work, we have proposed an architecture for the four-legged soccer domain but since the proposed architecture is very flexible, it can easily be tailored for the other robot soccer leagues by adjusting hyperparameters and defining appropriate soccer tactics in the play layer. Allowing the high level soccer tactics and game level strategies without sacrificing from the benefits of the reactive systems make it both powerful and flexible.

A set of metrics are proposed and evaluated for their informativeness and consistency. A novel contingency table based metric validation is used in metric evaluation process. For the used set of hyperparameters, only one metric has shown a consistent behavior in similar game situations.

5.1. Future Work

There are many hyperparameters in both metrics and task allocation algorithm. We have used handcrafted values for the hyperparameters but in fact they can be optimized via many different optimization techniques. Since such optimizations are beyond the scope of this work, they have been left as future works.

Metric evaluation (or metric validation) is a very important issue in performance evaluation and calculation of the fitness. We are working on the metric validation issue and detailed investigation of possible opportunities in development of such a standardized validation test is left as a future work.

The algorithm could not be tested on real AIBOs since we only have four robots so performance evaluation of the proposed algorithm on real robots and possible practical issues that should be handled are left as a future work. That issues includes tailoring the proposed metrics for partially observable and very noisy environment in real robots.

REFERENCES

1. Arkin, R. C. *Behavior-Based Robotics*, MIT Press, Cambridge, Massachusetts, 1998.

2. Zlot, R., A. Stenz, M. B. Dias, and S. Thayer, "Multi-Robot Exploration Controlled by a Market Economy," Proceedings of the *IEEE International Conference on Robotics and Automation*, May 2002.

3. Kaplan, K. *Design and Implementation of fast controllers for Mobile Robots,* M.S. Thesis, January 2003.

4. Veloso, M., S. Lenser, D. Vail, M. Roth, A. Stroupe, and S. Chernova, *CMPack-02: CMU's Legged Robot Soccer Team,* Carnegie Mellon University, Pittsburgh, October 2, 2002.

5. Köse, H., Ç. Meriçli, K. Kaplan and H. Levent Akın, "All Bids for One and One Does for All: Market-Driven Multi-Agent Collaboration in Robot Soccer Domain", *Computer and Information Sciences-ISCIS 2003, 18th International Sympsium*, Antalya, Turkey, Proceedings, LNCS 2869, pp. 529-536., November 2003.

6. Murphy, R. R., *Introduction to AI Robotics*, MIT Press, Cambridge, Massachusetts, 2000.

7. Meystel A., "Nested Hierarchical Control," In K. M. Passino and P. J. Antsaklis (Eds.), *Introduction to Intelligent and Autonomous Control,* Kluwer Academic Publishers, 1992.

8. Gat E., "On Three-Layer Architectures," In D. Kortenkamp, R. P. Bonnasso, and R. Murphy (Eds.), *Artificial Intelligence and Mobile Robotics,* AAAI Press, 1998.

9. Lyons D.M. and D.M. Hendricks , "A Practical Approach to Integrating Reaction and Deliberation," *Proceedings of the 1st International Conference on Artifical In-*

telligence Planning Systems, pp. 153-162, 1992.

10. Brooks R., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation RA-2,* pp. 14-23, April, 1986.

11. HORN, H. S. "Measurement of 'overlap' in Comparative Ecological Studies," *Amer. Natur.* Vol. 100: pp. 419-424, 1966.

12. Khatib O., "Real-time Obstacle Avoidance for Manipulators and Mobile Robots," *Proceedings of International Conference on Robotics and Automation (ICRA)*, pp. 500-505, 1985.

13. Dudek, G., M. Jenkin, E. Milios, and D. Wilkes, "A Taxonomy ofor Swarm Robots," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, Yokohama, Japan, pp. 441-47, 1993.

14. FIRA *http://www.fira.net/,* 2003.

15. Robocup Organization *http://www.robocup.org,* 2005.

16. H. L. Akin, *et al.*, "Cerberus 2003" Robocup 2003: Robot Soccer World Cup VII, *The 2003 International Robocup Symposium Pre-Proceedings*, pp.448, Padova, June 24-25, 2003.

17. Kose, H., K. Kaplan, C. Mericli and H. L. Akin, "Genetic Algorithms Based Market-Driven Multi-Agent Collaboration in the Robot-Soccer Domain", *FIRA Robot World Congress 2003*, Vienna, Austria, October 1 - 3, 2003.

18. Kose, H, U. Tatlidede, C. Mericli, K. Kaplan and H. L. Akin, "Q-Learning based Market-Driven Multi-Agent Collaboration in Robot Soccer," *Proceedings, TAINN 2004, Turkish Symposium On Artificial Intelligence and Neural Networks*, Izmir, Turkey, pp.219-228, June 10-11, 2004.

19. Tatlidede, U. Kaplan, K. Kose, H and Akin, H. L., "Reinforcement Learning for

Multi-Agent Coordination in Robot Soccer Domain", *Fifth European Workshop on Adaptive Agents and Multi-Agent Systems*, Paris, France, March 21-22, 2005.

20. Kim D.-H. and J.-H. Kim, "Limit-Cycle Navigation Method for Robot Soccer," *Proceedings of the 2002 FIRA Robot World Congress*, May, 2002.

21. Dylla, F. Ferrein, A. Lakemeyer, G. Murray, J. Obst, O. Rofer, T. Stolzenburg, F. Visser, U. and Wagner, T. "Towards a League-Independent Qualitative Soccer Theory for RoboCup", *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, Springer, 2004

22. Balch, T. "Teambots", *http://www.teambots.org,* 2000.

23. Cohen, P. R., *Empirical Methods for Artificial Intelligence*, MIT Press, Cambridge, Massachusetts, 1995.

24. Hodrick, R. J., and Prescott, E. C., "Postwar U.S. Business Cycles: An Empirical Investigation." *Journal of Money, Credit and Banking Vol. 29 (1)*, February 1997.