

OBSTACLE ADAPTIVE APPROACHES FOR DISTRIBUTED TASK ASSIGNMENT IN
AUTONOMOUS MOBILE ROBOTS

A Thesis by

Syam Kavala

Bachelor of Electronics and Communication Engineering, Andhra University, 2005

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

July 2009

© Copyright 2009 by Syam Kavala

All Rights Reserved

OBSTACLE ADAPTIVE APPROACHES IN DISTRIBUTED TASK ASSIGNMENT IN
AUTONOMOUS MOBILE ROBOTS

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Electrical Engineering.

Vinod Namboodiri, Committee Chair

Preethika Kumar, Committee Member

Krishna Krishnan, Committee Member

DEDICATION

To my empowering and loving parents,
and my dear friends.

“We are responsible for what we are, and whatever we wish ourselves to be, we have the power to make ourselves. If what we are now has been the result of our own past actions, it certainly follows that whatever we wish to be in future can be produced by our present actions; so we have to know how to act.” – Swami Vivekananda

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Professor Sejun Song and my co-advisor, Professor Kamesh Namuduri for their guidance, advice, encouragement and sharing their expertise. Their support encouraged me to think beyond my knowledge during my research and thesis at Wichita State University.

A special thanks to my 10th grade mathematics teacher Aruna, who is very caring, friendly and inspiring. Her teachings boosted my confidence and helped me to explore beyond my knowledge.

I would like to extend my gratitude to my committee members Dr.Krishna Krishnan and Dr. Vinod Namboodiri for their valuable time and consideration.

Thanks are also due to Harini, Praveen and Sree ramya, my colleagues who worked under Dr. Kamesh Namuduri during my masters, for the immense support, help, guidance, caring they showed me during my graduation at Wichita State University.

A special thanks to Mr. Murali Krishna kadiyala, Amithav and Prathi Krishna chaitanya chowdary for the valuable suggestions they gave me while I am working on my thesis which helped me to finish my graduation on time.

Finally, I would like to say thanks to all of my family and friends, particularly my loving parents and sister. Thank you to Monsa and Cathy Dugger who are also like my parents here. Thanks to my friend Nagalakshmi Vemuri for the immense help, caring and support she gave me during my graduation. Last but not the least, Thanks to my wonderful room mates Ravi Shankar, Yateendra, Ramakanth, Chaitanya, Rajesh,Araval Rao and Ashok for the support and suggestions they gave me during my graduation.

ABSTRACT

It has many challenges to assign a group of mobile robots to individual targets according to the specific constraints. In addition to the group behavior constraints (one-to-one or one-to-many) of the task assignment, some of the performance constraints include (1) proximity from robot to target (2) suitability of robot in performing a task and (3) quality of connectivity among the robots. Due to the computational complexities and the nature of the dynamical systems, the task assignment approaches have been developed as distributed and dynamical systems approaches with various simplified assumptions.

In this thesis, first, I investigate one of the most recently proposed distributed task assignment approaches (Peter Molnar's approach) that combines target assignment and motion planning in order to minimize the robots travelling distance and overhead cost for its one-to-one target assignment. Second, I find that the approach does not provide an efficient path finding algorithm in the environment with obstacles. It simply uses proximity sensors to direct the robots away from obstacles. Based upon the observation, third, I propose efficient task assignment approaches to minimize the robot's travel distance and overhead cost in an environment with one immobile obstacle of any shape and size.

In detail, the thesis addresses 1) obstacle modeling and simplification: Initially the vertices of an obstacle were provided to the robots. The robots reconstruct the obstacle to a rectangle shape that encircles the original obstacle, 2) obstacle decomposition for an adjustment: The reconstructed obstacle is further decomposed in case if there was a robot or target object present within the reconstructed area, and 3) optimal target path calculation: Two approaches are designed for calculating distance from robot to target by taking the reconstructed obstacle into account. Approach 1 calculates the shortest distance

from robot to target along the perimeter of the obstacle. Approach 2 further optimizes the path by connecting the robot and target to the shortest distance vertices of the obstacle. The computational overhead and task assignment efficiency of the proposed approaches are compared via MATLAB simulations.

TABLE OF CONTENTS

Chapter		Page
I.	INTRODUCTION	1
II.	LITERATURE REVIEWED	4
	Communication fault tolerance in distributed robotic systems	4
	Generalized Geometric Triangulation Algorithm	6
	Flexible Autonomous Robotic Task Scheduling using Advanced Risc machines	8
III.	HARDWARE AND SOFTWARE USED	9
	Hardware Used	9
	Garcia robots- out of the box	9
	Sensors and their uses	10
	Stargate Board	11
	Programming on Garcia robots	11
	Software Used	12
IV.	APPROACH TO CALCULATE PREFERENCE VALUE IN THE PRESENCE OF OBJECT IN A GIVEN ENVIRONMENT	13
	Initial Preference value calculation	14
	Object Modeling	16
	Approach to distance calculation in the presence of object	18
	Approach 1 Drto calculation	19
	RT line	23
	Robot Location	24
	Robot Entry point on obstacle(REP)	24
	Robot Leaving Point on obstacle(RLP)	24
	Target Location	24
	Vertices of the obstacle	24
	Euclidean Distance between REP and RLP	25
	Distance between REP and RLP along permineter of obstacle	25

TABLE OF CONTENTS (Continued)

Chapter	Page
Approach 2 Drto calculation	25
Determining REV and RLV values	28
Special Cases in Obstacle modeling.	28
V. SIMULATIONS AND RESULTSTS	32
Robot Modeling	32
Target Modeling.	33
Obstacle Modeling.	42
Comparison of Approach 1 and Approach2 for Drto Calculation.	44
VI. CONCLUSIONS AND FUTURE WORK	48
Conclusions from the research	48
Future work.	49
REFERENCES	50

LIST OF FIGURES

Figure	Page
1. Preferences vector matrix between robots(A-J) to targets(0-9) over time	4
2. Difference vector to Target 1	5
3. Generalized Geometric Triangulation Algorithm.....	6
4. Garcia robot facing right	10
5. Robots and targets in a map	14
6. Map showing different objects	16
7. Different Kinds of obstacles.....	17
8. Distance calculation in the presence of obstacle in between robot & target.....	18
9. Determining REP and RLP points	19
10. Drto calculation	22
11. Dperi formulation	23
12. Approach2 to calculate Drto from robot to target	26
13. Approach2 to calculate Drto from robot to target (contd..)	27
14. Special case 1: the robot was inside reconstructed area.....	29
15. Special case 1 solution	29
16. Special case 2: the target was inside the reconstructed area	30
17. Special case 2 solution	30
18. Drto calculation for special cases	31
19. Code representing target modeling.....	32
20. Code representing robot in MATLAB	33
21. Entering Robot parameters	35

LIST OF FIGURES (Continued)

Figure	Page
22. Map generated by MATLAB showing robots and targets	36
23. Distance vector matrix in robot obtained from MATLAB	37
24. Initial preference vectors generated by MATLAB	38
25. Initial Iterations for Peter Molnar’s coupled equation.	39
26. Converged preference vector matrix.	41
27. Map generated by MATLAB showing target assignment	42
28. Obstacle creation using MATLAB	42
29. Map generated by MATLAB showing robots, obstacle and targets	42
30. Distance vector matrix generated by MATLAB with obstacle in place	43
31. Initial Preference values generated in presence of obstacle	43
32. Final preference values in case of obstacle presence	44
33. Target assignment in case of obstacle using approach2	44
34. robot and targets in place for comparing approach1 and approach2	45
35. Distance comparison for approach 1 and approach2 for different cases	46
36. Computational time taken for approach1 and approach2 for different cases	46

CHAPTER - 1

INTRODUCTION

Robots were the electro-mechanical devices which were designed to do a specific or many tasks. Though there was no particular definition by which we can distinguish a robot, but there were some characteristics which makes a device a robot. The characteristics which the present day robots have were physical movement, ability to make decisions, programmable, sensing, learning, communicating with other devices or robots etc... Present day robots can accommodate large number of features and each feature adds a new dimension to the complexity of arranging these features.

The usage of robots had increased very much in the last two decades. In industry the robots were used for packaging, handling and automation. Mars rover, moon exploration were the space oriented applications of the robots. Autonomous robots were used for exploring, mapping and collaboratively fulfilling a task. Defense based applications like robot patrolling, surveying an area were developing.

A part from traveling, these robots were efficient in taking decision and coming up with what to do. These robots should facilitate humans in starting a new life in different planets. They should figure their own targets and they should formulate their own path to reach their destination by avoiding obstacles, humans and other robots.

Once you have a group of robots to do an n number of tasks, then the robots should debate among themselves in handling tasks. In stead of all robots handling the same target at a time, one robot can be assigned to one target and perform the required task. This assignment was done by taking some parameters in to account.

In Peter Molnar's approach to the task assignment, he calculated a value for each robot which shows how likely that robot was interested in reaching a particular target. These values were called preference values. Preference value (IJ) explains how likely a robot 'I' was inclined towards serving target 'J'. The preference values to all targets from a specific robot were collected to form a preference vector. These preference vectors changes from time to time based on the preference vectors of other robots and finally converge to a single solution which helps robots to know their respective targets to serve. Robot plans the motion and starts moving in the direction of linear combination of preference vectors and direction vector before the final preference values are calculated. This part is called as motion planning in Peter Molnar's approach.

The preference value was calculated on the basis of cost from a robot to target. The cost the peter used in his algorithm was the Euclidean distance from the robot to target. Peter Molnar's algorithm assumes that there were no obstacles in the environment and the number of robots and targets were fixed. This helps in assigning one robot to one target in an obstacle free environment.

In this research, we came up with an idea of calculating preference values and preference vectors in the presence of an obstacle in an environment. All the robots have assumed to have full connectivity and equipped with proximity sensors, network cards. The robots can move freely from one place to another place. The targets were assumed to be immobile and their locations were initially known to all the robots. The obstacles were assumed to be regular sided with definite vertices. In most cases, the object was reconstructed as rectangle shaped. In special cases, the object was decomposed and reconstructed to non-complex object with definite vertices. The cost parameter that was

used in peter molanar approach was still used here i.e., distance but its obstacle adaptive distance. If there was an obstacle in an environment, the robot initially calculates the preference to the robot by taking perimeter of the obstacle in to account. An algorithm was developed to calculate the distance from robot to the target along the perimeter of the obstacle and the simulation results show the way the preference values change with the presence of obstacle in the same environment. Theoretically, this algorithm calculates the shortest path from robot to the target along the perimeter of the obstacle and helps the robot to reach the target via the shortest path.

Once the robots calculate their respective initial preference values and initial preference vectors based on the cost, they will calculate the subsequent preference values and vectors interdependently. The final preference vector for each robot helps it to know which target it should serve.

This chapter of introduction was followed by chapter 2 on Literature survey, chapter 3 on Practical work and theoretical work done on robots and the tools, equipment and software used for it, Chapter 4 deals with my approach to calculate the preference vectors in the presence of obstacle in an environment, Chapter 5 had simulations and results, and finally Chapter6 had conclusion and future work.

CHAPTER 2

LITERATURES SURVEYED

Communication fault tolerances in distributed robotic systems [1]:

Peter Molnar's algorithm deals with assignment of group of robots to bunch of tasks. The algorithm was designed for task assignment in an obstacle-free environment with equal number of robots and targets. Tasks were given least importance in this paper and mainly dealt with task assignment.

The motto of task assignment was achieved by using dynamic systems approach of coupled selection equations. These coupled selection equations converge to a single solution in deciding which robot to sever which target.

Robot's travel distance and overhead cost were taken as parameters while assigning the robots to the targets. Minimization of these parameters was taken in to account.

Peter Molnar's combines the aspect of motion planning and target assignment which allows the robot to move before the destination was finalized. The robots movement was determined by the preference vector matrix. The preference vector matrix is collection of preference vectors from all robots. Preference vectors were collection of all preference values. The initial preference values are calculated by using the equation [2] which is present later in the document. The preference values from one robot were combined to form a preference vector of that robot. The preference vectors help the robot to determine which direction it should move. The robot moves in the direction linear combination of difference vectors and preference.

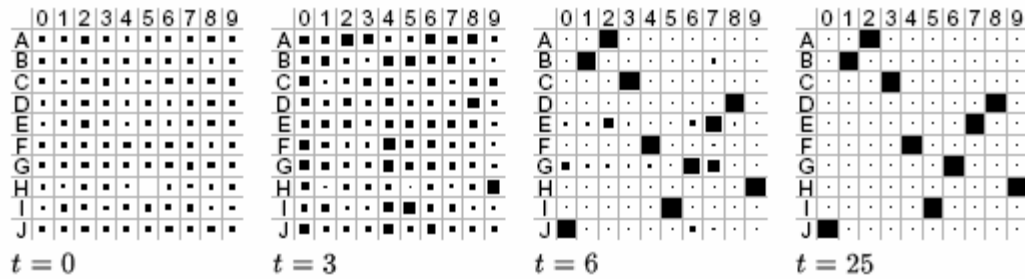


Figure 1: Preference vector matrix consisting of Preference values of all the robots (A-J) to target(0-9) over certain intervals of time[1].

The robot initially calculates the preference values based on the proximity of the robots and targets. Preference vectors are obtained by these calculations. Then the robot starts motion planning using the destination vector (e^0) which was the linear combination of difference vectors and preference values.

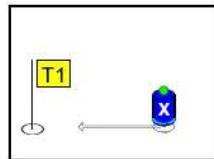


Figure2: Difference vector from robot to target T1[1].

The preference vectors were calculated and transmitted again to all the robots through broadcast. All other robots update their preference vector matrix and ag motion planning was done. These preference vectors converge over a period of time so that the robots can point out their final destination.

In Peter Molnar’s algorithm it was assumed that there were no obstacles for calculating initial preference values and motion planning. The obstacles and robots (which were also considered as obstacles) were avoided by using proximity sensors.

This research was an extension of Peter Molnar’s algorithm by modeling an obstacle and taking its presence in calculating preference values and also in motion planning. The initial preference calculation was given more weight than motion planning.

Generalized Geometric Triangulation Algorithm for Mobile Robot Absolute Self Localization [2]

Triangulation algorithm helps in localizing any obstacle in a given plane. The robots move from place to place in a given plane. If the initial location and alignment of the robot were provided to the robot itself, it can localize itself by backtracking. But its painstaking process and uses robots processor intensively. Instead of that, Triangulation can be used to localize the robot in an environment with at least three beacons in the environment.

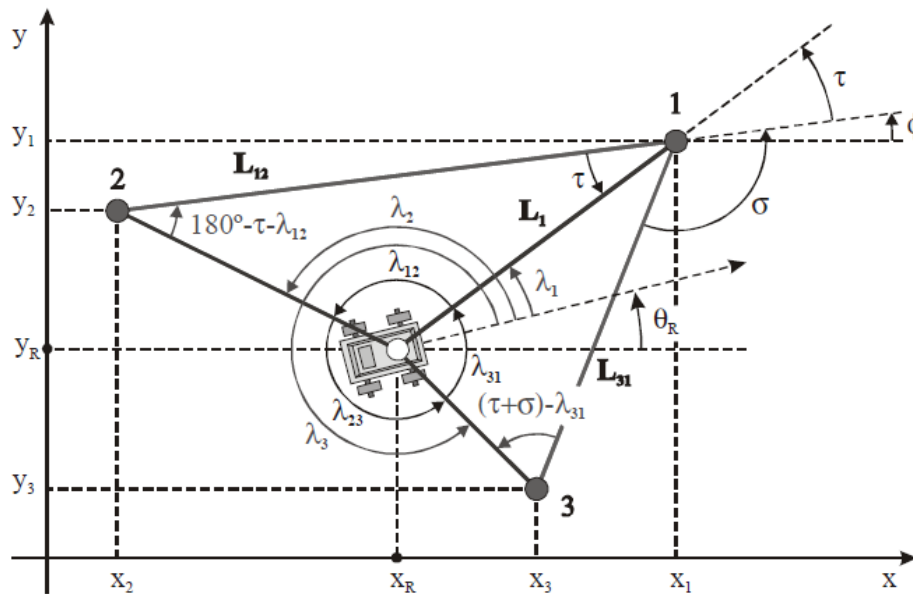


Figure 3: Generalized geometric triangulation algorithm [2]

In triangulation algorithms, there were some of the restrictions like (a) three beacons should be placed in consecutive order (1, 2, and 3), (b) The robot must be seen by all beacons, (c) the robot and all the beacons should not be on the same circumference of a circle. (d) the robot should be placed within the triangle formed by the three beacons to get localized (d) The angle between beacons 1 and 2 (λ_{12}), and the angle between beacons 1 and 3 (λ_{31}) should be less than 180 degrees, otherwise labels were changed until requirements were met.

The restrictions in Triangulation algorithms were solved in this literature [2] by making some changes and making it generalized geometric triangulation algorithm. The obtained Generalized geometric triangulation algorithm helps us to do the following things: (a) beacon labeling can be done randomly (2, 1 and 3), (b) the beacons can be placed anywhere in the plane, (c) the robot can be placed in the given plane i.e., inside or outside the triangle formed by the three beacons, (d) the angle between 1 and 2 (λ_{12}) and the angle between beacons 3 and 1 (λ_{31}) may be more equal or greater than 180 degrees.

This approach of Generalized Geometric Triangulation algorithm can be used to localize the robot in our research. It helps robot to know about its location whenever it needs it.

Flexible Autonomous Robotic Task Scheduling using Advanced RISC Machines [3]:

This was master's thesis done by Zach under computer science department, Wichita state university in May 2008. It is an extension to Peter Molnar's algorithm addressing

some of the problems that were present in the Peter Molnar's algorithm [1]. In his research, assignment was done by any number of robots to any number of targets where as the Peter Molnar's algorithm deals with just equal number of robots to targets. The result of his thesis showed two quick methods for task scheduling and assignment. [3] was used to review his ideas and their implementation. The obstacles were not still considered in his research which helped us to come up with an idea of preference value calculation in the presence of an obstacle.

CHAPTER 3

HARDWARE AND SOFTWARES USED

The research work was done in two phases. The first phase was done using hardware mainly Garcia robots and the second phase was mainly dealt with software like Mat lab simulation of the scenario. The first phase actually gave hands on experience on Garcia robots which gave concentric idea how the real time robots were. In the second phase, all the work was done using Mat lab for formulating a solution to the research

Hardware Used:

While the thesis was in progress, practical work was done on real time robots. These robots were called Garcia robots. The work was done on many aspects. List of things that were done were shown below:

Garcia robots- out of the box:

When the Garcia robot was given, the first thing done was to know all about it. The only way that can be done was going through the specification sheet and Garcia manual. Garcia robots were manufactured by Acroname Company. It was highly sophisticated robot. It was equipped with Stargate board which was used for programming the robot. There were different sensors that were present on the Garcia robots like IR range finders, ledge detectors, and laser sensors. Garcia robots have two processors. Brainstem GP 2.0 was the processor responsible for Additional IO, IR communication capability and serial

interface. Moto 1.0 was responsible for mobility of the robot and sensor inputs. Garcia robots were equipped with Ambicom wireless network cards.

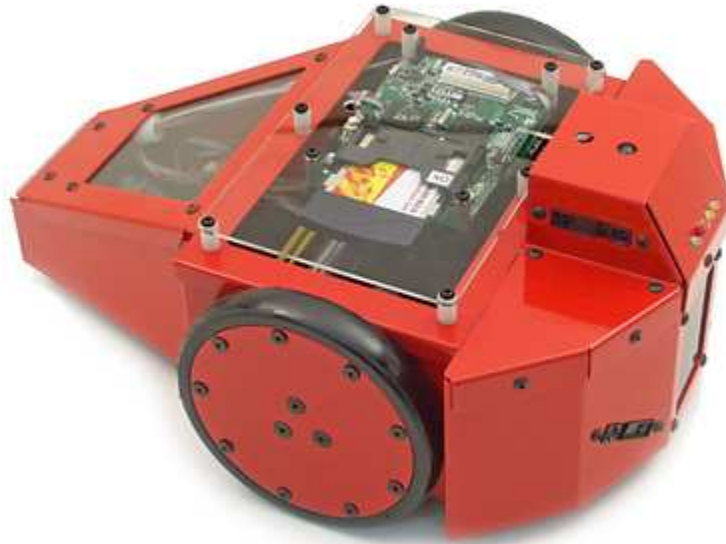


Figure4: Garcia Robot facing right

Sensors and their uses:

IR range detectors were used to detect any kind of obstacles in the path of the robots. They have a range of 4 to 18 inches. These IR range detectors can detect any obstacles within the mentioned range. Ledge detectors were used for detecting the edges or sudden pitfalls in the robot's path. Laser sensors were used for scanning a given area. These sensors were mainly used for mapping a given area.

Stargate Board:

It was present inside Garcia robots. It operates at a frequency of 400 MHz X scale. The programming board had embedded Linux installed on it. It had 64 Mb of memory for storage, a wireless network card to communicate with other Garcia robots equipped with Stargate board. GNU ARM-Linux cross compiler was used to compile the programs on computers to execute them on the Star gate board.

Programming on Garcia robots:

The first program executed on the robot was the obstacle avoidance program whose executable was preloaded on the robot. This demo program was initiated by just turning on the robot while holding program switch on the robot. As the program was executed, the robot starts moving along the floor. When there were any obstacles on its way, the robot stops and turns to certain angle away from the obstacle and again starts moving and if again if it encounters the obstacle the process was repeated.

The program that was written, compiled and executed on the robot was the server-client program. It was done with the help of Zach, computer science student. The code for basic server and client was written in C- Language. The server and client code was compiled separately on the Ubuntu Linux laptop which had ARM- Linux cross compiler. The executables were created on the host laptop. These executables were placed on the robots using ftp applications like win scp. The server program was copied on to one robot and client program was copied on to the other robot using winscp or ftp. Server program and client program were executed on two different robots by logging in to each of them by

using secure shell(Computy). We can see the broad cast messages sent between the robots on the host laptop on ssh sessions.

Photographs can be taken using the Garcia robots. Installing video drivers on Stargate board helps to take pictures using Garcia robot. The whole process was done under the guidance of my colleague Harini Balasubrahmanian, at Wichita State University.

SOFTWARES USED:

For the work done on the Garcia robots, different soft wares were used. Linux computer with ARM-Linux X compiler was used to compile files on the host computer. Winscp was used for transferring executables from host computer to the Garcia robot. Computy was used to logging in to the robots using SSH and execute the programs on the robot. C, C++ were used to write programs that were to be executed on the robot.

Matlab was used extensively for this thesis. All the simulations, problem formulation and solving were done on the matlab.

CHAPTER 4

APPROACH TO CALCULATE PREFERENCE VALUE IN THE PRESENCE OF OBSTACLE IN THE ENVIRONMENT

The initial setup had equal number of robots and targets in the given environment with out any obstacles. All the robots were identical, independent of each other and equipped with sensors like IR sensors, ledge detectors. They were equipped with network cards which help them to communicate with other robots and also establish TCP or UDP based communication between them. IP addresses are also assigned to each of the robots which facilitate communication. Robots move from place to place in the map and perform their work based on their own decision or decision taken collaboratively with other robots. All these computations decision making was done by the processor present in them, where as data was stored in their memory. All the robots were assumed to have full connectivity. There was no loss of any information broadcasted by the robots.

Targets used in this environment were immobile. Their location was known to all the robots initially. All the robots and targets in the environment were independent of one another initially and no where attached physically. All the robots share a common map. This map was updated from time to time with the help of broadcast messages sent between robots. Robots were attached with location sensors. These sensors help the robot to know its location time to time. To localize, these location sensors need beacons in the environment.

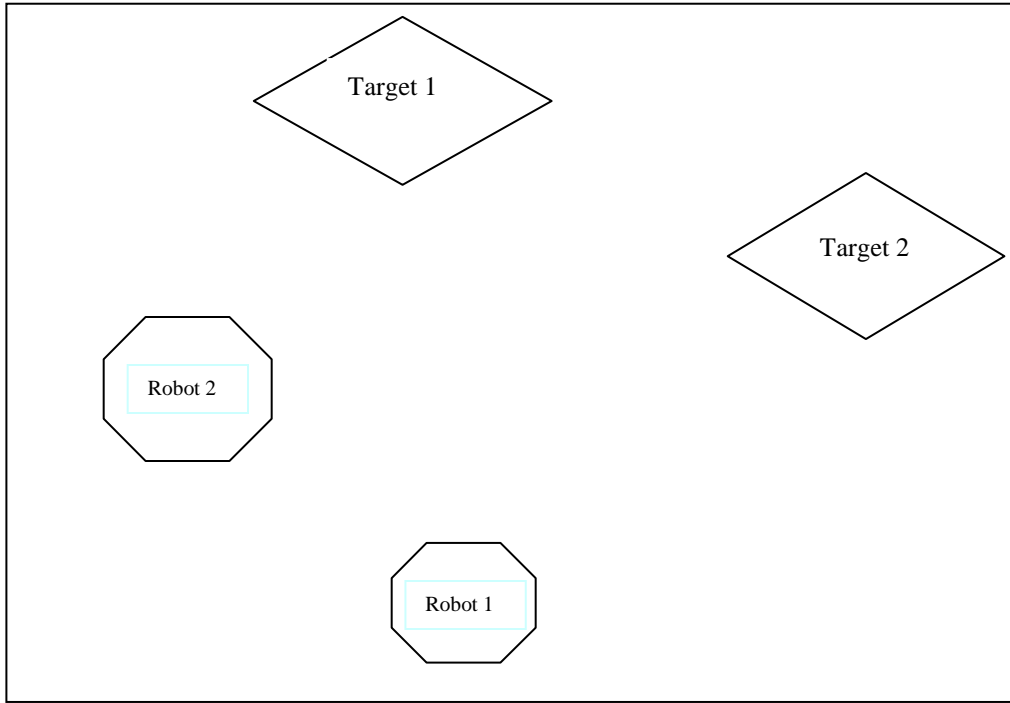


Figure 5: Robots and targets in a map

Once the robots, targets were in place, the robots start calculating their distances to the targets. The distances mentioned were the Euclidean distance from robot to target. The collection of all distances from robot1 to all targets was the distance vector of robot1.

Each robot broadcasts its distance vector to all the other robots. All other robots receiving these distances, update them in their distance vector matrix. Each target would calculate its initial preference values to all other targets with the help of the distance vectors it obtained.

Initial preference vector calculation:

The preference value shows how likely a robot was to reach a specific target. The initial preference values were created by the equation given by the Peter Molnar.

$$\xi_{ij}(0) = 1 - \frac{\|\mathbf{r}_i(0) - \mathbf{g}_j\|}{\max_{i',j'} (\|\mathbf{r}_{i'}(0) - \mathbf{g}_{j'}\|)}.$$

Equation 1:Initial preference equation of Molnar[1]. \mathbf{r}_i represents robot and \mathbf{g}_j represents target

These preference values were ranged from 0 to 1. Set of all preference values from one robot to all different targets was called preference vector. These preference vectors were helpful for determining how inclined a robot to the targets it knows. Initially each robot knows its own preference vector to all other targets. Upon broad casting and receiving broadcasts it creates a preference vector matrix where it maintains all the preference values from all robots to all other targets. The subsequent preference values were created based on the values that were already present in the preference vector matrix. All the robots would calculate the preference values to all other targets interdependently by the equation 2.

$$\dot{\xi}_{ij} = \kappa \xi_{ij} \left(1 - \xi_{ij}^2 - \beta \sum_{i' \neq i} \xi_{i'j}^2 - \beta \sum_{j' \neq j} \xi_{ij'}^2 \right)$$

Equation 2: Peter Molnar's coupled equation.

Finally the preference matrix converges in such a way that each row and each column had only one value with 1 and other values as 0 which shows the robot assignment to target. It was how task assignment was done using peter Molnar's equations.

. The distance parameter used by the Peter Molnar in the equation 1 for the calculation preference value was Euclidean distance from robot to target. Distance parameter was reformulated in this research and showed how it would affect the task assignment. This research mainly deals with the calculation of initial preference values when there was an obstacle in the environment.

Obstacle modeling:

Obstacle was considered as a physical entity that causes obstruction to the robotic movement. Obstacle can be of any shape but not complex and can be located anywhere in a given space (grid in which robots were moving). Location and size of the obstacle was very important while calculating the preference values of a particular robot to a particular target.

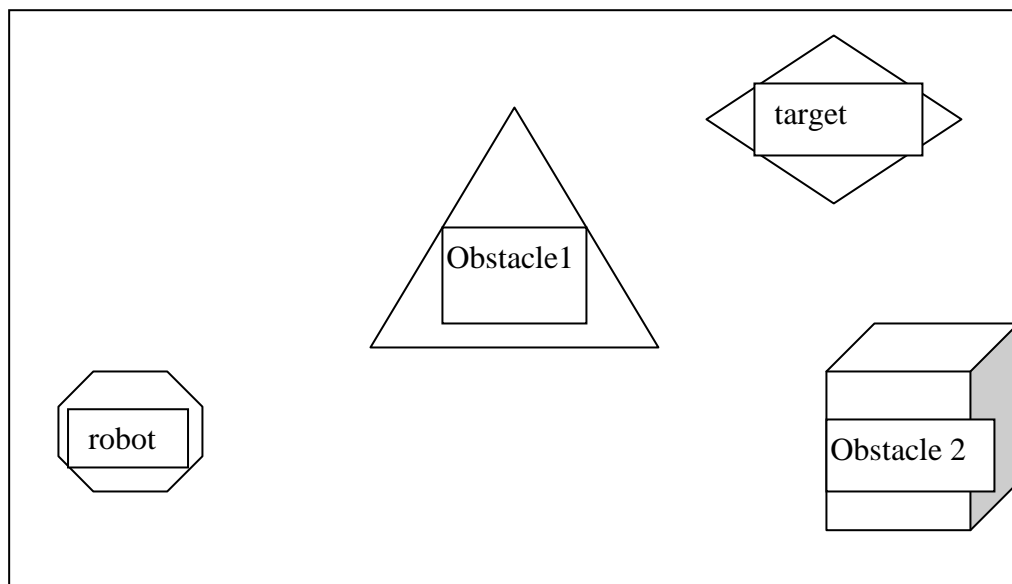


Figure 6: Map showing robot, obstacles and Target

The location of the obstacles was very important. They were going to play a vital role in the calculation of preference values. As shown in figure 6, obstacle 1 was located at the centre

of the grid where as, obstacle 2 was located at corner of the grid. Obstacle2 does not affect the preference values as much as obstacle1. Even some times the affect of Obstacle2 on preference values was negligible.

Only one obstacle was considered in the environment. Obstacle was regular shaped with definite vertices and sides. The vertices of the obstacle were known to the robot initially. In case of irregular obstacle or obstacle with no vertices, imaginary non -complex obstacle was created in robot's memory that had sides and vertices.

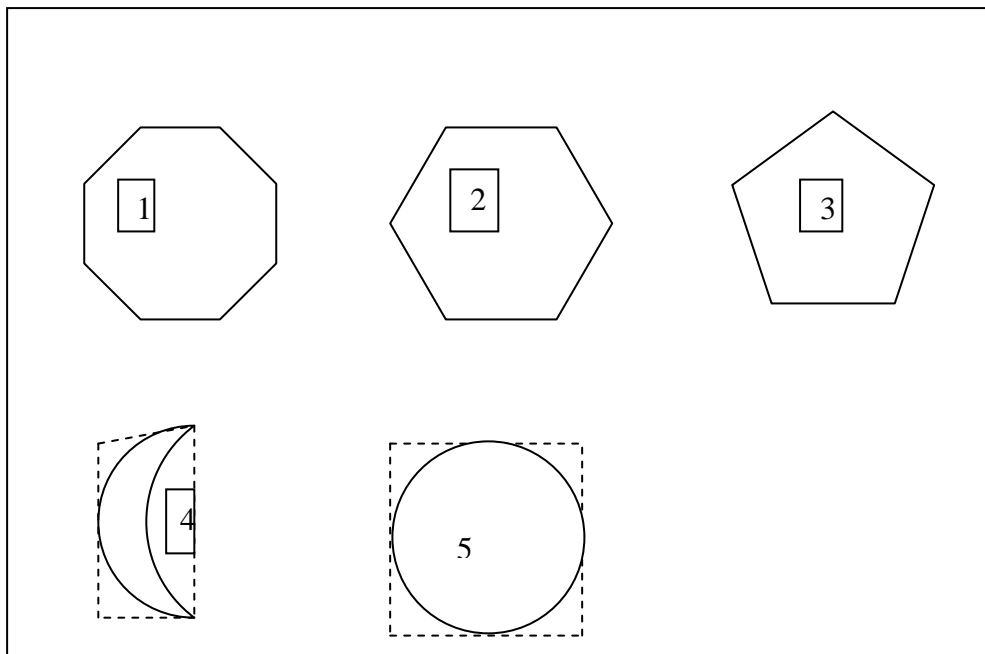


Figure 7: Different kinds of Obstacles

In the above figure 7, the obstacles 1, 2, 3 shows regular sided obstacles with definite vertices where as obstacles 4 and 5 were irregular for which imaginary obstacle with regular sides and definite vertices were created in robot's memory.

Obstacles should have definite sides and the number of sides should be greater than two. For the remaining discussion obstacle was assumed to be a rectangle shaped for convenience.

Approach to distance calculation in presence of obstacle:

In Peter Molnar's approach, where there were n robots and n targets, there was a concept called initial preference vector calculation for assigning each robot to one target. When there were obstacles (considering only one obstacle) in between robots and targets, the initial preference value equation was modified to accommodate changes in the environment because of presence of obstacle and achieve the goal.

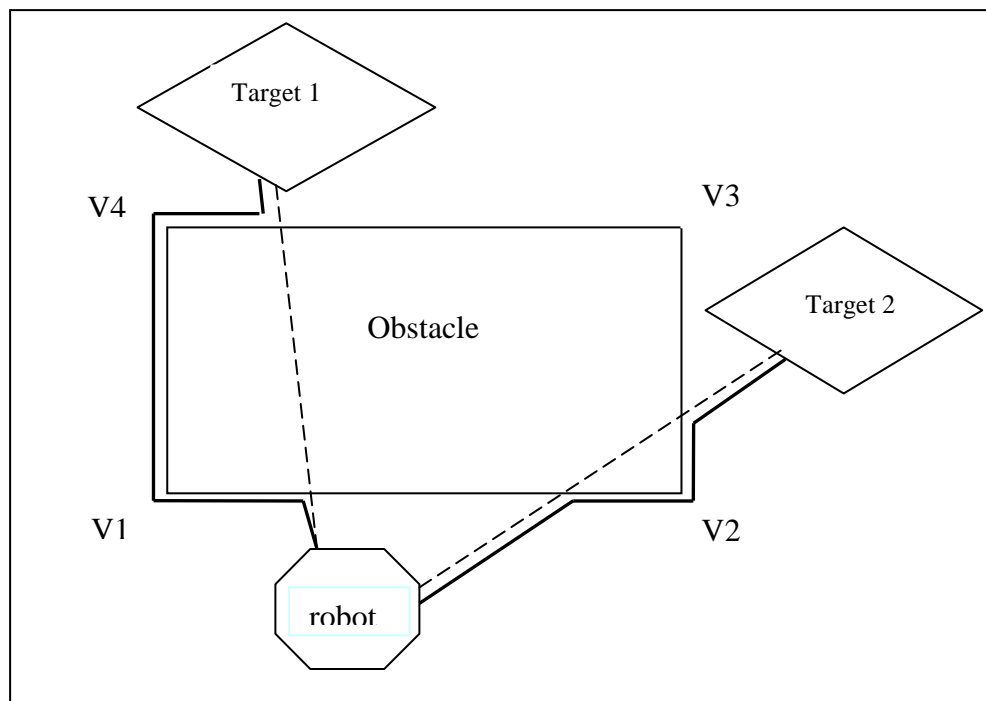


Figure 8: Distance calculation when there was an obstacle in between the robot and target

In the above figure.8, V1, V2, V3, V4 denotes the vertices of the obstacle. The dotted line represents the Euclidean distance between robot and target. Thick line represents robots distance

to the target when obstacle was in between them ($Drto$: Distance from robot to target to distance when obstacle is in between). The parameter $Drto$ was taken in to consideration while calculating preference values. In Peter Molnar's algorithm the 2nd equation can be modified as

$$\xi_{ij} = 1 - \frac{Drto_{ij}}{\max_{i',j'} (Drto_{ij})}$$

Equation3 :where E_{ij} was preference value from Robot 'i' to Target 'j'.

Approach 1 $Drto$ Calculation (Distance from robot to target when obstacle in its path):

When there was an obstacle between robot and target in an environment, the Euclidean distance between robot to target was replaced by $Drto$, which was the distance from robot to target by taking perimeter of the obstacle in to the account. All the environment entities were setup like robot, target and obstacle. The location of targets and obstacles were known to the robot. Map was created in the robot using all the entities. Obstacle modeling was done in the map i.e., what ever the shape of obstacle, it was reconstructed in the map of the robot in such a way that obstacle was regular sided with definite vertices. Then robot starts calculating the distances to each target. Before calculating, it checks whether there was an obstacle in between the robot and target. In case if there was no obstacle, normal Euclidean distance was calculated from robot to target. In case if there was an obstacle, it calculates distance taking perimeter of the obstacle in to account. But how would robot know that there was an r in between target and it self.

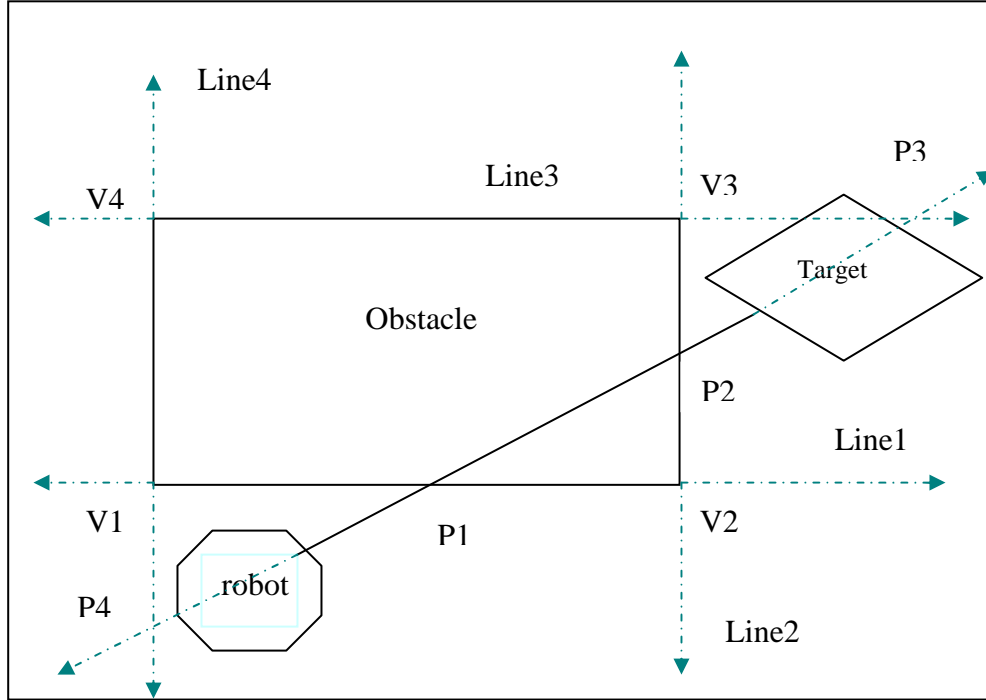


Figure 9: Determining REP and RLP points

The vertices of the obstacle were stored in a perfect order in robot's memory. Robot knows that V1 was first vertex of obstacle and it was followed by V2, V3 and V4 respectively. It considers the next vertex to V4 was again V1 making obstacle a closed figure. Robot co-ordinates were taken as $R(x_r, y_r)$ and targets vertex as $T(x_{tar}, y_{tar})$. The co-ordinates of vertices of obstacle was taken as $V1(x_1, y_1)$, $V2(x_2, y_2)$, $V3(x_3, y_3)$, $V4(x_4, y_4)$. An imaginary line was drawn in robot's map between Robot and Target and was denoted as lineRT. Imaginary lines were drawn on robots map in such a way that line 1 passes from V1 to V2, line 2 passes along V2 and V3, line 3 passes along V3 and V4, and finally line 4 passes from V4 through V1. LineRT intersects line1, line2, line3 and line 4 at four points P1 (x_{p1}, y_{p1}), P2 (x_{p2}, y_{p2}), P3(x_{p3}, y_{p3}), and P4(x_{p4}, y_{p4}) respectively. The points were checked whether they were present on the obstacle. If there were two or more

points that lie on the obstacle, then it was understood by robot that there was an obstacle in its path towards target. So, it starts calculating Drto.

In the figure 9, the points P1 and P2 were on the perimeter of the obstacle, so these two points were considered for deciding REP (Robot Entry Point) and RLP (Robot Leaving Point) for that robot to target. The robot calculates its distance to P1 and P2. In the figure, distance from robot to P1 was less than that of robot to P2. It decides P1 as REP for that robot to target. Then it calculates points P1, P2 distances from target. The point which was near to target was taken as Robot leaving point (RLP) for that Robot to Target. In the figure P2 was taken as RLP.

In brief, the intersection points were obtained by solving equation of lineRT with that of line1, line 2, line3 and line4. If there were number of intersection points on the perimeter of the obstacle was greater than two, then the robot understands that there was an obstacle in its path and calculates Drto. The intersection point on the perimeter of the obstacle which was near to the robot was called Robot Entry point (REP_{ij}) from robot 'i' to target 'j' and the one which was near to the target was called Robot Leaving point (RLP_{ij}). REP_{ij} and RLP_{ij} were very important in determining the distance from robot to target when there was an obstacle in between. They change from another robot to another target.

Once the robot knows that there was an obstacle in its path to target, it uses Drto calculation.

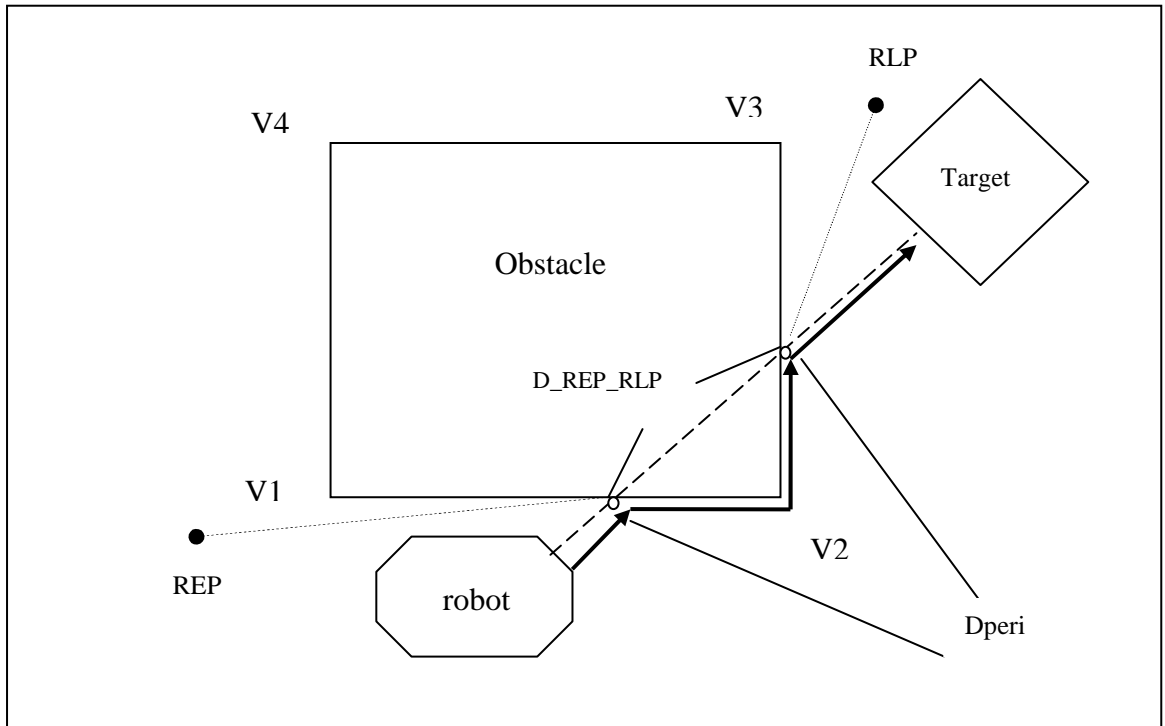


Figure 10: Drto calculation

The figure 10 explains how to calculate Drto from robot to target. The dotted line represents the imaginary line drawn from robot location to the target location whose length was the Euclidean distance from robot to target. The thick arrowed line was the Drto, which was calculated and taken in to account for preference value calculation. This Drto was selected based on the Dperi1 and Dperi2. Dperi1 is the distance between the REP and RLP points along one path. Dperi2 is the distance between REP and RLP through alternate path. The Dperi1 and Dperi2 were compared and the least was taken in to Dperi(Distance from RLP to REP along the perimeter).

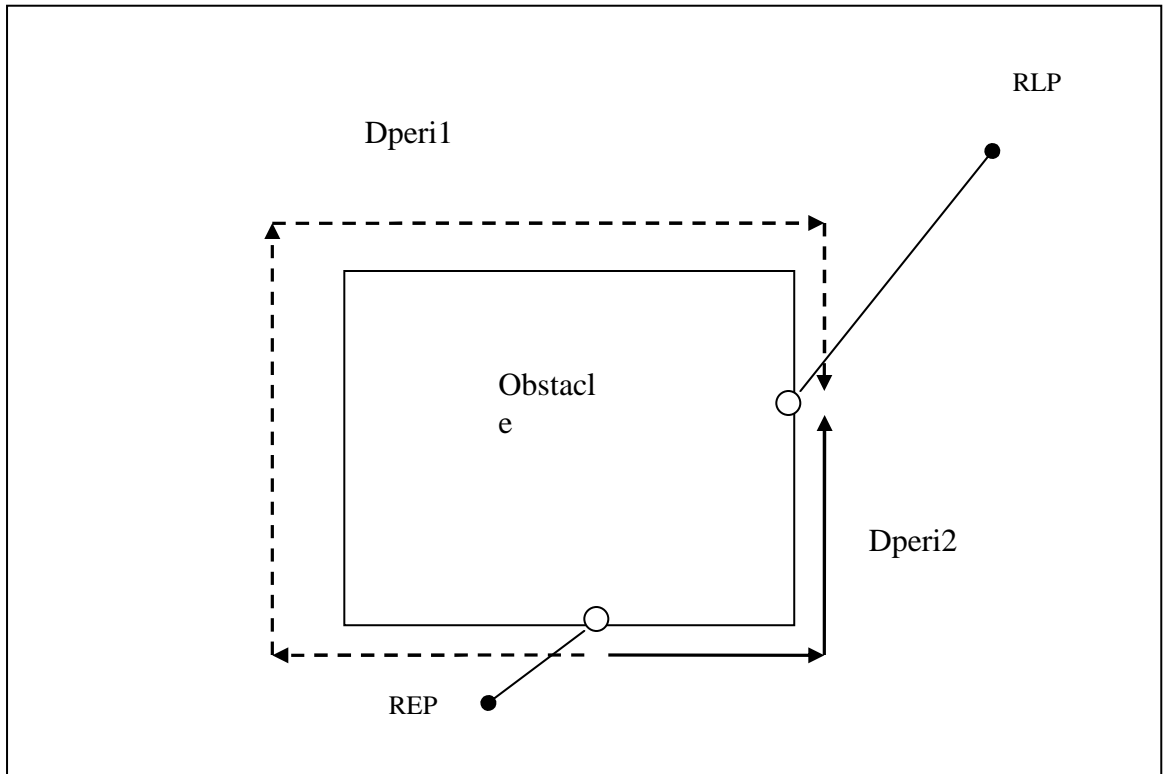


Figure 11: Dperi formulation

D_{periij} was the distance from REP_{ij} to RLP_{ij} along the perimeter of the obstacle. D_{peri1} and D_{peri2} were two values for D_{peri} . The minimum of the two was considered as D_{peri} value.

$$D_{periij} = \min(D_{peri1ij}, D_{peri2ij})$$

Equation 4: Determining the D_{periij}

Some important terms while calculating D_{rto} :

RT line: Robot Target Line

It was an imaginary line taken for convenience. This RT line was a straight line drawn from the Robot location to the target location irrespective of obstacle (dotted line between robot and target in fig)

RL: Robot Location (X_r, Y_r).

It is the location of the robot in a map. The robot location was known to the robot by using GPS sensors.

REP_{ij}: Robot Entry Point on obstacle ($X_{rep_{ij}}, Y_{rep_{ij}}$)

This was an imaginary co-ordinate calculated by the robot 'i' when it knows that there was an obstacle in between the robot 'i' and target 'j'. The REP_{ij} co-ordinate was close to the robot 'i'. It was the co-ordinate on obstacle where RT line intersects the side of the obstacle, which was near to the robot. The REP point changes from one target to another target for a robot.

RLP_{ij}: Robot Leaving Co-ordinate on obstacle ($X_{rlp_{ij}}, Y3_{rlp_{ij}}$)

It was also an imaginary co-ordinate calculated by robot. It was near to the target. It was the co-ordinate on obstacle where RT line intersects the side of the obstacle, which was near to the target.

TL: Target Location (X_{tar}, Y_{tar})

The targets were assumed to be immobile. So the target locations were initially known to all robots.

V1, V2, V3, V4: Vertices of the Obstacle:

These vertices of the obstacle were stored in exact order on all robots.

Dist_REP_RLP: Euclidean distance between the REP and RLP

Dotted line between REP point and RLP point in fig. 4 represents REP and RLP.

Dist_REP_RLP was formulated as Euclidean distance between REP and RLP, which were calculate for robot 'i' to target 'j'.

$$Dist_REP_RLP_{ij} = \|REP_{ij} - RLP_{ij}\|$$

Equation 5: Distance between REP_{ij} and RLP_{ij}

Dperi: Distance between REP and RLP along the perimeter of the obstacle.

Thick arrowed line between REP point and RLP point represents Dperi. (as shown in fig.4). Dp can have two values but the lower value was chosen.

$$Drto_{ij} = \|RL_i - TL_j\| + Dperi_{ij} - Dist_REP_RLP_{ij}$$

Equation6: Drto calculation for robot 'i' to 'target 'j'

Dperi_{ij} – Dist_REP_RLP_{ij} ranges from 0 to ½*(perimeter of the obstacle). It was 0 when there was no obstacle in between the robot and target

In this way Drto value is calculated and this Drto value is used in the calculation of preference values.

Approach 2- Drto calculation:

Instead of calculating the cost along the perimeter of the obstacle, the alternative way can be calculating the cost using the perimeter partially. In the figure, the obstacle was in between the robot and target. The dotted line represents the Euclidean distance from

robot to the target where as the thick arrow line represents the cost taking object in to consideration. The whole process is same as approach 1 until the REP and RLP points were calculated. Once REP and RLP points were determined, and then Robot Entering Vertex (REV) and Robot Leaving Vertex (RLV) were determined in approach 2. In the below figure 12, the vertex V2 is considered as both REV and RLV. So the sum of distances from Robot to REV and Target to RLV were taken as the drto.

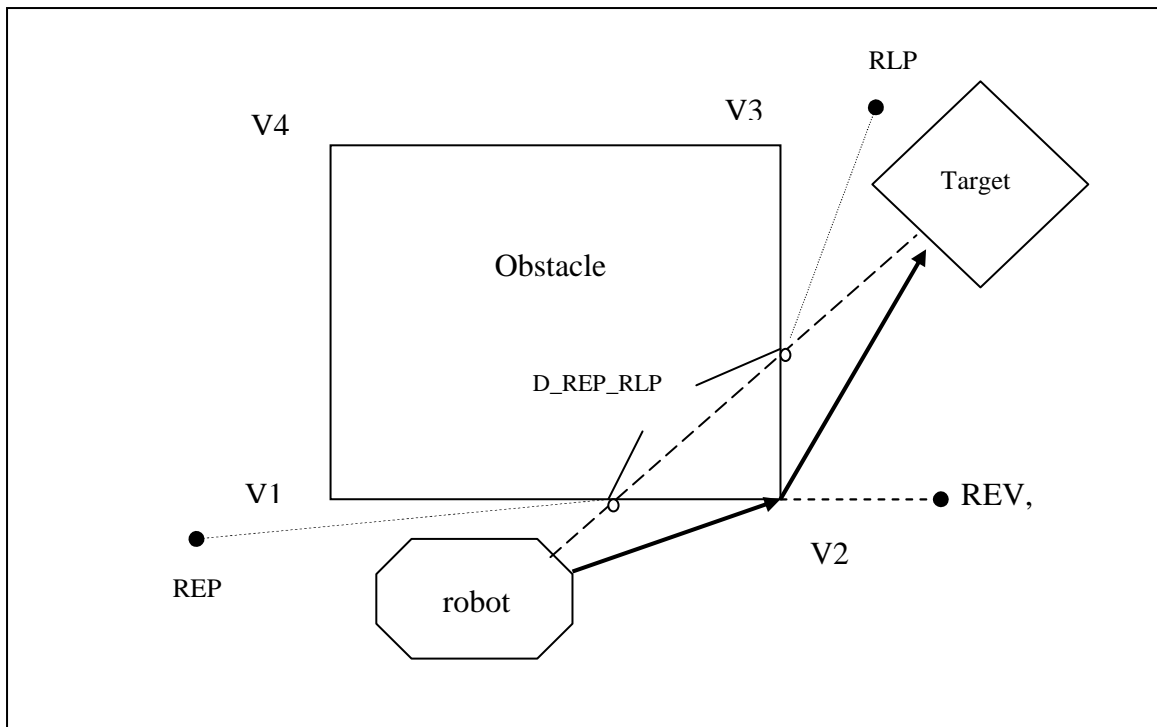


Figure 12: Approach 2 to calculate Drto from robot to target.

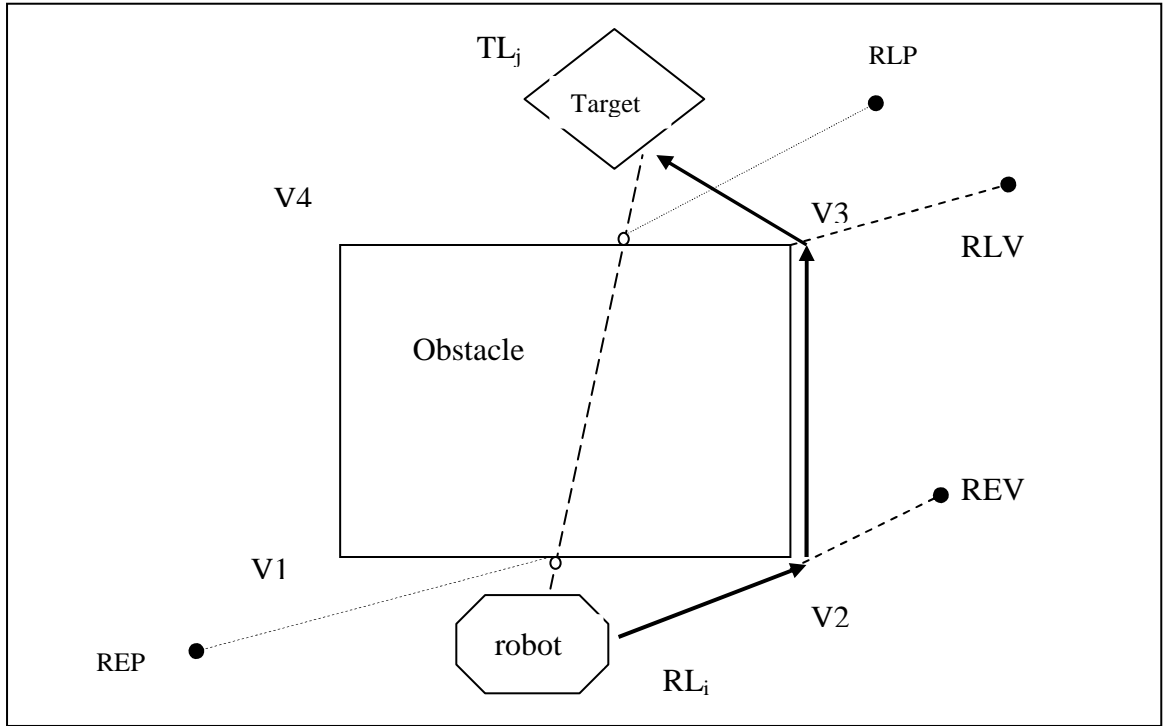


Figure 13: Approach 2 to calculate Drto from robot to target.

The Drto calculation for the Approach 2 can be formulated as:

$$Drto_{ij} = \|RL_i - REV_{ij}\| + Dperi_{ij} + \|TL_i - RLV_{ij}\|$$

Equation 7: Drto calculation using approach 2

The Drto calculated in the second approach was the shortest. The $Dperi_{ij}$ is the distance traversed through the perimeter of the obstacle by the robot and its value ranges from 0 to $\frac{1}{2}$ of the perimeter of the obstacle. In the figure 13, $Dperi_{ij}$ was distance between vertex 2 to vertex 3. But the point of concern was determining the REV and RLV values.

Determining REV and RLV values:

The Dperi meter which was used in approach 1 can be used as deciding factor. As we saw the Dperi can have two values and the one with lowest value is taken in to consideration. In figure, Dperi was calculated along the vertices REP, V2, V3 and RLP. In this order, the vertex closer to the REP was taken as REV and the vertex closer to the RLP was taken as RLV. Once RLV and REV were known the Drto calculation in eq was followed.

Special cases in Obstacle modeling:

The obstacle was assumed to be regular sided with definite vertices. If there were any irregular obstacles with no definite shape or obstacles which were curved i.e., convex obstacles, concave obstacles and circular obstacles, then obstacle reconstruction was done. The reconstructed obstacle was assumed to be rectangle in all cases. There exists a problem how the robot can handle if there was a target or robot itself in the reconstructed rectangular obstacle.

If the robot or target was inside the reconstructed area, then again reconstruction of obstacle was done using decomposition algorithms. The obtained reconstructed obstacle may not be regular obstacle or need not to be a rectangle but obstacle with definite vertices and sides. If the obstacle and target both were in the reconstructed area, obstacle decomposition was done too. Once the obstacle decomposition was done, then procedure for task assignment was followed.

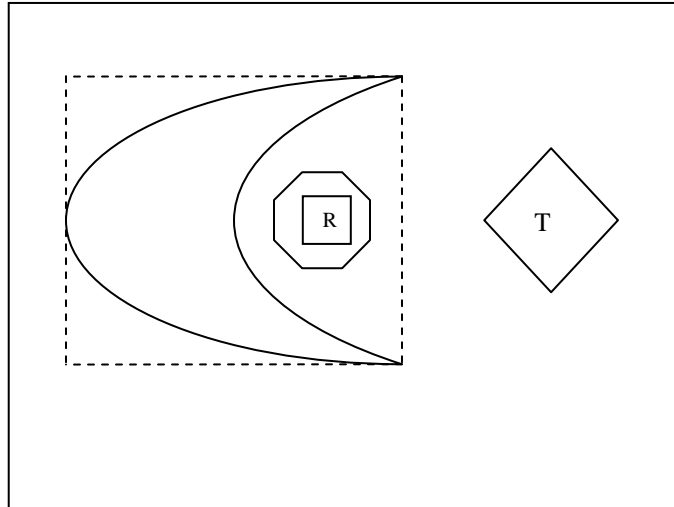


Figure 14: Special Case 1: The robot was inside the reconstructed rectangle.

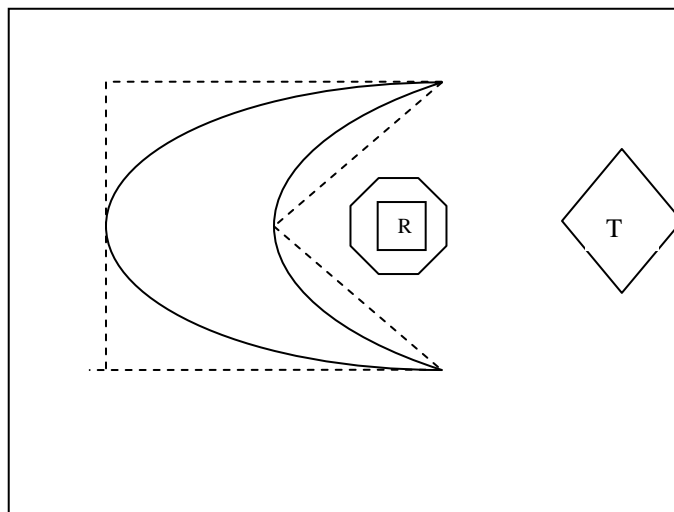


Figure 15: special case 1 solution

Figure 14 and 15 show what steps were followed if there is a robot in the reconstructed area of the obstacle. The irregular concave shaped obstacle had some more vertices on the inner curve which were not taken in to account while constructing rectangle. The obstacle was decomposed by taking one of the vertexes on inner curve into consideration.

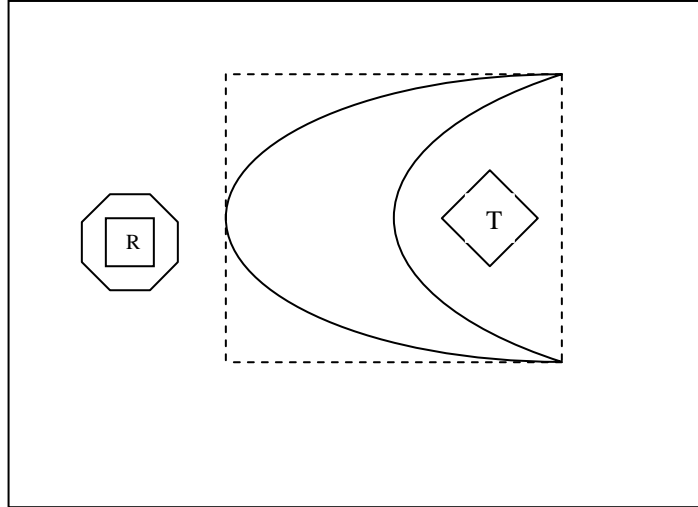


Figure 16: Special Case 2: Target was inside the reconstructed area.

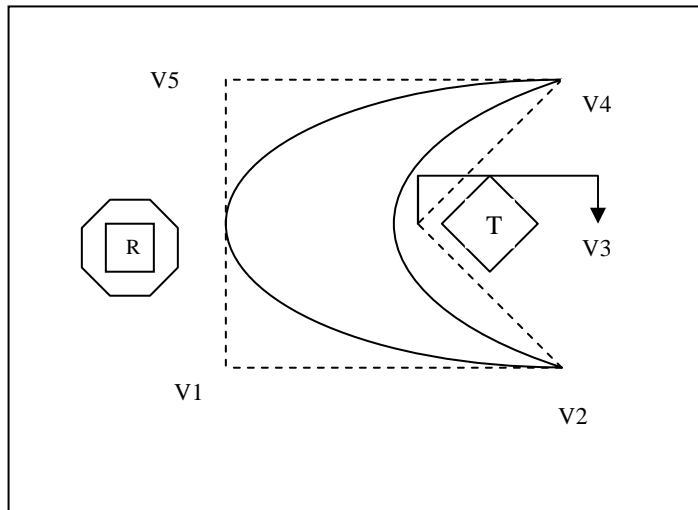


Figure17: Solution to the special case 2

In the above figure 17, the obstacle was decomposed when there was a target in the reconstructed area around the convex obstacle. The newly formed vertex was named as V3. The vertices V3, V4 which were present before the decomposition of the obstacle were changed to V4 and V5 respectively. Now V1, V2 ,V3, V4 and V5 should be in exact order while calculations.

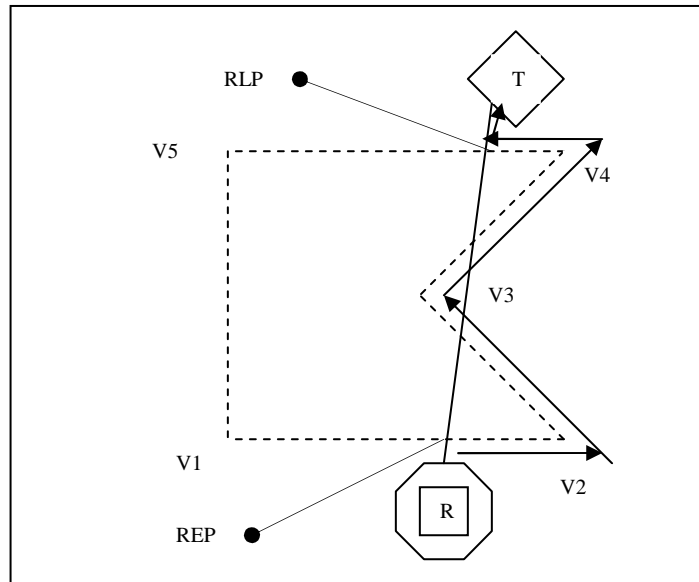


Figure18: Drto calculation along the perimeter of reconstructed obstacle obtained by the decomposition

An imaginary line RT was drawn from robot to target. Line1 was drawn along V1 and V2. Line 2 was drawn along V2 to V3. Similarly line3, line4 , line 5 was drawn ending lin6 joining v5 and v1 vertices. LineRT intersects line 1 to line6 at six different points. Of these six points, the ones that are on the reconstructed obstacle were taken in to consideration. In the above figure 18, we have four such points. Of these four points, the point which was closer to the robot was considered as REP and the point which was near to the target was considered as RLP. The remaining procedure was same as the calculation of Drto in the previous section. The arrowed line in the above diagram shows that the cost calculation from robot to target.

CHAPTER V

SIMULATIONS & RESULTS

The approach started with simulating the Peter Molnar's equation. After the Peter Molnar's equation was simulated, the obstacle was introduced in to the environment and came up with task allocation. MATLAB was used for the simulations. The environment containing robots and targets was simulated using MATLAB software. The robots were simulated in the form of structures with different fields, where as targets were shown as a structure with only one field which holds the co-ordinates of their own location. When the simulation started we were asked to provide with information about number of targets and robots in a given environment. The value was chosen as 3 which mean that there are 3 targets and 3 robots in the environment. The following explanation shows how the robots and targets were modeled in MATLAB

Target Modeling:

Targets were immobile. They don't perform any intelligent operations. The only parameter that was of interest about a given target was the location. Targets were represented in the following way in the MATLAB:

```
target(1).location = [8 12];  
target(2).location = [12 2];  
target(3).location = [1 7]
```

Figure 19: code representing target modeling in MATLAB

The above figure 19 show that targets 1, 2 and 3 were located at the locations (8,12), (12, 2), (1, 7) respectively. These locations were static and these values remain unchanged through out the program.

Robot Modeling:

Robots were intelligent devices. They can perform various tasks at a given time and also maintain databases. Each robot can have their own name an IP address as an identity to distinguish from other robots. They were designed in the MATLAB as a structure having different fields. The following code shows howthe robots were programmed

```
for i=1:n
    robot(i).location =0;
    robot(i).distance =[0 0 0 0 0];
    robot(i).preferencematrix =0;
    robot(i).preferencevector =0;
    robot(i).ipaddress=0;
    robot(i).name =0;
end
```

Figure 20: code representing Robot initialization in MATLAB

The above figure 20 shows how the robot parameters were initialized. The various fields that were in the robot were location, distance, IP address, name, and preference vector and preference vector matrix.

Location gives the present location of the robot. The starting location of the robot can be given manually. Alternative way can be through the location sensor the robot will have an idea where it was in each point of time. Robot(1).location = [12,3] represents that robot 1's present location was (12,3).

Distance field in the robot was assigned to maintain the distance vector. Distance vector of a robot represents the robot's distance to various targets in the environment. Distance vector matrix was also created in a robot, when it receives broadcast messages from all other robots. These distances were very important in calculation preference values.

Robot(1).distance= [23.3 34 7] shows robot 1's distance to target1, target2 and target 3 were 23.3, 34 and 7 units respectively. These distance vector was calculated by the robot as it knows the location of all the targets. By broadcasting or multicasting this distance vectors were shared with other robots.

Preference vector determines the interest of the robot to serve its targets. The preference values in the preference vectors take values from 0 to 1. The higher the preference value, the greater robot wants to serve the target. The sum all preference values in a preference vector should be equal to 1. These preference vectors were stored in robot (i).preference vector. A sample preference value for the robot 1 should have the following format Robot (1).preference vector = [0.231 0.152 0.617]. It shows that robot 1's preference value to target1, target2 and target3 were 0.231, 0.152 and 0.617 respectively. It also shows that robot 1 have more preference towards target 3 as its preference value was 0.617 which was far greater than preference values towards other targets. After a series of iterations, the values in the preference vector may converge to the solution [0 0 1] depending on the preference vectors of other robots. If Preference vector for robot 1 converges to [0 0 1], it shows that robot 1 was assigned to target 3.

Robot(1).name = 'ruby' was just assigning a name ruby to the robot 1 for convenience. Robot(1). IP address = '192.168.3.10' was the way how one was going to assign IP address in MATLAB simulation to a robot 1. When working on the real robots the way of assigning IP address to the robots was different. These IP addresses will help the robots to communicate with each other or forming groups etc.

The messages sent between robots can be of different types. They can send the information used for processing or just information to store. If a robot sends its preference vector to all other robots, it should make sure that information is received by all other robots. If the robot sends its name to all other robots, it doesn't care even if all other robots don't receive it. Former way of sending the information requires TCP connection where as the latter requires UDP.

The robot parameters like IP address, name, location were entered through MATLAB. Initial target locations were also known to the robots. So they were provided to the robots manually.

```
Enter Ip address'192.168.3.10'  
Enter robot location[5 3]  
Enter robot name'y'  
Enter Ip address'192.168.3.2'  
Enter robot location[13 8]  
Enter robot name'z'  
Enter Ip address'192.168.4.1'  
Enter robot location[2 11]
```

Figure 21: shows how robotic parameters were entered in MATLAB.

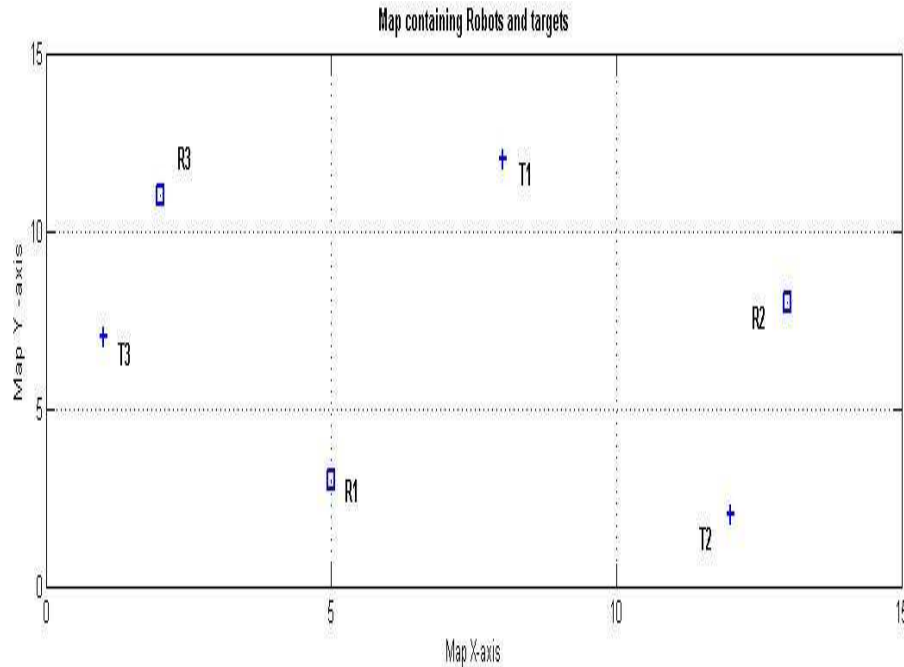


Figure 22 : generated by MATLAB showing robots and targets in a given map.

The map figure 22 was generated for the initial location co-ordinates for robots and targets which were given in robot and target modelling. R1, R2 and R3 represent robots and T1, T2, T3 represent targets. This map was shared by all robots. When ever robot moves from place to place, it sends its location to all other robots which help them to update the value in their map and database.

Initially, when all the robots and targets were in place, then starts the time for the robot to act. Robot starts calculation its distance to all other targets. The distances calculated were Euclidean distances from robot to target. Once it had done calculating, it creates distance vector in its memory. The distance vector was a collection of distances to all its targets. The first element in distance vector tells about the distance from the robot to 1st target and 2nd value will tell about distance to the 2nd robot. Once the distance vectors

were ready, each robot starts broadcasts those distance vectors to all other robots. It also receives distance vectors from other robots, which helps robot to create distance vector matrix in its memory. The distance vector matrix is very important in calculating preference values and direction vectors.

Distance calculation from robot to targets was done by the following formula.

$$D_{rj} = || r_i(0) - g_j || \text{-----} \quad \text{Equation 8}$$

r_i to g_j represents the EUCLIDEAN distance from robot 'i' to target 'j'. $r_i(0)$ represents that robot location at time 0 i.e., robots initial location. The Distance vector matrix was created when other robots send their distance vectors. Distance vector matrix obtained from the MATLAB was shown below:

	T1	T2	T3
R1	9.4868	7.0711	5.6569
R2	6.4031	6.0828	12.0416
R3	6.0828	13.4536	4.1231

Figure 23 : Distance vector matrix in Robot 1 Obtained from MATLAB simulation

Row 1 in distance vector matrix was the distance vector of robot1. Row2 and Row3 represents the distance vectors of robot 2 and robot 3 in robot 1's database received from the broadcast messages sent by robot 2 and robot3. If any broadcasts containing distance were received from 2nd robot, robot1 will update the 2nd row its distance vector matrix. Distance vector matrix helps in calculating initial preference values. The initial preference values from robot to target were calculated by using the equation 1.

The equation 1 shows how the initial preference values were calculated. These preference values range from 0 to 1. E_{ij} represents the preference value from robot i to target j . Once the preference values were calculated each robot creates a preference vector which had preference values to all other targets. Once the preference vectors were created, the robot sends preference vectors to all other robots. Robots also receive preference vectors from other robots and create a preference vector matrix in their memory.

	T1	T2	T3
R1	0.2186	0.3517	0.4297
R2	0.4453	0.4655	0.0892
R3	0.4413	0	0.5587

Figure 24: shows initial preference vectors Obtained from MATLAB simulation.

Robot uses preference vector matrix to calculate new preference vector for it. The new preference vector is broadcasted to other robots. The robots which receive the preference vector update their preference vector matrix and calculate their own preference vector and broadcast it to other robots. In this way, the whole process was repeated until the preference matrix converges to a matrix having only one element in a row and column was 1 and all other elements were zero. These preference vectors were calculated interdependently.

Peter Molnar's coupled equation was used for calculating the new preference vectors from the existent preference vector matrix. Please refer equation 2.

When this coupled equation was used and the simulation was run for 84 iterations for beta value =0.6 and k =1, the preference vector matrix converged to 1's and 0's, where there was only one element with value in row each row and column. The target assignment was done based on the obtained output.

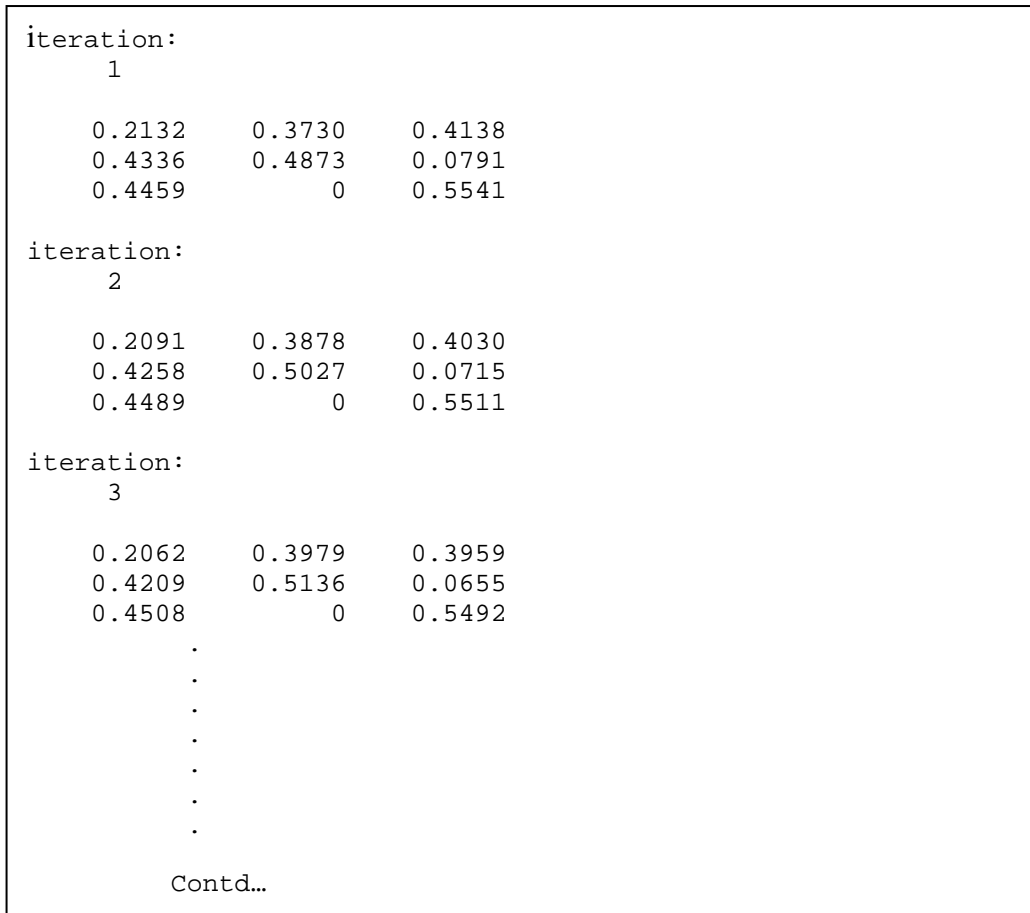


Figure 25: Initial Iterations for the Peter Molnar's Coupled equation

These initial iterations show that the preference vector matrix was converging, with only one value was increasing while the other two values were diminishing in a row.

```

Contd.. from figure 25

      .
      .
      .
      .
      .
iteration:
  83

    0.0000    0.9999    0.0001
    0.9999    0.0001    0.0000
    0.0001         0    0.9999

iteration:
  84

    0.0000    1.0000    0.0000
    1.0000    0.0000    0.0000
    0.0000         0    1.0000

```

Figure 26: converged preference vector matrix at iteration 84

The final iterations show that the solution was converged at 84th iteration. The 84th iteration shows that preference vector matrix had the value 1 in the column3 of row 1, column 2 of row 2 and column1 of row 3. It gives us the implication that target assignment was done.

The important factors observed while this simulation were beta and k factors. If the beta value goes beyond 1, negative values crept in to the preference vector matrix. The more the k value in equation the faster the preference vector matrix converges.

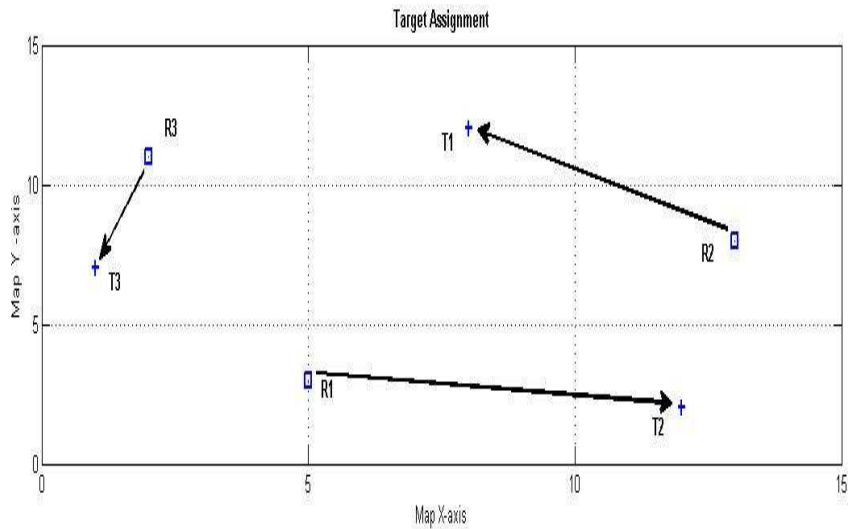


Figure 27: Map generated by MATLAB showing the target assignment

Above figure 27 shows that target assignment was done among the robots. Peter Molnar's algorithm does not have any obstacles in place.

Obstacle was being introduced in the above environment. The obstacle mostly preferred was a rectangle. Other shapes can be still used but to simplify only rectangular obstacles were taken in to consideration. Introduction of an obstacle changes the preference values from robot to target.

Obstacle Modeling:

The obstacle was introduced for the same above simulation. The obstacle was simulated in the MATLAB following way.

```
x=[0.5 4 4 0.5]
y=[8 8 10 10]

fill (x,y,'y');
```

Figure 28 : Obstacle creation using MATLAB

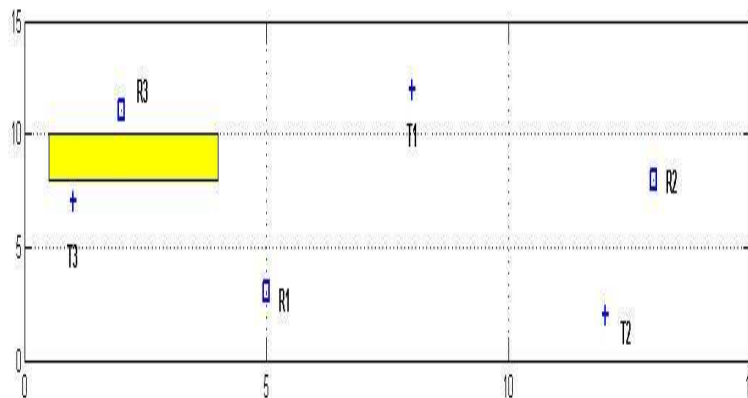


Figure 29: Map generated by MATLAB software with robots, targets and obstacle.

Figure 29 was same as figure 22 except that an obstacle was introduced in to the environment. The map had an yellow obstacle in between robot (R3) and Target (T3). The introduction of obstacle changes the cost from robot to target in the environment, preference values and finally target assignment.

	T1	T2	T3
R1	9.4868	7.0711	5.6569
R2	6.4031	6.0828	12.0416
R3	6.0828	13.9466	8.5560

Figure 30: Distance vector matrix generated by MATLAB when obstacle in place.

The distance vector matrix shown in figure 23 was changed to the above values because of introduction of an obstacle. Now initial preference values were calculated using the equation. The preference value matrix obtained as the following figure:

	T1	T2	T3
R1	0.2272	0.3503	0.4224
R2	0.4357	0.4542	0.1100
R3	0.5933	0	0.4067

Figure 31: preference values generated by MATLAB when there was an obstacle

The preference values in figure were different from that of in figure. The change in these values occurred due to presence of an obstacle. The final preference vectors shown in the following figure implicates that target assignment.

iteration		
82		
0.0000	0.0000	1.0000
0.0000	1.0000	0.0000
1.0000	0	0.0000

Figure 32: final preference values

The final preference vector shows that Robot 1 was assigned to Target 3, Robot 2 was assigned to Target 2, and Robot 3 was assigned to Target 1. All the robots found their new targets because of the presence of the obstacle in the environment.

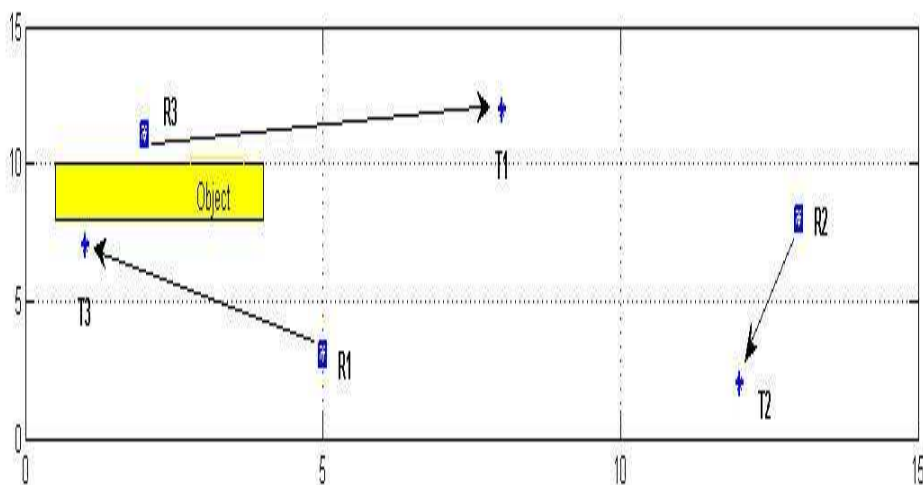


Figure 33: Target assignment in the case of an obstacle using approach 2

Comparison of Approach 1 and Approach2 for Drto Calculation:

Approach 1 and Approach2 for Drto calculation were compared in terms of distance and computational time in MATLAB. The simulation results show that Approach2 was more efficient than Approach1 in all cases. The distance calculated by approach 2 is always

smaller than approach 1 and also computational time. This makes approach 2 better than approach 1 for drto calculation.

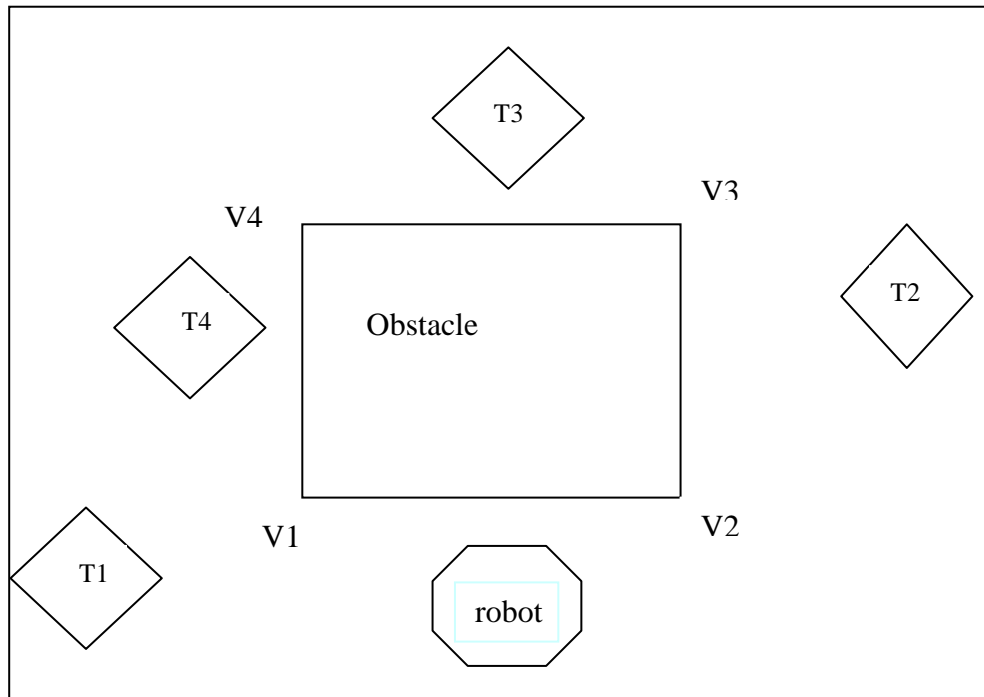


Figure 34: Robot and targets in place for comparing approach1 and approach2

In figure 34, shows the map with robot and targets were in place for calculating the distance and time taken for calculation for approach 1 and approach2. The robot was placed at co-ordinates [8 3]. T1[1 2], T2[12 8], T3[9 13], T4[2 8] were co-ordinates of target1, target2, target 3 and target 4 respectively. The case1 was done by taking robot and target1 in to account and case 2 was done by taking robot and target2 in to account and so on. The distance and computational time was calculated for all the four cases using approach1 and approach2. The resulting graphs are shown as below:

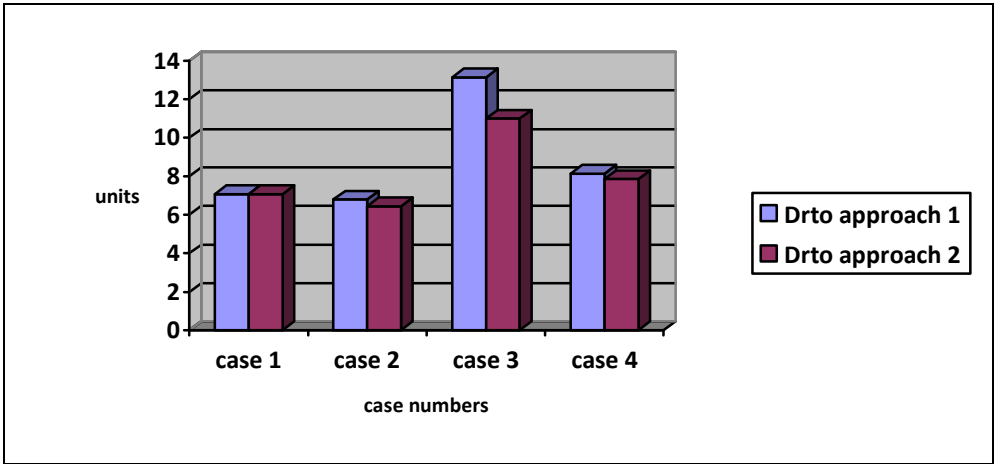


Figure 35: Distance comparison between approach1 and Approach2 for different cases

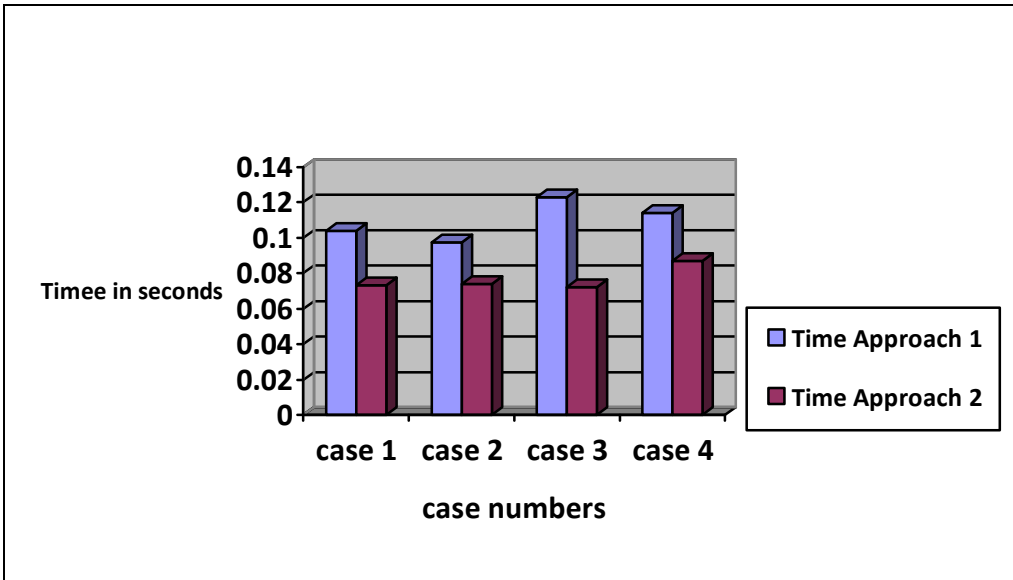


Figure 36: Computational Time taken for Approach 1 and Approach 2 while calculating Drto for different cases

Observing figure 35 and figure 36, the computational time and distance calculated using the approach2 was less in all cases. This implicates that approach 2 for calculation of Drto was better than that of approach 1. By using approach 2, the motto in peter molnar's algorithm of shortest path to the target by avoiding overhead cost can be maintained in the presence of obstacle.

CHAPTER VI

CONCLUSIONS & FUTURE WORK

This chapter deals with the results of the thesis. The conclusion from this thesis helps to understand the contribution this research has done to the Task assignment in robots.

Conclusions from the research:

The Peter Molnar's task assignment which was considered as basis for this research had some limitations. Some of them were the obstacles were avoided by proximity sensors and topology of the obstacle was not taken in to consideration while calculating distance or preference values from robot target. The motto of the Peter Molnar's task assignment was to assign one robot to target by minimizing the distance and overhead cost. If the obstacle's topology was not taken in to consideration the motto was not fulfilled. In this research, the obstacles were taken in to consideration. The obstacles were modeled, reconstructed and taken in to consideration while calculating the distance from robot to target. Two approaches were designed to calculate the distance from the robot to target and they were compared to select the best one. The task assignment was then done based on the results obtained from the best approach.

The resulted task assignment shows that assignment was done by minimizing the robot's distance to the target and over head cost through the adaptive path along the perimeter of the obstacle. The only concern was the robot does several initial calculations for calculating the initial preference values and this process requires time. But this helps in assigning the robots to their correct targets by using the correct distances. The Drto

calculation procedure calculates the lowest distance from robot to target when obstacle was in between them. The task assignment was then done by using coupled selection equations [1] which shows the obtained result agrees with theoretical approach.

Future work:

In this research the obstacle was considered as static and its vertices were known to the robot. The better way can be robot it self try to find the topology of the object and targets dynamically. This can be done by using location sensors such as cricket sensors and use some localization algorithm or use some kind of GPS systems available. The distance calculation which was shown in the research by asking perimeter of the object can be extended to multiple obstacles with any kind of shapes in an environment.

The obstacles were considered as immobile in this research and this algorithm can be modeled in such a way that it can handle dynamic objects too.

Motion planning was considered but not given more weight in this research. Motion planning was important in showing the robot movement. This motion planning equation can be modeled in such a way to avoid the objects.

All the robots were assumed to have similar capabilities and targets were taken as identical. The robots and targets can be given some suitability value which helps the robots to address the targets it can serve. Even if there more number of targets, instead of taking all the targets in to account and coming up with one target to serve, the robot can make a subset of the targets which suits it and perform task assignment to those targets.

REFERENCES

LIST OF REFERENCES

- [1] P Molnar and J Starke, "Communication fault tolerance in distributed robotic systems". Distributed Autonomous Robotic Systems 4, L.E. Parker, G. Bekey, and J. Barhen, 99-108, 2000
- [2] Esteves, J.S. Carvalho,A. and couto,c., "Generalized geometric triangulation algorithm for mobile robot absolute self localization". IEEE international symposium on Industrial Electronics 2003,June 2003.
- [3] Zachary James Z, "Flexible autonomous robots task scheduling using advanced RISC machines", thesis summated to Computer Science Department, Wichita State University May2008.
- [4] Stephen J. Chapman, " MATLAB Programming for Engineers" third edition.2006.
- [5] Acroname, Garcia manual, <http://www.acroname.com/garcia/man/man.html>, Date cited: 2/3/2007, Date retrieved:8/23/2007.
- [6] Symbolic Math toolbox, http://texas.math.ttu.edu/~gilliam/ttu/mlhtml_1/symbolic/symbolic.html, Date cited: 4/9/1996, Date retrieved: 5/21/2009.
- [7] Generalized Assignment Problem, http://en.wikipedia.org/wiki/Generalized_assignmentproblem, Date cited:6/1/2008,Date retrieved:4/25/2009.