A HYBRID VIRTUAL FORCE FIELD MODEL FOR AUTONOMOUS MOBILE ROBOT NAVIGATION

By

AYOMOH, Michael Kweneojo
B.Eng (Ilorin), M.Sc (Ibadan), MNSE
(039047007)

Thesis Submitted to the School of Postgraduate Studies, University of Lagos, in partial fulfilment of the requirement for the degree of Doctor of Philosophy

In

Systems Engineering

Department of Systems Engineering
Faculty of Engineering
University of Lagos, Nigeria

November, 2008

SCHOOL OF POSTGRADUATE STUDIES
UNIVERSITY OF LAGOS

CERTIFICATION

This is to certify that the Thesis:

"A HYBRID VIRTUAL FORCE FIELD MODEL FOR AUTONOMOUS MOBILE ROBOT NAVIGATION"

Submitted to the
School of Postgraduate Studies
University of Lagos

For the award of the degree of
DOCTOR OF PHILOSOPHY (Ph. D)
is a record of original research carried out

By

AYOMOH, MICHAEL KWENEOJO
in the Department of Systems Engineering

AYOMOH, MICHAEL K.	[illegible]	27/11/08
AUTHOR'S NAME	**SIGNATURE**	**DATE**
Prof V.O.S. Olunloyo	[illegible]	27-11-2008
1ST SUPERVISOR'S NAME	**SIGNATURE**	**DATE**
Professor O Ibidapo-Obe	[illegible]	27-11-08
2ND SUPERVISOR'S NAME	**SIGNATURE**	**DATE**
Prof O. Abass	[illegible]	27/11/2008
1ST INTERNAL EXAMINER	**SIGNATURE**	**DATE**
Dr. O. Damisa	[illegible]	27-11-08
2ND INTERNAL EXAMINER	**SIGNATURE**	**DATE**
Prof James Katende	[illegible]	27-11-2008
EXTERNAL EXAMINER	**SIGNATURE**	**DATE**
Dr E.O Oyeyemi	[illegible]	27/11/2008
SPGS REPRESENTATIVE	**SIGNATURE**	**DATE**

DECLARATION

I declare that this thesis is a record of the research work carried out by me. I also certify that neither this nor the original work contained therein has been accepted in any previous application for a degree.

All sources of information are specifically acknowledged by means of references.

[illegible]

AYOMOH, M.K.O

15/12/08

Date

DEDICATION

This research work is dedicated to God Almighty for seeing me through from the inception time to the end.

ACKNOWLEDGEMENTS

My profound gratitude goes to my Supervisors: Distinguished Professor V.O.S. Olunloyo and Prof. O. Ibidapo-Obe for their relentless effort and extreme commitment to ensuring that all physical and mental resources needed for this research work are made available.

I will specially thank the family members of my Supervisors for their unique understanding, diligence and strong spirit of accommodation throughout the course of this project. I must uniquely express my sincere gratitude to the family of Distinguished Prof. V.O.S. Olunloyo for being instrumental to the ready availability of most of the imported components I used in building the prototype machine. Even at very short notices, they never hesitated.

Special regards goes to the Head of Department Prof. O.A. Fakinlede for his encouragement and ever willing desire to assist especially in the administrative issues around the completion of this research. I am also grateful to the PG Coordinator Dr. T.A. Fashanu for his commitment and time ensuring that no stone was left unturned in the processing of results in the Graduate School.

My warm regard goes to all members of staff of Systems Engineering Department for their rare concern and support in diverse ways. In addition, I am expressing my sincere gratitude to all my senior colleagues, colleagues and friends in the faculty of Engineering for their numerous support. My warm regard goes to Engr. Adedayo, Mr. Fofo and others in the Faculty workshop for their availability and assistance whenever the need arose while fabricating the chassis of the prototype robot.

Many thanks to Distinguished Prof. V.O.S. Olunloyo's research team members. The constructive criticism during our Saturday meetings has significantly influenced not just my PhD work but the totality of my person as an academic. At this point, I say thanks to Prof. R.I. Salawu, Dr. C.A. Osheku , Dr O. Kamiyo and other members of the team.

Many thanks to Distinguished Prof. V.O.S. Olunloyo's administrative workers for their ever readiness to support even at odd times. Most significantly, I say thanks to Mr. O. Moshood, Mr. Robert, Mr. Femi and others.

My warmest regards goes to my family members. I say thanks to my Dad, Mr. F.I. Ayomoh, Mum, Mrs. P.I. Ayomoh and siblings: Mr. S.O. Ayomoh, Miss Winifred, Elizabeth and Roseline Ayomoh for being their always for me. I appreciate their timeless support in prayers, financial assistance and moral support.

I am also indebted to my friends in the Catholic Charismatic Renewal for their prayerful support. Finally, I wish to say thank you to everyone who has in any way influenced the completion of this research work. May God bless you all.

In conclusion, I give all the glory to God Almighty for reasons too numerous to be mentioned.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATIONS

x_r	- Robot x-coordinate
y_r	- Robot y-coordinate
$c_{i,j}$	- Dimensional description of robot's active window
dia_{robot}	- Diameter of robot
$d_{obstacle}$	- Distance between the robot and an obstacle in the robot's active window
$d_{t\arg et}$	- Distance between the robot and the desired goal state
Fcr	- Repulsive Force variable of sensed obstacle
rel_angle	- Orientation of active sensor relative to robot head
y_grid	- Gridline equivalent on the y-axis of cell currently occupied by the robot within the active window
x_grid	- Gridline equivalent on the x-axis of cell currently occupied by the robot within the active window
i	- Range of robot's active window on the x-axis.
j	- Range of robot's active window on the y-axis.
$x_{obstacle}$	- Obstacle's x-coordinate
$y_{obstacle}$	- Obstacle's y-coordinate
r_c	- Positioning distance between robot and virtual goal
$s(n)$	- Number of sensors on robot
$x_{t\arg et}$	- x coordinate of robot goal state
$y_{t\arg et}$	- y coordinate of robot goal state
${}_{initial}U^{y}_{rep}(q_r)$	- Initialized repulsive force (y-component)
${}_{current}U^{y}_{rep}(q_r)$	- Current repulsive potential generated by the obstacle (y-component)
${}_{initial}U^{x}_{rep}(q_r)$	- Initialized repulsive force (x-component)
${}_{current}U^{x}_{rep}(q_r)$	- Current repulsive potential generated by the obstacle (x-component)

$U_{att}^{y}(q_r)$	- Attractive force generated by the goal state (y-component)
$U_{att}^{x}(q_r)$	- Attractive force generated by the goal state (x-component)
$U_{rep}^{x}(q_r)$	- Repulsive potential generated by obstacle(s) within sensing range (x-component)
$U_{rep}^{y}(q_r)$	- Repulsive potential generated by obstacle(s) within sensing range (y-component)
$U^{x}(q_r)$	- Resultant potential in the x axis
$U^{y}(q_r)$	- Resultant potential in the y axis
$U(q_r)$	- Combined resultant potential from x and y axes
$U_{att}(q_r)$	- Total attractive potential from the target point
$U_{rep}(q_r)$	- Total repulsive potential from obstacles
θ	- Initial orientation of robot after rotational difference *(before navigation).*
δ	- robot's orientation during navigation
ψ_1	- Orientational difference between initial and current orientation: case of the left wheel
ψ_2	- Orientational difference between initial and current orientation: case of the right wheel
∇	- Grad operator
q_r	- Free configuration space describing current robot position
$q_{t\arg et}$	- Free configuration space describing target point
$d_{vi\_t\arg et}$	- Distance of the virtual goal from the real goal
v_i	- Generalized representation of virtual goal
v_{xi}	- x coordinate of virtual goal
v_{yi}	- y coordinate of virtual goal
r_c	- Incremental distance for virtual goal placement
s_o	- Minimum permissible sensor reading
s	- current sensor reading

ABSTRACT

This work improves upon the Potential Field Method (PFM) for the navigation of autonomous mobile robots, through the introduction of a new Hybrid Virtual Force Field (HVFF) concept. The HVFF concept integrates the virtual force field (VFF), which is based on the principles of artificial attractive and repulsive potentials, with the virtual obstacle concept (VOC) and the virtual goal concept (VGC).

In particular, the specific challenges resolved by the HVFF, include the local minima problem posed by either lengthy or concave shaped obstacles as well as the potential field induced oscillatory motion of such a robot when maneuvering in the corridor between two narrowly spaced obstacles. These were all studied in a static obstacle domain. Furthermore, we extend our solution technique to the unstructured obstacle environment all in a 2-D domain. In this context the general problem of dynamic obstacles in a completely unknown environment was examined in detail.

Simulations of the motions were used to validate the efficacy of HVFF over existing algorithms for the dynamic and static obstacle architectures using MobotSim (a customized software for robot animation).

Furthermore, we also confirm the feasibility of the new HVFF concept by demonstrating the performance of a prototype robotic vehicle designed, built and operated as an implementation platform for the HVFF algorithm.

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

Autonomous robot path planning with obstacle avoidance is a fundamental and important problem in the study of robotics. Navigation is one of the most important tasks in intelligent control of an autonomous mobile robot. The degree of intelligence, dexterity and versatility of a robotic vehicle in respect of task performance is a function of so many varying factors. Infact the last few decades have recorded a paradigm shift from remotely operated to autonomous robotic vehicles. Consequently, research to enhance autonomous navigation has received so much attention as seen from recent literature. In this regard, for a machine to be christened "autonomous" it must possess a sense of direction independent of human intervention while it navigates.

The problem of navigation can be summarized using the three cardinal questions namely: where am I?", where am I going?", and how should I get there?" The first question is one of localization: The second and third questions are essentially those of specifying a goal and being able to plan a path that results in achieving this goal. Investigations of the latter two questions usually come under the domain of path planning and obstacle avoidance.

One of the most important advantages of using a reactive control strategy is that it is online compliant and has the ability to cope with unstructured environments. To achieve its goal, the robot must be able to perceive its environment sufficiently to allow it navigate safely. In recent times, some areas of success have been reported in the literature but nonetheless research in autonomous mobile robots is still in its infancy, and extensive research is going on for improvements to make their widespread use possible.

According to Sugihara and Smith (1997), motion planning is one of the important tasks in intelligent control of an autonomous mobile robot. Path planning according to Cleghorn et al. (1988) could refer either to a mobile vehicle or to an end effector on an arm moving through a cluttered workspace. In both instances there may exist many solutions, some of which are better than others, either in terms of distance traversed, energy expended, joint angle or even reach capabilities.

Robot Path planning, in a broader sense could be referred to as the process of identifying obstacle free configurations within a given workspace in order to enhance a collision free navigation of a robot from its current position to its desired position. To be sure, Path planning for mobile robots is a complex problem that does not only guarantee a collision-free path with minimum traveling distance but also requires smoothness and clearances.

Two fundamental classifications suggested by Fu, Gonzalez, and Lee (1987) to describe the robotic path planning problem are **Obstacle Constraint** and **Path Constraint**. **Obstacle Constraints** indicate that there are some points in space which are already occupied, and are not free for the robot to pass through. **Path Constraints** are usually provided as points on a path which the robot must follow. As a robotic vehicle translates and orientates her member components at different points with time and in space, while accomplishing an assigned task, it does this in some defined paths ensuring that obstacles are avoided both locally and globally. Local and global path planning are next discussed below:

1.1.1 **Local Path Planning** (sensor based planning)

Sometimes information may not be available at the inception of solving a problem, thus we must solve the problem in stages as information are progressively made available. Sensor based planning is an indispensable function when environments change with time, are unknown, or there are inaccuracies in the robotic equipment. A postieri knowledge can be used to find the next trajectory in a path (by collecting information about the outcome of the previous trajectory) or may also be used to guide the robot in a random sense when exploring an environment. These techniques correspond to an **"execute and evaluate"** strategy. The information feedback in such cases is acquired with the aid of sensors while the sensors used may range from vision systems to contact switches.

1.1.2 **Global Path Planning** (knowledge based planning)

It is much easier to solve a problem if all the information needed about the workspace is available at the beginning and prior to the onset of motion. In robotics we may plan paths before their execution if we have sufficient knowledge of the environment. Planning paths before execution facilitates solutions of a shorter path time, more efficient dynamics, and absolute collision avoidance. When working in this mode, **a priori** knowledge (i.e. known before) is used. Techniques are available to solve a variety of problems, when given a priori information. Some of the knowledge which we use for a priori path planning may come from different sources such as vision systems, engineering specifications, or CAD programs. Such a priori knowledge may also be applicable to moving objects, if they have a predictable frequency or motion. However, a priori knowledge cannot be used for unpredictable or random moving objects.

1.2 Statement of Problem

The establishment of a trajectory to navigate a robot from an initial position to a target point within a 2-D workspace clustered with static or mobile obstacles forms the basis of our problem definition. It follows that the path planning problem is an optimization problem that involves computing a collision free path between two locations viz: the current position of the robot and the desired position of the robot. Consider a 2D workspace W whose configuration is such that the Robot R is a single rigid object moving in a Euclidean space defined by $W = C^N$, where the dimensionality $N = 2$. Let $\beta_1........\beta_n$ be rigid obstacles distributed in W and q is the free configuration space. If R is described as a compact subset of W, then a configuration of R is a specification of the position of every point in R relative to a fixed cartesian coordinate system. Assuming the moving robot R is subject to both attractive and repulsive potentials resulting from the target position and obstacles respectively then one of the cardinal questions in robot navigation "how do I get to the target?" is asked. Getting to the target point without any form of obstacle collision along an optimal path is the first challenge posed. Another issue is the likelihood of the robot getting trapped in local minima as a result of some specific type of obstacle configurations. These obstacle types as discussed below are shown in Figs. 1a, 1b and 1c. Over the years researchers have proliferated the literature of robot navigation for different navigation schemes without leaving out the imaginary force techniques. During the past few years, the idea of imaginary forces christened the potential field method (PFM) acting on a robot was suggested by Andrews and Hogan (1983) and Khatib (1985). In this robot navigation approach, obstacles exert repulsive forces on the robot, while the target applies an attractive force to the robot. The resultant force determines the subsequent direction of motion.

The PFM is popular with researchers in the domain of autonomous navigation due to its simplicity (relatively lesser computation time), continuous nature (adaptive to changing environment) and its ability to generate smooth paths. Borenstein and Koren, (1989) developed a potential field (PF) driven method christened the virtual force field (VFF) technique. Some of the specific problems associated with the VFF concept that were discussed in their paper are inherent in the PFMs. The following underlisted problems form a highlight of these associated problems.

(i) Trap situations due to local minima
(ii) Inhibited passage between closely spaced obstacles
(iii) Oscillations in the presence of obstacles and
(iv) Oscillations in narrow passages.

The specific problems to be addressed in this research work include the local minima problem posed by:

(i) lengthy obstacles
(ii) concave shaped obstacles as well as
(iii) the potential field induced oscillatory motion of a robot when maneuvering in the corridor between two narrowly spaced obstacles.

Illustrated in Figs. 1a-1c below are some of the flaws associated with the VFF approach. In particular Fig. 1a illustrates the occurrence of a local minimum trap occasioned by a concave shaped obstacle which results in continuous re-circulation of the robot within the trap while in Fig. 1b the dimensions of the intervening obstacle is such that the additional force gained from mere displacement to either side is insufficient to take it across the line of the obstacle towards the goal thereby leading to some zig-zag or irregular motion. On the other hand Fig. 1c illustrates the oscillatory motion of the robot when maneuvering between two narrowly spaced obstacles as it is successively repelled by the nearer of the two obstacles as it moves within the corridor.

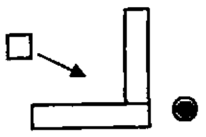


Figure 1a: local minimum trap by a concave shaped obstacle

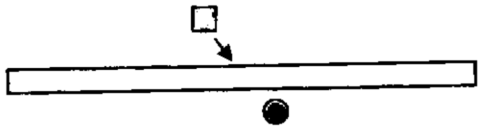


Figure 1b: local minimum trap by lengthy obstacle.

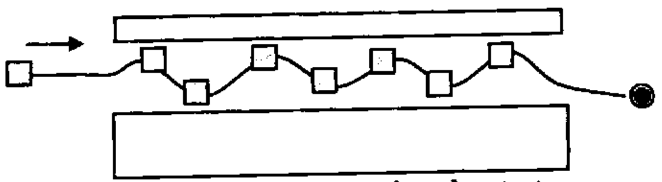


Figure 1c: oscillatory motion due to two narrowly spaced obstacles

Robot	□
Target	●

1.3 Objective of Study

Our research aims at finding a suitable algorithm based on the virtual force field (VFF) concept, to control the navigation of a robot in a 2-D domain containing either static or mobile obstacles. The proposed navigation scheme is to be validated firstly through simulation experiments on:

(i) varying workspace formations of complex static obstacles in a partially known environment and

(ii) varying workspace formations of dynamic concave shaped obstacles in an unknown environment.

(iii) Following this is a second stage of validation which will be conducted on the platform of a prototype mobile vehicle.

1.4 Motivation for the present work

Our motivation for this study is hinged in part on the growing demand for automated and intelligent systems in different facets of human life and various sectors of the economy in the twenty-first century. These systems range from stationary intelligent sensing and monitoring systems such as surveillance and monitoring stations, airport entrance and exit channels, banks, shopping malls etc to industrial robots such as robotic assemblers in automobile plants, industrial robots for mixing, pick and place operations in hazardous environments such as in tobacco industries, asbestos industries, etc. The deployment of dynamic autonomous navigation systems is also becoming standard practice in high risk operations or hostile environments such as with the Mars rover, intelligent subsea rovers, unmanned aircraft systems, unmanned ground vehicles, etc.

More often than not, the robots we see in science fiction movies appear to navigate with effortless precision; however, in reality mobile robot navigation is a fairly formidable problem. Indeed, answering the basic question; "where am I?" in a truly autonomous fashion is a serious challenge for today's mobile robot. The technical advancements in the twentieth century made possible the advent of robotic systems. Initially, these systems were used in manufacturing industries as industrial robots deployed to improve product quality and reduce overall manufacturing cost. Next came field robots which handle tasks such as space exploration, deep sea research, mine exploration, data collection in volcano prone areas, surveillance and rescue operations, etc. This development has facilitated the emergence of another sub-class of devices referred to as service robots. Service robotics addresses the increasing affinity in man-machine interaction such as is found in robot-based cleaning/sanitary operations, delicate surgical operations guided by robotic devices as well as in the general field of entertainment, etc. The relative flexibility and reprogrammable capability of these microprocessor driven systems has resulted in the advent of flexible automation. The effect of such skills in promoting technical and economic advancement is wide and encompassing, dynamic and global.

1.5 Scope of the Study

In Chapter one the thesis is introduced. Chapter two is a review of the literature. In Chapter three the static obstacle problem for complex mazes in a partially known environment is solved. In Chapter four the algorithmic procedure and formulation for the dynamic concave shaped obstacle problem in a completely unknown environment is presented. Chapter five is a description of the prototype machine developed for the implementation of our scheme while Chapter six is a presentation of simulation results for the static obstacle problem, as well as the dynamic concave shaped obstacle problem and the built machine. Chapter seven is the conclusion which contains the summary of our findings, the contribution to knowledge and identified areas for future research work.

1.6 Significance of the Study

The robustness and versatility of application areas for autonomous mobile robots (AMR) is all encompassing and provides a good indicator of the significance of this research. The quest for improved or enhancing navigation algorithms capable of meeting the overall navigation objectives especially in relation to effectiveness and efficiency amidst workspace obstacles is unending and forms a core area of activity in today's literature. The underlisted examples give a few of the areas of application of AMR.

- Automated Guided Vehicles (AGVs) in harbors and airports: These are mobile robots that are used in large facilities such as warehouses and container ports, for the movement of goods, or even for safety and security patrols. Such vehicles follow wires, markers or laser-guidance to navigate around the work space and can be programmed to move between locations to deliver goods or patrol a certain area.

- For high risk operations such as bomb disposal or hazardous material management, which would be potentially dangerous for humans.

- For subsea pipeline corrosion and leakage detection and high level precision or spot welding

- In the area of geo-resources tracking by integrating the concept of electronic nose sensing for detecting mines especially in human unfriendly terrains.

- For enhancement of agricultural technology in developing nations e.g. development of robotic planting machines, fertilizer applicators etc

- In medicine: Robots are being used to perform highly delicate and precision surgery. Also, they allow a surgeon who is located remotely from their patient to perform a procedure using a remotely controlled robot. More recently, robots can be used autonomously in surgery.

- For automatic navigation of vehicles in constrained environments, missile guidance, aircraft routing

- For surveillance, rescue operations and military technologies.

- Unmanned ground vehicles (UGVs) have important military reconnaissance and materials handling applications. Many of these applications require an UGV to move at high speeds through uneven, natural terrain with various compositions and physical parameters.

The main use of robots has to date been in the automation of mass production industries, where the same definable tasks must be performed repeatedly in exactly the same fashion. Robots are used in that capacity for painting, welding and assembly of cars. Robots are particularly good for such tasks because the tasks can be accurately defined and must be performed the same way every time.

As indicated earlier, they are also useful in environments which are unpleasant or dangerous for humans to work in, for example bomb disposal, work in outer space or underwater environment, in mining and for the cleaning of toxic waste. Robots are now being used for patrolling toxic areas; the Robowatch OFRO, and Robowatch MOSRO are examples of such deployment.

For the home environment, domestic robots are now available that perform simple tasks such as vacuum cleaning and grass cutting. By the end of 2004 over a million of such vacuum cleaner units had been sold. Examples of these domestic robots are the Scooba and Roomba robots from iRobot Corporation, the Friendly Robotics' Robomower, Electrolux's Automower, and Samsung. Thus ability to propose advancement in path planning technology will contribute positively to the safe and efficient deployment of such devices in the economy.

1.7 Operational Definition of Terms

Path Planning: This is a process of identifying obstacle free configuration spaces in a given navigation environment in order to facilitate the smooth navigation of a robot from its initial state to desired target point without any form of collision with the obstacles in the environment.

Reactive Algorithm: A reactive algorithm is defined as one which facilitates the process of continuous adaptive decision making by the robot through the process of replanning while navigating to its goal position.

Static Obstacles: An obstacle is said to be static when its position and orientation relative to a known and fixed origin does not change with time.

Dynamic Obstacles: An obstacle is said to be dynamic when either its position or orientation or both relative to a known and fixed origin does change with time.

Workspace: The area (2D) or volume (3D) of space within which a robot is confined to carry out its activities. It is also referred to as the environment.

Obstacle: Any item, material or object, other than the goal, within the line of sight of the robot's sensor(s) is an obstacle.

Autonomous Mobile Robot: An autonomous mobile robot (AMR) is one that is capable of performing its task absolutely without any form of human intervention.

Target point: Also referred to as the goal state, this is the desired destination of the robot.

Completely Known Environment (CKE): A robot navigation environment is christened Completely Known Environment when the robot has a predefined trajectory from its initial position to the target position prior to navigation. The implication here is that the position and geometric configuration of all the obstacles in the workspace are programmed and stored in the robot's knowledge base ahead of navigation. This type of path planning technique is confined to static environment.

Completely Unknown Environment (CUE): This type of path planning is said to be invoked when the robot has no prior knowledge of the workspace before navigation. The robot develops its own trajectory of motion progressively as it navigates to the goal state independent of human but dependent on its control algorithmic procedures and sensing system.

Partially Known Environment (PKE): A navigation type is said to be partially known (PKE) when prior to navigation, the robot already has knowledge of some areas within the workspace i.e. areas likely to pose local minima problem. These areas are identified through the process of workspace mapping usually by a human agent.

CHAPTER TWO

LITERATURE REVIEW

2.0 Preamble

This chapter gives a detailed review of literature of various robot pathplanning techniques. Researchers over the years have developed and used different concepts to solve the robot path planning problem. These concepts have been categorized into the following sub-class namely: the graphical approach, the classical methods, the heuristic approach as well as the use of the potential field methods.

2.1 Graphical Approach

Several graphical techniques such as: voronoi diagram, spatial planning, vgraph, transformed space, oct trees, cell decomposition and configuration space have been proposed and extensively used in the past few decades. Graphical methods are founded on the principles of geometry and they are dominantly known for their limitation to static environment.

In his work, Lozano-Perez (1987) discussed a generate and test strategy which is used for path planning. Another well known graphical approach is the Configuration Space method as used by Laumond (1986), Palma-Villalon and Dauchez (1988) and Tseng et al. (1988). On the other hand Meng (1988); Takahashi and Schilling (1989); as well as Song et al. (2001) suggested the Voronoi Diagrams as an approach to representing space. In fact Wong and Fu (1986) carried out a study which allows a path planning method to be run with three views of a work cell, and from these three views deduced the maximum filled volume. A method proposed by Kant and Zucker (1988) involves collision avoidance of rectangles based upon the search of a VGRAPH.

Furthermore, the Oct-tree was used by Soetadji (1986) who in addition used the breadth first search technique. Others include Faverjon and Tournassoud (1987) and Muck (1988) who focused primarily on the Oct-tree technique. Moreover, the use of slack variables was suggested vaguely in a paper by Zaharakis and Guez (1988) while Lengyel et. al. (1990)

developed a path planner based on the integration of the configuration space and some dynamic programming concepts. This facilitated easy handling of any polyhedra geometry of robot and obstacles, including disjoint and highly concave unions of polyhedra while Behring et al. (2000) on the other hand used the cellular automata concept which is dominantly hinged on the configuration space approach to grow obstacles. Lingelbach (2004a; 2004b and 2005) also proposed a novel path planning method called Probabilistic Cell Decomposition (PCD).

The problem of motion planning among multiple robots in a dynamic environment of obstacles was recently investigated by Gayle et al. (2007). Their approach was based on a new roadmap representation christened Reactive Deforming Roadmap (RDR), for multiple robots. Furthermore, they deployed Newtonian Physics and Hooke's Law to update the position of the milestones and deform the links in response to the motion of other robots and the obstacles. Their solution is however not collision free and has high computational storage requirements for multirobot scenarios

2.2 Classical Approach

A second approach which has also been widely explored over the years is the use of classical algorithms. Some of the classical methods that have been used over time include: gradient descent, steepest descent, steepest ascent dynamic programming, iterative technique, randomized methods, kinodynamic, fibonacci Improved Network Optimization algorithms etc.

Park (1984) used the state-space representation technique in his quest to find a fast path planning method while Verbeek et al. (1986) proposed the steepest descent method in context with the work of Park (1984). Montano and Sagues (1991) studied the stability limits and the dynamic behaviour as a function of the control parameters. They presented a non-contact compliant motion control scheme embedded in a multi-sensorial robotic system. Control in their case was based on the idea of generalized damping with the aim of finding the most suitable control scheme for each task. System parameters vary with the robot location and when it approaches object surfaces, the controller is programmed to take the most conservative values which gives an over-damped response in any workspace location close to the object.

Clark et al. (2000) on the other hand carried out path planning of dynamic robot networks. The system they developed enables multiple mobile robots that have limited ranges of sensing and communication to maneuver safely in dynamic, unstructured environments. The motion planning algorithm used within such networks was based on kinodynamic randomized motion planning techniques that construct trajectories in real-time. Both simulations and real robot experiments were used to validate the system. Several assumptions had to be made to allow such a concise world model. The first is that all objects can be considered circular, or approximated by a set of circular objects. This allows the geometry to be described completely by one parameter (i.e. a radius). The second assumption is that obstacles have constant velocity.

A commonly used classical technique for shortest path search technique is Dijkstra's algorithm. It finds the shortest path between two nodes in a graph and in the process also extracts the minimum cost path from all nodes to the source node. Relative distance information between all points involved is required. Obviously, this requires complete knowledge of the robot's environment and is not suitable for dynamic obstacle environment. Asaolu (2001) applied Dijkstra's algorithm to solve the shortest route problem. Furthermore, a new approach christened the intercept approach was used to solve the pursuit problem while some classical and geometrical approach were deployed in the obstacle avoidance problem. Ibidapo-Obe et al. (2002; 2006) used some classical, numerical and optimization techniques in solving problems related to intelligent MetaHeuristics. They also developed generalized solutions of the pursuit problem in three dimensional Euclidean spaces.

Around the same time, Wong et al. (2002) proposed two measures namely; percentage of coverage using computer vision and distance travelled by the robot. Their problem type is a sub-category of the multi-goal attainment problem. Carpin and Parker (2002) introduced and discussed the cooperative leader following task concept for multi-robot teams. They described the design and implementation of a distributed technique to coordinate team level and robot level behaviors for the task, as well as a multi-threaded framework for the implementation of a heterogeneous multi-robot system. Their concept enables robots to remain in formation as they deal with other obstacles that may appear within the formation. Han and Amato (2000); Song (2001) and Kim (2003) used the probabilistic roadmap technique while Trihatmo and Jarvis (2003) used the Distance Transform and the linear vector combination in the development of their path planning method. Sabe et al. (2004) later on focused on obstacle avoidance and path planning for humanoid robots using Stereo Vision.

Malik (2004) used a technique which predicts the possible future positions of obstacles. This technique is to be coupled with the generalized Voronoi diagram. The aim of the robot is to select avoidance maneuvers so as to avoid dynamic obstacles. Spenko et al. (2004) presented a model-based analysis tool and hazard avoidance algorithm for unmanned ground vehicle (UGVs) in rough terrain using the concept of trajectory space. Soucy and Payeur (2004) studied new strategies and compared them with classical ones viz: multi-resolution vs standard occupancy grids and probabilistic vs deterministic datasets. The goal of their research was to identify the best solutions for path planning and collision avoidance in semi autonomous robotic systems operating in complex environments.

Similar to the study of Yannier et al. (2003), Karlsson and Munich (2004) presented three core technologies that enable the next generation of service robots for robot configurations. The first technology is an object recognition system. The second technology is a vision-based navigation system while, the third technology is a flexible and rich software platform. The problems to be solved for each technology are respectively listed below and can:

i. be used by the robot to interact with the environment.

ii. assists developers in rapid design and prototyping of robotics applications.

iii. simultaneously be used to build a map and localize the robot in the map.

Heero et al. (2004) also presented a method for mobile robot navigation in environments where obstacles are partially unknown. The method uses a path selection mechanism that creates innovative paths through the unknown environment and learns to use routes that are more reliable. Ismael et al. (2004) described how robot trajectory planning could be formulated as a semi-infinite programming (SIP) problem. The formulation as a SIP problem allowed for the treatment of the problem with one of the three main classes of methods for solving SIP, the discretization class. Two of the robotics trajectory planning problems formulated were coded in the SIPAMPL environment which is publicly available. A B-Spline library was also created to allow the codification of the robotics trajectory problem.

Fasola et al. (2005) developed an algorithm that makes use of a single monocular camera for both localization and obstacle detection. Their paper described a technique by which a robot can visually navigate to globally-defined goal points on the soccer field while avoiding static obstacles. The algorithm alternates between two different navigation modes. When the area in front of the robot is unobstructed, the robot navigates straight towards the goal. However when the path is obstructed, the robot follows the contours of the obstacles until the way is clear.

A novel approach for trajectory optimization of a mobile robot with dynamic obstacle avoidance was developed by Belkhous et al. (2005). The new method presented in the work combined the static and dynamic modes of trajectory planning to provide an algorithm that gives fast and optimal solutions for static environments, and generates a new path when an unexpected situation occurs. The novelty of their solution method lies in the representation of the static environment in a judicious way facilitating the path planning and reducing the processing time. Moreover, when an unexpected obstacle blocks the robot trajectory, the method uses the robot sensors to detect the obstacle, finds an optimum way to circumvent it and then resumes its path toward the desired destination. Experimental results showed the effectiveness of the proposed approach.

Murrieta-Cid et al. (2005) worked on the surveillance problem of computing the motions of one or more robot observers in order to maintain visibility of one or several moving targets. The targets are assumed to move unpredictably and the distribution of obstacles in the workspace is assumed to be known in advance. The algorithm computes a motion strategy by maximizing the shortest distance to escape. The algorithms were implemented using real mobile robots for the single target case, and simulation results for the case of two targets-two observers. The probabilistic method of randomization was adopted.

Bikdash (2006) used the finite element mesh analysis and COMSOL software for the robot path planning problem. In their article, Becker et al. (2006) described the development of a new obstacle avoidance procedure based on the Obstacle Velocity approach and adapted their work to both autonomous and semi-autonomous robots. Recently, Philippsen et al. (2006) presented a work on sensor-based motion planning in initially unknown dynamic environments. Motion detection and probabilistic motion modeling are combined with a smooth probabilistic navigation function (PNF) to perform on-line path planning and replanning in cluttered dynamic environments such as public exhibitions. Their algorithm, an extension of Iterative Closest Point, combines motion detection from a mobile platform with position estimation. This information is then processed using probabilistic motion prediction to yield a co-occurrence risk that unifies dynamic and static elements. The risk is translated into traversal costs for an E* path planner. It produces smooth paths that trade off collision risk against detours.

Kunwar and Benhabib (2006) developed a rendezvous-guidance trajectory planning for dynamic obstacle avoidance and interception. This was achieved through the development of an online trajectory-planning method for the autonomous robotic interception of moving targets in the presence of dynamic obstacles. The proposed time-optimal interception method was a hybrid algorithm that augments a novel rendezvous-guidance (RG) technique with the velocity-obstacle approach, for obstacle avoidance. The obstacle-avoidance algorithm itself which could not be used in its original form was modified to ensure that the online planned path deviates minimally from the one generated by the RG algorithm.

In their paper, Li et al. (2006) analyzed the vehicle dynamics of redundantly actuated wheeled mobile robots with powered caster wheels. The condition that describes the limits of contact stability in terms of contact forces, was also derived and force distribution scheme proposed to satisfy the stable contact condition. Furthermore, a Sliding mode observer was proposed to estimate the system states and its effectiveness demonstrated by simulation. For Future work they proposed to work on the problem of slip based traction control. Also, is the plan of implementing the proposed method on the real robot.

2.3 Heuristic Approach

Another class of dominating navigation technique is the heuristic based technique. This technique is based on inferential and evolutionary theory. Usually they are associated with some degree of inconsistency. Some widely used heuristic techniques include A* algorithm, E* algorithm, genetic algorithm, mementic algorithm and simulated annealing

A widely used heuristic search approach is the A* algorithm. Originally proposed by Hart et al. (1968), the literature over the years has shown the effectiveness of integrating this concept with some form of graphical methods such as the configuration space technique which was used by Davis and Camacho (1984). The A* heuristic algorithm is used to compute optimal path from one given point to another amidst alternative routes. Here, the relative distance information between all nodes as well as the direct distance estimate from each node to goal are required, meaning that the robot must have global knowledge of the environment. It is also not feasible for a dynamic obstacle environment.

Furthermore, Brooks and Lozano-Perez (1985) researched into integration of A* into the generalized cones for the representation of Free Space. Whitley et al. (1990) combined Genetic Algorithm and Neural Networks while Shibata and Fukuda (1993) combined Fuzzy logic and Genetic Algorithm. Glasius et al. (1995) demonstrated the effectiveness of a Hopfield Neural Network algorithm with nonlinear analog neurons for path planning and obstacle avoidance. This deterministic system can rapidly provide a proper path, from any arbitrary start position to any target position, avoiding both static and moving obstacles of arbitrary shape. The model assumes that an (external) input activates a target neuron corresponding to the target position and specifies obstacles in the topologically ordered neural map. The path follows from the neural network dynamics and the neural activity gradient in the topologically ordered map. The analytical results are supported by computer simulations to illustrate the performance of the network.

Xu et al. (1998) and Xu, W.L., (2000) used a virtual target approach to resolve the limit cycle problem in navigation of a behavior-based mobile robot with the incorporation of a fuzzy behavior-based controller. The model was further validated through simulation and a real experiment with a Nomad 200 robot incorporating a fuzzy behavior-based controller.

Yang and Meng (2000) researched on a biologically inspired neural network for real-time collision-free motion planning of mobile robots or robot manipulators in a non-stationary environment. Each neuron in the topologically organized neural network has only local connections, whose neural dynamics is characterized by a shunting equation. Thus the computational complexity linearly depends on the neural network size. The real-time robot motion is planned through the dynamic activity landscape of the neural network without any prior knowledge of the dynamic environment. This is done without explicitly searching over the free workspace or the collision paths, and without any learning procedures. Therefore it is computationally efficient. The global stability of the neural network is guaranteed by qualitative analysis and the Lyapunov stability theory. The effectiveness and efficiency of the proposed approach are demonstrated through simulation studies.

Stachniss and Burgard (2002) presented an approach that integrates path planning with sensor-based collision avoidance. Their proposed algorithm simultaneously considers the robot's pose and velocities during the planning process. Their technique applies the popular A* algorithm and the grid-graph induced by the occupancy grid map to compute heuristic values in finding the shortest navigation path. The proposed technique assumes that a map representing the static aspects of the environment is given prior to navigation. This assumption classifies their technique under the sub-categories of path planning in partially known environment. Their work was tested on both simulated and real environment. In the case of simulated environment, obstacles were formed by using point objects with a convex geometry while humans and real objects formed the obstacles for the real world. In all experiments their technique was able to generate safe trajectories. They compared the proposed approach to the popular Dynamic Window Approach (DWA). The experiments accordingly demonstrate that the algorithm yields more efficient trajectories which are often close to the optimal ones.

Geisler and Manikas (2002) improved on the genetic algorithm performance by developing a more efficient genotype structure case for a known environment with static obstacles. Motion was constrained to only row-wise navigation. Sedighi et al. (2004) presented results of a genetic algorithm based path-planning model developed for local obstacle avoidance (local feasible path) of a mobile robot in a given search space. Zacharia and Aspragathos (2004) introduced a method to determine the optimum sequence of task points visited by the tip of the end effector of an articulated robot and it can be applied to any non-redundant manipulator. This method is based on genetic algorithms and an innovative encoding is introduced to take into account the multiple solutions of the inverse kinematic problem. The results show that the method can determine the optimum sequence of a considerable number of task points for robots with up to six-degrees of freedom.

Janglová, (2004) used the artificial neural network while Mbede et al. (2004) developed two robust adaptive Neuro-Fuzzy motion controllers for the navigation of mobile manipulators among dynamic obstacles. The first controller was used to generate the commands for the servo-systems of robot arm in order to allow for autonomous meandering to the goal while the second fuzzy reactive navigation was implemented to maintain a permanent flexible path between two nodes in network generated by a probabilistic roadmap approach.

Youssef (2005) developed a new Evolutionary Neuro-based Approach (ENA) that combines the neuro-based reinforcement learning capabilities with an evolutionary path planning algorithm. Du et al. (2005) combined Neural Network and Genetic Algorithm. Their paper proposed a method for global path planning in static environment and for convex shaped obstacles. Simionescu et al. (2006) discussed a new approach to solving constrained nonlinear programming problems using evolutionary computations while Qiao et al. (2006) proposed other evolutionary computation techniques.

In their paper, Valavanis et al. (2006) presented fundamental aspects of a multi layer, hybrid, deliberative and reactive Distributed Field Robot Architecture (DFRA) that has been designed to support functionality of heterogeneous teams of unmanned (ground and aerial) robot vehicles. The DFRA was implemented in Java using Jini to manage distributed objects, services and modules between robots and other system components. Navigation experiments in the field utilized single and multiple robots and included scenarios in which a single robot navigated through an environment with many unknown obstacles to reach a distant goal location and scenarios in which robots executed search routines by traveling through sets of intermediate points. Robots negotiated both static obstacles and dynamic obstacles including other robots.

In their work, Ayari and Chatti (2007) used the neuro-fuzzy controller in an unknown obstacle environment while Yu et al. (2007) worked on the detection of static and dynamic obstacles in environmental mapping of mobile robot. They presented an unsupervised clustering algorithm to realize feature extraction of obstacles based on the analysis of ranging data obtained from 2D laser scanner. Considering the unknown clustering number in advance, a validation index function was introduced into the self-learning mechanism to determine the accurate clustering number automatically. At the same time, fuzzy logic was integrated into incremental data association of obstacle features to make the static or dynamic obstacles classification decision reduce the uncertain influence. Their office was used as the operating environment to implement the experiment of feature extraction and obstacles classification.

Ordonez et al. (2008) presented a solution named the Virtual Wall Approach (VWA) to the limit cycle problem for robot navigation in very cluttered environments. This algorithm is composed of three stages: detection, retraction, and avoidance. The detection stage uses spatial memory to identify the limit cycle. Once the limit cycle has been identified, a labeling operator is applied to a local map of the obstacle field to identify the obstacle or group of obstacles that are causing the deadlock enclosure. The retraction stage defines a waypoint for the robot outside the deadlock area. When the robot crosses the boundary of the deadlock enclosure, a virtual wall is placed near the endpoints of the enclosure to designate this area as off-limits. Finally, the robot activates a virtual sensor so that it can proceed to its original goal, avoiding the virtual wall and obstacles found on its way. A fuzzy based behavioral system was used to navigate the robot. Simulation experiments were further carried out to validate the algorithm.

A new fuzzy method was presented by Jaafar and McKenzie (2008) for an action selection method. The action selection problem is based on fuzzy α-levels incorporated with the Huwicz criterion. The objective was to solve behaviour conflict in behaviour-based architectures for virtual agent navigation in unknown virtual environments.

2.4 Potential Field Method

We now review some of the potential field models on which the virtual force field concept is hinged. This technique is predicated on the gravitational force field. It is popular because of its simplicity, online adaptive nature and real time prompting. However, it is associated with some basic navigation limitations which results in "cu de sac" also known as the local minima problem.

To be sure, the concept of virtual forces acting on a robot under navigation was introduced long ago by Andrews and Hogan (1983) and later by Khatib (1986). Within this context, and for avoidance of collisions, obstacles are deemed to exert repulsive forces on the robot while the target is presumed to simultaneously apply an attractive force on the robot. In such circumstances the resultant force determines the robot's subsequent direction and speed of motion. Indeed the simplicity and elegance of the technique popularized the use of the potential field method (PFM) in robot navigation over the years.

Subsequent adaptations of the technique have been applied and generalized to cover for example, off line path planning problems by Thorpe (1984) as well as for combined global and local path planning by Krough and Thorpe (1986). Khosla and Volpe (1988) also developed an alternate approach which avoids the local minima found in traditional potential field methods. Notwithstanding its simplicity and elegance, the PFM still suffers from a number of significant problems itemized by Koren and Borenstein (1991). Some of these problems as further illustrated in chapter 3 of this thesis involve the occurrence of trap situations associated with local minima as well as inhibited passage between closely spaced obstacles. Others include pronounced oscillations in the presence of obstacles especially when entrapped in narrow passages in-between obstacles. Whitcomb and Koditschek (1991) have also extended the application of the PFM to automated assembly planning and control.

Montano and Sagues (1991) studied the stability limits and the dynamic behaviour as a function of the control parameters of a robot. While analyzing a more complex setting, Hwang and Ahuja (1992) as well as Dozier et al. (1998) had earlier developed a method that utilises polygons and polyhedra to represent an object from which a potential field is generated. Murray and Little (2000) presented a working implementation of a robot that uses correlation-based stereo vision and occupancy grid mapping to successfully navigate and autonomously explore unknown and dynamic indoor environments. The path planning algorithm is however potential field based, although enhanced with the shortest path algorithm concept. One limitation with their technique is found in their update rule i.e. low quality data can obscure better data when obstacles are viewed at longer range.

Moravec and Elfes (1985) developed the certainty grid concept which represents an obstacle map of the environment using probabilistic representation of obstacle locations in order to overcome the inaccuracies emanating from sensory data. One limitation with this technique however is that it assumes a static environment. Moreover, Borenstein and Koren (1989; 1990) had developed a new concept for robot navigation. The novelty of this approach, entitled the Virtual Force Field, lies in the integration of two known concepts viz: Certainty Grids for obstacle representation and Potential Fields for navigation. This approach permits the simultaneous detection of unknown obstacles and the steering of the mobile robot to avoid collisions while advancing towards the target. This combination is especially suitable for the accommodation of inaccurate sensor data (such as, may be produced by ultrasonic sensors) as well as for sensor fusion, and enables continuous motion of the robot without stopping in front of obstacles. This navigation algorithm also takes into account the dynamic behavior of a fast mobile robot and solves the "local minimum trap" problem.

Furthermore, Borenstein and Koren (1991) developed another PFM based algorithm christened the vector field histogram (VFH) which was also referred to in Ulrich and Borenstein (2000). Other recent attempts to develop the artificial potential field model include the work of Vadakkepat et al. (2000), Park et al. (2001); Savage et al. (2004) all in an effort to resolve the local minima problem. Ogren and Leonard (2002) have recently proposed a modified concept of the dynamic window approach (DWA) which is also a potential field driven technique. Mbede et al. (2000) introduced a planning technique based on artificial potential fields and fuzzy motion for robot manipulator navigation among dynamic obstacles. The focus was on autonomous motion planning of manipulators in known environments with unknown dynamic obstacles. In their case the navigation technique of robot control using artificial potential functions was based on fuzzy logic and the stability was guaranteed by Lyapunov theory. In this application, the fuzzy system presented was used to approximate the gradient of the harmonic functions which handled both the mapping between the input and the output space, and the navigation problem.

The research work conducted by Ge and Cui (2000a) describes the problem of goals nonreachable with obstacles nearby when using potential field methods for mobile robot path planning. They presented new repulsive potential functions by taking the relative distance between the robot and the goal into consideration, which ensures that the goal position is the global minimum of the total potential. Simulations were conducted on convex obstacles and the results verified that the new repulsive potential function developed can solve the problem of goals non reachable with obstacles nearby (GNRON) effectively.

A new potential field method for motion planning of mobile robots in a dynamic environment where the target and obstacles are moving was proposed by Ge and Cui (2000b). Firstly, the new potential function and the corresponding virtual force are defined. Then, an on-line motion planning algorithm based on the new potential field method is presented. Finally, computer simulation was used to demonstrate the effectiveness of the dynamic motion planning scheme based on the new potential field method.

The attractive potential is defined as a function of the relative position and velocity of the target with respect to the robot. The repulsive potential is also defined as the relative position and velocity of the robot with respect to the obstacles. Accordingly, the virtual force is defined as the negative gradient of the potential with respect to both position and velocity rather than position only. The new definitions of the potential functions and the virtual forces allow the robot to track the target in a desired manner.

The motion planning problem for a mobile robot in a dynamic environment is to plan and control the robot motion from an initial position to track a moving target in a desired manner while avoiding moving obstacles. To simplify the analysis, we have the following assumptions:

Assumption 1: The robot is a point mass which can move omni-directionally, whose mass, position and velocity are known and its maximum velocity is greater than that of the target. The acceleration of the robot is omnidirectional.

Assumption 2: The point target moves at constant velocity and its position and velocity are known.

Assumption 3: The obstacles are assumed to be balls of radius centered at p_{obsi}, with i =1; 2; : : :; n_{obs}, where n_{obs} is the number of obstacles. The positions p_{obsi} and velocities v_{obsi} of the obstacles can be measured accurately.

Assumption 4: At each time instant, only one obstacle is close enough to the robot and needs to be avoided. The rest are assumed to be farther away and their influences are neglected for that instant.

Furthermore, Wei et al. (2001) worked on intelligent motion planning. Their solution method was based on fuzzy rules for the idea of artificial potential fields using analytic harmonic functions. The purpose of combining these controller types was to design a realistic controller for nonlinear electromechanical systems such as an electric motor actuating a robot arm. The control algorithm was applied to the three basic navigation problems of intelligent robot systems in unstructured environments viz: autonomous planning, fast non-stop navigation without collision with obstacles and dealing with structured and/or unstructured uncertainties. Also, Bruce and Veloso (2003) used a vision based pattern detector to solve the robot navigation problem while Castro et al. (2003) used the reactive local navigation technique. Wolf et al. (2004) discussed an improved Potential field method which tries to resolve the problems identified in the traditional PFM particularly with regard to the oscillation problems. In addition, Iagnemma and Dubowsky (2004) worked on Traction Control of Wheeled Robotic Vehicles in Rough Terrain with Application to Planetary Rovers.

In his research thesis, Beckhaus (2002) developed a method to provide guided exploration in virtual environments. The adopted technique is hinged on the concept of dynamic potential fields, a local method derived from the physics of the motion of a charged particle however, in this research work, a camera serves as the charged particle (in an electric potential field). It uses a discretized representation of the environment in a uniform rectangular grid.

They also presented the CubicalPath system which is able to deal with interactive, real-time input by the client application and the user, which can consist of dynamic, unpredictable object locations, dynamic or re-adjusted targets, a modified application view and a force input generated by an interaction tool. As part of the future works recommended in his thesis, emphasis was placed on the issue of unwanted local minima, the main drawback of the potential field approach and also a drawback for this research work. Some recommendations as stated in the thesis towards tackling this problem are namely: (i) automatically filling concave objects (ii) introducing global knowledge by navigation objects, using Brownian motion (iii) using a search technique to move out of the minimum or (iv) modifying the field function in a way such that it has minima positions only where a target is defined.

Furthermore, the developed technique, works best in sparse environments with convex objects. As a result of unwanted local minima, it will fail in complex environments like mazes. In addition, the parameters controlling the motion behavior of the system are manually modified when the need arises to meet certain desired target. This limitation calls for the development of an automatic tuning system which analyzes the geometric environment and adjusts the field functions and control parameters to match a specified behavior description.

In this study, Park (2003) combined the artificial potential field technique (APF) and the virtual obstacle concept. The virtual obstacle helps to bring the robot out of local minima trap during navigation. The virtual obstacle comes in as an extra potential which is added to the global potential and is located at the point of local minima trap. A sensor based discrete modeling method is also proposed for modeling of the mobile robot with range sensors. This modeling method is adaptable for a real-time path planning because it provides lower complexity. Some limitations associated with this technique include:

i. The extra potential which also represents the virtual obstacle is said to repel the robot in the opposite direction to the trap point. The extent of repulsion was not clearly stated. It was also stated that the extra potential reduces with distance from the trap meaning that the robot may still return to the trap once the extra potential is exhausted following that the attractive potential of the target is still active and the target position has not changed.

ii. The need for a relatively large number of ultrasonic sensors, viz: about sixteen in number mounted in an arc-like form at the front side and back of the proposed robot design does not indicate economic and optimal use of hardware. It is however not clear whether the effectiveness of this proposed virtual obstacle technique is a critical function of either the number of sensors or even the relative distance of one sensor from another.

iii. Validation was done only by simulation. The degree of authenticity and effectiveness of this scheme would have been demonstrated better on a real robot.

Baştan (2004) worked on concave shaped obstacles and dynamic obstacles in general. The results obtained from his experiments showed that some modifications and improvements have to be made to get satisfactory results from the implemented algorithm, which was claimed to work well in simulations. He further observed that it is usually impossible to have a complete model of the real world in simulations, or the model adopted in simulations may be too simple to faithfully represent the real system. Therefore, some modifications and improvements are suggested with experimental results analyzing their successful and problematic aspects. Shimoda et al. (2005) however looked at high Speed Unmanned Ground Vehicles on Uneven Terrain.

Nguyen et al. (2005) considered a motion planning method based on cooperative biological swarming models with virtual attractive and repulsive potentials (VARP). The motion planning map results led to the development and implementation of a point to point controller which is subsequently used as part of a cooperative searching algorithm. The VARP control method is scalable and can be used to organize a swarm of robotic vehicles.

Ögren and Leonard (2005) proposed a new navigation scheme based on the combination of a model-based optimization scheme and a convergence-oriented potential field method.

Inspired by the works of Primbs et al. (1999), they presented a way to merge the convergent Koditschek scheme with the fast reactive dynamic window approach (DWA). This was done by casting the two approaches in a model predictive control (MPC) and control Lyapunov function (CLF) framework, respectively and combining the two as suggested by Primbs et al. This combined technique significantly aided the convergence process.

In their paper, Nourani-Vatani et al. (2006) presented a robotic platform for autonomous coverage tasks. The system architecture adopted for their case integrates laser based localization and mapping using the Atlas Framework with Rapidly-Exploring RandomTrees (RRT) path planning and Virtual Force Field obstacle avoidance. Their model was validated by simulation as well as with real world experiments. Some of the limitations of their work include: (i) as a result of the randomness of the algorithm, the trajectories created by planner do not always follow the desired way-points, which influences the efficiency of the coverage. (ii) the number of iterations RRT has to grow its trees to ensure real-time performance was limited. As a consequence, RRT occasionally does not complete a path connecting the way-points. Future work as stated in their paper comprises improving the RRT based local path planner to follow the desired paths more closely and to extend the global path planner to generate paths that can completely cover a given area while taking into account all of the vehicle's kinematic constraints.

Heinemann et al. (2006) presented a path planning algorithm based on time variant potential fields that efficiently plan paths around moving obstacles. Their research was adapted to robot soccer game where planning collision-free paths is one of the basic skills for a mobile robot performing a goal-oriented task in highly dynamic environment. In this problem, there is need for smooth navigation avoiding both the cooperating and competing players. Several approaches for tracking objects have been implemented and tested over the past few decades however, they have also developed robot control algorithms that incorporate the velocity of the objects for ball interception as goal keeper or pass receiver and for path planning. One of the main challenges for their high-level software was how to accurately model the robot's environment including the velocity of the objects resulting from a robust tracking over time.

Their result showed that the iterative path planning process needs very little iteration to converge to a reasonable path that avoids the trajectories of moving obstacles.

Chengqing et al. (2000) earlier pioneered the virtual obstacle concept as a strategy for local-minimum-recovery in potential-field based navigation. Im and Oh (2000) used an approach christened Extended Virtual Force Field (EVFF) by integrating neural network theory and evolutionary programming. Zou and Zhu (2003) were the first set of investigators to work with the virtual local target approach for solving the local minima problem associated with the potential field method. Hussein and Hamid (2006) recently employed the Artificial potential approach combined with Maxwell's equations and christened FDTD Method. The objective of their research is Autonomous Motion Planning in Global Dynamic Environment. They stated that the proposed model is extendable to 3D.

In this chapter we have reviewed the literature in robot path planning. Also, the different pathplanning techniques have been grouped into different sub-classes ranging from the graphical approach through the classical methods to the heuristic approach and the potential field methods.

CHAPTER THREE

PATH PLANNING MODEL FOR AN AUTONOMOUS MOBILE PLATFORM IN A COMPLEX STATIC OBSTACLE DOMAIN.

3.1 PREAMBLE

This chapter describes an obstacle avoidance method designed to plan paths for an autonomous mobile robot in a partially known 2-D workspace. The workspace consists of static obstacles and a robot desired goal state. The Hybrid Virtual Force Field (HVFF), which is an integration of the Virtual Obstacle Concept (VOC) and the Virtual Goal Concept (VGC) in combination with the traditional Virtual Force Field concept is proposed for an efficient robot path planning. The specific problems addressed in this chapter include the local minima problem posed by either lengthy or concave shaped obstacles as well as the potential field induced oscillatory motion of such a robot when maneuvering in the corridor between two narrowly spaced obstacles.

3.2 The Virtual Force Field (VFF) Algorithm

Assuming q_r and q_{target} respectively describes the configuration space of a robot and its target point in a 2D domain such that $q_r = f(x_r, y_r)$ with $x_r \equiv i \; and \; y_r \equiv j$

it then follows that $q_r = q_r(i) + q_r(j)$ (1)

is the directional representation of q_r. If the vector operator ∇ defined by $\nabla = i\frac{\partial}{\partial x} + j\frac{\partial}{\partial y}$ is considered, then $\nabla(q_r) = \left(i\frac{\partial q_r}{\partial x} + j\frac{\partial q_r}{\partial y} \right)$ (2)

If at every point in time the moving robot R is subject to both attractive and repulsive potentials resulting from the target position and stationary obstacles respectively then $U_{att}(q_r)$ and $U_{rep}(q_r)$ would represent the resultant potential acting on the robot where, $U_{att}(q_r)$ is the sum of the total attractive potential from the target point at a given instant while $U_{rep}(q_r)$ the total repulsive potential from the obstacle(s) in the work space at the same instant such that instantaneous motion of the robot is governed by these resultant potentials. The gradient of U at q_r is given as:

$$\nabla U(q_r) = \begin{bmatrix} \dfrac{\partial U}{\partial x} \\ \\ \dfrac{\partial U}{\partial y} \end{bmatrix} \tag{3}$$

$\nabla U(q_r)$ is a vector that points in the direction of the fastest change of U at configuration q_r.

The magnitude of the rate of change is expressed in terms of

$$|\nabla U(q_r)| = \sqrt{\left(\frac{\partial U}{\partial x}\right)^2 + \left(\frac{\partial U}{\partial y}\right)^2} \tag{4}$$

Moreover each of the potentials can be further decomposed in terms of attraction and repulsion on both x and y axes. As such we have:

$$U_{att}(q_r) = U_{att}^{x}(q_r) + U_{att}^{y}(q_r) \tag{5}$$

while

$$U_{rep}(q_r) = U_{rep}^{x}(q_r) + U_{rep}^{y}(q_r) \tag{6}$$

Assuming that $d_{t\arg et} = \sqrt{((x_{t\arg et} - x_r)^2 + (y_{t\arg et} - y_r)^2)}$ (7)

where,

$d_{t\arg et}$ = Euclidean distance between the robot and the target point where,

x_r = robot x-coordinate and

y_r = robot y-coordinate

It follows that $d_{target} = \left\| q_{target} - q_r \right\|$ (8)

Assuming displacement is considered along the x- axis then

$$d_{target} = \left\| q_{target} - q_r \right\| = \left(\sum_{r=1}^{n} (x_{target} - x_r)^2 \right)^{1/2} \quad (9)$$

$$\nabla d_{target}(q_r(x_r)) = \nabla \left(\sum_{r=1}^{n} (x_{target} - x_r)^2 \right)^{1/2} \quad (10)$$

$$= 1/2 \left(\sum_{r=1}^{n} (x_{target} - x_r)^2 \right)^{-1/2} \nabla \left(\sum_{r=1}^{n} (x_{target} - x_r)^2 \right) \quad (11)$$

$$= 1/2 \left(\sum_{r=1}^{n} (x_{target} - x_r)^2 \right)^{-1/2} \left(2\left(x_{target} - x_1 \right) \ldots 2\left(x_{target} - x_n \right) \right) \quad (12)$$

$$= \frac{\left(x_{target} \right) - \left(x_1, \ldots\ldots\ldots x_n \right)}{\left(\sum_{r=1}^{n} \left(x_{target} - x_r \right)^2 \right)^{1/2}} = \frac{\left(x_{target} - x_r \right)}{\left(\sum_{r=1}^{n} \left(x_{target} - x_r \right)^2 \right)^{1/2}} \quad (13)$$

Assuming the robot's displacement is considered along the y- axis then

$$d_{target} = \left\| q_{target} - q_r \right\| = \left(\sum_{r=1}^{n} (y_{target} - y_r)^2 \right)^{1/2} \quad (14)$$

$$\nabla d_{target}(q_r(y_r)) = \nabla \left(\sum_{r=1}^{n} (y_{target} - y_r)^2 \right)^{1/2} \quad (15)$$

$$= 1/2 \left(\sum_{r=1}^{n} (y_{target} - y_r)^2 \right)^{-1/2} \nabla \left(\sum_{r=1}^{n} (y_{target} - y_r)^2 \right) \quad (16)$$

$$= 1/2\left(\sum_{r=1}^{n} (y_{target} - y_r)^2 \right)^{-1/2} \left(2\left(y_{target} - y_1 \right) \ldots 2\left(y_{target} - y_n \right) \right) \tag{17}$$

$$= \frac{\left(y_{target} \right) - \left(y_1, \ldots\ldots\ldots y_n \right)}{\left(\sum_{r=1}^{n} \left(y_{target} - y_r \right)^2 \right)^{1/2}} = \frac{\left(y_{target} - y_r \right)}{\left(\sum_{r=1}^{n} \left(y_{target} - y_r \right)^2 \right)^{1/2}} \tag{18}$$

From (12) and (17) above, it could be deduced that

$$\nabla d_{target}(q_r(x_r, y_r)) = \frac{\left(q_{target} - q_r \right)}{\left(\sum_{r=1}^{n} \left(x_{target} - x_r \right)^2 \right)^{1/2} + \left(\sum_{r=1}^{n} \left(y_{target} - y_r \right)^2 \right)^{1/2}} \tag{19}$$

$$= \frac{q_{target} - q_r}{\left\| q_{target} - q_r \right\|} \tag{20}$$

Hence for our problem,

$$U_{att}^{x}(q_r) = \frac{(x_{target} - x_r)}{d_{target}} \tag{21}$$

and

$$U_{att}^{y}(q_r) = \frac{(y_{target} - y_r)}{d_{target}} \tag{22}$$

Within the same context, we can, by following the work of Koren and Borenstein represent the repulsive potentials as:

$$_{current}U_{rep}^{x}(q_r) = {}_{initial}U_{rep}^{x}(q_r) + \frac{Fcr}{3} * \frac{c_{i,j}}{d_{obstacle}^{2}} * \frac{(x\_grid - x_{obstacle})}{d_{obstacle}} \tag{23}$$

$$_{current}U^{y}_{rep}(q_r) = {}_{initial}U^{y}_{rep}(q_r) + \frac{Fcr}{3} * \frac{c_{i,j}}{d^2_{obstacle}} * \frac{(y\_grid - y_{obstacle})}{d_{obstacle}} \tag{24}$$

where,

F_{cr} is the repulsive constant defined as shown below:

$$F_{cr} = \begin{cases} 0, & \text{when } c_{i,j} = 0 \\ 1 & \text{when } c_{i,j} <> 0 \end{cases}$$

Also,

$_{initial}U^{x}_{rep}(q_r)$ and $_{initial}U^{y}_{rep}(q_r)$ both have initial boundary values of zero.

The initial boundary values of zero assigned to $_{initial}U^{x}_{rep}(q_r)$ and $_{initial}U^{y}_{rep}(q_r)$ are in line with the assumption that the robot ideally takes off from a state of rest where it is absolutely ignorant of its environment,

As the vehicle moves, a dynamic window $c_{i,j}$ overlying a region of the workspace accompanies it. We call this region the active region and cells that momentarily belong to the active region are called active cells. In our current implementation, the dimension of the window is square shaped and the window is always centered about the robot position.

Also, each cell in the active window of the robot has an attribute called the certainty value (CV). This parameter is a measure of confidence by the robot as to whether or not an obstacle exists in a given cell. The concept was originally developed by Moravec and Elfes, (1985) and the CV of a cell increases once the range reading from a sensor indicates the presence of an obstacle in that cell.

In particular,

$$c_{i,j} = \begin{cases} 0 & \text{if no obstacle is present } in \ any \text{ cell } in \ the \ robots \ active \ window \\ 1 & \text{if obstacle is present } in \ any \text{ cell } in \ the \ robots \ active \ window \end{cases}$$

where, i, j represents the grid lines of the cell that forms the active window

$if \ c_{i,j} <> 0 \ and \ d_{target} > 0 \ then$

$$d_{obstacle} = \sqrt{((x_r - x_{obstacle})^2 + (y_r - y_{obstacle})^2)} \tag{25}$$

where,

$d_{obstacle}$ measures the distance between the closest obstacle and the robot.

Here, $x_{obstacle}$ and $y_{obstacle}$ are the coordinates of the obstacle and thereby as the robot navigates towards the target point, it is updated by the control unit based on whether or not an obstacle is present in any cell within its active window.

Each active cell exerts a virtual repulsive force towards the robot given by the components of their potentials as spelt out in equations (23) and (24). The variable force of repulsion Fcr, contained in the afore-mentioned relations, is in our case given by the expression

$$Fcr = \frac{3 * rel\_angle}{d_{obstacle}} \tag{26}$$

Fcr is usually activated when an obstacle is present in the robot's active window. . In the absence of an obstacle, Fcr gives an output of zero. Thus in general

$$U^x(q_r) = U^x_{att}(q_r) + U^x_{rep}(q_r) \tag{27}$$

and

$$U^y(q_r) = U^y_{att}(q_r) + U^y_{rep}(q_r) \tag{28}$$

It should be noted that the variables U^x_{att} $(q_r)$and U^x_{rep} (q_r) of (27) were earlier obtained from (21) and (23) respectively while variables U^y_{att} $(q_r)$and U^y_{rep} $(q_r)$of (28) were both obtained from (22) and (24) respectively. Equations (27) and (28) denote the resultant potential in the x and y axes respectively. The actual navigation path of the robot is determined by these two variables as they jointly determine the robot's new position as it navigates from one state space towards the target point. The robot's orientation at every new configuration is usually in the direction of the resultant which tends towards the target point.

The virtual force field and the potential field methods are fundamentally hinged on the concept of repulsion of one body from another and attraction of the same body to another through the use of algorithms capable of generating virtual or artificial attractive potentials and artificial repulsive potentials. Our proposed approach christened Hybrid virtual force field (HVFF) concept is basically aimed at improving the performance of the VFF technique through the integration of some methodologies hinged on three concepts namely; geometry, vectors and production rules:

3.2.1 Geometry: This is a branch of mathematics that is concerned with the properties of configurations of objects - points, (straight) lines and circles being the most basic of these. It is the study of shapes and configurations and basically uses the physical attributes of the workspace objects to represent, describe and establish interrelationship among them. Such objects include the robot and the obstacle(s). Geometry attempts to understand and classify spaces in various mathematical contexts.

3.2.2 Vectors: This involves the use of geometrical primitives such as points, lines, curves and shapes which are all based upon certain mathematical equations. Usually, a study of motion will involve the introduction of a variety of quantities which are used to describe the physical world. Such quantities include: distance, displacement, speed, velocity, acceleration, force, mass, momentum, energy, work, power and as further depicted in Fig. 2a. Vectors are usually described by both magnitude and direction. Vector quantities are often represented by scaled vector diagrams as in Figs. 2b and 2c. Vector diagrams depict a vector by use of an arrow drawn to scale in a specific direction. In the context of this work, we apply vectors in order to understand motion and direction which occurs in two dimensions. Also, vectors have been used to represent the positions of obstacles, the desired goal state and changing positions of the robot as it navigates towards the goal state.

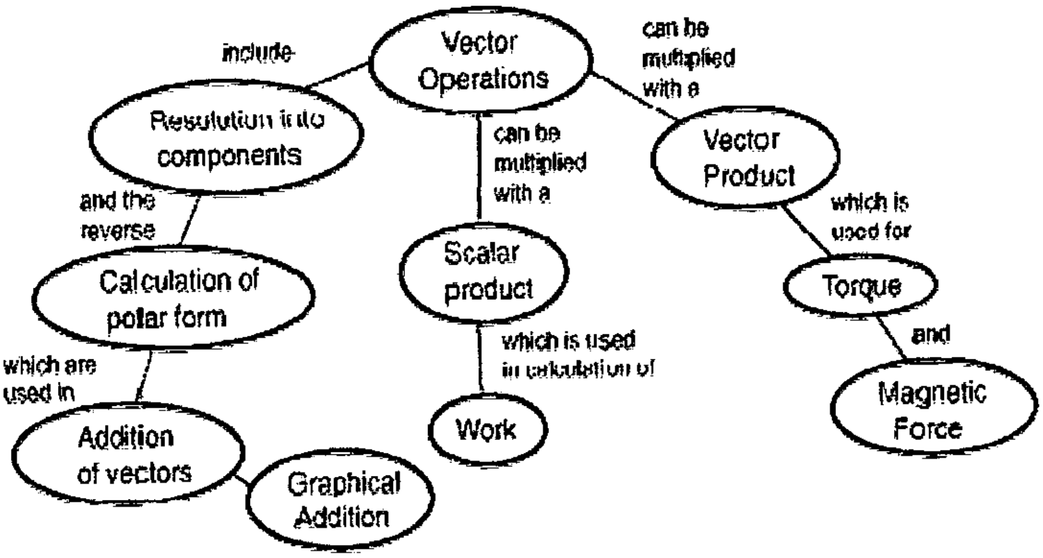


Figure 2a: Basic Vector Operations

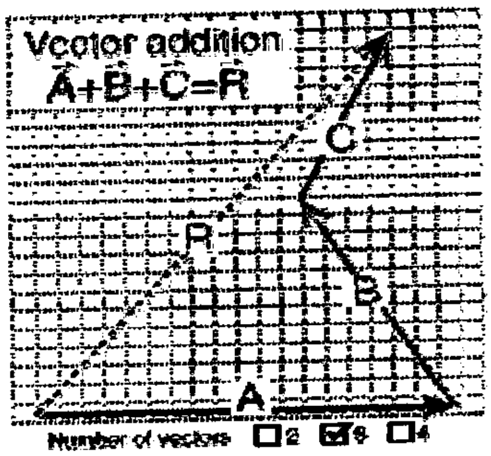


Figure 2b: Vector Addition, Three Vectors

Source: R. Nave

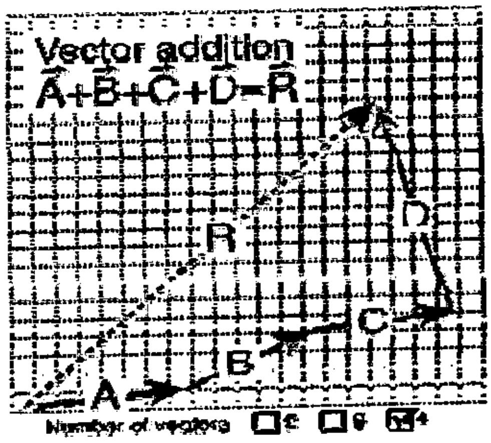


Figure 2c: Vector Addition, Four Vectors

3.2.3 Production Rules: These are also referred to as the if-then rules or condition-action rules. Rule-based systems consist of a set of rules, a working memory and an inference engine. The rules encode domain knowledge as simple condition-action pairs. The working memory initially represents the input to the system, but the actions that occur when rules are fired can cause the state of working memory to change. The inference engine must have a conflict resolution strategy to handle cases where more than one rule is eligible to fire. The **match-resolve-act cycle** is what the inference engine does. The components of a rule-based system have the form:

`if <condition> then <conclusion>` or `if <condition> then <action>`

Rules can be evaluated by using either the forward chaining or backward chaining techniqupe as shown in Figs. 2d and 2e respectively.

Forward Chaining

- Given some facts, work forward through inference net.
- Discovers what conclusions can be derived from data.

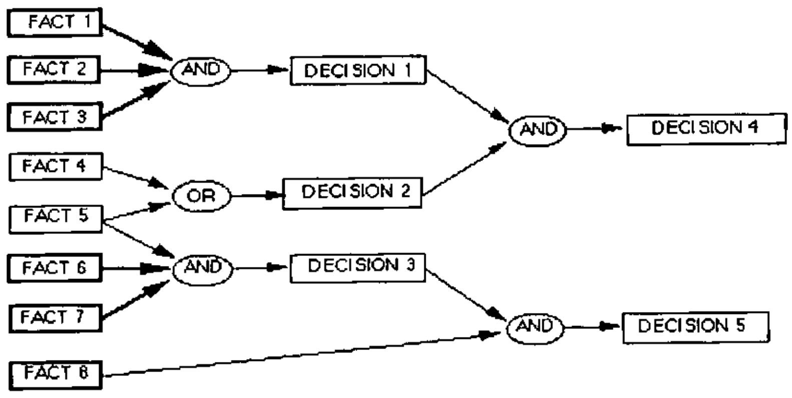


Figure 2d: Forward Chaining in Production Rules

Backward Chaining

- To determine if a decision should be made, work backwards looking for justifications for the decision.
- Eventually, a decision must be justified by facts.

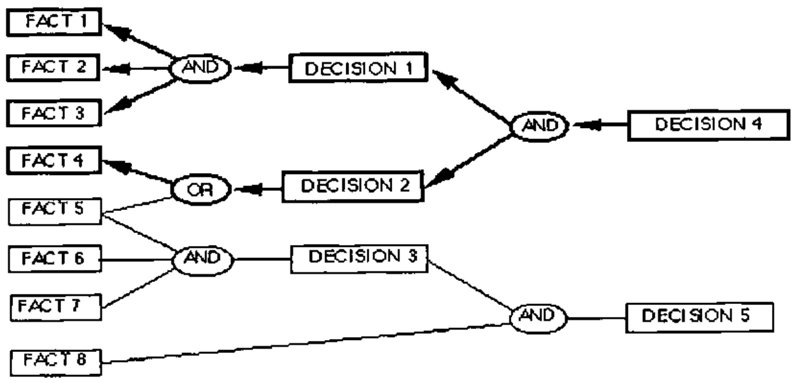


Figure 2e: Backward Chaining in Production Rules

The primary objective of our algorithm is to address the local minima problem associated with the potential field method. Our algorithm uses the concept of "line of sight" derived from the concept of ray theory. Prior to navigation, the robot has knowledge of areas that could result in the local minima problem. Once the robot is prompted for motion, it first checks whether its line of sight (which is usually a straight line to the goal state) is obstructed by any obstacle in the workspace. If the line of sight is obstructed then the next issue to be investigated is the nature of obstruction whether it falls in the category of those that could result in the local minima problem. If yes then the HVFF concept which involves the use of VOC and VGC is deployed. This is in contrast with the traditional VFF concept, where the robot ordinarily would continue its navigation even along a path of local minima until it gets trapped and thereafter begins to oscillate.

The VOC in our algorithm impedes the robot from motion. This prevents the robot from moving in the direction of local minima and spontaneously triggers the VGC. Unlike the VOC which is a preventive algorithm, the VGC is a directive algorithm. It helps redirect the robot's line of sight away from the local minima region hence resulting in navigation along a new line of sight defined by the virtual goal. Further details on how our algorithm works is given in sections 3.4.1 and 3.4.2 where the VOC and VGC are respectively described in detail.

3.3 Algorithmic Procedure

We next itemize how the traditional VFF concept is embedded within our proposed hybrid concept by giving an outline of the sequential order of the overall algorithm as illustrated in Fig. 2f below:

Step I: This module allows us to create a 2D workspace of dimension X by Y and select a fixed reference point for the workspace. The emphasis in this step is the availability of a workspace or navigation environment. Since the current problem is 2D, we have the x and y axes for the purpose of position tracking. A navigation environment may be regular or irregularly shaped; what is important is the availability and delineation of the boundaries of the workspace. This task is next followed by the identification of a point within the workspace which serves as a reference point for the robot navigation. The reference point helps the robot to know its own relative position, as well as those of the goal state or target point and the workspace obstacles.

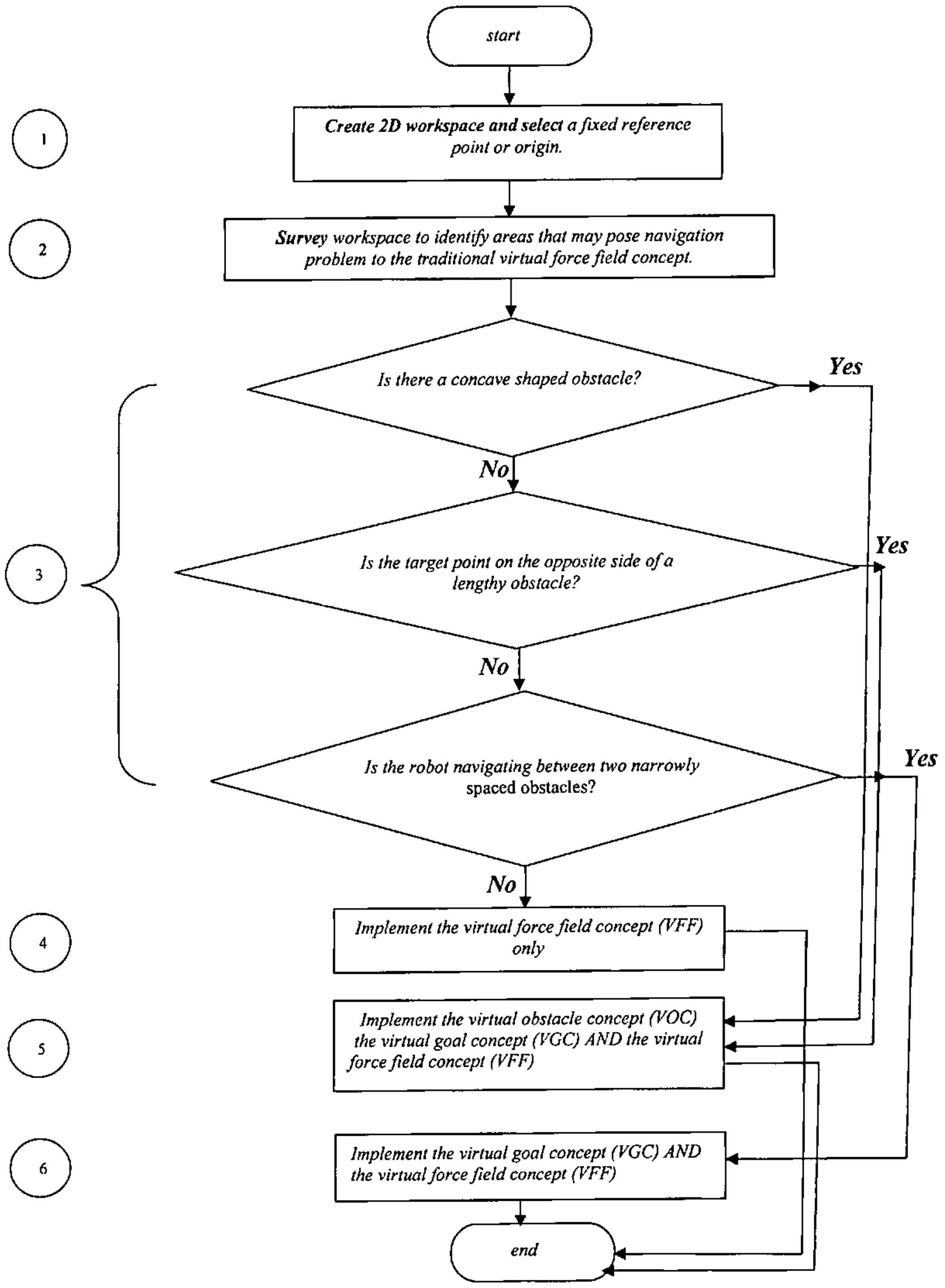
Step II: This module carries out a survey of workspace for identification of areas that may pose navigation problems to the traditional Virtual Force Field concept. Such areas are characterized by navigation impediments leading to local minima problems etc as earlier discussed in section 1.2. Once the areas prone to local minima occurrence are identified, it is complemented by a mapping exercise which aids in the identification of their relative position vectors from a specified reference point. Such information is encoded in the robot's knowledge base prior to commencement of the navigation exercise. The relevance of this step is hinged on the fact that our proposed algorithm is for a partially known environment where some a priori knowledge of the navigation environment is needed.

Step III: This module basically serves as a sequel to step II above in that the outcome of the survey exercise in step II determines the activation or deactivation of certain algorithms in the control sub-unit. If the environment is such that a local minima trap is not likely to occur then we proceed to **step IV**. However, if navigation traps are suspected then we skip step IV and proceed to either **step V or step VI.** Hence, the essential purpose of this module is to link step II with either step IV or step V as the case may be.

Step IV: This module, when triggered implements the traditional Virtual Force Field (VFF) algorithm. If the outcome of the workspace survey in step II is such that local minima trap is not likely to occur, then the flow is from **step III** to **step IV**. At this point the environment of navigation is referred to as a completely unknown environment since the robot's navigation control is completely reactive and no a priori knowledge about the workspace obstacles is needed. If the outcome of the workspace survey is such that local minima trap is likely to occur then the algorithmic flow is from **step III** to either **step V or step VI depending on the causative factor(s) of the local minima problem as earlier stated**. Either of these steps is prompted by local minima problems.

Step V. This module is prompted when local minima problem results from either concave shaped obstacles or from a relatively lengthy obstacle as modeled in Fig. 3e below. Here, the virtual obstacle concept (VOC), the virtual goal concept (VGC) and the virtual force field (VFF) concepts are co-implemented. At this point the environment of navigation is classified as a partially known environment since the robot's navigation to the target point is dependent on a priori knowledge of some obstacles in the workspace. The implementation procedures for the VOC and VGC are described in sections 3.4.1 and 3.4.2 respectively.

Step VI: This module is prompted when the local minima problem occurs as a result of two closely spaced obstacles which requires the robot to go in between them to arrive at the goal state. Here, the virtual goal concept (VGC) as well as the virtual force field (VFF) concept are co-implemented. For this scenario, the environment of navigation is also classified as a partially known environment since the robot's navigation to the target point is dependent on the a priori knowledge of some obstacles in the workspace.



start
1
Create 2D workspace and select a fixed reference point or origin.
2
Survey workspace to identify areas that may pose navigation problem to the traditional virtual force field concept.
3
Is there a concave shaped obstacle?
Yes
No
Is the target point on the opposite side of a lengthy obstacle?
Yes
No
Is the robot navigating between two narrowly spaced obstacles?
Yes
No
4
Implement the virtual force field concept (VFF) only
5
Implement the virtual obstacle concept (VOC) the virtual goal concept (VGC) AND the virtual force field concept (VFF)
6
Implement the virtual goal concept (VGC) AND the virtual force field concept (VFF)
end

Figure 2f: A flow chart diagram for the HVFF algorithm

3.4 Virtual Obstacle and Virtual Goal Concepts.

By way of implementation of our new algorithm, we propose a combination of two emerging concepts for solving the inherent problems associated with the traditional VFF technique. Although it has earlier been established in the literature that either of these concepts could be used independently, the dual use of both concepts as an integral part of the solution process is infact novel. To be sure, it was Chengqing et. al (2000) who first integrated the virtual obstacle technique with the potential field method to maneuver cylindrical mobile robots in unknown environments and christened the method as the virtual obstacle method. In a similar fashion, Zou and Zhu (2003) later introduced the concept of integrating a virtual goal with the potential field principle and they called this approach the Virtual Local Target (VLT) method.

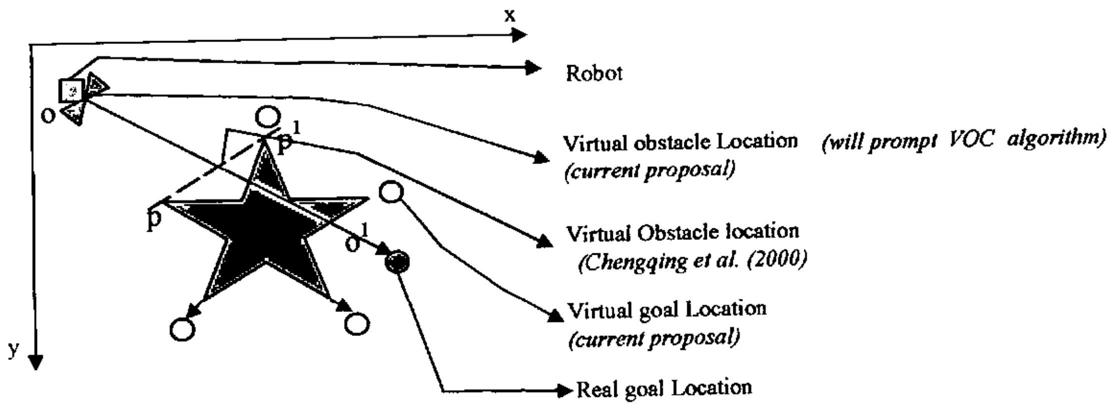

x
y
o
o^1
p
p^1
Robot
Virtual obstacle Location (will prompt VOC algorithm) (current proposal)
Virtual Obstacle location (Chengqing et al. (2000)
Virtual goal Location (current proposal)
Real goal Location

Figure 3a: Case I sample workspace demonstrating the virtual obstacle concept and the virtual goal concept

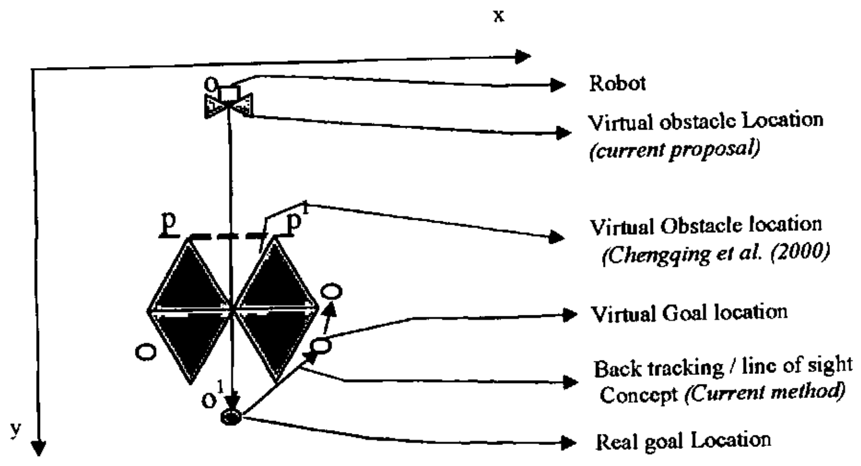

x
y
o
o^1
p
p^1
Robot
Virtual obstacle Location (current proposal)
Virtual Obstacle location (Chengqing et al. (2000)
Virtual Goal location
Back tracking / line of sight Concept (Current method)
Real goal Location

Figure 3b: Case II sample workspace demonstrating the virtual obstacle concept and the virtual goal concept

For example Fig. 3a illustrates the use of VOC and VGC concepts in an environment containing a star shaped obstacle while in Fig. 3b the goal is located diametrically opposite the robot but with the line of sight grazing two touching obstacles leaving no clearance through which the robot could maneuver.

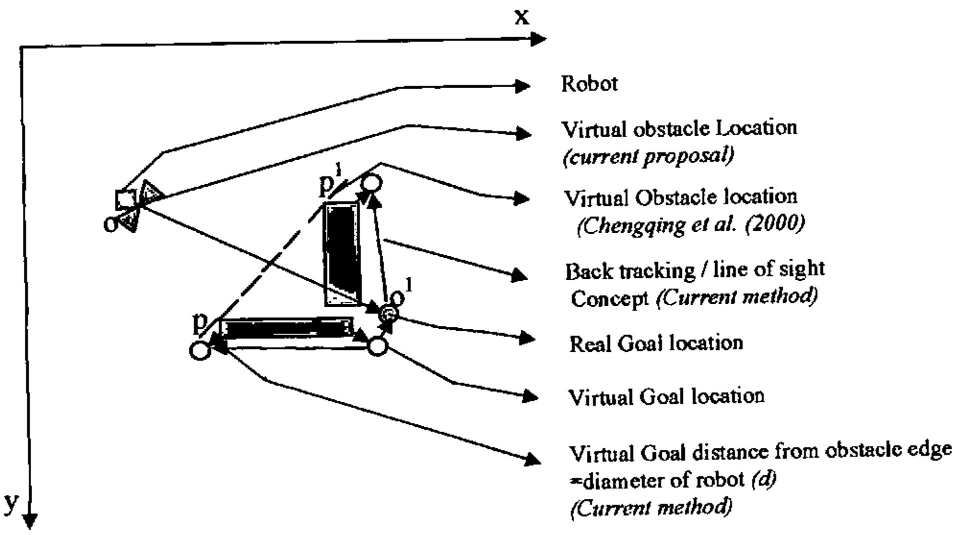

x
y
Robot
Virtual obstacle Location (current proposal)
Virtual Obstacle location (Chengqing et al. (2000)
Back tracking / line of sight Concept (Current method)
Real Goal location
Virtual Goal location
Virtual Goal distance from obstacle edge =diameter of robot (d) (Current method)
p^1
o^1
p
o

Figure 3c: Case III sample workspace demonstrating the virtual obstacle concept and the virtual goal concept

Fig. 3c on the other hand treats the case where two obstacles create a concave architecture concealing the goal state from the robot whereas in Fig. 3d there is sufficient clearance between two stationary obstacles separating the robot from the goal state. Lastly, the case of a thin and relatively lengthy obstacle shielding the goal state from the robot is illustrated in Fig. 3e.

For a start we note that in Figs. 3a to 3c the HVFF scheme locates a virtual obstacle right next to the robot. This is merely to trigger the VOC scheme as described in section 3.4.1 which allows the robot to respond faster than in the algorithm of Chenquing et al. (2000) who placed their virtual obstacle farther down the line. At the same time the complementary effect of using back tracking along the line of sight from a hidden goal state around the perimeter of the obstacle facilitates optimal location of virtual goals.

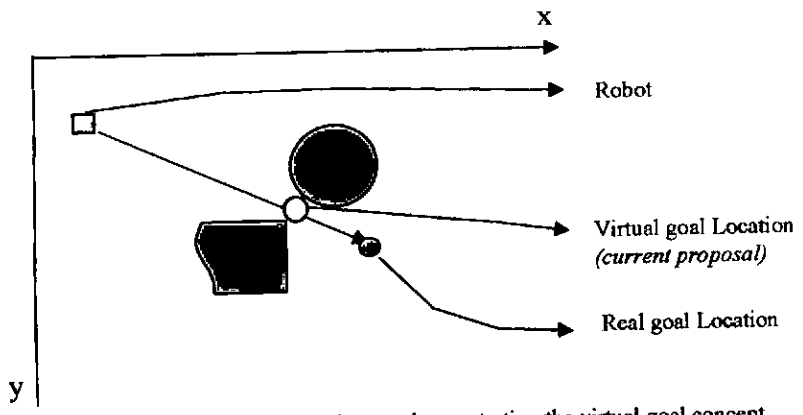


Figure 3d: Case IV sample workspace demonstrating the virtual goal concept

Fig. 3e is a schematic model of a lengthy obstacle with a display of the robot and its target point at opposite sides of the obstacle.

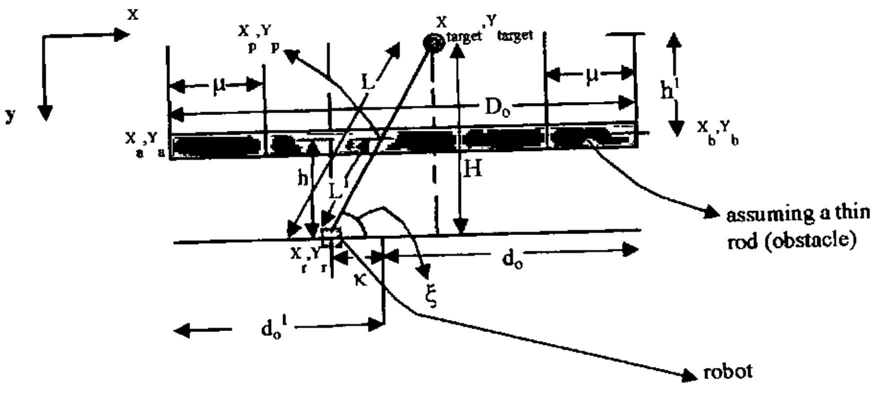


Figure 3e: Model of an assumed thin and lengthy obstacle.

Different cases that could emerge as a result of the relative position of the robot and the target at either side of the obstacle is as represented below. The procedure to adopt is a function of the robot-obstacle relative position as well as the interplay of the variables $L^1, \xi, \kappa, X_p, Y_p, d_o, d_o^1$ *and* D_o as analyzed below.

$$\kappa = L^1 \cos\xi \tag{29}$$

$$L = \|(X_r - X_{target}) + (Y_r - Y_{target})\| \tag{30}$$

$$H = L\sin\xi \tag{31}$$

$$D_o = \|(X_a - X_b) + (Y_a - Y_b)\| \tag{32}$$

$$d_o = \|((X_r + \kappa) - X_b) \;\; + ((Y_r - h) - Y_b) \;\; \| \tag{33}$$

$$d_o^1 = \|((X_r + \kappa) - X_a) \;\; + ((Y_r - h) - Y_a) \;\; \| \tag{34}$$

$$d_o = D_o - d_o^1 \tag{35}$$

Assuming that $\mu = f\{dia_{robot}, D_o, (x_{target}, y_{target}), (x_{robot}, y_{robot})\}$ where μ is the maximum distance of the robot from either side of the obstacle's vertices for which the resultant potential acting on the robot would be capable of overcoming the local minima trap, either of the following cases become valid:

Case I: if $d_o <= \mu$ and $d'_o > \mu$

Case II: if $d_o > \mu$ and $d'_o <= \mu$

In cases I and II above the robot navigates to the goal state along the obstacle's edge whose distance from the goal is shorter

Case III: if $d_o <= \mu$ and $d'_o <= \mu$

In case III above the robot navigates to the goal state in the direction of the steepest descent along the goal position.

Case IV: if $d_o > \mu$ and $d'_o > \mu$ and $abs(\theta - \delta) >= \psi_1$ *and* $abs(\theta - \delta) <= \psi_2$ (i.e. robot has orientated by close to rev/2) then VOC is automatically prompted followed by VGC.

3.4.1 Virtual Obstacle Concept (VOC):

The virtual obstacle concept in our context is basically a proactive algorithm that ensures that the robot is not attracted into a corner region of obstacles that has no outlet. It is hinged on the concept of completely blocking off such passageways that could lead the robot to a local minimum trap. The approach we are proposing for the representation of our VOC is the concept of intersecting vertices. This involves the introduction of a new line that closes the edges of the obstacles that frame the local minimum trap. The intersection of the line of sight of the target from the robot with this new line represents the farthest location the virtual obstacle can be placed from the robot; otherwise it is located right next to the robot along the line of sight of the target from the robot. The VOC representation for some sample configurations is as shown in Figs. 3a to 3c.

The location of the target point relative to the robot's initial position at the start of navigation determines whether or not the virtual obstacle algorithm should be implemented or not. Our concept of virtual obstacle is generalized in principle, and is in no way limited by the shape of the obstacles that circumscribe the concave environment. We have tried to illustrate its level of completeness and generalization by creating different scenarios of concave shaped problems using different types of obstacle shapes. In comparison with the earlier work of Chengqing et al (2000) our VOC algorithm is more efficient.

The reader is also invited to note that in the case of Chengqing et al (2000), entrapment in local minima was avoided by replacing the two converging obstacles with a virtual third side of the triangle that blocks the trap. In our case we do subscribe to the concept of preventing access into the corner region but instead of introducing a virtual third side we are proposing the intersecting vertices concept as described below:

3.4.1.1 Itemization of the procedural flow of the VOC

- **Need For a Virtual Obstacle (VO)**

The function VOC=f(I_{Ls}) is an index used to determine whether the VO is needed.

I_{Ls} = intercepted line of sight of robot from the goal position by obstacles that could cause a local minima problem as earlier stated in section 3 .

$$I_{Ls} = \begin{cases} 0 & \text{line of sight } \textit{of robot} \text{ is } \textit{not} \text{ int}\textit{ercepted by obstacle likely to cause a} \\ & \textit{local} \text{ min}\textit{ima problem} \\ 1 & \text{line of sight } \textit{of robot} \text{ is int}\textit{ercepted by obstacle likely to cause a} \\ & \textit{local} \text{ min}\textit{ima problem} \end{cases}$$

if I_{Ls}<>0 then VOC is prompted.

➢ **Positioning a Virtual Obstacle (VO)**

Step 1: Locate the vertices at the corners of the obstacles that form the entry into suspected minima traps e.g. Points p and p^1 in Figs. 3a, 3b and 3c are typical examples.

Step 2: Project a line $\overline{pp^1}$ as illustrated in Figs. 3a, 3b and 3c.

Step 3: Draw a straight line $\overline{oo^1}$ from the robot's position to the desired target point. This shows the natural trajectory of the robot to the target point assuming no obstacle was present see Figs. 3a, 3b and 3c.

Step 4: If $\overline{oo^1}$ intersects $\overline{pp^1}$ at any point along $\overline{pp^1}$ it implies that the robot's natural trajectory is in the direction of a trap.

Steps 1-4 are verification steps as to whether or not the VOC should be initiated. If it is verified that the robot's trajectory is in the direction of a trap then Step 5 is initiated.

Step 5: This introduces a virtual obstacle at the current position of the robot. This is done by using some form of production rules as control algorithms to impede the robot from motion. This prevents the robot from moving in the direction of local minima trap and hence facilitates the eventual minimization of the overall navigation time. Immediately following this step, proceed to implement the VGC as discussed below by setting $f_{voc} = 1$.

3.4.2 Virtual Goal Concept (VGC)

Assuming an object at an arbitrary position say $p_{i,j}$ is viewed from a translated position $p_{i+1,j+1}$ then the visibility of this object at its initial state $p_{i,j}$ from its current state $p_{i+1,j+1}$ forms a significant question posed by the VGC. The VGC in our work hinges on the concept of relative visibility. It operates on the principle of backtracking from an initially known position to a newly navigated position. The primary objective of the VGC in this context is to lead the robot from the point where the virtual obstacle was activated to the final target point through the aid of one or more temporary goals referred to as virtual goals.

The number of virtual goals and their relative position is a function of the geometry of the obstacles intercepting the visibility of the robot from the goal state. The degree of visibility of an object from its initial position $p_{i,j}$ as earlier described, relative to its new position $p_{i+n,j+n}$ determines whether the new position should be considered a virtual position of the object. If the object's visibility is obscured at point $p_{i+n,j+n}$ then the preceding point $p_{i+n-1,j+n-1}$ is considered for the virtual position of the object. For this purpose the intermediate points between $p_{i,j}$ and $p_{i+n,j+n}$ are selected along and close to the vertices of the intervening obstacle boundary. Figs. 3a to 3d above are also case illustrations of the VGC. The required steps for the VGC are itemized as follows:

3.4.2.2 Itemization of the procedural flow of the VGC

- **Need For a Virtual Goal (VG)**

Step 1: Is a VG needed?

If either the function f_{voc} or T_{nso} =1 is satisfied then a VG is needed.

Here

$$f_{voc} = \begin{cases} 0 & \text{virtual obstacle module is } \textit{not activated} \\ 1 & \text{virtual obstacle module is } \textit{activated} \end{cases}$$

T_{nso} has also been introduced to describe the situation that occurs when the robot and goal are at opposite ends of the space in between two narrowly spaced obstacles. Here although the orifice is wide enough the robot might still find it difficult to meander its way unaided through the clearance between the two obstacles to the goal position.When such a condition exists we set T_{nso} =1; otherwise T_{nso}=0 which is equivalent to the situation that would trigger the introduction of a virtual obstacle sealing off entry into the region.

- **Locating a Virtual Goal (VG)**

Step 1: Locate the goal position or the target point. This is as shown in Figs. 3a, 3b, 3c and 3d at point o^1.

Step 2: Locate the robot's position. This could also be seen in Figs. 3a, 3b, 3c and 3d at point o.

Step 3: From the goal position, backtrack along a projected line of sight towards the robot position skirting the perimeter of the intervening obstacle. Ensure that the line of sight is not intercepted by any obstacle(s). See Figs. 3a, 3b and 3c.

NOTE: The line of sight is drawn using the concept of ray theory. It is drawn to circumnavigate the obstacle(s) whose boundaries frame the local minima problem.

Step 4: locate and position virtual goals at appropriate points along the line of sight as you back track from the real goal to the robot position. The exact location of each virtual goal is a function of two variables: (V_{rg}, d).

where,

V_{rg}= relative visibility of most recent virtual goal from immediate preceding virtual goal or relative visibility of virtual goal from the real goal (in the case of the first virtual goal).

d= distance between VG and the edge of the obstacle obstructing robot's line of sight from the real goal.

These are as shown in Figs. 3a, 3b and 3c.

Furthermore,

$$V_{rg} = \begin{cases} 0 & \textit{when} \text{ relative visibility of virtual goal from immediate preceding virtual goal does not exist} \\ 1 & \text{when relative visibility of virtual goal from immediate preceding virtual goal does exist} \end{cases}$$

$d >= 0.5(dia_{robot})$

where,

V_{rg} = visibility of the virtual goal and

$dia_{robot} = diameter \ \ of \ \ robot$

Another important variable which is necessary to terminate the process of introducing VG is the variable (V_{mr}).

where,

V_{mr} represents the relative visibility of the most recent virtual goal from the robot.

$$V_{mr} = \begin{cases} 0 & \text{virtual } goal \text{ is } visible\ \ from\ \ robot\ \ position \\ 1 & \text{virtual } goal \text{ is } not\ \ visible\ \ from\ \ robot\ \ position \end{cases}$$

If V_{mr} =0 then no more VG is needed however if V_{mr} =1 then another VG is needed.

Step 4: Repeat step 3 until $V_{mr} = 0$.

3.5 Description of the Environment/Obstacle behaviour

The HVFF scheme described in Fig. 2f could be expressed using a mathematical model as shown in (36). This representation is a generalized form of the HVFF model. However the actual functionality of the model is subject to the nature of the navigation environment which could be described as being completely known, partially known or completely unknown. In addition, the obstacle configuration and behaviour could also influence the prompting of the HVFF model as expressed in (37) through (39). Obstacle configuration could be concave or convex while the behavior could be static or dynamic. It is to be noted that the governing theories for (38) and (39) are contained in chapter 4.

HVFF=VFF+VOC+VGC (36)

where,

HVFF= Hybrid Virtual Force Field,

VFF= Virtual Force Field

VOC= Virtual Obstacle Concept and

VGC= Virtual Goal Concept

3.5.1 Case of Partially Known Environment with Static Obstacles

$$HVFF = VFF + VOC + VGC \tag{37}$$

3.5.2 Case of completely Unknown Environment with Static Obstacles

$$HVFF = VFF + VGC \tag{38}$$

3.5.3 Case of completelyUnknown Environment with Dynamic Obstacles

$$HVFF = VFF + VGC \tag{39}$$

This chapter has presented the Hybrid Virtual Force Field model (HVFF) in a partially known 2-D environment for an autonomous mobile robot. The workspace consists of static obstacles and a robot desired goal state. The, Hybrid Virtual Force Field (HVFF), is an integration of the Virtual Obstacle Concept (VOC) and the Virtual Goal Concept (VGC) in combination with the traditional Virtual Force Field concept. The specific problems we have addressed in this chapter include the local minima problem posed by either lengthy or concave shaped obstacles as well as the potential field induced oscillatory motion of such a robot when maneuvering in the corridor between two narrowly spaced obstacles.

CHAPTER FOUR

A PATH PLANNING MODEL IN A COMPLETELY UNKNOWN DOMAIN FOR CONCAVE SHAPED OBSTACLE

4.1 PREAMBLE

In this chapter, the performance of the Hybrid Virtual Force Field (HVFF) concept on both static and dynamic concave shaped obstacles in a completely unknown environment is examined. Within this context, the problem domain poses considerable challenge when the robot navigates in a dynamic or non-stationary environment and particularly when there is no prior knowledge of the kinematics of the mobile obstacles. This type of path planning technique is obstacle constrained and is associated with reactive rather than deliberative control strategy. The obstacles were modeled as discrete circles albeit arranged in a concave arc. The robot considers the affinity of the discrete obstacles with priority given to the closest one which is assumed to represent the whole obstacle.

4.2 Algorithmic Procedure

The proposed hybrid concept is now described in some detail by giving an outline of the sequential order of the overall algorithm as illustrated in the Flow Chart in Fig. 4 below: We hereby note that unlike in the previous chapter, the activation of VOC and VGC in the current application is absolutely based on sensory data. The hybrid concepts are automatically prompted when a local minima trap is ascertained.

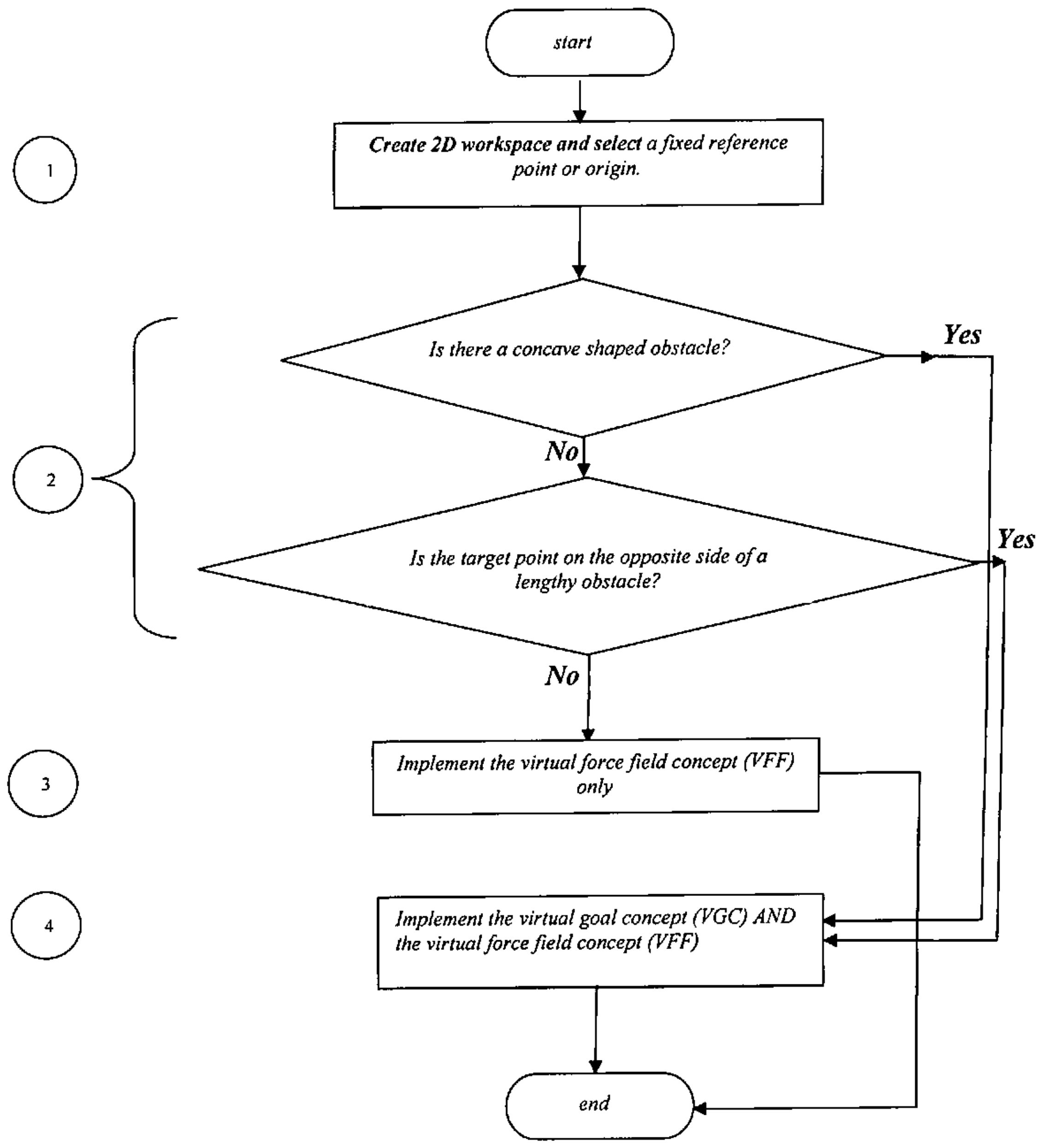

start
Create 2D workspace and select a fixed reference point or origin.
1
Is there a concave shaped obstacle?
Yes
No
2
Is the target point on the opposite side of a lengthy obstacle?
Yes
No
Implement the virtual force field concept (VFF) only
3
Implement the virtual goal concept (VGC) AND the virtual force field concept (VFF)
4
end

Figure 4: A flow chart diagram for the HVFF algorithm for dynamic obstacles

The mathematical concept of lengthy obstacles as shown in equations (29) through (35) for Fig. 3e is as well adaptable to Fig. 5.

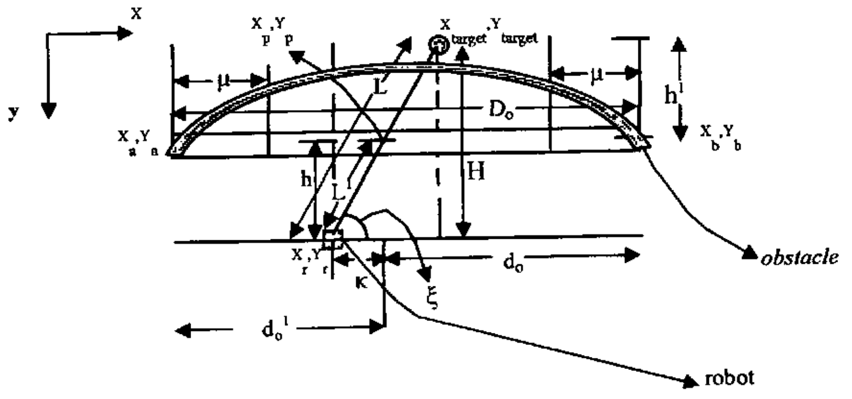


Figure 5: Model of an assumed thin and lengthy obstacle.

Consider the different cases that emerged as a result of the relative position of the robot and the target on either side of the obstacle. The procedure to adopt is a function of the robot-obstacle relative position as well as the interplay of the variables $L^1, \xi, \kappa, X_p, Y_p, d_o, d_o^1$ *and* D_o as analyzed below.

4.3 Virtual Obstacle Concept (VOC):

Unlike in our previous application where the robot is not attracted into a corner region of obstacles that has no outlet, this application, in the first instance, allows the robot to navigate towards the target point not mindful of whether a concave trap is on its line of sight. One significant way by which the robot confirms the presence of local trap is with the aid of on-board sensors. The robot updates and compares its current orientation with orientation at the onset of navigation. If the difference has an absolute value ranging between ψ_1 a default lower orientation limit on the cardinal system and ψ_2 a default upper orientation limit on the cardinal system of navigation then it is an indication that a local trap is present. This prompts the VOC and VGC into action in the case of static obstacle and VGC alone in the case of dynamic obstacle.

4.3.1 Itemization of the procedural flow of the VOC

- **Need For a Virtual Obstacle (VO)**

The function VOC=f(I_{Ls}) is an index used to determine whether the VO is needed.

I_{Ls} = index for recording the presence of an intercepted line of sight of robot from the goal position by obstacles that could cause a local minima problem as earlier stated in section 3

$$I_{Ls} = \begin{cases} 0 & \text{line of sight } of\ robot \text{ is } not \text{ int}ercepted\ by\ obstacle\ likely\ to\ cause\ a \\ & local \text{ min}ima\ problem \\ 1 & \text{line of sight } of\ robot \text{ is int}ercepted\ by\ obstacle\ likely\ to\ cause\ a \\ & local \text{ min}ima\ problem \end{cases}$$

$If\ I_{LS} = 1\ \ then$,

$if\ abs(\theta - \delta) >= \psi_1\ and\ abs(\theta - \delta) <= \psi_2\ then$ prompt VOC. (40)

4.4 Virtual Goal Concept (VGC)

The primary objective of the VGC in this context is to redefine a new line of sight capable of bringing out the trapped robot from the local minima environment and redirect same to the target point. In the current application, the virtual goal (VG) placement scheme is hinged on two procedures viz: (i) axial line selection from a generalized coordinate system and (ii) VG placement position on selected axial line. Assuming that the ends of the axial lines defining the generalized coordinate frame as shown in Fig. 6 are numbered corresponding to the virtual goal they represent such that $v_i = f(v_{xi}, v_{yi})$ where $i = 1\ to\ n$ then the different virtual goal locations could be obtained as shown below viz:

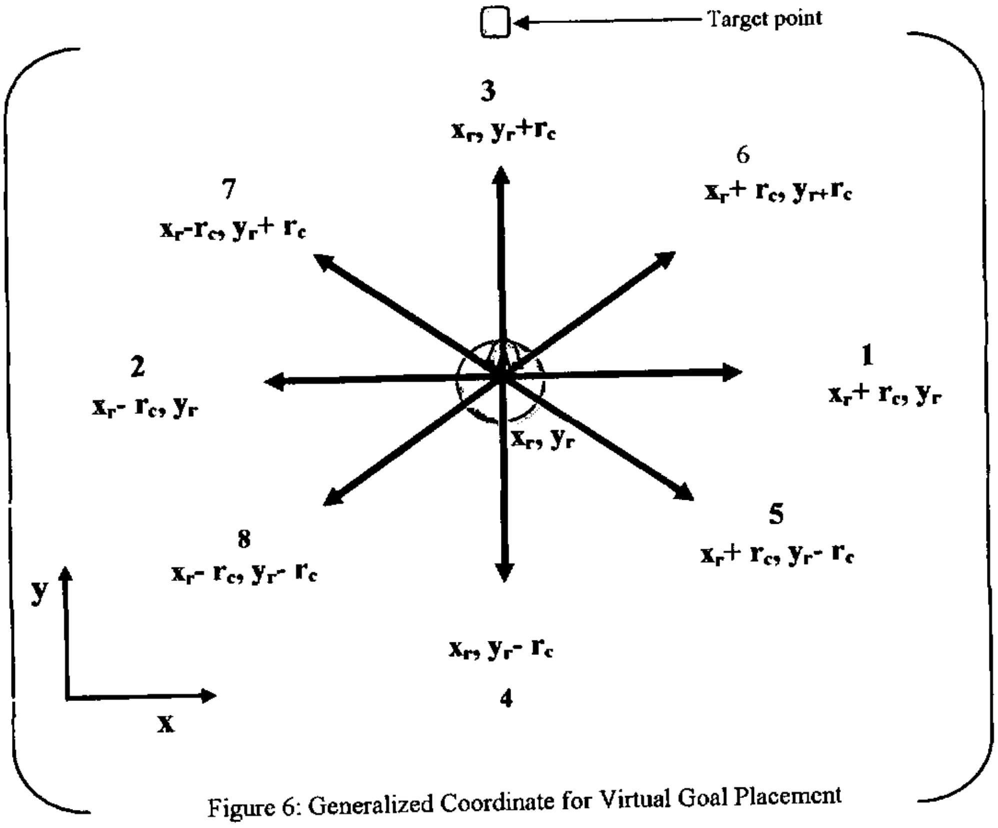


Figure 6: Generalized Coordinate for Virtual Goal Placement

$$\left\{\begin{array}{ll}
v_{x1} = f(x_r + r_c) & v_{x5} = f(x_r - r_c) \\
v_{y1} = f(y_r) & v_{y5} = f(y_r + r_c) \\
 & \\
v_{x2} = f(x_r - r_c) & v_{x6} = f(x_r + r_c) \\
v_{y2} = f(y_r) & v_{y6} = f(y_r - r_c) \\
 & \\
v_{x3} = f(x_r) & v_{x7} = f(x_r - r_c) \\
v_{y3} = f(y_r + r_c) & v_{y7} = f(y_r - r_c) \\
 & \\
v_{x4} = f(x_r) & v_{x8} = f(x_r + r_c) \\
v_{y4} = f(y_r - r_c) & v_{y8} = f(y_r + r_c)
\end{array}\right\} \tag{41}$$

4.5 AXIAL LINE SELECTION PROCEDURE

The axial line selection procedure is dominantly a function of the relative positions of the (robot, target point and concave obstacle) workspace objects. By default from our model, the VGC is a function of the VOC i.e. $VGC = f(VOC)$ hence,

$$VGC = \begin{cases} 0 & \text{if VOC does not activate} \\ 1 & \text{if VOC does activates} \end{cases}$$

If VGC =1 then the axial line selection algorithm activates.

Assuming the end points of each axial line and the corresponding virtual target placed at these points are denoted by i and v_i respectively such that i ranges from $1 \; to \; n$ then if d_{vi_target} represents the Euclidean distance between the virtual targets at the end points of the axial lines v_i and the real target, then the generalized concept

$d_{vi\_target} = \|(X_{target} - v_{xi}) + (Y_{target} - v_{yi}) \|$ holds for all axial lines.

In the current application, $n = 8$ i.e. number of points available for VG placement. In an elaborate form our model could be represented as follows:

$$\begin{aligned}
d_{v1\_target} &= \|(X_{target} - v_{x1}) + (Y_{target} - v_{y1}) \| & d_{v2\_target} &= \|(X_{target} - v_{x2}) + (Y_{target} - v_{y2}) \| \\
d_{v3\_target} &= \|(X_{target} - v_{x3}) + (Y_{target} - v_{y3}) \| & d_{v4\_target} &= \|(X_{target} - v_{x4}) + (Y_{target} - v_{y4}) \| \\
d_{v5\_target} &= \|(X_{target} - v_{x5}) + (Y_{target} - v_{y5}) \| & d_{v6\_target} &= \|(X_{target} - v_{x6}) + (Y_{target} - v_{y6}) \| \\
d_{v7\_target} &= \|(X_{target} - v_{x7}) + (Y_{target} - v_{y7}) \| & d_{v8\_target} &= \|(X_{target} - v_{x8}) + (Y_{target} - v_{y8}) \|
\end{aligned} \tag{42}$$

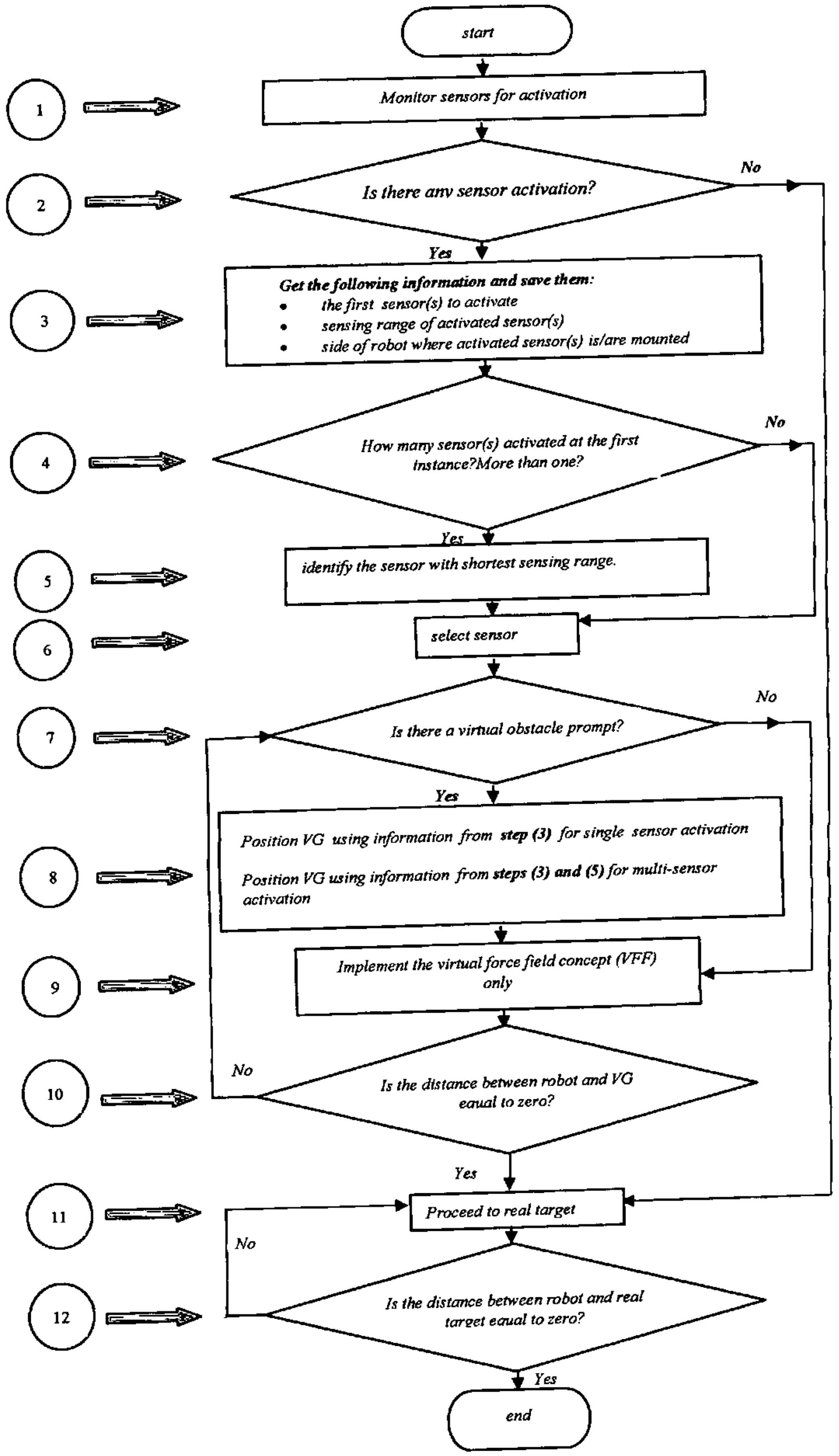

start
1
Monitor sensors for activation
2
Is there anv sensor activation?
No
Yes
3
Get the following information and save them:
the first sensor(s) to activate
sensing range of activated sensor(s)
side of robot where activated sensor(s) is/are mounted
4
How many sensor(s) activated at the first instance?More than one?
No
Yes
5
identify the sensor with shortest sensing range.
6
select sensor
7
Is there a virtual obstacle prompt?
No
Yes
8
Position VG using information from step (3) for single sensor activation
Position VG using information from steps (3) and (5) for multi-sensor activation
9
Implement the virtual force field concept (VFF) only
10
No
Is the distance between robot and VG eaual to zero?
Yes
11
Proceed to real target
No
12
Is the distance between robot and real target eaual to zero?
Yes
end

Figure 7: A flow chart diagram for VG placement

Assuming points v_i *and* v_{i+1} represent the two virtual goal points of a particular axial line then it follows that if $d_{vi\_target} > d_{target}$ *and* $d_{v(i+1)\_target} > d_{target}$ then the axial line with end points i *and* $i+1$ would be selected for VG placement. Taking Fig. 6 as a case for application of our axial line selection algorithm, it would be seen that the axial line with end points 1 and 2 above others satisfy our generalized model resulting in

$d_{v1\_target} > d_{target}$ *and* $d_{v2\_target} > d_{target}$

4.6 VIRTUAL GOAL (VG) PLACEMENT POSITION

Assuming our robotic system is fully autonomous i.e. sensor driven such that we have onboard sensors $s(i)$ mounted with i ranging from (1 to n) then we could have a sensor distribution system of the form $s(i)$, $s(i+1),\ldots.., s(i+n-1)$ and $s(i+n)$. The selection of VG placement position between the two end points of any axial line of choice such as points: **1&2, 3&4, 5&7 and 6&8** as shown in Fig. 6 is sensor driven and instantaneous. This process directly follows the axial line selection process. The flow chart in Fig. 7 describes the steps involved in the VG placement algorithm.

As summarized in Fig. 7, the interest is to identify either the first sensor to activate i.e. a case of single sensor activation and the side of the robot where the sensor is mounted while in the case of multi-sensor activation, the focus is on identification of the particular sensor with the shortest radiation cone and the side of the robot where the sensor is mounted.

The proximity concept of the onboard sensors is used. By this, it is meant that the sensing range is reduced to a very small distance usually of the order of a few multiples of the robot's diameter. The robot is allowed this close range with obstacles along its line of sight in order to minimize or isolate interference effects from other workspace obstacles whose navigation path for dynamic obstacles may not be directly in the way of the robot.

4.7 Reoccurring Position Technique (RPT)

The distance between the VG and the robot along the axial lines is given by r_c as deployed in the set of equations in (19) where, r_c is the first sensor reading as the robot senses an obstacle assuming only one sensor activates while in the case of two or more sensors activating simultaneously, r_c is the sensor reading with the shortest radiation cone. The reoccurring position technique (RPT) is a control algorithm that keeps prompting the VGC provided the robot is still within the confines of the local minima trap. It is a feedback concept that checks the current status of the robot in respect of its position and updates. Below is a generalized form of the RPT control algorithm:

$$if \;\; abs(\theta - \delta) >= \psi_1 \;\; and \;\; abs(\theta - \delta) <= \psi_2 \;\; then \;\; for \tag{43}$$

$$v_i = f(\, v_{xi}, v_{yi}) \;\; do$$

$$v_{xi} = f(\, x_r \pm r_c) \;\; and$$

$$v_{yi} = f(\, y_r \pm r_c) \qquad where,$$

$$r_c = \begin{cases} 0 & \text{if } v_{xi} = x_r \text{ or } v_{yi} = y_r \\ r_c & \text{if } v_{xi} \neq x_i \text{ or } v_{yi} \neq y_r \end{cases}$$

$$if \;\; abs(\theta - \delta) <= \psi_1 \;\; or \;\; abs(\theta - \delta) >= \psi_2 \;\; then \;\; exit \;\; RPT \tag{44}$$

4.8 Adaptive Velocity Concept

The adaptive velocity model predicts how the robot will respond to changes in the speed of an on coming obstacle under varying conditions. Our adaptive scheme is hinged on a predictive concept that depends on the rate of change of the length of the radiation cone generated by sensors when an obstacle is encountered in the workspace.

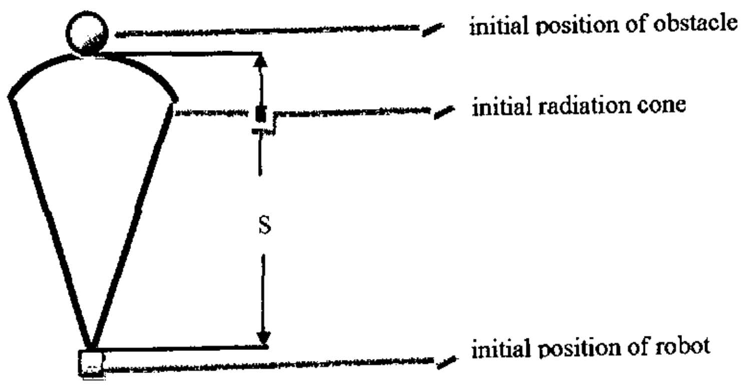


Figure 8: A view of the radiation cone on first sight of an obstacle by the robot.

Fig. 8 shows the initial conditions of the radiation cone at the robot's first encounter with an obstacle. Fig. 9 is a schematic representation of the initial and current conditions of the radiation cones as the robot navigates although the obstacle in this figure is static. Motion can be detected as differences between successive scans since moving objects change sensor readings.

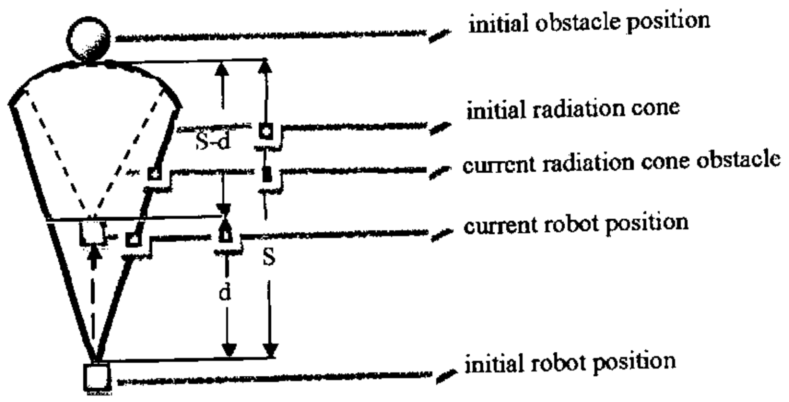


Figure 9: A view of the radiation cone assuming a static obstacle and a moving robot

Assuming the robot's velocity prior to navigation is set as v_{robot} then, the current distance between the robot and the obstacle is given by

$$\Omega = S - (d + d^1) \tag{45}$$

where,

S is the initial distance between the robot and the obstacle

$$d = \sqrt{(x_{initial} - x_{current})^2 + (y_{initial} - y_{current})^2)} \quad (46)$$

is the distance between the initial and current position of the robot while the time taken

$$t = d / v_{robot} \quad (47)$$

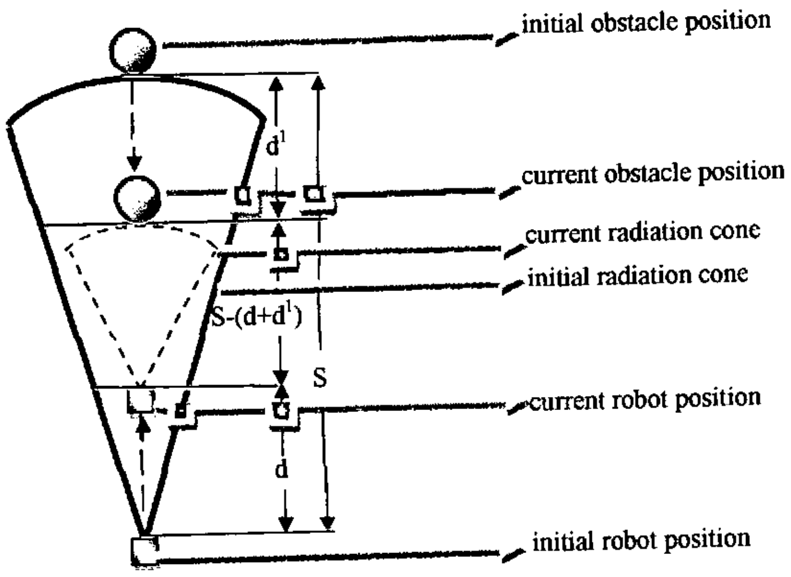


Figure 10: A view of the radiation cone assuming both obstacle and robot are dynamic

In the case of static obstacles as shown in Fig. 9, $d^1 = 0$ while $d^1 > 0$ when the sensed obstacle is dynamic as for the example shown in Fig. 10. The speed of the obstacle on the other hand is given by:

$$v_{obstacle} = d^1 / t \quad (48)$$

$If\ d + \Omega < S\ then\ d^1 > 0\ however,$

$If\ d + \Omega = S\ then\ d^1 = 0$

For the prompting of the adaptive velocity, the following conditions as stated below must be satisfied i.e.

$if\ v_{robot} - v_{obstacle} < = \in\ and\ abs(\theta - \delta) >= \psi_1\ and\ abs(\theta - \delta) <= \psi_2\ then$

$if\ v_{robot} - v_{obstacle} <= \in\ and\ abs(\theta - \delta) <= \psi_1\ or\ abs(\theta - \delta) >= \psi_2\ then$

(49)

$$v_{robot} = \lambda * v_{obstacle} \tag{50}$$

where,

λ is a real number (constant) >= 2

$\in$ represents a minimum value below which the adaptive velocity concept prompts.

The robot's velocity however is returned to the initial value when the conditions in (51) below are satisfied i.e. the (obstacle→ target) distance > (robot→target) distance. It should be noted that this concept is applicable to dynamic obstacles and when they are along the robot line of sight.

$$If\ (\sqrt{(x_{current} - x_{target})^2 + (y_{current} - y_{target})^2} <= \sqrt{(x_{orient} - x_{target})^2 + (y_{orient} - y_{target})^2}) \tag{51}$$

$$then\ \ v_{robot} = v_{robot}$$

where,

$x_{current}\ and\ y_{current}$ represents the current coordinates of the robot on the x and y axes respectively

$x_{orient}\ and\ y_{orient}$ represents the coordinates of the robot at the point of orientation where the VG was prompted.

In this chapter, we have presented the Hybrid Virtual Force Field (HVFF) concept on both static and dynamic obstacles in a completely unknown environment. The obstacle geometry emphasized in this chapter is primarily the concave shape obstacle which has proven to be the major cause of the local minima trap in robot navigation. The obstacles were modeled as discrete circles albeit arranged in a concave arc. The robot considers the affinity of the discrete obstacles with priority given to the closest one which is assumed to represent the whole obstacles. Furthermore, the adaptive velocity concept was also considered for dynamic obstacles.

CHAPTER 5

VALIDATION EXERCISE: SYSTEMS DESIGN AND IMPLEMENTATION

5.1 PREAMBLE

Having developed the HVFF concept for the navigation of a robot within an obstacle clustered 2-D workspace in search of a specified goal, two separate exercises were conducted to validate the algorithms developed in Chapters 3and 4. For one thing, by using a customized mobile robot simulator, it is possible to conduct simulations to test the efficacy of the proposed schemes and in particular compare them with other existing algorithms developed by earlier workers in the field. In carrying out such an exercise, we payed special attention to some of the peculiar workspaces where existing algorithms either have limited or varied levels of success. The results of this exercise are discussed in Chapter Six.

A second exercise involves the design and construction of a simple prototype autonomous mobile robot (AMR) on which has been deployed a navigation sub-system based on our proposed HVFF paradigm and algorithms. The successful performance of such a system in a 2-D workspace test bed environment will give added confidence as to the feasibility of the scheme.

We next describe some of the components used to construct a laboratory AMR for the demonstration of the practicability of the HVFF navigation software.

5.2 HARDWARE

The basic hardware components deployed for this implementation process are viz: BX-24 Microcontroller, Hitec Servomotors, MaxSonar-EZ1 ultrasonic sensors, Connecting cables, Caster wheel, 3 inches diameter Rolling wheels and 3mm thickness Aluminium Sheet.

5.2.1 BX-24 Microcontroller

The BX-24 microcontroller was developed by Netmedia inc. In general terms, a micro-controller is a highly integrated chip that contains all the components comprising a controller. Typically this includes a CPU, RAM, ROM, I/O ports, amd timers. They are sometimes called "embedded" chips meaning that they are part of a large device or system. Following that it is the central processing unit of the robot, a microcontroller is responsible for task coordination, monitoring and control. Intra-modular communication among the hardware components is also done by the microcontroller. It allows a coalition of programs to share data and commands through an efficient shared memory mechanism. In our context, BX-24 is a 24- pin controller having 3-pin equivalence on the motherboard. Each of these pins has a specific identity and signal type. The pin closest to the microcontroller is the signal pin, while the middle pin is meant for power and the fartherest pin is the ground pin for earthing. We used a Pentium IV PC with windows XP operating system for downloading our codes via serial cable to the microcontroller.

5.2.2 Hitec Servomotors

Servomotors are prime movers of the robotic system. They are actuators that convert digital signals into physical signals. They are the agents that transform perception to action. Servos have a measure of intelligence following their feedback capability. Though electromechanical devices, they are most commonly found in radio controlled (R/C) airplanes, cars and boats. Furthermore, they are small mechanical devices whose sole purpose is to rotate a tiny shaft extending from the top of the servo housing.

5.2.3 MaxSonar-EZ1 ultrasonic sensors

The MaxSonar-EZ1 ultrasonic sensor is a proximity sensor produced by MaxBotix® Inc. It is a sound based sensor. Usually, sound pulses of frequencies relatively higher than the human hearing range are emitted and echo received in turn. The time interval between sound signal dispatch and echo reception is used to determine the position of an obstacle in the workspace. Sound frequency which is above the limit of human hearing is described as ultrasound. The lower limit is at approximately 20 kHz. The particular characteristics of ultrasound applied to proximity sensors are the result of the high frequency and the correspondingly short wavelength.

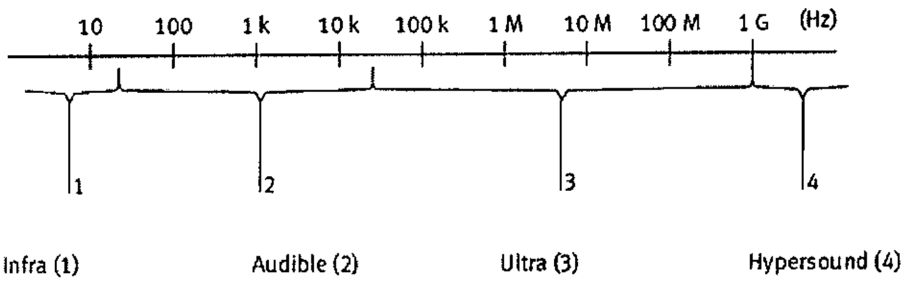
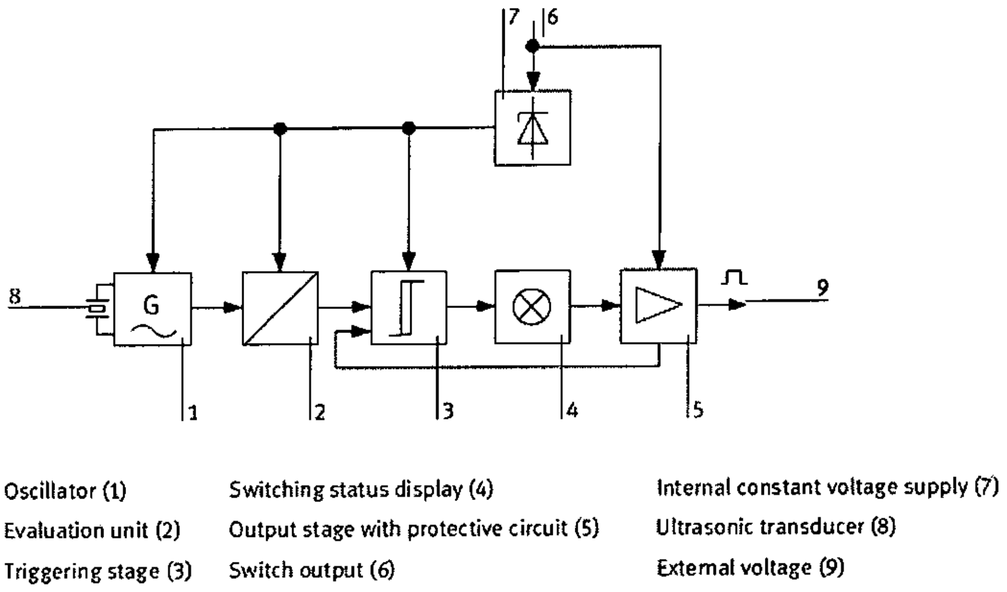


Figure 11a: Sound Frequency Range



7
6
8
G
9
1
2
3
4
5
Oscillator (1)
Evaluation unit (2)
Triggering stage (3)
Switching status display (4)
Output stage with protective circuit (5)
Switch output (6)
Internal constant voltage supply (7)
Ultrasonic transducer (8)
External voltage (9)

Figure 11b: Block Circuit diagram of an Ultrasonic sensor (Source: Festo Didactic GmbH & Co. KG • FP 1110)

A high frequency alternating voltage is generated to excite the piezoceramic module into oscillation. This AC voltage is switched through to the ceramic module by means of a pulse generator, when the transmitting pulse is to be emitted. Distance measurement is calculated according to the ultrasound propagation time. A ramp generator is triggered at the time of transmission, which generates a time dependent voltage. Thereupon, the piezoceramic module is switched over to receiving. The ultrasonic signal is reflected if an object is present in the active range of the proximity sensor. The proximity sensor receives the signal and the ramp generator is stopped. The voltage level is evaluated at this point and an output signal emitted.

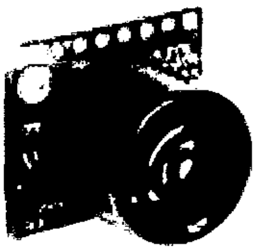


Figure 11c: A pictorial view of MaxSonar EZ1 sensor

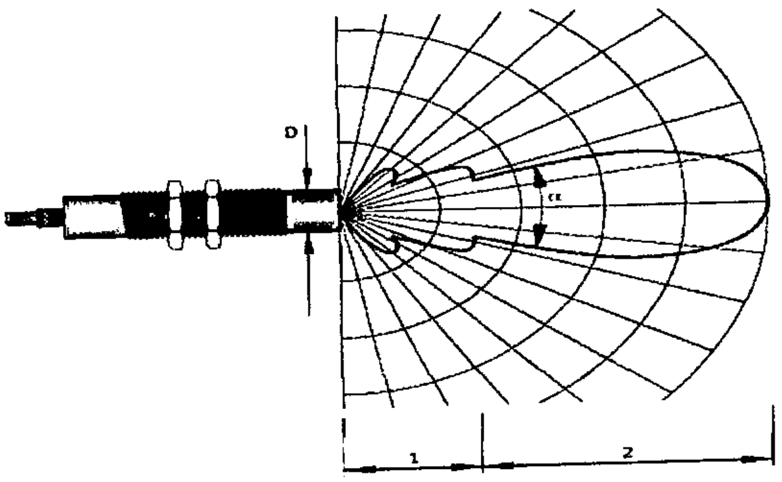


Figure 11d: Block Circuit diagram of an Ultrasonic sensor (source: Festo Didactic GmbH & Co. KG • FP 1110)

An object must not be present in the sound field of the proximity sensor within the so-called near field, as this can lead to error pulses at the proximity sensor output. For an ultrasonic proximity sensor with a transducer diameter of 15 mm and an emitting frequency of 200 kHz, the range of the near field is approximately 130 mm.

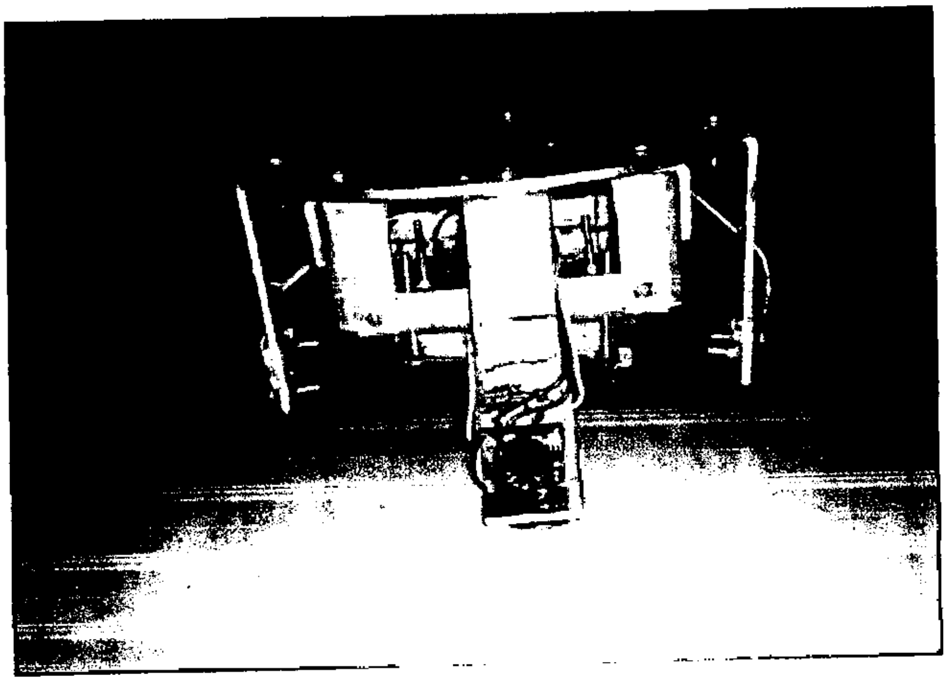


Figure 12a: Front view of our AMR

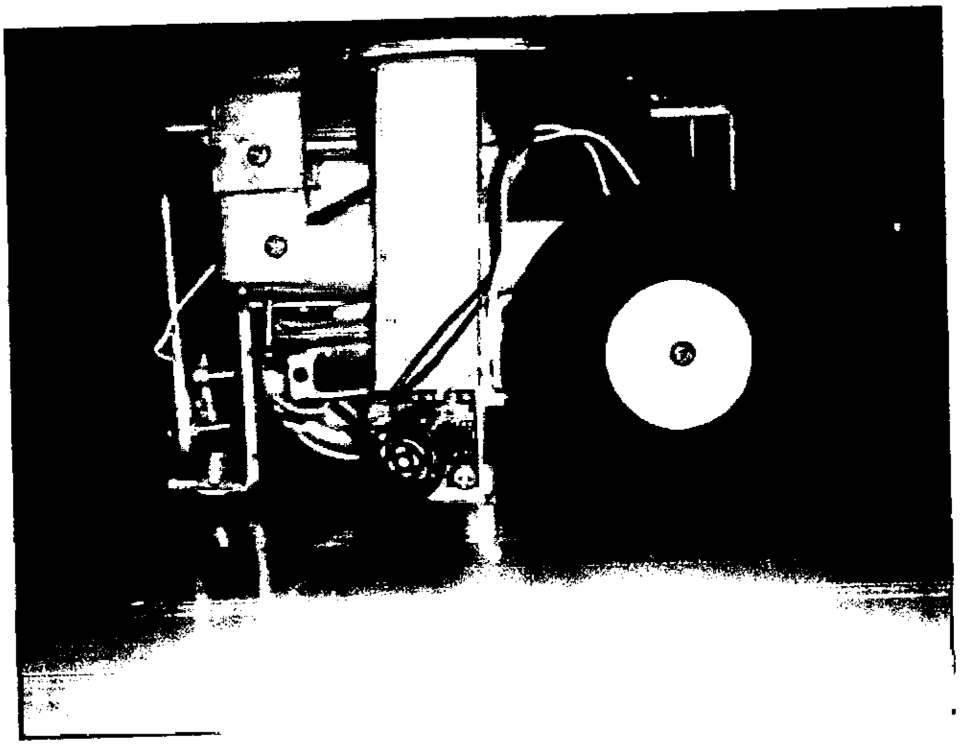


Figure 12b: Side view of our AMR

5.2.4 Connecting cables

The connecting cables deployed are pre-designed for a 3-pin controller system. The cables are glued together into an entity of distinct signal types. Usually, the lightest with yellow or white colour is meant for signal transmission, while the middle cable which is red in colour is for power and the third coloured black is for ground or earthing.

5.2.5 Caster wheel

The caster wheel is basically a redundant wheel mounted at the front of the AMR to facilitate navigation. We used a ball like wheel of relatively small size just enough to aid motion.

5.2.6 Rolling wheels

For the locomotion of mobile robots the most often used components are: wheels or rollers, crawlers and feet for walking robots. The wheels are driven by actuators. The two wheels mounted at the back of the AMR to the left and right sides are differential wheels. Each of these wheels has a dedicated servo and is independently controlled. The wheels are about 3 inches in diameter.

5.2.7 Aluminium Sheet

The car-like design of our AMR was achieved through design and fabrication of the chasis. Aluminium sheet of 3mm thickness was used due to its relative lightness. Details on the Chasis design using Aluminium sheet is available in the appendix.

5.3 Software

The software used for this validation procedure is an integral part of the microcontroller. The BX-24 microcontroller uses a hybrid software christened basicX. Though with some unit instruction set, BasicX emerged from the Qbasic and Vbasic softwares.

5.4 Control System Architecture

The control system architecture of our built AMR is as shown in Fig. 13. It is a Sensor-based control system capable of operating in an unstructured environment with variable traversability of the state space. The structure is made up of a closed loop which consists of sensors, perception, control and actuation. A typical autonomous mobile robot system requires at least five control levels, running at different cycle times. These are namely motor control or actuation, emergency supervision, obstacle avoidance, localization and planning of the task.

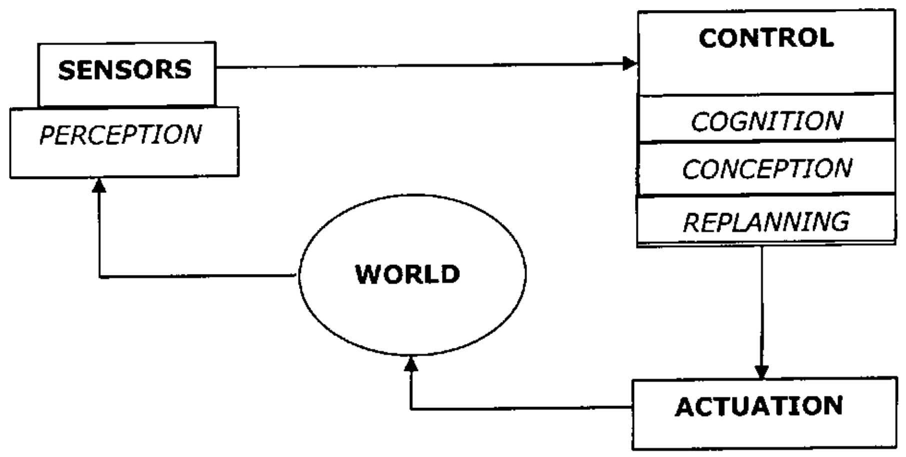


Figure 13: Control System for our AMR .

The control module implements both wall following and obstacle avoidance in the same module. When an unexpected obstacle blocks the robot trajectory, the machine uses the robot sensors to detect the obstacle, finds a way to circumvent it and then resumes its path towards the desired destination.

5.5 Robotic Testbed

A testbed was developed for implementation of our AMR. The platform made of polished wooden sheet has a dimension of (170 x120x8.75) cm. Obstacles of different sizes were carved out from the same polished woods and used to create different obstacle configurations ranging from convex shapes to concave shapes.

5.6 Forward Kinematics Model of The Differential Drive Wheels

For the purpose of effective control of the vehicle motion, a differential drive mechanism was deployed on the rear wheels with the use of servo controlled motors. Fig. 14a shows a stationary vehicle with the independent wheels. The two rear wheels are servo controlled based on the differential drive mechanism while the front wheel is a redundant caster wheel.

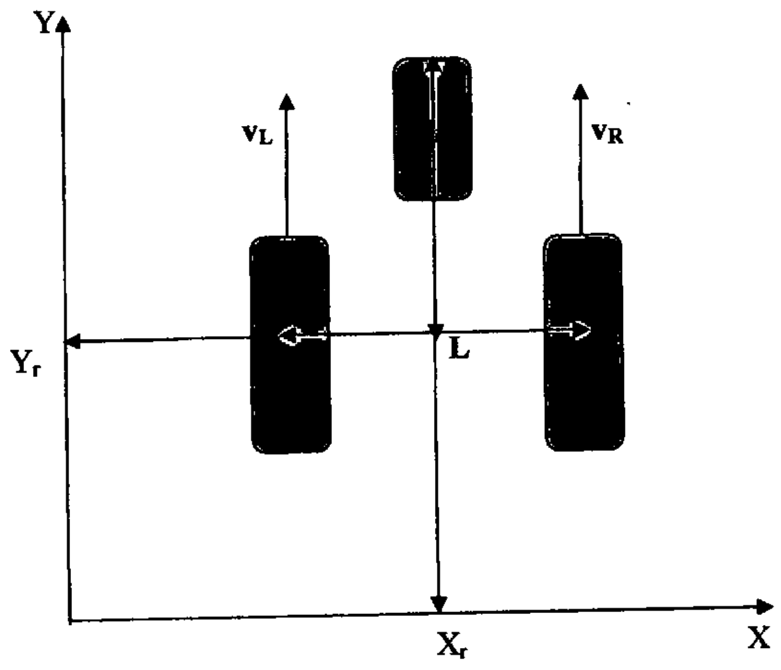


Figure 14a: Differential wheel drive system: a schematic view

The velocities v_L and v_R are the respective velocities of the left and right rear wheels. L is the distance interval between the two rear wheels. It ranges from the mid-point of one wheel to the other. The vectorial position of the vehicle on the measurement plane is given by the coordinate values (X_r, Y_r). The differential mechanism operates in two ways viz:

i. keep one wheel stationary and drive the other wheel forward or backwards.

ii. drive one wheel in the reverse direction and the other in the forward direction and vice-versa

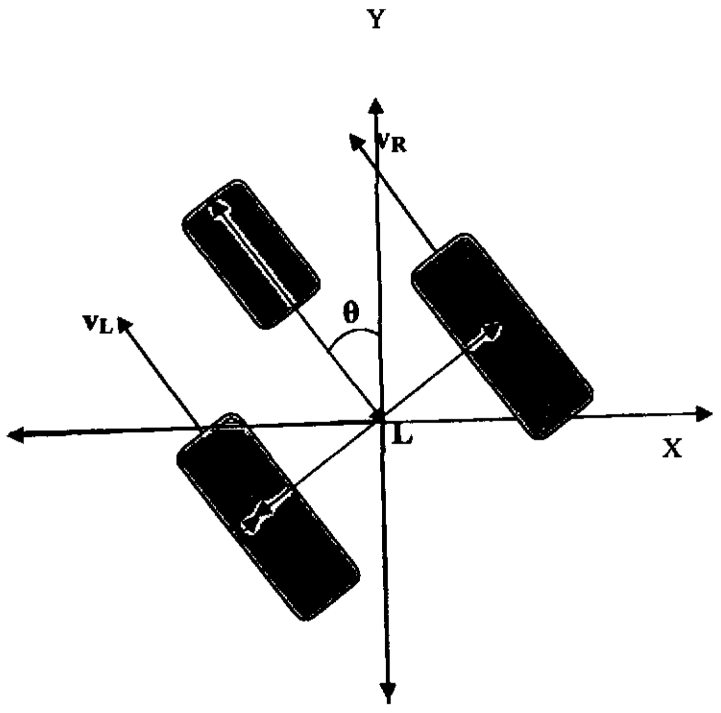


Figure 14b: A rotated view of an AMR in the direction of the target point

Fig. 14b is an orientated view of the vehicle with an orientation of θ. The quadrant system in Fig. 14c, aims at discretizing the navigation process into sub-modules for effective position and orientation tracking.

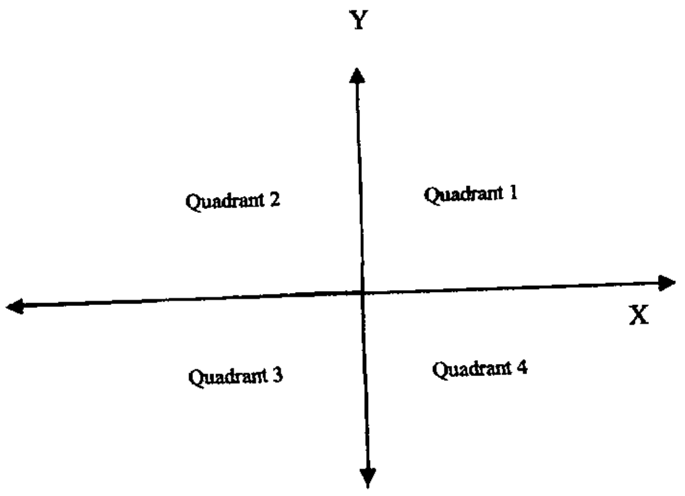


Figure 14c: A view of the robot's quadrant

5.7 Model Assumptions for the prototype robot:

The following underlisted are highlights of the assumptions considered in modeling the real prototype robot.

i. at rest the robot's head is usually in the direction of the +ve (y axis) with an initial orientation of 90 degrees.

ii. the cartesian plane is divided into four quadrants as seen in Fig. 14c above

iii. at rest position, the coordinate reading for the robot is zero for both axes.

iv. apart from the sensory based obstacle, every other conditions that could inhibit navigation are assumed normal

5.8 Modeling of the Robot's Rotational Difference

Below is a diagrammatic representation of the robot's orientation before and at the prompt of navigation. In Fig. 14d, the robot is inactive with an initial orientation of $\theta_{initial}$ while Fig. 14c shows the robot's new orientation $\theta_{rotated}$ at the prompt of navigation. With this the robot aligns its line of sight with the target position before translating.

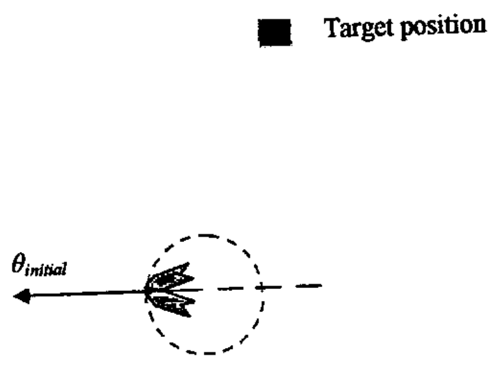


Figure 14d: Initial orientation of robot before navigation

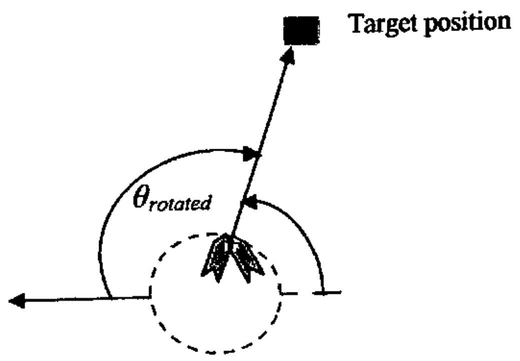


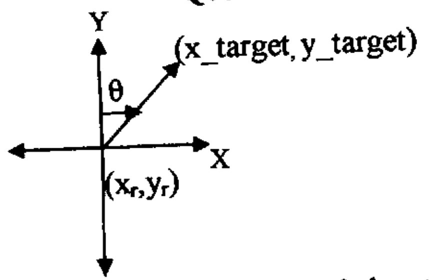
Figure 14e: Current orientation of robot during navigation

5.8.1 Identifying Quadrants for Robot Initial Orientation

The quadrant to which a robot orientates is a function of the position vector of the target point. Below is an highlight of the different target point positions within the quadrant system:

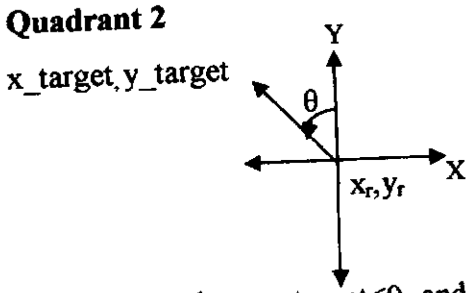
Case I: when x_target>0 and y_target>0 then following that by default the robot is at the centre of origin as shown below where $x_r = 0$ *and* $y_r = 0$ then the robot's orientation would be in the direction of the first quadrant.

Quadrant 1

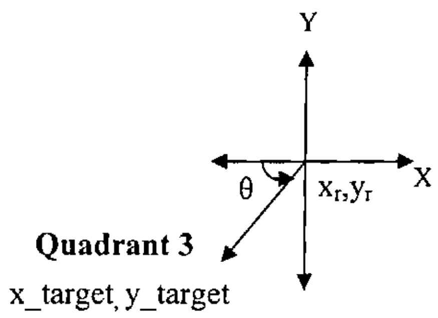


Case II: when x_target<0 and y_target>0 then the rotational difference of the robot orientates towards the second quadrant

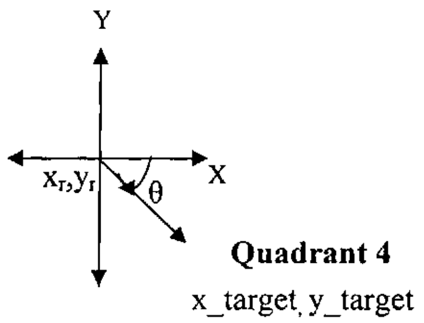
Quadrant 2



Case III: when x_target<0 and y_target<0 then the rotational difference of the robot orientates towards the third quadrant



Case IV: when x_target>0 and y_target<0 then the rotational difference of the robot orientates towards the fourth quadrant



In cases 1 and 2 above, $\theta = \arctan\left(\frac{\text{x\_target} - x_r}{\text{y\_target} - y_r} \right)$ (52)

While,

In cases 3 and 4, $\theta = \arctan\left(\frac{\text{y\_target} - y_r}{\text{x\_target} - x_r} \right)$ (53)

Where,

θ = rotational difference between the robot and target

x_target= x coordinate of target point and,

y_target= y coordinate of target point

5.9 Navigation Algorithm for Real Robot

Fig. 14f represents a simple trajectory of a robot navigating from an initial point O to a target point. This is a display of both obstacle avoidance and pathplanning. The triangle OAB formed by the robot's trajectory is a simple description of different aspects of the entire navigation process to the target point. Between the interval $|OA|$ the robot navigates in an obstacle free path hence moving along the shortest path which is a trajectory formed by a straight line. At point A, an obstacle is sensed and this prompts the repulsive algorithm resulting in a new trajectory along $|AB|$ by an orientation δ.

The prompting of the virtual objects i.e. the virtual obstacle and the virtual goal is a function of the orientation δ. If δ >= 4/9 of a revolution it implies that there is a trap surrounding the robot and preventing it from reaching the goal along the initial trajectory hence the virtual objects are prompted to establish a new robot trajectory and redirect it from the trap. Detailed explanation about the virtual objects modeling is contained in chapter 3 for the static case of partially known environment and chapter 4 for the completely unknown environment respectively.

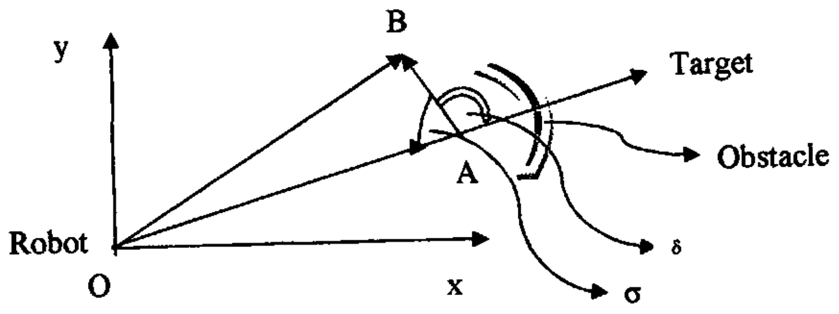


Figure 14f: A simple robot navigation sketch

In modeling the navigation process, we first initialize the robot's coordinate at point O the origin:

$x_r = 0$ and $y_r = 0$

where,

x_r = robot coordinate on the x-axis.

y_r = robot coordinate on the y-axis

Also, the navigation counter is initialized as: counter = 0

while the initial distance L travelled = 0

5.9.1 Navigation Algorithm without obstacle

The model representing trajectory $|OA|$ i.e. navigation path free of obstacle is as given below:

counter = counter+1.0

length_per_step = constant

L=length_per_step*counter

$$x_r = L * \sin(\theta) \tag{54}$$

$$y_r = L * \cos(\theta) \tag{55}$$

$$dis\_t\arg = \|(x_r - x\_t\arg et) + (y_r - y\_t\arg et)\| \tag{56}$$

if $dis\_t\arg <= 0.01$ then exit.

where,

counter = the number of times navigation pulses are sent to the servo.

dis-targ =Euclidean distance between robot and target

5.9.2 Navigation Algorithm with obstacle

In addition to the models in sub-section 5.9.1, once an obstacle is sensed during navigation the following algorithms are prompted

$$Fry = (\mathrm{y_r - y\_target}) * [\ (\frac{1}{s}) - (\frac{1}{s_o})\ ] \tag{57}$$

$$Frx = (\mathrm{x_r - x\_target}) * [\ (\frac{1}{s}) - (\frac{1}{s_o})\ ] \tag{58}$$

$$Fcx = (\mathrm{x\_target - x_r}) * \frac{1}{\mathrm{dis\_targ}} \tag{59}$$

$$Fcy = (\mathrm{y\_target - y_r}) * \frac{1}{\mathrm{dis\_targ}} \tag{60}$$

where,

Frx = x coordinate value for repulsive potential

Fry = y coordinate value for repulsive potential

Fcx = x coordinate value for attractive potential

Fcy = y coordinate value for attractive potential

s = actual sensor reading

s_o = minimum threshold sensor reading for repulsion

dis_targ = euclidean distance between robot and target

From Fig. 14f above, the vectors $|\vec{OA}|$, $|\vec{OB}|$ and $|\vec{AB}|$ could be determined using the following equations:

$$|\vec{OB}| = \|(x_r - (x_r - rx)) + (y_r - (y_r - ry))\| \tag{61}$$

$$|\vec{OA}| = \|(x_r - (x_r - rx)) + (y_r - (y_r - ry))\| \tag{62}$$

$$|\vec{AB}| = \|(x_r - (x_r - rx)) + (y_r - (y_r - ry))\| \tag{63}$$

Furthermore, from cosine rule we can determine the orientation σ from Fig. 14f

$$\sigma = \cos^{-1}\left[\frac{|\text{OA}|^2 + (|\text{AB}|^2 - |\text{OB}|^2)}{2 * |\text{OA}|^{2} * |\text{AB}|^2} \right] \tag{64}$$

resulting in,

$$\delta = 180^0 - \sigma \tag{65}$$

δ represents the robot orientation while avoiding an obstacle in its travel direction

$$Rx = Frx + Fcx \tag{66}$$

$$Ry = Fry + Fcy \tag{67}$$

where,

R_x = robot resultant position on x axis

R_y = robot resultant position on y axis

$$x_r = x_r + Rx \tag{68}$$

$$y_r = y_r + Ry \tag{69}$$

x_r and y_r are cumulative variables driven by the resultants from the potentials

if dis_targ <=0.01 then exit navigation.

where,

x_r = x coordinate of robot.

y_r = y coordinate of robot.

This chapter has provided a summary of the modeling procedure, analysis, and general considerations in the design, construction and implementation of a simple prototype autonomous mobile robot which has served as a platform for further validation of our proposed HVFF scheme.

CHAPTER SIX

RESULTS AND DISCUSSIONS

6.1 Results

As a means of validating our algorithm, simulations were conducted both on newly developed workspaces and some selected workspaces developed by earlier workers. We implemented our simulation using MobotSim software, a customized mobile robot simulator, on an Intel Pentium®4, 2.4GHz, 1GB of RAM.

Figs. 15a,16a,17a,18a,19a,31a and 32a are mazes originally developed in this work to demonstrate the completeness and generalized applicability of our algorithm in complex static obstacle domains. The corresponding robot trajectories for each of these developed mazes is as shown in Figs. 15b, 16b,17b,18b,19b, 31b – 31e and 32b-32e respectively. We may observe here that the developed workspaces included both obstacles with regular and irregular shapes so as to simulate a more generalized model of the real world.

Comparison with the results of other researchers are as shown in Figs. 20a through Figs. 30c. This exercise was carried out by reproducing their workspaces in terms of the shape, size and position of the obstacles in the respective workspaces. However, it was not always possible to reproduce these selected workspaces exactly to scale in our work. This is mostly due to non-availability of detailed information such as the obstacle dimensions and in some cases, the workspace dimensions. Nevertheless, our priority is to be able to reproduce such workspace environment to a justifiable degree to carry out our validation.

We further ensured that the workspaces selected for this exercise are mostly those that ordinarily look complex or where earlier authors recorded difficulties in navigating the robot to the goal state. Our objective in doing this is to be able to validate the overall effectiveness of our new concept relative to some existing models.

Firstly, we recall the work of Yahja et al. (1998) where three different cell decomposition techniques viz: the approximate technique, the quad-tree and the framed quad-tree were validated on a common sample workspace as initially shown in Fig. 20a. The framed quad-tree was proposed as a result of some inherent limitations associated with the conventional quad-tree approach. From the workspaces as shown in Figs. 20b,c,d and e the resulting trajectories show that our proposed technique is as effective as the framed-quadtree technique. The algorithmic complexity of the framed-quadtree could be a little bit demanding following from the fact that it is hinged on the D* algorithm which is an extension of the effective but greedy A* algorithm.

Furthermore, Fig. 21a shows a problem workspace as developed by Geraerts and Overmars (2004). The algorithmic complexity of this technique lies in the fact that it involves the integration of two planner types namely: the Probabilistic Road Map technique (PRM) and the Potential Field Method (PFM). Firstly possible navigation paths are identified after which the optimal path is selected to be then followed by the actual navigation. Our proposed algorithmic simulation result shown in Fig.21c is quite comparable to the result obtained by Geraerts and Overmars in terms of effectiveness and efficiency.

Also, shown in Fig. 22a is a complex maze as developed by Geraerts and Overmars (2004). The algorithmic complexity here is just as mentioned in respect of Fig. 21 above. The resulting trajectories as shown in Figs.22b and 22c demonstrate the effectiveness of our algorithm.

In addition, Geraerts and Overmars developed a 3d-like workspace shown in Fig.23a. Three different planner types viz: the Corridor Map Method, the A* algorithm and Probabilistic Road Map method were validated here. The respective trajectories developed shows that Fig. 23c is much demanding in terms of algorithmic complexity. Next is Fig. 23d which generates low quality paths due to the probabilistic nature i.e. paths that represent many unnecessary motions or do not obey user defined criteria Kim et. al (1992) and Song et al. (2001). The path generated by our proposed algorithm as shown in Fig. 23e is efficiently driven and also effective as that displayed in Fig. 23b.

The cellular Automata approach was deployed by Behring et al. (2000) in the navigation problem of Fig.24a with the authors' solution shown in Fig. 24b and ours shown in Fig. 24c in a similar workspace. The concept is dominantly hinged on the theory of configuration space and involves the growing of obstacles. It is observed that the actual execution time of the algorithm depends on several factors. The most important of which are the frame rate of the camera and the data transfer rate by the parallel port that limits the transfer velocity. The authors however affirmed that further study involving many more benchmarking examples are necessary.

The computational cost for the cellular automata is proportional to the number of updates executed on the cells. Sometimes a cell has more than one neighbor with the same Manhattan distance to the goal. To follow always the steepest descend of the function may not work as there could be cases where the gradient may have the same value in several directions. In order to select a reasonable good path, heuristics are applied among the neighbors of a cell:

Table 1: Success Rate for typical Sample Workspaces, *Sedighi et. al. (2004)*

Search Space	Success Rate (%)			
	Geisler's GA	Hermami's GA	Sedighi's GA	**PROPOSED (HVFF)**
SPSet01	100	93.3	93.3	100
SPSet02	0.00	86.7	100	100
SPSet03	73	86.7	100	100
SPSet04	53	80	100	100
SPSet05	0.00	100	100	100
SPSet06	0.00	20	100	100
SPSet07	0.00	0.00	86.7	100
SPSet08	0.00	0.00	73.3	100

Furthermore, we recall the work of Sedighi et al. (2004). where their genetic algorithm was compared with those of earlier workers for specific work spaces as reproduced in Table 1. In particular it was reported that for search spaces SPSet 06, 07 and 08 the solutions of both Geisler and Harmani had poor performance in relation to the genetic algorithm of Sedighi et al. for which success rates (i.e ability to reach the desired goal) of 100%, 86.7% and 73.3% were recorded respectively.

Fig. 25b and 26b are sample workspaces SPSet07 and SPSet08 as shown in Table 1. From this table as shown in section V of Sedighi et al. (2004) it could be seen that these two workspaces were the most difficult cases considered by Sedighi et al. On the other hand, their GA gave a success rate of 93.3% for SPSet01, 100% in work spaces SPSet02- SPSet06 while SPSet07 and SPSet08 gave a success rate of 86.7% and 73.3% respectively. For the workspaces SPSet07 and SPSet08, Figs. 25a and 26a represent the initial positions of the robot relative to the goal state before navigation while Fig. 25c and 26c are navigation results obtained from current research. The trajectories in Figs. 25b, 25c, 26b and 26c clearly show that our model is efficient. It is also to be noted that where the solution is hinged on evolutionary algorithm, the evolving set of paths for different trials may not be consistent. Our concept does not vary the path over trials as long as the conditions of the workspace remain constant.

Similar comparison has been carried out in respect of the recent work of Zou and Zhu (2003) where the algorithm is not based on genetic programming but rather on the use of intermediate local targets. Two particular workspaces of interest were selected as shown in Figs. 27a and 29a respectively. In respect of Fig. 27a we find that the workspace is such that the robot is almost completely encircled by obstacles, ab initio, in an enclosure. Fig. 29a on the other hand presents a complex maze-like workspace with the position of either the robot or goal state being arbitrarily located.

We also observe from Fig. 27b that while the robot in the Zou and Zhu (2003) scheme is eventually able to get to the target it is first bounced around a few times within the enclosure before it finds its way. In comparison our scheme on the other hand as shown in Fig. 27c is able to avoid such wandering and is set on track of the goal right from the onset thereby saving time. Fig. 28a presents the situation for the converse to the problem in Fig. 27a; i.e. in this case it is the target that is virtually enclosed by obstacles and for which Fig. 28b presents the solution navigation path. In respect of the maze-like environment shown in Fig. 29a both schemes appear equally efficient at navigating their way through the work to the target without any wandering.

The environment in Fig. 30a is one that simulates narrow passages within a cluster of obstacles and was the problem for which Chengqing et. al. introduced the concept of virtual obstacle as a strategy for recovery from local minima in potential field driven navigation algorithms. Here we find that whereas the robot is bounced around between obstacles and oscillates as shown in Fig. 30b before ultimately finding its way to the target, the HVFF scheme as shown in Fig. 30c allows the robot to chart a direct minimum path trajectory to the goal without any of such handicaps. Figs. 31(a-g) is a set of robot navigation frames representing a special case of dynamic obstacles formed into concave shape. While Fig. 31a represents the problem frame, Fig. 31(b-f) are the different intermediary frames showing how the virtual concepts of our model aids the robot from being trapped in the concave trap formed. Fig. 31g show the robot at the target position. Figs. 32(a-e) is a set of intermediary frames for a case demonstration of our real robot while navigating to its target point. The trajectory of navigation was redefined by the prompting of the virtual goal bringing it out of the trap zone. This demonstrated case is for an unknown environment i.e robot has no knowledge of workspace prior to navigation.

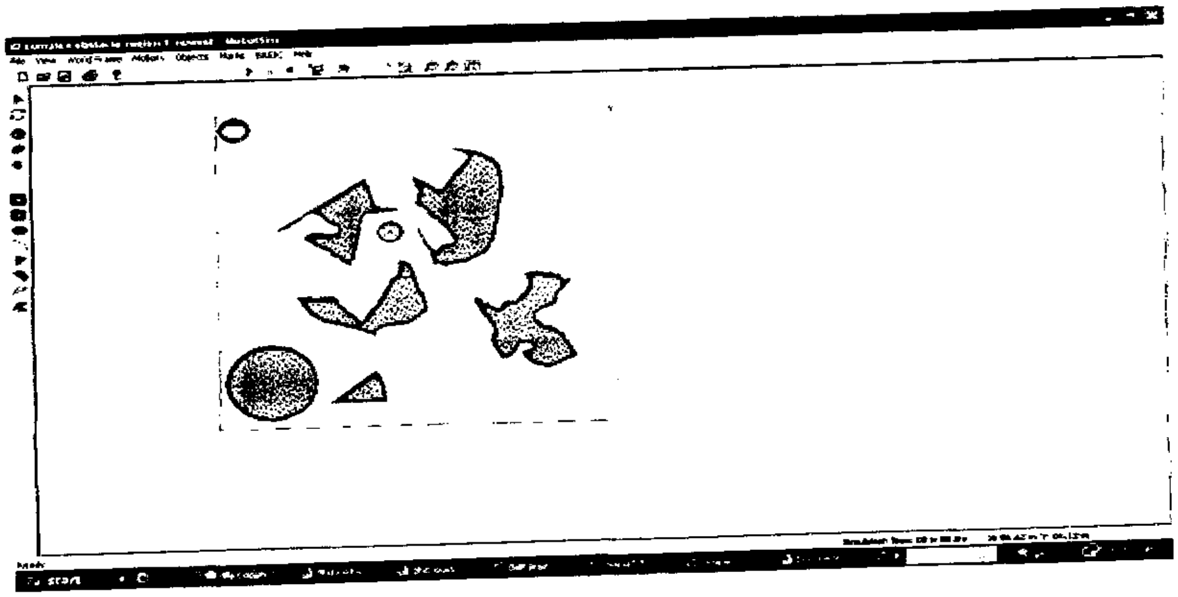


Figure 15a: Developed Problem Sample Space01

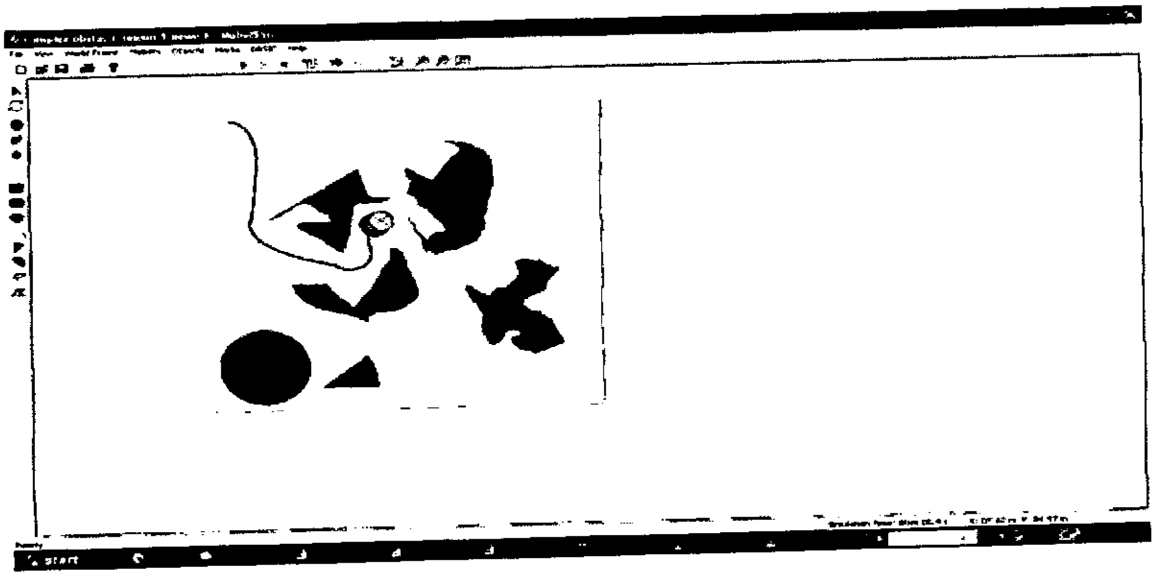


Figure 15b: Solution for developed Sample Space01

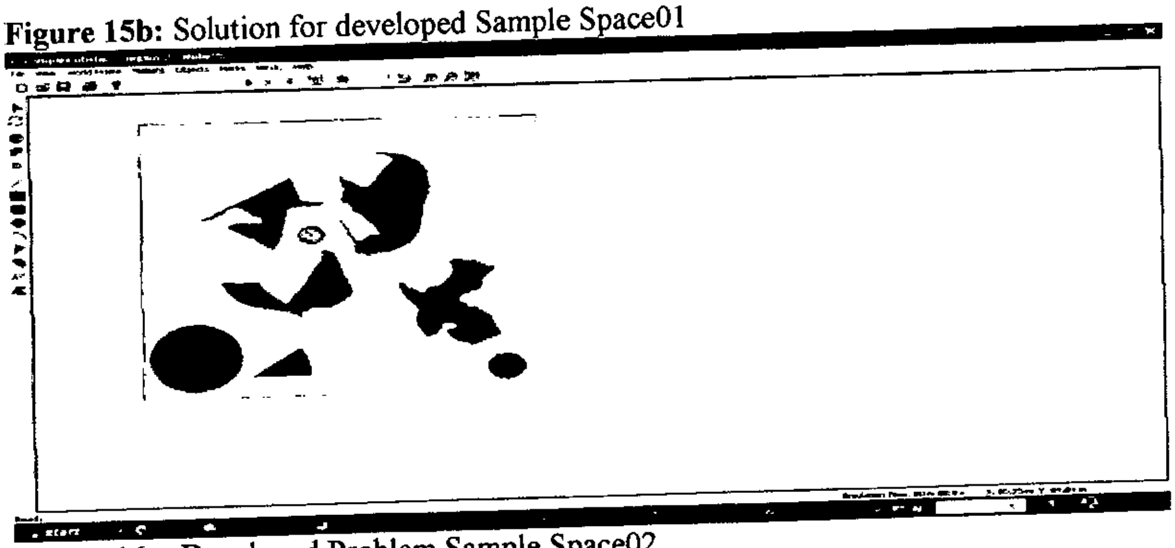


Figure 16a: Developed Problem Sample Space02

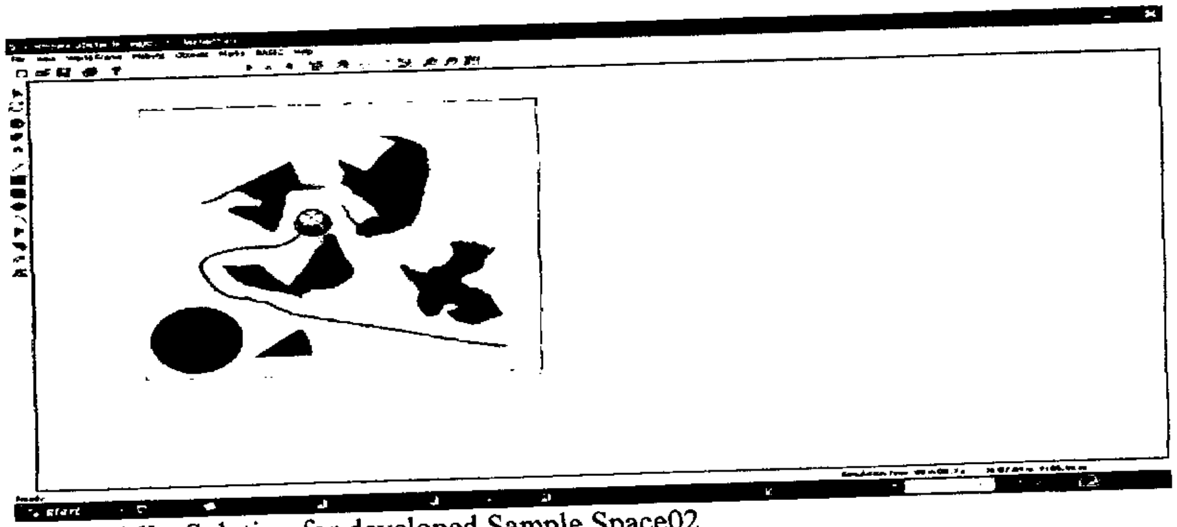


Figure 16b: Solution for developed Sample Space02

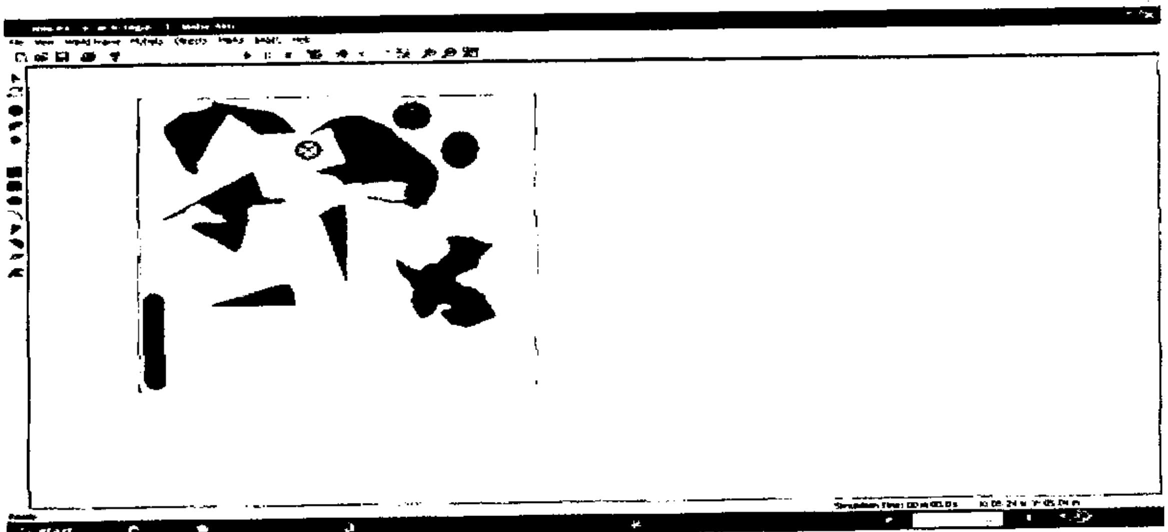


Figure 17a: Developed Problem Sample Space03

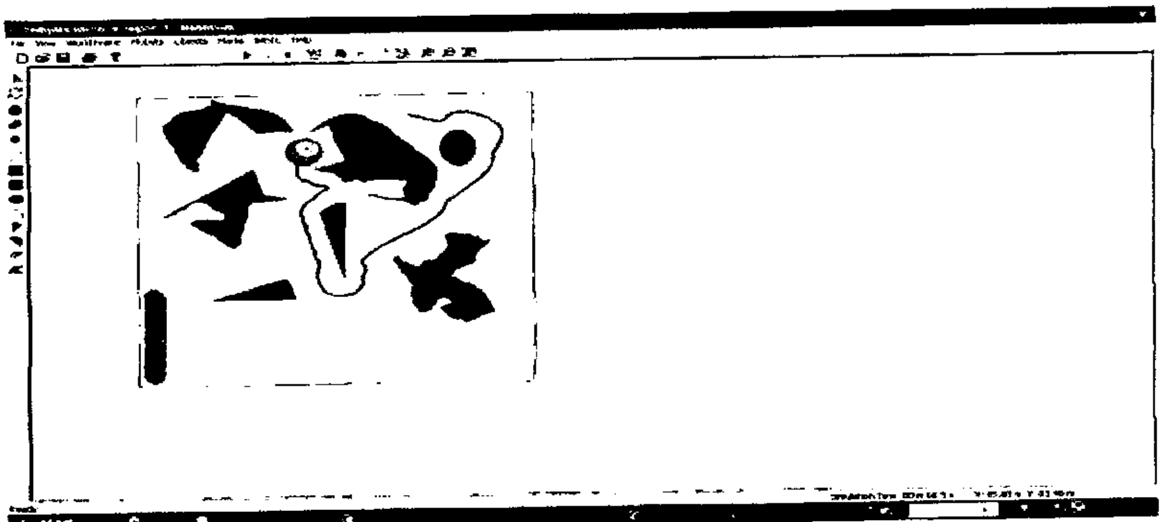


Figure 17b: Solution for developed Sample Space03

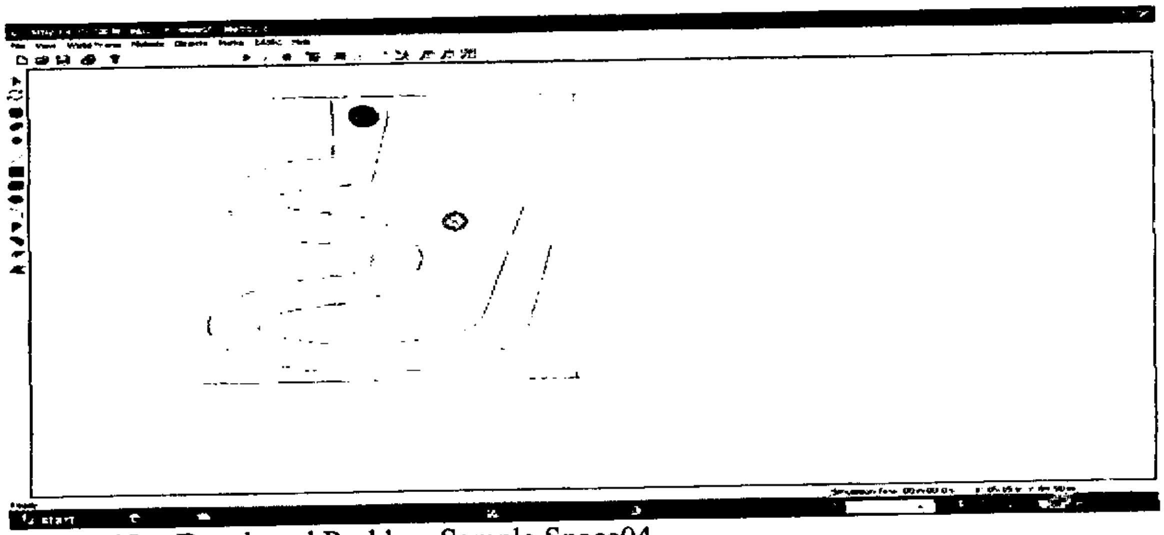


Figure 18a: Developed Problem Sample Space04

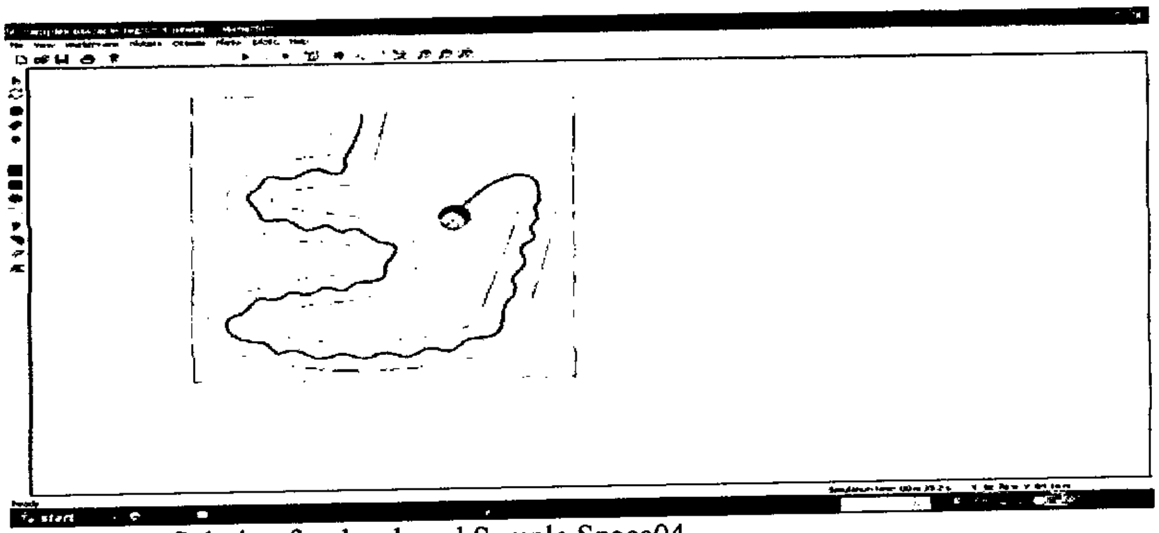


Figure 18b: Solution for developed Sample Space04

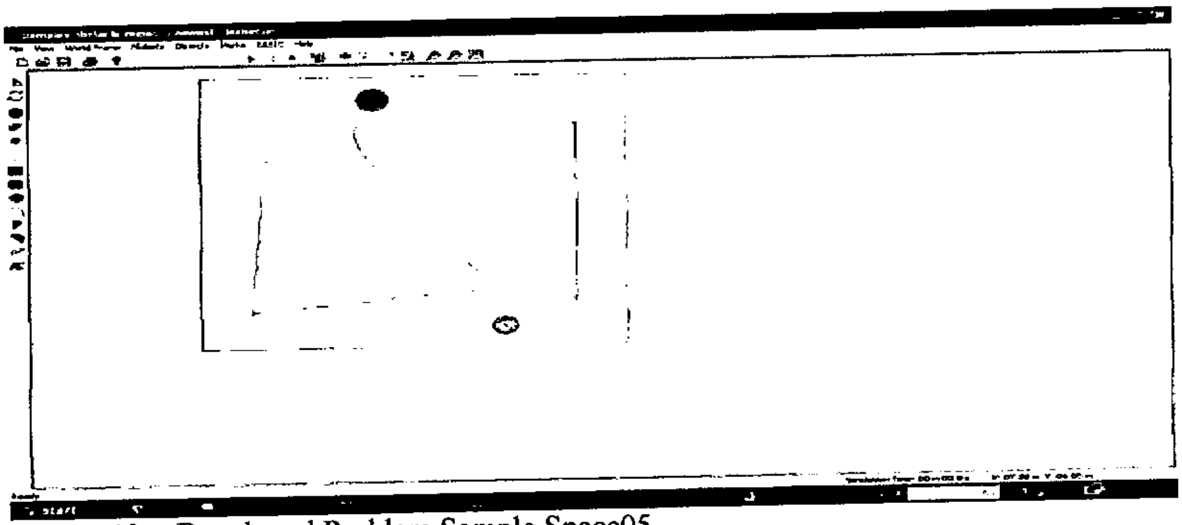


Figure 19a: Developed Problem Sample Space05

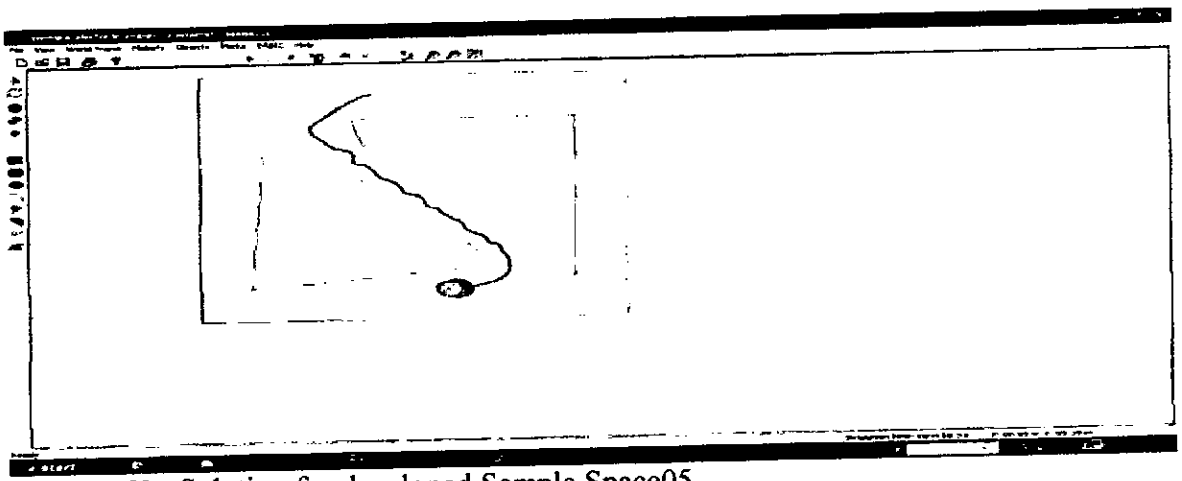


Figure 19b: Solution for developed Sample Space05

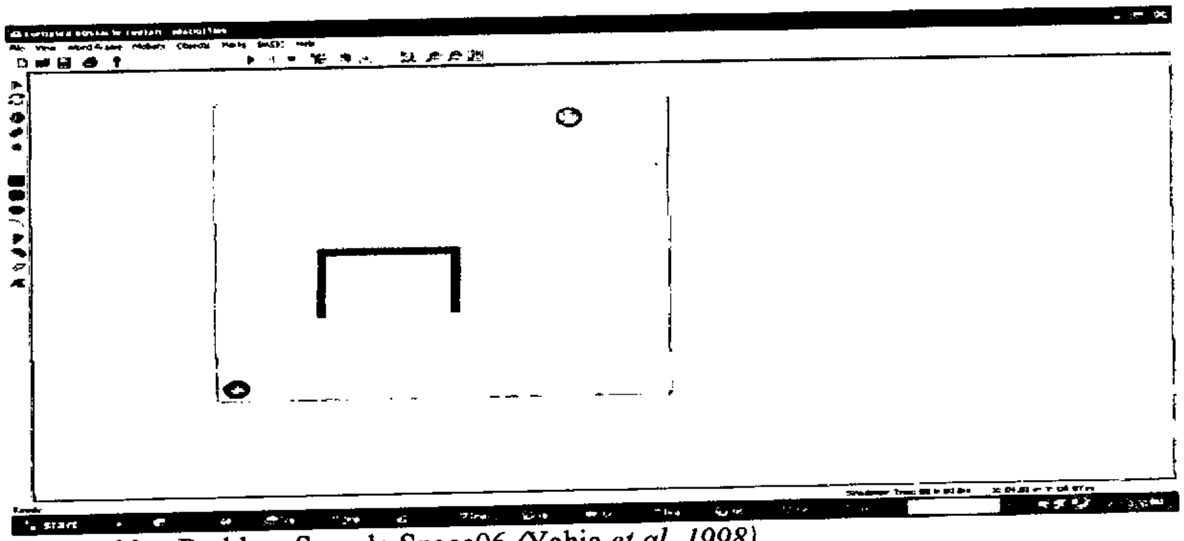
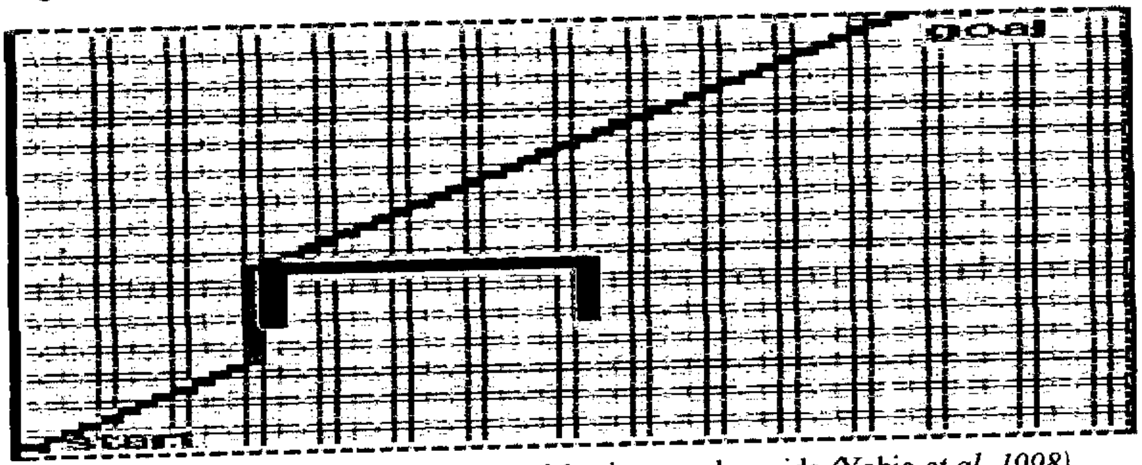
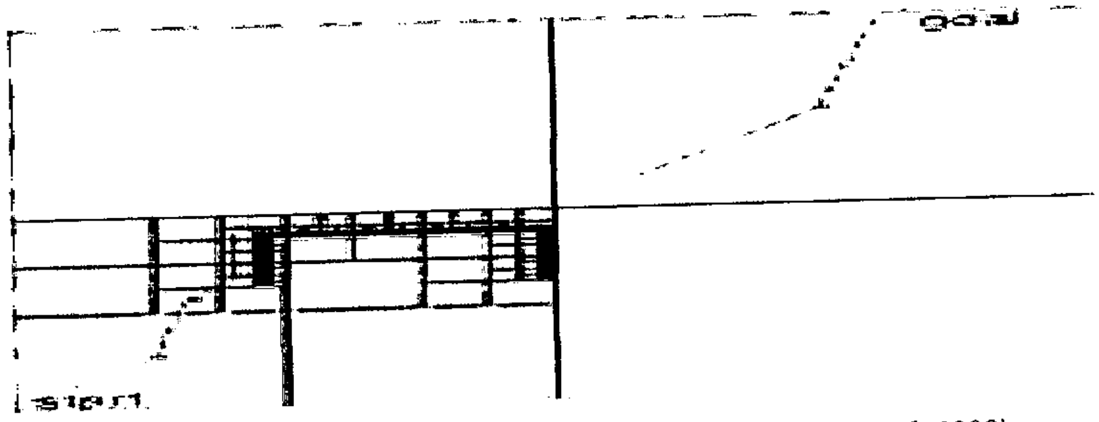


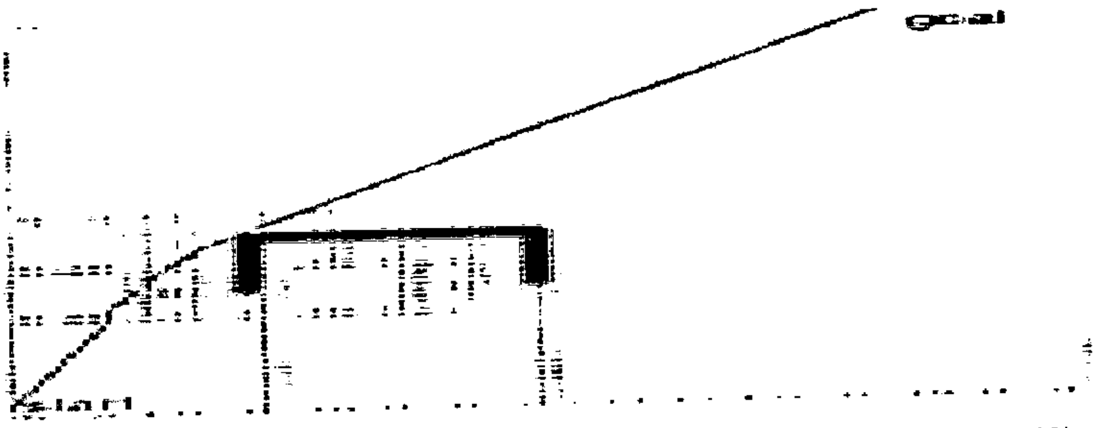
Figure 20a: Problem Sample Space06 (Yahja *et al. 1998)*



Figures 20b: Solution for Sample Space06 using regular grids (Yahja *et al. 1998)*



Figures 20c: Solution for Sample Space06 using quadtree (Yahja *et al. 1998)*



Figures 20d: Solution for Sample Space06 using framed-quadtre *(Yahja et al. 1998)*

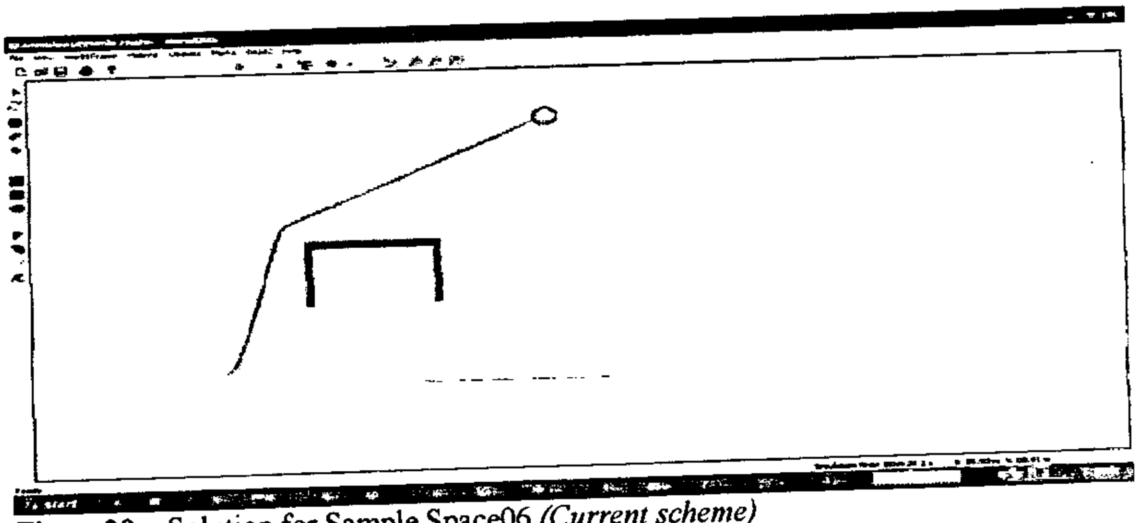


Figure20e: Solution for Sample Space06 *(Current scheme)*

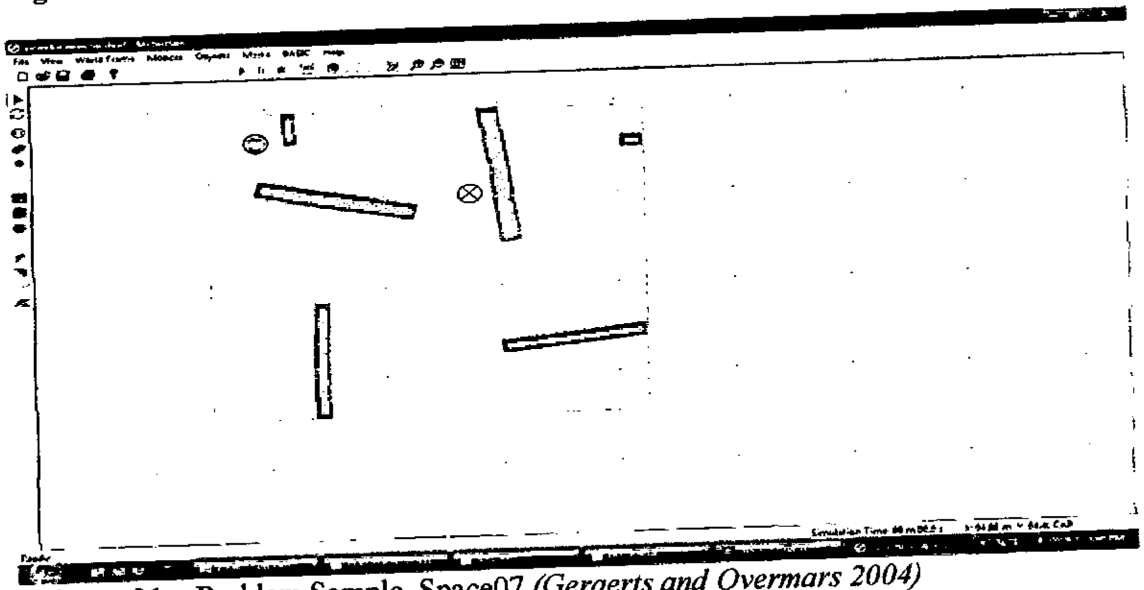


Figure 21a: Problem Sample Space07 *(Geraerts and Overmars 2004)*

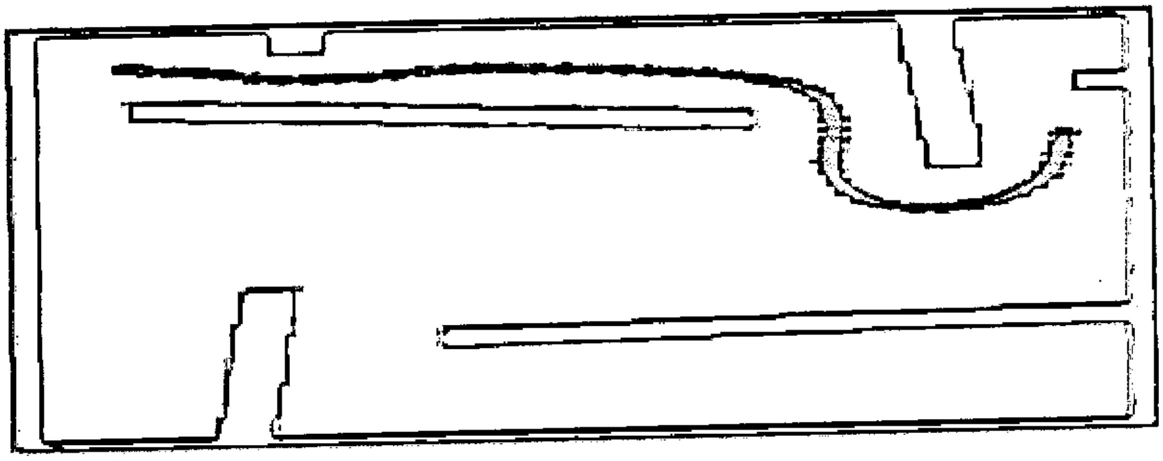


Figure 21b: Solution for Sample Space07 *(Geraerts and Overmars 2004)*

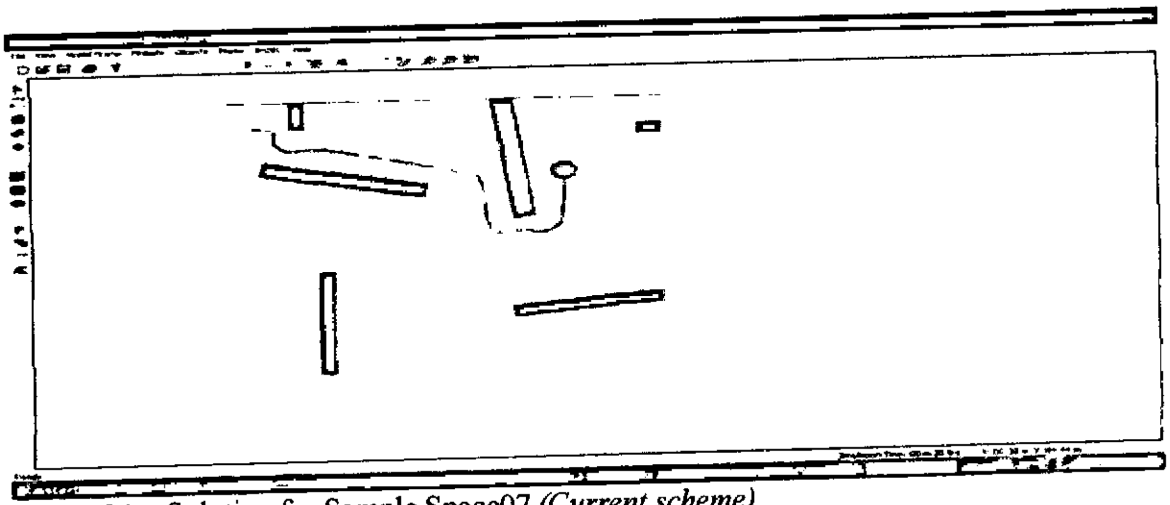


Figure21c: Solution for Sample Space07 *(Current scheme)*

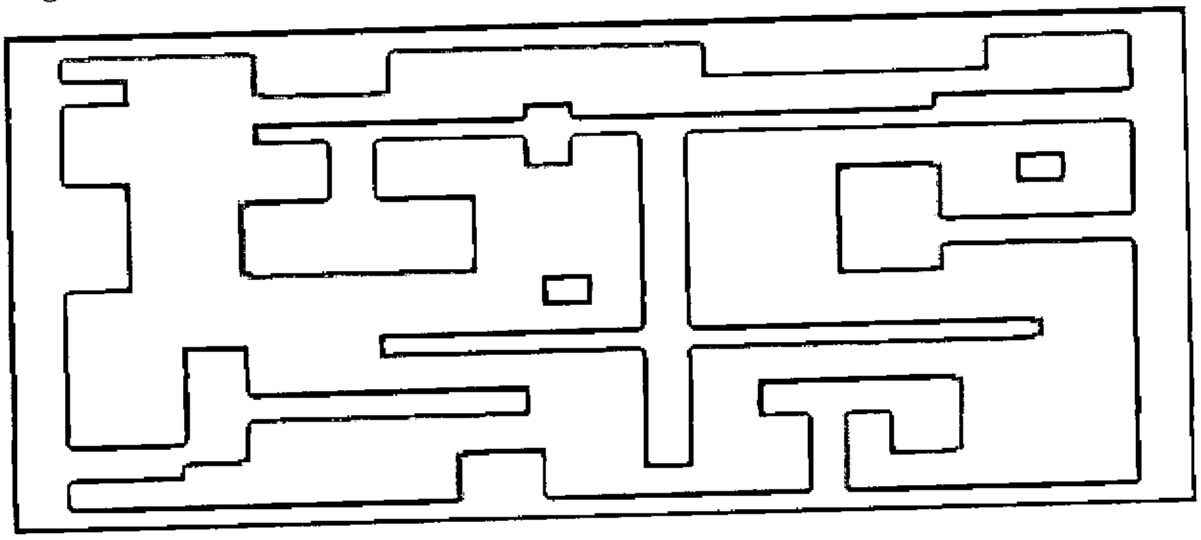


Figure 22a: Problem Sample Space08 *(Geraerts and Overmars 2004)*

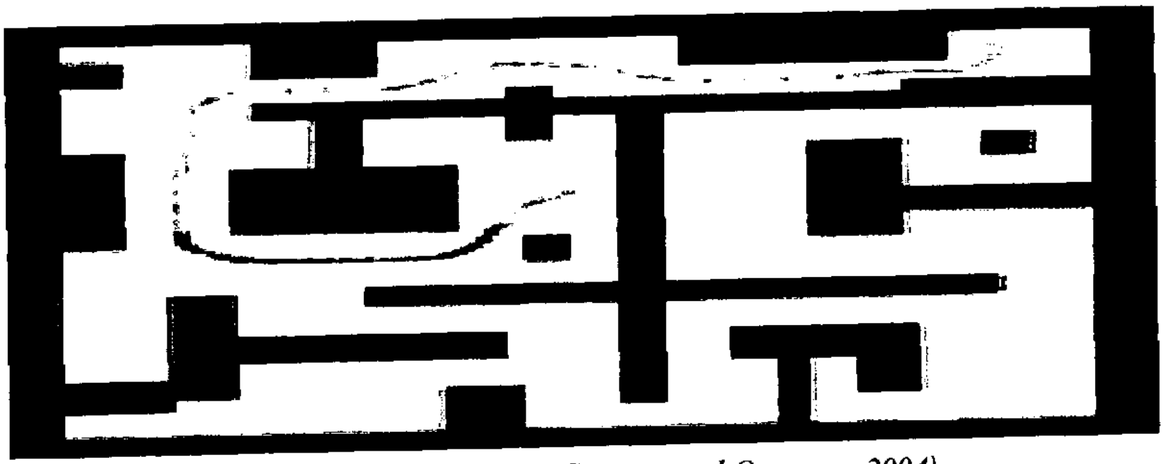


Figure 22b: Solution for Sample Space08 *(Geraerts and Overmars 2004)*

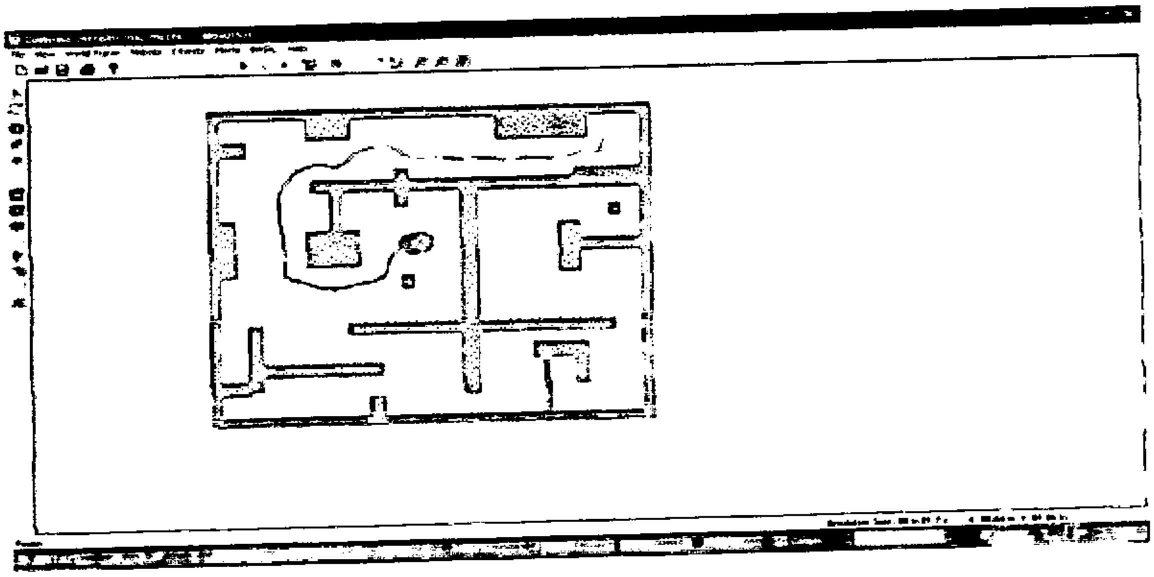


Figure22c: Solution for Sample Space08 *(Current scheme)*

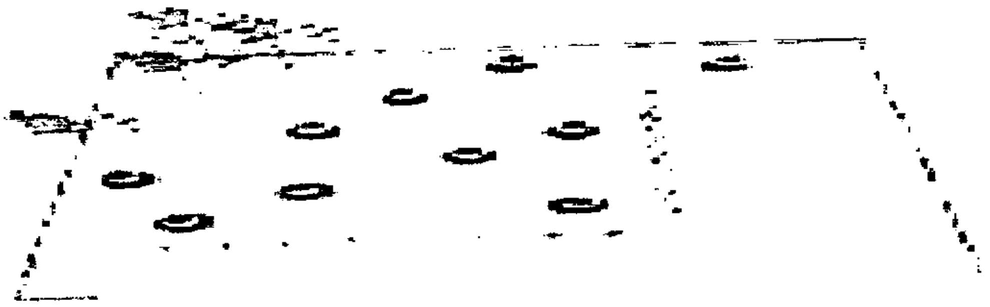


Figure 23a: Problem Sample Space09 *(Geraerts and Overmars 2004)*

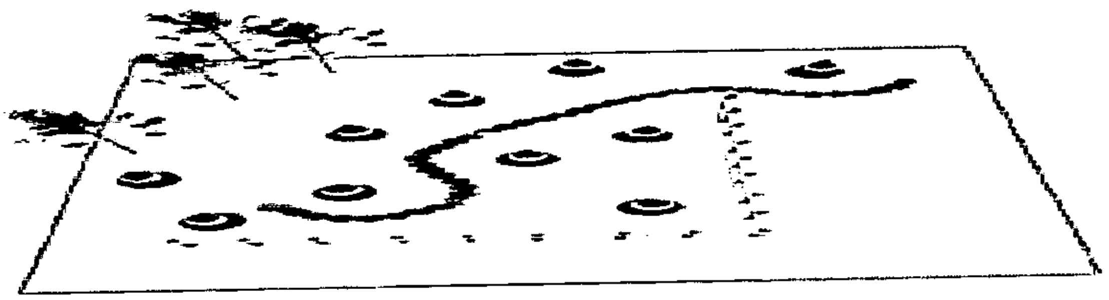


Figure 23b: Solution for Sample Space09 using CMM *(Geraerts and Overmars 2004)*

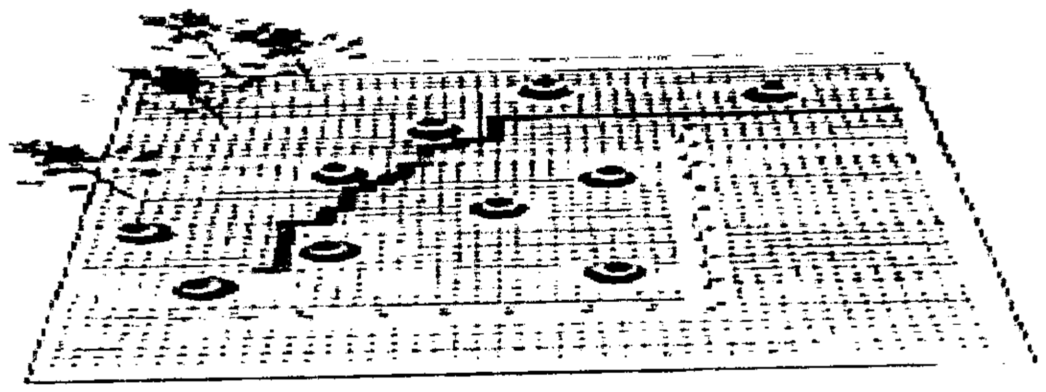


Figure 23c: Solution for Sample Space09 using A* *(Geraerts and Overmars 2004)*

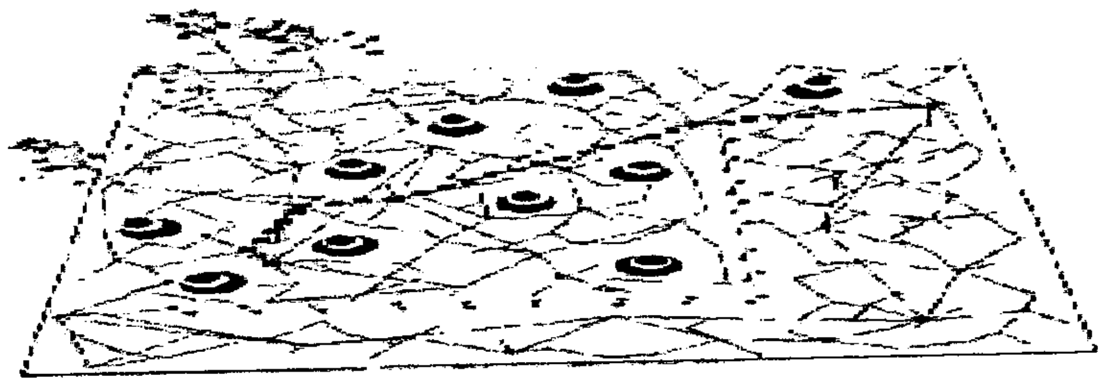


Figure 23d: Solution for Sample Space09 using PRM *(Geraerts and Overmars 2004)*

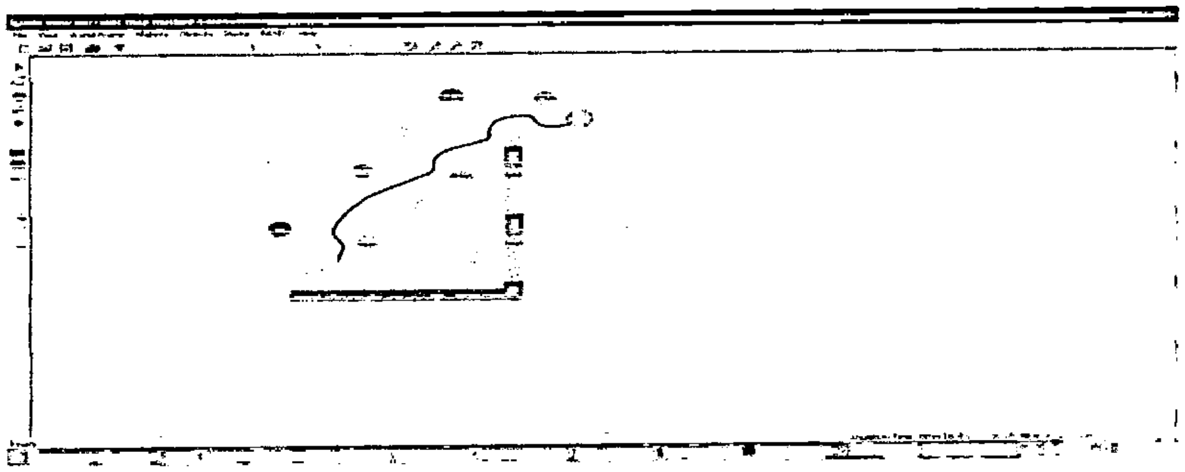


Figure23e: Solution for Sample Space09 *(Current scheme)*

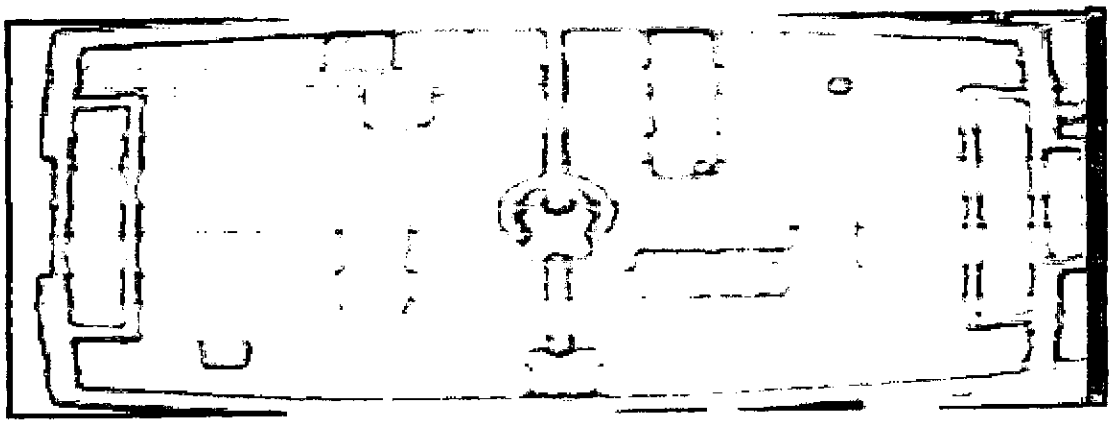


Figure 24a: Problem Sample Space10 (Behring *et al. 2000)*

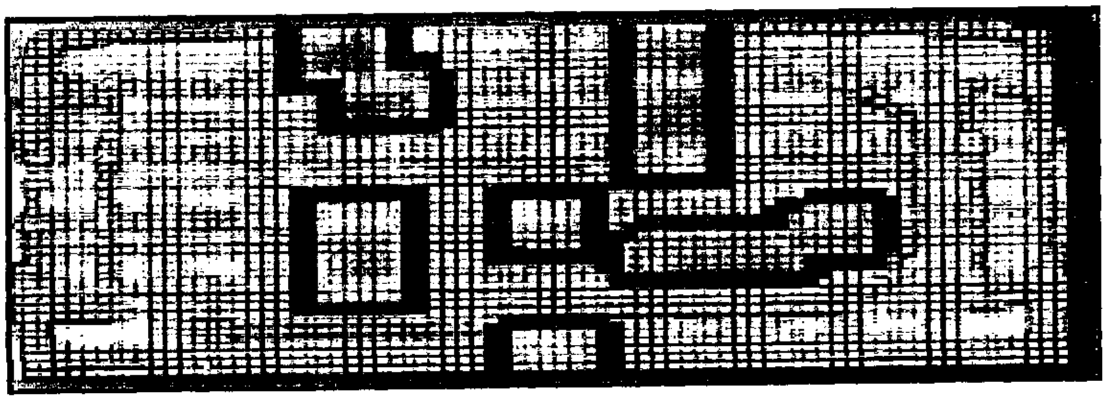


Figure 24b: Solution for Sample Space10 (Behring *et al. 2000)*

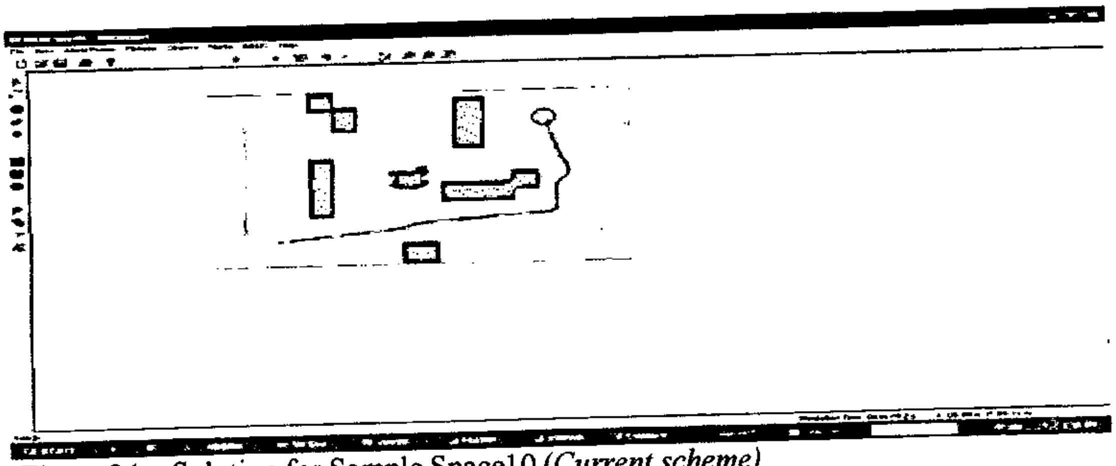


Figure24c: Solution for Sample Space10 (*Current scheme)*

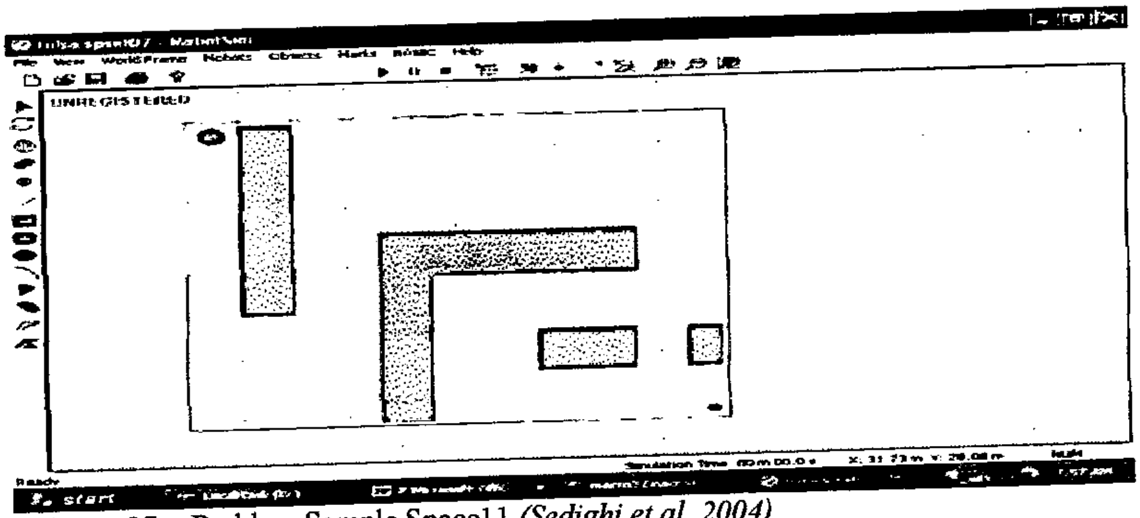


Figure 25a: Problem Sample Space11 *(Sedighi et al. 2004)*

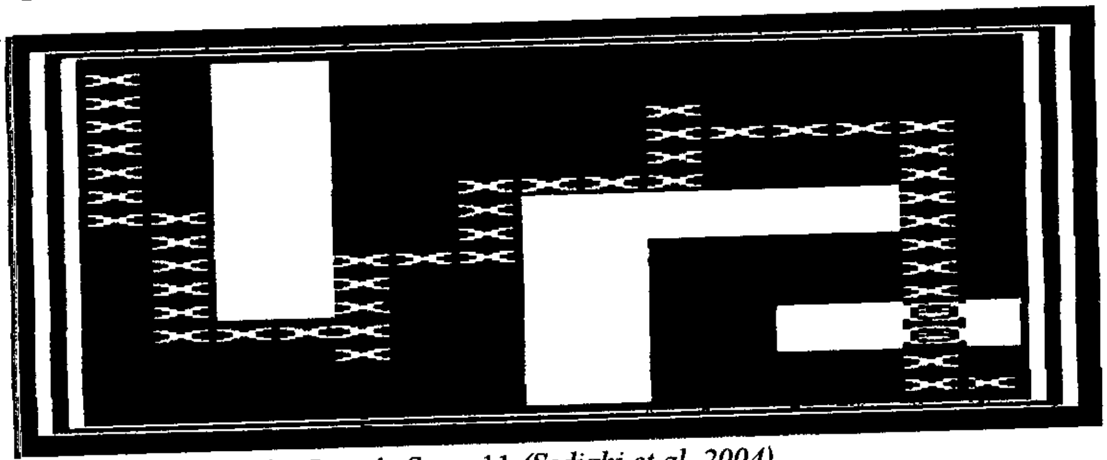


Figure 25b: Solution for Sample Space11 *(Sedighi et al. 2004)*

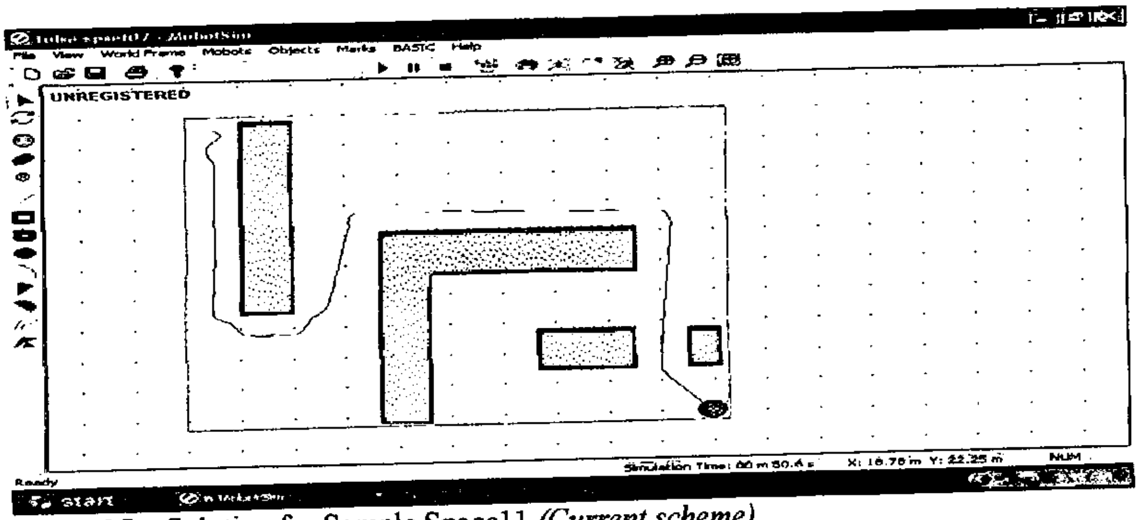


Figure25c: Solution for Sample Space11 *(Current scheme)*

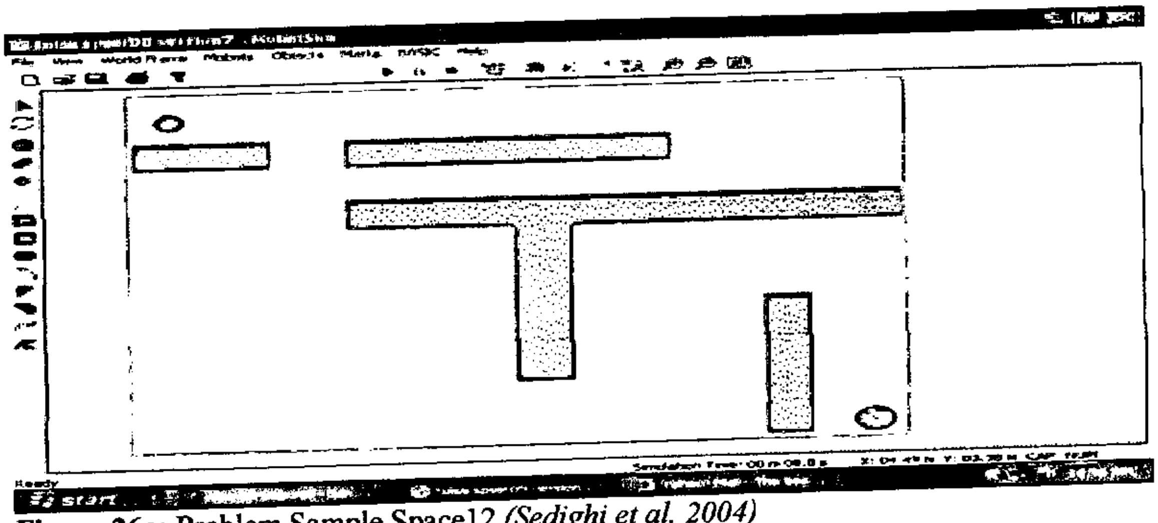


Figure 26a: Problem Sample Space12 *(Sedighi et al. 2004)*

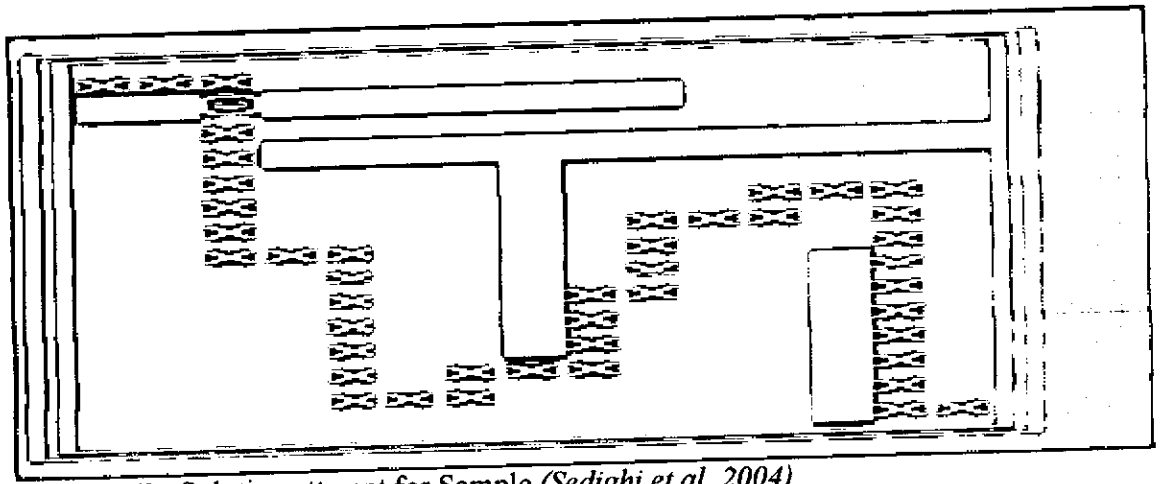


Figure 26b: Solution attempt for Sample *(Sedighi et al. 2004)*

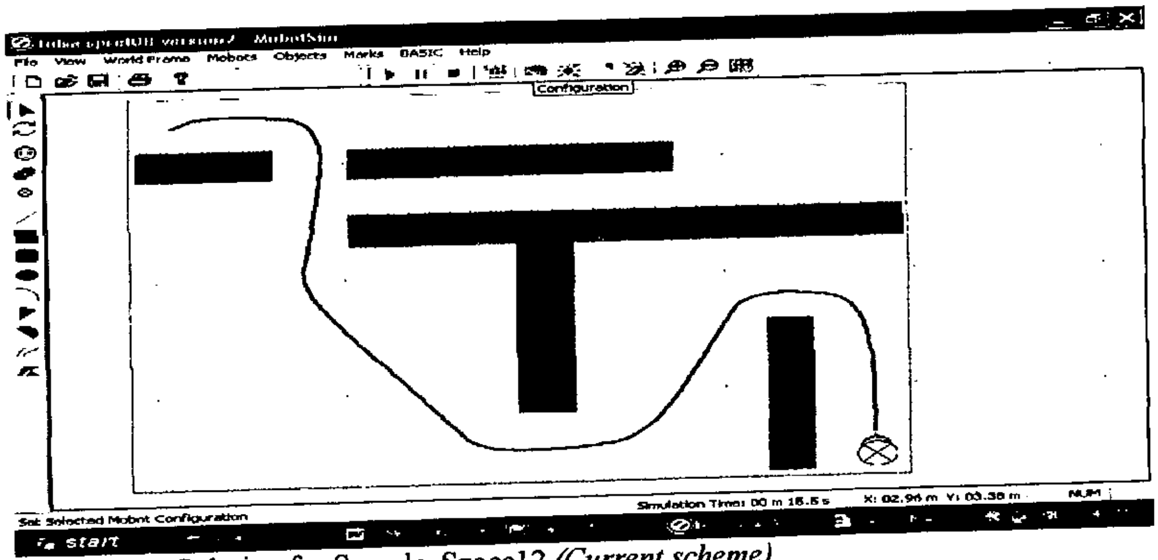


Figure 26c: Solution for Sample Space12 *(Current scheme)*

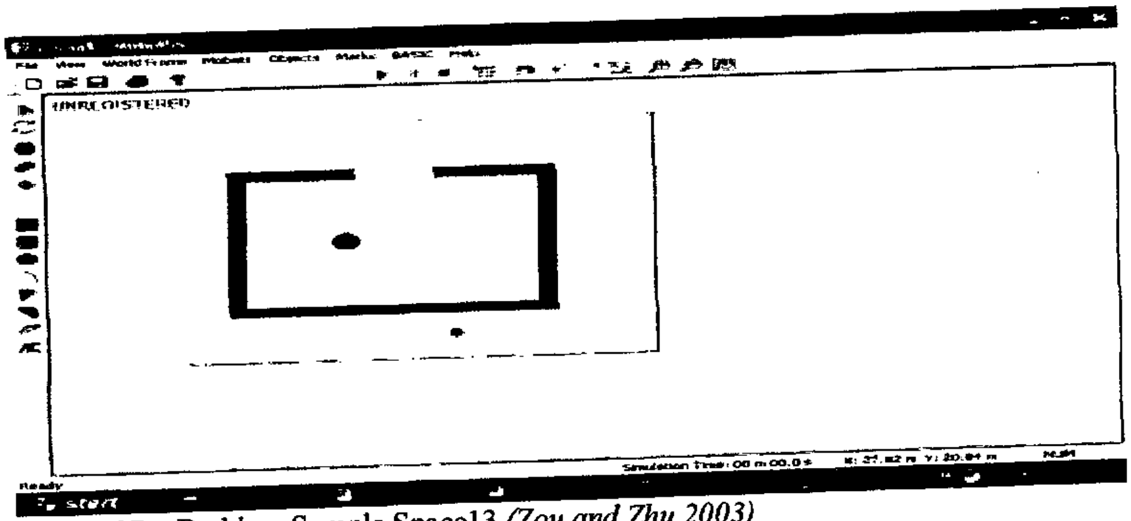


Figure 27a: Problem Sample Space13 *(Zou and Zhu 2003)*

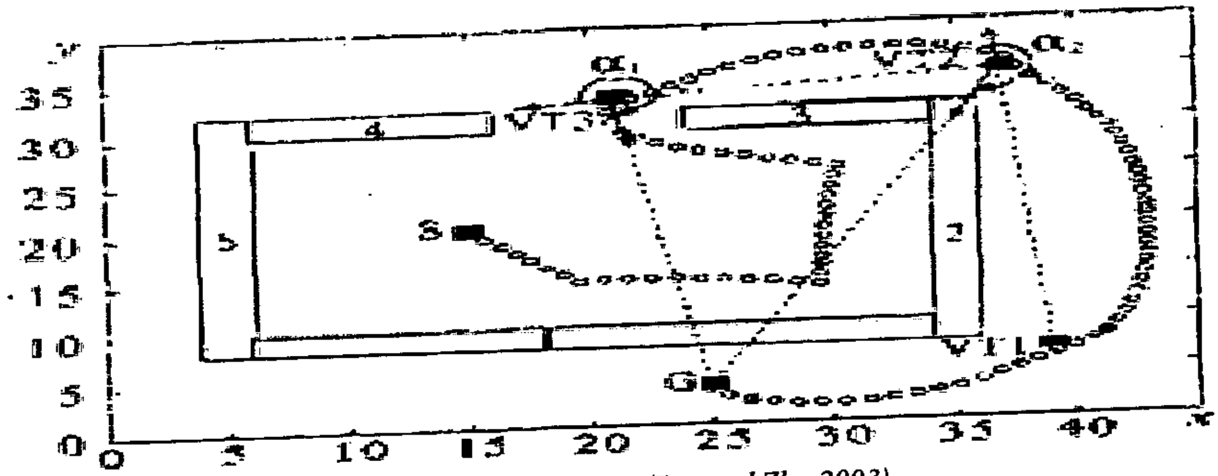


Figure 27b: Solution for Sample Space13 *(Zou and Zhu 2003)*

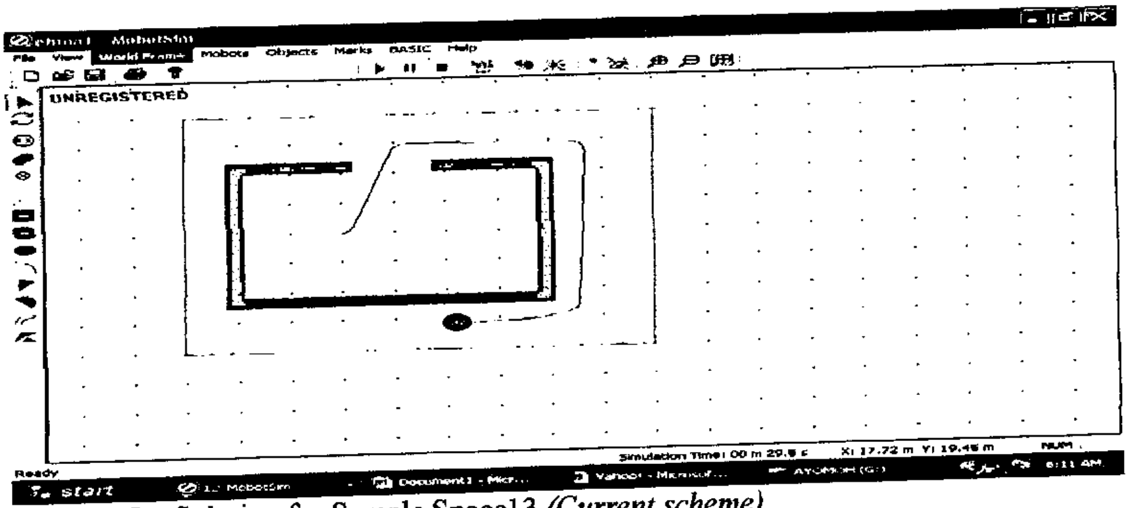


Figure 27c: Solution for Sample Space13 *(Current scheme)*

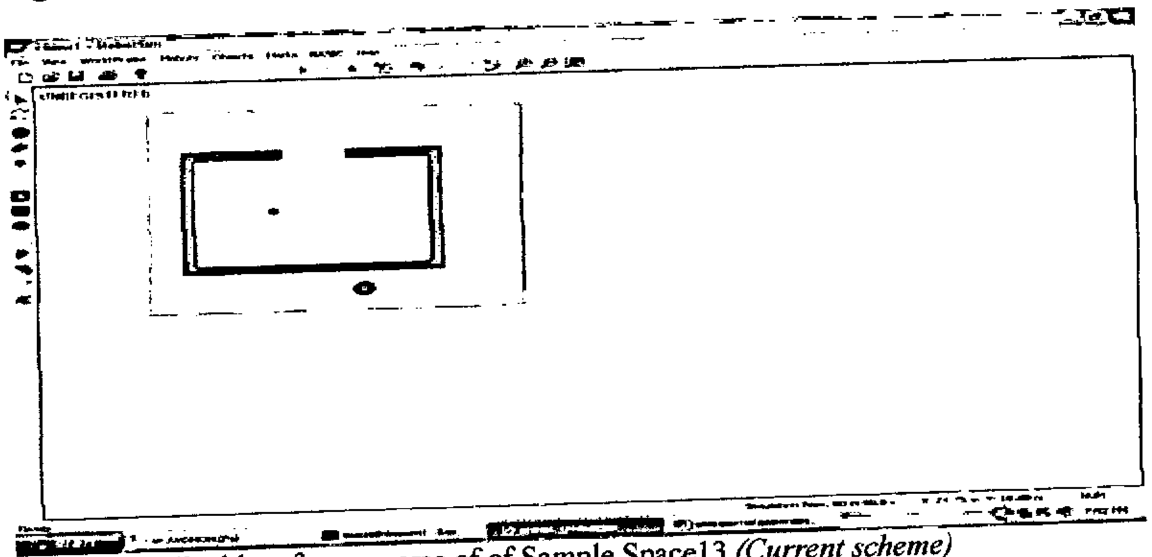


Figure 28a: Problem for converse of of Sample Space13 *(Current scheme)*

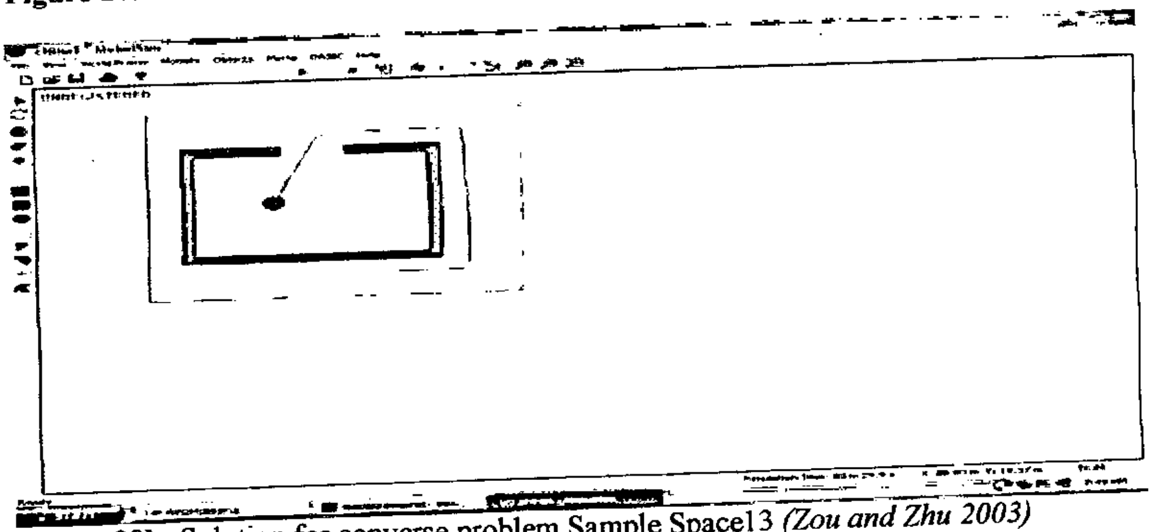


Figure 28b: Solution for converse problem Sample Space13 *(Zou and Zhu 2003)*

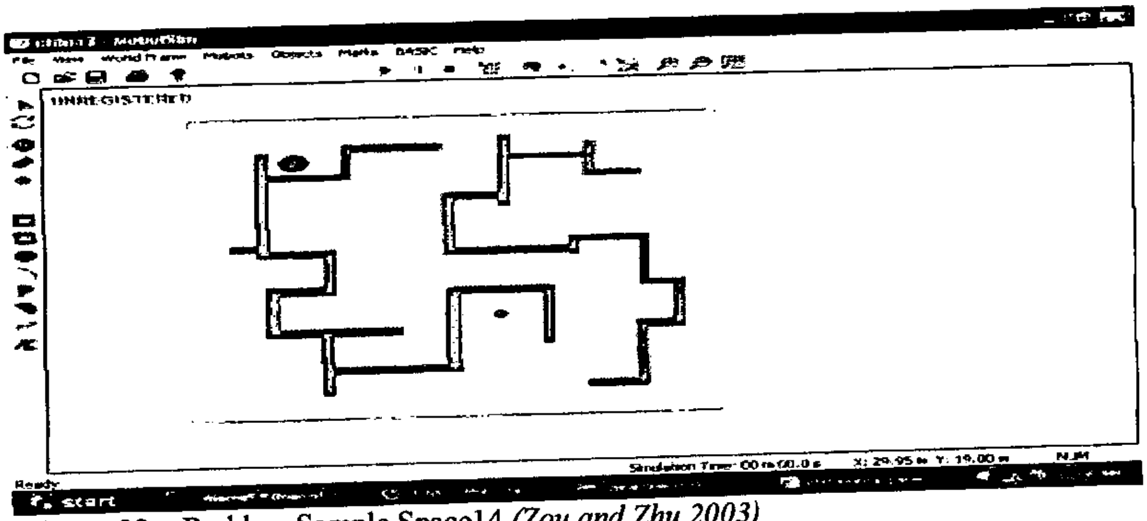


Figure 29a: Problem Sample Space14 *(Zou and Zhu 2003)*

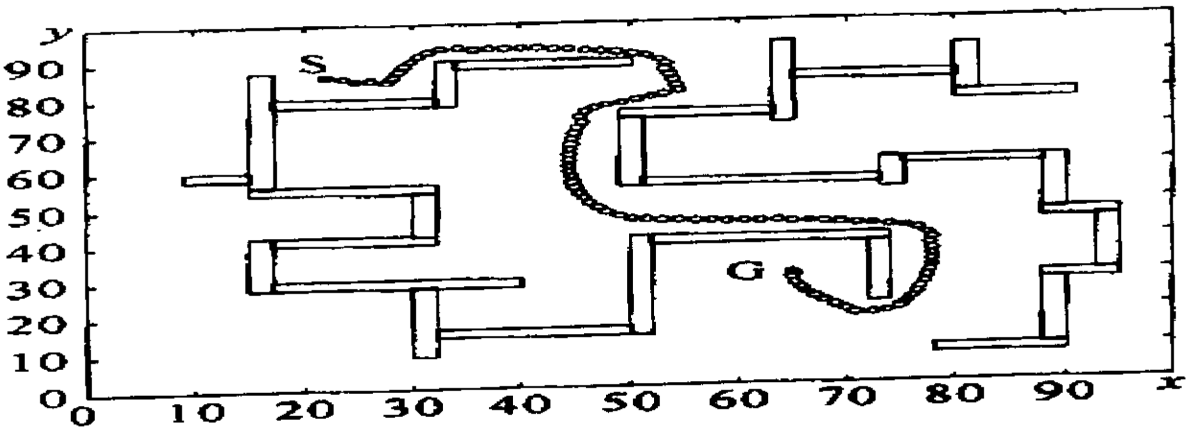


Figure 29b: Solution for Sample Space14 *(Zou and Zhu 2003)*

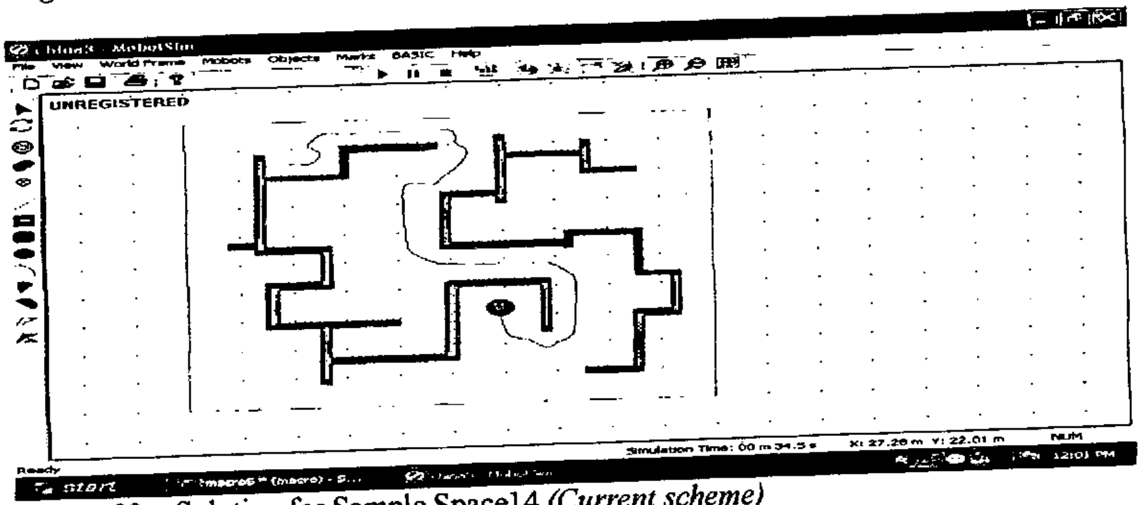


Figure 29c: Solution for Sample Space14 *(Current scheme)*

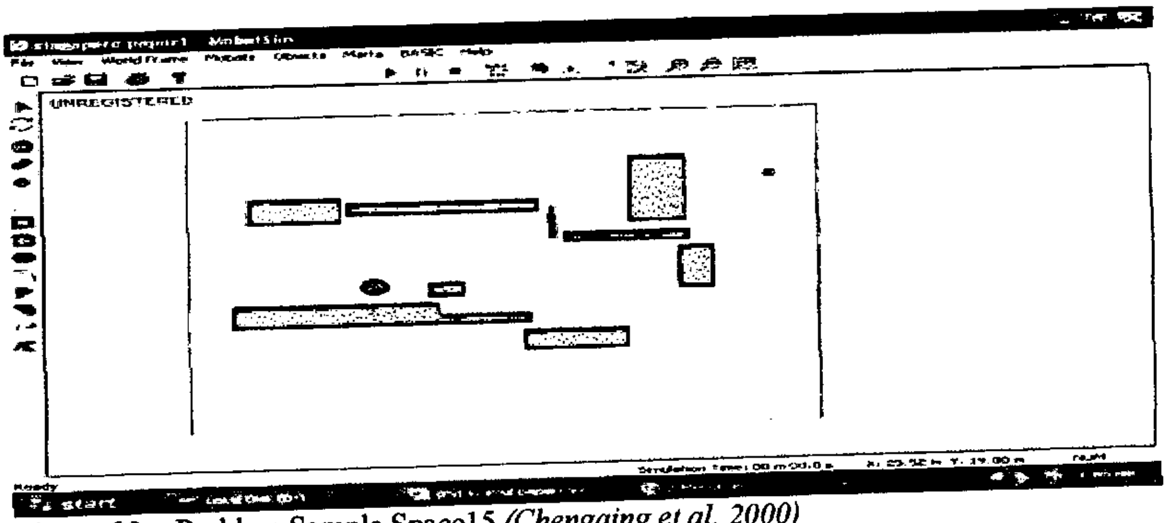


Figure 30a: Problem Sample Space15 *(Chengqing et al. 2000)*

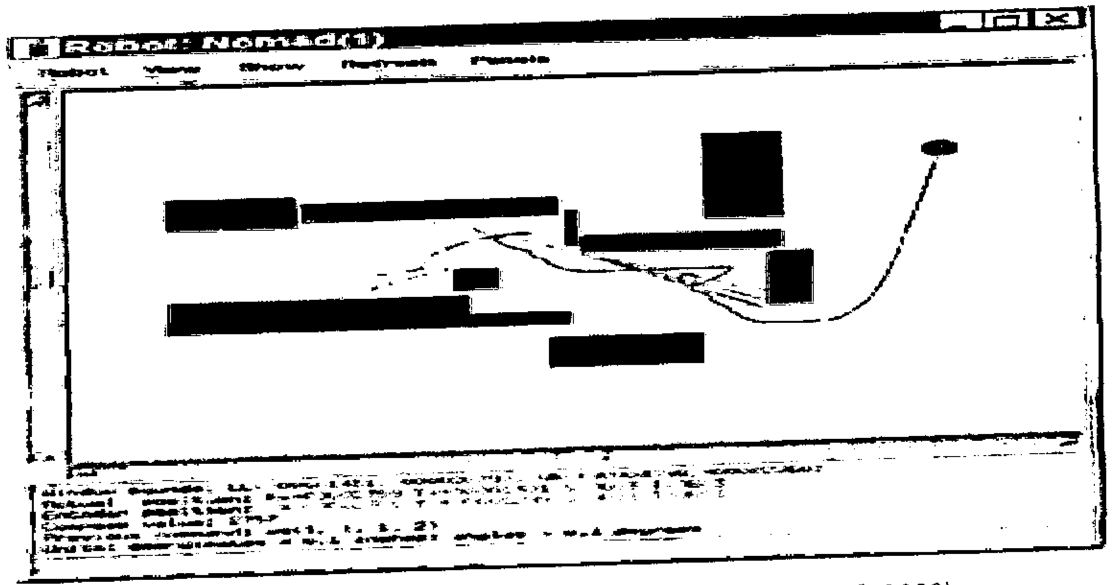


Figure 30b: Solution for Problem Sample Space15 *(Chengqing et al. 2000)*

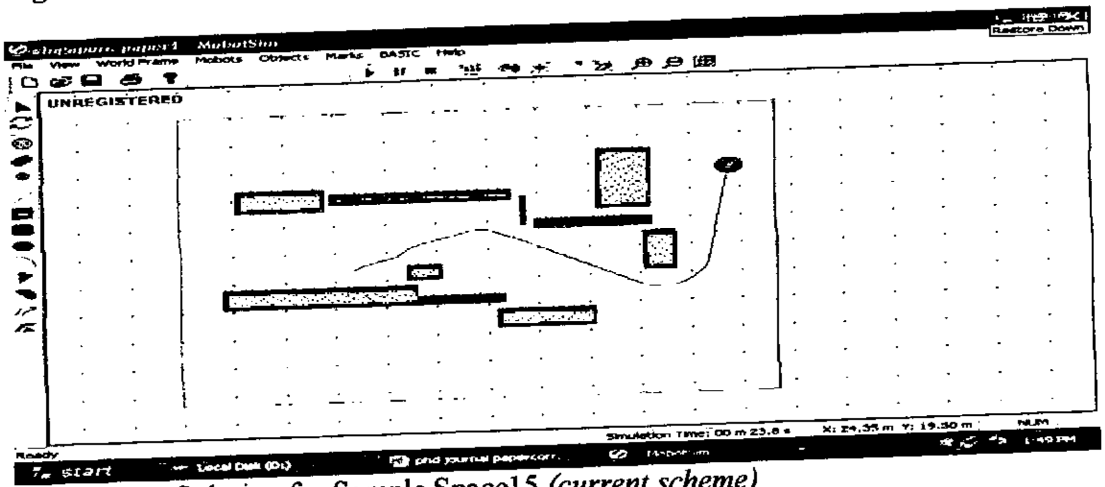


Figure 30c: Solution for Sample Space15 *(current scheme)*

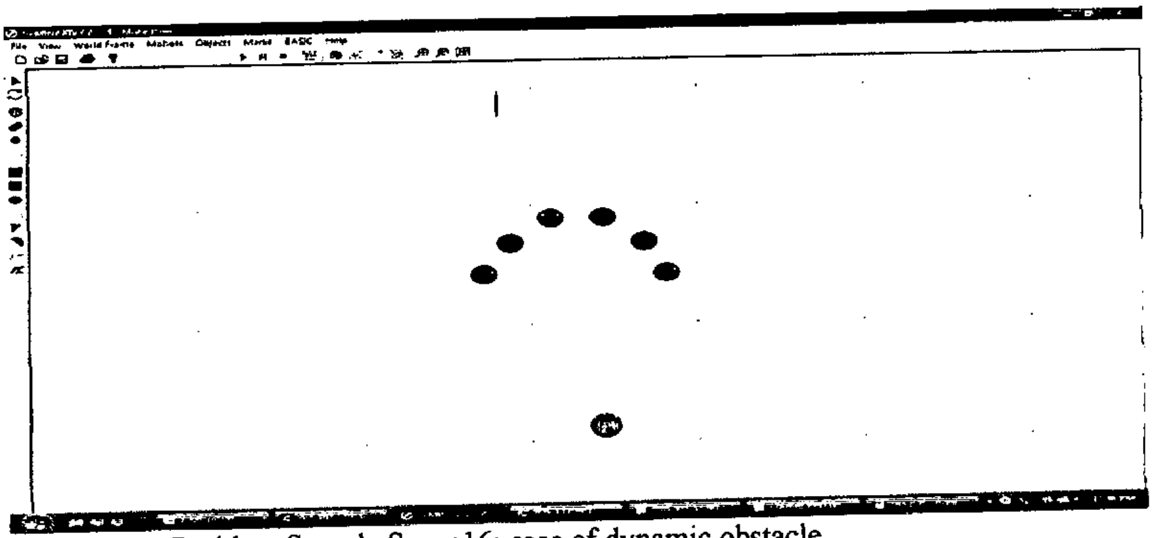


Figure 31a: Problem Sample Space16: case of dynamic obstacle

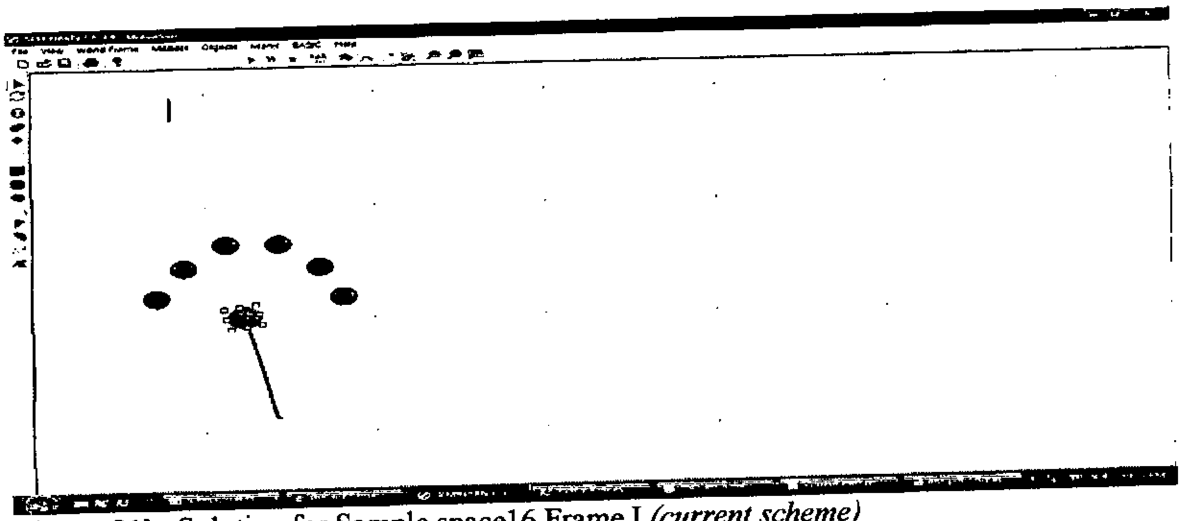


Figure 31b: Solution for Sample space16 Frame I *(current scheme)*

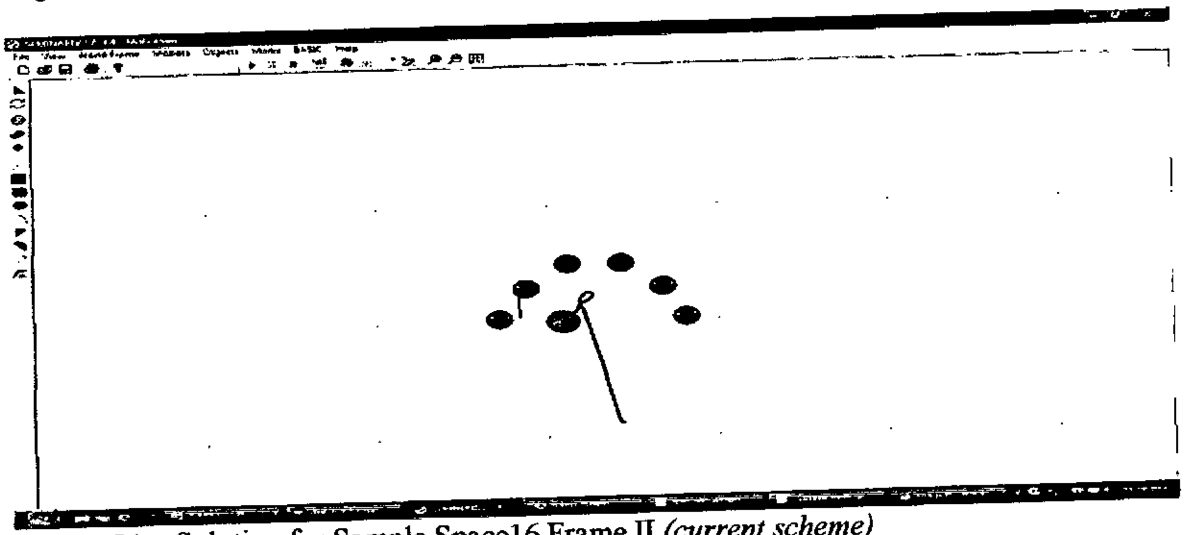


Figure 31c: Solution for Sample Space16 Frame II *(current scheme)*

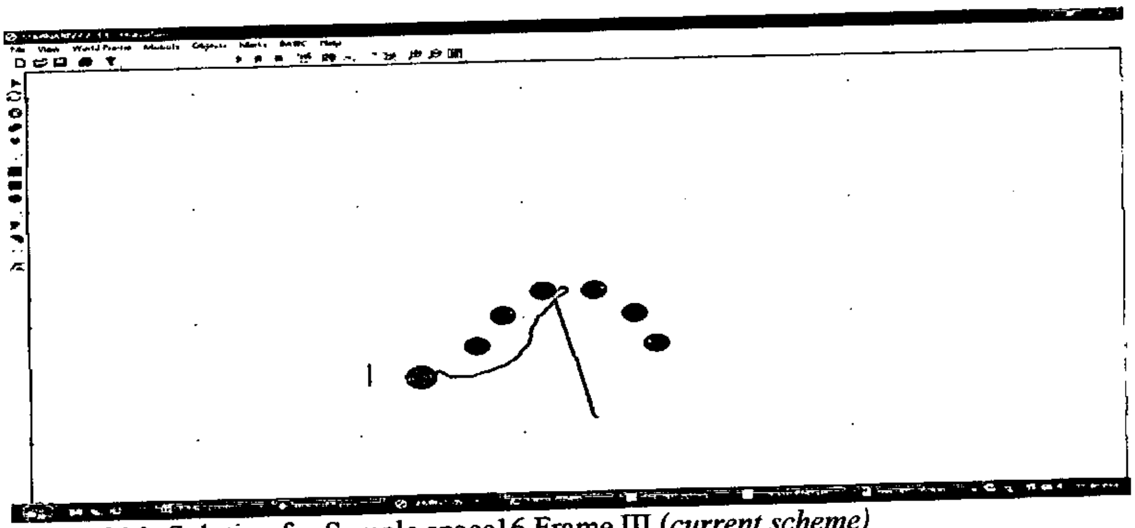


Figure 31d: Solution for Sample space16 Frame III *(current scheme)*

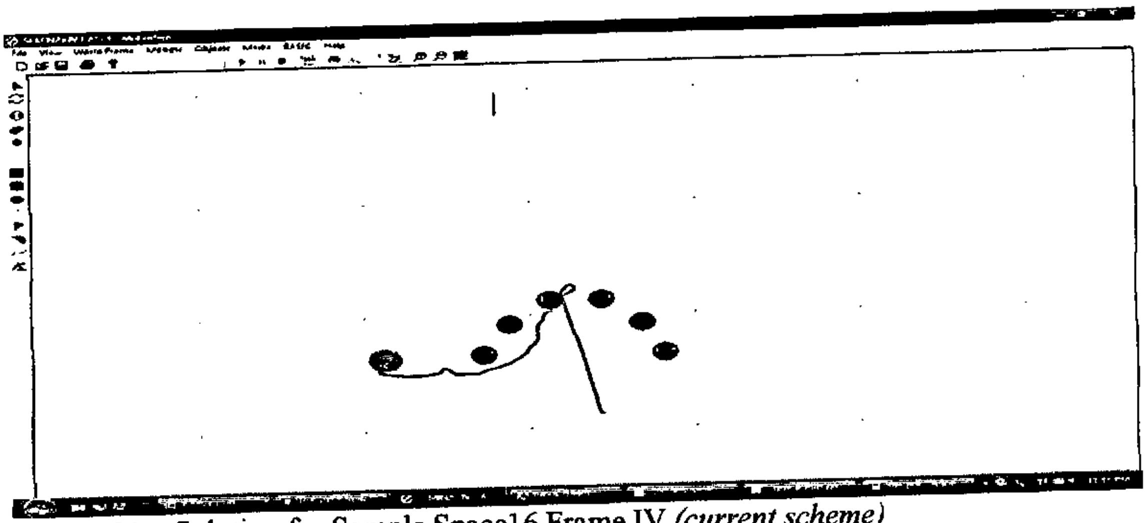


Figure 31e: Solution for Sample Space16 Frame IV *(current scheme)*

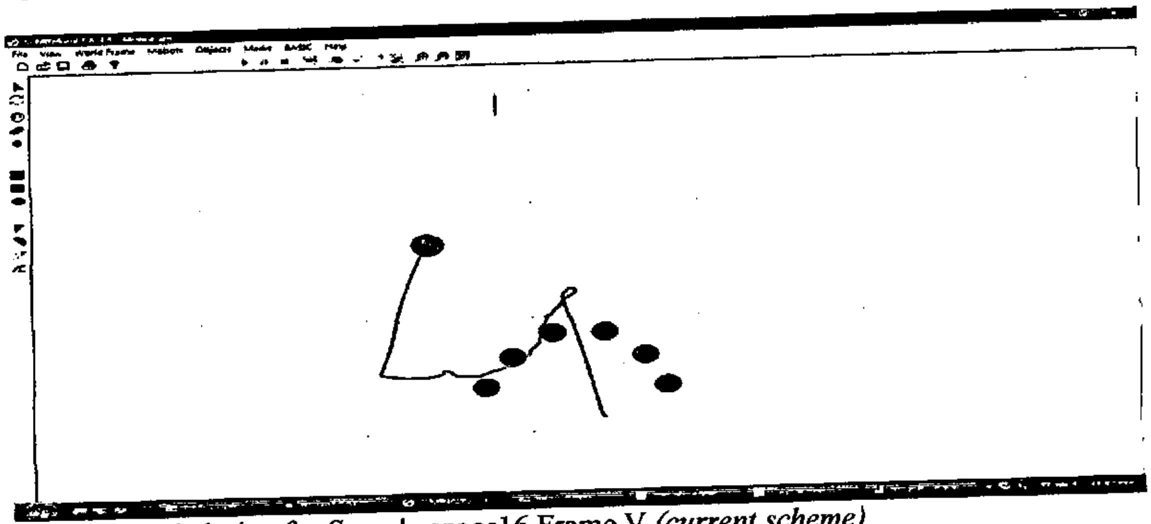


Figure 31f: Solution for Sample space16 Frame V *(current scheme)*

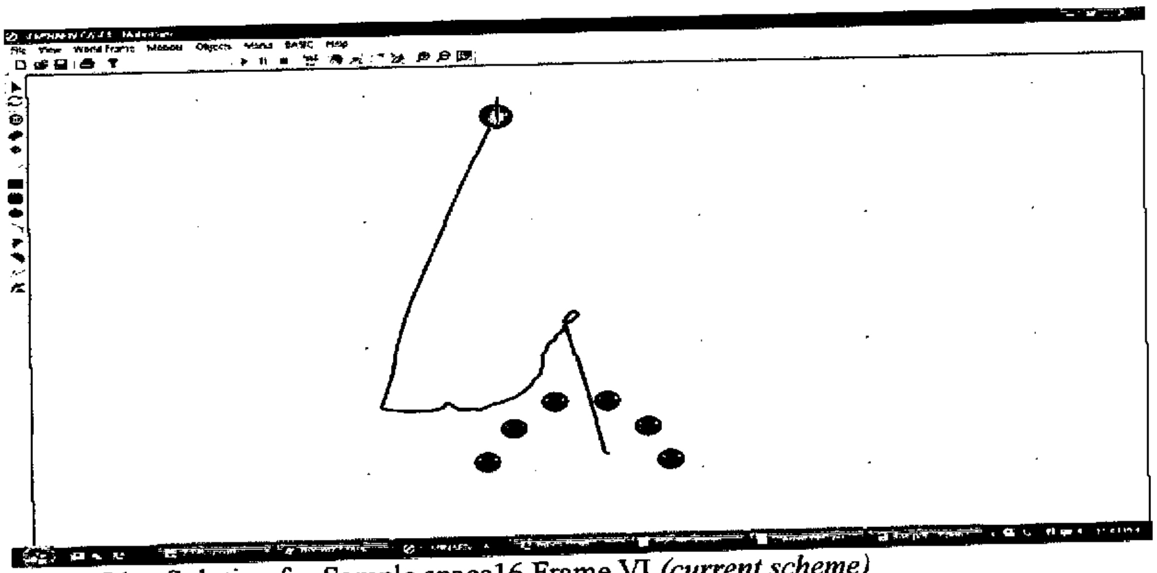


Figure 31g: Solution for Sample space16 Frame VI *(current scheme)*

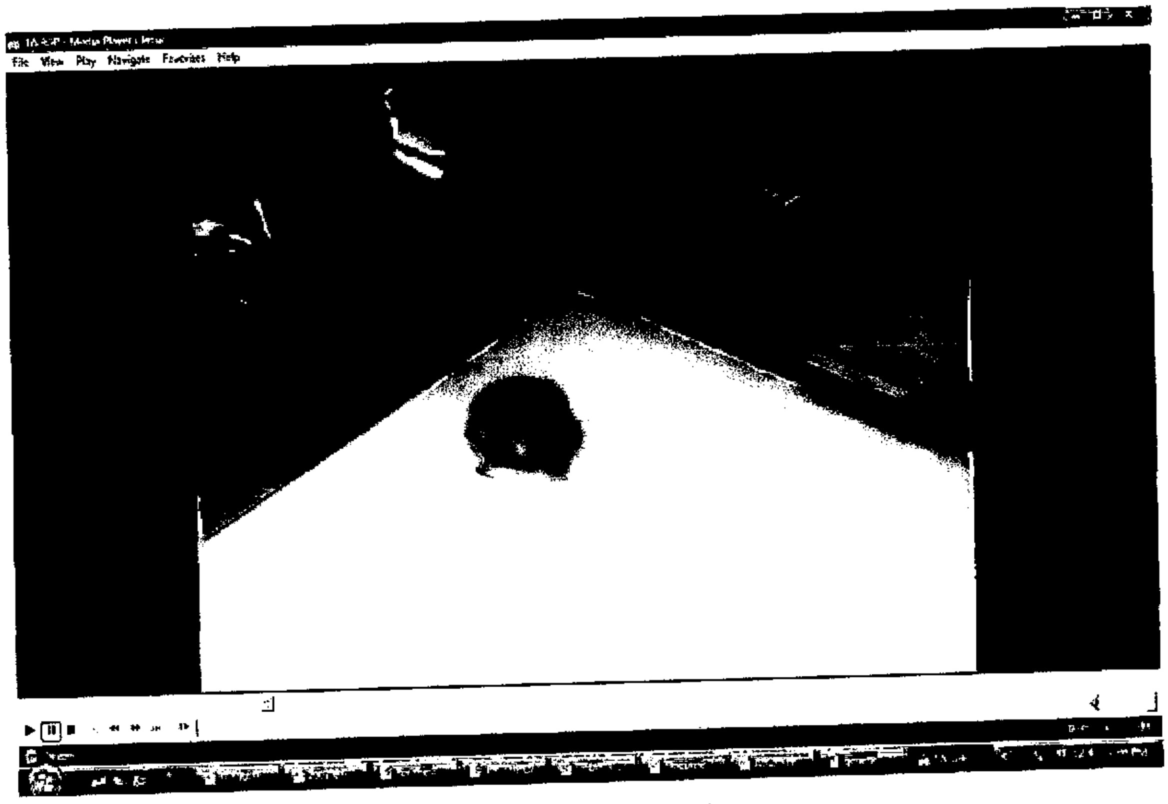


Figure 32a: Problem Sample space17 *(current scheme)*

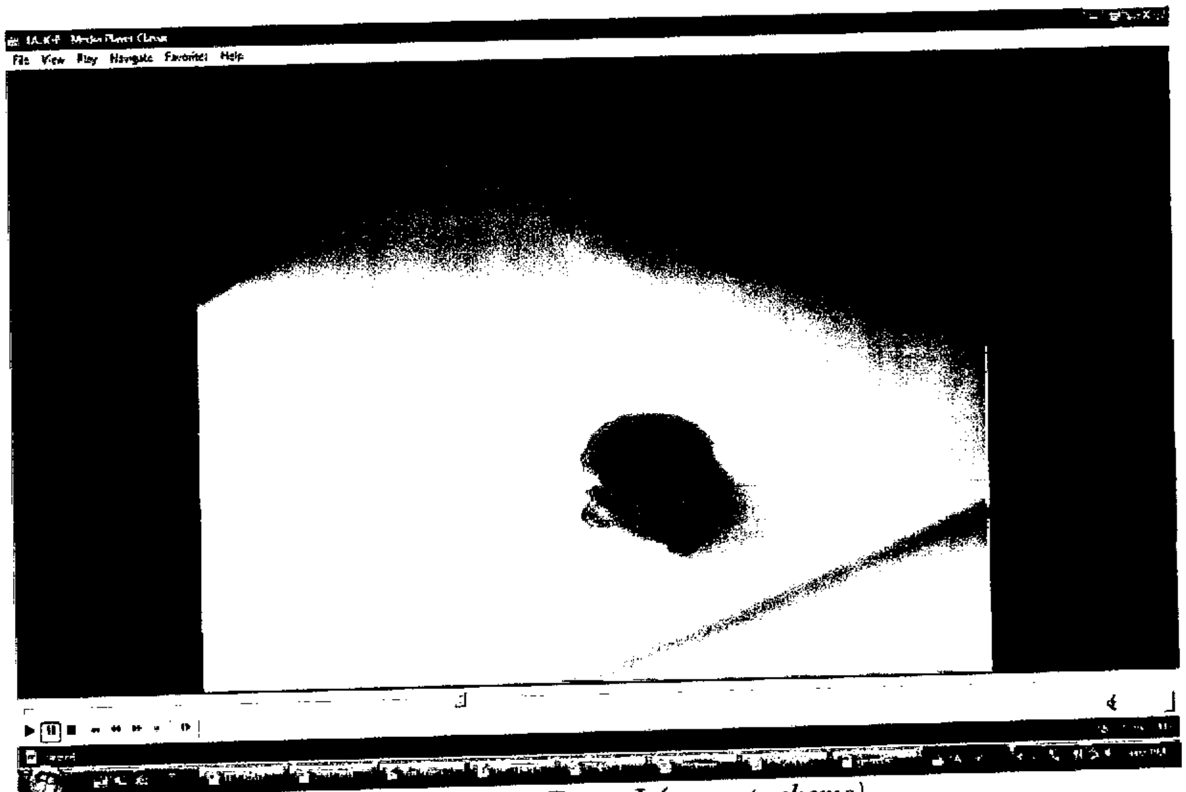


Figure 32b: Solution for Sample space17 Frame I *(current scheme)*

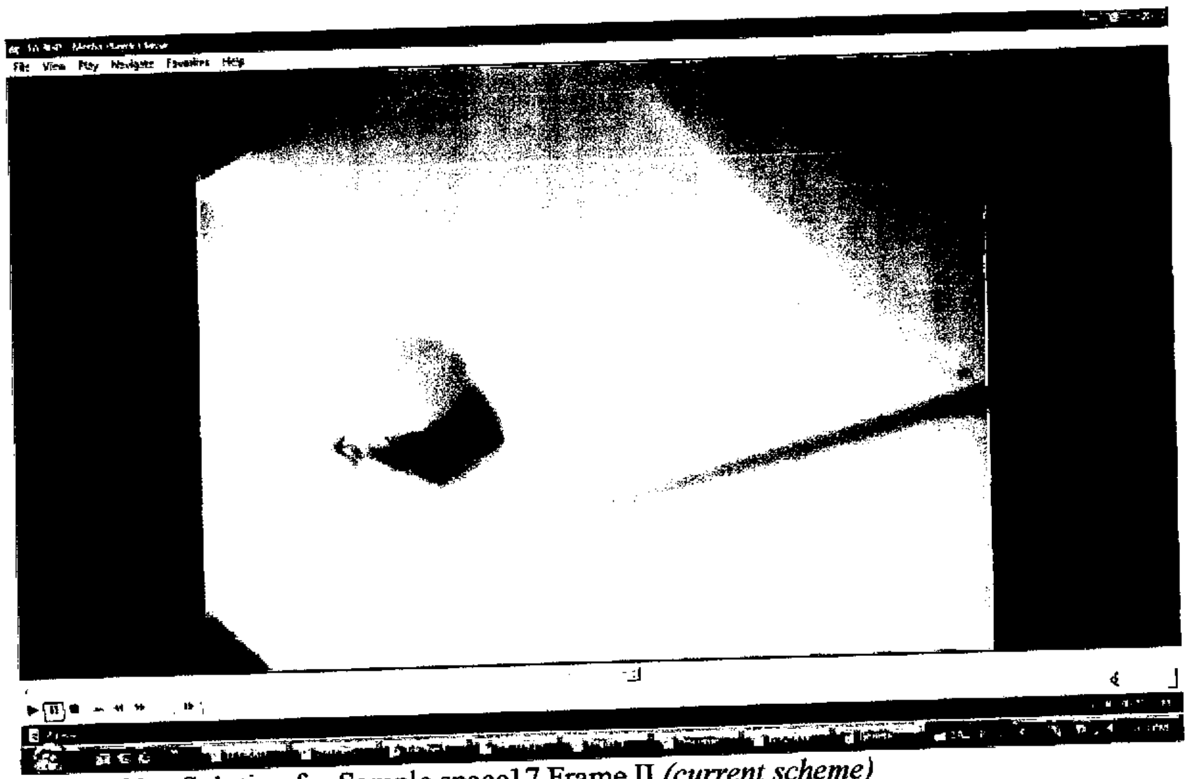


Figure 32c: Solution for Sample space17 Frame II *(current scheme)*

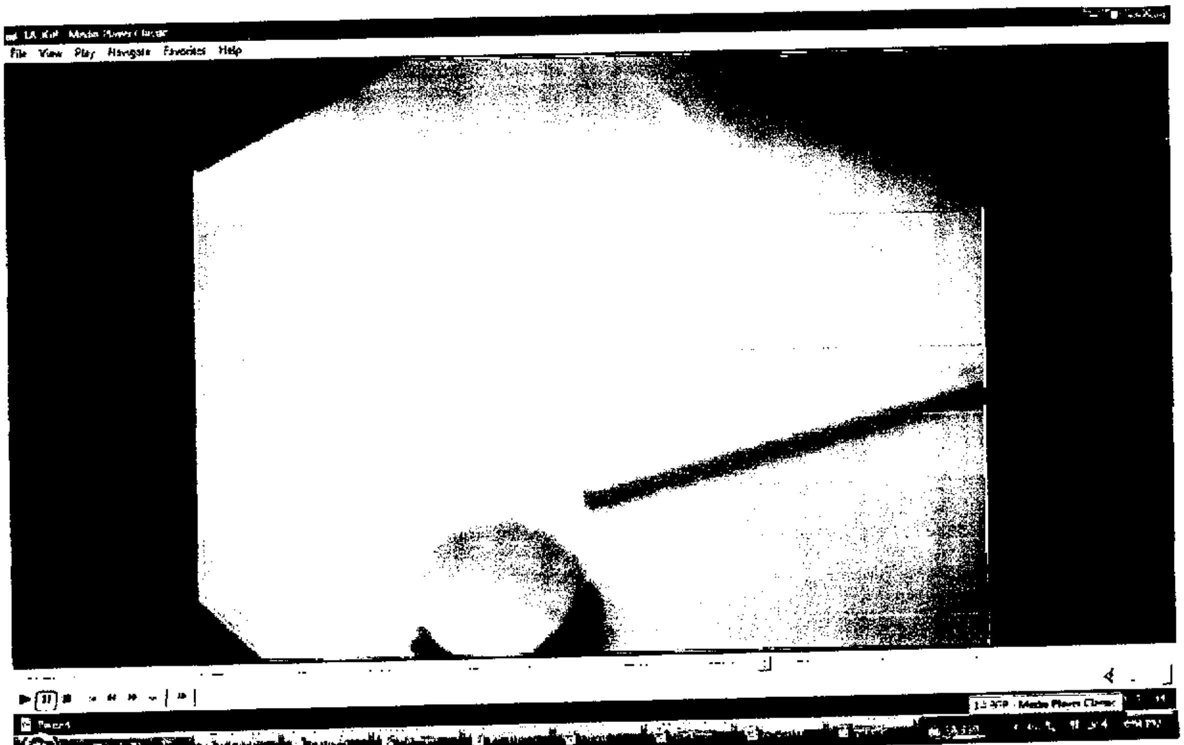


Figure 32d: Solution for Sample space17 Frame III *(current scheme)*

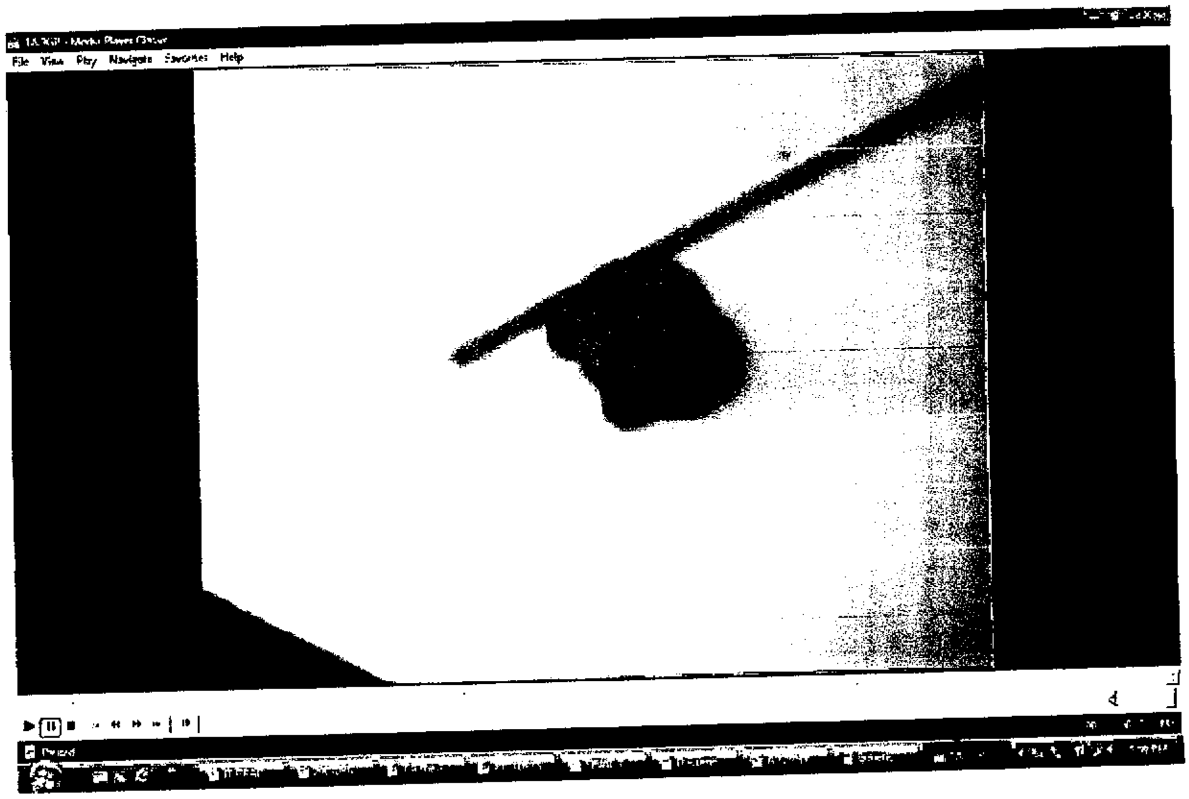


Figure 32e: Solution for Sample space17 Frame IV *(current scheme)*

6.2 ALGORITHMIC COMPARISON

Whereas a comparative analysis of path planning algorithms is desirable, its significance sometimes becomes contentious when the evaluation process is carried out using different pedestals for the host navigation environments, or operating with different underlying libraries and machines. To enable an unbiased, fair and easy evaluation process of path planning techniques, the need to have uniform standards for source codes and ensure that testing is carried under the same workspace conditions, remain paramount. It is only under such conditions that the use of running time becomes a good measurement criterion. Traditionally some of the popular criteria invoked for algorithmic comparison in robot path planning include: computational time, ability to find a solution, optimality of solution, ability of method to avoid getting trapped, ability to find alternate paths and degree of automation in the planning process. These set of criteria are rarely fully discussed in most path planning papers. For instance in considering the use of execution time as a measure of the efficacy of an algorithm we are confronted with the problem that computers, languages and programs run at different speeds, and that the execution time for the same algorithm may vary between different implementation environments. Nonetheless comparisons between algorithms can still be carried out as shown in Table 2 and it might be sufficient at this stage to highlight some of the other salient features that should be given attention when analyzing a path planning algorithm.

6.2.1 Level of Automation

Some of the path planning techniques require a complete knowledge of the navigation environment prior to the robot's motion. Otherwise the robot may find it difficult to adapt to either a changed or changing environment while it is still in motion and this might require restructuring the control programmes each time a replan is needed. Fundamental planning techniques that fall in this category include: spatial relationships, configuration space, visibility graph and Voronoi diagram. A disadvantage of these roadmap-based methods according to Geraerts and Overmars is that they necessarily output a fixed path in response to a query. This leads to predictable motions and lacks the level of flexibility required when the environment or robot changes.

Table 2: A Comparative Chart of some widely used Path Planning Methods Adapted from Jack (2001)

S/N	Path Planning Technique	Space	Machine & Time	Robot Rotation	Setup of Obstacle space	Optimized Variable	Solution Method	Remarks
1	Cartesian Configuration Space (Lozano-Perez, 1979)	2D & 3D	PL/1 on IBM370/16 time Unknown	No	convex polygons	Distance	search of obstacle corners	Good for 2D mobile robots. Also for completely known environment
2	Generalized Cones (Brooks,1983)	2D	Not available	No	Convex Polygons	Distances and avoidance	Search of midpoints between obstacles	Only good for 2D mobility; Also for completely known environment
3	Joint Configuration Space (Red, et.al., 1985)	3D	2 to 5 minute setup, 15 second solution on VAX 11/780	yes	Convex Polygons	Time	Search of joint Configura tion Space	Good for planar motion.
4	Spatial Planning Vision Based (Wong, et al, 1986)	3D	VAX 11/750 1 to 20 Seconds	yes	Work cell views	Avoidance	Search of Oct-Trees	Excellent for 3D mobile robot
5	Velocity Optimization (Dubowsky et al, 1986)		minutes on micro VAX	yes	unknown	Velocities, avoidance and joint limits time	Optimize Path splines	Offline program package available
6	Oct-Tree (Faverjon, 1986)	3D	Perkin -Elmer under 1 minute	yes	Solid Primitives from custom CAD	Distance & Avoidance	Search of Oct-Tree	Has Potential in CIM System
7	Potential Field (Y.K.Hwang, N.Ahuja, 1988)	2D or 3D	SUN 260, Tens of Minutes	yes	convex Polygons or Polyhedra	Avoidance & Distance	3 Findpath Routines in Potentials	This is slow, but has produced some good paths.
8	Proposed HVFF	2D or 3D	Core duo Pentium4; average of 0.8m/s; however it could be faster than this.	Yes	Concave or convex polygons	Distance	Artificial Potential, virtual obstacle and virtual goal concepts	Good for overcoming concave traps for mobile robots; partially known environment

On the other hand incremental replanning is a characteristic feature of the VFF concept. This makes it possible for a planner to greatly reduce computational cost, as it only updates the path locally, and when possible, obtains a globally optimal path.

6.2.2 Obstacle motion types and Geometry

Obstacles may be categorized into the following motion categories viz: static (un-moving), deterministic i.e. has predictable occurrence and positions and random i.e. freely moving, with no regular pattern. A path planning algorithm that is developed for a static or structured obstacle environment may not be effective in a dynamic or unstructured obstacle domain. Thus comparison of path planners from different obstacle environments may be of limited value.

Also, the geometrical nature of obstacles is another criterion that cannot be ignored. In this regard, an algorithm that performs in a domain of convex obstacles may not perform in a domain of concave obstacles. Hence, the above highlights the significance of ensuring uniformity of certain features before selecting an index for comparing or ranking the effectiveness of path planning techniques.

6.2.3 Rotations

Another inherent problem for some path planners lies in their rotating capability. During navigation, some planners do not permit for robot rotation, some will rotate only at certain 'safe' points, and some will rotate along a complete path. The effective combination of orientation and translation capabilities suggests the need for a level of intelligence in this area.

6.2.4 Growing of obstacles

Some techniques such as the Road Map techniques e.g. spatial relationships, visibility graph and configuration space plan their paths by extending the sides, edges or vertices of the obstacles. Their mode of operation is such that they try to travel at an optimal path by locating free configurations around the edges of contact obstacles.

6.2.5 Computer Memory Size Optimization

According to Yahja et al. (1998), approaches to path planning for mobile robots can be broadly classified into two categories viz: exact representations of the world as mentioned in Lozano-Perez (1979) or; Whitcomb and Koditschek (1987) and those that use a discretized representation such as found in Connolly and Grupen (1993) and Lengyel et al (1990). Coding of any representation takes up computer memory space and there is need to maintain some balance between detailing of the workspace representation and available memory space for such a task.

According to Samet (1988), one way to curtail memory requirements is to use a quadtree representation instead of a regular grid. Such quadtree representation allows for the successive subdivision of a region into four equally sized quadrants but is of limited use in environments where cost values vary over a continuum, unless the environment includes large regions with constant traversability costs. Yahja et al. (1998) on the other hand have suggested that quadtrees allow efficient partitioning of the environment since single cells can be used to encode large empty regions. However, paths generated using quadtrees are suboptimal because they are constrained to segments between the centers of the cells. Thus what we need is an appropriate robotic system that can allow auto scaling of Path Planning methods within the working environment, and still provide reasonable accuracy.

Furthermore, we would like to state that our hybrid algorithm inherits a number of attributes from standard path planning algorithms. One of the concepts adapted from the traditional path planning approach is the use of uniform cell decomposition, with a sub-class known as the approximate cell decomposition. This is a sub-category of the cell decomposition approach as contained in the Road-Map path planning method.

The approximate cell decomposition is basically a grid system made up of a set of square shaped arrays. In our work this has effectively enhanced capturing of the configuration space of various objects in the workspace relative to a fixed reference point. Use of arrays is good when fast recall of information from a map is required. Even though it has some inherent limitations such as long set up time, large working memory requirement and slow processing of algorithms, the availability of high speed computers has reduced the significance of these limitations. Some of the other methods by which objects are represented in robot path planning include: Polygons, Polyhedra (constructed with 3D polygons), Ellipsoids, sets of points, analytic surfaces, Oct-trees, Quad-trees, Constructive Solid Geometry (CSG), and Balanced Trees.].

The main advantage of discretization is that the computational complexity of path planning can be controlled through the adjustment of cell sizes. Road-map concepts are basically hinged on planned control systems. This invariably means that these methods capture the global connectivity of the robot's free space into a condensed graph which is subsequently searched for a path. It is clear from the above that the method of representation chosen can limit the use of complex shapes as obstacles. Some methods are very receptive to data acquired through sensors and CAD systems.

Since autonomous navigation of mobile robots is widely recognized by researchers as the state of the art in mobile robot technology it implies that a path planning concept that has the capability of making navigation decisions online is strongly desired. Our hybrid concept we believe is in this direction of thought.

In this chapter we have presented results obtained from the validation process of the HVFF algorithm. Simulations of the motions were used to validate the efficacy of HVFF over existing algorithms for the dynamic and static obstacle architectures using MobotSim (a customized software for robot animation).

Furthermore, we also confirmed the new HVFF concept by demonstrating the performance of a prototype robotic vehicle designed, built and operated based on the above HVFF concept.

CHAPTER SEVEN

CONCLUSION

[SUMMARY AND FINDINGS, CONTRIBUTION TO KNOWLEDGE AND FUTURE WORK]

7.1 SUMMARY AND FINDINGS

In this research, we have presented a robotic platform for both semi-autonomous (partially known environment) and fully-autonomous (completely unknown environment) navigation tasks. Experimental results from simulation and real robot validation have shown the effectiveness of our proposed approach. Our novel technique exhibits combined optimum planning/control capabilities i.e. it is capable of finding on-line and in real time the location and trajectory of motion which provides a desired value of some goodness measure in respect of some cost function.

The most important advantage of using a reactive based control strategy is that it is online compliant; hence it has the ability to cope with unstructured environments. To achieve this goal, the robot must be able to perceive its environment sufficiently to allow it navigate safely. The key requirements and research issues for future evolution of experimental robots, service robots and field robots are therefore in the areas of environment perception and scene interpretation, navigation (localization and map-building), multi-modal interaction and adaptability.

Further to the above, the mechanical and electrical design of the robot system is of key importance, in order to best adapt to various environments and situations. The physical architecture of the robotic vehicle also goes a long way to influence the energy consumption dynamics and the general performance measure or productivity in respect of task performance.

Our results have demonstrated that the HVFF hybrid concept as characterized by a set of timely executable algorithms and production rules is quite versatile and robust. It has the merits of high efficiency and effectiveness in terms of goal state attainment. Irrespective of the degree to which a workspace is clustered with obstacles of different shapes and sizes, a robot controlled by this approach can always reach the goal state without collision with obstacles provided a feasible path that is not smaller than the robot's diameter exists.

We find that our approach of representation for both the VOC and VGC is generalized for all shapes of obstacles and is in no way limited by the shape of the envelope of the obstacles creating a concave environment. We have also tried to establish some level of completeness and generalization by comparing its performance with those of earlier workers. The indications are that the HVFF is clearly more efficient than the earlier algorithms.

7.2 CONTRIBUTION TO KNOWLEDGE

Contributions to the Field of path planning for Autonomous Mobile Robots include:

i. development of a hybrid navigation concept that can be effectively and efficiently utilized in a maze of densely clustered static obstacles. A significant application area is a system where navigation is considered to be mission critical.

ii. development of a novel scheme capable of overcoming the traumatic local minimum problem associated with the popular artificial potential field technique posed by either lengthy obstacles or concave shaped obstacles for both static and dynamic cases.

7.3 FUTURE WORK

In subsequent work, we shall consider the following underlisted:

i. the multi-goal problem amidst static and dynamic obstacles. Likely areas of application of this work include problems of distribution of resources, conduct of inspection and search operations, especially in humanly unfriendly environments. Others include the development of intelligent planting machines for agriculture etc.

ii. Finally, there is the challenge of adapting our HVFF concept to higher-dimensional configuration-spaces for our built robot. In that case we shall be focusing on the navigation dexterity of the autonomous vehicle in a 2D environment while performing its task in a 3D domain. We can extend this concept to the robot arm manipulation. Subsequently, we foresee an integration of the autonomous mobile vehicle and the arm to form a unified system.

REFERENCES

1. Andrews, J.R. and Hogan, N. (1983) "Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator, Control of Manufacturing Processes and Robotic Systems", Eds. Hardt, D.E. and Book, W., ASME, Boston. pp. 243-251.

2. Arras, K.O., Persson, J., Tomatis, N. and Siegwart, R. (2002) "Real-Time Obstacle Avoidance For Polygonal Robots With A Reduced Dynamic Window", IEEE Proceedings of the International Conference of Robotics and Automation, Washington, USA. pp. 3050-3055.

3. Asaolu, O.S. (2001) "An Intelligent Path Planner for Autonomous Mobile Robots", PhD Thesis, Department of Systems Engineering, University of Lagos, Nigeria.

4. Ayari, I. and Chatti, A. (2007) "Reactive Control Using Behavior Modelling of a Mobile Robot", International Journal of Computers, Communications & Control Vol. II, No. 3, pp. 217-228.

5. Baştan, M. (2004) "Visual Servoing Of Mobile Robots Using Potential Fields", M.Sc. Thesis, Sabanci University.

6. Becker, M., Dantas, C. M. and Macedo, W. P. (2006) "Obstacle Avoidance Procedur e for Mobile Robots", ABCM Symposium series in Mechatronics vol. 2, pp. 250-257, copyright by ABCM.

7. Behring, C., Bracho, M., Castro, M. and Moreno, J. A. (2000) "An Algorithm for Robot Path Planning with Cellular Automata", Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry: Theoretical and Practical Issues on Cellular Automata, Springer-Verlag, pp.11-19.

8. Bikdash, M., Karagol, S. and Charifa, M. (2006) "Mesh Analysis with Applications in Reduced-Order Modeling and Collision Avoidance" Proceedings of the COMSOL Users Conference Boston, pp. 1-7.

9. Borenstein, J. and Koren, Y. (1989) "Real-time obstacle avoidance for fast mobile robots" IEEE Transactions on Systems, Man and Cybernetics, Volume (19), Issue: 5, pp. 1179-1187.

10. Borenstein, J. and Koren, Y. (1990) "Real-time obstacle avoidance for fast mobile robots in cluttered environments", Proceedings of IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, pp. 572-577.

11. Borenstein, J. and Koren, Y. (1991) "The Vector Field Histogram - Fast Obstacle-Avoidance for Mobile Robots", IEEE Journal of Robotics and Automation, Vol.7, No. 3, pp: 278-288.

12. Borenstein, J. and Koren, Y. (1991) "Histogramic in-motion mapping for mobile robot obstacle avoidance", IEEE Transactions on Robotics and Automation, Volume: 7, Issue: 4, pp. 535- 539.

13. Brooks, R.A. and Lozano-Perez, T. (1985) "A Subdivision Algorithm in Configuration Space for find path with Rotation", IEEE Transactions on Systems, Man, and Cybernetics, Vol.SMC-15 (2), 224-233.

14. Bruce, J. and Veloso, M. (2003) "Fast and Accurate Vision-Based Pattern Detection and Identification", Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taiwan, pp. 1277-1282.

15. Carpin, S. and Parker, L. E. (2002) "Cooperative Motion Coordination amidst Dynamic Obstacles", Distributed Autonomous Robotic Systems 5, pp. 145-154.

16. Caselli, S., Reggiani, M. and Rocchi, R. (2001) "Heuristic Methods for Randomized Path Planning in Potential Fields", Proc. of the IEEE Symp. on Computational Intelligence in Robotics and Automation, Banff, Alberta, Canada, pp. 426-431.

17. Castro, D., Nunes, U. and Ruano, A. (2002) "Reactive Local Navigation", Proceedings of 28th Annual Conference of the IEEE Industrial Electronics Society - IECON'02, Sevilla, vol.3, pp. 2427-2432 .

18. Chengqing, L., Ang Jr, M. H., Krishnan, H. and Yong, L.S. (2000) "Virtual Obstacle Concept for Local-minimum recovery in Potentialfield Based Navigation", Proc. of IEEE. International Conf. on Robotics and Automation, San Francisco, pp. 983-988.

19. Clark, C. M., Rock, S. M. and Latombe, J.C. (2000) "Motion Planning for Multiple Mobile Robot Systems using Dynamic Networks", Proc. of IEEE Int. Conf. on *Robot.* & Autom. Taipei, Taiwan 2003. pp. 4222-4228.

20. Connolly, C. I. and Grupen, R.A. (1993) "The Application of Harmonic Functions to Robotics", Journal of Robotic Systems, 10(7): 931-946.

21. Cosío, F. A. and Castañeda, M. A. P. (2004) "Autonomous robot navigation using adaptive potential fields Mathematical and Computer Modelling", Published by Elsevier Ltd. Volume 40, Issues 9-10, pp. 1141-1156.

22. Davis, R.H. and Camacho, M. (1984) "The Application of Logic Programming to the Generation of Paths for Robots" Robotica vol.2, pp 93-103.

23. Di Gesu, V., Lenzitti, B., Lo Bosco, G. and Tegolo, D. (2000) "A distributed architecture for autonomous navigation of robots", Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception, Washington, DC, USA, pp.190-194.

24. Dozier, G., Homaifar, A., Bryson, S. and Moore, L.(1998) "Artificial potential field based robot navigation, dynamic constrained optimization and simple genetic hill-climbing", The IEEE International Conference on Evolutionary Computation, Anchorage,AK,USA, pp.189–194.

25. Fasola, J., Rybski, P.E. and Veloso, M.M. (2005) "Fast Goal Navigation with Obstacle Avoidance Using a Dynamic Local Visual Model", VII SBAI / II IEEE LARS. São Luís, setembro de, pp.1-6.

26. Faverjon, B. and Tournassoud, P. (1987) "A Local Based Approach for Path Planning of Manipulators With a High Number of Degrees of Freedom", IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, pp. 1152-1159.

27. Gayle, R., Sud, A., Lin, M.C and Manocha, D. (2007) "Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments", In Proc IEEE International Conference on Intelligent Robots and Systems, San Diego, CA, pp. 3777-3783.

28. Ge, S. S. and Cui, Y. J. (2000a) "New Potential Functions For Mobile Robot Path Planning", IEEE Transactions On Robotics And Automation, Vol. 16, No. 5, pp. 615-621.

29. Ge, S. S. and Cui, Y. J. (2000b) "Dynamic Motion Planning For Mobile Robots Using Potential Field Method", Autonomous Robot, 13, pp. 207-222.

30. Geisler, T. and Manikas, T. (2002) "Autonomous Robot Navigation System Using a Novel Value Encoded Genetic Algorithm", Proceeding of IEEE Midwest Symposium on Circuits and Systems, Tulsa, OK, pp. 45-48.

31. Geraerts, R. and Overmars, M. H. (2005) "Reachability Analysis of Sampling Based Planners", Proceedings of the IEEE International Conference on Robotics and Automation Barcelona, Spain, pp. 406-412.

32. Gilbert, E.G. and Johnson, D.W. (1985) "Distance Functions and Their Application to Robot Path Planning in the Presence of Obstacles", IEEE Journal of Robotics and Automation, Vol. (1) No. 1, pp. 21-30.

33. Glasius, R., Komoda, A. and Gielen, S.A.M. (1995) "Neural Network Dynamics for Path Planning and Obstacle Avoidance", Neural Networks, Vol. 8, (1), pp. 125-133.

34. Han, L. and Amato, N. M. (2000) "A kinematics-based probabilistic roadmap method for closed chain systems", in Proceedings of the Workshop on Algorithmic Foundations of Robotics, WAFR' (00), pp. 233–246.

35. Hand, A., Godugu, J., Ashenayi, K., Manikas, T. W. and Wainwright, R.L. (2005) "Benchmarking of Robot Path Planning Algorithms, in Intelligent Engineering Systems Through Artificial Neural Networks: Smart Engineering Systems Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Complex Systems and Artificial Life", C.H. Dagli, et al., Editors. 2005, ASME Press: New York.

36. Hart, P.E., Nilsson, N.J. and Raphael, B. (1968) "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. of Systems Science and Cybernetics, 4 (2), pp. 100-107.

37. Heero, K., Willemson, J., Aabloo, A. and Kruusmaa, M. (2004) "Robots Find a Better Way: A Learning Method for Mobile Robot Navigation in Partially Unknown Environments", In Proceedings of the 8th Conference on Intelligent Autonomous Systems, (IAS-8), Amsterdam pp.559-566.

38. http://www.cut-the-knot.org/WhatIs/WhatIsGeometry.shtml [What Is Geometry]

39. http://hyperphysics.phy-astr.gsu.edu/hbase/vect.html (Basic Vector Operations)

40. http://www.glenbrook.k12.il.us/gbssci/Phys/Class/vectors/u3l1a.html Lesson 1: Vectors - Fundamentals and Operations

41. Hwang, Y.K. and Ahuja, N. (1992), Gross Motion Planning-A Survey, ACM Computing Survey, Vol.24, no.3, pp. 219-291.

42. Iagnemma, K. and Dubowsky, S. (2004) "Traction Control of Wheeled Robotic Vehicles in Rough Terrain with Application to Planetary Rovers", International Journal of Robotics Research, 23 (10), 1029-1040.

43. Ibidapo-Obe, O. and Asaolu, O.S. (2006) "Optimization Problems in Applied Sciences: From Classical Through Stochastic To Intelligent MetaHeuristic Approaches", 22:1-18; Contributed chapter in Handbook of Industrial and Systems Engineering, Edited by Badiru, A.B, CRC Press, Taylor and Francis Group, New York.

44. Ibidapo-Obe, O., Asaolu, O.S. and Badiru, A.B. (1999) "Generalized Solutions of the Pursuit problem in Three Dimensional Euclidean Space", Applied Mathematics and Computation, Elsevier Science Inc. 6580, pp. 1-11.

45. Ibidapo-Obe, O., Asaolu, O.S. and Badiru, A.B. (2002) "A New Method for the Numerical Solution of Simultaneous Non-Linear Equations", Applied Mathematics and Computation, Elsevier Science Inc. 125(1), pp. 133-140.

46. Im, K.Y. and Oh, S.Y. (2000) "An Extended Virtual Force Field Based Behavioral Fusion with Neural Networks and Evolutionary Programming for Mobile Robot Navigation", Evolutionary Computation, IEEE Congress Volume 2, pp. 1238-1244.

47. Ismael, A., Edite, F.V., Fernandes, M. G. P., Paula, M. and Gomes, S. F. (2004) "Robot trajectory planning with semi-infinite programming", European Journal

of Operational Research EURO Young Scientists Elsevier B.V. Volume 153, Issue 3 , pp. 607-617.

48. Jaafar, J. and Mckenzie, E. (2007) "Escape from local minima agent navigation in unknown virtual environment", 3rd International Conference on Intelligent Environments, publisher: Institute of Engineering Technology (IET) , pp. 191-197.

49. Jaafar, J. and McKenzie, E. (2008) A Fuzzy Action Selection Method for Virtual Agent Navigation in Unknown Virtual Environments, Journal of Uncertain Systems Vol.2, No.2, pp.144-154, 2008

50. Janglová, D. (2004) "Neural Networks in Mobile Robot Motion", International Journal of Advanced Robotic Systems, Volume 1 Number 1, ISSN 1729-8806, pp.15-22.

51. Kant, K. and Zucker, S. (1988) "Planning Collision-Free Trajectories in Time-Varying Environments: A Two-Level Hierarchy", Proceedings IEEE International Conference on Robotics and Automation, Philadelphia, pp 1644-1649.

52. Karlsson, N., Munich, M. E., Goncalves, L., Ostrowski, J., Bernardo, E. D. and Pirjanian, P. (2004) "Core Technologies for Service Robotics", Proc. of Int. Conf. on Intelligent Robots and Systems (IROS), Sendai, Japan, vol(3), pp. 2979-2984.

53. Khatib, O. (1986) "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", Int. J. Robotics Res. 5(1). pp 90-98.

54. Khosla, P. and Volpe, R. (1988) "Superquadric Artificial Potentials for Obstacle Avoidance and Approach", proc. of the IEEE Int. Conf. on Robotics and Automation, Philadelphia, PA, pp.1178-1784.

55. Kim, J., Pearce, R. and Amato, N. (2003) "Extracting optimal paths from roadmaps for motion planning", IEEE Int. Conf. on Robotics and Automation, pp. 2424.2429.

56. Koren, Y. and Borenstein, J. (1991) "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", Proc. of the IEEE Conference on Robotics and Automation, Sacramento, California, 1991, pp. 1398-1404.

57. Krogh, B.H. and Thorpe, C.E. (1986) "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles", Proceedings of the IEEE International Conference on Robotics and Automation San Francisco, California, April 7-10, 1986, pp. 1664-1669.

58. Kalmár-Nagy, T., D'Andrea, R. and Ganguly, P. (2004) "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle", Robotics and Autonomous Systems Elesevier computer science.com 46, pp 47–64.

59. Kunwar F. and Benhabib, B. (2006) "Rendezvous-guidance trajectory planning for robotic dynamic obstacle avoidance and interception" IEEE Trans Syst Man Cybern B Cybern, 36(6): pp. 1432-1441.

60. Laumond, J. (1986) "Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints", Intelligent Autonomous Systems, An International Conference held in Amsterdam, The Netherlands, pp 346-354.

61. Lengyel, J., Reichert, M., Donald, B. R. and Greenberg, D. P., (1990) "Real Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware," In Proc. SIGGRAPH.

62. Li, Y.P., Zielinska, T., Ang Jr. M.H. and Linz, W. (2006) "Vehicle Dynamics of Redundant Mobile Robots with Powered Caster Wheels", Proceedings of the Sixteenth CISM_IFToMM Symposium, Romansy 16, Robot Design, Dynamics and Control, ed. Teresa Zielinska and Cezary Zielinski, Warsaw: Springer. (Romansy 16 Robot Design, Dynamics and Control, Warsaw University of Technology, Warsaw, Poland, pp. 221-228.

63. Lingelbach, F. (2004a) "Path planning using probabilistic cell decomposition," in Proc. of the IEEE Int. Conf. on Robotics and Automation, New Orleans, LA, USA, pp. 467–472.

64. Lingelbach, F. (2004b) "Path Planning for Mobile Manipulation using Probabilistic Cell Decomposition", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan.

65. Lingelbach, F. (2005) "Path Planning using Probabilistic Cell Decomposition", Tech. rept. Frank Lingelbach. Licentiate Thesis.

66. Lozano-Perez, T. (1987). "A simple Motion-Planning Algorithm for General Robot Manipulators", IEEE J. of Robotics and Automation.RA-3(3): 224-238.

67. Mbede, J. B., Huang, X. and Wang, M. (2000) "Fuzzy motion planning among dynamic obstacles using artificial potential fields for robot manipulators", Robotics and Autonomous Systems 32(1): 61-72.

68. Mbede J. B., Ma S., Zhang L., Toure Y and Graefe V. (2004) "Robust neuro-fuzzy navigation of mobile manipulator among dynamic obstacles", Proceedings of IEEE International Conference on Robotics and Automation New Orleans Riverside, New Orleans, LA, USA.

69. Meng, A.C. (1988) "Dynamic Motion Re-planning for Unexpected Obstacles", Proceedings IEEE International Conference on Robotics and Automation, Philadelphia, pp. 1848-1849.

70. Montano L., Sagues C., (1991) "Non-Contact Compliant Robot Motion: Dynamic Behavior and Application to Feature Localization", IMACS Symposium Modelling and Control of Technological Systems, pp. 457-464.

71. Moravec, H.P. and Elfes, A. (1985) "High Resolution Maps from Wide Angle Sonar, IEEE Conference on Robotics and Automation", Washington D.C. pp 116-121.

72. Muck, K.L. (1988) "Motion Planning in Constraint Space", Proceedings 1988 IEEE International Conference on Robotics and Automation, Philadelphia, pp 633-635.

73. Murray, D. and Little, J. (2000) "Using real-time stereo vision for mobile robot navigation", Autonomous Robots, Publisher: Springer, Vol. 8, Number 2, pp. 161-171(11).

74. Murrieta-Cid, R., Tovar, B. and Hutchinson, S. (2005) "A Sampling-Based Motion Planning Approach to Maintain Visibility of Unpredictable Targets", Autonomous Robots, Vol. 19, Number 3, pp. 285-300.

75. Nancy Amato 2004, http//: www parasol.tamu.edu/~amato/Courses/padova04/ lectures/L6.poten-field.ps

76. Nguyen, B. Q., Chuang, Yao-Li., Tung, D., Hsieh, C., Jin, Z., Shi, L., Marthaler, D., Bertozzi, A. and Murray, R. M. (2005), Virtual Attractive-Repulsive Potentials for Cooperative Control of Second Order Dynamic Vehicles on the Caltech MVWT2005 American Control Conference, Portland, OR, USA

77. Nourani-Vatani, N., Bosse, M., Roberts, J., and Dunbabin, M. (2006) "Practical Path Planning and Obstacle Avoidance for Autonomous Mowing", In Proc. of the Australasian Conference of Robotics and Automation.

78. Ogren, P. and Leonard, N. E. (2002) "A Provably Convergent Dynamic Window Approach to Obstacle Avoidance", IFAC World Conference, Barcelona, Spain,-nt.ntnu.no 15th Triennial World Congress of the International Federation of Automatic Control, Barcelona.

79. Ögren, P. And Leonard, N.E. (2005) "A Convergent Dynamic Window Approach To Obstacle Avoidance", IEEE Transactions On Robotics, vol.(21), Issue 2, PP. 188-195.

80. Ordonez, C., Collins Jr. E.G., Selekwa, M.F., Dunlap, D.D. (2008) The virtual wall approach to limit cycle avoidance for unmanned ground vehicles, Elsevier, Robotics and Autonomous Systems 56, pp. 645–657.

81. Palma-Villalon, E. and Dauchez, P. (1988) "World Representation and Path Planning for a Mobile Robot", Robotica, Volume 6, pp 35-40.

82. Park, W.T. (1984) "State-Space Representation for Coordination of Multiple Manipulators", 14th ISIR, Gothenburg, Sweden, pp 397-405.

83. Park, M.G. and Lee, M.C. (2003) "Artificial Potential Field Based Path Planning for Mobile Robots Using a Virtual Obstacle Concept", Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), Victoria, pp. 735-740.

84. Park, M.G., Jeon, J. H. and Lee, M. C.(2001) "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing", in IEEE International Symposium on Industrial Electronics, South Korea, vol. 3, pp. 1530–1535, June 2001.

85. Philippsen, R., Jensen, B. and Siegwart, R. (2007) "Towards Real-Time Sensor-Based Path Planning in Highly Dynamic Environments", Autonomous Navigation in Dynamic Environments, Springer Tracts in Advanced Robotics, vol.(35), pp. 135-148.

86. Primbs, J. A. Nevistic, V. and Doyle, J. C. (1999) "Nonlinear optimal control: A control Lyapunov function and receding horizon perspective", *Asian J. Control*, vol. 1, no.1, pp. 14-24.

87. Qiao, S., Tang, C., Peng, J., Hu, J. and Zhang, H. (2006) "BPGEP: Robot Path Planning based on Backtracking Parallel-Chromosome GEP", Proceedings of the International Conference on Sensing, Computing and Automation Copyright Watam Press.

88. Sabe, K. Fukuchi, M. Gutmann, J. S. Ohashi, T. Kawamoto, K. and Yoshigahara, T. (2004) "Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision", Proceedings of the International Conference on Robotics and Automation, New Orleans,vol. (1), pp. 592-597.

89. Savage, J., Marquez, E., Pettersson, J., Trygg, N., Petersson, A. and Wahde, M. (2004) "Optimization of waypoint-guided potential field navigation using evolutionary algorithms", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS, Sendi, Japan, vol(4), pp. 3463-3468.

90. Sedighi, K.H., Ashenayi, K., Manikas, T.W., Wainwright, R.L. and Tai, H.M. (2004) "Autonomous Local Path Planning for a Mobile Robot Using a Genetic Algorithm", Proceeding IEEE Congress on Evolutionary Computation (CEC),1338-1345.

91. Shimoda, S. Kuroda, Y. and Iagnemma, K. (2005) "Potential Field Navigation of High Speed Unmanned Ground Vehicles on Uneven Terrain", Proceedings of the IEEE International Conference on Robotics and Automation Barcelona, Spain, pp. 2828-2833.

92. Shibata, T. and Fukuda,T. (1993) "Intelligent Motion Planning by Genetic Algorithm with Fuzzy Critic", Proceeding of the 1993 International Symposium on intelligent Control, Chicago, Illinois, USA, pp. 565-570.

93. Simionescu, P.A., Dozier, G.V. and Wainwright, R.L. (2006) "A Two-Population Evolutionary Algorithm for Constrained Optimization Problems", IEEE Congress on Evolutionary Computation, 2006. CEC 2006. Department of Mechanical Engineering, The University of Tulsa, 600 S. College Ave., Tulsa, OK 74104 USA, pp:1647- 1653.

94. Soetadji, T. (1986) "Cube Based Representation of Free Space for the Navigation of an Autonomous Mobile Robot", Intelligent Autonomous Systems, An International Conference held in Amsterdam, The Netherlands, pp 546-560.

95. Song, G., Miller, S. and Amato, N. (2001) "Customizing PRM roadmaps at query time", IEEE Int. Conf. on Robotics and Automation, Seoul, Korea, pp. 1500-1505.

96. Soucy, M. and Payeur, P. (2004) "Robot Path Planning with Multiresolution Probabilistic Representation: A Comparative Study", CCECE 2004 - CCGEI 2004, Niagara Falls, May/mai 2004 0-7803-8253-6/04/$17.00/©2004 IEEE.

97. Spenko, M. Iagnemma, K. and Dubowsky, S. (2004) "High Speed Hazard Avoidance for Mobile Robots in Rough Terrain", Paper presented in the SPIE Conference on Unmanned Ground Vehicles, Orlando, FL,USA, pp. 439-450.

98. Stachniss, C. and Burgard, W. (2002) "An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments", Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Lausanne, Switzerland, pp. 508–513.

99. Sugihara, K. and Smith, J. (1997) "Genetic Algorithms for Adaptive Motion Planning of an Autonomous Mobile Robot", Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, pp. 138-146.

100. Takahashi, O. and Schilling, R.J. (1989) "Motion Planning in a Plane Using Generalized Voronoi Diagrams", IEEE Transactions on Robotics and Automation, Vol.5, No. 2, pp. 143-150.

101. Thorpe, C. and Matthies, L. (1984) "Path Relaxation: Path Planning for a Mobile Robot", Proceedings of the National Conference on Artificial Intelligence, OCEANS 84, vol (16), pp. 576-581.

102. Trihatmo, S. and Jarvis, R.A. (2003) "Short-Safe Compromise Path for Mobile Robot Navigation In A Dynamic Unknown Environment", Australasian Conference on Robotics and Automation.

103. Tseng, C., Crane, C. and Duffy, J. (1988) "Generating Collision-FreePaths for Robot Manipulators", Computers in Mechanical Engineering, pp. 58-64.

104. Ulrich, I., Borenstein, J. (2000) "VFH*: local obstacle avoidance with look-ahead verification", Proceedings of IEEE International Conference on Robotics and Automation, San Francisco, CA, USA,Vol. 3, pp: 2505 - 2511.

105. Vadakkepat, P. and Chen, T.K. (2000) "Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning", Proceeding of the Congress on Evolutionary Computation, San Diego, CA, pp. 256-264.

106. Valavanis, K. P. Nelson, A. L. Doitsidis, L. Long, M. and Murphy, R. R. (2006), "Validation of a Distributed Field Robot Architecture Integrated with a MATLAB Based Control Theoretic Environment: A Case Study of Fuzzy Logic Based Robot Navigation", IEEE Robotics and Automation Magazine, Vol. 13, No. 3, pp: 93-107.

107. Verbeek, P.W., Dorst, L., Verwer, B.J.H. and Groen, F.C.A.(1986) "Collision Avoidance and Path Finding Through Constrained Distance Transformation in Robot State Space", Intelligent Autonomous Systems, An International Conference held in Amsterdam, The Netherlands, pp. 627-634.

108. Wang, L.C., Yong, L. S. and Ang Jr., M. H. (2002) "Hybrid of Global Path Planning and Local Navigation Implemented on a Mobile Robot in Indoor Environment", Proceedings of the IEEE International Conference on Intelligent Control, Vancouver, Canada.

109. Wei, W., Mbede, J. B. and Zhang, Q. (2001) "Fuzzy Sensor-Based Motion Control among Dynamic Obstacles for Intelligent Rigid-Link Electrically Driven Arm Manipulators" Journal of Intelligent and Robotic Systems, Vol. 30, Issue 1, pp. 49-71.

110. Whitcomb, L. L. and Koditschek, D. E. (1991) "Automatic Assembly Planning and Control via Potential Functions", In Proc. IEEE/RSJ International Workshop on Intelligent Robots and Systems. Osaka, Japan, vol. (1), pp. 17-23.

111. Whitley, D., Starkweather, T., and Bogart, C. (1990) "Genetic algorithms and neural networks: Optimizing connections and connectivity", Parallel Computing, vol. 14, pp. 347-361.

112. Wolf, J.C., Robinson, P., and Davies, J.M. (2004) "Vector Field Path Planning and Control of an Autonomous Robot in a Dynamic Environment", Proceedings of the 2004 FIRA World Congress (October), Busan, Korea, paper 151.

113. Xu, W.L., (2000) "A Virtual Target Approach for Resolving the Limit Cycle Problem in Navigation of a Fuzzy Behaviour-Based Mobile Robot, Elsevier, Robotics and Autonomous Systems 30, pp. 315-324.

114. Xu, W.L., Tso, S.K. and Lu, Z.K. (1998) "A Virtual Target Approach for Resolving the Limit Cycle Problem in Navigation of a Fuzzy Behaviour-Based Mobile Robot, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Victoria, B.C., Canada, pp. 44-49.

115. Yahja, A., Singh, S., and Stentz, A. (1998) "Recent Results in Path Planning for Mobile Robots Operating in Vast Outdoor Environments", In Proc. Symposium on Image, Speech, Signal Processing and Robotics, The Chinese University of Hong Kong.

116. Yang, S. X. and Meng, M. (2000) "An efficient neural network approach to dynamic robot motion planning", Neural Networks, Volume 13, Issue 2, pp. 143-148.

117. Yannier, S., Onat, A. and Sabanovic, A. (2003) "Basic Configuration for Mobile Robots", International Conference on Industrial Technology, ICIT'03, Maribor,Slovenia, pp. 256-261.

118. Youssef, S. M. (2005) "Neuro-based Learning of Mobile Robots with Evolutionary Path Planning", The ICGST Congress International Conference on Automation, Robotics and Autonomous Systems (ARAS-05), ICGST/ARAS, Cairo, Egypt, pp.64-70.

119. Yu, J. Cai, Z. Duan, Z. (2007) "Detection of Static and Dynamic Obstacles Based on Fuzzy Data Association with Laser Scanner", Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), Haikou, Vol.4, pp.172-176.

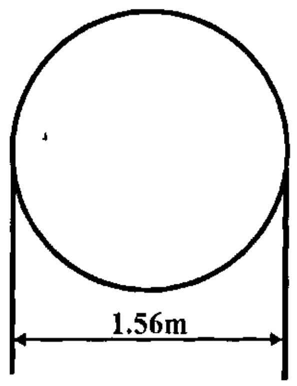
120. Zacharia, P. T. and Aspragathos, N.A. (2004) "Optimal robot task scheduling based on genetic algorithms", Robotics and Computer Integrated Manufacturing, Elsevier Ltd. Vol. 21, Issue 1, pp. 67-79.

121. Zaharakis, S.C. Guez, A. (1988) "Time Optimal Navigation Via Slack Time Sets", Proceedings 1988 IEEE International Conference on Robotics and Automation, Philadelphia, pp 650-651.

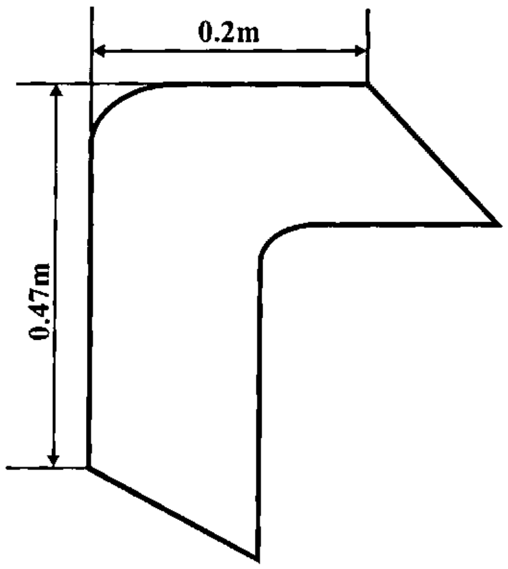
122. Zou, X.Y. and Zhu, J. (2003) "Virtual Local target method for avoiding local minimum in potential field based robot navigation", Journal of Zhejiang University of Science. 4(3), 264-269.

APPENDIX I

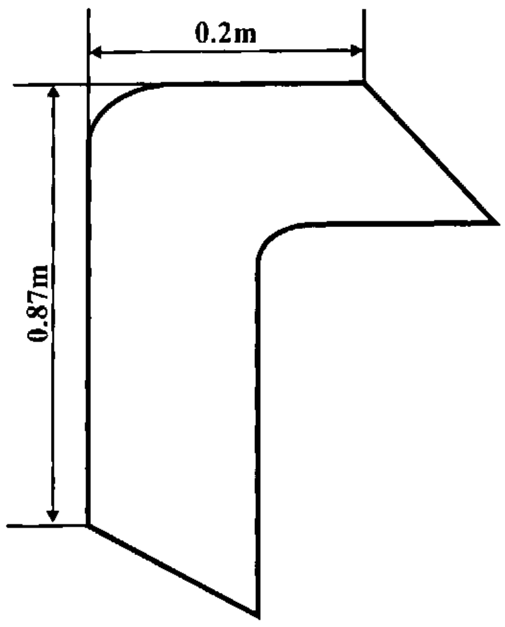
PROTOTYPE DESIGN



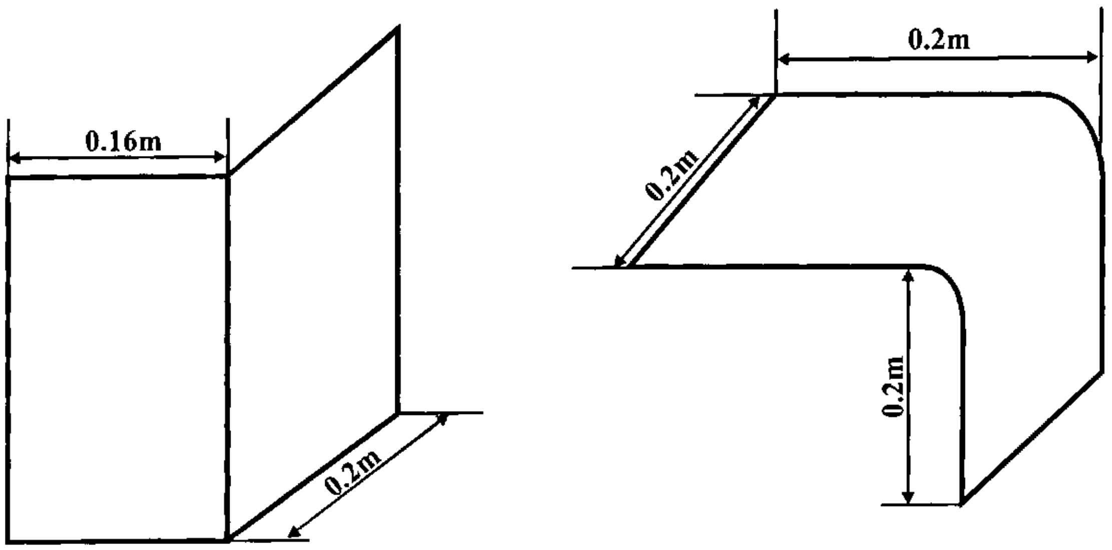
TOP COVER PLATE



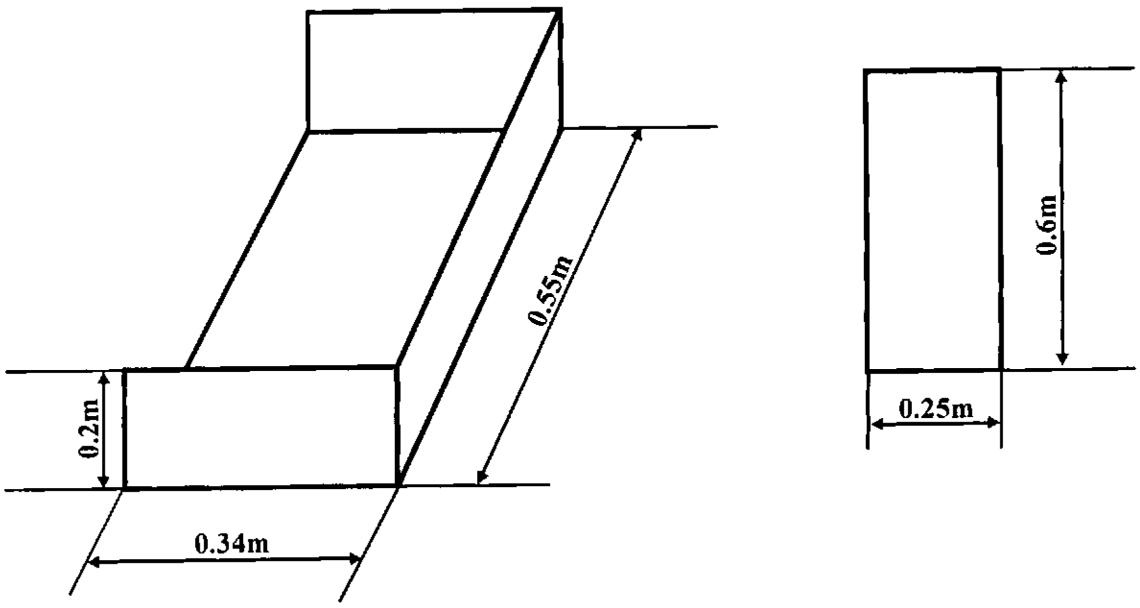
(COVER PLATE - BACK MOUNT)



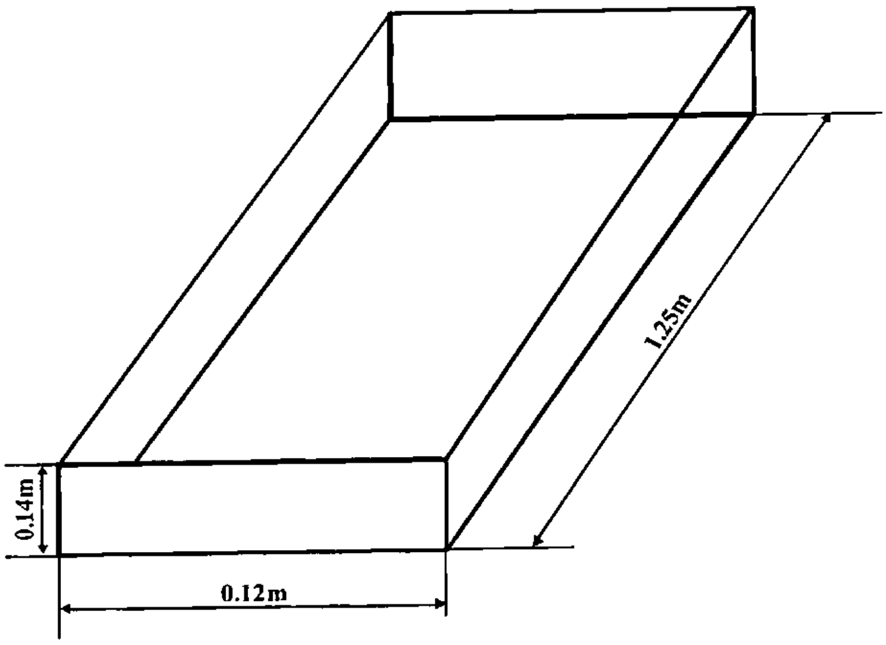
(SENSOR MOUNT X 3 PIECES)



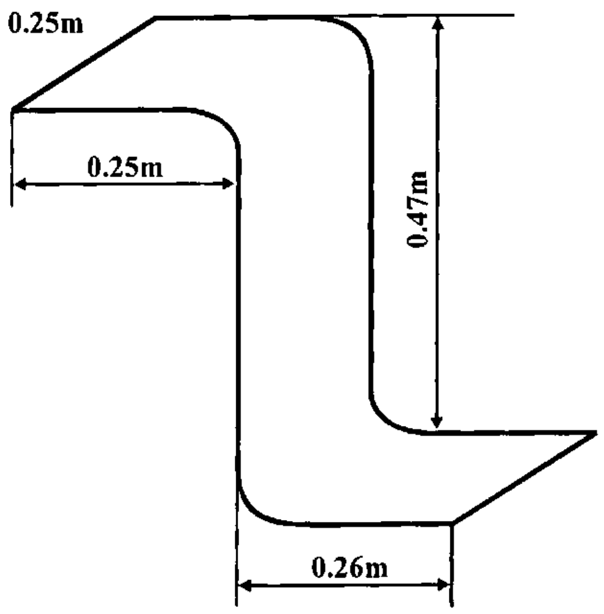
(COVER PLATE - FRONT MOUNT X 2 PIECES)



(SERVO CASING X 2 PIECES)



(BASEMENT PLATE/MICROCONTROLLER MOUNT AND BATTERY HOUSING



<u>(CASTER WHEEL MOUNT)</u>

APPENDIX II

PROGRAMME FOR PARTIALLY KNOWN ENVIRONMENT

```
Sub Main
Dim Y As Single
Dim  theta2 As Single
Dim theta1 As Single
Dim theta As Single
Dim dis_targ As Single
SetTimeStep(.05)  'Set simulation time step of 0.1 seconds
Fct=1  'Force constant (attraction to the target)
SetMobotPosition(0,4.52,6.12,120)
SetMarkPosition(0,2.92,4.45)
X_target=GetMarkX(0)  'Target coordinates (mark 0)
Y_target=GetMarkY(0)
a=X_target
b=Y_target
X1=GetMobotX(0)     ' Present mobot coordinates (in meters)
Y1=GetMobotY(0)
SetDrawTrajectory(0,1)
EraseTrajectories
SetCellSize(0,0.05)
numcells=GetNumCells(0)
Wait 1
theta1=GetMobotTheta(0)
dia=GetDiameter(0)
r=dia/2
PointXa=1.52
PointYa=4.3
PointXb=2.02
```

```
PointYb=3.75
PointXc=2.02
PointYc=4.30
Do                                      ' Start main loop
theta2=GetMobotTheta(0)
X2=GetMobotX(0)    ' Present mobot coordinates (in meters)
Y2=GetMobotY(0)
diff_x=x2-X
diff_y=y2-Y
        X=GetMobotX(0)      ' Present mobot coordinates (in meters)
                Y=GetMobotY(0)
        X_grid=CoordToGrid(0,X) ' indexes of cells where the
                Y_grid=CoordToGrid(0,Y)    ' mobot center is
                                ' Perform a range scan and update
                                        ' the Certainty Grid (max. cell value=3)
For e=.1 To .2
SetSensorRange(0,.05,e)
Next e
theta2=GetMobotTheta(0)
For sen=0 To 5
s=MeasureRange(0,sen,3)
rel_angle=(GetSensorAngle(0,sen,1))
Next sen
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
                Frx=0  ' Repulsive Force (x component)
                Fry=0  ' Repulsive Force (y component)
```

```
dis_targ=Sqr((X-X_target)^2+(Y-Y_target)^2)
dis_targAB=(2.03-1.52)
dis_targDB=(4.30-3.77)
dis_targABvar=(2.03-Y)
dis_targDBvar=(X-3.77)
Gradient_ADvar=dis_targDBvar/dis_targABvar
Gradient_AD=dis_targDB/dis_targAB
      ' Each occupied cell inside the windows applies a repulsive force to the mobot.
            For i=X_grid-10 To X_grid+10
                  For j=Y_grid-10 To Y_grid+10
            C=GetCell(0,i,j)
If C<>0 Then
      d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=rel_angle/d^2
                  Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d^2
                  Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d^2
If GetCollisionAngle(0)<=180 Then
    SetSteering(0,-0.1,0)
    StepForward
    SetSteering(0,0,180)
For t=1 To 5
      StepForward
    Next
    SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
Next
Next
' The target generates a constant-magnitud attracting force
```

```
                    Fcx=Fct*(X_target-X)/dis_targ
                    Fcy=Fct*(Y_target-Y)/dis_targ
                    Rx=Frx+Fcx   ' Resultant Force Vector
                    Ry=Fry+Fcy
                    rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                                              'current direction of travel and
                                                                              'direction of vector
R
                    SetSteering(0,.5,rot*3)   'mobot turns into the direction of R at constant speed
and steering rate proportional to the rotational difference
                    StepForward              ' Dynamics simulation progresses one time step
sim_time=GetSimulationTime
If dis_targ<=.1 Then Exit Do
If x<=3.38 And Y<=5.57 Then Exit Do
Debug. Print dia
Loop
vx1=1.73
vy1=4.55
SetMarkPosition(0,vx1,vy1)
Do                                   ' Start main loop
theta2=GetMobotTheta(0)
X2=GetMobotX(0)    ' Present mobot coordinates (in meters)
Y2=GetMobotY(0)
diff_x=x2-X
diff_y=y2-Y
          X=GetMobotX(0)      ' Present mobot coordinates (in meters)
                    Y=GetMobotY(0)
          X_grid=CoordToGrid(0,X) ' indexes of cells where the
                    Y_grid=CoordToGrid(0,Y)    ' mobot center is
                                        ' Perform a range scan and update
                                                  ' the Certainty Grid (max. cell value=3)
```

```
For e=.1 To .2
SetSensorRange(0,.05,e)
Next e
theta2=GetMobotTheta(0)
For sen=0 To 5
s=MeasureRange(0,sen,3)
rel_angle=(GetSensorAngle(0,sen,1))
Next sen
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
            Frx=0  ' Repulsive Force (x component)
            Fry=0  ' Repulsive Force (y component)
dis_targ=Sqr((X-vx1)^2+(Y-vy1)^2)
dis_targAB=(2.03-1.52)
dis_targDB=(4.30-3.77)
      ' Each occupied cell inside the windows applies a repulsive force to the mobot.
            For i=X_grid-10 To X_grid+10
                  For j=Y_grid-10 To Y_grid+10
            C=GetCell(0,i,j)
If C<>0 Then
      d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=rel_angle/d^2
                  Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d^2
                  Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d^2
If GetCollisionAngle(0)<=180 Then
    SetSteering(0,-0.1,0)
    StepForward
    SetSteering(0,0,180)
For t=1 To 5
      StepForward
    Next
    SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
Next
Next
' The target generates a constant-magnitud attracting force
            Fcx=Fct*(vx1-X)/dis_targ
            Fcy=Fct*(vy1-Y)/dis_targ
            Rx=Frx+Fcx   ' Resultant Force Vector
            Ry=Fry+Fcy
            rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                   'current direction of travel and direction of vector R

            SetSteering(0,.5,rot*3)        'mobot turns into the direction of R
                                              'at constant speed and steering rate
                                          'proportional to the rotational difference
            StepForward          ' Dynamics simulation progresses one time step
sim_time=GetSimulationTime
If dis_targ<=.1 Then Exit Do
Debug. Print dia
Loop
vx2=2.63
vy2=4.83
SetMarkPosition(0,vx2,vy2)
```

```
Do                                     ' Start main loop
theta2=GetMobotTheta(0)
X2=GetMobotX(0)    ' Present mobot coordinates (in meters)
Y2=GetMobotY(0)
diff_x=x2-X
diff_y=y2-Y
        X=GetMobotX(0)      ' Present mobot coordinates (in meters)
                Y=GetMobotY(0)
        X_grid=CoordToGrid(0,X) ' indexes of cells where the
                Y_grid=CoordToGrid(0,Y)    ' mobot center is
                                ' Perform a range scan and update
                                        ' the Certainty Grid (max. cell value=3)
For e=.1 To .2
SetSensorRange(0,.05,e)
Next e
theta2=GetMobotTheta(0)
For sen=0 To 5
s=MeasureRange(0,sen,3)
rel_angle=(GetSensorAngle(0,sen,1))
Next sen
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
                Frx=0  ' Repulsive Force (x component)
                Fry=0  ' Repulsive Force (y component)
dis_targ=Sqr((X-vx2)^2+(Y-vy2)^2)
        ' Each occupied cell inside the windows applies a repulsive force to the mobot.
                For i=X_grid-10 To X_grid+10
```

```
                        For j=Y_grid-10 To Y_grid+10
                C=GetCell(0,i,j)
If C<>0 Then
        d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=rel_angle/d^2
                        Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d^2
                        Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d^2
If GetCollisionAngle(0)<=180 Then
     SetSteering(0,-0.1,0)
     StepForward
     SetSteering(0,0,180)
For t=1 To 5
        StepForward
     Next
     SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
Next
Next
' The target generates a constant-magnitud attracting force
                Fcx=Fct*(vx2-X)/dis_targ
                Fcy=Fct*(vy2-Y)/dis_targ
                Rx=Frx+Fcx   ' Resultant Force Vector
                Ry=Fry+Fcy
                rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                                 'current direction of travel and
                                                         'direction of vector R
                SetSteering(0,.5,rot*3)          'mobot turns into the direction of R
                                              'at constant speed and steering rate
```

```
                                                          'proportional to the rotational difference
                StepForward             ' Dynamics simulation progresses one time step
sim_time=GetSimulationTime
If dis_targ<=.1 Then Exit Do
Debug. Print dia
Loop
SetMarkPosition(0,a,b)
Do                                      ' Start main loop
theta2=GetMobotTheta(0)
X2=GetMobotX(0)    ' Present mobot coordinates (in meters)
Y2=GetMobotY(0)
diff_x=x2-X
diff_y=y2-Y
      X=GetMobotX(0)     ' Present mobot coordinates (in meters)
            Y=GetMobotY(0)
      X_grid=CoordToGrid(0,X) ' indexes of cells where the
            Y_grid=CoordToGrid(0,Y)    ' mobot center is
                         ' Perform a range scan and update
                                     ' the Certainty Grid (max. cell value=3)
For e=.1 To .2
SetSensorRange(0,.05,e)
Next e
theta2=GetMobotTheta(0)
For sen=0 To 5
s=MeasureRange(0,sen,3)
rel_angle=(GetSensorAngle(0,sen,1))
Next sen
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
```

```
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
            Frx=0  ' Repulsive Force (x component)
            Fry=0  ' Repulsive Force (y component)
dis_targ=Sqr((X-a)^2+(Y-b)^2)
      ' Each occupied cell inside the windows applies a repulsive force to the mobot.
            For i=X_grid-10 To X_grid+10
                  For j=Y_grid-10 To Y_grid+10
            C=GetCell(0,i,j)
If C<>0 Then
      d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=rel_angle/d^2
                  Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d^2
                  Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d^2
If GetCollisionAngle(0)<=180 Then
    SetSteering(0,-0.1,0)
    StepForward
    SetSteering(0,0,180)
For t=1 To 5
      StepForward
    Next
    SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
Next
Next
' The target generates a constant-magnitud attracting force
            Fcx=Fct*(a-X)/dis_targ
            Fcy=Fct*(b-Y)/dis_targ
```

```
                Rx=Frx+Fcx   ' Resultant Force Vector
                Ry=Fry+Fcy
                rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                                      'current direction of travel and
                                                                   'direction of vector
R
                SetSteering(0,.5,rot*3)          'mobot turns into the direction of R
                                                    'at constant speed and steering rate
                                                    'proportional to the rotational difference
                StepForward             ' Dynamics simulation progresses one time step
sim_time=GetSimulationTime
If dis_targ<=.1 Then Exit Do
Debug. Print dia
Loop
End Sub
```

APPENDIX III

PROGRAMME FOR COMPLETELY UNKNOWN OBSTACLE DOMAIN: A SIMULATION STUDY

```
Sub Main
Dim Y1_target As Single
Dim distarg As Single
Dim dis_targ1 As Single
Dim time_ratio As Double
SetTimeStep(.1)
SetDrawTrajectory(0, 1)
EraseTrajectories
SetDrawTrajectory(1, 0)
Fct=1  'Force constant (attraction to the target)
robot_angle_initial=0
obstaclespeed=.01
robotspeed=.06
robotspeed1=robotspeed
robotspeed_initial=robotspeed
SetMarkPosition(0,0.8,4.13)
X_target=GetMarkX(0)  'Target coordinates (mark 0)
Y_target=GetMarkY(0)
q=270
real_targetx=GetMarkX(0)
real_targety=GetMarkY(0)
a=X_target
b=Y_target

dia_=GetDiameter(0)
dia_1=GetDiameter(1)
```

```
dia_2=GetDiameter(2)
dia_3=GetDiameter(3)
dia_4=GetDiameter(4)

r=dia/2
r1=dia_1/2
r2=dia_2/2
r3=dia_3/2
r4=dia_4/2
r_sum=r2+r
inc=.91
'*****************robot**************
SetDiameter(0,.2)
dia=GetDiameter(0)
r=dia/2
SetMobotPosition(0,1.44,6.87,200)
x_initial=GetMobotX(0)
y_initial=GetMobotY(0)
X_target=GetMarkX(0)  'Target coordinates (mark 0)
Y_target=GetMarkY(0)
SetDrawTrajectory(0,1)
SetCellSize(0,0.05)
numcells=GetNumCells(0)
SetSensorRange(0,.05,.35)
'***********obstacle one***************
SetSteering(1,obstaclespeed,0)
SetMobotPosition(1,0.8743,5.3077,q)
SetMobotRelPosition(1,0,0,0)
newX1_target=GetMobotX(1)
newY1_target=GetMobotY(1)
SetSensorRange(1,0,0.01)
```

```
X1=GetMobotX(1)
Y1=GetMobotY(1)
X1_target=x1+inc
Y1_target=y1+inc
'**************obstacle two*************
SetSteering(2,obstaclespeed,0)
SetMobotPosition(2,1.1192,5.0999,q)
SetMobotRelPosition(2,0,0,0)
newX2_target=GetMobotX(2)
newY2_target=GetMobotY(2)
SetSensorRange(2,0,0.01)
X2=GetMobotX(2)
Y2=GetMobotY(2)
X2_target=x2+inc
Y2_target=y2+inc
'**************obstacle three*************
SetSteering(3,obstaclespeed,0)
SetMobotPosition(3,1.6754,5.3077,q)
SetMobotRelPosition(3,0,0,0)
newX3_target=GetMobotX(3)
newY3_target=GetMobotY(3)
SetSensorRange(3,0,0.01)
X3=GetMobotX(3)
Y3=GetMobotY(3)
X3_target=x3+inc
Y3_target=y3+inc
'**************obstacle four*************
SetSteering(4,obstaclespeed,0)
SetMobotPosition(4,1.4296,5.0999,q)
SetMobotRelPosition(4,0,0,0)
newX4_target=GetMobotX(4)
```

```
newY4_target=GetMobotY(4)
SetSensorRange(4,0,0.01)
X4=GetMobotX(4)
Y4=GetMobotY(4)
X4_target=x4+inc
Y4_target=y4+inc
x1=GetMobotX(1)
y1=GetMobotY(1)
x2=GetMobotX(2)
y2=GetMobotY(2)
x3=GetMobotX(3)
y3=GetMobotY(3)
x4=GetMobotX(4)
y4=GetMobotY(4)
dis_targ1=Sqr((x1-x_target)^2+(y1-y_target)^2)
dis_targ2=Sqr((x2-x_target)^2+(y2-y_target)^2)
dis_targ3=Sqr((x3-x_target)^2+(y3-y_target)^2)
dis_targ4=Sqr((x4-x_target)^2+(y4-y_target)^2)
Wait .5
dis_targ=Sqr((X-X_target)^2+(Y-Y_target)^2)
x_initial=GetMobotX(0)
y_initial=GetMobotY(0)
Do
x=GetMobotX(0)
y=GetMobotY(0)
x1=GetMobotX(1)
y1=GetMobotY(1)
x2=GetMobotX(2)
y2=GetMobotY(2)
x3=GetMobotX(3)
y3=GetMobotY(3)
```

```
x4=GetMobotX(4)
y4=GetMobotY(4)
dis_targ=Sqr((X-X_target)^2+(Y-Y_target)^2)
dis_x1_targ=Sqr((x1-x_target)^2+(y1-y_target)^2)
dis_x2_targ=Sqr((x2-x_target)^2+(y2-y_target)^2)
dis_x3_targ=Sqr((x3-x_target)^2+(y3-y_target)^2)
dis_x4_targ=Sqr((x4-x_target)^2+(y4-y_target)^2)
dis_x1_robot=Sqr((x1-x)^2+(y1-y)^2)
dis_x2_robot=Sqr((x2-x)^2+(y2-y)^2)
dis_x3_robot=Sqr((x3-x)^2+(y3-y)^2)
dis_x4_robot=Sqr((x4-x)^2+(y4-y)^2)
dis_targ_x1_robot=dis_x1_robot+dis_x1_targ
dis_targ_x2_robot=dis_x2_robot+dis_x2_targ
dis_targ_x3_robot=dis_x3_robot+dis_x3_targ
dis_targ_x4_robot=dis_x4_robot+dis_x4_targ
dis_x1_robot=Sqr((x1-x)^2+(y1-y)^2)
dis_x2_robot=Sqr((x2-x)^2+(y2-y)^2)
dis_x3_robot=Sqr((x3-x)^2+(y3-y)^2)
dis_x4_robot=Sqr((x4-x)^2+(y4-y)^2)
If dis_targ1>0 And X1<X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1>Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1<X1_target And Y1>Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1<X1_target And Y1=Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1=Y1_target Then
StepForward
```

```
ElseIf dis_targ1>0 And X1=X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1=X1_target And Y1>Y1_target Then
StepForward
End If
If dis_targ1<=0.1 And dis_targ<=0.1 Then Exit Do
'main Start main loop
                X=GetMobotX(0)      ' Present mobot coordinates (in meters)
                Y=GetMobotY(0)
                X_grid=CoordToGrid(0,X) ' indexes of cells where the
                Y_grid=CoordToGrid(0,Y)    ' mobot center is
                                ' Perform a range scan and update
                                        ' the Certainty Grid (max. cell value=3)
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
rel_angle=Abs(GetSensorAngle(0,sen,1))
                Frx=0  ' Repulsive Force (x component)
                Fry=0  ' Repulsive Force (y component)
For i=Abs(X_grid-10) To X_grid+10
For j=Abs(Y_grid-10) To Y_grid+10
C=GetCell(0,i,j)
If C<>0  Then
If s0<.3 Or s1<.3 Or s2<.3 Or s3<.3 Or s4<.3 Or s5<.3 Then
d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=3*rel_angle/d^2
                        Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d
                Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d
If GetCollisionAngle(0)<=180 Then
    SetSteering(0,-0.1,0)
    StepForward
    SetSteering(0,0,180)
For t=1 To 5
        StepForward
    Next
    SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
End If
Next
Next
      ' The target generates a constant-magnitud attracting force
            Fcx=(X_target-X)/dis_targ
            Fcy=(Y_target-Y)/dis_targ
            Rx=Frx+Fcx   ' Resultant Force Vector
            Ry=Fry+Fcy
            rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                              'current direction of travel and
                                                  'direction of vector R
            SetSteering(0,robotspeed,rot*3)     'mobot turns into the direction of R
                                              'at constant speed and steering rate
                                              'proportional to the rotational difference
            StepForward           ' Dynamics simulation progresses one time step
If dis_targ <=.2 And dd-r_sum>=.01 Then
frx=0
fry=0
End If
```

```
robot_angle=GetMobotTheta(0)
'Capturing the robot's angle along its line of sight after taking care of the rotational difference
If s0>0 Then
r_adaptive=CStr(s0)
ElseIf s1 >0 Then
r_adaptive=CStr(s1)
ElseIf s2>0 Then
r_adaptive=CStr(s2)
ElseIf s3>0 Then
r_adaptive=CStr(s3)
ElseIf s4>0 Then
r_adaptive=CStr(s4)
ElseIf s5>0 Then
r_adaptive=CStr(s5)
End If
If Abs(x_initial-x) Or Abs(y_initial-y)>=0.15 Then Exit Do
Debug.Print FCX
Debug.Print FCy
Loop
r_adaptive=r_adaptive
robot_angle=GetMobotTheta(0)

'******************PHASE TWO*********************************************
*
'*NAVIGATION TOWARDS DETECTING THE CLOSEST OBSTACLE CONFIGURATI
ON***
coutr:
Do
x=GetMobotX(0)
y=GetMobotY(0)
x1=GetMobotX(1)
```

```
y1=GetMobotY(1)
x2=GetMobotX(2)
y2=GetMobotY(2)
x3=GetMobotX(3)
y3=GetMobotY(3)
x4=GetMobotX(4)
y4=GetMobotY(4)
dis_targ=Sqr((X-X_target)^2+(Y-Y_target)^2)
dis_x1_targ=Sqr((x1-x_target)^2+(y1-y_target)^2)
dis_x2_targ=Sqr((x2-x_target)^2+(y2-y_target)^2)
dis_x3_targ=Sqr((x3-x_target)^2+(y3-y_target)^2)
dis_x4_targ=Sqr((x4-x_target)^2+(y4-y_target)^2)
dis_x1_robot=Sqr((x1-x)^2+(y1-y)^2)
dis_x2_robot=Sqr((x2-x)^2+(y2-y)^2)
dis_x3_robot=Sqr((x3-x)^2+(y3-y)^2)
dis_x4_robot=Sqr((x4-x)^2+(y4-y)^2)
dis_targ_x1_robot=dis_x1_robot+dis_x1_targ
dis_targ_x2_robot=dis_x2_robot+dis_x2_targ
dis_targ_x3_robot=dis_x3_robot+dis_x3_targ
dis_targ_x4_robot=dis_x4_robot+dis_x4_targ
robot_new_angle=GetMobotTheta(0)
rot=RotationalDiff(1,X1_target,Y1_target)
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
dis_x1_robot=Sqr((x1-x)^2+(y1-y)^2)
dis_x2_robot=Sqr((x2-x)^2+(y2-y)^2)
dis_x3_robot=Sqr((x3-x)^2+(y3-y)^2)
```

```
dis_x4_robot=Sqr((x4-x)^2+(y4-y)^2)
If dis_targ1>0 And X1<X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1>Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1<X1_target And Y1>Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1<X1_target And Y1=Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1=Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1=X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1=X1_target And Y1>Y1_target Then
StepForward
End If
If dis_targ1<=0.1 And dis_targ<=0.1 Then Exit Do
' main Start main loop
            X=GetMobotX(0)      ' Present mobot coordinates (in meters)
            Y=GetMobotY(0)
            X_grid=CoordToGrid(0,X) ' indexes of cells where the
            Y_grid=CoordToGrid(0,Y)    ' mobot center is
                        ' Perform a range scan and update
                              ' the Certainty Grid (max. cell value=3)
rel_angle=Abs(GetSensorAngle(0,sen,1))
            Frx=0  ' Repulsive Force (x component)
            Fry=0  ' Repulsive Force (y component)
For i=Abs(X_grid-10) To X_grid+10
For j=Abs(Y_grid-10) To Y_grid+10
```

```
C=GetCell(0,i,j)
If C<>0  Then
If s0<.3 Or s1<.3 Or s2<.3 Or s3<.3 Or s4<.3 Or s5<.3 Then
d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=3*rel_angle/d^2
                              Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d
                              Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d
If GetCollisionAngle(0)<=180 Then
    SetSteering(0,-0.1,0)
    StepForward
    SetSteering(0,0,180)
For t=1 To 5
      StepForward
    Next
    SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
End If
Next
Next
      ' The target generates a constant-magnitud attracting force
            Fcx=Fct*(X_target-X)/dis_targ
            Fcy=Fct*(Y_target-Y)/dis_targ
            Rx=Frx+Fcx   ' Resultant Force Vector
            Ry=Fry+Fcy
            rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                            'current direction of travel and
                                                  'direction of vector R
            SetSteering(0,robotspeed,rot*3)    'mobot turns into the direction of R
```

```
                                    'at constant speed and steering rate proportional to the
                                    rotational difference
          StepForward               ' Dynamics simulation progresses one time step
If dis_targ <=.2 And dd-r_sum>=.01 Then
frx=0
fry=0
End If
        'sensing the first obstacle or edge closest to the robot
If S0>0 Or s1>0 Or S2>0 Or s3>0 Or S4>0 Or s5>0 Then Exit Do
Debug.Print CStr(s0)
Loop
xnew=GetMobotX(0)
ynew=GetMobotY(0)
rr_circum=Sqr((Xnew-X_target)^2+(Ynew-Y_target)^2)
'choosing a positioning distance for the virtual goal from the current postion of the robot
which would be used in the Next segment after an oscillatory motion is exhibited
cout9:
'approximating negative sensor readings to zero
'assuming that only one sensor activates first
If s0>0 And s1<0 And s2<0 And s3<0 And s4<0 And s5<0 Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
If s1>0 And s0<0 And s2<0 And s3<0 And s4<0 And s5<0 Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
If s2>0 And s0<0 And s1<0 And s3<0 And s4<0 And s5<0 Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
```

```
If s3>0 And s0<0 And s1<0 And s2<0 And s4<0 And s5<0 Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
If s4>0 And s0<0 And s1<0 And s2<0 And s3<0 And s5<0 Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
End If
If s5>0 And s1<0 And s2<0 And s3<0 And s4<0 And s0<0 Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
End If
'Assuming more than one sensor activates from either side of the robot at a given time
' a case of sensors 3 and 0
If CStr(s3)>0 And CStr(s0)>0 Then
If CStr(s3)<CStr(s0) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s3)>CStr(s0) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 3 and 1
If CStr(s3)>0 And CStr(s1)>0 Then
If CStr(s3)<CStr(s1) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s3)>CStr(s1) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
```

```
End If
End If
' a case of sensors 3 and 2
If CStr(s3)>0 And CStr(s2)>0 Then
If CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s3)>CStr(s2) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 3,1 and 0
If CStr(s3)>0 And CStr(s0)>0 And CStr(s1)>0 Then
If CStr(s3)<CStr(s0) And CStr(s3)<CStr(s1) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 3,2 and 0
If CStr(s3)>0 And CStr(s0)>0 And CStr(s2)>0 Then
If CStr(s3)<CStr(s0) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s0)<CStr(s3) And CStr(s0)<CStr(s2) Then
```

```
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 3,1 and 2
If CStr(s3)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s3)<CStr(s1) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s1)<CStr(s3) And CStr(s1)<CStr(s2) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 3,1,2 and 0
If CStr(s3)>0 And CStr(s0)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s3)<CStr(s1) And CStr(s3)<CStr(s2) And CStr(s3)<CStr(s0) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s3) And CStr(s1)<CStr(s0) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s3) And CStr(s2)<CStr(s0) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
```

```
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s2) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 4 and 0
If CStr(s4)>0 And CStr(s0)>0 Then
If CStr(s4)<CStr(s0) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s4)>CStr(s0) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 4 and 1
If CStr(s4)>0 And CStr(s1)>0 Then
If CStr(s4)<CStr(s1) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s4)>CStr(s1) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 4 and 2
If CStr(s4)>0 And CStr(s2)>0 Then
If CStr(s4)<CStr(s2) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s4)>CStr(s2) Then
```

```
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 4,1 and 0
If CStr(s4)>0 And CStr(s0)>0 And CStr(s1)>0 Then
If CStr(s4)<CStr(s0) And CStr(s4)<CStr(s1) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s4) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s4) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 4,2 and 0
If CStr(s4)>0 And CStr(s0)>0 And CStr(s2)>0 Then
If CStr(s4)<CStr(s0) And CStr(s4)<CStr(s2) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s0)<CStr(s2) And CStr(s0)<CStr(s4) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s4) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 4,1 and 2
```

```
If CStr(s4)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s4)<CStr(s1) And CStr(s4)<CStr(s2) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s4) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s4) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 4,1,2 and 0
If CStr(s4)>0 And CStr(s0)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s4)<CStr(s1) And CStr(s4)<CStr(s2) And CStr(s4)<CStr(s0) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s4) And CStr(s1)<CStr(s0) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s4) And CStr(s2)<CStr(s0) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s2) And CStr(s0)<CStr(s4) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 5 and 0
If CStr(s5)>0 And CStr(s0)>0 Then
If CStr(s5)<CStr(s0) Then
```

```
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s5)>CStr(s0) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 5 and 1
If CStr(s5)>0 And CStr(s1)>0 Then
If CStr(s5)<CStr(s1) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s5)>CStr(s1) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 5 and 2
If CStr(s5)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s2) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s5)>CStr(s2) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 5,1 and 0
If CStr(s5)>0 And CStr(s0)>0 And CStr(s1)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s1) Then
r_circum=CStr(s5)
```

```
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s5) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s5) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 5,2 and 0
If CStr(s5)>0 And CStr(s0)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s2) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s2) And CStr(s0)<CStr(s5) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s5) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 5,1 and 2
If CStr(s5)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s2) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s5) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s5) Then
```

```
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 5,1,2 and 0
If CStr(s5)>0 And CStr(s0)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s0) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s0) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s0) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s2) And CStr(s0)<CStr(s5) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 3,5 and 0
If CStr(s3)>0 And CStr(s5)>0 And CStr(s0)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s3)<CStr(s0) And CStr(s3)<CStr(s5) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s0)<CStr(s5) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
```

```
End If
End If
' a case of sensors 3,5 and 1
If CStr(s3)>0 And CStr(s5)>0 And CStr(s1)>0 Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s3)<CStr(s1) And CStr(s3)<CStr(s5) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s1)<CStr(s5) And CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 3,5 and 2
If CStr(s3)>0 And CStr(s5)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s2) And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s3)<CStr(s2) And CStr(s3)<CStr(s5) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s2)<CStr(s5) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 3,5,1 and 0
If CStr(s3)>0 And CStr(s5)>0 And CStr(s0)>0 And CStr(s1)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s1) And CStr(s5)<CStr(s3)Then
```

```
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s5) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s3)<CStr(s0) And CStr(s3)<CStr(s5) And CStr(s3)<CStr(s1) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,5,2 and 0
If CStr(s3)>0 And CStr(s5)>0 And CStr(s0)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s3)Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s2) And CStr(s0)<CStr(s5) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s3)<CStr(s0) And CStr(s3)<CStr(s5) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,5,1 and 2
If CStr(s3)>0 And CStr(s5)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s3)Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s3)<CStr(s1) And CStr(s3)<CStr(s5) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,5,0,1 and 2
If CStr(s3)>0 And CStr(s5)>0 And CStr(s1)>0 And CStr(s2)>0 And CStr(s0)>0Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s3) And CStr(s5)<CStr(s0)
Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s3) And
CStr(s1)<CStr(s0)Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s3) And
CStr(s2)<CStr(s0) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s3)<CStr(s1) And CStr(s3)<CStr(s5) And CStr(s3)<CStr(s2) And
CStr(s3)<CStr(s0) Then
r_circum=CStr(s3)
```

```
r_circum3=CStr(s3)
End If
End If
' a case of sensors 4,5 and 0
If CStr(s4)>0 And CStr(s5)>0 And CStr(s0)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s4) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s4)<CStr(s0) And CStr(s4)<CStr(s5) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s0)<CStr(s5) And CStr(s0)<CStr(s4) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 4,5 and 1
If CStr(s4)>0 And CStr(s5)>0 And CStr(s1)>0 Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s4) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s4)<CStr(s1) And CStr(s4)<CStr(s5) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s1)<CStr(s5) And CStr(s1)<CStr(s4) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 4,5 and 2
If CStr(s4)>0 And CStr(s5)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s2) And CStr(s5)<CStr(s4) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s4)<CStr(s2) And CStr(s4)<CStr(s5) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s2)<CStr(s5) And CStr(s2)<CStr(s4) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 4,5,1 and 0
If CStr(s4)>0 And CStr(s5)>0 And CStr(s0)>0 And CStr(s1)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s1) And CStr(s5)<CStr(s4)Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s5) And CStr(s0)<CStr(s4) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s4) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s4)<CStr(s0) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s1) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
End If
End If
' a case of sensors 4,5,2 and 0
If CStr(s4)>0 And CStr(s5)>0 And CStr(s0)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s4)Then
r_circum=CStr(s5)
```

```
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s2) And CStr(s0)<CStr(s5) And CStr(s0)<CStr(s4) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s4) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s4)<CStr(s0) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s2) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
End If
End If
' a case of sensors 4,5,1 and 2
If CStr(s4)>0 And CStr(s5)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s4)Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s4) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s4) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s4)<CStr(s1) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s2) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
End If
End If
' a case of sensors 4,5,0,1 and 2
If CStr(s4)>0 And CStr(s5)>0 And CStr(s1)>0 And CStr(s2)>0 And CStr(s0)>0Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s4) And CStr(s5)<CStr(s0)
Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s4) And
CStr(s1)<CStr(s0)Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s4) And
CStr(s2)<CStr(s0) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s4)<CStr(s1) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s2) And
CStr(s4)<CStr(s0) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
End If
End If
' a case of sensors 3,4 and 0
If CStr(s3)>0 And CStr(s4)>0 And CStr(s0)>0 Then
If CStr(s4)<CStr(s0) And CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s3)<CStr(s0) And CStr(s3)<CStr(s4) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s0)<CStr(s4) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 3,4 and 1
```

```
If CStr(s3)>0 And CStr(s4)>0 And CStr(s1)>0 Then
If CStr(s4)<CStr(s1) And CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s3)<CStr(s1) And CStr(s3)<CStr(s4) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s1)<CStr(s4) And CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
End If
End If
' a case of sensors 3,4 and 2
If CStr(s3)>0 And CStr(s4)>0 And CStr(s2)>0 Then
If CStr(s4)<CStr(s2) And CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s3)<CStr(s2) And CStr(s3)<CStr(s4) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s2)<CStr(s4) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
End If
End If
' a case of sensors 3,4,1 and 0
If CStr(s3)>0 And CStr(s4)>0 And CStr(s0)>0 And CStr(s1)>0 Then
If CStr(s4)<CStr(s0) And CStr(s4)<CStr(s1) And CStr(s4)<CStr(s3)Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s4) And CStr(s0)<CStr(s3) Then
```

```
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s4) And CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s3)<CStr(s0) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s1) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,2 and 0
If CStr(s3)>0 And CStr(s4)>0 And CStr(s0)>0 And CStr(s2)>0 Then
If CStr(s4)<CStr(s0) And CStr(s4)<CStr(s2) And CStr(s4)<CStr(s3)Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s0)<CStr(s2) And CStr(s0)<CStr(s4) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s4) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s3)<CStr(s0) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,1 and 2
If CStr(s3)>0 And CStr(s4)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s4)<CStr(s1) And CStr(s4)<CStr(s2) And CStr(s4)<CStr(s3)Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
```

```
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s4) And CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s4) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s3)<CStr(s1) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,0,1 and 2
If CStr(s3)>0 And CStr(s4)>0 And CStr(s1)>0 And CStr(s2)>0 And CStr(s0)>0 Then
If CStr(s4)<CStr(s1) And CStr(s4)<CStr(s2) And CStr(s4)<CStr(s3) And CStr(s4)<CStr(s0)
Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s4) And CStr(s1)<CStr(s3) And
CStr(s1)<CStr(s0)Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s4) And CStr(s2)<CStr(s3) And
CStr(s2)<CStr(s0) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s3)<CStr(s1) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s2) And
CStr(s3)<CStr(s0) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s4) And CStr(s0)<CStr(s2) And
CStr(s0)<CStr(s3) Then
```

```
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
' a case of sensors 3,4,5 and 0
If CStr(s4)>0 And CStr(s5)>0 And CStr(s3)>0 And CStr(s0)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s4) And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s4)<CStr(s0) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s0)<CStr(s5) And CStr(s0)<CStr(s4) And CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s3)<CStr(s5) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s0) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,5 and 1
If CStr(s4)>0 And CStr(s5)>0 And CStr(s1)>0 And CStr(s3)>0  Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s4)And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s4)<CStr(s1) And CStr(s4)<CStr(s5)And CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s1)<CStr(s5) And CStr(s1)<CStr(s4) And CStr(s1)<CStr(s3)Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
```

```
ElseIf CStr(s3)<CStr(s5) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s1) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,5 and 2
If CStr(s4)>0 And CStr(s5)>0 And CStr(s2)>0 And CStr(s3)>0 Then
If CStr(s5)<CStr(s2) And CStr(s5)<CStr(s4) And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s4)<CStr(s2) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s2)<CStr(s5) And CStr(s2)<CStr(s4) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s3)<CStr(s5) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,5,1 and 0
If CStr(s4)>0 And CStr(s5)>0 And CStr(s0)>0 And CStr(s1)>0 And CStr(s3)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s1) And CStr(s5)<CStr(s4) And CStr(s5)<CStr(s3)
Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s1) And CStr(s0)<CStr(s5) And CStr(s0)<CStr(s4) And
CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
```

```
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s4)And
CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s4)<CStr(s0) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s1) And
CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s3)<CStr(s5) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s0) And
CStr(s3)<CStr(s1) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,5,2 and 0
If CStr(s4)>0 And CStr(s5)>0 And CStr(s0)>0 And CStr(s2)>0 And CStr(s3)>0 Then
If CStr(s5)<CStr(s0) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s4) And CStr(s5)<CStr(s3)
Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s0)<CStr(s2) And CStr(s0)<CStr(s5) And CStr(s0)<CStr(s4) And
CStr(s0)<CStr(s3) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s4)And
CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s4)<CStr(s0) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s2) And
CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
```

```
r_circum4=CStr(s4)
ElseIf   CStr(s3)<CStr(s5)   And   CStr(s3)<CStr(s4)   And   CStr(s3)<CStr(s0)   And
CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
End If
End If
' a case of sensors 3,4,5,1 and 2
If CStr(s4)>0 And CStr(s5)>0 And CStr(s2)>0 And CStr(s1)>0 And CStr(s3)>0 Then
If CStr(s5)<CStr(s2) And CStr(s5)<CStr(s1) And CStr(s5)<CStr(s4) And CStr(s5)<CStr(s3)
Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf   CStr(s2)<CStr(s1)   And   CStr(s2)<CStr(s5)   And   CStr(s2)<CStr(s4)   And
CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf   CStr(s1)<CStr(s2)   And   CStr(s1)<CStr(s5)   And   CStr(s1)<CStr(s4)And
CStr(s1)<CStr(s3) Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf   CStr(s4)<CStr(s2)   And   CStr(s4)<CStr(s5)   And   CStr(s4)<CStr(s1)   And
CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf   CStr(s3)<CStr(s5)   And   CStr(s3)<CStr(s4)   And   CStr(s3)<CStr(s2)   And
CStr(s3)<CStr(s1) Then
r_circum=CStr(s3)
r_circum=CStr(s3)
End If
End If
```

```
' a case of sensors 3,4,5,0,1 and 2
If CStr(s4)>0 And CStr(s5)>0 And CStr(s1)>0 And CStr(s2)>0 And CStr(s0)>0 And
CStr(s3)>0 Then
If CStr(s5)<CStr(s1) And CStr(s5)<CStr(s2) And CStr(s5)<CStr(s4) And CStr(s5)<CStr(s0)
And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
r_circum5=CStr(s5)
ElseIf CStr(s1)<CStr(s2) And CStr(s1)<CStr(s5) And CStr(s1)<CStr(s4) And
CStr(s1)<CStr(s0) And CStr(s1)<CStr(s3)Then
r_circum=CStr(s1)
r_circum1=CStr(s1)
ElseIf CStr(s2)<CStr(s1) And CStr(s2)<CStr(s5) And CStr(s2)<CStr(s4) And
CStr(s2)<CStr(s0) And CStr(s2)<CStr(s3) Then
r_circum=CStr(s2)
r_circum2=CStr(s2)
ElseIf CStr(s4)<CStr(s1) And CStr(s4)<CStr(s5) And CStr(s4)<CStr(s2) And
CStr(s4)<CStr(s0) And CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
r_circum4=CStr(s4)
ElseIf CStr(s3)<CStr(s5) And CStr(s3)<CStr(s4) And CStr(s3)<CStr(s0) And
CStr(s3)<CStr(s1) And CStr(s3)<CStr(s2) Then
r_circum=CStr(s3)
r_circum3=CStr(s3)
ElseIf CStr(s0)<CStr(s5) And CStr(s0)<CStr(s4) And CStr(s0)<CStr(s3) And
CStr(s0)<CStr(s1) And CStr(s0)<CStr(s2) Then
r_circum=CStr(s0)
r_circum0=CStr(s0)
End If
End If
'Assuming more than one sensor activates from just one side of the robot.
' a case of sensors 3,4 and 5
```

```
If CStr(s4)>0 And CStr(s5)>0 And CStr(s3)>0 Then
If CStr(s5)<CStr(s4) And CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
ElseIf CStr(s4)<CStr(s3) And CStr(s4)<CStr(s5) Then
r_circum=CStr(s4)
ElseIf CStr(s3)<CStr(s4) And CStr(s3)<CStr(s5) Then
r_circum=CStr(s3)
End If
End If
' a case of sensors 3,4
If CStr(s4)>0 And CStr(s3)>0 Then
If CStr(s4)<CStr(s3) Then
r_circum=CStr(s4)
ElseIf CStr(s3)<CStr(s4) Then
r_circum=CStr(s3)
End If
End If
' a case of sensors 3,5
If CStr(s5)>0 And CStr(s3)>0 Then
If CStr(s5)<CStr(s3) Then
r_circum=CStr(s5)
ElseIf CStr(s3)<CStr(s5) Then
r_circum=CStr(s3)
End If
End If
' a case of sensors 4,5
If CStr(s5)>0 And CStr(s4)>0 Then
If CStr(s5)<CStr(s4) Then
r_circum=CStr(s5)
ElseIf CStr(s4)<CStr(s5) Then
r_circum=CStr(s4)
```

```
End If
End If
'Assuming more than one sensor activates from just one side of the robot.
' a case of sensors 0,1 and 2
If CStr(s0)>0 And CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s0)<CStr(s1) And CStr(s0)<CStr(s2) Then
r_circum=CStr(s0)
ElseIf CStr(s1)<CStr(s0) And CStr(s1)<CStr(s2) Then
r_circum=CStr(s1)
ElseIf CStr(s2)<CStr(s0) And CStr(s2)<CStr(s1) Then
r_circum=CStr(s2)
End If
End If
' a case of sensors 0,1
If CStr(s0)>0 And CStr(s1)>0 Then
If CStr(s0)<CStr(s1) Then
r_circum=CStr(s0)
ElseIf CStr(s1)<CStr(s0) Then
r_circum=CStr(s1)
End If
End If
' a case of sensors 1,2
If CStr(s1)>0 And CStr(s2)>0 Then
If CStr(s1)<CStr(s2) Then
r_circum=CStr(s1)
ElseIf CStr(s2)<CStr(s1) Then
r_circum=CStr(s2)
End If
End If
' a case of sensors 0,2
If CStr(s0)>0 And CStr(s2)>0 Then
If CStr(s0)<CStr(s2) Then
r_circum=CStr(s0)
ElseIf CStr(s2)<CStr(s0) Then
r_circum=CStr(s2)
End If
End If
robot_angle_first_obstacle=GetMobotTheta(0)

'******************PHASE THREE*******************************************
**
'*********NAVIGATION TOWARDS DETECTING LOCAL MINIMA SITUATION****
*
Do
x=GetMobotX(0)
y=GetMobotY(0)
x1=GetMobotX(1)
y1=GetMobotY(1)
x2=GetMobotX(2)
y2=GetMobotY(2)
x3=GetMobotX(3)
y3=GetMobotY(3)
x4=GetMobotX(4)
y4=GetMobotY(4)
dis_targ=Sqr((X-X_target)^2+(Y-Y_target)^2)
dis_x1_targ=Sqr((x1-x_target)^2+(y1-y_target)^2)
dis_x2_targ=Sqr((x2-x_target)^2+(y2-y_target)^2)
dis_x3_targ=Sqr((x3-x_target)^2+(y3-y_target)^2)
dis_x4_targ=Sqr((x4-x_target)^2+(y4-y_target)^2)
dis_x1_robot=Sqr((x1-x)^2+(y1-y)^2)
dis_x2_robot=Sqr((x2-x)^2+(y2-y)^2)
dis_x3_robot=Sqr((x3-x)^2+(y3-y)^2)
```

```
dis_x4_robot=Sqr((x4-x)^2+(y4-y)^2)
dis_targ_x1_robot=dis_x1_robot+dis_x1_targ
dis_targ_x2_robot=dis_x2_robot+dis_x2_targ
dis_targ_x3_robot=dis_x3_robot+dis_x3_targ
dis_targ_x4_robot=dis_x4_robot+dis_x4_targ
robot_new_angle=GetMobotTheta(0)
rot=RotationalDiff(1,X1_target,Y1_target)
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
dis_x1_robot=Sqr((x1-x)^2+(y1-y)^2)
dis_x2_robot=Sqr((x2-x)^2+(y2-y)^2)
dis_x3_robot=Sqr((x3-x)^2+(y3-y)^2)
dis_x4_robot=Sqr((x4-x)^2+(y4-y)^2)
If dis_targ1>0 And X1<X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1>Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1<Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1<X1_target And Y1>Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1<X1_target And Y1=Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1>X1_target And Y1=Y1_target Then
StepForward
ElseIf dis_targ1>0 And X1=X1_target And Y1<Y1_target Then
StepForward
```

```
ElseIf dis_targ1>0 And X1=X1_target And Y1>Y1_target Then
StepForward
End If
If dis_targ1<=0.1 And dis_targ<=0.1 Then Exit Do
' main Start main loop
                X=GetMobotX(0)      ' Present mobot coordinates (in meters)
                Y=GetMobotY(0)
                X_grid=CoordToGrid(0,X) ' indexes of cells where the
                Y_grid=CoordToGrid(0,Y)   ' mobot center is
                                ' Perform a range scan and update
                                    ' the Certainty Grid (max. cell value=3)
rel_angle=Abs(GetSensorAngle(0,sen,1))
                Frx=0  ' Repulsive Force (x component)
                Fry=0  ' Repulsive Force (y component)
For i=Abs(X_grid-10) To X_grid+10
For j=Abs(Y_grid-10) To Y_grid+10
C=GetCell(0,i,j)
If C<>0  Then
If s0<.3 Or s1<.3 Or s2<.3 Or s3<.3 Or s4<.3 Or s5<.3 Then
d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=3*rel_angle/d^2
                    Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d
                    Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d
If GetCollisionAngle(0)<=180 Then
   SetSteering(0,-0.1,0)
   StepForward
   SetSteering(0,0,180)
For t=1 To 5
     StepForward
   Next
```

```
          SetSteering(0,0.1,Rnd*30-15)
      End If
      End If
      End If
      End If
      Next
      Next
            ' The target generates a constant-magnitud attracting force
                  Fcx=(X_target-X)/dis_targ
                  Fcy=(Y_target-Y)/dis_targ
                  Rx=Frx+Fcx   ' Resultant Force Vector
                  Ry=Fry+Fcy
                  rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                           'current direction of travel and direction of vector R
                  SetSteering(0,robotspeed,rot*3)   'mobot turns into the direction of R
                                                     'at constant speed and steering rate
                                                     'proportional to the rotational difference
                  StepForward             ' Dynamics simulation progresses one time step
      If dis_targ <=.2 And dd-r_sum>=.01 Then
      frx=0
      fry=0
      End If
      'Capturing the robot's angle along the initial line of sight
      If dis_targ<.05 Then Exit All
      If  Abs(robot_angle-robot_new_angle)>=160  And  Abs(robot_angle-robot_new_angle)<=200
      Then Exit Do
      Debug.Print d
      Debug.Print d
      Loop
```

```
'*****************STAGE FOUR***********************************
**********VIRTUAL GOAL POSITIONING*************************
coute:
x_new=x
y_new=y
vx1=x_new+r_circum
vy1=y_new
vx2=x_new-r_circum
If vx2<0 Then
vx2=0
End If
vy2=y_new
vx3=x_new
vy3=y_new+r_circum
vx4=x_new
vy4=y_new-r_circum
If vy4<0 Then
vy4=0
End If
vx5=x_new+r_circum
vy5=y_new-r_circum
If vy5<0 Then
vy5=0
End If
vx6=x_new+r_circum
vy6=y_new+r_circum
vx7=x_new-r_circum
If vx7<0 Then
vx7=0
End If
vy7=y_new+r_circum
```

```
vx8=x_new-r_circum
If vx8<0 Then
vx8=0
End If
vy8=y_new-r_circum
If vy8<0 Then
vy8=0
End If
dis_target_v1=Sqr((X_target-vx1)^2+(y_target-vy1)^2)
dis_target_v2=Sqr((X_target-vx2)^2+(y_target-vy2)^2)
dis_target_v3=Sqr((X_target-vx3)^2+(y_target-vy3)^2)
dis_target_v4=Sqr((X_target-vx4)^2+(y_target-vy4)^2)
dis_target_v5=Sqr((X_target-vx5)^2+(y_target-vy5)^2)
dis_target_v6=Sqr((X_target-vx6)^2+(y_target-vy6)^2)
dis_target_v7=Sqr((X_target-vx7)^2+(y_target-vy7)^2)
dis_target_v8=Sqr((X_target-vx8)^2+(y_target-vy8)^2)
If dis_target_v1>dis_targ And dis_target_v2>dis_targ And dis_target_v1<dis_target_v3 And
dis_target_v1<dis_target_v6 And dis_target_v1<dis_target_v7 Then
If dis_target_v1>dis_targ And dis_target_v2>dis_targ And dis_target_v2<dis_target_v3 And
dis_target_v2<dis_target_v6 And dis_target_v2<dis_target_v7 Then
GoTo cout1a
End If
End If
If dis_target_v1>dis_targ And dis_target_v2>dis_targ And dis_target_v1<dis_target_v3 And
dis_target_v1<dis_target_v6 And dis_target_v1<dis_target_v7 Then
If dis_target_v1>dis_targ And dis_target_v2>dis_targ And dis_target_v2<dis_target_v3 And
dis_target_v2<dis_target_v6 And dis_target_v2<dis_target_v7 Then
GoTo cout1a
End If
End If
```

```
'*************CHECKING EDGES FOR VIRTUAL GOAL POSITIONING
'************** ALGORITHM TO POSITION VIRTUAL GOAL********************
**
cout1a:
'**CHOOSING BETWEEN POSITIONS #1 and #2 FOR VIRTUAL GOAL POSITIONING
***
'****A CASE WHERE ACTIVATED SENSOR(S) ARE FROM ONLY ONE SIDE*******
If r_circum0 > 0  Or r_circum1 > 0 Or r_circum2 > 0 And r_circum3 < 0  And r_circum4 < 0
And r_circum5 < 0 Then
GoTo cout1b
ElseIf r_circum3 > 0  Or r_circum4 > 0 Or r_circum5 > 0 And r_circum0 < 0 And r_circum1
< 0 And r_circum2 < 0 Then
GoTo cout1c
End If
'**CHOOSING BETWEEN POSITIONS #1 and #2 FOR VIRTUAL GOAL
POSITIONING**
'A CASE WHERE ACTIVATED SENSOR(S) ARE FROM EITHER SIDES OF THE
ROBOT
'A CASE OF SENSORS 3 AND 0
If r_circum0 > 0  And r_circum3 > 0 Then
If r_circum0< r_circum3 Then
GoTo cout1b
ElseIf r_circum0> r_circum3 Then
GoTo cout1c
End If
End If
'A CASE OF SENSORS 3 AND 1
If r_circum1 > 0  And r_circum3 > 0 Then
If r_circum1< r_circum3 Then
GoTo cout1b
ElseIf r_circum1> r_circum3 Then
```

```
GoTo coutlc
End If
End If

'A CASE OF SENSORS 3 AND 2
If r_circum2 > 0  And r_circum3 > 0 Then
If r_circum2< r_circum3 Then
GoTo coutlb
ElseIf r_circum2> r_circum3 Then
GoTo coutlc
End If
End If
'A CASE OF SENSORS 3,1 AND 0
If r_circum3 > 0  And r_circum1 > 0 And r_circum0 > 0 Then
If r_circum3< r_circum1 And r_circum0 Then
GoTo coutlc
ElseIf r_circum0 Or r_circum1< r_circum3 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 3,2 AND 0
If r_circum3 > 0  And r_circum2 > 0 And r_circum0 > 0 Then
If r_circum3< r_circum2 And r_circum0 Then
GoTo coutlc
ElseIf r_circum0 Or r_circum2< r_circum3 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 3,1 AND 2
If r_circum3 > 0  And r_circum2 > 0 And r_circum1 > 0 Then
If r_circum3< r_circum2 And r_circum0 Then
```

```
GoTo cout1c
ElseIf r_circum1 Or r_circum2< r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,1,2 AND 0
If r_circum3 > 0  And r_circum2 > 0 And r_circum1 > 0 And r_circum0 > 0 Then
If r_circum3< r_circum2 And r_circum0 And r_circum1 Then
GoTo cout1c
ElseIf r_circum1 Or r_circum2 Or r_circum0 < r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4 AND 0
If r_circum4 > 0  And r_circum0 > 0 Then
If r_circum4< r_circum0 Then
GoTo cout1c
ElseIf r_circum0 < r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4 AND 1
If r_circum4 > 0  And r_circum1 > 0 Then
If r_circum4< r_circum1 Then
GoTo cout1c
ElseIf r_circum1 < r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4 AND 2
If r_circum4 > 0  And r_circum2 > 0 Then
If r_circum4< r_circum2 Then
GoTo cout1c
ElseIf r_circum2 < r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4,1 AND 0
If r_circum4 > 0  And r_circum0 > 0 And r_circum1 > 0 Then
If r_circum4< r_circum1  And r_circum0 Then
GoTo cout1c
ElseIf r_circum1 Or r_circum0 < r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4,2 AND 0
If r_circum4 > 0  And r_circum0 > 0 And r_circum2 > 0 Then
If r_circum4< r_circum2 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum0 < r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4,2 AND 1
If r_circum4 > 0  And r_circum1 > 0 And r_circum2 > 0 Then
If r_circum4< r_circum2 And r_circum1 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 < r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4,2,1 AND 0
```

```
If r_circum4 > 0  And r_circum1 > 0 And r_circum2 > 0 Then
If r_circum4< r_circum2 And r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 Or r_circum0 < r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5 AND 0
If r_circum5 > 0  And r_circum0 > 0 Then
If r_circum5< r_circum0 Then
GoTo cout1c
ElseIf r_circum0 < r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5 AND 1
If r_circum5 > 0  And r_circum1 > 0 Then
If r_circum5< r_circum1 Then
GoTo cout1c
ElseIf r_circum1 < r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5 AND 2
If r_circum5 > 0  And r_circum2 > 0 Then
If r_circum5< r_circum2 Then
GoTo cout1c
ElseIf r_circum2 < r_circum5 Then
GoTo cout1b
End If
End If
```

```
'A CASE OF SENSORS 5,1 AND 0
If r_circum5 > 0  And r_circum0 > 0 And r_circum1 > 0 Then
If r_circum5< r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum1 Or r_circum0 < r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5,2 AND 0
If r_circum5 > 0  And r_circum0 > 0 And r_circum2 > 0 Then
If r_circum5< r_circum2 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum0 < r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5,2 AND 1
If r_circum5 > 0  And r_circum1 > 0 And r_circum2 > 0 Then
If r_circum5< r_circum2 And r_circum1 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 < r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5,2,1 AND 0
If r_circum5 > 0 And r_circum0 >0 And r_circum1 > 0 And r_circum2 > 0 Then
If r_circum5< r_circum2 And r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 Or r_circum0 < r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,5 AND 0
If r_circum5 > 0 And r_circum3>0 And r_circum0 > 0 Then
If r_circum5 Or  r_circum3 < r_circum0 Then
GoTo cout1c
ElseIf r_circum0 < r_circum5 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,5 AND 1
If r_circum5 > 0 And r_circum3 >0 And r_circum1 > 0 Then
If r_circum5 Or r_circum3< r_circum1 Then
GoTo cout1c
ElseIf r_circum1 < r_circum5 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,5 AND 2
If r_circum5 > 0 And r_circum3 >0 And r_circum2 > 0 Then
If r_circum5 Or r_circum3< r_circum2 Then
GoTo cout1c
ElseIf r_circum2 < r_circum5 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,5,1 AND 0
If r_circum5 > 0  And r_circum3 > 0 And r_circum0 > 0 And r_circum1 > 0 Then
If r_circum5 Or r_circum3 < r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum1 Or r_circum0 < r_circum5 And r_circum3 Then
GoTo cout1b
```

```
End If
End If
'A CASE OF SENSORS 3,5,2 AND 0
If r_circum5 > 0  And r_circum3 > 0 And r_circum0 > 0 And r_circum2 > 0 Then
If r_circum5 Or r_circum3 < r_circum2 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum0 < r_circum5 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,5,2 AND 1
If r_circum5 > 0  And r_circum3 > 0 And r_circum1 > 0 And r_circum2 > 0 Then
If r_circum5 Or r_circum3< r_circum2 And r_circum1 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 < r_circum5 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,5,2,1 AND 0
If r_circum5 > 0  And r_circum1 > 0 And r_circum0 > 0 And r_circum3 >0 And r_circum2 >
0 Then
If r_circum5 Or r_circum3 < r_circum2 And r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 Or r_circum0 < r_circum5 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4,5 AND 0
If r_circum5 > 0 And r_circum4>0 And r_circum0 > 0 Then
If r_circum5 Or  r_circum4 < r_circum0 Then
GoTo cout1c
```

```
ElseIf r_circum0 < r_circum5 And r_circum4 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 4,5 AND 1
If r_circum5 > 0 And r_circum4 >0 And r_circum1 > 0 Then
If r_circum5 Or r_circum4< r_circum1 Then
GoTo coutlc
ElseIf r_circum1 < r_circum5 And r_circum4 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 4,5 AND 2
If r_circum5 > 0 And r_circum4 >0 And r_circum2 > 0 Then
If r_circum5 Or r_circum4< r_circum2 Then
GoTo coutlc
ElseIf r_circum2 < r_circum5 And r_circum4 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 4,5,1 AND 0
If r_circum5 > 0  And r_circum4 > 0 And r_circum0 > 0 And r_circum1 > 0 Then
If r_circum5 Or r_circum4 < r_circum1 And r_circum0 Then
GoTo coutlc
ElseIf r_circum1 Or r_circum0 < r_circum5 And r_circum4 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 4,5,2 AND 0
If r_circum5 > 0  And r_circum4 > 0 And r_circum0 > 0 And r_circum2 > 0 Then
If r_circum5 Or r_circum4 < r_circum2 And r_circum0 Then
```

```
GoTo cout1c
ElseIf r_circum2 Or r_circum0 < r_circum5 And r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4,5,2 AND 1
If r_circum5 > 0  And r_circum4 > 0 And r_circum1 > 0 And r_circum2 > 0 Then
If r_circum5 Or r_circum4< r_circum2 And r_circum1 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 < r_circum5 And r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 4,5,2,1 AND 0
If r_circum5 > 0  And r_circum1 > 0 And r_circum0 > 0 And r_circum4 > 0 And r_circum2 >
0 Then
If r_circum5 Or r_circum4 < r_circum2 And r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 Or r_circum0 < r_circum5 And r_circum4 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,4 AND 0
If r_circum4 > 0 And r_circum3>0 And r_circum0 > 0 Then
If r_circum4 Or  r_circum3 < r_circum0 Then
GoTo cout1c
ElseIf r_circum0 < r_circum4 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,4 AND 1
```

```
If r_circum4 > 0 And r_circum3 >0 And r_circum1 > 0 Then
If r_circum4 Or r_circum3< r_circum1 Then
GoTo cout1c
ElseIf r_circum1 < r_circum4 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,4 AND 2
If r_circum4 > 0 And r_circum3 >0 And r_circum2 > 0 Then
If r_circum4 Or r_circum3< r_circum2 Then
GoTo cout1c
ElseIf r_circum2 < r_circum4 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,4,1 AND 0
If r_circum4 > 0  And r_circum3 > 0 And r_circum0 > 0 And r_circum1 > 0 Then
If r_circum4 Or r_circum3 < r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum1 Or r_circum0 < r_circum4 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,4,2 AND 0
If r_circum4 > 0  And r_circum3 > 0 And r_circum0 > 0 And r_circum2 > 0 Then
If r_circum4 Or r_circum3 < r_circum2 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum0 < r_circum4 And r_circum3 Then
GoTo cout1b
End If
End If
```

```
'A CASE OF SENSORS 3,4,2 AND 1
If r_circum4 > 0  And r_circum3 > 0 And r_circum1 > 0 And r_circum2 > 0 Then
If r_circum4 Or r_circum3< r_circum2 And r_circum1 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 < r_circum4 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,4,2,1 AND 0
If r_circum4 > 0  And r_circum1 > 0 And r_circum0 > 0 And r_circum3 > 0 And r_circum2 >
0 Then
If r_circum4 Or r_circum3 < r_circum2 And r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 Or r_circum0 < r_circum4 And r_circum3 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5,3,4 AND 0
If r_circum5 > 0  And r_circum4 > 0 And r_circum3 > 0 And r_circum0 > 0 Then
If r_circum4 Or  r_circum3 Or r_circum5 < r_circum0 Then
GoTo cout1c
ElseIf r_circum0 < r_circum4 And r_circum3 And r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 5,3,4 AND 1
If r_circum5 > 0  And r_circum4 > 0 And r_circum3 >0 And r_circum1 > 0 Then
If r_circum5 Or r_circum4 Or r_circum3< r_circum1 Then
GoTo cout1c
ElseIf r_circum1 < r_circum5 And r_circum4 And r_circum3 Then
GoTo cout1b
```

```
End If
End If
'A CASE OF SENSORS 5,3,4 AND 2
If r_circum5 > 0  And r_circum4 > 0 And r_circum3 >0 And r_circum2 > 0 Then
If r_circum4 Or r_circum3 Or r_circum5 < r_circum2 Then
GoTo coutlc
ElseIf r_circum2 < r_circum4 And r_circum3 And r_circum5 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 5,3,4,1 AND 0
If r_circum5 > 0  And r_circum4 > 0  And r_circum3 > 0 And r_circum0 > 0 And r_circum1
> 0 Then
If r_circum4 Or r_circum5 Or r_circum3 < r_circum1 And r_circum0 Then
GoTo coutlc
ElseIf r_circum1 Or r_circum0 < r_circum4 And r_circum3 And r_circum5 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 5,3,4,2 AND 0
If r_circum5 > 0  And r_circum4 > 0  And r_circum3 > 0 And r_circum0 > 0 And r_circum2
> 0 Then
If r_circum4 Or r_circum3 Or r_circum5 < r_circum2 And r_circum0 Then
GoTo coutlc
ElseIf r_circum2 Or r_circum0 < r_circum4 And r_circum3 And r_circum5 Then
GoTo coutlb
End If
End If
'A CASE OF SENSORS 5,3,4,2 AND 1
If r_circum5 > 0  And r_circum4 > 0  And r_circum3 > 0 And r_circum1 > 0 And r_circum2
> 0 Then
```

```
If r_circum5 Or r_circum4 Or r_circum3< r_circum2 And r_circum1 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 < r_circum4 And r_circum3 And r_circum5 Then
GoTo cout1b
End If
End If
'A CASE OF SENSORS 3,4,5,2,1 AND 0
If r_circum5 > 0  And r_circum4 > 0  And r_circum1 > 0 And r_circum0 > 0 And r_circum3
> 0 And r_circum2 > 0 Then
If r_circum5 Or r_circum4 Or r_circum3 < r_circum2 And r_circum1 And r_circum0 Then
GoTo cout1c
ElseIf r_circum2 Or r_circum1 Or r_circum0 < r_circum4 And r_circum3 And r_circum5
Then
GoTo cout1b
End If
End If
cout1b:
vx1=Abs(x+r_circum)
vy1=Abs(y)
Do
SetSensorRange(0,.05,.3)
SetMarkPosition(0,vx1,vy1)
X=GetMobotX(0)      ' Present mobot coordinates (in meters)
                Y=GetMobotY(0)
        X_grid=CoordToGrid(0,X) ' indexes of cells where the
                Y_grid=CoordToGrid(0,Y)    ' mobot center is
                                ' Perform a range scan and update
                                ' the Certainty Grid (max. cell value=3)
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
```

```
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
rel_angle=Abs(GetSensorAngle(0,sen,1))
            Frx=0  ' Repulsive Force (x component)
            Fry=0  ' Repulsive Force (y component)
dis_targ=Sqr((X-vx1)^2+(Y-vy1)^2)
            ' Each occupied cell inside the windows applies a repulsive force to the mobot.
            For i=Abs(X_grid-10) To X_grid+10
                   For j=Abs(Y_grid-10) To Y_grid+10
            C=GetCell(0,i,j)
X_target=GetMarkX(0) 'Target coordinates (mark 0)
Y_target=GetMarkY(0)
robot_new_angle=GetMobotTheta(0)
If C<>0 Then
If s0<.3 Or s1<.3 Or s2<.3 Or s3<.3 Or s4<.3 Or s5<.3 Then
d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=3*rel_angle/d^2
                   Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d
                   Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d
If GetCollisionAngle(0)<=180 Then
    SetSteering(0,0.1,0)
    StepForward
    SetSteering(0,0,180)
For t=1 To 5
      StepForward
    Next
SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
End If
Next
Next
' The target generates a constant-magnitude attracting force
          Fcx=Fct*(vx1-X)/dis_targ
          Fcy=Fct*(vy1-Y)/dis_targ
          Rx=Frx+Fcx   ' Resultant Force Vector
          Ry=Fry+Fcy
          rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                         'current direction of travel and direction of vector R
          SetSteering(0,robotspeed,rot*3)      'mobot turns into the direction of R
                         'at constant speed and steering rate proportional to the
                         rotational difference
          StepForward        ' Dynamics simulation progresses one time step
sim_time=GetSimulationTime
Debug.Print frx
Debug.Print fry
If dis_targ<.05 Then
GoTo couta
ElseIf    Abs(robot_angle-robot_new_angle)>=155     And     Abs(robot_angle-
robot_new_angle)<=205 Then
GoTo coutw
End If
Loop
coutw:
x=GetMobotX(0)
y=GetMobotY(0)
r_circum=r_circum
GoTo cout1b
'*************************
```

```
coutlc:
vx2=Abs(x-r_circum)
vy2=Abs(y)
Do
SetSensorRange(0,.05,.3)
SetMarkPosition(0,vx2,vy2)
dis_radius_robot=Sqr((x_new-x)^2+(y_new-y)^2)
robot_new_angle=GetMobotTheta(0)
X=GetMobotX(0)      ' Present mobot coordinates (in meters)
            Y=GetMobotY(0)
      X_grid=CoordToGrid(0,X) ' indexes of cells where the
            Y_grid=CoordToGrid(0,Y)    ' mobot center is
                        ' Perform a range scan and update
                                    ' the Certainty Grid (max. cell value=3)
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
s5=MeasureRange(0,5,3)
rel_angle=Abs(GetSensorAngle(0,sen,1))
            Frx=0  ' Repulsive Force (x component)
            Fry=0  ' Repulsive Force (y component)
dis_targ=Sqr((X-vx2)^2+(Y-vy2)^2)
            ' Each occupied cell inside the windows applies a repulsive force to the mobot.
            For i=Abs(X_grid-10) To X_grid+10
                  For j=Abs(Y_grid-10) To Y_grid+10
            C=GetCell(0,i,j)
If C<>0 And dis_targ>0 Then
X_target=GetMarkX(0) 'Target coordinates (mark 0)
Y_target=GetMarkY(0)
```

```
If s0<.3 Or s1<.3 Or s2<.3 Or s3<.3 Or s4<.3 Or s5<.3 Then
d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=3*rel_angle/d^2
                    Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d
                    Fry=Fry+Fcr/3*C/d^2*(y_grid-j)/d
If GetCollisionAngle(0)<=180 Then
    SetSteering(0,-0.1,0)
    StepForward
    SetSteering(0,0,180)
For t=1 To 5
      StepForward
    Next
    SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
End If
Next
Next
' The target generates a constant-magnitude attracting force
             Fcx=(vx2-x)/dis_targ
             Fcy=(vy2-y)/dis_targ
             Rx=Frx+Fcx   ' Resultant Force Vector
             Ry=Fry+Fcy
             rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                 'current direction of travel and direction of vector R
             SetSteering(0,robotspeed,rot*3)        'mobot turns into the direction of R
                                               'at constant speed and steering rate
                                        'proportional to the rotational difference
             StepForward            ' Dynamics simulation progresses one time step
```

```
sim_time=GetSimulationTime
Debug.Print frx
Debug.Print fry
If dis_targ<.05 Then
GoTo couta
ElseIf      Abs(robot_angle-robot_new_angle)>=155      And      Abs(robot_angle-
robot_new_angle)<=210 Then
GoTo coutwc
End If
Loop
coutwc:
x=GetMobotX(0)
y=GetMobotY(0)
r_circum=r_circum+.002
GoTo cout1c
couta:
xnewa=GetMobotX(0)
ynewa=GetMobotY(0)
Do
SetMarkPosition(0,a,b)
X=GetMobotX(0)     ' Present mobot coordinates (in meters)
Y=GetMobotY(0)
X_grid=CoordToGrid(0,X) ' indexes of cells where the
Y_grid=CoordToGrid(0,Y)   ' mobot center is
              ' Perform a range scan and update
                        ' the Certainty Grid (max. cell value=3)
s0=MeasureRange(0,0,3)
s1=MeasureRange(0,1,3)
s2=MeasureRange(0,2,3)
s3=MeasureRange(0,3,3)
s4=MeasureRange(0,4,3)
```

```
s5=MeasureRange(0,5,3)
robot_new_angle=GetMobotTheta(0)
obstacle_new_angle=GetMobotTheta(1)
rel_angle=Abs(GetSensorAngle(0,sen,1))
Frx=0  ' Repulsive Force (x component)
Fry=0  ' Repulsive Force (y component)
dis_targ=Sqr((X-a)^2+(Y-b)^2)
                ' Each occupied cell inside the windows applies a repulsive force to the mobot.
For i=Abs(X_grid-10) To X_grid+10
For j=Abs(Y_grid-10) To Y_grid+10
C=GetCell(0,i,j)
If C<>0 And dis_targ>0 Then
X_target=GetMarkX(0) 'Target coordinates (mark 0)
Y_target=GetMarkY(0)
If s0<.3 Or s1<.3 Or s2<.3 Or s3<.3 Or s4<.3 Or s5<.3 Then
d=Sqr((X_grid-i)^2+(Y_grid-j)^2)
If d<>0 Then
Fcr=3*rel_angle/d^2
Frx=Frx+Fcr/3*C/d^2*(X_grid-i)/d
Fry=Fry+Fcr/3*C/d^2*(Y_grid-j)/d
If GetCollisionAngle(0)<=180 Then
SetSteering(0,-0.1,0)
StepForward
SetSteering(0,0,180)
For t=1 To 5
      StepForward
    Next
    SetSteering(0,0.1,Rnd*30-15)
End If
End If
End If
End If
Next
Next
If dis_targ <=.2 And dd-r_sum<=.2 Then
frx=0
fry=0
End If
dis_targ_a=Sqr((x-xnewa)^2+(y-ynewa)^2)
' The target generates a constant-magnitude attracting force
            Fcx=Fct*(a-X)/dis_targ
            Fcy=Fct*(b-Y)/dis_targ
            Rx=Frx+Fcx   ' Resultant Force Vector
            Ry=Fry+Fcy
            rot=RotationalDiff(0,X+Rx,Y+Ry) 'shortest rotational difference between
                                            'current direction of travel and
                                                   'direction of vector R
            SetSteering(0,robotspeed,rot*3)      'mobot turns into the direction of R
                                        'at constant speed and steering rate
                                            'proportional to the rotational difference
            StepForward          ' Dynamics simulation progresses one time step
sim_time=GetSimulationTime
If theta_x1=360 Then
If x<x1 Or x<x2 Or x<x3 Or x<x4 Then
robotspeed=robotspeed1
End If
End If
If dis_targ<=.05 Then Exit Do
Debug.Print d
If Abs(robot_angle-robot_new_angle)<=20 And dis_targ_a>=.4 Then
If  Abs(robot_angle-robot_new_angle)>=155  And  Abs(robot_angle-robot_new_angle)<=210
Then
```

```
GoTo coutr
End If
End If
Loop
End Sub
```

APPENDIX IV

PROGRAMME FOR THE REAL ROBOT

```
Option Explicit
public const pi as single=3.14159
public hour as byte
public minute as byte
public second as single
const an13 as byte=13 '\\\leftsen
const an15 as byte=15 '\\\leftsen
const an17 as byte=17 '\\\leftsen
const an18 as byte=18 '\\\leftsen
const an19 as byte=19 '\\\leftsen
dim analogoutput13 as integer
dim analogoutput15 as integer
dim analogoutput17 as integer
dim analogoutput18 as integer
dim analogoutput19 as integer
dim serialoutput as byte
dim pwoutput as byte
dim avalue13 as integer
dim avalue15 as integer
dim avalue17 as integer
dim avalue18 as integer
dim avalue19 as integer
dim i1 as single,theta as single, numr_steps as single, rad as single,alpha1 as single,alpha2deg
as single,length13_15 as single,length13_17 as single,length13_18 as single
dim gamma as single,beta as single,alpha as single,dis_targ as single,rot_diff as
single,length13_19 as single,length15_13 as single,length15_17 as single
```

```
dim dis_targ1 as single, new_theta as single,L_new as single,L_total as single,rho as single,e
as single,length15_18 as single,length15_19  as single,length17_13 as single
dim xnew_robot as single, ynew_robot as single, L as single, counter as single,L3 as single,
L2 as single,cone_length17 as single,length17_15 as single,length17_18 as single
dim fcx as single, fcy as single, frx as single, fry as single, rx as single, ry as single,
cone_length13 as single,length17_19 as single,length18_13 as single,length18_15 as single
dim frx13 as single, fry13 as single, frx15 as single, fry15 as single,stp as single,c as
single,cone_length15 as single,length18_17 as single,length18_19 as single,length19_13 as
single
dim a13 as single, a15 as single, frx17 as single, fry17 as single, a17 as single,a18 as
single,cone_length18 as single,length19_15 as single,length19_17 as single,length19_18 as
single
dim frx18 as single, frx19 as single,divisor as single, denominator as single, numerator as
single, fry19 as single,cone_length19 as single,cone_length as single
dim dis_targAB as single, dis_targOA as single, dis_targOB as single
const degperstepl as single=1.0
const degperstepr as single=1.0
const min_sen_range as single=10.0
const length_per_step as single=0.24
const rad2deg as single= 57.2958

'************initial robot coordinate******************
public const x_robot as single=0.0
public const y_robot as single=0.0

'************input target point************************
public const x_target as single=10.0
public const y_target as single=10.0
```

```
Public Sub Main()
call delay(1.0)
const lws14 as single=0.00157986 'left wheel speed quad 1 and 4
const rws14 as single=0.00157986 'right wheel speed quad 1 and 4
const lws23 as single=0.00157986 'left wheel speed quad 2 and 3
const rws23 as single=0.00157986 'right wheel speed quad 2 and 3
call delay(0.0)

'QUADRANT ONE ORIENTATION
if x_target > x_robot then
if y_target > y_robot then
rad= atn(x_target/y_target) ' angle orientated by robot in rad
theta=rad2deg*rad ' angle orientated by robot in degree
numr_steps=theta/degperstepr 'conversion of theta to orientation steps: right rotation
alpha=90.0-theta 'current robot orientation on the quadrant system
i1=0.0
end if
end if
do until i1>=numr_steps
call delay(0.02)
call pulseout (5,0.001583,1)
call pulseout (6,0.001528,1)
i1=i1+1.0
loop

'QUADRANT TWO ORIENTATION
if x_target < x_robot then
if y_target > y_robot then
rad= atn(abs(x_target)/y_target) ' angle orientated by robot in rad
theta=rad2deg*rad ' angle orientated by robot in degree
numr_steps=theta/degperstepr 'conversion of theta to orientation steps: right rotation
```

```
alpha=90.0+theta 'current robot orientation on the quadrant system
i1=0.0
end if
end if
do until i1>=numr_steps
call delay(0.02)
call pulseout (5,0.001483,1)
call pulseout (6,0.001483,1)
i1=i1+1.0
loop

'QUADRANT THREE ORIENTATION
if x_target < x_robot then
if y_target < y_robot then
rad= atn(abs(y_target)/abs(x_target)) ' angle orientated by robot in rad
theta=rad2deg*rad ' angle orientated by robot in degree
theta=90.0+theta
numr_steps=theta/degperstepr 'conversion of theta to orientation steps: right rotation
alpha=90.0-theta 'current robot orientation on the quadrant system
i1=0.0
end if
end if
do until i1>=numr_steps
call delay(0.02)
call pulseout (5,0.001483,1)
call pulseout (6,0.001483,1)
i1=i1+1.0
loop

'QUADRANT FOUR ORIENTATION
if x_target > x_robot then
if y_target < y_robot then
rad= atn(abs(y_target)/abs(x_target)) ' angle orientated by robot in rad
theta=rad2deg*rad ' angle orientated by robot in degree
theta=90.0+theta
numr_steps=theta/degperstepr 'conversion of theta to orientation steps: right rotation
alpha=90.0-theta 'current robot orientation on the quadrant system
i1=0.0
end if
end if
do until i1>=numr_steps
call delay(0.02)
call pulseout (5,0.001583,1)
call pulseout (6,0.001528,1)
i1=i1+1.0
loop
call translation()
end sub

'translation and orientation within quadrant one
sub translation()
xnew_robot=0.0
ynew_robot=0.0
counter=0.0
do
analogoutput13= rangea13 'get the range
analogoutput15= rangea15 'get the range
analogoutput17= rangea17 'get the range
analogoutput18= rangea18 'get the range
analogoutput19= rangea19 'get the range
L=length_per_step*counter 'one translation step to distance
xnew_robot=L*sin(theta)
```

```
ynew_robot=L*cos(theta)
dis_targ=sqr((xnew_robot-x_target)^2+(ynew_robot-y_target)^2)
call delay(0.02)
call pulseout(5,0.002,1)'forward left wheel
call pulseout(6,0.001,1)'forward right wheel
counter=counter+1.0
if analogoutput13 <6 then
call vff()
end if
if analogoutput15 <6 then
call vff()
end if

if analogoutput17 <6 then
call vff()
end if

if analogoutput18 <6 then
call vff()
end if

if analogoutput19 <6 then
call vff()
end if
loop
end sub

'***********vff************
sub vff()
xnew_robot=0.0
ynew_robot=0.0
```

```
counter=0.0
L=0.0
dis_targ=sqr((xnew_robot-x_target)^2+(ynew_robot-y_target)^2)
fcx=x_target-xnew_robot/dis_targ
fcy=y_target-ynew_robot/dis_targ

frx13=(xnew_robot-x_target)*((1.0/csng(analogoutput13))-(1.0/min_sen_range))
frx15=(xnew_robot-x_target)*((1.0/csng(analogoutput15))-(1.0/min_sen_range))
frx17=(xnew_robot-x_target)*((1.0/csng(analogoutput17))-(1.0/min_sen_range))
frx18=(xnew_robot-x_target)*((1.0/csng(analogoutput18))-(1.0/min_sen_range))
frx19=(xnew_robot-x_target)*((1.0/csng(analogoutput19))-(1.0/min_sen_range))
rx=fcx+frx
ry=fcy+fry
xnew_robot=xnew_robot+rx
ynew_robot=ynew_robot+ry
end sub
'***********sensor detecting functions****************
function rangea13() as integer
dim avalue13 as integer
avalue13=getADC(an13)
RANGEA13=(AVALUE13\2)
end function

function rangea15() as integer
dim avalue15 as integer
avalue15=getADC(an15)
RANGEA15=(AVALUE15\2)
end function

function rangea17() as integer
dim avalue17 as integer
```

```
avalue17=getADC(an17)
RANGEA17=(AVALUE17\2)
end function

function rangea18() as integer
dim avalue18 as integer
avalue18=getADC(an18)
RANGEA18=(AVALUE18\2)
end function

function rangea19() as integer
dim avalue19 as integer
avalue19=getADC(an19)
RANGEA19=(AVALUE19\2)
end function
```