VISUAL SERVOING OF MOBILE ROBOTS

USING POTENTIAL FIELDS

by

MUHAMMET BAŞTAN

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

SABANCI UNIVERSITY

Spring 2004

VISUAL SERVOING OF MOBILE ROBOTS

USING POTENTIAL FIELDS

APPROVED BY:

Assistant Prof. Dr. AHMET ONAT …….

(Thesis Supervisor)

Assistant Prof. Dr. AYHAN BOZKURT …….

Prof. Dr. ASIF SABANOVIC ……..

DATE OF APPROVAL: ……

ABSTRACT

The popularity of autonomous mobile robots have been rapidly increasing due to their new emerging application areas, from room cleaning, tourist guidance to space explorations. However, the development of a satisfactory control algorithm that will enable the autonomous mobile robots to navigate safely especially in dynamic environments is still an open research problem.

In this work, a newly proposed potential field based control method is implemented, analyzed, and improvements are suggested based on experimental results obtained from a real robot.

The experimental system, planned to be the groundwork for robot soccer team, consists of an overhead global vision camera, personal computer, wireless module, and a non-holonomic mobile robot, which is supposed to navigate safely among obstacles and reach its goal point. Images of the robot's operating area are acquired by the camera, color processed by the PC in real-time to retrieve the required position and orientation information of the robot, goal, and obstacles. Then, the control algorithm is applied, and resulting reference wheel velocities are sent to the robot via wireless link. Finally, robot has to drive with these reference velocities for safe navigation and goal tracking.

Experimental results in the form of motion history images, graphs and movies are presented to demonstrate the successful as well as problematic aspects of the implemented algorithm and the setup.

ÖZET

Otonom gezer robotların, oda temizliği ve turist rehberliğinden uzay araştırmalarına kadar değişik uygulama alanlarında kullanılmaya başlanması popülerliklerini artırmıştır. Ancak, bu robotların özellikle dinamik ortamlarda güvenle hareket etmesini sağlayabilecek tatmin edici düzeyde kontrol algoritmalarının geliştirilmesi hala araştırma konusudur.

Bu çalışmada, önceden önerilmiş "potansiyel alan" tabanlı bir kontrol metodu analiz edilmiş ve gerçek bir robot ile yapılan deneysel verilere dayanarak iyileştirmeler önerilmiştir.

Bir robot futbol takımı için altyapı olarak planlanan deneysel sistem bütün ortamı görmeyi sağlayacak bir kamera, bilgisayar, telsiz iletişim modülü ve holonom olmayan iki tekerlekli, ve engeller arasında güvenle dolaşıp hedefine ulaşmyı amaçlayan bir robottan oluşmaktadır. Robotun çalışma alanının görüntüleri kamera tarafından alınıp, robotun ueri, hedef ve engellerin uzaklık ve yönünü bulmak için bilgisayar tarafından gerçek zamanlı renkli görüntü işlemeye tabi tutulur. Daha sonra, elde edilen hedef ve engel bilgilerine kontrol algoritması uygulanıp bulunan referans tekerlek hızları, telsiz iletişimle robota gönderilir. Engellerden sakınarak hedefe ulaşmak için robotun bu referansları takip etmesi gerekr.

Uygulanan algoritmanın kuvvetli ve zayıf yönleri elde edilen deneysel sonuçlarla ortaya konulmuştur. Veriler resim, grafik ve video film şeklinde sunulmuştur. Zayıflıkların düzeltilmesi için bazı emprik yöntemler de önerilmiştir.

To my beloved family and friends…

To Mr. Sakıp Sabancı (Sabancı Dedemiz)…

To the world of science…

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my thesis supervisor Ahmet ONAT for his accompany, patience, and helps in the course of my thesis studies, and finally for his kindness to proofread the whole thesis.

I would like to thank Prof. SABANOVIC for his guidance, which was extremely helpful, and also all other Mechatronics professors, and graduate students.

I am of course indebted to my family, without them I would not be in such a position.

My special thanks are for SABANCI family, especially Mr. SAKIP SABANCI, for establishing and maintaining such a perfect university.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
Atlantis	*A T*hree-*L*ayer *A*rchitecture for *N*avigating *T*hrough *I*ntricate *S*ituations
AuRA	Autonomous Robot Architecture
AVENUE	Autonomous Vehicle for Exploration and Navigation in Urban Environment
bps	bits per second
CMU	Carnegie Mellon University
CRC	Cyclic Redundancy Check
d	distance from robot to goal/obstacle
dll	Dynamic link library
DTG	Drive Toward Goal
e_r, e_θ	Errors in the virtual forces acting on the robot
F	Force
FM	Frequency modulation
fps	frames per second
F_r	Force component in the direction of motion of the robot
F_θ	Force component perpendicular to the direction of motion of the robot
GPS	Global Positioning System
GUI	Graphical User Interface
HSI	Hue Saturation Intensity color model
IDE	Integrated Development Environment
IRDA	Infra Red Data Association
K_{pr}, $K_{p\theta}$	Proportional gains
L	Interwheel distance of robot

LCD	Liquid Crystal Display
MIT	Massachusetts Institute of Technology
OA	Obstacle Avoidance
OA, GTr	Obstacle Avoidance, Drive Toward Goal layer weights
OpenCV	Intel® Open Computer Vision library
PC	Personal Computer
PI	Proportional and Integral controller
PSD	Position Sensitive Device
PTU	Pan Tilt Unit
RAM	Random Access Memory
RGB	Red Green Blue color model
RoBIOS	Robot Basic Input Output System, operating system of the robot
ROM	Read Only memory
r^w-θ^w	robot's local coordinate frame
SUV	Sport Utility Vehicle
UART	Universal Asynchronous Receiver Transmitter
UHF	Ultra High Frequency
u_r, u_θ	Control inputs
V	Velocity of the robot
VFF	Virtual Force Field method
VFH	Virtual Field Histogram method
V_L,V_R	Left and right wheel velocities of the robot
w_s	Size of active window in VFF method
θ	Angle (goal, obstacle)
θ_{turn}	Turning angle of robot used to calculate the weights OA, and GTr
ω	Rotational velocity of the robot
Φ	Orientation angle of robot with respect to the global coordinate system

VISUAL SERVOING OF MOBILE ROBOTS

USING POTENTIAL FIELDS

by

MUHAMMET BAŞTAN

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

SABANCI UNIVERSITY

Spring 2004

VISUAL SERVOING OF MOBILE ROBOTS

USING POTENTIAL FIELDS

APPROVED BY:

Assistant Prof. Dr. AHMET ONAT …….

(Thesis Supervisor)

Assistant Prof. Dr. AYHAN BOZKURT …….

Prof. Dr. ASIF SABANOVIC ……..

DATE OF APPROVAL: ……

ABSTRACT

The popularity of autonomous mobile robots have been rapidly increasing due to their new emerging application areas, from room cleaning, tourist guidance to space explorations. However, the development of a satisfactory control algorithm that will enable the autonomous mobile robots to navigate safely especially in dynamic environments is still an open research problem.

In this work, a newly proposed potential field based control method is implemented, analyzed, and improvements are suggested based on experimental results obtained from a real robot.

The experimental system, planned to be the groundwork for robot soccer team, consists of an overhead global vision camera, personal computer, wireless module, and a non-holonomic mobile robot, which is supposed to navigate safely among obstacles and reach its goal point. Images of the robot's operating area are acquired by the camera, color processed by the PC in real-time to retrieve the required position and orientation information of the robot, goal, and obstacles. Then, the control algorithm is applied, and resulting reference wheel velocities are sent to the robot via wireless link. Finally, robot has to drive with these reference velocities for safe navigation and goal tracking.

Experimental results in the form of motion history images, graphs and movies are presented to demonstrate the successful as well as problematic aspects of the implemented algorithm and the setup.

ÖZET

Otonom gezer robotların, oda temizliği ve turist rehberliğinden uzay araştırmalarına kadar değişik uygulama alanlarında kullanılmaya başlanması popülerliklerini artırmıştır. Ancak, bu robotların özellikle dinamik ortamlarda güvenle hareket etmesini sağlayabilecek tatmin edici düzeyde kontrol algoritmalarının geliştirilmesi hala araştırma konusudur.

Bu çalışmada, önceden önerilmiş "potansiyel alan" tabanlı bir kontrol metodu analiz edilmiş ve gerçek bir robot ile yapılan deneysel verilere dayanarak iyileştirmeler önerilmiştir.

Bir robot futbol takımı için altyapı olarak planlanan deneysel sistem bütün ortamı görmeyi sağlayacak bir kamera, bilgisayar, telsiz iletişim modülü ve holonom olmayan iki tekerlekli, ve engeller arasında güvenle dolaşıp hedefine ulaşmyı amaçlayan bir robottan oluşmaktadır. Robotun çalışma alanının görüntüleri kamera tarafından alınıp, robotun ueri, hedef ve engellerin uzaklık ve yönünü bulmak için bilgisayar tarafından gerçek zamanlı renkli görüntü işlemeye tabi tutulur. Daha sonra, elde edilen hedef ve engel bilgilerine kontrol algoritması uygulanıp bulunan referans tekerlek hızları, telsiz iletişimle robota gönderilir. Engellerden sakınarak hedefe ulaşmak için robotun bu referansları takip etmesi gerekr.

Uygulanan algoritmanın kuvvetli ve zayıf yönleri elde edilen deneysel sonuçlarla ortaya konulmuştur. Veriler resim, grafik ve video film şeklinde sunulmuştur. Zayıflıkların düzeltilmesi için bazı emprik yöntemler de önerilmiştir.

To my beloved family and friends…

To Mr. Sakıp Sabancı (Sabancı Dedemiz)…

To the world of science…

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my thesis supervisor Ahmet ONAT for his accompany, patience, and helps in the course of my thesis studies, and finally for his kindness to proofread the whole thesis.

I would like to thank Prof. SABANOVIC for his guidance, which was extremely helpful, and also all other Mechatronics professors, and graduate students.

I am of course indebted to my family, without them I would not be in such a position.

My special thanks are for SABANCI family, especially Mr. SAKIP SABANCI, for establishing and maintaining such a perfect university.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
Atlantis	*A T*hree-*L*ayer *A*rchitecture for *N*avigating *T*hrough *I*ntricate *S*ituations
AuRA	Autonomous Robot Architecture
AVENUE	Autonomous Vehicle for Exploration and Navigation in Urban Environment
bps	bits per second
CMU	Carnegie Mellon University
CRC	Cyclic Redundancy Check
d	distance from robot to goal/obstacle
dll	Dynamic link library
DTG	Drive Toward Goal
e_r, e_θ	Errors in the virtual forces acting on the robot
F	Force
FM	Frequency modulation
fps	frames per second
F_r	Force component in the direction of motion of the robot
F_θ	Force component perpendicular to the direction of motion of the robot
GPS	Global Positioning System
GUI	Graphical User Interface
HSI	Hue Saturation Intensity color model
IDE	Integrated Development Environment
IRDA	Infra Red Data Association
K_{pr}, $K_{p\theta}$	Proportional gains
L	Interwheel distance of robot

LCD	Liquid Crystal Display
MIT	Massachusetts Institute of Technology
OA	Obstacle Avoidance
OA, GTr	Obstacle Avoidance, Drive Toward Goal layer weights
OpenCV	Intel® Open Computer Vision library
PC	Personal Computer
PI	Proportional and Integral controller
PSD	Position Sensitive Device
PTU	Pan Tilt Unit
RAM	Random Access Memory
RGB	Red Green Blue color model
RoBIOS	Robot Basic Input Output System, operating system of the robot
ROM	Read Only memory
r^w-θ^w	robot's local coordinate frame
SUV	Sport Utility Vehicle
UART	Universal Asynchronous Receiver Transmitter
UHF	Ultra High Frequency
u_r, u_θ	Control inputs
V	Velocity of the robot
VFF	Virtual Force Field method
VFH	Virtual Field Histogram method
V_L,V_R	Left and right wheel velocities of the robot
w_s	Size of active window in VFF method
θ	Angle (goal, obstacle)
θ_{turn}	Turning angle of robot used to calculate the weights OA, and GTr
ω	Rotational velocity of the robot
Φ	Orientation angle of robot with respect to the global coordinate system

CHAPTER 1

INTRODUCTION

1.1 Motivation

With the exponential increase in the scientific knowledge and thus technology in recent years and in particular with the advent of computers, stories in science fiction books started to become true, and robots started to appear in daily life.

Robots can help human beings; they can clean the room, clean sewerage pipes, disarm land mines, dig for oil, make explorations on the Moon, help surgeons in operations, work in factory environments, run production lines doing the jobs hard, dangerous or boring for humans. They can even be companions, interacting with humans, entertaining them; guiding tourists in museums. Therefore, there has been an ever-growing interest in autonomous mobile robots for their applicability in everyday life as well as in specific missions. However, autonomous mobile robotics is still in its infancy, and extensive research is going on for improvements to make their widespread use possible.

One of the fundamental problems of autonomous mobile robots is safe navigation in especially dynamic environments. It is still an open research problem to develop a satisfactory control algorithm for autonomous mobile robots that will enable them to navigate safely in dynamic environments, where obstacles are moving around or the goal of the robot is changing its position.

For navigation, robots utilize several sensors to monitor the external world. Cameras have started to draw a growing attention from the mobile robotics research community, due to its ability to supply the robot with rich information about its environment, which may not be provided even by combination of several sensors, or it may be too costly. Being analogous to the eye, which is probably the most important

sensor for living beings, the popularity of using cameras as sensors will increase especially with improved techniques to retrieve information from the images, and with increasing computational power.

In the light of what have been mentioned above, this work aimed to contribute to the efforts to develop a satisfactory control algorithm for autonomous mobile robots. For this, an experimental setup, using a camera as the primary sensor, has been prepared to implement, analyze, and improve a potential field mobile control algorithm.

1.2 Literature Survey: Mobile Robot Servoing, Methods, and Applications

One of the fundamental challenges in robotics is motion or path planning, which means to generate a continuous path for a given robot between an initial and a goal configuration of the robot. Along this path, the robot must not intersect given forbidden regions, which are usually obstacles. Different versions of the motion planning problem correspond to the cases where the obstacles are stationary or moving and where the system of objects consists of a point, polygonal object, a single polyhedral object or a set of polygonal or polyhedral objects. Practical instances of the motion-planning problem occur frequently in robot-based assembly operations where a continuous trajectory must be planned for a robot such that collision with neighboring objects and other robots in the same workspace is avoided and the structural limits of the joint actuators are not violated. Similar problems are also encountered in the automatic navigation of vehicles in constrained environments, missile guidance, VLSI circuit layout and aircraft routing. Due to its widespread application, there is a lot of interest in this problem in the academic community and various approaches to its solution have been proposed [28].

1.2.1 Deliberative Control

A robot employing deliberative control requires relatively complete knowledge about its world and uses this knowledge to predict the outcome of its actions, an ability that enables it to optimize its performance relative to its world model. The knowledge must be consistent, reliable, and certain. In a dynamic world, where the objects are

moving arbitrarily, it is dangerous to rely on past information that may no longer be valid. Representational world models are therefore generally constructed from both prior knowledge about environment and incoming sensor data in support of deliberation [1]. It has the following characteristics:

- Hierarchical in structure with a clearly identifiable subdivision of functionality, therefore, it is also named *hierarchical control.*
- Communication and control occur in a predictable and predetermined manner, flowing up and down the hierarchy, with little or no lateral movement.
- Higher levels in the hierarchy provide sub goals for the lower levels.
- It relies heavily on symbolic representation of world models.

Using deliberative control strategy, Albus (1991) developed a robotic system having a layered structure, each layer consisting of 4 components: sensory processing, world modeling, task decomposition, and value judgment. All layers are joined by a global memory through which representational knowledge is shared. Perception is the correspondence between the internal world model and external world. Thus, perception is not directly tied to action, which is the property of deliberative reasoning [1].

Parallel with this, researchers at Drexel University have focused on the theory of intelligent hierarchical control, consisting of 6 levels. The set of nested hierarchical controllers consists of a high-level planner, navigator, pilot, path monitor, controller, and low-level control system [1].

In yet another representative of the intelligent controls community, research at the Rensselaer Polytechnic Institute restricted the hierarchy to three primary levels: organization level (conducts high level planning and reasoning), coordination level (provides integration across various hardware subsystems), and execution level (supports basic control and hardware). This approach implements the principle of increasing precision with decreasing intelligence as one descends down the hierarchy [1].

Hierarchical control is well suited to structured and highly predictable environments (e.g., manufacturing systems). Reactive systems were developed in response to several apparent drawbacks associated with the hierarchical design paradigm including a perceived lack of responsiveness in unstructured and uncertain

environments due both to requirements of world modeling and the limited communication pathways [1].

An example is the robot *Xavier* of Carnegie Mellon University (figure 1.1), with a deliberative architecture capable of integrating a priori map knowledge into a behavior-based control. A topological map of the robot environment is generated from floor plans, representing the world model; Markov models (specialized probabilistic models) encode the actions that a robot can take at different locations within the model. The planner uses an A* search algorithm to specify the actions to be taken at each point within the topological model. The Markov model based planner issues directives to the robot for turning right, left, moving forward or stopping. It is reported to have successfully completed 88 percent of its missions using this strategy with more than one-kilometer distance traveled [29].

Figure 1.1 Xavier of CMU [29].

1.2.2 Reactive Control

Reactive control is a technique for tightly coupling perception and action, typically in the context of motor behaviors, to produce timely robotic response in dynamic and unstructured environments. Use of abstract representational knowledge is avoided in the generation of a response, and the robot reacts directly to the world as it is sensed. This is of particular value in dynamic and hazardous environments, where unpredictability and potential hostility are inherent. Constructing abstract world models

is a time consuming and error-prone process and thus reduces the potential correctness of a robot's action in all but the most predictable worlds [1].

Reactive control is inherently modular from a software design perspective, which enables the designer to expand the robot by adding new behaviors without redesigning or discarding the old. Mapping between the stimulus domain and response range is achieved by discrete encoding in the form of if-then-else; or continuous functional encoding allowing a robot to have infinite space of potential reactions to its world. Instead of having a set of enumerated responses that discretizes the way in which the robot can move (e.g., {forward, backward, right, left, etc.}), a mathematical function transforms the sensory input into a behavioral action [1]. One of the most common methods for implementing a continuous response is based on a technique referred to as the potential fields, which will be discussed later.

The originator of the method is Rodney Brooks, who developed subsumption architecture in mid 80s at MIT, a purely reactive behavior based approach. He argued that the sense-plan-act paradigm used in some of the first autonomous robots was in fact detrimental to the construction of real working robots. He further argued that building world models and reasoning using explicit symbolic representational knowledge at best was an impediment to timely robotic response and at worst led robotics researchers in the wrong direction [1].

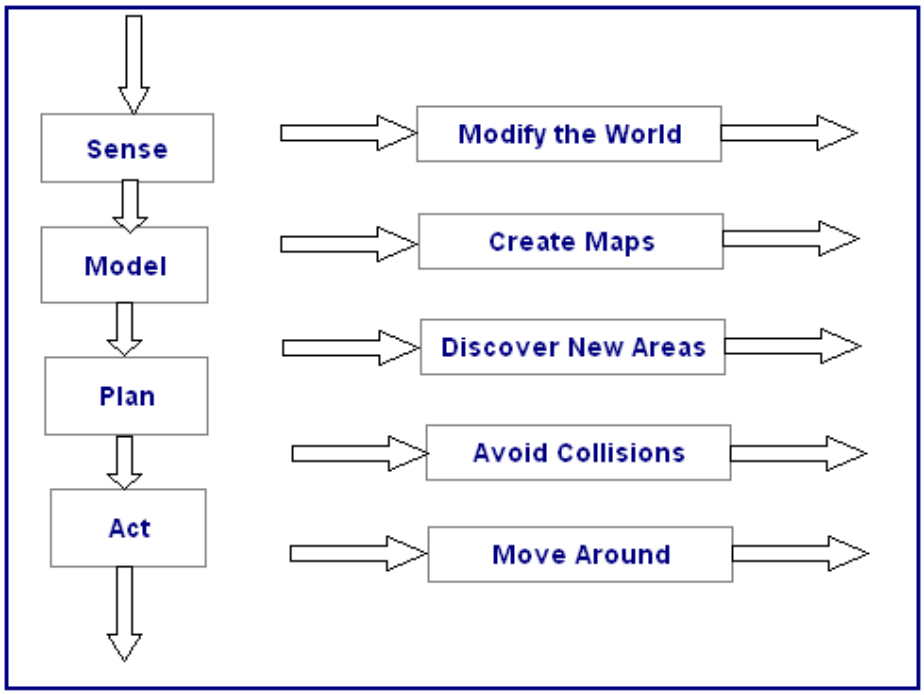

Figure 1.2 Sense-plan-act (left), and subsumption (right) model [1].

1.2.3 Hybrid Systems

Reactive behavior based robotic control can effectively produce robust performance in complex and dynamic environments. In some ways, however, the strong assumptions that reactive systems make can serve as a disadvantage. These assumptions are:

- The environment lacks temporal consistency and stability.
- The robot's immediate sensing is adequate for the task at hand.
- It is difficult to localize a robot relative to a world model.
- Symbolic representational world knowledge is of little or no value.

These assumptions may not be valid all the time. Therefore, purely reactive robotic systems are not appropriate for all robotic applications. Hybrid deliberative/reactive robotic architectures have recently emerged combining aspects of traditional AI symbolic methods and their use of abstract representational knowledge, but maintaining the goal of providing the responsiveness, robustness, and flexibility of reactive systems.

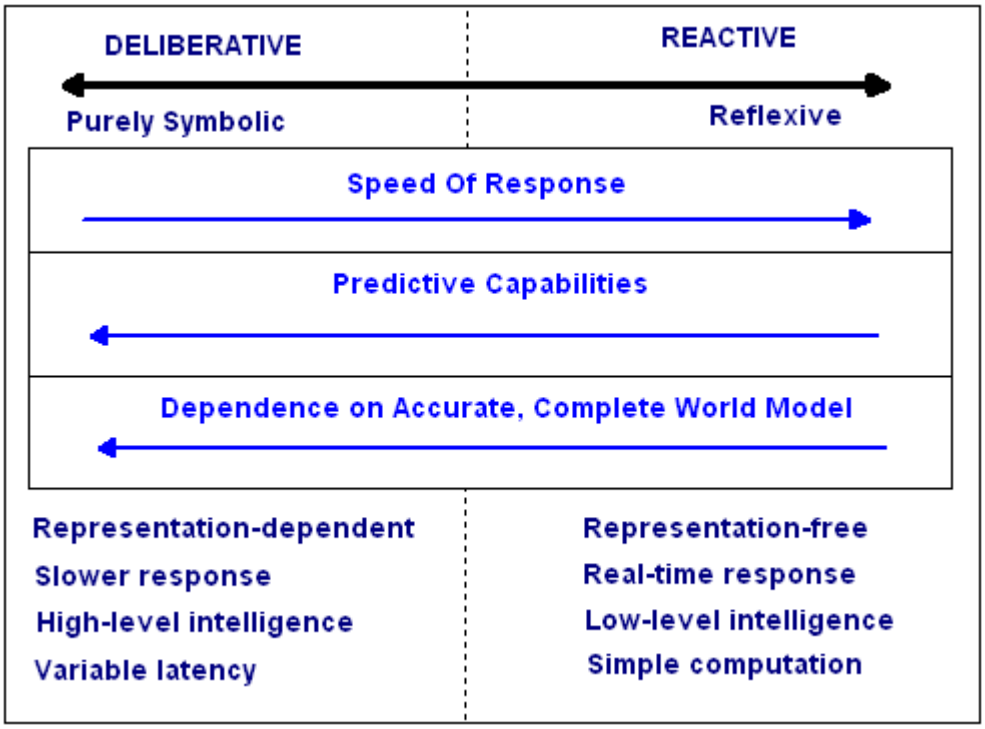

Figure 1.3 Robot Control Spectrum [1].

Two example architectures based on hybrid systems are AuRA (Autonomous Robot Architecture) built by Arkin (1986, 1987b), and Atlantis built at the Jet Propulsion Laboratory (JPL) by Gat in 1991.

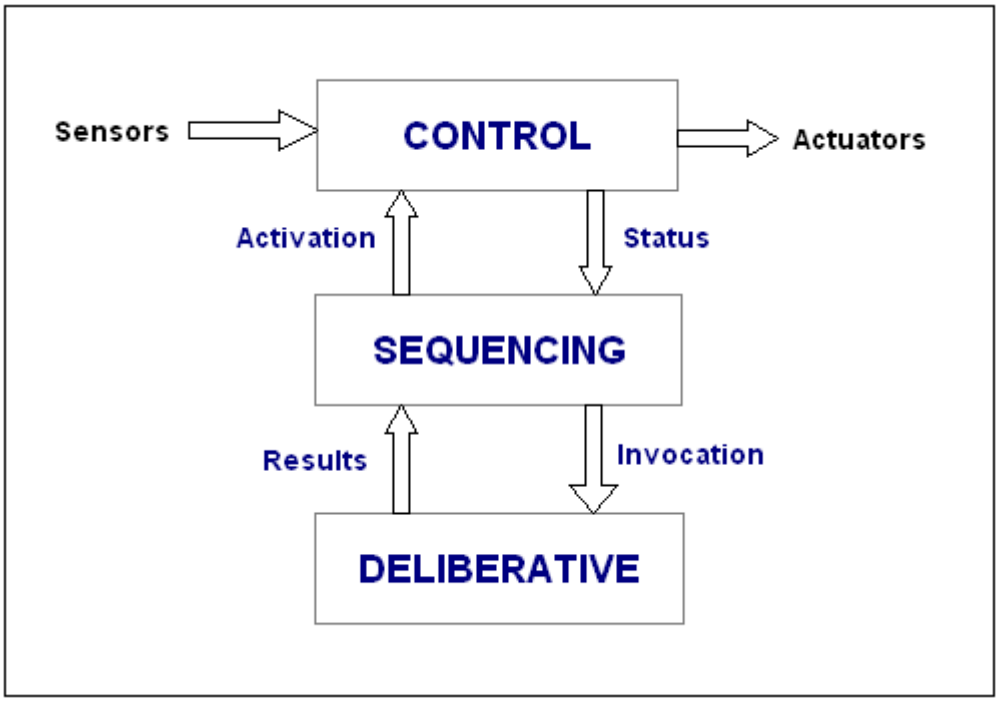

Figure 1.4 Atlantis architecture [1].

Atlantis, (or *A Three-Layer Architecture for Navigating Through Intricate Situations*), like the subsumption architecture, is built in layers. In Atlantis, however, all instantiations of the architecture have the same three layers, each of which always performs the same duty. The Control Layer directly reads sensors and sends reactive commands to the effectors based on the readings. The stimulus-response mapping is given to it by the *sequencing layer* which has a higher level view of robotic goals than the *control layer*. It tells the *control layer* below it when to start and stop actions. The deliberative layer responds to requests from the sequencing layer to perform deliberative computations [30].

1.2.4 Robot Navigation, Path Planning

Navigation is not just driving from one location to a specified location; it depends on the particular task to be carried out. For example, are the destination points known or

do they have to be searched, are the dimensions of the driving environment known (will the robot drive on a flat plane or fly in 3D space), are all objects in the environment known, are they moving or stationary, and so on [2]?

There are a number of well-known navigation algorithms, some of which will be discussed here briefly. Some of these algorithms are of a more theoretical nature and do not closely match the real problems encountered in practical navigation scenarios. For example, some of the shortest path algorithms require a set of node positions and full information about their distances, but in many practical applications there are no natural nodes or their location or existence is unknown, like partially or completely unexplored environments [2].

1.2.4.1 Dijkstra's Algorithm

Dijkstra's Algorithm [Dijkstra 1959] is for computing all shortest paths from any node to any other node in a fully connected graph. Relative distance information between all nodes is required; distances must not be negative. Obviously, it requires complete knowledge of the robot's environment and is not suitable for dynamic cases, in which case robot has to recalculate the shortest paths at each sampling interval, which, due to processing power and speed limitations, makes the algorithm inappropriate for real time applications [2].

1.2.4.2 A* Algorithm

A* [Hart, Nilsson, Raphael 1968], pronounced "A-Star", heuristic algorithm is developed to compute the shortest path from one given start node to one given goal node. Like the previous algorithm, 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 [2].

1.2.4.3 Certainty Grids Method

Certainty grids method [H. Movarec, 1987] produces an obstacle map of the environment using probabilistic representation of obstacle locations in order to overcome the inaccuracies coming from sensory data. The robot creates a two dimensional array representing its environment; each element of the array is a cell containing the certainty values – measure of confidence that an obstacle exists within the cell area or not. The content of each cell is updated continuously according to the sensor readings with a probabilistic method considering the characteristics of the sensors used. Obviously, this method assumes a static environment without any moving obstacles; otherwise, due to the additive nature of the method all grids might be filled due to moving obstacles and there will be no free space left in the resulting map for the robot to move [2].

1.2.4.4 Potential Fields Method

Probably none of the robot navigation methods has attracted so great an interest from researchers as potential fields method [Khatib 1985, Krough 1984], such that so many variations of the method have been developed and used.

The method was developed as a basis for generating smooth trajectories for both mobile and manipulator robotic systems. Separate attractive and repulsive potential fields are constructed to represent the relationship between the robot and each of the objects within the object's sensory range. These fields are then combined to yield a single global field. A smooth trajectory is computed based upon the gradient within the globally computed potential field.

Drawing from the field theory concept in physics, this method models obstacles as emitting a repellant force and the goal point as emitting an attractive force on the robot. Navigation is performed by moving the robot so as to minimize the potential energy.

This approach assumes knowledge of the type of obstacles in the environment and in the planning phase polygons or spheres approximate these known obstacles. The environment in the original formulation of this idea was assumed to be static; however there have been adaptations to use this approach for dynamic environments. Potentials are associated with the objects in the environment as and when it is encountered.

This method is designed to work with only a single goal. If more than one goal is required than there exists an implicit stack of goals that is popped once the current target is reached. There is no change in behavior if a sequence of waypoints is specified.

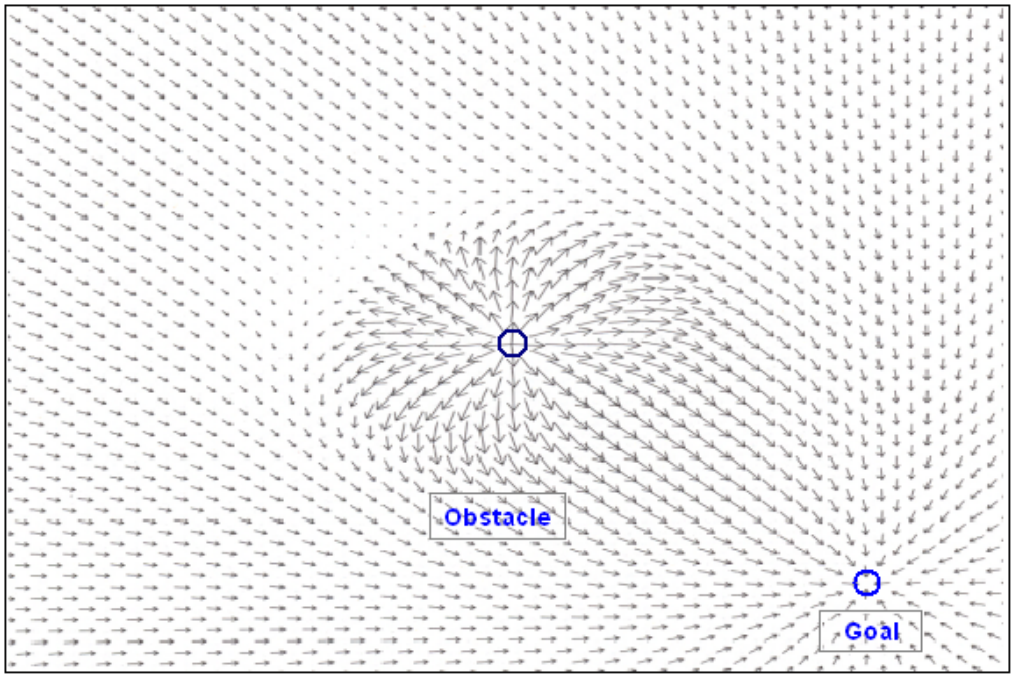

Figure 1.5 Potential field generated by an obstacle and a goal [1].

Recent potential field based approaches incorporate dynamic sensing feedback into robot control and hence try to overcome the limitations of being computationally heavy and unable to react to unexpected obstacles in the environment that optimization based approaches suffer from. The potential is calculated at each point along the trajectory and the robot then moves down the gradient of the potential field till it reaches a minimum, local or global [28].

Being one of the most successful methods developed, potential field is not without its problems. Koren and Borenstein analyzed the potential field method; in their work [4] they explained its inherent limitations with experimental results obtained using a three-wheel drive robot (CARMEL). The problems can be summarized as follows:

- Trap situations due to local minima: Perhaps the best-known and most often-cited problem of potential field method is *local minima* or *trap situations*. A trap situation may occur when the robot runs into a dead end (e.g., inside a U-shaped obstacle). It can be resolved by heuristic or global recovery.

- No passage between closely spaced obstacles: Figure 1.6 shows a mobile robot at an attempt to pass through two closely spaced obstacles. The repulsive forces due to obstacle 1 (F_{r1}) and obstacle 2 (F_{r2}) add up to a force (F_r) pointing away from the passage. Depending on the relative magnitudes of attractive force (F_t) and total repulsive force, the robot will either approach the passage further or it will turn away.

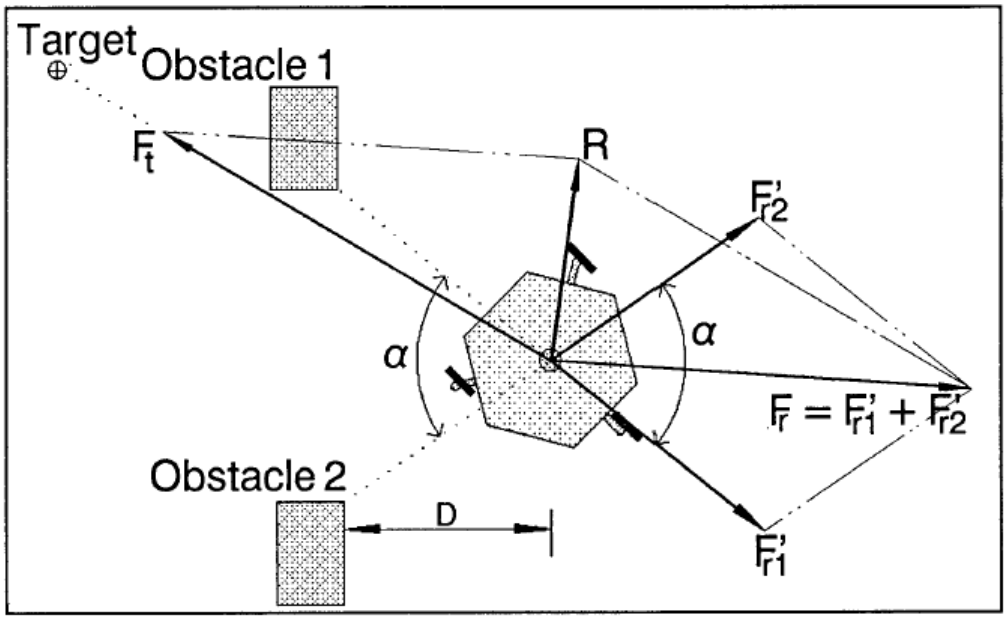

Figure 1.6 Under potential field method, the robot does not pass through closely spaced obstacles [4]

- Oscillatory motion in narrow passages, corridors: When the robot travels in narrow corridors (figure 1.7), it experiences repulsive forces from opposite sides. A sudden change in the width of a narrow corridor excites the robot into unstable oscillations and eventually a collision.

Another problem of potential fields that is usually overlooked is the fact that the robot cannot reach its goal if the goal is located in such position that the repulsive force is larger than the attractive force resulting in a motion away from the goal instead of reaching it [20].

With these problems in mind, Borenstein developed new algorithms (VFF, VFH, etc.) based on potential fields in an effort to eliminate, or at least reduce the problems, as many researchers have been doing [8, 11, 12, 13, 14, 15].

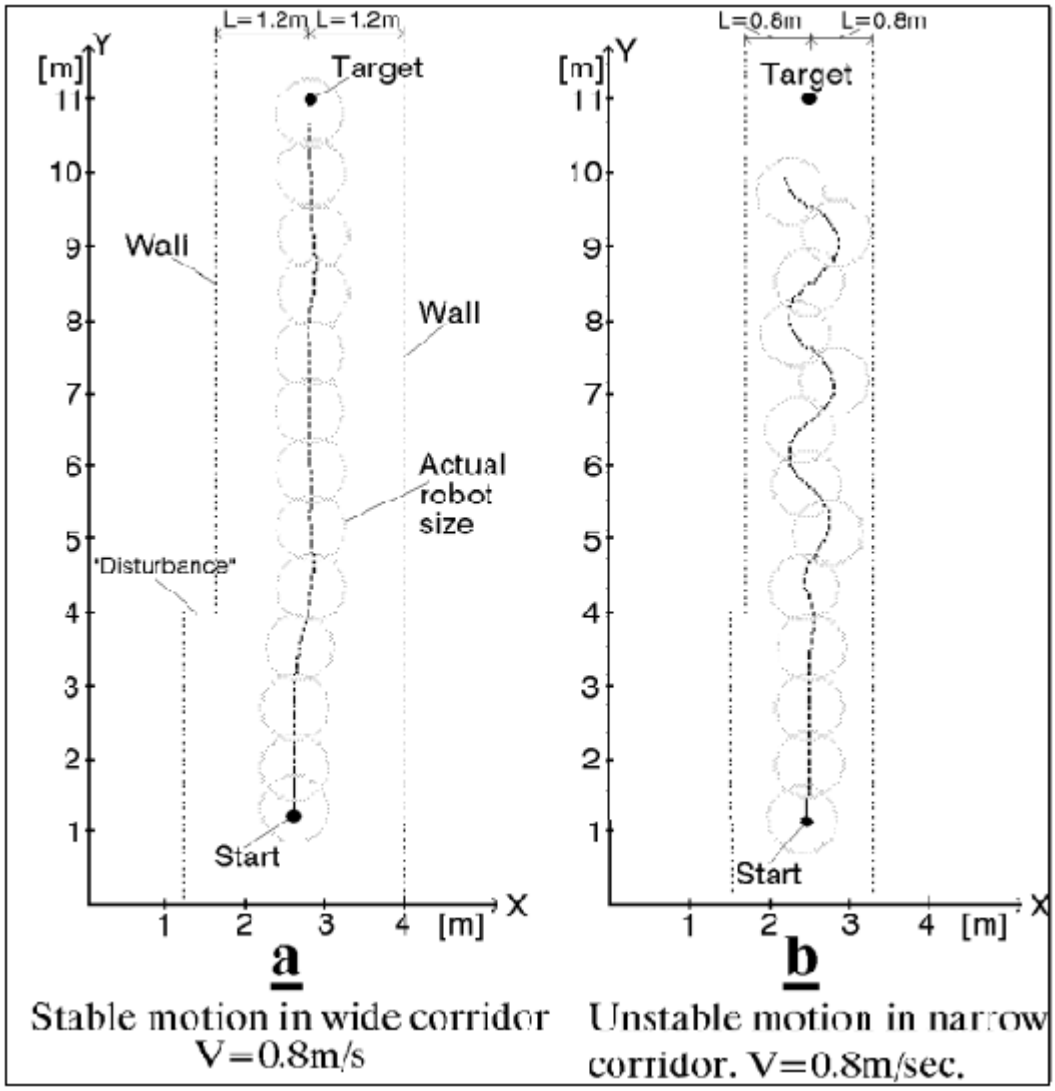

Figure 1.7 Stable and unstable motion in corridors [4].

1.2.4.5 Virtual Force Field (VFF) Method

Developed by Koren and Borenstein, virtual force field (VFF) method can be considered to be a combination of certainty grids and potential fields method. A virtual window moves with the robot and overlays a region of a histogram grid. The window is called the *active window* and covers an area of w_s by w_s; the cells within the active window are called active cells. Each active cell applies a virtual repulsive force on the robot similar to the potential field method; the magnitude of the force being proportional to the certainty value of the cell, and inversely proportional to the square of the distance between the cell and the robot. All virtual repulsive forces are added to a

constant attractive force from the target point to obtain a resultant force vector; and the robot's steering is aligned with this resultant force to avoid the obstacles.

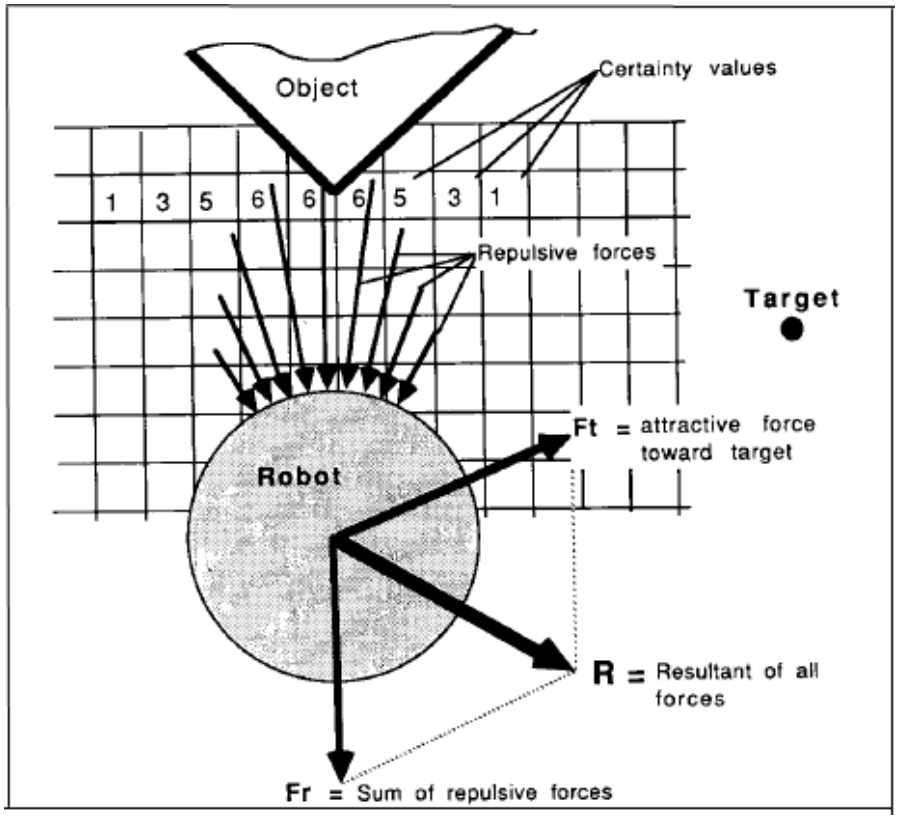

Figure 1.8 Virtual Force Field Method. [8,11].

Since it is based on the potential field method, it also has similar problems; but it helps reduce the sensor inaccuracies due to its probabilistic nature. Moreover, sensor data influences the steering immediately whenever a new obstacle is detected, and a fast reflexive behavior is obtained. Main drawback of the method is caused by the size of the cells; increasing the cell size causes drastic changes in force magnitudes resulting in a non-smooth trajectory, decreasing the cell size increases the required computational power [8,11,12,13].

1.2.4.6 Vector Field Histogram (VFH) Method

To remedy the shortcomings of VFF method, Koren and Borenstein came up with the vector field histogram method built on VFF. This method uses a two-stage data

reduction rather than one, which was the case in VFF. It has three levels of data representation:

- The highest level holds the detailed description of the robot's environment. The two-dimensional Cartesian histogram grid is continuously updated in real-time, similar to VFF method.
- At the intermediate level, a one-dimensional polar histogram is constructed around the robot's momentary location.
- The lowest level is the output of the VFH method, giving the reference values for the drive and steering controllers of the robot. [12,13].

This method solved the data reduction problem of VFF, but it still has the limitations of potential field method.

1.2.4.7 Wandering Standpoint Algorithm

Wandering Standpoint Algorithm [Puttkamer 2000] is a local path planner algorithm, requiring local distance measurements. The robot tries to reach the goal in direct line, when it encounters an obstacle; it measures the avoidance angle for turning left and right, and turns to the smaller angle. Then it continues with boundary following around the object, until goal direction is clear again [2].

The algorithm can lead to an endless loop for extreme obstacle placements. In this case the robots continues driving but never reaches the goal [2].

1.2.4.8 DistBug Algorithm

This algorithm [Kamon, Rivlin 1997] combines the local path planning with global information and guarantees convergence. Therefore, it requires local sensor data and global information. It is similar to the wandering standpoint algorithm, but boundary following stops only if goal is directly reachable or if future hit point with next obstacle would be closer to goal. This global information together with detection of unreachable goal if robot has turned 360 degrees guarantees convergence [2].

Although it has very nice theoretical properties, it is not always usable in practice, since it requires global information in the form of path intersection points of future possible collision points with objects [2].

1.2.4.9 Overview of Navigation Methods

Some of the well-known methods have been discussed until now. There are other methods developed to improve the existing methods. For example, new variations of VFF, VFH were developed by their original authors. Since this is not an exhaustive study of all methods, there are some other methods not discussed here, such as edge detection.

No perfect method has been developed until now, therefore, research is going on, and new methods or refinement of existing methods are being proposed. Some of the discussed methods require knowledge of the robot's environment being on the side of deliberative control, while some of them are trying to avoid the requirement of prior knowledge as much as possible being closer to reactive control. It is of course advantageous not to require knowledge of the environment as reactive control is trying to do, but this will not be 100 percent satisfied in all of the methods, because at least the location the target point relative to robot is required in all algorithms if the robot is trying to go to a target location rather than just wandering around. The requirement of a priori knowledge should be kept at a minimum due to the memory and processing power requirements of today's robots. Even if there is no memory or processing power constraints, as will be the case in the near future, complete knowledge of the robot's environment is impossible to achieve, especially for outdoor applications. Besides, the world is dynamic, and changing continuously that invalidates most of the prior knowledge. As a result, the reactive control methods will keep their popularity, while researchers are trying to solve their shortcomings. On the other hand, hybrid systems, combining deliberative and reactive methods started to gain increased attention from the scientific world, in an effort to get the best of two worlds. However, the interface between deliberation and reactivity is not completely understood and serves as the focus of research in this area.

In summary, both deliberative and reactive systems have limitations when each is considered in isolation.

1.2.5 Brief Overview of Robot Sensors

For a robot to function properly, it is crucial for it to interpret the information about its immediate surroundings. Because it is not possible to react to external events without perceiving them, and most of the time only partial knowledge of the environment is available or any information available may be inaccurate or obsolete. Perception in robotics is tightly coupled to actions. Robotic perception can be categorized as follows [1]:

- *Action-oriented perception:* perception is tuned according to the action requirements of the robot.
- *Expectation-based perception:* Prior knowledge of the environment can constrain the interpretation of what is perceived.
- *Focus-of-attention methods:* Prior knowledge can constrain where things may appear.
- *Active perception:* perceptual requirements determine the robot's actions [1].

Rapidly advancing sensor technology enabled robots to utilize low-cost high quality sensors. However, improvements are still essential to get better or even satisfactory performance in some cases.

1.2.5.1 Shaft Encoders

Shaft encoders provide information regarding how far a robot has traveled based on the rotation of its motors, or wheels. Using the output of the encoders, the present location of the robot can be estimated, a method called 'dead reckoning', which can be extremely misleading due to slippage on the wheels or locomotion mechanism of the mobile robots, finite resolution and sampling of encoders. Therefore, they have to be supplemented with environmental sensing to produce reliable results when determining the robot's actual position. They are primarily used as the fundamental feedback sensor for motor control of the wheels, although there have been attempts to reduce the dead reckoning errors and thus utilized for position estimation.

1.2.5.2 Position Sensitive Devices (PSD), Orientation Measurement

Among the most important sensors in mobile robotics are the ones for distance measurement, especially to measure the distance to obstacles around for navigation. Sonar, laser, and infrared sensors are heavily used for this purpose. They are active sensors in that they emit a signal and calculate the distance to the nearby objects by processing the reflected signal.

Compass is used to determine the robot's absolute orientation. A further step could be interfacing to a receiver module for the satellite-based *global positioning system* (GPS), which is very useful to get global knowledge about the world, but it is complex and can only work outdoors in unobstructed areas [2].

Gyroscope, accelerometer, inclinometer are orientation sensors to determine a robot's orientation in 3D space, and are useful for tracked, walking, balancing, and flying robots.

1.2.5.3 Camera (Computer Vision)

Cameras are probably the most complex sensors utilized in robotics. They have not been used extensively in robotics until recently due to high computational and memory requirements, as well as high cost. Extensive research are going on to incorporate vision into robotics, since robots of future cannot be thought of a sense of visual perception of environment like human beings or animals.

Currently robots are using either global vision or local vision. In global vision, the camera is external to the robot, placed in such a way that it can see a region large enough to enclose both the robot and its environment. In local vision, robot has its own camera doing all the vision processing onboard. The latter approach is more difficult to implement especially for small robots due to space, memory, and processing power constraints. However, it is natural to have visual perception onboard, since no animal or human has global vision; rather every living being has its own sensors. Therefore, the trend is to have local vision on the robots, which will get easier with ever-increasing memory and processing capacity being embedded in them. On the other hand, global

vision is much easier to realize, and also has some implementations, such as in factory environments, and the popular competition of robot soccer.

It should be noted that vision is still in its infancy and has to be improved tremendously to be employed in robotics more effectively. Thinking of ourselves with great visual ability to see, recognize objects, and differentiate them from one another in a matter of milliseconds with great accuracy, computer vision turns out to be quite far behind to provide high quality information for the robots at the required speed, especially for real-time applications.

1.2.5.4 Combining Sensor Outputs

There are different approaches as to how to combine the outputs of different sensors for a robot. Inspired of the living things, robots have different kinds of sensors, e.g., camera, laser scanner, etc to get more information about their environments. However, combining the outputs of sensor, termed *sensor fusion*, is not a trivial job. The incoming information from different sensors might be complementary (in support of each other), or competitive (contradicting with each other). Further, they might be arriving asynchronously, since some sensors take longer to process than others. Therefore, it is imperative to design the sensory system carefully considering the types of sensors employed, and this is still an open research area how to effectively combine sensor outputs.

1.2.6 Representative Examples

There are numerous examples of autonomous mobile robot applications, few of which will de summarized in the next sections.

1.2.6.1 Robot Soccer

One of the most interesting and exciting examples of robotics is probably soccer playing robots. There are currently two well-known initiatives to organize robot soccer

competitions, RoboCup and FIRA. RoboCup is an international research and education initiative. Its goal is to foster artificial intelligence and robotics research by providing a standard problem where a wide range of technologies can be examined and integrated [31].

The concept of soccer-playing robots was first introduced in 1993. Following a two-year feasibility study, in August 1995, an announcement was made on the introduction of the first international conferences and football games. In July 1997, the first official conference and games were held in Nagoya, Japan. From then on, the competitions are held in a different country each year with more and more participants from around the world with more and more advanced robot teams [31].

Figure 1.9 Robots playing soccer (RoboCup small-size league) [32].

The main focus of the RoboCup activities is competitive football. The games are important opportunities for researchers to exchange technical information. They also serve as a great opportunity to educate and entertain the public. There are different leagues:

- Simulation league: Independently moving software players (agents) play soccer on a virtual field inside a computer.

- Small-size robot league (f-180): Small robots of no more than 18 cm in diameter play soccer with an orange golf ball in teams of up to 5 robots on a field with the size of bigger than a ping-pong table.
- Middle-size robot league (f-2000): Middle-sized robots of no more than 50 cm diameter play soccer in teams of up to 4 robots with an orange soccer ball on a field with a size of 12x8 meters.
- Four-legged robot league: Teams of 4 four-legged entertainment robots (Sony's AIBO) play soccer on a 3 x 5 meter field.
- Humanoid league: This league was introduced in 2002 RoboCup. Biped autonomous humanoid robots compete in "walking" and "shooting". The robots play also in "penalty kick," and "1 vs. 1" matches. "Free style" competitions are to be expected as well [31].

The ultimate goal of RoboCup project is stated in official RoboCup website [31] as "By 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer".

Looking at the small-size league as an example will demonstrate how robot soccer works. A small-size robot soccer game takes place between two teams of five robots each. Each robot must conform to the dimensions as specified in the F180 rules: The robot must fit within a 180mm diameter circle and must be no higher than 15cm unless they use on-board vision. The robots play soccer on a green-carpeted field that is 2.8m long by 2.3m wide with an orange golf ball. Robots come in two flavors, those with local on-board vision sensors and those with global vision. Global vision robots, by far the most common variety, use an overhead camera and off-field PC to identify and track the robots as they move around the field. The overhead camera is attached to a camera bar located 3m above the playing surface. Robots, ball, goal, and field are identified by their differing colors. Local vision robots have their sensing on the robot itself. The vision information is either processed on-board the robot or is transmitted back to the off-field PC for processing. An off-field PC is used to communication referee commands and, in the case of overhead vision, position information to the robots. Typically the off-field PC also performs most, if not all, of the processing required for coordination and control of the robots. Communications is wireless and typically uses dedicated commercial FM transmitter/receiver units although at least one team has used IRDA successfully.

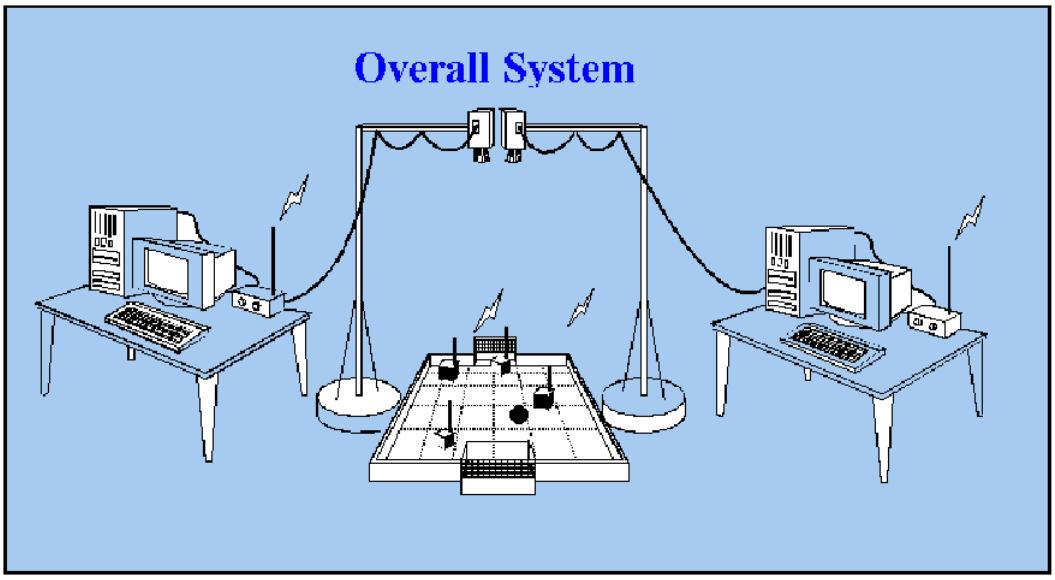

Figure 1.10 Robot Soccer setup [34].

Building a successful team requires clever design, implementation and integration of many hardware and software sub-components into a robustly functioning whole making small-size robot soccer a very interesting and challenging domain for research and education [31]. It is apparent how crucial it is for each robot in the team to navigate safely, tracking its goal (the ball) and avoiding stationary and dynamic obstacles (walls, other robots) around it.

1.2.6.2 Autonomous Road Following

Many research centers have been working on developing systems that can drive on the road autonomously at high speeds. VaMoRS developed at Military University of Munich, Terregetor, SCARF, RALPH, ALVINN, and NAVLAB series at CMU are some examples being able to cruise at maximum speeds 100 km/hour. Vision systems are used to detect the road and nonroad regions to drive safely without human intervention. RALPH was developed for *The No Hands Across America Navlab 5 USA* tour, being one of the most ambitious exhibitions of autonomous driving. It has a simple control process:

- An image is acquired, irrelevant portions are trimmed, remaining portion of the image is subsampled to yield 30x32 image array, which includes important road features.

- The road curvature and the vehicle's lateral offset to road's centerline are computed.
- A steering command is issued and the process is repeated.
- The vehicle was able to drive almost 3000 miles from Pittsburgh to Pennsylvania autonomously.

The CMU Navlab group builds computer-controlled vehicles for automated and assisted driving. Since 1984, they have built a series of robot cars, vans, SUVs (Sport Utility Vehicle), and buses. The latest in the Navlab family is the Navlab 11 (figure 1.11), a robot Jeep Wrangler equipped with a wide variety of sensors for short-range and mid-range obstacle detection.

Figure 1.11 NAVLAB 11 of CMU [29].

Another interesting example is the AVENUE (*Autonomous Vehicle for Exploration and Navigation in Urban Environments*) project of University of Colombia (figure 1.12), targeting the automation of urban site planning. The main goal is to build not only realistically looking but also geometrically accurate and photometrically correct models of complex outdoor urban environments. The models are needed in a variety of applications, such as city planning, urban design, historical preservation and

archeology, fire and police planning, military applications, virtual and augmented reality, geographic information systems and many others [33].

The task of the mobile robot is to go to desired locations and acquire requested 3-D scans and images of selected buildings. The locations are determined by the sensor planning (view planning) system and are used by the path planning system to generate reliable trajectories, which the robot then follows. When the robot arrives at the target location, it uses the sensors to acquire the scans and images and forwards them to the modeling system. The modeling system registers and incorporates the new data into the existing partial model of the site (which in the beginning could be empty). After that, the view planning system decides upon the next best data acquisition location and the above steps repeat. The process starts from a certain location and gradually expands the area it covers until a complete model of the site is obtained.

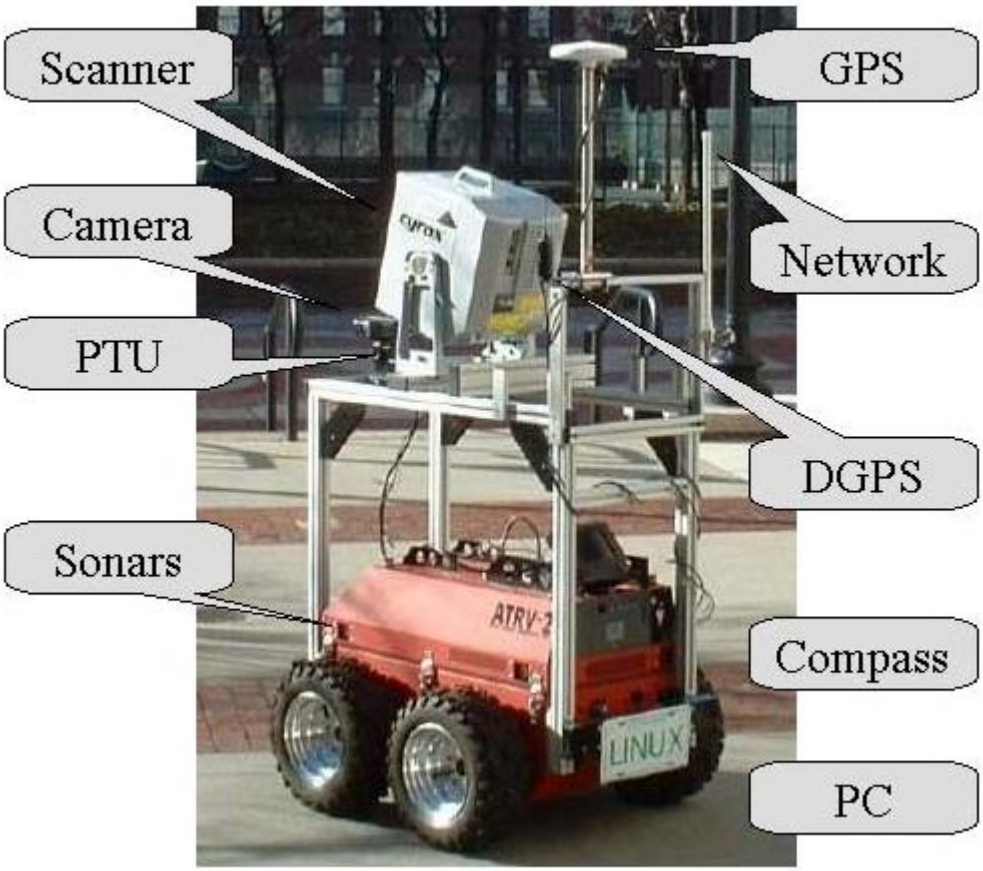

Figure 1.12 Mobile platform of AVENUE project [33].

1.2.7 Problem Definition

As has been discussed until now with real projects going on around the world, the fundamental problem of mobile robotics is safe navigation in known or unknown environments. The task becomes easier if the environment is known or it is static, in which case deliberative methods can be utilized. On the other hand, if the robot has no prior information, or if it has to navigate in a dynamic world, then it is indispensable to employ a reactive control strategy for real-time performance since pure deliberative control methods have computational overhead that can significantly hinder real-time operation. Or, depending on the situation some hybrid control structure might be necessary to get the best performance.

Unfortunately there is no widely accepted control method for mobile robots that can solve all the problems. In fact, the algorithms developed until now are not even satisfactory, and only performing well under restricted conditions. Therefore, a well performing method with as few as limitations as possible is needed, and research is going on to suggest remedies for the problem.

To develop solutions, many researchers have focused on the potential field method as a reactive control strategy due to its simplicity and promising features, working on different variations and suggesting solutions for the problematic cases discussed earlier (section 1.2.4.5). A new variation of potential field is suggested in [6] claiming good performance with some solutions to the problematic cases, supported with simulation results, which in many cases might be misleading. In this work, the suggested algorithm is implemented on a real system with modifications for applicability and improvement, and its performance, problems and potential improvements are analyzed.

The actual goal of this work is, in fact, to develop a system that will be the groundwork for a robot soccer team. Since navigation of robots in such an application is crucial, the system naturally turns out to be a good test bed for experimenting mobile robot control algorithms. The suggested method, setup, and implementation details will be given in the next chapters.

1.2.8 Organization of the Thesis

The proposed algorithm and modifications are explained in chapter 2; the experimental setup and implementation details are described in chapter 3, experimental results are presented in chapter 4, and conclusions with chapter 5.

CHAPTER 2

PROPOSED SOLUTION

2.1 A Brief Problem Definition

In chapter 1, it was stated that a widely accepted mobile robot control algorithm is still an open research problem, and it is crucial in all mobile robot applications such as robots designed to navigate in outside terrain for different purposes, or robot soccer.

The aim of this work is to develop a setup that can be the groundwork for robot soccer, and also test bed for experimenting mobile robot navigation algorithms; as well as implement, analyze and improve the potential field based control method presented in [6].

2.2 Suggested Solution

As a solution, an experimental setup is developed and the proposed potential field based control algorithm is implemented on this system with modifications for applicability and better performance. In the following sections and chapters, the details of the experimental system and the method implemented are presented.

2.3 Potential Field With a New Approach

In the previous chapter the potential field concept was described along with its problems; and it was stated that lots of variations have been proposed to eliminate or at least relieve these problems. Such a new approach is suggested in [6], and it will be

described in this chapter with some modifications for applicability and improvements as part of this work.

2.3.1 Mathematical Model of Robot

The robot used in this work is differential drive mobile robot. It is nonholonomic, having restrictions in the way it can move, due to kinematic or dynamic constraints, such as limited turning ability or momentum at high velocities [1]. Assuming no slip at the tires, the kinematic model of such a robot is given by,

$$\left.\begin{aligned} \dot{x} &= v \cdot \cos(\phi) \\ \dot{y} &= v \cdot \sin(\phi) \end{aligned}\right\} \quad \begin{aligned} v &= (V_R + V_L)/2 \\ \omega &= (V_R - V_L)/L \end{aligned} \tag{2.1}$$

where v and ω are the translational and rotational velocities of the robot, VR and VL are the right and left wheel velocities, L is the interwheel distance, and ϕ denotes the orientation of the robot as shown in figure 2.1.

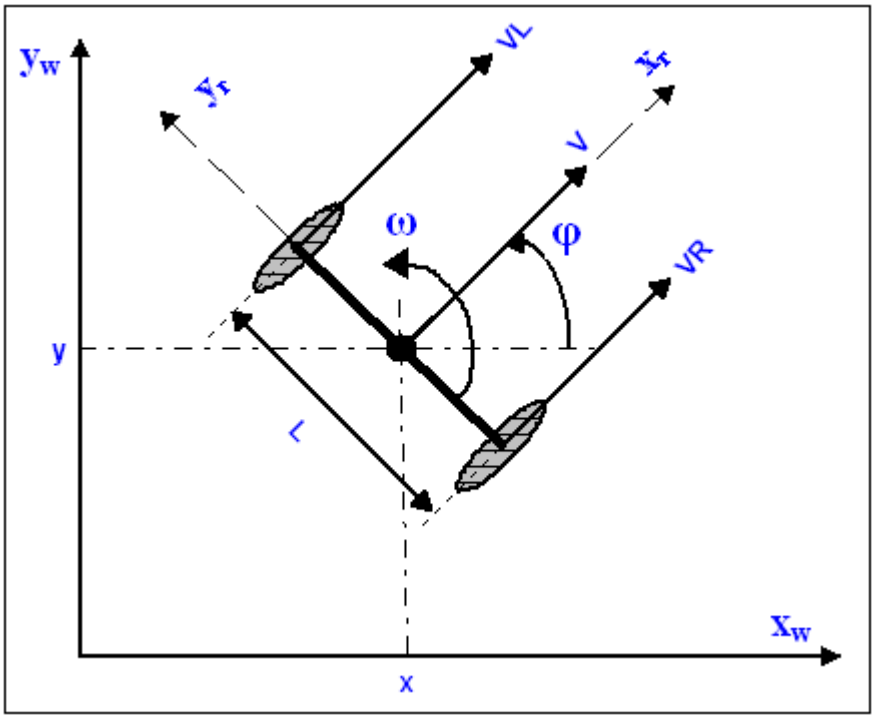

Figure 2.1 Kinematic model of the robot [6].

2.3.2 Sensors

Robots have two kinds of sensors. The fist type is for monitoring the internal state of the robots, such as measuring the motor shaft position or velocity using encoders. The other type is for perceiving the state of their environments, such as ultrasonic distance sensors and cameras to detect obstacles or to locate them in a coordinate frame. A robot can only perceive its environment to the extent that its sensors enable it; therefore, there is always a range outside which the robot is not aware of what is going on. In typical applications, distance sensors are placed around the robot to scan a sufficient area in the vicinity of the robot. If the robot is designed to go only in one direction (forward), then the sensors are usually placed at the front covering an angle of 180 degrees. Figure 2.2 illustrates the sensor range of a robot; the robot can perceive the parts of the obstacles lying within this range.

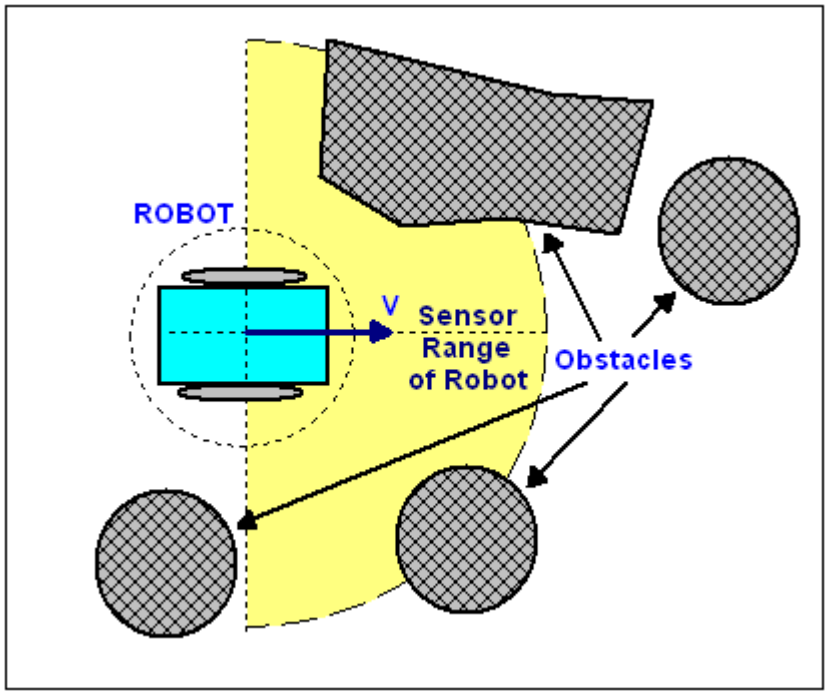

Figure 2.2 Sensor range of the robot.

2.3.3 Formulation of Potential Field

As described in the previous chapter, the main idea of potential field method is to assume repulsive forces from obstacles and attractive forces from the goal to the robot. The repulsive force is inversely proportional to the square of the distance and calculated as,

$$\vec{F}_{obs} = -A \cdot \sum_{i=1}^{n} \frac{1}{d_i^2} \cdot \hat{r}_i \tag{2.1}$$

where A is a constant scaling factor, n is the number of obstacles, d_i is the distance between obstacle i and the robot, $\hat{r}_i$ is the direction vector from robot to representative point of obstacle (e.g., center of obstacle, point of obstacle closest to the robot). This is the original repulsive force formulation of the potential field and it requires the knowledge of all obstacles in the environment. The magnitude of the repulsive force increases significantly when the robot is close to the obstacle, which is the main idea for avoiding the obstacle.

The attractive force of the goal is in the form,

$$\vec{F}_{goal} = B \cdot d^2 \cdot \hat{r} \tag{2.2}$$

where B is similarly a scaling factor, d is the distance from robot to the goal, and $\hat{r}$ is the direction vector from robot to the goal.

Many different forms of these forces are adapted in literature [6, 7, 16, 17, 20]. For example, an exponential function can be used for the repulsive force, and a constant attractive force may be assumed for the goal.

In applications, only the forces due to the obstacles which can be detected by robot sensors contribute to the potential field, that is, robot only feels an egocentric potential field due to objects in its vicinity, and this continuously changing field is recalculated at each sampling interval of the robot's sensors. Incidentally, the coordinate system in [6] is a global x^w-y^w system, while in this work a local r^w-θ^w system (figure 2.3) is preferred for the sake of simplicity in the application.

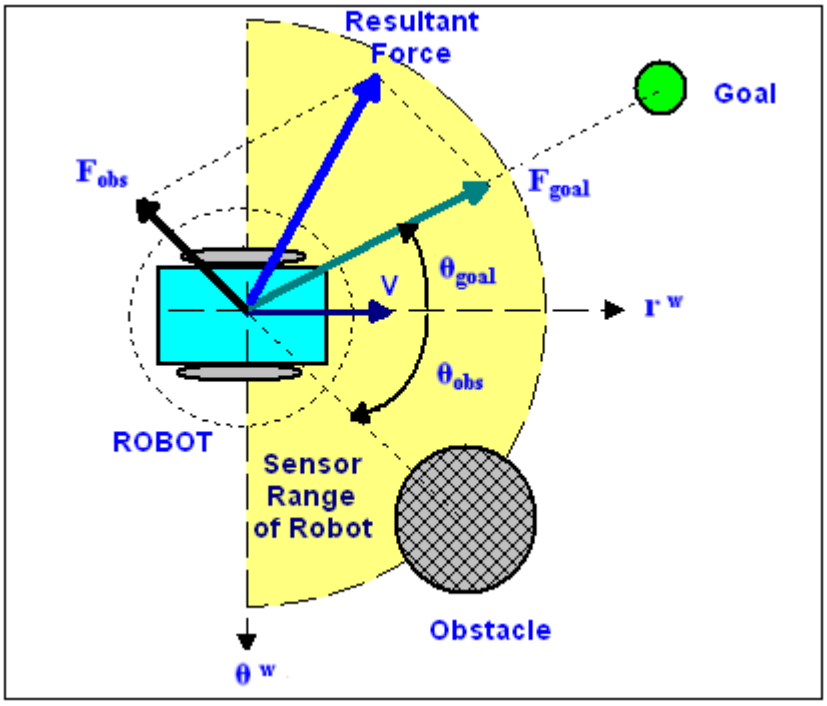

Figure 2.3 Potential field forces on the robot.

After calculating each repulsive and attractive force, the total force on the robot is calculated by the vector sum of these individual forces, and robot moves in the direction of the resultant force vector (figure 2.3). The limitations of this formulation were discussed in chapter 1.

2.4 Design of a Layered Control System

In the proposed new approach a layered structure is adopted, such that instead of adding up the attractive and repulsive forces as in classical potential field method, these forces are treated separately in parallel layers, Obstacle Avoidance (OA) and Drive Toward Goal (DTG) layer (figure 2.4).

Each layer produces its own output using potential field method and these outputs are later combined by the behavior arbitration layer to result in the reference orientation of the robot for obstacle avoidance and goal tracking.

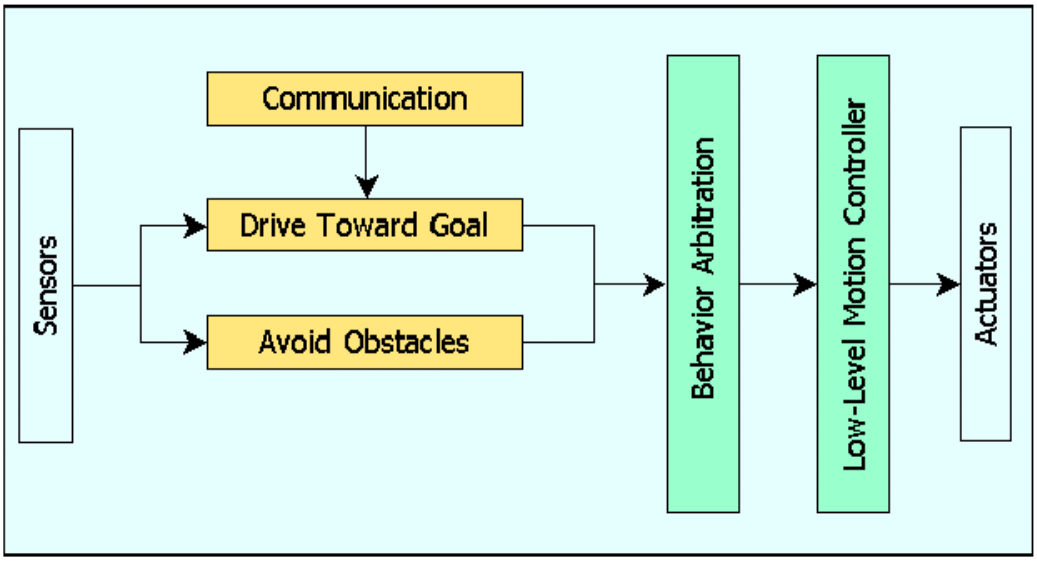

Figure 2.4 Layered control structure [6].

2.4.1 Obstacle Avoidance Layer

Using the outputs of the distance sensors, the net repulsive force is calculated and decomposed to its components, one along the direction of motion of the robot and one perpendicular to it.

$$\begin{aligned} F_r &= \left|\vec{F}_{obs}\right| \cdot \cos(\theta_{obs}) \\ F_\theta &= \left|\vec{F}_{obs}\right| \cdot \sin(\theta_{obs}) \end{aligned} \tag{2.3}$$

For safe navigation, the robot should try to keep the force component along its direction of motion, F_r, minimum or ideally zero. This can be achieved by changing the orientation of the robot, since the force components are dependent on the orientation. To this end, a controller can be used for the optimization. The rate of change of the force components with respect to the obstacle angle is,

$$\begin{aligned} \dot{F}_r &= -\left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \sin(\theta_{obs}) \\ \dot{F}_\theta &= \left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \cos(\theta_{obs}) \end{aligned} \tag{2.4}$$

Then the controls can be chosen as,

$$u_r = \dot{F}_r \quad , \quad u_\theta = \dot{F}_\theta \tag{2.5}$$

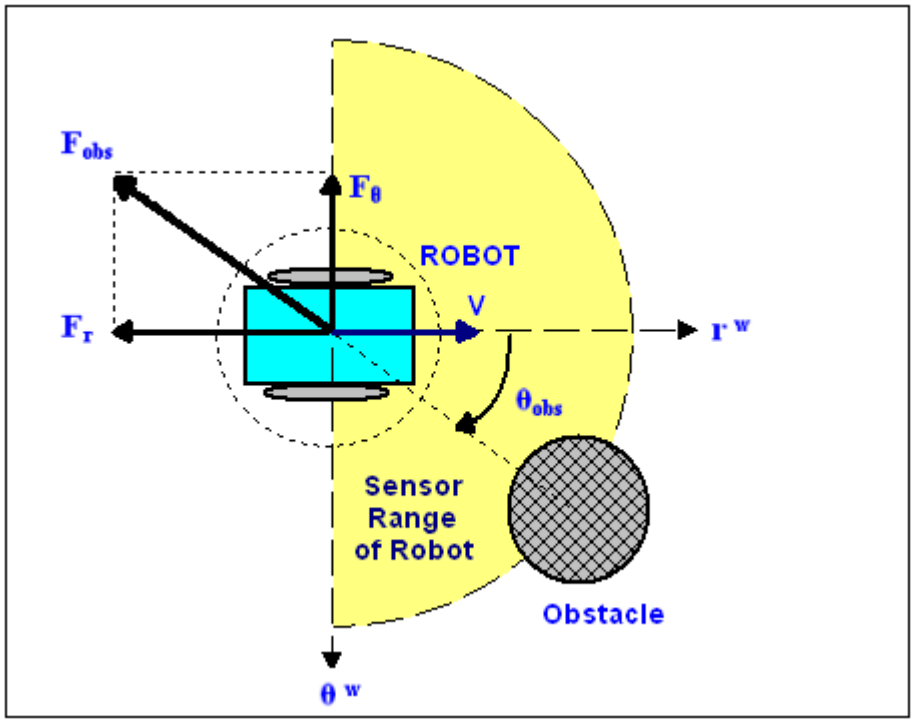

Figure 2.5 Obstacle force components.

and errors to be minimized are,

$$e_r = F_r^{ref} - F_r$$
$$e_\theta = F_\theta^{ref} - F_\theta \tag{2.6}$$

Now, any suitable control method can be utilized for the control of this first order system. In [6], sliding mode controller is used with positive definite Lyapunov function $\gamma = e^T \cdot e/2 \geq 0$. In this work, a simple proportional controller is used, resulting in the following controls.

$$u_r = K_{pr} \cdot e_r$$
$$u_\theta = K_{p\theta} \cdot e_\theta \tag{2.7}$$

where K_{pr} and $K_{p\theta}$ are proportional control gains.

Finally, using equations (2.4) and (2.7) together, the desired orientation of the robot for an obstacle free path can be calculated.

$$\frac{u_r}{u_\theta} = \frac{-\sin(\theta_{obs})}{\cos(\theta_{obs})} \Rightarrow \theta_{OA}^{ref} = \tan^{-1}(-\frac{u_r}{u_\theta}) \tag{2.8}$$

Obviously due to the function arctan(), the output will be in the range -90^0 to $+90^0$.

2.4.2 Drive Toward Goal (DTG) Layer

Similar to the obstacle avoidance layer, an attractive force is calculated and decomposed to its components in the direction of motion and perpendicular to it.

$$\begin{aligned} F_r &= \left|\vec{F}_{goal}\right| \cdot \cos(\theta_{goal}) \\ F_\theta &= \left|\vec{F}_{goal}\right| \cdot \sin(\theta_{goal}) \end{aligned} \tag{2.9}$$

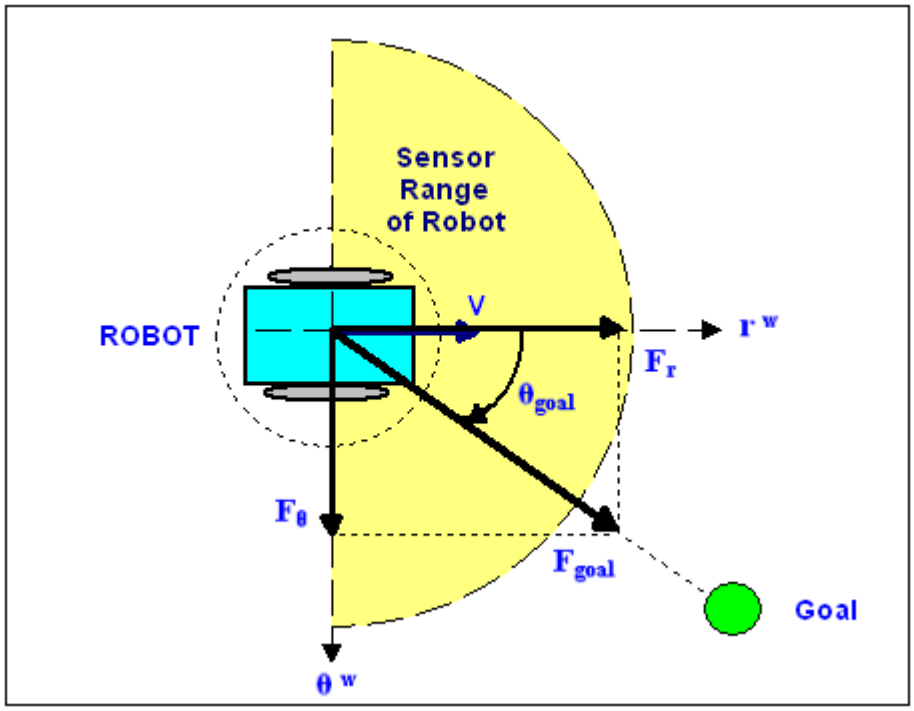

Figure 2.6 Goal force components.

Contrary to obstacle avoidance layer, this time the force along the motion direction must be maximized and the perpendicular force must be minimized.

$$\begin{aligned} F_r^{ref} &= F_{\max} = \left|\vec{F}_{goal}\right| \\ F_\theta^{ref} &= F_{\min} = 0 \end{aligned} \tag{2.10}$$

In this way the robot is attracted to its goal point. Calculations are the same as the OA layer afterwards.

The rate of change of the force components with respect to the goal angle is

$$\begin{aligned} \dot{F}_r &= -\left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \sin(\theta_{obs}) \\ \dot{F}_\theta &= \left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \cos(\theta_{obs}) \end{aligned} \tag{2.11}$$

The controls are chosen as

$$u_r = \dot{F}_r \quad , \quad u_\theta = \dot{F}_\theta \tag{2.12}$$

Errors in force components are calculated

$$\begin{aligned} e_r &= F_r^{ref} - F_r \\ e_\theta &= F_\theta^{ref} - F_\theta \end{aligned} \tag{2.13}$$

The controls are computed

$$\begin{aligned} u_r &= K_{pr} \cdot e_r \\ u_\theta &= K_{p\theta} \cdot e_\theta \end{aligned} \tag{2.14}$$

Finally, the desired orientation of robot for goal tracking is obtained using

$$\frac{u_r}{u_\theta} = \frac{-\sin(\theta_{obs})}{\cos(\theta_{obs})} \Rightarrow \theta_{OA}^{ref} = \tan^{-1}(-\frac{u_r}{u_\theta}) \tag{2.15}$$

2.4.3 Behavior Arbitration Layer

Having calculated two reference orientations, one for obstacle avoidance and the other for goal tracking, the resultant reference orientation of robot should be calculated by the combination of these references, which will be done by the behavior arbitration layer being central to the success or failure of the algorithm. The two outputs might be conflicting. However, the behavior arbitration should combine them in such a way that both obstacle avoidance and goal tracking are partially fulfilled. Dynamic weights that are calculated from the geometric relations between the robot, goal and obstacle are assigned to the outputs of the two layers, and the overall reference orientation is calculated by the addition of them.

$$\theta^{ref} = OA^2 \cdot \theta_{OA}^{ref} + GTr^2 \cdot \theta_{DTG}^{ref} \tag{2.16}$$

The weights OA and GTr in equation (2.11) are complementary, $OA + GTr = 1$, and are calculated by considering the angle between the direction of velocity vector of the robot and repulsive force vector on the robot (figure 2.7).

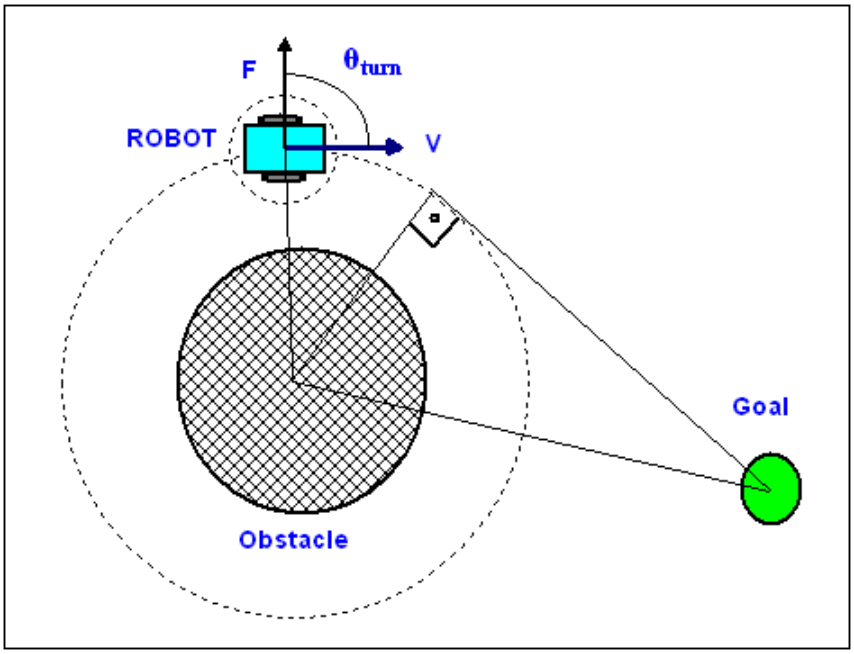

Figure 2.7 Behavior arbitration, calculation of weights.

$$OA = 1 - GTr = \begin{cases} 1 & if \quad \theta_{turn} = \pi \\ & \vdots \\ 0 & if \quad \theta_{turn} \leq \pi/2 \end{cases} \tag{2.17}$$

This formulation shows that when the obstacle is directly in front of the robot ($\theta_{turn}=180^0$), the weight of obstacle avoidance layer is one ($OA = 1$), while the weight of goal tracking is zero ($GTr = 0$). On the other hand, if the robot is moving parallel to the obstacle or away from the obstacle (in which case the obstacle will not probably be in the sensor range of the robot), the weight OA becomes zero and GTr becomes one and gains full priority.

2.4.4 Motion Control Layer

Now that the reference orientation of robot has been calculated, it is time to drive the robot according to this reference for safe navigation. The obstacle avoidance and drive toward goal parallel layers give only the reference orientation of the robot and does not tell anything related to speed of the robot. In [6], velocity reference is taken constant, but it is also stated that an acceleration or deceleration can be implemented depending on how free of obstacles robot's path is.

In the original form of potential field method, the velocity is taken to be proportional to the magnitude of the resultant force. If the attractive force is chosen to be proportional to the square of the distance from robot to goal point, the velocity becomes proportional to the distance to the goal. Accordingly, robot should drive fast when it is far away, and should naturally stop when it reaches the goal. There is of course an upper limit for the velocity for safe navigation. A different approach is implemented in this work, and it will be discussed later in this chapter.

Having the reference orientation, θ^{ref}, and velocity, V^{ref}, a motion controller can be implemented according to the local coordinate system of the robot to calculate the individual wheel velocities assuming the robot is as defined in section 2.1.1.

Similar to the θ^{ref} calculations, separate controls are implemented in r^w and θ^w components. Proportional controller is used instead of sliding mode controller. First, errors are calculated.

$$e_r = r^{ref} - r \ , \quad e_\theta = \theta^{ref} - \theta \tag{2.18}$$

Where r is the distance from robot to goal point, $r^{ref} = 0$, since the desired distance is zero; θ is the reference orientation generated by behavior arbitration layer, and θ^{ref} is zero. Figure 2.8 illustrates the parameters.

Controls are chosen as

$$\begin{aligned} u_r &= (V_R + V_L)/2 \\ u_\theta &= (V_R - V_L)/L \end{aligned} \ , \quad \begin{aligned} \dot{r} &= u_r \\ \dot{\theta} &= u_\theta \end{aligned} \tag{2.19}$$

where V_R and V_L are right and left wheel velocity references to be used by the controller on the robot, L is the robot's interwheel distance. Then the controls are calculated using a proportional controller.

$$\begin{aligned} u_r &= K_{pr} \cdot e_r \\ u_\theta &= K_{p\theta} \cdot e_\theta \end{aligned} \tag{2.20}$$

where K_{pr} and $K_{p\theta}$ are the proportional control gains. Finally, wheel velocity references can be calculated.

$$\begin{aligned} V_R^{ref} &= u_r - L \cdot \frac{u_\theta}{2} \\ V_L^{ref} &= u_r + L \cdot \frac{u_\theta}{2} \end{aligned} \tag{2.21}$$

On the robot, one more controller is required to drive the robot with these reference velocities.

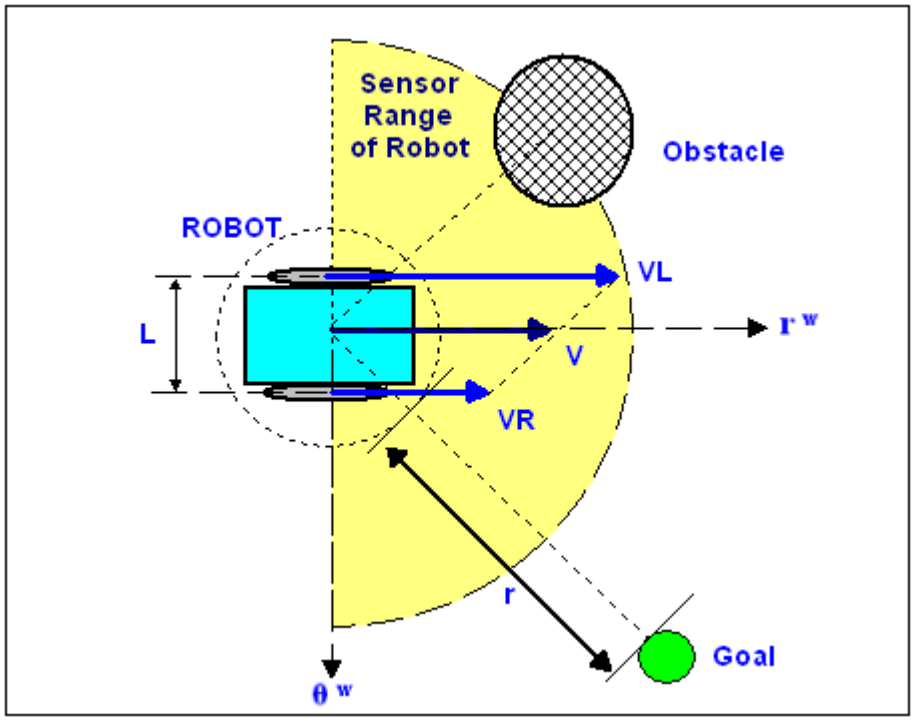

Figure 2.8 Wheel velocities.

2.4.5 Robot Velocity Controller

On the lowest level, robot should follow the velocity references sent by the upper layer. Open loop control is not enough due to disturbances, especially static and dynamic frictional effects. In fact, integrator action is required to overcome the friction. Therefore, a PI (Proportional + Integral) controller (figure 2.9) is used to control the velocity of each wheel separately.

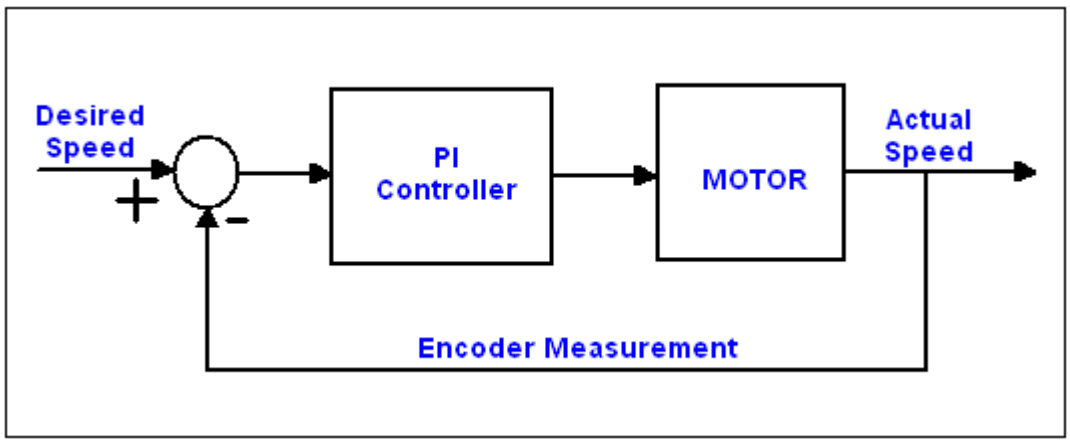

Figure 2.9 PI wheel velocity control.

2.5 Modifications, Improvements

The proposed solution is first implemented as it is with little change and later some modifications are applied for improvements or to eliminate some problems encountered in the experiments.

2.5.1 Modified Behavior Arbitration

The weights of the obstacle avoidance (OA) and drive toward goal (DTG) layers are calculated as being proportional to the angle between the velocity vector direction and repulsive force direction. This might be working well in the simulations where inertia of the robot is probably not considered, time delay between successive samples is small, and there is no noise.

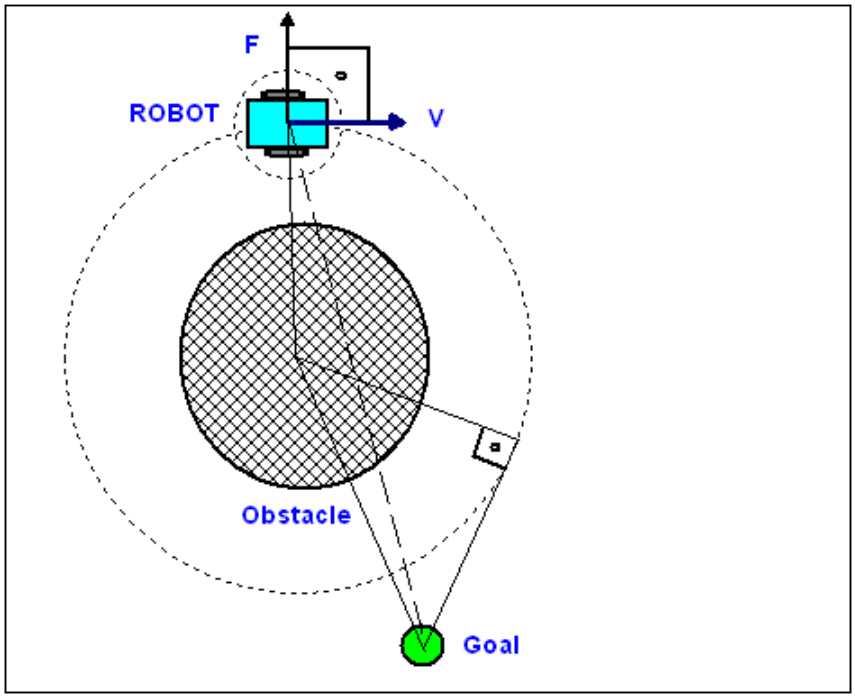

Figure 2.10 Problem in the behavior arbitration.

Considering a scenario as in figure 2.10 where a robot is turning around an obstacle to reach the goal, which is behind the obstacle, the problem occurs as follows: At the instant shown, the robot is moving parallel to the obstacle, in which case the output of OA layer has no importance since its weight is zero. However, due to the position of the

goal, the DTG layer produces a large output as orientation reference, and since its weight is, *GTr=1*, the resulting reference orientation for the robot will be large and in the direction of the goal. This will cause the robot to turn fast in the reference direction. If the robot has large inertia and the sampling time of sensor is also large, this situation will cause the robot to hit the obstacle, or at best will cause severe directional oscillations.

The modification is to change the definition of θ_{turn} in equation (2.18) to make the arbitration more conservative in such cases, which actually occurs frequently.

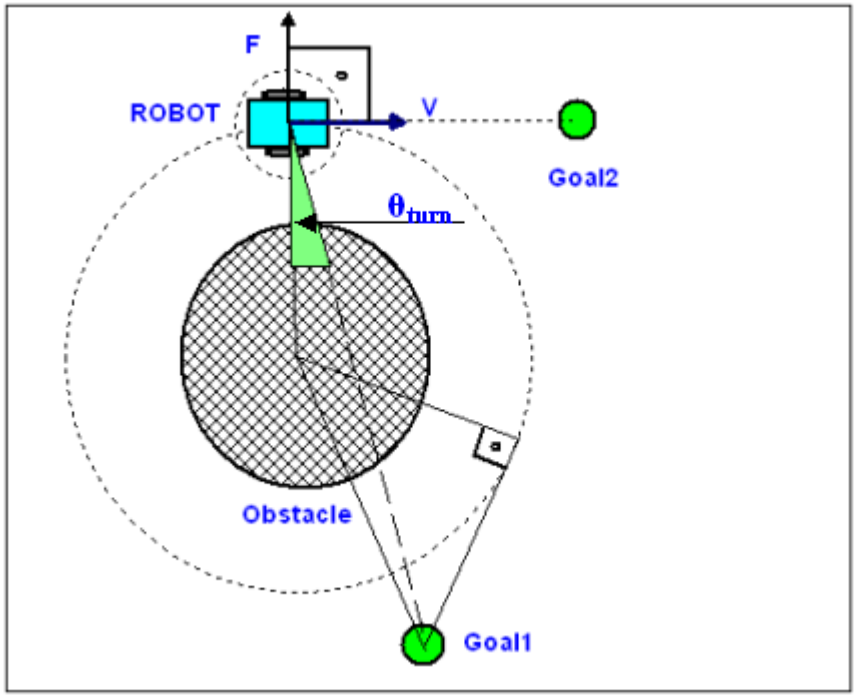

Figure 2.11 New turning angle in behavior arbitration.

The new turning angle is calculated as the difference between the angle to obstacle and angle to goal point, as shown in figure 2.11 for goal point 1. New formulation of the weights becomes,

$$OA = 1 - GTr = \begin{cases} 1 & if \quad \theta_{turn} = 0 \\ & \vdots \\ 0 & if \quad \theta_{turn} \geq \pi/2 \end{cases} \tag{2.22}$$

In this case, the weight of obstacle will be large since the turning angle is small according to equation 2.17, and the robot will continue to turn around it until it comes to a position where the goal is directly in front of it, like goal point 2.

Another problem, which is also inherent in potential field method, is that the robot can not reach the goal if it is located too close to an obstacle, due to large repulsive forces in the vicinity of obstacle resulting in large reference orientation changes calculated by the OA layer. This can be relieved if the behavior arbitration layer takes into account the obstacle and goal distances.

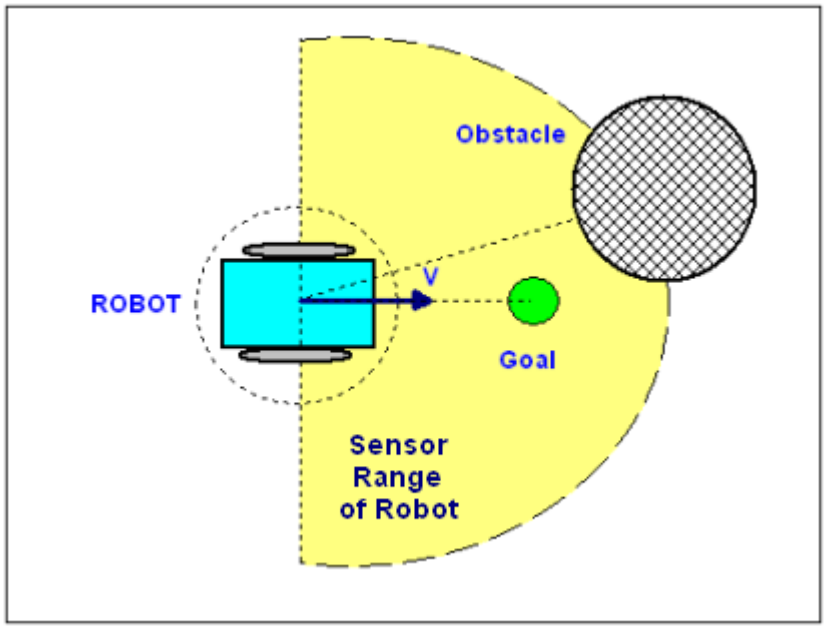

Figure 2.12 Goal is located too close to an obstacle.

In figure 2.12, goal is close to an obstacle. Since the robot is approaching the obstacle, it will be repelled and forced to go away even though the goal is in front of it. In such a case, if behavior arbitration layer sets the weight of OA layer to zero, the robot will be able to reach the goal. This approach only relieves the problem, since for example, if the goal were on the other side of the obstacle, the distance to the goal would not be smaller while the robot is turning around the obstacle, and it will enter an infinite loop turning around the obstacle forever and never reaching the goal.

2.5.2 Velocity Reference Calculation

In [6], a constant velocity reference is suggested, and in classical potential field velocity is proportional to the distance to the goal.

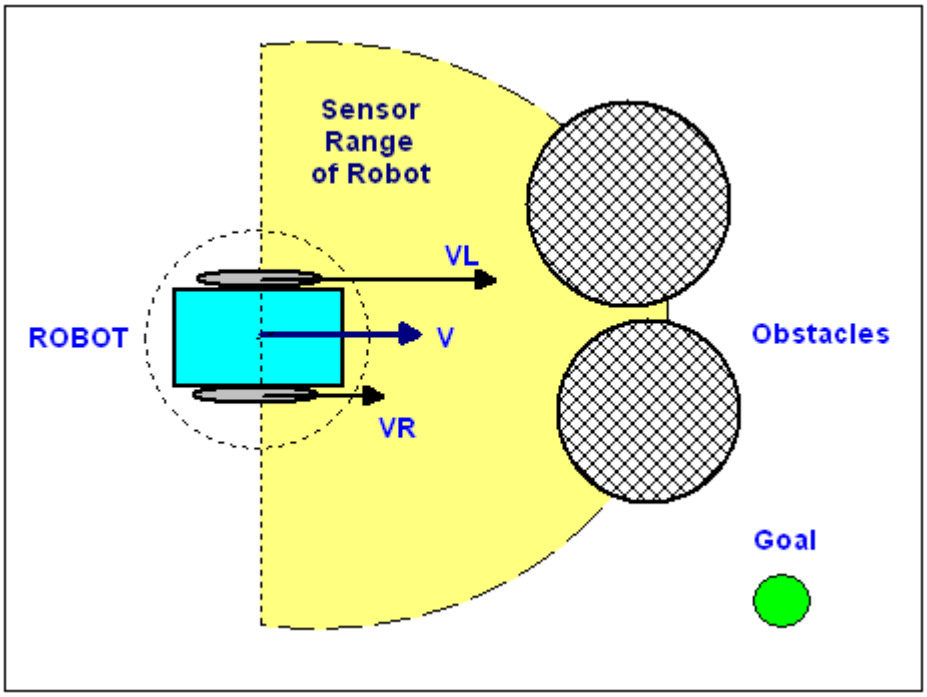

Figure 2.13 Turning ability of robot.

Taking velocity constant, or proportional to the distance to the goal means there is always a linear velocity (V in figure 2.13) of the robot which restricts the maneuvering capability of the robot; for example robot cannot turn around itself or even in small arcs. In figure 2.13 obstacles directly in front of the robot will require the robot to take a sharp turn, but if the linear velocity is large, due for example to large goal distance, it may not be able to turn enough before hitting the obstacle. Therefore, it is necessary to change the velocity of the robot whenever required; for example when robot is close to an obstacle, slowing it down will let it safely avoid the obstacle. This requires establishing a relationship between the linear velocity of the robot and reference orientation it should turn. Although individual wheel velocity references are related to the reference orientation, linear velocity is not, as can be seen from equations 2.19 and 2.21.

In this work, the velocity reference is taken to be proportional to the goal distance and cosine of the reference orientation. Then equation 2.18 will be

$$e_r = r^{ref} - r_{act} \text{ where } r_{act} = r \cdot \cos(\theta^{ref}) \qquad (2.23)$$

where r is the distance from robot to the goal. This way the linear velocity of the robot will automatically decrease when it should take a sharp turn, providing the robot more flexibility. However, the robot must always have a nonzero linear velocity otherwise the robot might oscillate around one point continuously. Therefore, the reference orientation in equation 2.23 needs to be saturated, for example between [-80^0, $+80^0$], so that the robot will have a small linear velocity when it should take a sharp turn (e.g., when an obstacle is directly in front of it), and large linear velocity when it is going straight.

2.5.3 Obstacle Modeling

For simplicity obstacles are usually modeled as circles in 2D or cylinders in 3D space, as shown in figure 2.14, or as simple geometric shapes like rectangles. The idea is to treat the obstacle as a point object and put a safe distance as the radius of the representative circle between the point obstacle and the robot.

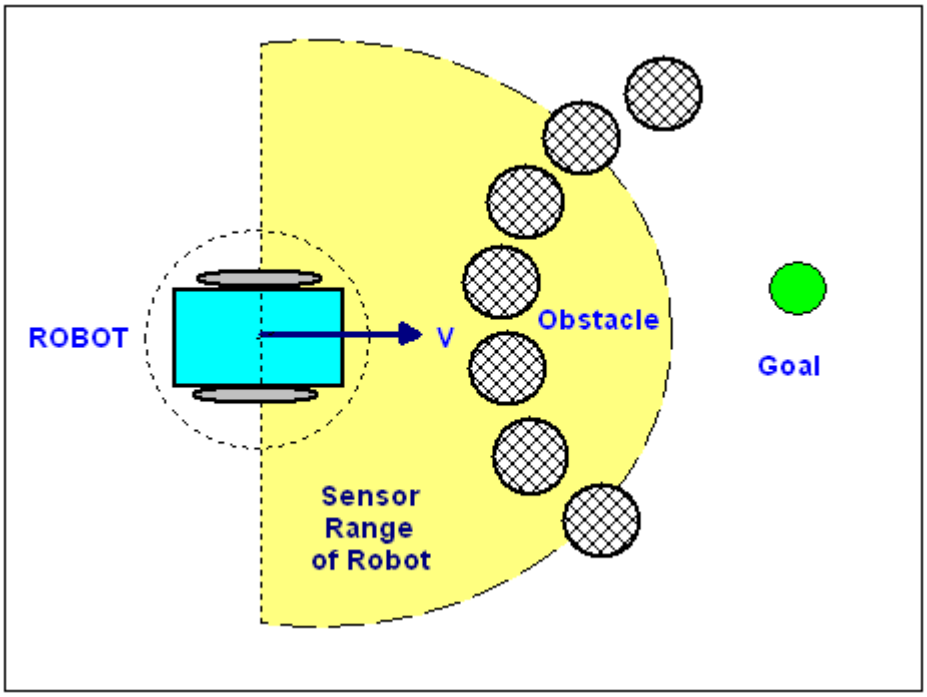

Figure 2.14 Simplistic obstacle modeling.

Although this approach is useful in modeling simple obstacles, which are really nearly circular in shape (e.g., modeling other robots as circular obstacles in robot soccer), or in modeling complex shapes by combining circles (fig 2.14) in simulation, it is not appropriate in modeling complex shaped obstacles in real life as in figure 2.15.

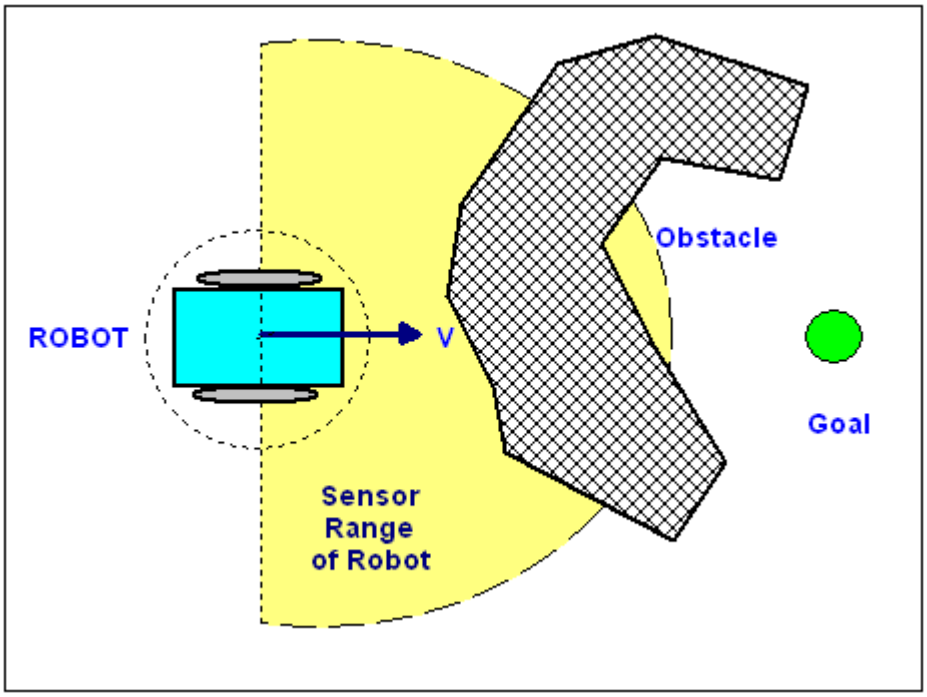

Figure 2.15 Complex shaped obstacle.

How should the robot perceive the continuous and complex shaped obstacle in figure 2.15? Putting a large representative circle around it is obviously not the solution, which will actually enclose the goal as well. Therefore a continuous obstacle modeling is necessary. Heuristics can be developed from this point on.

One approach could be to represent the continuous obstacle as its closest point to the robot, assuming closest point is the most critical for the robot to avoid.

Another approach is to divide the sensor range of robot into discrete pies (e.g., at 15 degrees interval) similar to VFF method (discussed in chapter 1) and taking the closest point in each pie as the representative of the portion of obstacle lying within that pie as shown in figure 2.16. Then, reference robot orientation change for each pie is calculated, and the maximum is selected, since maximum indicates that it is most critical for the robot according to the algorithm implemented. Further, the closest point among all the pies can be calculated and according to on which side of the robot (left, right) closest point lies, the maximum reference orientation on that side is taken as the

reference. Thus, the closest point of an obstacle does not necessarily lead to the maximum reference orientation change of the robot depending its angle to the robot.

These are all heuristics, and new approaches can be developed and tested on the experimental system to see their performance, as is done in this work.

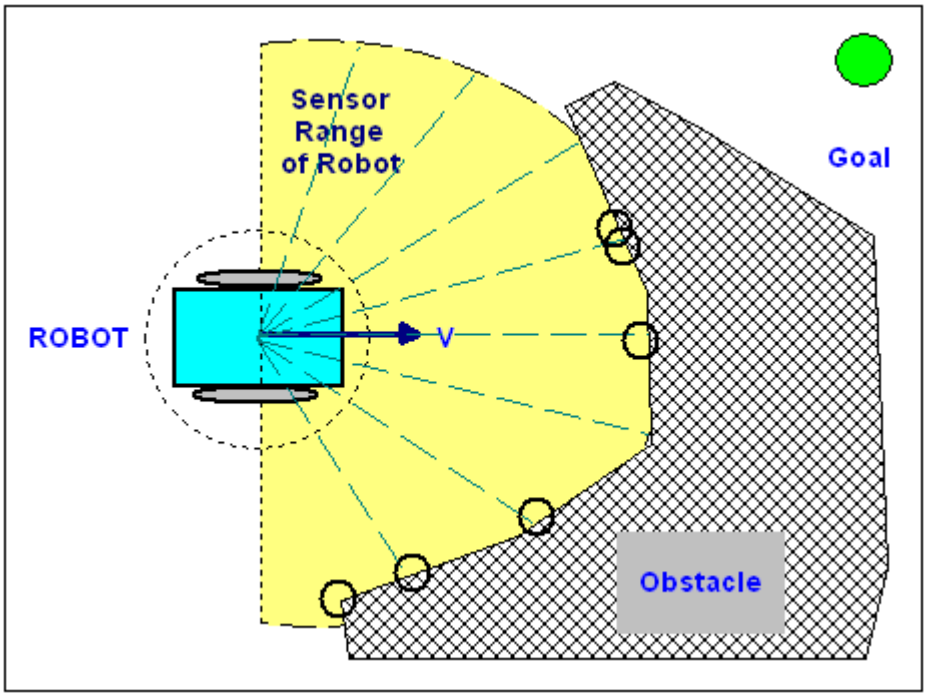

Figure 2.16 Discretizing the sensor range of robot to model continuous obstacles.

CHAPTER 3

EXPERIMENTAL SETUP

3.1 Overview

The heart of this work is the experimental setup, which is specially designed to be extended for robot soccer. It is also a very useful setup to test the mobile robot navigation algorithms, by just plugging in the algorithm to be tested into the software developed. In the following sections, the components of the setup and how it works will be explained.

3.1.1 System Components

System components can be grouped into two: hardware, and software components. Hardware components are:

- Camera and frame grabber
- Personal Computer (PC)
- Mobile robot platform
- Wireless module

Software components are:

- Application running on PC
- Application running on the robot
- Control algorithms implemented within the applications

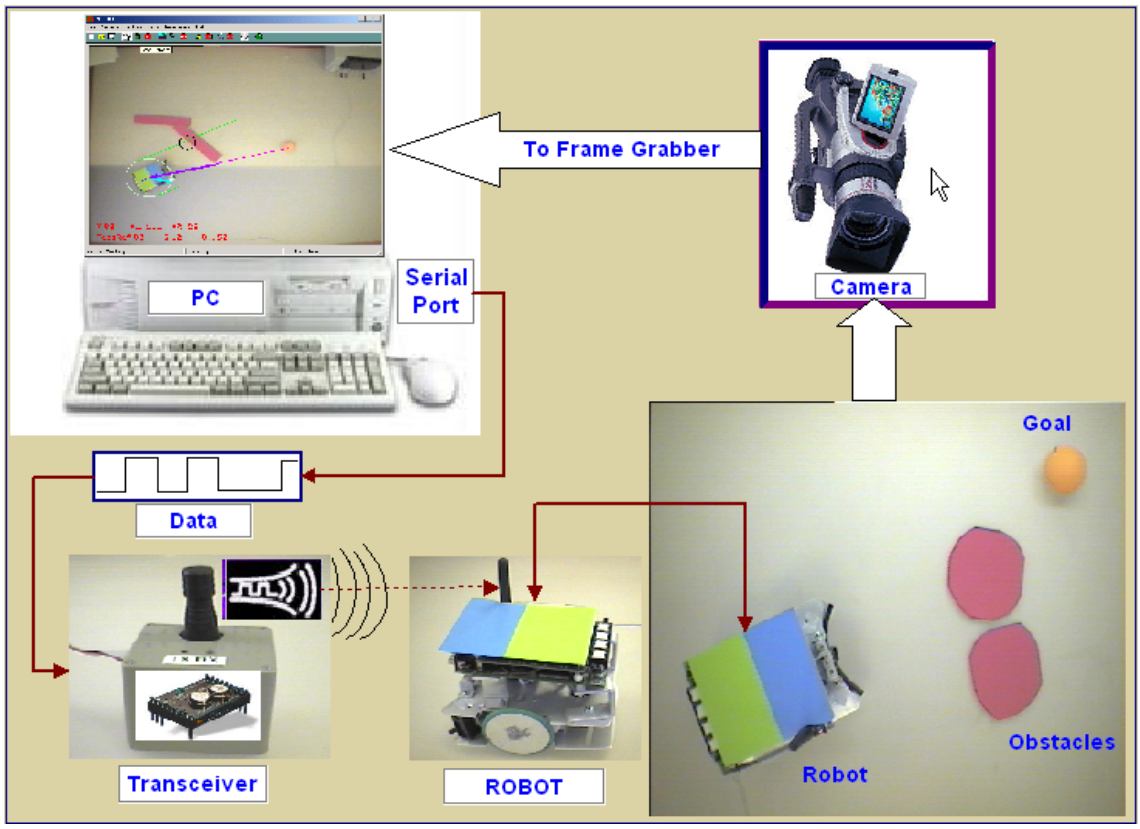

Figure 3.1 Setup hardware overview.

As it is, the system is a typical robot soccer setup, described in chapter 1.

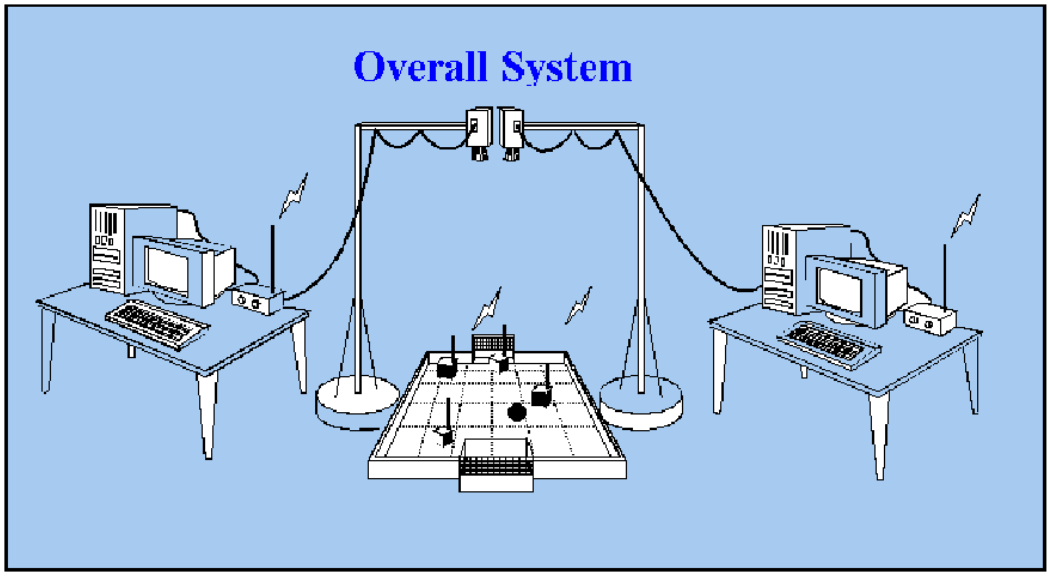

Figure 3.2 Typical robot soccer setup [34].

These components will be discussed in detail in the following sections.

3.2 Data Acquisition, Image Processing

In order to implement the control algorithm described in chapter 2 for mobile robot servoing, certain parameters have to be known by sensor measurements. These are the distance and angle of the goal and obstacles to the robot assuming a local coordinate frame located at the center of the robot. These parameters can be obtained in different ways: ultrasonic distance sensors for obstacle detection and GPS for goal position information, or an overhead global vision camera for both obstacle and goal information. In this application a camera is used due to its advantages over the other methods.

Basically, the system acquires an image of the scene containing the robot, the goal and possibly some obstacles all having different colors as shown in figure 3.3. Then, color image processing is done to retrieve the required information (distances, angles to the robot) out of this image by first finding the center or representative coordinates of these objects. Then, the control algorithm is applied using the acquired information, and resulting reference wheel velocities are sent to the robot via wireless link.

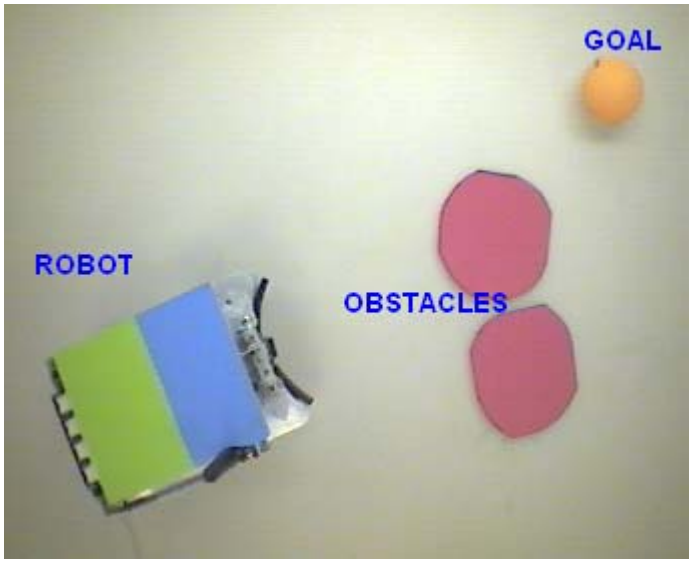

Figure 3.3 Robot, goal, and obstacles to be detected by camera.

All this processing has to be done in **real-time** to be able to control the robot, which is operating in real-time.

The application running on the PC, and doing the required processing is written in C language and on windows operating system. It has a Graphical User Interface (GUI)

created with win32 libraries. The image processing, being central to the implementation, is done using Intel® Open Computer Vision (OpenCV). OpenCV is a cross-platform middle-to-high level Application Programming Interface (API) consisting of more than 300 C functions and few C++ classes that implement image processing operations and some popular image processing algorithms. It is free for commercial and noncommercial use.

The software will be discussed more in detail later in this chapter.

3.2.1 PC Hardware

The main processing power of the system is a PIII 1.0 GHz PC with 512 MB of RAM running Windows 2000. It has a frame grabber installed for image acquisition. The serial port is necessary for wireless communication module and also for downloading the programs to the robot.

3.2.2 Camera and Frame Grabber

Cameras have started to be extensively utilized in robotics (e.g., the main sensor used in robot soccer applications is a local or global vision camera), since they can provide extensive information that can only be acquired using a combination of several other sensors. For example, to implement the potential field algorithm, the position of the goal has to be known. This can be achieved using a GPS (global positioning system). Moreover, to detect the obstacles, some kind of distance sensor has to be used as well. In a system using a global vision camera, all the required information is acquired by interpreting the incoming images. On the other hand, mobile robots cannot use global vision except for some specific examples; therefore, other sensors will always be necessary to complement the deficiencies. One more disadvantage of using camera as a sensor is its large memory and processing power requirements, which will hopefully decrease in near future.

Unlike the one shown in figure 3.1, the camera used in this setup is a simple analog color camera, that can deliver images of maximum 640 by 480 pixel resolution

to the frame grabber, which has a maximum frame rate of 30 frames per second (fps), for the images to be digitized and be ready for processing in PC.

Camera is placed at the top of the field, therefore orthographic projection can be assumed, such that the distances measured at the center and sides of the image will have approximately the same scale.

3.2.3 Color Models

There are several color models in use to represent digitized images, RGB, HSI, CMY, CMYK, YIQ, YUV, etc. RGB is the best-known and most widely used color model, especially in monitors and cameras. In RGB model, each color is represented by 3 values; red (R), green (G) and blue (B), positioned along the axes of the Cartesian coordinate system (figure 3.4). The values of RGB are assumed to be in the range of [0,1] or [0-255]. This way, black is represented as (0, 0, 0), white is represented as (1, 1, 1) or (255, 255, 255).

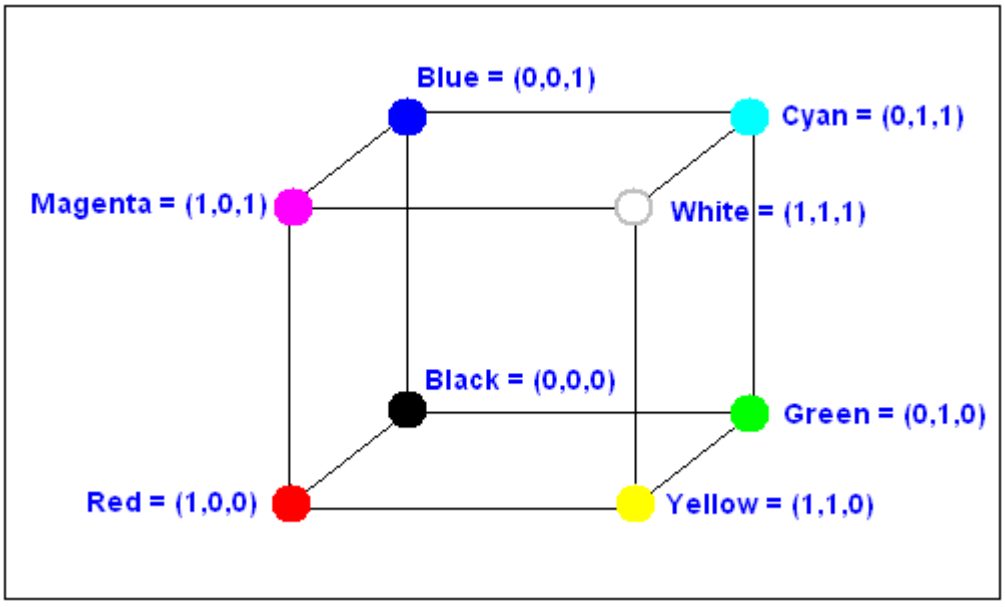

Figure 3.4 RGB Cartesian coordinate system.

The HSI color space is a very important and attractive color model for image processing applications because it represents colors similar to the way that the human eye senses colors, and it is robust to lighting changes, contrary to RGB model, which is very much effected by changing light intensities.

The HSI color model represents every color with three components: hue (H), saturation (S), and intensity (I). Figure 3.5 illustrates how the HSI color space represents colors.

The Hue component describes the color itself in the form of an angle between [0,360] degrees. The Saturation component signals how much the color is polluted with white color. The range of the S component is [0,1]. The Intensity represents the amount of light and its range is between [0,1] and 0 means black, 1 means white.

The conversion between the RGB and HSI color models is somewhat complex; therefore, it is not given here. OpenCV library has functions to do the conversion automatically.

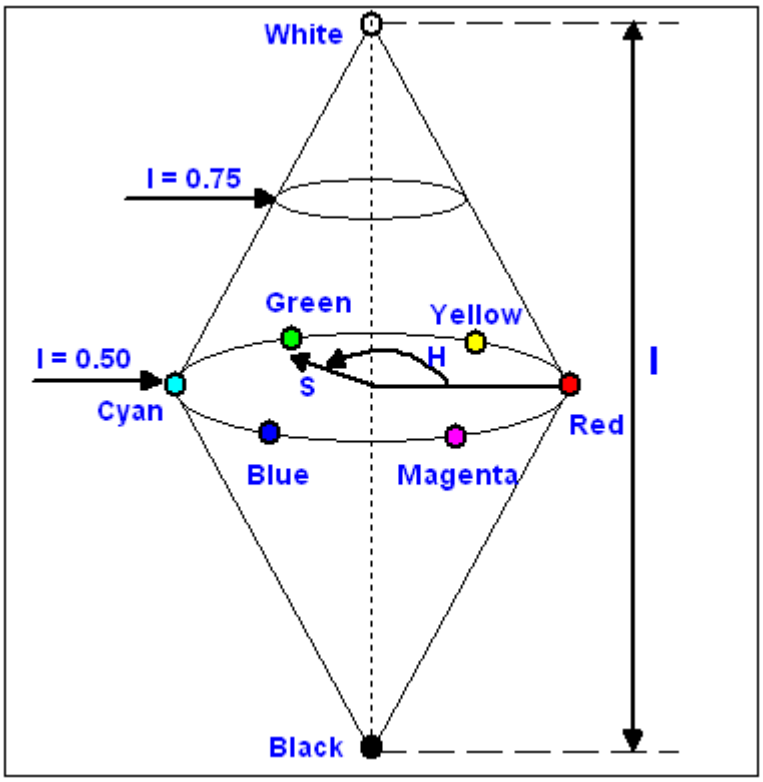

Figure 3.5 The HSI color space.

3.3 Image Acquisition and Processing Details

As was discussed above, robot, goal and obstacles all have different colors for easy identification. Two patches of different colors (e.g., green, blue) on the robot are used to find its direction, since two points are required to define a direction vector.

Images from analog color camera are digitized by the frame grabber. The digitized images are then converted from RGB to HSI color space.

After conversion to HSI color space, the individual hue and saturation channels are retrieved and segmented by thresholding and a logical 'AND' operation is applied to resulting two images to get the desired connected components for the desired color. A *connected component* in an image is defined as the region with pixels having the same values. Four-connected or eight-connected regions are formed if pixels on the four or eight neighbors of each pixel have the same value respectively. The process of acquiring the connected components is named *segmentation*, segmenting the image into regions with the same properties (e.g., same color, edge). Segmentation of an HSI image is described in figure 3.6.

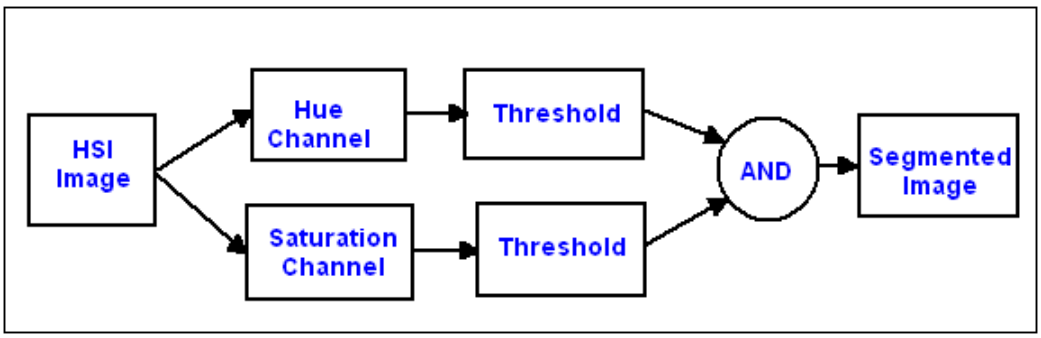

Figure 3.6 Segmentation of HSI image.

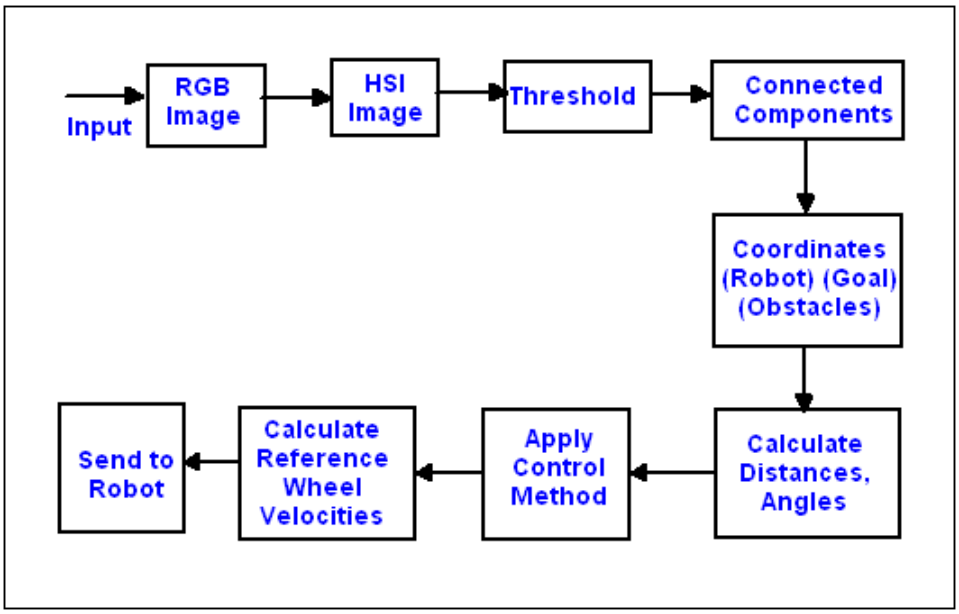

Figure 3.7 Processing sequence.

The sequence of operations from acquisition of an image from camera to wheel velocity calculations are summarized in figure 3.7 as,

- Acquire an RGB image
- Convert the RGB image to HSI image
- Apply segmentation (figure 3.6)
- Retrieve connected components
- Apply area thresholding to get rid of spurious objects (noise)
- Find the center coordinates of connected components (robot, goal, obstacles)
- Calculate the required distances and angles
- Apply the control method discussed in chapter 2.
- Obtain the wheel velocity references and send them to the robot

These sequences of operations will now be illustrated step by step with a sample image in figure 3.8 taken from the experimental setup, as the system is working in real-time.

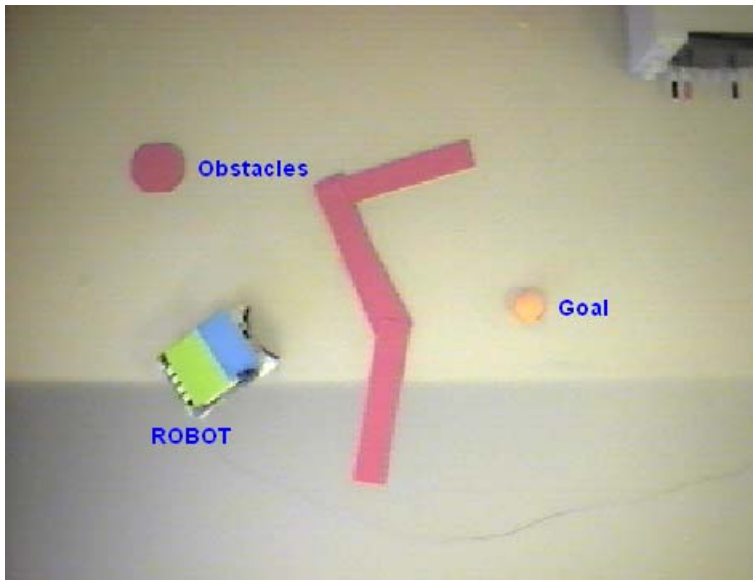

Figure 3.8 An RGB image of robot, goal and obstacle scene.

3.3.1 Finding the Robot Position

The segmentation procedure described above is applied to one of the colored areas on the robot to determine the robot's position within the image. In this case, it is applied to the green patch (on the back side of the robot), and the result is shown in figure 3.9.

In addition to the desired green area, some noise is also acquired. To get rid of this noise an area threshold is applied, such that only areas larger than a threshold are accepted. Connected component analysis, implemented in OpenCV, is used for getting all the connected areas (connected components) in the threshold image, calculating their areas, and finding their center of gravity using moments. Then, the center coordinates of the green area found. Now the other patch (blue in this case) is to be found to locate the robot's position and orientation in the image.

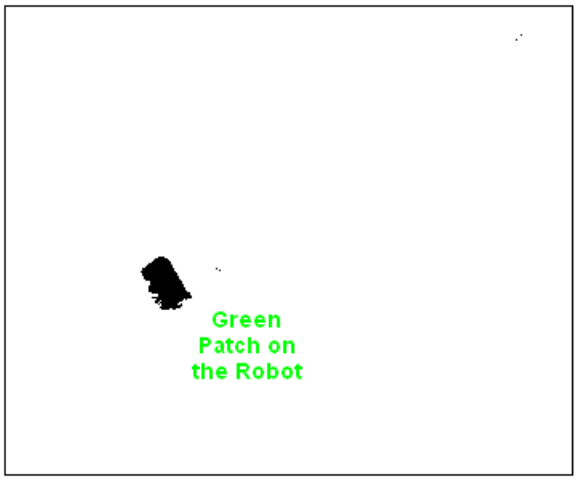

Figure 3.9 HSI image is threshold for green to find the location of robot (color is inverted).

Instead of searching the entire image for blue, a window is placed around the green (figure 3.10) with size proportional to the image size, and only this window is processed.

Figure 3.10 Window placed around the robot (RGB image for better visualization, magnified 2X).

This will not only make processing fast, but will also prevent false alarms. The system is not fooled if green or blue is placed in the scene unless they are placed next to each other as on the robot. In multiple robot soccer teams, the positions of all robots can be determined using a data association rule [18, 32]. Knowing the initial positions of all

robots and maximum velocities they can attain, a robot will be searched in a circle with center at the previous position and radius proportional to their maximum velocity, in the current image frame. This way all the robots can be identified without any problem.

The windowed HSI image is segmented using the same procedure (figure 3.6), and the result is shown in figure 3.11. Then, area thresholding is applied, and center coordinates of blue area are found.

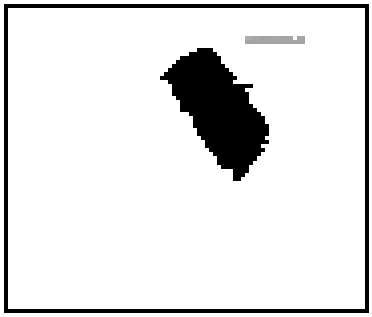

Figure 3.11 Threshold for blue.

Having found the coordinates of the two patches (green, blue), the center coordinates of robot are calculated by taking the midpoint of green and blue center coordinates; the direction of the robot is determined by the vector from center of green to center of blue.

Incidentally, the distance between the two colored patches on the robot serves as a reference to relate the distances from image pixels to actual distances approximately, assuming orthographic projection with the camera at the top. The actual distance in millimeters is measured and a scaling factor is calculated by dividing the distance in terms of pixels to the actual distance. This is important for distance calculations (e.g., robot's radius is proportional to this distance) and wheel velocity calculations. When the camera distance to the field is changed, distance values in pixels also changes. Then a new scaling factor is calculated, thus making the system robust to camera distance changes.

3.3.2 Finding the Goal Position

Next, the coordinates of the goal point are calculated in a similar way by processing the entire image for the predefined color of the goal (e.g., orange). If the goal were static, then calculating its position only once would be enough. However, since both goal and obstacles are not assumed to be static (they can be static or moving), processing for all has to be carried out for each frame.

The distance from robot to goal is calculated between the center point of the robot and center of the goal; the radius of robot and goal is subtracted to find the net distance. The angle to the goal is calculated as the angle between the robot's direction vector and vector from robot to goal, being positive in clockwise direction (figure 3.12).

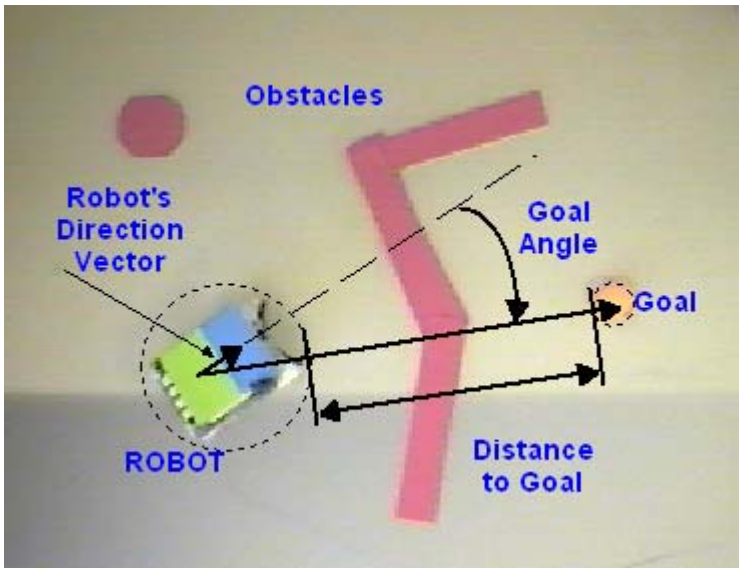

Figure 3.12 Robot and goal, distances, angles.

3.3.3 Obstacle Detection

In the simplistic model of assuming obstacles as point objects and putting a circle around to represent their size, above-mentioned segmentation procedure is applied to the whole image for the color of the obstacles (e.g., red). Centers of all obstacles are found, then distances and angles to the robot are calculated; those within the sensor range of robot affect the robot, while the rest are discarded.

In figure 3.13, an example of simple obstacle segmentation is given. In this approach, only the closest obstacle is taken into account in calculations to apply a repulsive force on the robot.

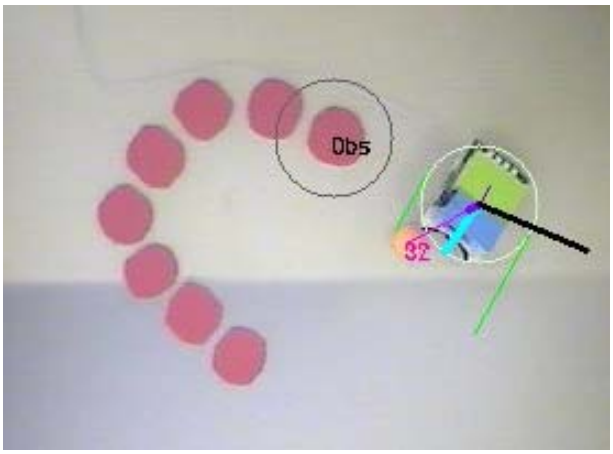

Figure 3.13 Simplistic obstacle modeling. An arc shaped obstacle is decomposed into simple circular obstacles.

Later, a more realistic and a more efficient obstacle modeling is implemented. Knowing the position of the robot and its visibility range (simulating the sensor range of the robot), a window is placed around this visibility range and only this portion of the image is processed for the obstacles.

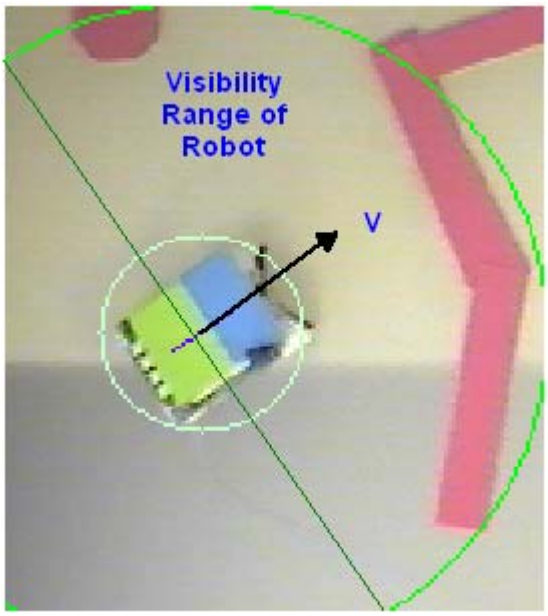

Figure 3.14 A window is fit around the visibility range of robot.

In figure 3.14, the semi circle around the robot demonstrates the limits of robot's sensor range, and only obstacles within this circle are detected. Same segmentation procedure is applied to this window for obstacles; resulting segmented image is shown in figure 3.15. After segmentation, several approaches are possible as discussed in chapter 2.

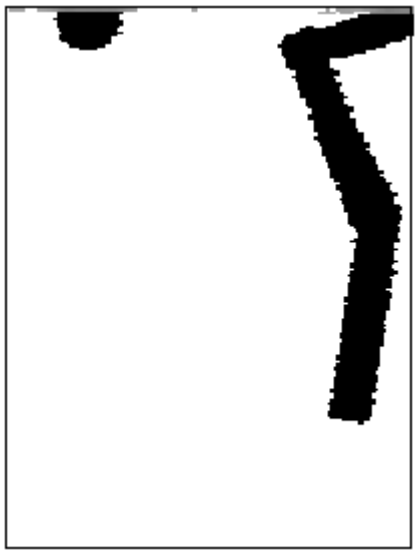

Figure 3.15 Segmentation result of obstacle image.

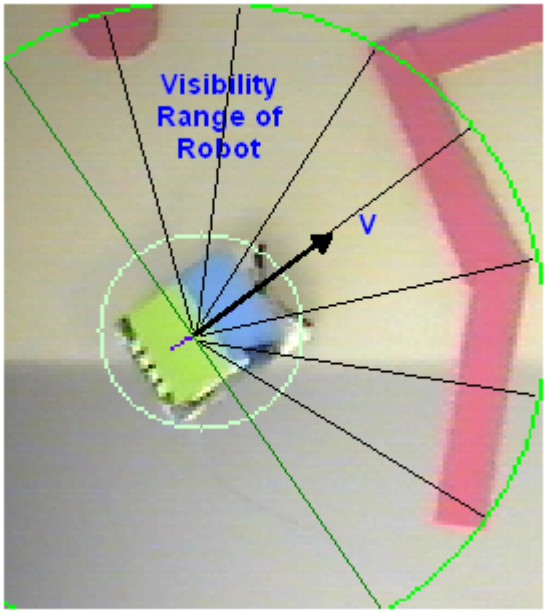

Figure 3.16 Discretizing the sensor range of robot.

3.3.4 Reference Orientation and Velocity Calculations

After calculating the distances and angles to goal and obstacles, and selecting an appropriate method for obstacle representation, the reference orientation of robot is calculated using the formulations given in chapter 2, with a proportional controller implemented as the force controller. Then the reference wheel velocities are calculated using again a proportional controller and sent to the robot via wireless link.

3.3.5 Result

To show the results of calculations visually, a snapshot from the running program is given in figure 3.17. As can be seen, there are the robot, goal, and an obstacle in the scene. After image processing and control calculations mentioned above, actual and the resulting reference orientation of robot, reference wheel velocities are shown graphically as well as numerically on the image. Numeric values V, VL, VR show the linear, left and right wheel reference velocities of the robot in mm/sec; G, O, and TetaRef show the reference orientations from the DTG, OA and the resulting orientation after the behavior arbitration layer respectively.

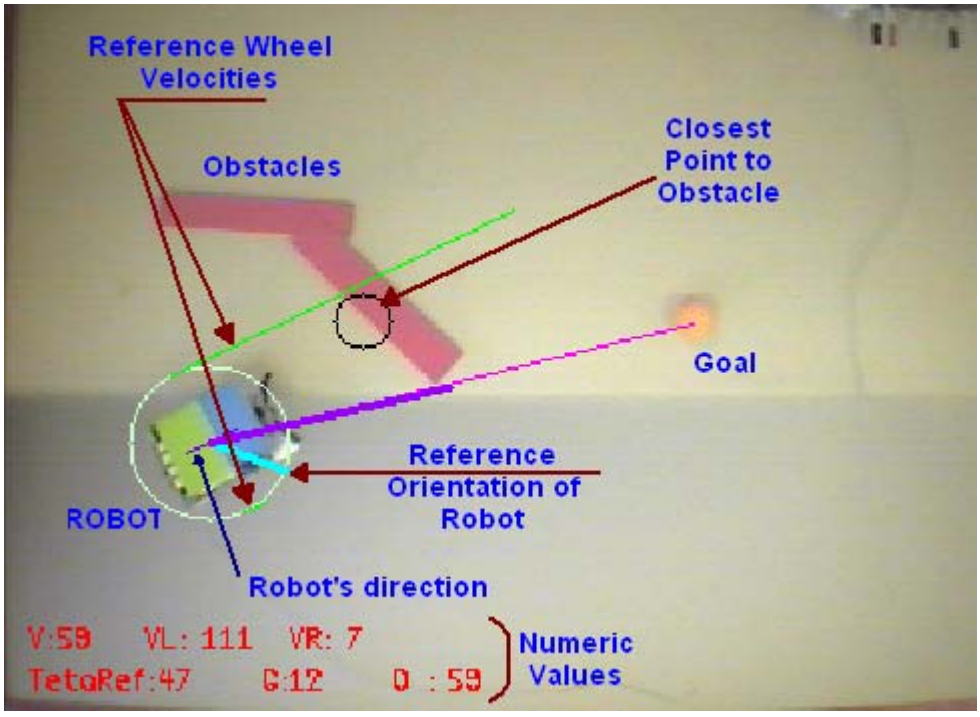

Figure 3.17 A snapshot from the running program,

showing the results of calculations on the image.

Figure 3.17 shows the implementation result, considering the closest point of the obstacle.

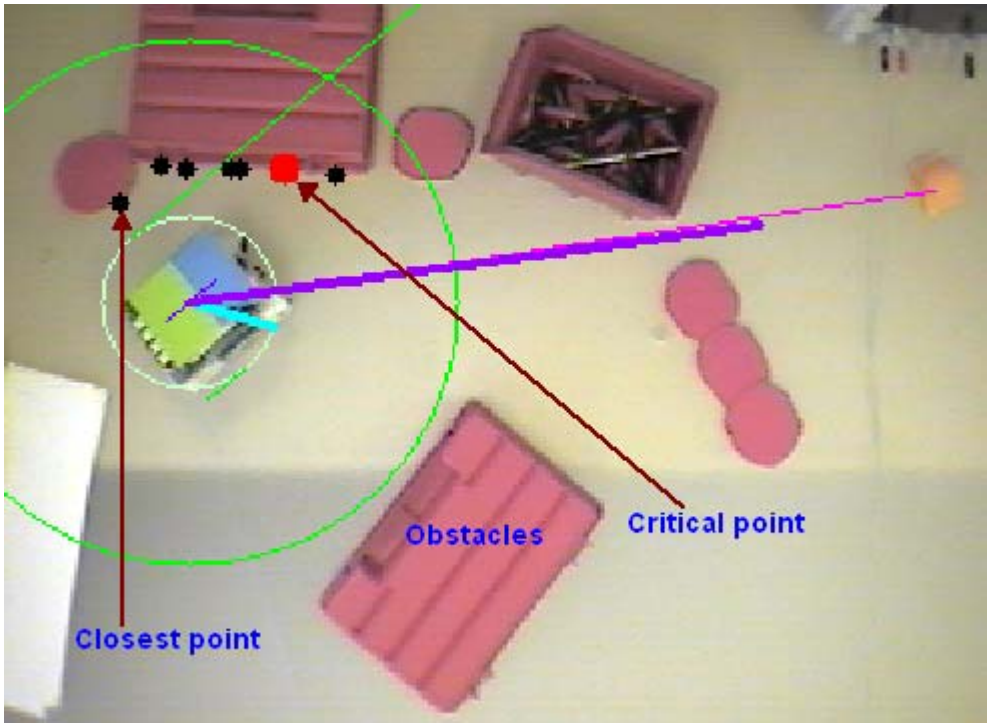

Figure 3.18 Discretizing the sensor range of robot.

Figure 3.18 shows the discretization of sensor range of robot. In this case, instead of considering the 'closest point' (shown in the image), most critical point on the side of the closest point (left of the robot in this case) is considered in the calculations.

Experimental results demonstrating the performance of different approaches are presented in chapter 4.

3.4 Communicating with the Robot

Communication is achieved through wireless link, using wireless modules connected to the serial ports on the PC and robot, having 433 MHz carrier frequency and a baud rate of 9600 bits per second (bps).

Miniature UHF radio modules capable of half duplex data transmission at speeds up to 40 kbit/s, manufactured by BiM, are used on both the robot and PC side connected

to the serial ports. Thus, RS232 data is transmitted and received. Data must be packetised with no gaps between the bytes. Packets must be formed with the following rules:

- Data must be preceded with >3ms of preamble (with the value of 55h or AAh) to allow the data slicer in the receiver to settle.
- Follow by 1 or 2 bytes of value FFh to allow the receive UART to lock.
- Follow by a unique start of message byte (of value 01h).
- The actual data bytes
- The CRC or checksum.

Figure 3.19 Wireless communication packet structure.

In this work, data consists of only 3 bytes: 1 control byte, and one byte for left and right wheel velocities each. This brings a restriction on the largest velocity that can be sent, (-128 mm/s to +128mm/s), which is practically enough for this application. Larger values can be achieved (e.g., maximum 127 cm/s) in return for lower speed resolution.

3.5 Robot

A picture of the original robot used in the experiments is shown in figure 3.14. It is designed for robot soccer, having an onboard vision camera integrated to the controller board, EyeCon, of the robot. However, the frame rate is so low (2-3 fps) that it is not satisfactory, since it cannot move at relatively high speeds due to the speed limit of the camera. Therefore, this camera, as well as the PSD distance sensors in front of the robot are not used; instead a global vision camera provides all the information to the robot, as has been discussed above.

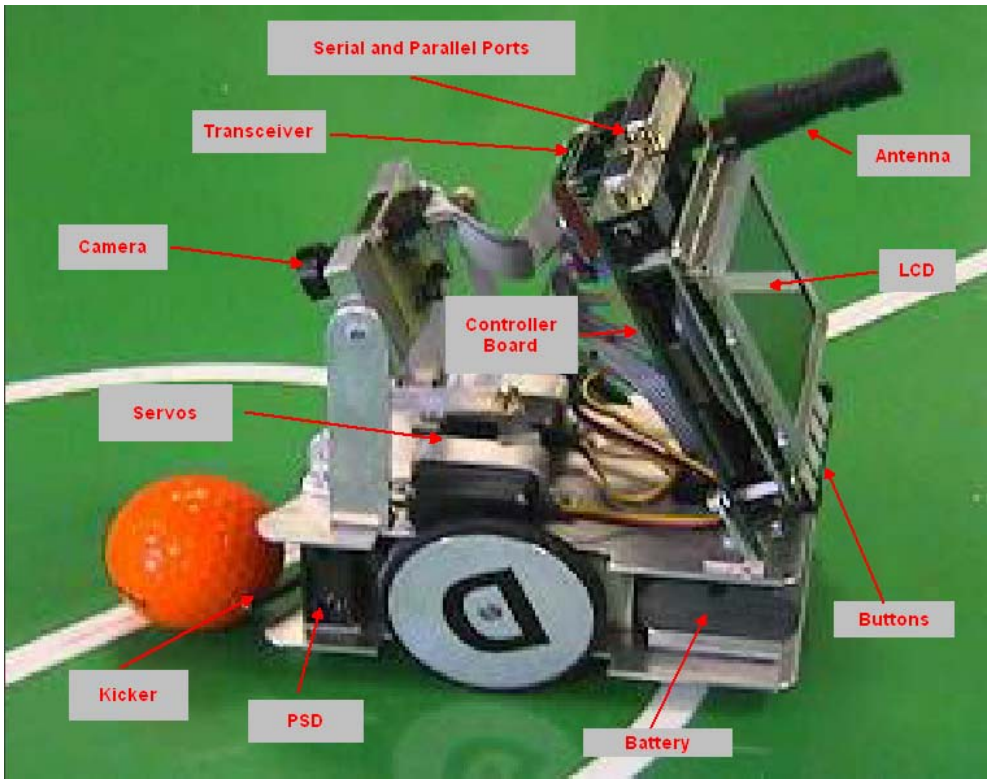

Figure 3.20 Original robot, Eyebot [27].

The robot has a 25 MHz 32 bit controller (Motorola M68332), 1 MB RAM, 512 KB ROM (divided into three slots of 128 KB), 1 parallel port, 3 serial ports, 128 by 64 pixels graphics LCD and 4 input buttons integrated on its controller board [2].

The EyeCon controller of the robot runs its own operating system named RoBIOS (Robot Basic Input Output System) that resides in controller's flash ROM. RoBIOS combines a small monitor program for loading, storing, and executing programs with a library of user functions that control the operation of all on-board and off-board devices. The library functions include displaying text graphics on the LCD, reading push-button status, reading sensor data, reading robot position data (very important for the control of the robot), driving motors, etc. It also supports thread based multitasking system with semaphores for synchronization [2].

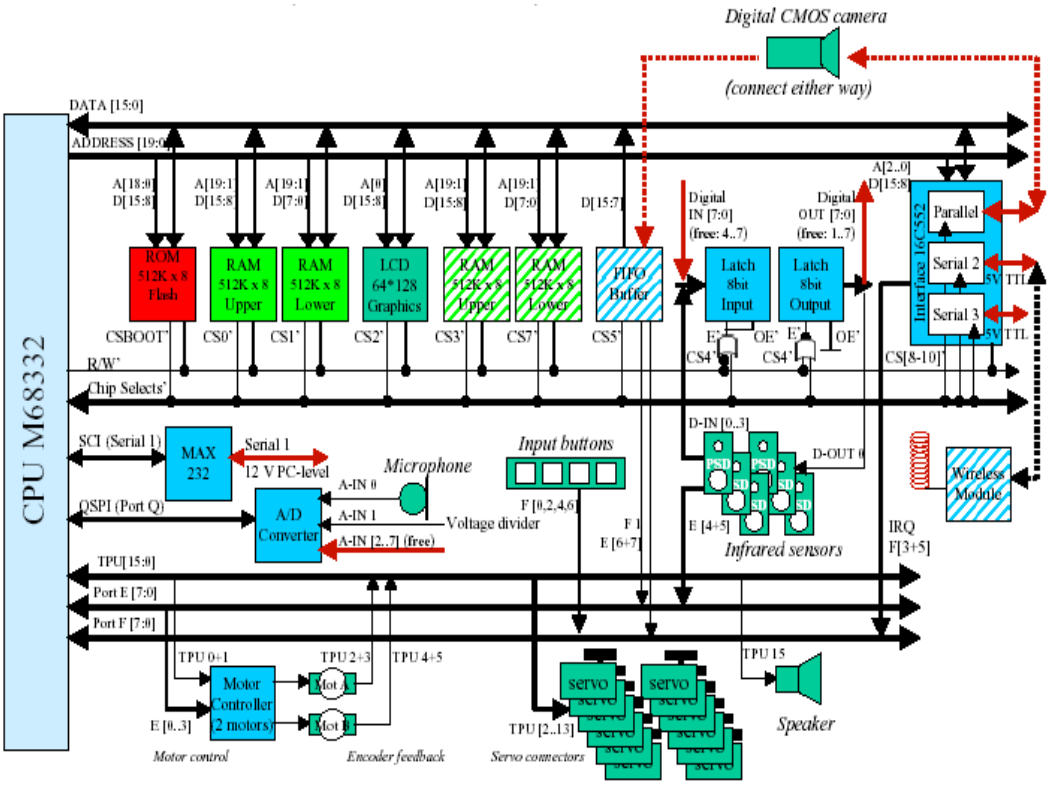

Figure 3.21 EyeCon schematics. [27].

The program running on the robot side is written in C language, utilizing the libraries provided to interface the robot hardware and software. It is compiled for the Motorola processor, and hex code is downloaded to the robot RAM. To keep the program for later use, it can be saved to one of the 3 slots of ROM.

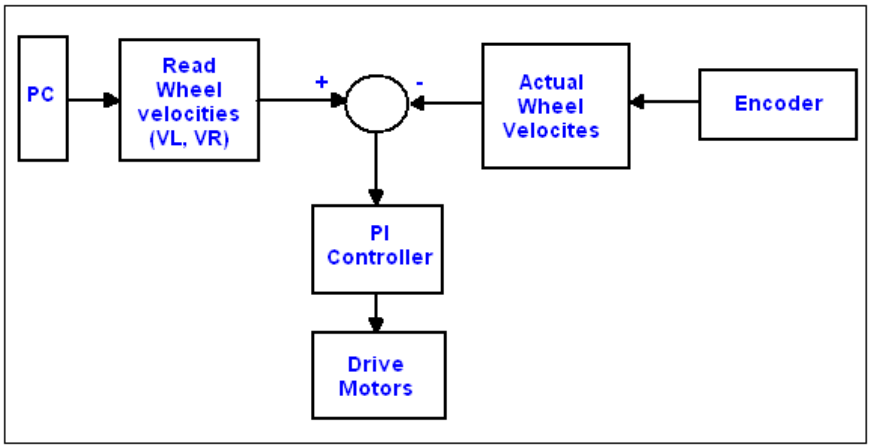

Figure 3.22 Wheel Velocity Control.

The program has two main purposes: it reads the incoming data continuously and at the same time implements a PI (Proportional + Integral) controller for each wheel

velocity to keep them at the desired level, taking as reference the data sent from the PC (figure 3.22).

A timer interrupt calls the controller function to implement the PI controller every 10 milliseconds.

3.6 Software Details

The heart of the setup is the software developed (on both the PC and robot side) to do all the processing.

3.6.1 Application Running on the PC Side

The application running on the PC side does all the processing work from image acquisition and processing, through control algorithm implementation to sending the final wheel velocity references to the robot (figure 3.23).

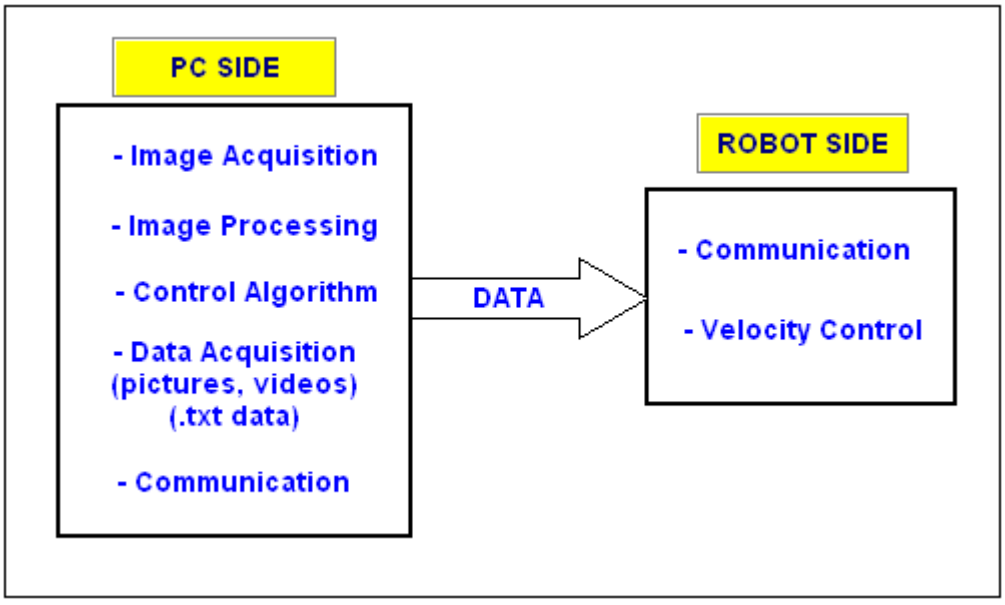

Figure 3.23 Processing on PC and Robot side.

The software is developed on windows 2000 platform, using Microsoft Visual Studio 6.0 Integrated Development Environment (IDE), compiler, linker, etc. The

Graphical User Interface (GUI) is created using win32 API, and image processing is performed using OpenCV. OpenCV dynamic link libraries (dll) have to be installed on the system for the program to work.

The software can be ported to Unix/Linux environment by changing the operating system related portions of the code, e.g., GUI, serial port handling. OpenCV has exactly the same libraries for Unix/Linux; therefore, image processing part need not be changed.

3.6.1.1 Structure and Flow of Program

The program consists of different modules, each having different tasks. A snapshot showing the different modules is given in figure 3.24. The program is an example of event-driven programming. After the creation of application instance and GUI, the program enters an endless loop waiting for user input to process. When the user starts the camera, it enters the image acquisition and processing loop and at the same time monitors and processes user input in the background.

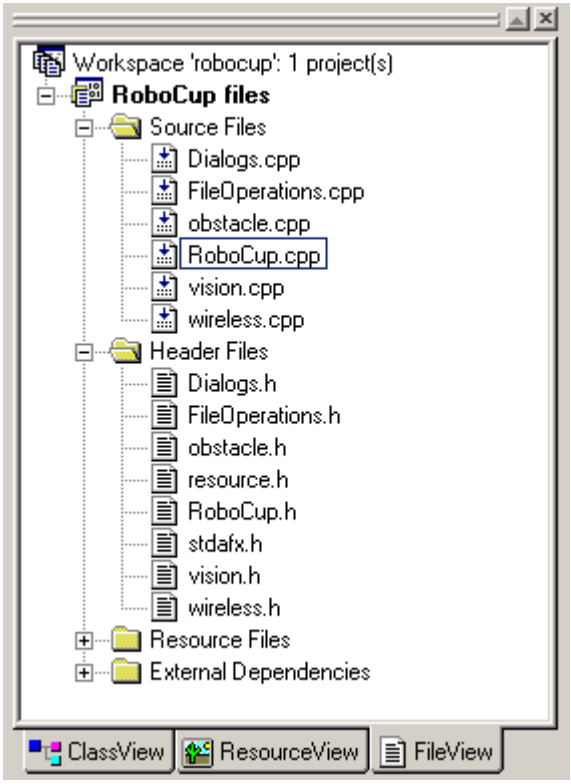

Figure 3.24 A snapshot of file view of project from Microsoft Visual Studio 6.0 IDE.

- RoboCup.cpp: The entry point of the program, where the application instance and GUI (windows, menus, toolbar, status bar, etc.) are created.
- vision.cpp: This file contains the main loop of the program which starts running when the camera is activated by the user. It initializes, starts and stops the camera. It has a callback function that acquires each incoming image, and does all the processing by calling the functions in other modules (e.g., obstacle.cpp, wireless.cpp).
- Dialogs.cpp: processes all the inputs from the user (from menus, toolbar, or accelerator keys), creates dialog boxes to acquire input from the user to change the parameters while the program is running or idle.
- obstacle.cpp: implements the robot control algorithm and calculates reference orientation and reference wheel velocities of the robot.
- wireless.cpp: responsible for the wireless communication between the PC and the robot. It opens and closes the serial port; and creates the packets according to the requirements, and sends them whenever called from other modules.
- FileOperations.cpp: responsible for creation, opening, saving, closing of files (e.g., text files, image files, AVI files).

3.6.1.2 Graphical User Interface (GUI)

Much effort has been spent on the program to make it as interactive as possible for easy use during the experiments. The complete GUI of the program is shown in figure 3.25.

The GUI enables the user to interact with the program while it is running; for example, to change the control parameters online and see the result immediately. For this purpose, dialog boxes are created to get inputs from the user. The program allows the user to do the following:

- Start, stop the camera.

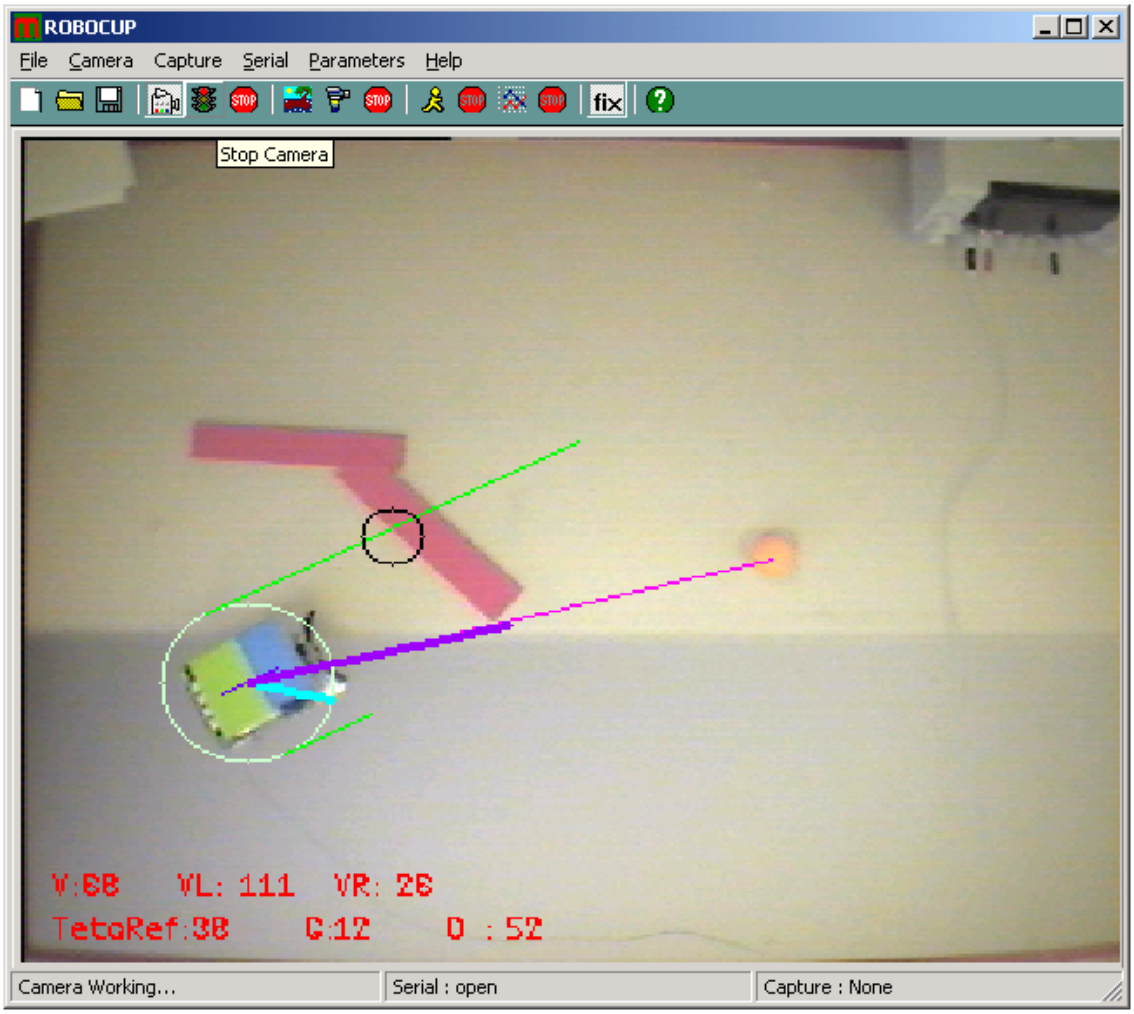

Figure 3.25 A snapshot from the GUI of the PC side application.

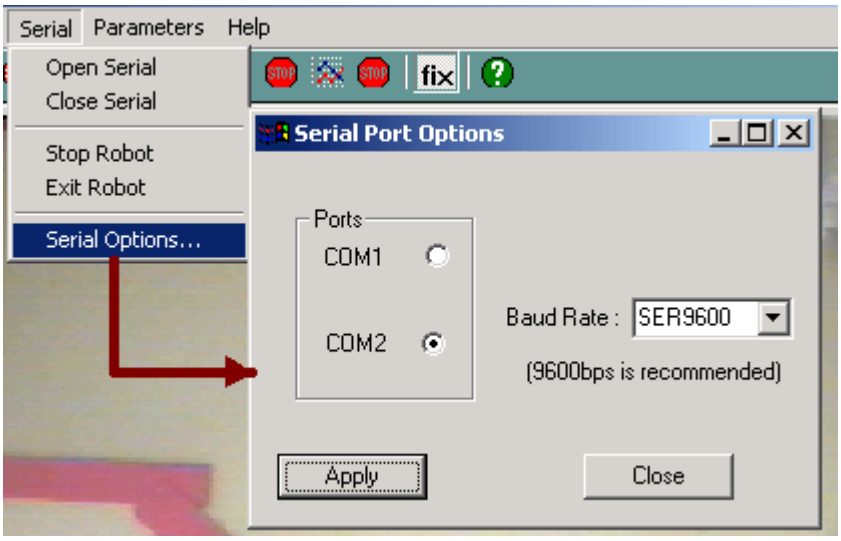

Figure 3.26 Serial port menu and dialog box.

- Change serial port options, stop the robot, exit the program running on the robot (figure 3.26).
- Capture: take image snapshots at any time while the program is running, capture the motion history of robot, capture AVI movies with various compression options, save the values of some variables (forces, orientations, and errors used in the control algorithm) to text files for further processing. These capabilities are accessible through both menus and toolbar buttons (figure 3.27).

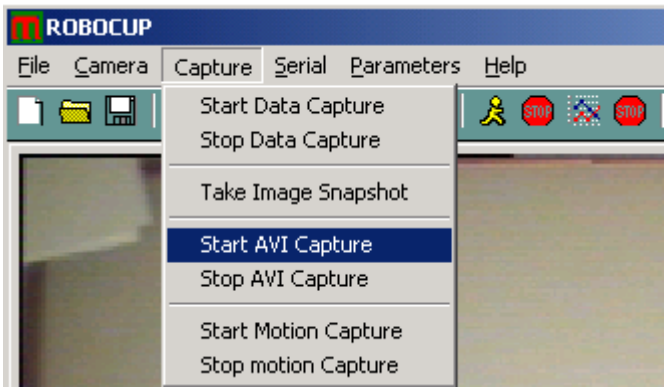

Figure 3.27 Capture abilities of the program.

- Change control parameters (figure 3.28).

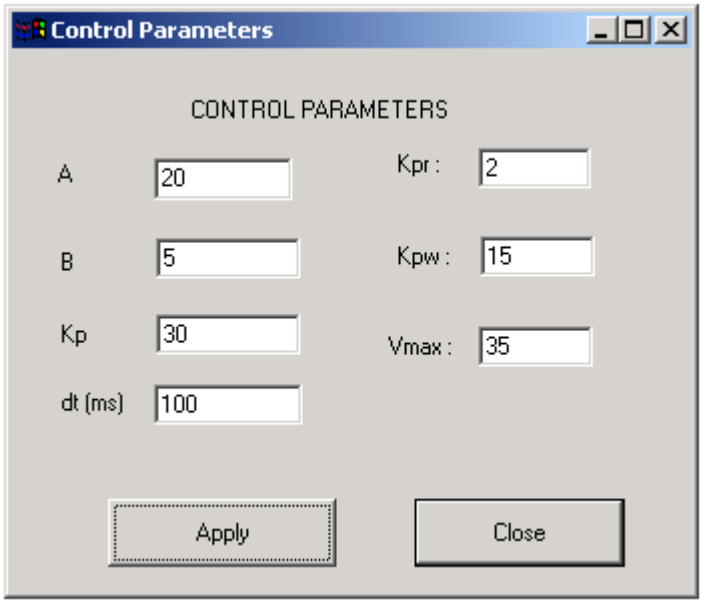

Figure 3.28 Control parameters dialog box.

- Change image processing parameters (threshold values). These changes take effect immediately when the user selects to apply them (figure 3.29).

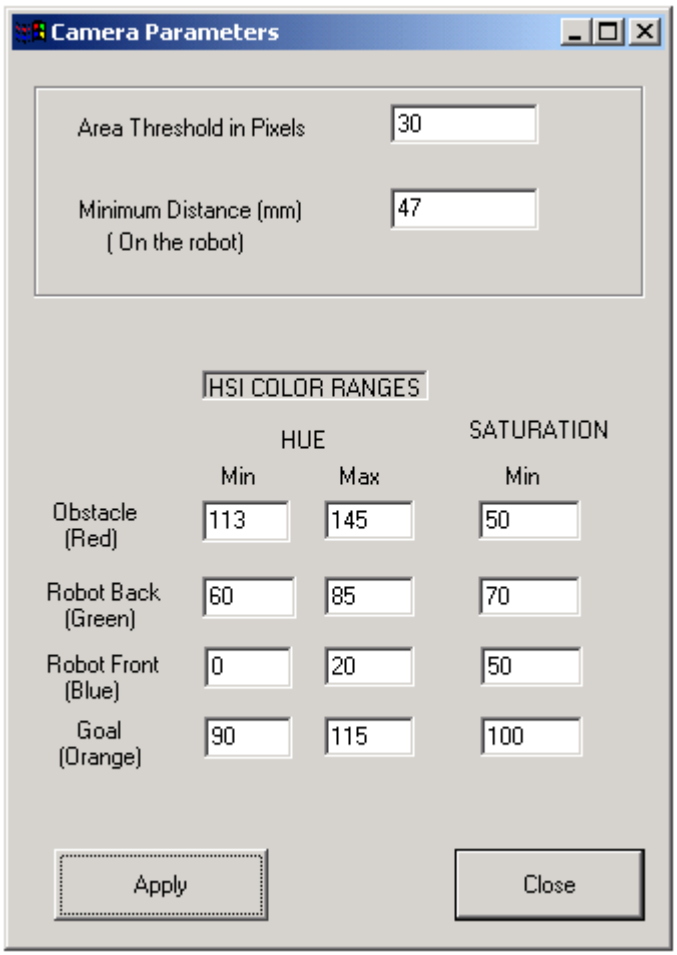

Figure 3.29 Image processing parameters dialog box.

3.6.2 Robot Side Application

The application running on the robot side is also written in C language, utilizing the C libraries (RoBIOS libraries) provided by the manufacturer of the robot. A simple development environment emulating Linux on windows using cygwin1.dll is also provided, having GNU cross compiler for C/C++ and assembly, a script (gcc68) to compile the code for robot's microprocessor (Motorola M68332), a utility program (transhex) to download the hex code to robot. The code is compiled for the microprocessor of the robot using the script gcc68 and downloaded to robot's RAM by transhex. For later use, the program can be saved to ROM before running it.

The program flow is as follows:

- Initialize the right and left wheel motors.
- Initialize the serial port, select the serial port, set the baud rate, etc.
- Initialize the right and left wheel motor encoders.
- Initialize the timer for implementing a timer based interrupt for speed control.
- Enter an infinite loop, continuously reading the data coming from the serial port. Check the control byte of the incoming data and exit if it requires.
- While reading the incoming data, implement speed control on each motor at 10 ms intervals, based on the timer interrupt, by getting the reference values from the incoming data and calculating the actual values from the encoders.
- When exit is requested, stop the robot, release the motors, encoders, and timer.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, experimental results in the form of motion history images (showing the path of the robot from start to finish), graphs, and movies (on the attached CD) obtained using the setup and algorithms discussed in the previous chapters will be presented. Various cases will be studied to demonstrate the successful aspects as well as shortcomings of the algorithms and the setup.

4.1 Experiment Environment

The robot with obstacles and goal are placed on a flat surface as shown in the images to follow. The images are taken at the beginning of the experiment and the center of the robot is marked on the image at each sampling time, thus obtaining the motion history of the robot.

The workspace of the experimental setup was not large enough to try as many configurations as possible. No special illumination is used; experiments are carried out under the laboratory illumination conditions. The image processing is done at 10 fps due to processing power limitation; therefore the system has an effective sampling time of 100 ms, which is large for typical control applications. As a result, the maximum speed of the robot has to be kept below a threshold (13 cm/s, which is in fact larger than what can be sent within a byte through wireless link) to avoid problems due to 100 ms delay.

4.2 Discrete Obstacle Modeling

In these experiments, obstacles are modeled as point objects with a radius representing their size. Complex obstacles are represented by a combination of discrete circular obstacles.

4.2.1 Experiment 1, 2

Robot is placed behind an arc shaped obstacle, while goal point is within the arc. At the startup, obstacle is within the sensor range of the robot; and since the robot is oriented to the left of arc shape, it turns to the left continues its way on the left due to the nature of the algorithm. That is, direction of motion (whether from left or right) is determined by the initial orientation of the robot with respect to the obstacle, or vice versa. The robot considers only one of the discrete obstacles, the closest one, representing the whole obstacle. The discontinuous obstacle modeling is reflected in the robot's path in the form of a low frequency oscillation around the arc.

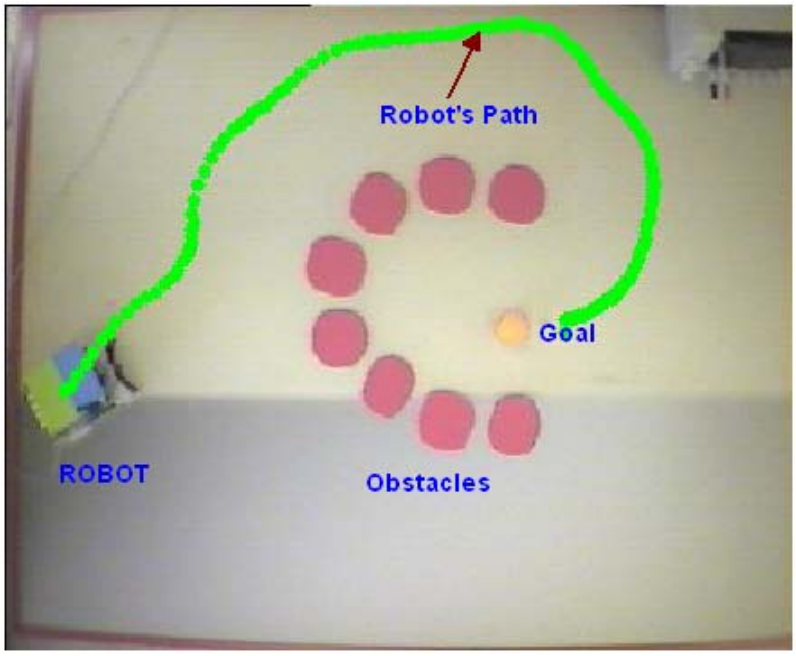

Figure 4.1 Avoiding an arc shaped obstacle modeled as discrete circles.

This time, the closest obstacle is to the left of the robot causing the robot to follow on the right of the obstacle, showing the dependence on the initial orientation.

In both experiments, the effect of behavior arbitration is clearly visible. Robot first wants to directly go to the goal; but it detects the obstacle and OA (obstacle avoidance) layer produces a new reference orientation which conflicts with what DTG (drive toward goal) layer produces. The behavior arbitration layer combines the two outputs by calculating a weight for each of them considering the geometric relations.

$$\theta^{ref} = OA^2 \cdot \theta_{OA}^{ref} + GTr^2 \cdot \theta_{DTG}^{ref} \quad \text{(Equation 2.16)}$$

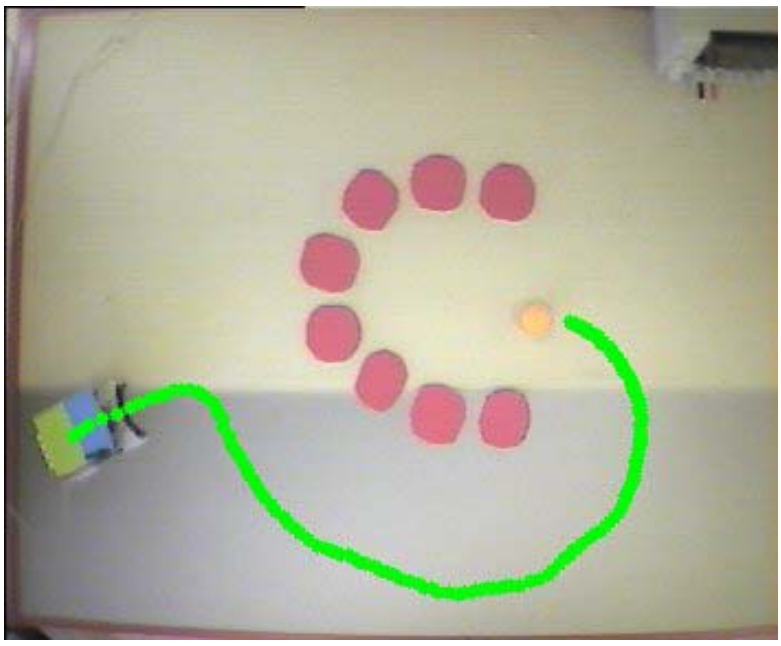

Figure 4.2 Demonstrating the effect of initial orientation.
Robot goes from right of the arc shaped obstacle.

When the robot is heading directly towards the obstacle, ($\theta_{obs} = 0^0$), the OA gains full priority ($OA = 1, \quad GTr = 0$); when the robot's way is clear of obstacles or $\theta_{obs} \geq 90^0$ then the DTG gains full priority ($OA = 0, \quad GTr = 1$).

4.2.2 Experiment 3: Passing Through Passages

One of the problems of classical potential field method is that the robot cannot pass through closely spaced obstacles even though the passage is large enough to pass. This experiment shows how the implemented algorithm performs under such a case.

Figure 4.3 shows how the robot can successfully pass through 2 obstacles. The reason is that the robot considers only one obstacle (closest) at a time contrary to the

classical potential fields method. This approach thus offers a solution, but it has also a problem. When the robot is going parallel to a wall and there is a passageway on its side, it may be repelled away depending on how close to obstacles and where the goal is.

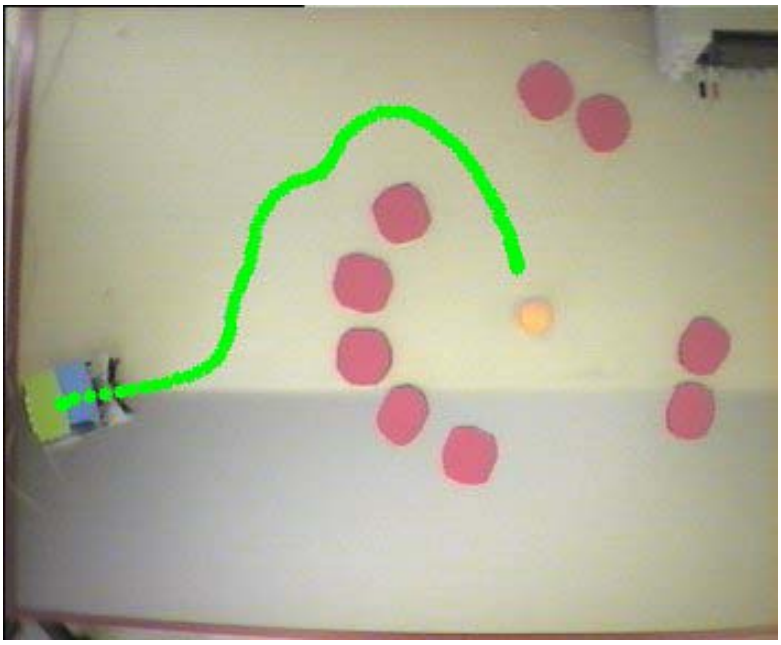

Figure 4.3 Passing through passages.

In figure 4.4, there is a potential passage to the goal between A and B. Since the closest obstacle is considered, the robot will first consider A, and then will start to detect B. Obstacle B will repel the robot to away from the passage due to its angle to the robot. Depending on the magnitudes of the references generated from OA and DTG, the robot may or may not pass through. In particular, if the robot is following the wall too closely it will not be able to pass through, since the reference orientation produced by OA will be large due to small distance between the robot and obstacle.

One important problem with discrete obstacle modeling becomes apparent when the robot and goal position changes positions in the previous obstacle configurations (e.g., robot is inside the arc, goal is on the opposite side). In such a case, due to the discreteness and considering closest obstacle, the robot tries to go through the small gaps in between the small obstacles instead of perceiving the whole obstacle as an arc and turning around it.

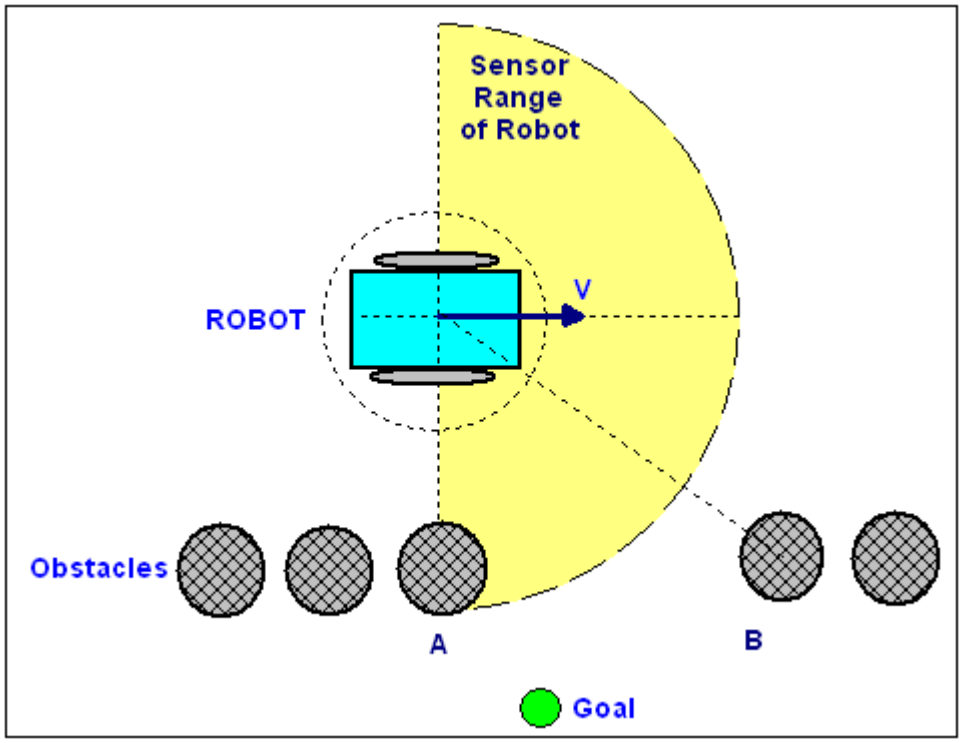

Figure 4.4 Problem in the passageways.

4.3 Modeling Continuous Obstacles

In these experiments, obstacles are not represented with discrete circular obstacles; rather, they are taken as they are. But a representative point on the continuous obstacle is selected as a heuristic solution. This enables the robot to see any obstacle, and is more realistic.

In figure 4.5, the closest point of all obstacles in the sensor range of robot is taken as the representative of all obstacles, assuming it is the most critical point for the robot. This is just a heuristic and its performance will be discussed.

The maximum speed of the robot in the following experiments is about 13 cm/sec. The speed is modified according to the reference orientation of the robot as discussed before.

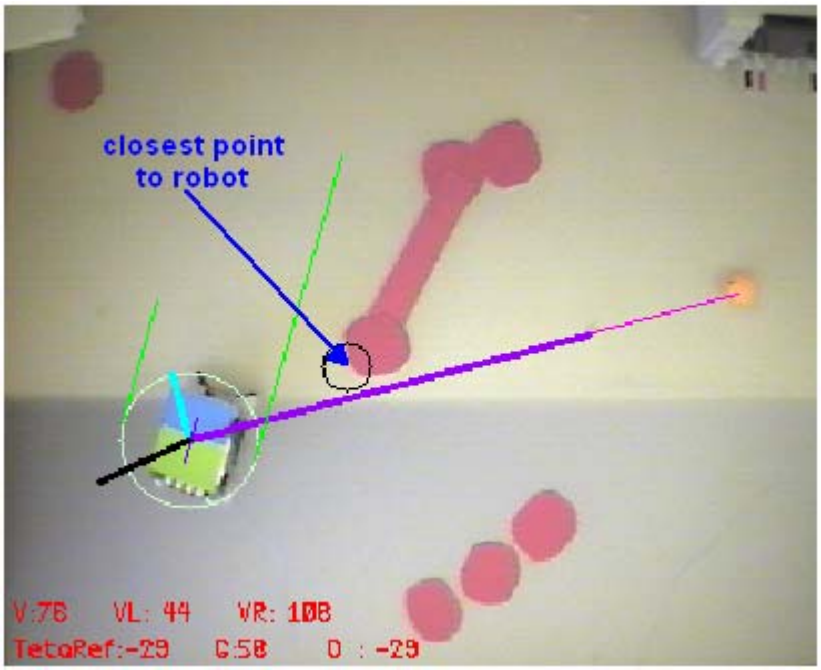

Figure 4.5 Instantaneous image illustrating choosing the closest point as the representative of obstacle.

4.3.1 Experiment 4: Circular Obstacles

Although it seems similar to the discrete obstacle case, this time instead of taking the center of each obstacle, the closest point is considered as in figure 4.5. Figure 4.6 shows how smoothly the robot reaches its goal.

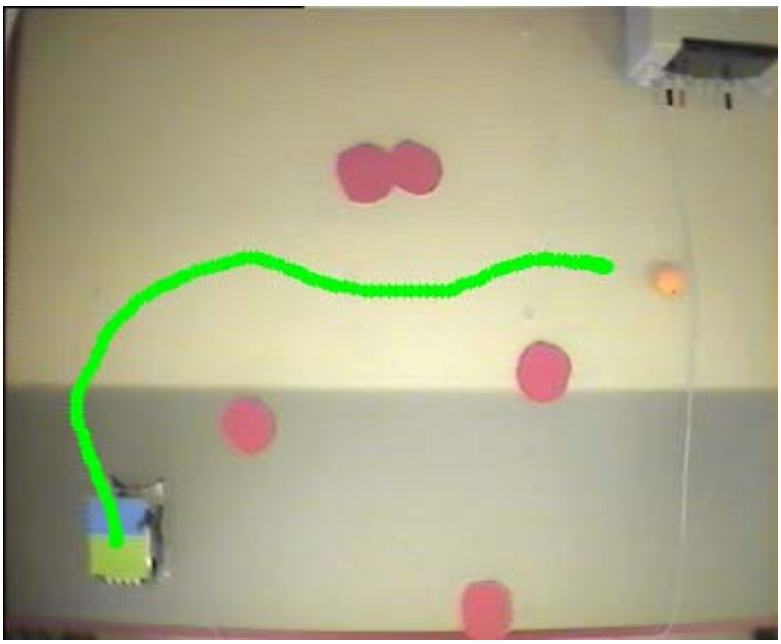

Figure 4.6 Circular obstacles.

4.3.2 Experiment 5, 6,7: Complex Shaped Obstacles

In this experiment, the robot is placed among complex shaped obstacles as shown in figure 4.7. As can be seen, the path of the robot reflects the outer boundaries of the obstacles, showing that the robot is following the boundaries. The obstacle boundaries as well as the robot's motion are quite smooth.

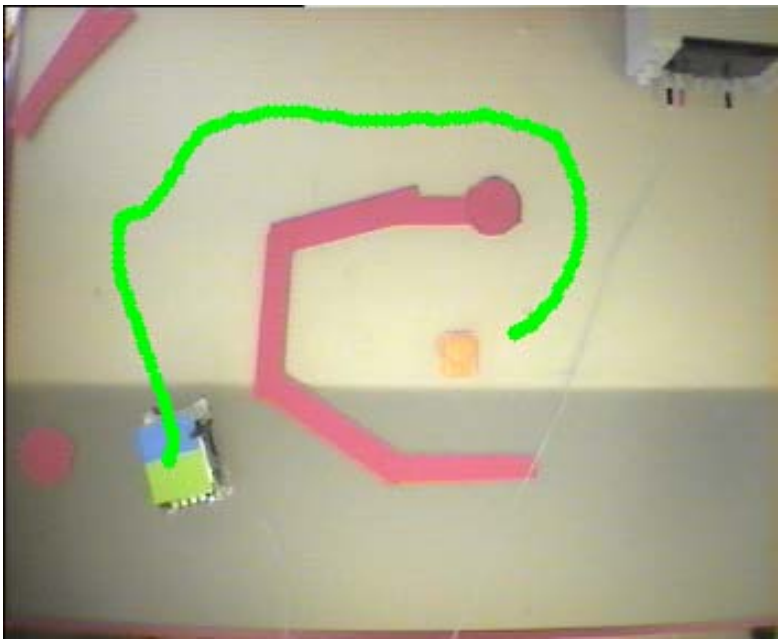

Figure 4.7 Continuous complex shaped obstacles.

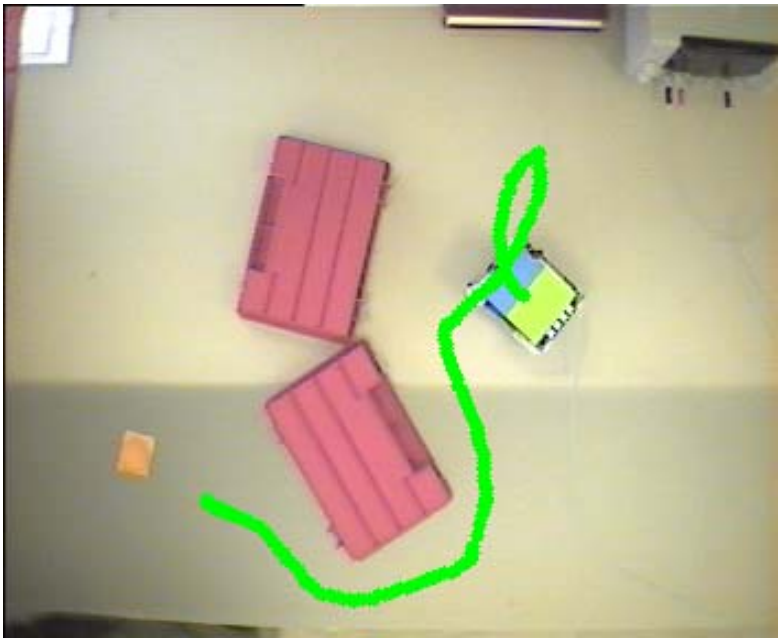

Figure 4.8 An interesting path showing the maneuvering capability of robot.

Figure 4.8 shows the superiority of continuous modeling over discrete modeling; since in such a situation, robot does not follow around the obstacle in discrete modeling case, but tries to go in between. In continuous modeling, since there is no perceived gap, there is no such problem. Robot followed the obstacle boundary and finally reached the goal.

Choosing the closest point of a continuous obstacle as the representative turned out to have one obvious problem. When the closest point is on the side of the obstacle as shown in figure 4.9, the OA layer will produce a reference orientation change close to zero, while the DTG layer will require the robot to turn right, and since the weight of DTG (*GTr*) will be significant in such situation the robot might hit the obstacle. This happens especially while turning around sharp edges, and the goal is placed behind the obstacle.

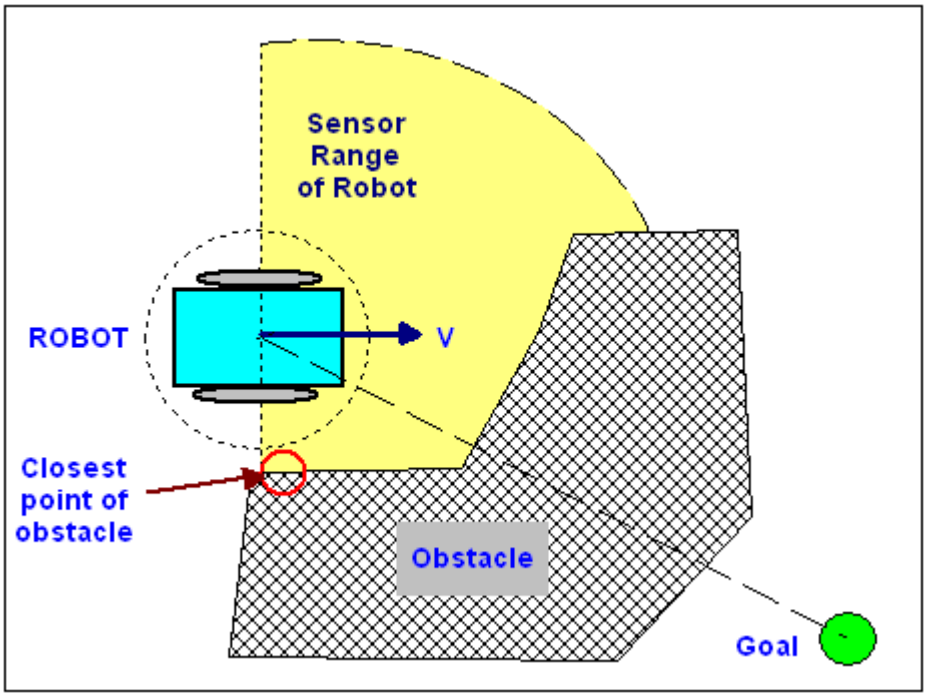

Figure 4.9 The problem of representing an obstacle as its closest point.

Figure 4.10 shown an experimental result taken for such a case, where the robot slightly touches the edge of the obstacle. If the goal were placed further up, then the result would be worse. This discussion suggests the usage of a heuristic that considers all the obstacle boundary and chooses the most critical point or an average of them as the representative in term of robot's possibility of hitting that obstacle.

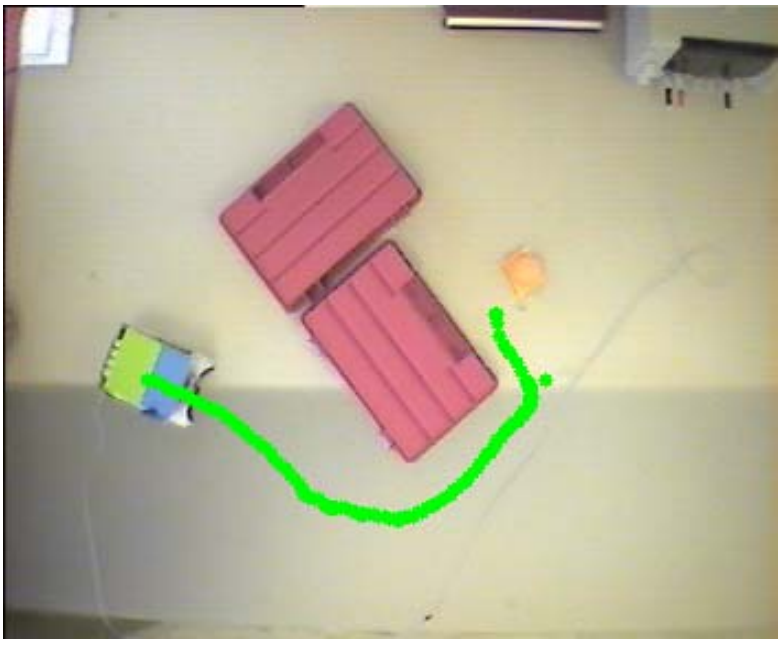

Figure 4.10 Experimental result showing the problem of representing an obstacle as its closest point.

4.3.3 Experiment 9,10: Goal is too Close to Obstacle

One of the problems of classical potential fields, and potential field based methods was that when goal is placed too close to an obstacle the robot cannot reach it due to very large repulsive force in the vicinity of the obstacle.

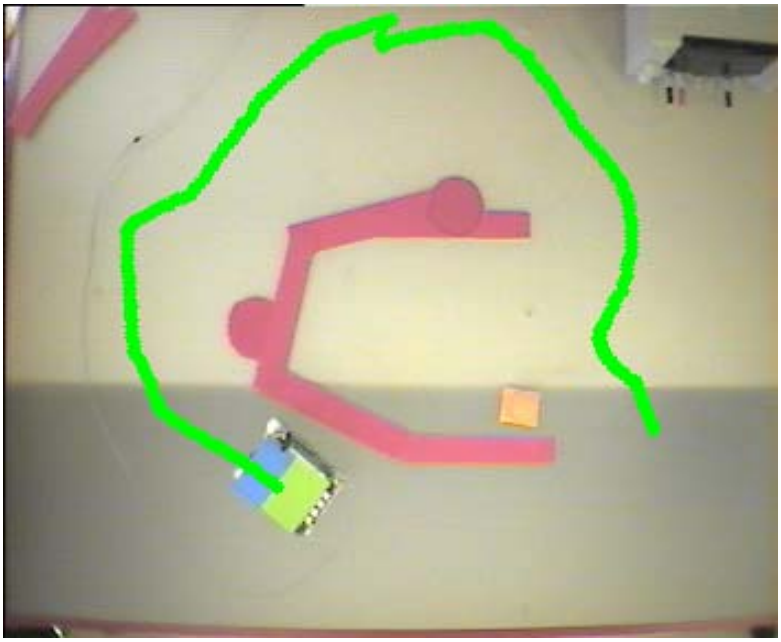

Figure 4.11 Goal is too close to an obstacle.

Figure 4.11 shows such an experiment, in which robot is repelled away when it is getting close to the goal. In the same situation, if the behavior arbitration layer considers the distances from robot to goal and obstacle while calculating the weights of OA and DTG the problem is eliminated as shown in figure 4.12.

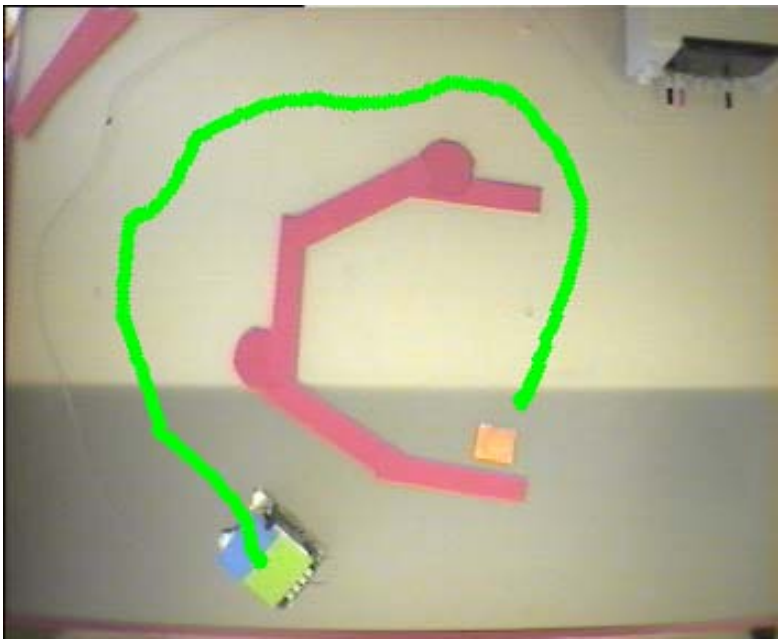

Figure 4.12 Solution to the problem when goal is too close to an obstacle.

4.3.4 Experiment 12: An Unsolvable Problem (Inside a U-Shaped Obstacle)

One of the problematic cases that is not solvable with the implemented pure reactive control method is illustrated in figure 4.13. The robot is placed inside a U-shaped obstacle, and goal is behind the obstacle. The robot follows the sides of obstacle going towards the goal until reaches the bottom of U-shape where it returns back. While going back the goal becomes situated either on its right or left (both goal and obstacle produces a reference orientation in the same direction), and it turns to that direction again reaching to the bottom of U-shape. This motion continues forever, as shown in the figure, and cannot be solved by this method. Some heuristics can be employed to get the robot out of the U-shape. For example, sub goal positions can be defined when such a situation is detected. Thus robot first gets out of the U-shape by reaching the sub goals and then it can proceed to reach its actual goal.

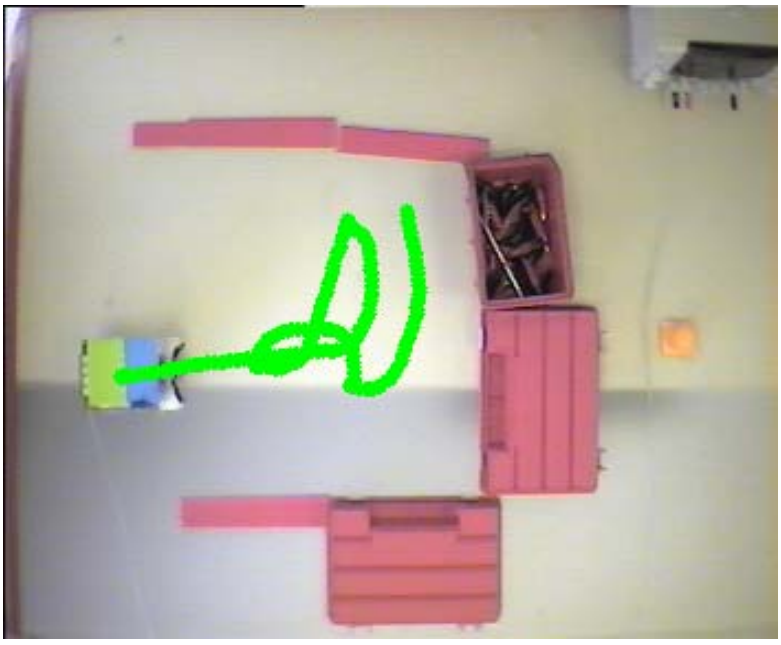

Figure 4.13 Robot is got stuck in a U-shaped obstacle.

4.3.5 Forces Acting on the Robot

To illustrate the forces acting on the robot while avoiding obstacles and reaching goal, the following simple case is selected.

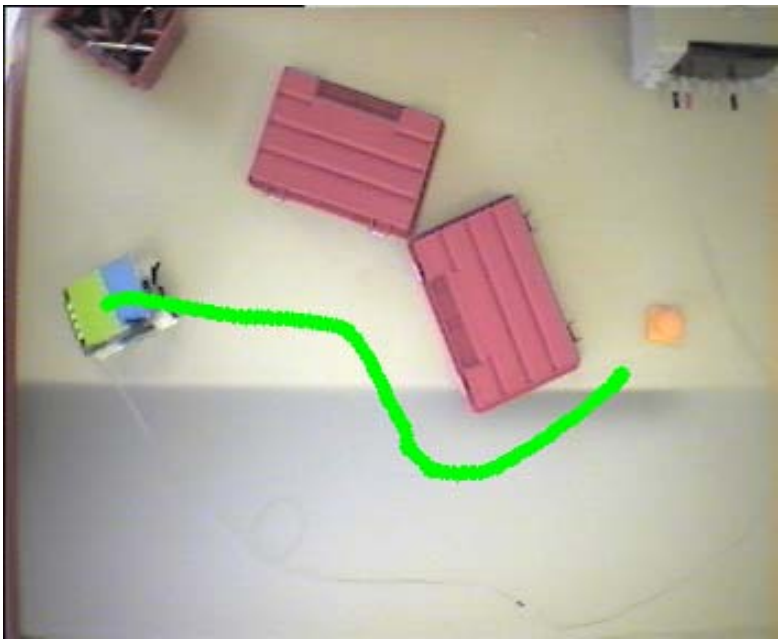

Figure 4.14 A simple case to illustrate the forces on the robot.

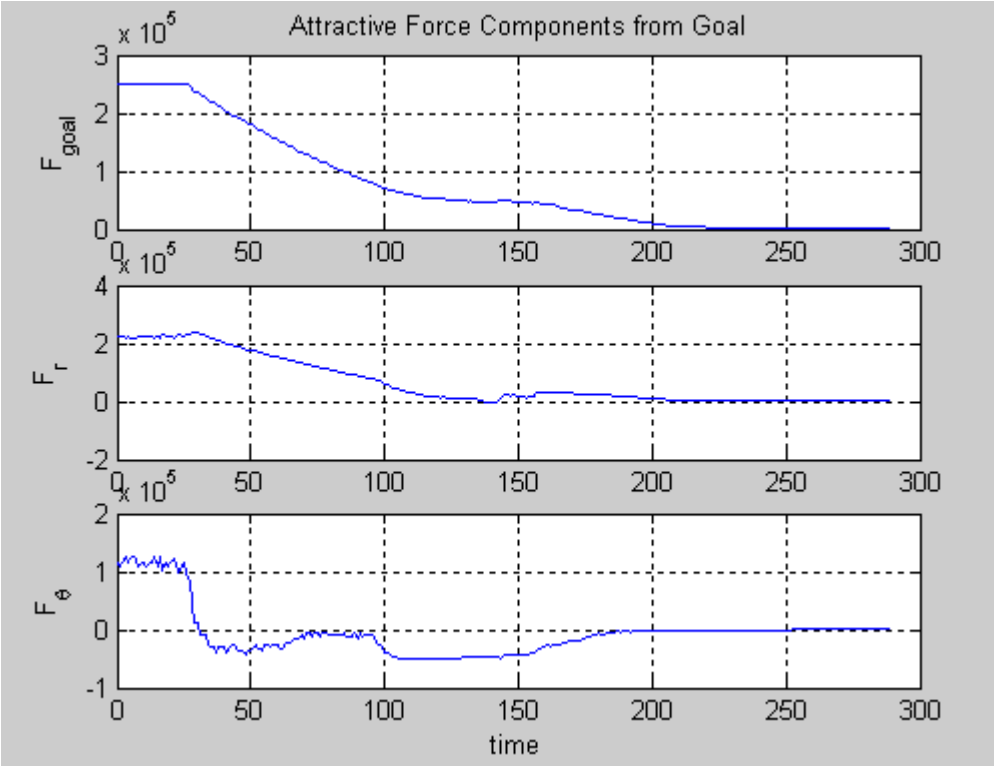

Figure 4.15 Attractive forces from the goal.

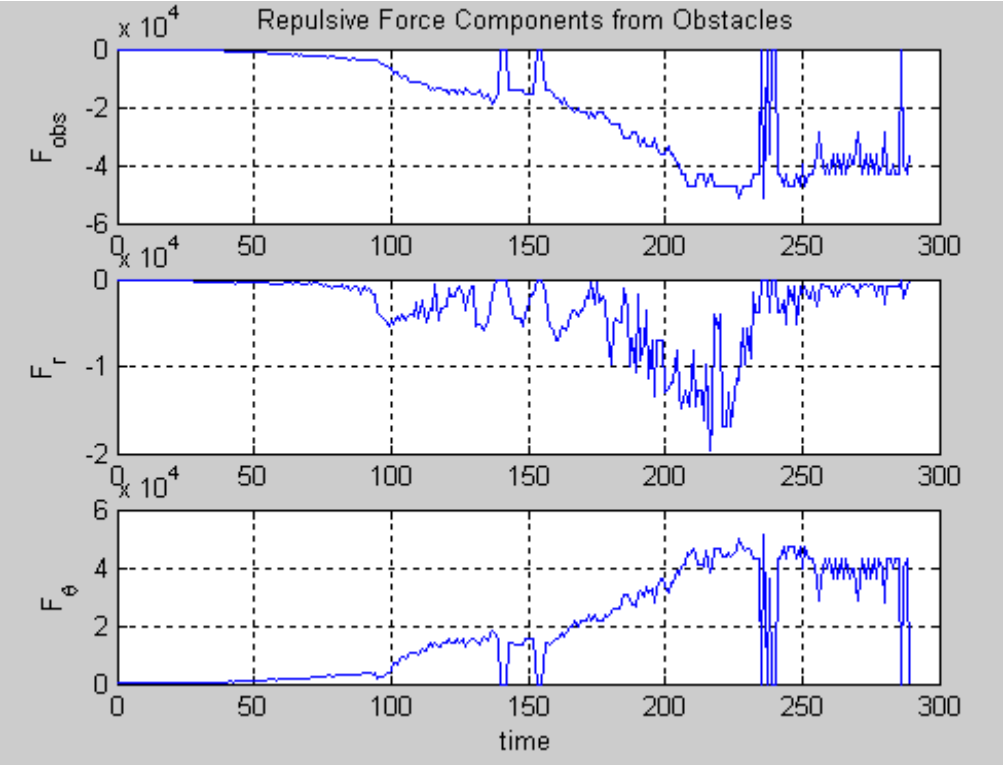

Figure 4.16 Repulsive forces from the obstacles.

In figure 4.15, the attractive force components from the goal are shown. The control algorithm is trying to keep the force component in the direction of motion, F_r, maximum, while the force in the perpendicular direction, F_θ, minimum. The opposite is valid for the obstacle force components shown in 4.16. Excessive oscillations are visible in obstacle forces, which is expected due to the nature of the algorithm. However, these oscillations are not reflected to the motion of the robot this much, since the robot's motion is not solely dependent on the obstacle forces, and also because robot has some inertia.

4.3.6 Experiment 11: An Improved Approach to Obstacle Modeling

In the previous approach, where obstacles were represented by their closest point to the robot, there were some problems in obstacle avoidance. Therefore, a new heuristic in which the sensor range of the robot was discretized in the form of pies (at 10-15 degrees) and the closest point in each pie was chosen as the representative in that pie, is also implemented to propose a solution (figures 2.16, 3.16). The result is shown in figure 4.17. This is a rather difficult obstacle configuration for the previous approaches since the robot is placed inside a sharp edge obstacle while the goal is behind it. The robot is able to successfully avoid the obstacle and reach the goal.

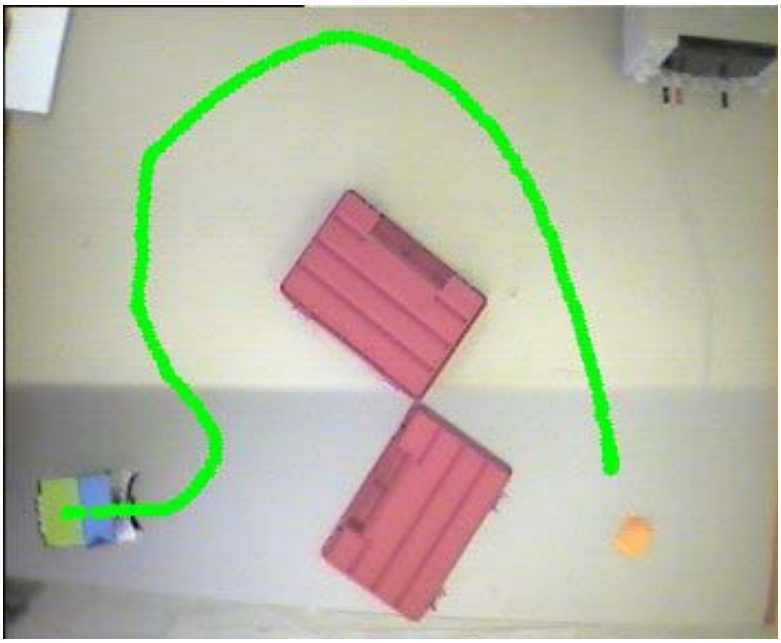

Figure 4.17 Sensor range of robot is discretized in the form of pies.

4.3.7 Dynamic Environment

One of the most important advantages of using a reactive control strategy is its ability to cope with dynamic environments; obstacles or goal may be moving.

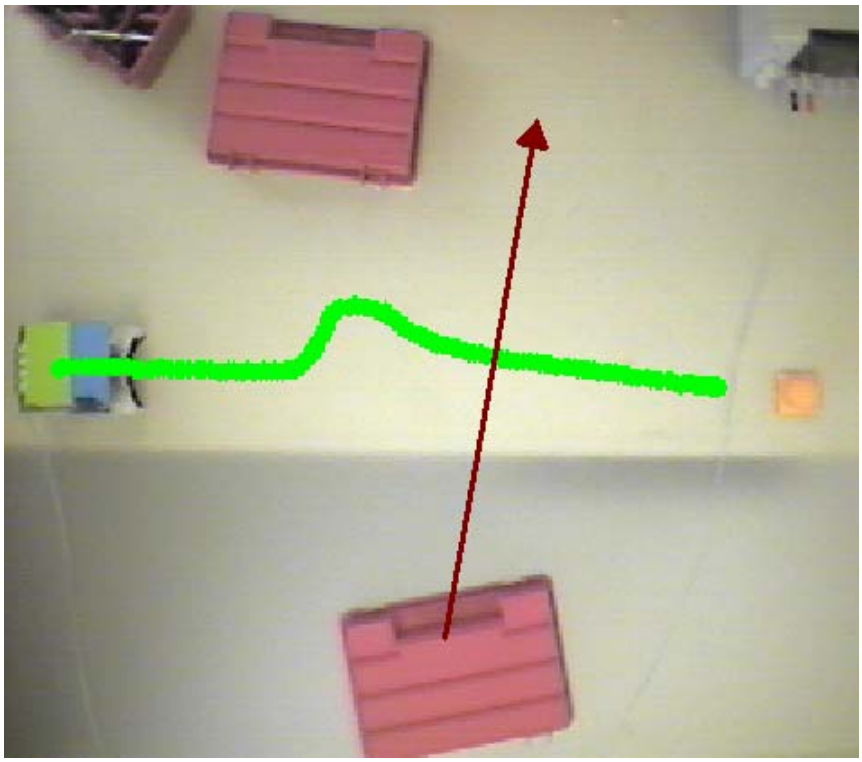

Figure 4.18 Robot's path in case of moving obstacles (1).

Figure 4.18 shows the motion history of the robot when one of the obstacles is manually moved at about the speed of the robot in the direction of the arrow. When the robot encountered the obstacle, it changed its direction, when its way got free it continued its way to reach the goal.

Another example is shown in figure 4.19, where again one of the obstacles is manually moved in the direction of the arrow shown. As in the previous case, the robot successfully avoided the moving obstacle and reached its goal.

There is of course a limit on the speed of the moving obstacles for which the algorithm can work successfully, depending on the sampling time of the system, robot's speed and inertia.

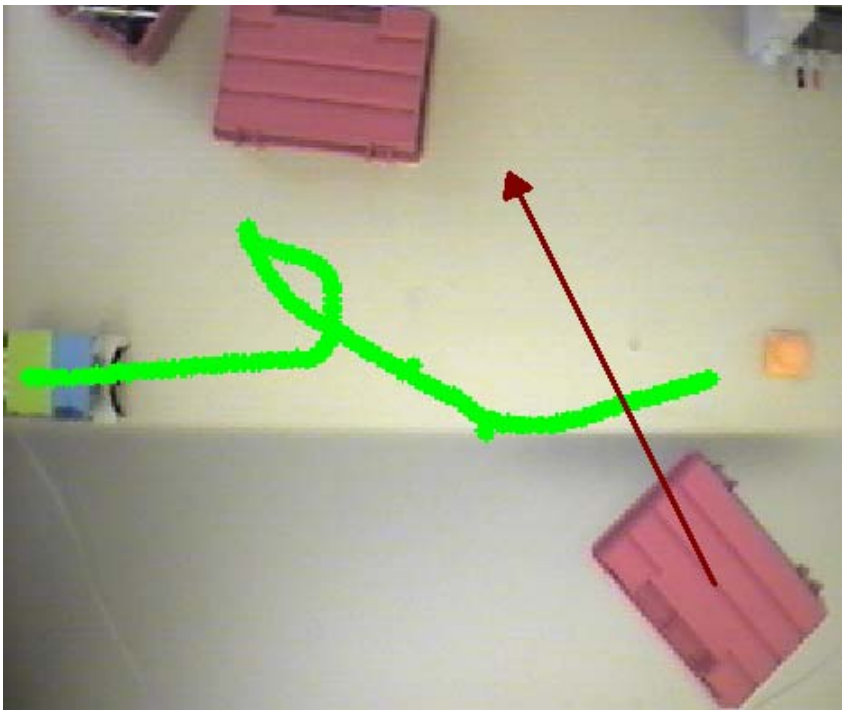

Figure 4.19 Robot's path in case of moving obstacles (2).

4.4 Evaluation of Experimental Results

Experimental results have shown 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. This is almost always the case, because 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.

The performance of the last approach, discretizing the sensor range of robot, is promising and is also well suited to sensors other than camera, such as ultrasonic distance sensors, which provide the robot with discrete model of its immediate environment. Therefore, the implementation is not specific to a system that uses a camera as sensor.

The overall performance of the experimental system and algorithm is satisfactory providing solutions to problematic cases of the similar methods in literature (e.g., passing through closely spaced obstacles). However, this method also has an inherent oscillation problem due to the discrete nature in the detection of obstacles (e.g., there is

an obstacle in the sensor range or not), or switching between different points on different obstacles. This is especially more pronounced at high speeds. All the same, the oscillation problem is not as pronounced as in the classical potential field method, and if the robot's environment is not too much crowded with complex shaped obstacles, the robot's path is smooth, and it has less oscillations as shown in the results above.

The large sampling time of the system, and noise from the camera and image processing are factors adversely affecting the performance of the system. The sampling time can be reduced by using a more powerful computer and optimizing the code for better performance, or by implementing the time consuming part of the processing (image processing) in hardware. Noise, on the other hand, is inherent in all sensors and has to be coped with within the algorithm to some extent. It can also be reduced by using a higher quality camera, since the camera used in the setup was of low quality.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this work, a potential field based mobile robot navigation algorithm is implemented on a real system. Modifications for improvements, and better performance are suggested in the light of experimental results. The results are evaluated and the shortcomings of the experimental setup and algorithms used are stated.

The performance of the system is satisfactory proposing solutions to the problematic cases of especially potential field based robot navigation algorithms and implementations, such as reducing oscillations, ability to pass through closely spaced obstacles, and navigation among complex shaped obstacles.

The setup prepared is similar to a typical small-size robot soccer setup, planned to be the groundwork for a future robot soccer team. As it is, the setup is also very useful to test the mobile robot navigation algorithms.

The only external sensor used to perceive the robot's environment is an overhead global vision camera, which has started to be extensively utilized in robotics due to its ability to provide rich information about the environment. Due to its promising features and increasing performance it will be an indispensable sensor in mobile robotics. This work, with other similar (e.g., robot soccer), demonstrates the good performance of a camera as a sensor for mobile robots.

5.2 Future Work

Stability analysis is one of the major concerns in control. Therefore, a stability analysis for predefined robot, goal and obstacle configurations should be done to see whether the implemented algorithm coupled with the experimental setup can provide a stable motion for the robot. Moreover, analysis of whether the robot can reach its goal in finite time with an appropriate controller should be done.

The robot's perception of the obstacles can be studied further to increase the performance of the robot; to cope with complex shaped obstacles better, to reduce the oscillations. The calculation of weights in the behavior arbitration layer to combine the obstacle avoidance and goal tracking might also be improved to account for problematic cases, like when the goal is placed too close to an obstacle.

The implemented method is purely reactive; therefore the robot is not guaranteed to reach its goal in all configurations. Problematic cases exist, where robot enters an infinite loop unaware of its situation. In such cases, it becomes necessary to utilize artificial intelligence to reason and get out of deadlock situations.

The system is designed to be the groundwork for robot soccer; thus, it can be easily extended by adding multiple robots, and implementing required algorithms like strategy development for a good game.

Using an overhead camera as a sensor can only be implemented in special cases like robot soccer, and factory environments. The trend is to have local onboard vision capability for mobile robots, as in the living beings in nature. Therefore, the algorithms should be implemented on mobile robots having onboard camera and processing coupled with other sensors. In this case, only the processing of input data coming from the sensors will change since the algorithm and implementation in this work are developed to be compatible with such a case.

Methods of integrating other sensors on board the robot to the algorithm should be considered, since such an augmentation might improve the performance of the robot.

Finally, it is worthwhile to consider a distributed computation approach where the image processing and explained algorithm work together with an onboard decision making algorithm on the robot which has significantly lower sampling time than vision.

REFERENCES

[1] R.C. Arkin, *Behavior Based Robotics*, MIT Pres, 1998.

[2] Thomas Braunl, *Embedded Robotics*, Springer-Verlag, Berlin, Heidelberg 2003

[3] R.C. Gonzalez, R.E.Woods, *Digital Image Processing*, Second Edition, Prentice Hall, 2002.

[4] P.A. Laplante, A.D.Stoyenko, *Real Time Imaging*, Real-Time Imaging, IEEE Press, 1996.

[5] R.C. Laplante, H. Bunke, H. Noltemeier, *Intelligent robots : sensing, modeling, and planning*, Singapore, River Edge, N.J. : World Scientific, 1997.

[6] S. Yannier, A. Onat, A. Sabanovic, "Basic Configuration for Mobile Robots", International Conference on Industrial Technology, ICIT'03, Maribor, Slovenia, 2003.

[7] H. Haddad, M. Khatib, S. Lacroix and R. Chatila, "Reactive Navigation in Outdoor Environments using Potential Fields", IEEE International Conference on Robotics and Automation (ICRA'98), Louvain (Belgique), 16-21 May 1998.

[8] J. Borenstein, Y. Koren, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", Proceedings of the IEEE International Conference on Robotics and Automation, California – April 1991.

[9] L. C. Wang, L. S. Yong, M. H. Ang Jr., "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-2002.

[10] H. Moravec, "Certainty Grids for Mobile Robots", NASA/JPL Space Telerobotics Workshop, Vol. 1, January, 1987, pp. 307-312.

[11] J. Borenstein, Y. Koren, "Real-time obstacle avoidance for fast mobile robots", IEEE Transactions on Systems, Man and Cybernetics, Volume: 19 , Issue: 5, 1989, Pages:1179 – 1187.

[12] J. Borenstein, Y. Koren, "The Vector Field Histogram - Fast Obstacle-Avoidance for Mobile Robots", IEEE Journal of Robotics and Automation, Vol. 7, No. 3., June 1991, Pages: 278-288

[13] I. Ulrich, J. Borenstein, "VFH*: local obstacle avoidance with look-ahead verification", Proceedings of IEEE International Conference on Robotics and Automation, 2000, Volume: 3, Pages: 2505 - 2511.

[14] J. Borenstein, Y. Koren, "Histogramic in-motion mapping for mobile robot obstacle avoidance", IEEE Transactions on Robotics and Automation, Volume: 7, Issue: 4, Aug. 1991, Pages: 535 – 539.

[15] J. Borenstein, Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments", Proceedings of IEEE International Conference on Robotics and Automation, 1990, Pages: 572 – 577, Vol.1.

[16] E. Rimon, D. E. Koditschek, "Exact robot navigation using artificial potential functions", IEEE Transactions on Robotics and Automation, 1992.

[17] J. Guldner, V. I. Utkin, "Sliding Mode Control for Gradient Tracking and Robot Navigation Using Artificial Potential Fields", IEEE Transactions on Robotics and Automation, Vol. 11 No. 2 April 1995.

[18] M. Veloso, P. Stone, K. Han, and S. Achim, "The CMUnited-97 Small Robot Team", Proceedings of RoboCup-97: The First Robot World Cup Soccer Games and Conferences, H. Kitano (ed.), 1998, published by Springer Verlag, Berlin.

[19] K. Sabe, M. Fukuchi, J. S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara, "Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision", Proceedings of the International Conference on Robotics and Automation, New Orleans, April 2004.

[20] S. S. Ge, Y. J. Cui, "Path Planning for Mobile Robots Using New Potential Functions", Proceedings of 3rd Asian Control Conference, July 4-7, 2000, Shanghai.

[21] O. Brock, O. Khatib. "High-Speed Navigation Using the Global Dynamic Window Approach", Proceedings of the 1999 IEEE International Conference on Robotics and Automation.

[22] M. Veloso, M. Bowling, S. Achim, K. Han, and P. Stone, "The CMUnited-98 champion small robot team", RoboCup-98: Robot Soccer World Cup II. Springer Verlag, Berlin, 1999.

[23] P. Ogren, N. E. Leonard, “A Convergent Dynamic Window Approach to Obstacle Avoidance”, IEEE Transactions on Robotics and Automation, Feb 19, 2003.

[24] J. Bruce, M. Veloso, “Fast and Accurate Vision-Based Pattern Detection and Identification”, Proceedings of the 2003 IEEE International Conference on Robotics and Automation.

[25] D. Castro, U. Nunes, A. Ruano, “Reactive Local Navigation”, Proceedings of 28th Annual Conference of the IEEE Industrial Electronics Society - IECON'02, Sevilla, 5-8 November 2002.

[26] K. Han, M. Veloso. “Reactive visual control of multiple non-holonomic robotic agents”, Proceedings of the International Conference on Robotics and Automation, Belgium, May 1998.

[27] http://www.joker-robotics.com/eyebot

[28] http://www.seas.upenn.edu/~pranav/thesis

[29] http://www.ri.cmu.edu/project_lists/index.html

[30] http://ai.eecs.umich.edu/cogarch0/atlantis

[31] www.robocup.org

[32] http://www-2.cs.cmu.edu/~robosoccer/main

[33] http://www1.cs.columbia.edu/~atanas/research/avenue

[34] www.fira.net

REFERENCES

[1] R.C. Arkin, *Behavior Based Robotics*, MIT Pres, 1998.

[2] Thomas Braunl, *Embedded Robotics*, Springer-Verlag, Berlin, Heidelberg 2003

[3] R.C. Gonzalez, R.E.Woods, *Digital Image Processing*, Second Edition, Prentice Hall, 2002.

[4] P.A. Laplante, A.D.Stoyenko, *Real Time Imaging*, Real-Time Imaging, IEEE Press, 1996.

[5] R.C. Laplante, H. Bunke, H. Noltemeier, *Intelligent robots : sensing, modeling, and planning*, Singapore, River Edge, N.J. : World Scientific, 1997.

[6] S. Yannier, A. Onat, A. Sabanovic, "Basic Configuration for Mobile Robots", International Conference on Industrial Technology, ICIT'03, Maribor, Slovenia, 2003.

[7] H. Haddad, M. Khatib, S. Lacroix and R. Chatila, "Reactive Navigation in Outdoor Environments using Potential Fields", IEEE International Conference on Robotics and Automation (ICRA'98), Louvain (Belgique), 16-21 May 1998.

[8] J. Borenstein, Y. Koren, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", Proceedings of the IEEE International Conference on Robotics and Automation, California – April 1991.

[9] L. C. Wang, L. S. Yong, M. H. Ang Jr., "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-2002.

[10] H. Moravec, "Certainty Grids for Mobile Robots", NASA/JPL Space Telerobotics Workshop, Vol. 1, January, 1987, pp. 307-312.

[11] J. Borenstein, Y. Koren, "Real-time obstacle avoidance for fast mobile robots", IEEE Transactions on Systems, Man and Cybernetics, Volume: 19 , Issue: 5, 1989, Pages:1179 – 1187.

[12] J. Borenstein, Y. Koren, "The Vector Field Histogram - Fast Obstacle-Avoidance for Mobile Robots", IEEE Journal of Robotics and Automation, Vol. 7, No. 3., June 1991, Pages: 278-288

[13] I. Ulrich, J. Borenstein, "VFH*: local obstacle avoidance with look-ahead verification", Proceedings of IEEE International Conference on Robotics and Automation, 2000, Volume: 3, Pages: 2505 - 2511.

[14] J. Borenstein, Y. Koren, "Histogramic in-motion mapping for mobile robot obstacle avoidance", IEEE Transactions on Robotics and Automation, Volume: 7, Issue: 4, Aug. 1991, Pages: 535 – 539.

[15] J. Borenstein, Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments", Proceedings of IEEE International Conference on Robotics and Automation, 1990, Pages: 572 – 577, Vol.1.

[16] E. Rimon, D. E. Koditschek, "Exact robot navigation using artificial potential functions", IEEE Transactions on Robotics and Automation, 1992.

[17] J. Guldner, V. I. Utkin, "Sliding Mode Control for Gradient Tracking and Robot Navigation Using Artificial Potential Fields", IEEE Transactions on Robotics and Automation, Vol. 11 No. 2 April 1995.

[18] M. Veloso, P. Stone, K. Han, and S. Achim, "The CMUnited-97 Small Robot Team", Proceedings of RoboCup-97: The First Robot World Cup Soccer Games and Conferences, H. Kitano (ed.), 1998, published by Springer Verlag, Berlin.

[19] K. Sabe, M. Fukuchi, J. S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara, "Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision", Proceedings of the International Conference on Robotics and Automation, New Orleans, April 2004.

[20] S. S. Ge, Y. J. Cui, "Path Planning for Mobile Robots Using New Potential Functions", Proceedings of 3rd Asian Control Conference, July 4-7, 2000, Shanghai.

[21] O. Brock, O. Khatib. "High-Speed Navigation Using the Global Dynamic Window Approach", Proceedings of the 1999 IEEE International Conference on Robotics and Automation.

[22] M. Veloso, M. Bowling, S. Achim, K. Han, and P. Stone, "The CMUnited-98 champion small robot team", RoboCup-98: Robot Soccer World Cup II. Springer Verlag, Berlin, 1999.

[23] P. Ogren, N. E. Leonard, “A Convergent Dynamic Window Approach to Obstacle Avoidance”, IEEE Transactions on Robotics and Automation, Feb 19, 2003.

[24] J. Bruce, M. Veloso, “Fast and Accurate Vision-Based Pattern Detection and Identification”, Proceedings of the 2003 IEEE International Conference on Robotics and Automation.

[25] D. Castro, U. Nunes, A. Ruano, “Reactive Local Navigation”, Proceedings of 28th Annual Conference of the IEEE Industrial Electronics Society - IECON'02, Sevilla, 5-8 November 2002.

[26] K. Han, M. Veloso. “Reactive visual control of multiple non-holonomic robotic agents”, Proceedings of the International Conference on Robotics and Automation, Belgium, May 1998.

[27] http://www.joker-robotics.com/eyebot

[28] http://www.seas.upenn.edu/~pranav/thesis

[29] http://www.ri.cmu.edu/project_lists/index.html

[30] http://ai.eecs.umich.edu/cogarch0/atlantis

[31] www.robocup.org

[32] http://www-2.cs.cmu.edu/~robosoccer/main

[33] http://www1.cs.columbia.edu/~atanas/research/avenue

[34] www.fira.net