REALIZATION OF REACTIVE CONTROL FOR MULTI PURPOSE MOBILE AGENTS

By

SELİM YANNİER

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 2002

Realization of Reactive Control for Multi Purpose

Mobile Agents

APPROVED BY:

Assistant Prof. Dr. AHMET ONAT (Dissertation Advisor)	....
Prof. Dr. ASIF ŞABANOVİÇ (Dissertation Co-Advisor)	....
Assistant Prof. Dr. KEMALETTİN ERBATUR	....
Assistant Prof. Dr. ERHAN BUDAK	....
Associated Prof. Dr. HAKAN TEMELTAŞ	....

DATE OF APPROVAL: ....

ABSTRACT

The development of control method for an autonomous mobile robot that can be part of a multiagent system is the subject of this thesis.

Mobile robots are built different purposes; for that reason, physical size, shape and mechanics, such as driving mechanism, of the robots would be very different. They are most of the time required to realize more than one goal at a time. Hence, as a minimum, required control must work in real-time and be able to respond to sudden changes that may occur in the environment.

There is extensive research carried out, about autonomous mobile and stationary robots, ranging from path planning and obstacle avoidance to multiple autonomous mobile robots.

Proposed approach, for mobile robot control, is a layered structure and supports multi sensor perception. Each control layer uses only necessary sensor functions for its own process. Moreover, layers of control do not evaluate the signal coming directly from the sensors but get information of the environment from a preprocessor. This way the addition of new sensors does only affect the mentioned preprocessor that will convert the signals to the necessary form asked by the layers. In this work, potential field method is implemented for obstacle avoidance. Unlike other solutions in the same framework, in this work assumed repulsive forces of the obstacles and attractive force of the goal point are treated separately. Then obstacles avoidance and goal tracking are fused in such a way that major drawbacks of the potential field method are overcome.

Proposed control is tested on simulations where different scenarios are studied. The simulation results confirmed the high performance of the method. The proposed control is a potential alternative for mobile robots control operating in dynamic environments and as an agent in a multiagent system.

ÖZET

Çok parçalı sistemlerde görev alabilecek mobil robotlar için bir denetim metodunun geliştirilmesi bu tezin ana konusudur.

Mobil robotlar çok değişik amaçlar için üretilirler. Bunun sonucu olarak şekilleri, boyutlar ve sürücü mekanizmaları oldukça farklı olabilir. Genellikle bu robotlardan aynı anda birden fazla amaca hizmet etmeleri istenir. Böylece, bu tip bir kontrol sisteminin en az gereksinimleri gerçek zamanda çalışabilmesi ve ortamdaki ani değişikliklere tepki verebilmesi olarak belirlenebilir.

Otonom mobil robotlar ve sabit robotlarla ilgili çok sayıda araştırma yapılmaktadır. Bunlar güzergah belirleme ve engellerden sakınmadan, çok parçalı sistemlere kadar uzanmaktadır.

Burada önerilen mobil robot kontrol metodu katmanlardan oluşur ve birden fazla algılayıcı kullanabilir. Her bir katman sadece ihtiyaç duyduğu bilgiyi bir ön işleyiciden alır. Böylece algılayıcı ekleme durumunda sadece ön işleyicide güncelleme yapılacak, sistemin kalanı etkilenmeyecektir. Bu çalışmada potansiyel alan metodu ile engellerden sakınma sağlanmıştır. Ancak diğer çalışmaların aksine, engellerden dolayı oluştuğu varsayılan itici kuvvetler ve ulaşılacak noktanın çekici kuvveti ayrı ayrı işlenmişlerdir. Engellerden sakınma ve hedefe yönelme daha sonra birleştirilmiştir. Bu şekilde, kullanılan metodun en önemli dezavantajları yok edilmiştir.

Önerilen kontrol benzeşimlerle denenmiş ve çeşitli senaryolar üzerinde çalışılmıştır. Benzeşim sonuçları metodun yüksek performansını onaylamaktadır. Önerilen kontrol çok parçalı sistemlerde görev alabilecek ve karmaşık ortamlarda çalışabilecek mobil robotlar için potansiyel bir seçenektir.

To Habib & V. Vivet Yannier, “*full-time parents”* since April 1978,
and Alper Yannier, “*full-time brother”* since August 1982...

ACKNOWLEDGEMENTS

I wish to express my thankfulness to a number of people who became involved with this thesis, one way or another.

My thesis co-advisor, Asıf Şabanoviç, a true gentleman whose suggestions and encouragement led me find my way throughout this research, has always been enthusiastic to answer my questions no matter how irrelevant they are. Thank you for being available whenever I needed you.

Ahmet Onat, who was also my instructor in the past semesters, was so polite in providing me some of the tools I needed for research and documentation.

In addition, thanks to all mechatronics people for their friendship. Especially to graduate students who have spent numerous nights, together with me, in the laboratory.

Finally, I believe I owe many thanks to my parents Habib and Vivet, as well as my brother Alper, who were supportive of my academic enthusiasms.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF SYMBOLS

(x_w, y_w)	World coordinate frame
ω_L	Angular velocity of the left wheel of the robot
ω_R	Angular velocity of the right wheel of the robot
v_L	Linear velocity of the left wheel of the robot
v_R	Linear velocity of the right wheel of the robot
v_{rb}, v	Velocity of the robot
ϕ_{rb}, ϕ	Orientation of the robot
ω_{rb}, ω	Velocity of the orientation of the robot
(x, y)	Position of the robot in world coordinate frame
L	Length between the wheels of the robot
R	Radius of the wheels of the robot
$d_{\max}$, $d_{\min}$	Maximum and minimum distance that can be detected by the sensors
d_{obs}	Distance to the obstacle as detected by sensor
$\vec{F}_{OB}$	Imaginary repulsive force caused by the obstacle
$\left\Vert\vec{F}_{OB}\right\Vert$	Magnitude of the imaginary repulsive force caused by the obstacle
(f_{obs}, θ_{obs})	Components of the imaginary repulsive force (magnitude and direction in world coordinate frame respectively)
$\vec{G}$	Imaginary attractive force caused by the goal point
$\left\Vert\vec{G}\right\Vert$	Magnitude of the imaginary attractive force caused by the goal point
(f_G, θ_G)	Components of the imaginary attractive force (magnitude and direction in world coordinate frame respectively)
α, β	Dummy variable used to define an angle
d_{obs}	Minimum distance between the center of mass of the robot and the obstacle

$v_{\min}$, $v_{\max}$	Minimum and maximum velocities
e_v, e_ϕ	Error in velocity and direction
x_{ref}, y_{ref}	Reference coordinates in world coordinate frame
e_x, e_y	Position errors in x and y directions
e_r, e_ϕ	Projection of the position errors onto the velocity and its perpendicular direction
r_{ref}^{corr}, ϕ_{ref}^{corr}	Corrected reference values of position and steering direction (in dimension of distance)
e_r^{corr}, e_ϕ^{corr}	Position and direction control errors
u_1, u_2	Controls
σ_v, σ_ϕ	Sliding manifolds
γ	Candidate Lyapunov function
$\omega_{\min}^{ref}$, $\omega_{\max}^{ref}$	Minimum and maximum possible reference wheel velocities that can be accepted by servo motor controllers
k	k^{th} time interval
dt	Discrete time interval
F_r, F_ϕ	Force along the heading direction and the direction perpendicular to it
F_r^{ref}, F_ϕ^{ref}	Reference forces along the heading direction and the direction perpendicular to it
u_r, u_ϕ	Controls for velocity and heading direction
χ	Proportionality constant
v^{ref}	Reference velocity
a^{ref}	Reference acceleration
a_0	Scalar gain that defines the maximum deceleration amount

TABLE OF ABBREVIATIONS

AFSM	Augmented Finite State Machine
CMU	Carnegie Melon University
DTG	Drive Toward Goal
GUI	Graphical User Interface
MEBCA	Multiagent Event Based Control Architecture
SMC	Sliding Mode Control
VFF	Virtual Force Field
VFH	Vector Field Histogram
VSS	Variable Structure Systems

1. INTRODUCTION

1.1 Motivation

It is always possible to carry out a complex task by calling upon an assembly of specialists possessing necessary skills. Either to increase performance while minimizing cost or to make a task realizable, many robots, mobile or stationary, can be programmed to work together. As long as the system is well controlled, it is guarantied that the efficiency of the system will be higher than the efficiency of each single agent.

Control and communication methods for multiple robot systems have been investigated by various researchers. In the past, rather then using autonomous agents that may decide their actions alone, central controller units are widely used. Those units, making all necessary operations, were commanding over all the elements of the system. However, those systems are fixed systems that are just capable of replicating previously programmed motions and acts, far away of offering flexibility to the users.

On the other hand, there is extensive research carried out about autonomous mobile and stationary robots. Solutions to many problems including path planning and obstacle avoidance were proposed and tested. However, most of the research on autonomous mobile robots was based on a single robot, which is actually a stand-alone agent.

In recent years there has been a growing fascination with multiple autonomous mobile robot systems not only due to their applicability to various tasks such as space missions, operations in hazardous environments, and military operations but also due to many advantages over single robot systems. Those systems, generally called "Decentralized Multi-Agent Systems", are missing a centralized controller and are more flexible, more powerful and more robust at the same time.

Of course, interest in decentralized systems, or simply in decentralization, is not entirely new. More than two hundred years ago, Adam Smith made a forceful argument against centralized government control of the economy. In his book “The Wealth of Nations”, published in 1776, Smith advocated decentralized markets as a more orderly and more efficient alternative to centralized control.

Nearly a century after Adam Smith, Charles Darwin in “Origin of Species”, offered the first serious alternative to the centralized approach, “God designed the complexity of creatures”. Darwin's challenge was to explain the organized complexity of living systems. Even the simplest creatures of the living world are more complex than the most complex machines of the technological world. Just as Adam Smith asserted that centralized government control is not needed to create order in the economy, Darwin asserted that, a centralized designer of life is not needed to create order in the living world. Instead, order and complexity arise from the decentralized processes of variation and selection. He’s theory called “Theory of Natural Selection” is one of the most accepted theories in his field.

Currently, ideas about decentralization and self-organization are spreading through the culture in almost all domains of life. Increasingly, people are choosing decentralized models for the organizations and technologies that they construct in the world, and for the theories that they construct about the world. Even governments are moving toward increasingly decentralized structure of administration.

On December 7, 1991, Russian president Boris Yeltsin met with the leaders of Ukraine and Belarus in a forest dacha outside the city of Brest. After two days of secret meetings, the leaders issued a declaration "The Union of Soviet Socialist Republics, as a subject of international law and a geopolitical reality, is ceasing its existence". With that announcement, Yeltsin and his colleagues sounded the final death knell for a centralized power structure that had ruled for nearly 75 years. In its place, the leaders established a coalition of independent republics, and they promised a radical decentralization of economic and political institutions.

The next day, halfway around the world, another powerful institution announced its own decentralization plans. IBM chairman John Akers publicly announced a sweeping reorganization of the computer giant, dividing the company into more than a dozen semiautonomous business units, each with its own financial authority and its own board of directors. The goal was to make IBM more flexible and responsive to the needs of rapidly changing markets. As Business Week magazine put it, "The reorganization could amount to no less than a revolution in the way IBM does business".

Thus, within days, two of the world's most powerful institutions announced radical transformations, leaving centralized hierarchies in favor of more decentralized structures. Of course, the reorganizations of the Soviet Union and IBM were not directly related to one another. Nevertheless, the two reorganizations are both part of a broad trend that is sweeping through our culture. Throughout the world, there is an unprecedented shift toward decentralization.

This shift cannot be seen strange, since decentralized systems offer many advantages. Most important advantage is the simplicity to control the whole system. Each part of the system has its own, specific problems, which must be solved. Therefore, what is necessary to realize is to provide required inputs data or knowledge to that part of the system, to solve problems. Moreover, each part of the system will have less problem than the total of the system, resulting less conflict that must be solved. Another income of the multiagent system is the robustness; even if one or more parts of the system fail, the rest of the system will run unaffected.

Beside advantages, some drawbacks also exist. In such systems, even though the stability of each part is guaranteed, a global stability may not be achieved. This is simply due to possible conflicts of the stability requirements of each part of the system. Another disadvantage is the non-deterministic nature of task accomplishment. The system is forced to realize given tasks, however, it may not be always satisfactory.

1.2 Literature Survey: Mobile Robot Control

Robots that have actuators helping it to change place in the environment are classified as mobile robots. Can a platform with two motors and a battery as power source be called a mobile robot? From different point of views, the answer may change a lot. However, if certain amount of intelligence is required to exist on the robot then obviously some other hardware must be installed. This new component is supposed to produce outputs in such a manner that the robot shows an intelligent behavior. In other words, movements of the robot are not random but have a reason.

First machine considered showing intelligent behavior is W. Gray Walter’s Tortoise called Machina Speculatrix (1950). It is very simple machine consisting of one photocell, one bump sensor and two motors, one driving and one steering (original schematic shown in Figure 1.1 and the hardware in Figure 1.2 below).

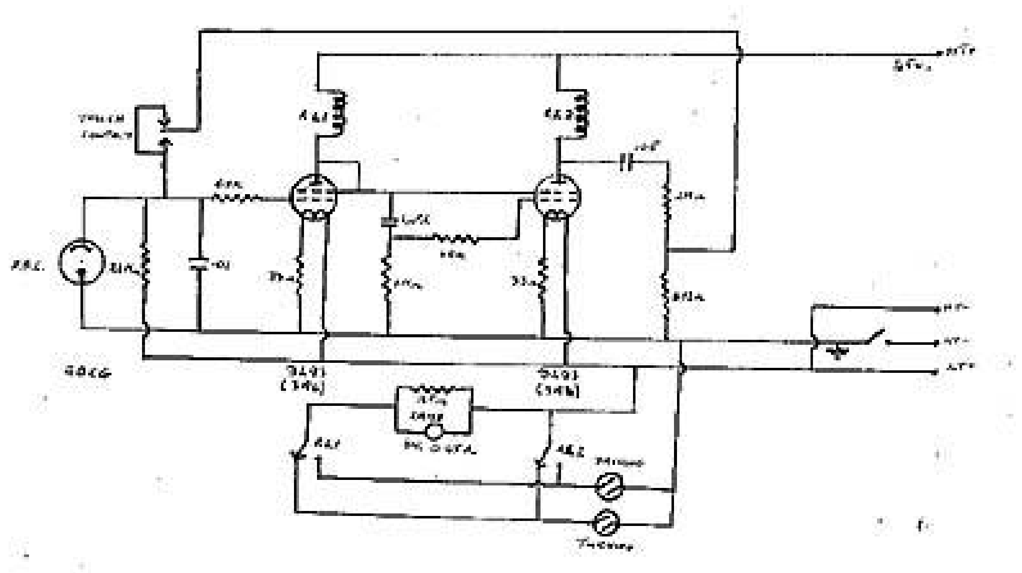


Figure 1.1: Original drawing of the electronic control (with vacuum tubes) of one of W. Grey Walter's tortoises. It's an extremely simple circuit, but that was able to generate complex and unpredictable behavior as its sensors interacted with the environment.

Behavior shown by Walter's reactive machine is seeking light, head to weak light, back from bright light and recharge battery. Walter studied many cases with its tortoises, but especially "dance of turtles" is very interesting. He placed two turtles with a light source and light sensor each. Then a strange movement is produced that reminds the mating behavior of animals.

Grey Walter's turtles exhibited simple behaviors and were based on the theories about simple reflexes and neural function that were known at the time. Several decades later and informed by a great deal more known about the brain, the neurobiologist, Valentino Braitenberg, tried to summarize out what he had learned in a set of simple models that he called vehicles.

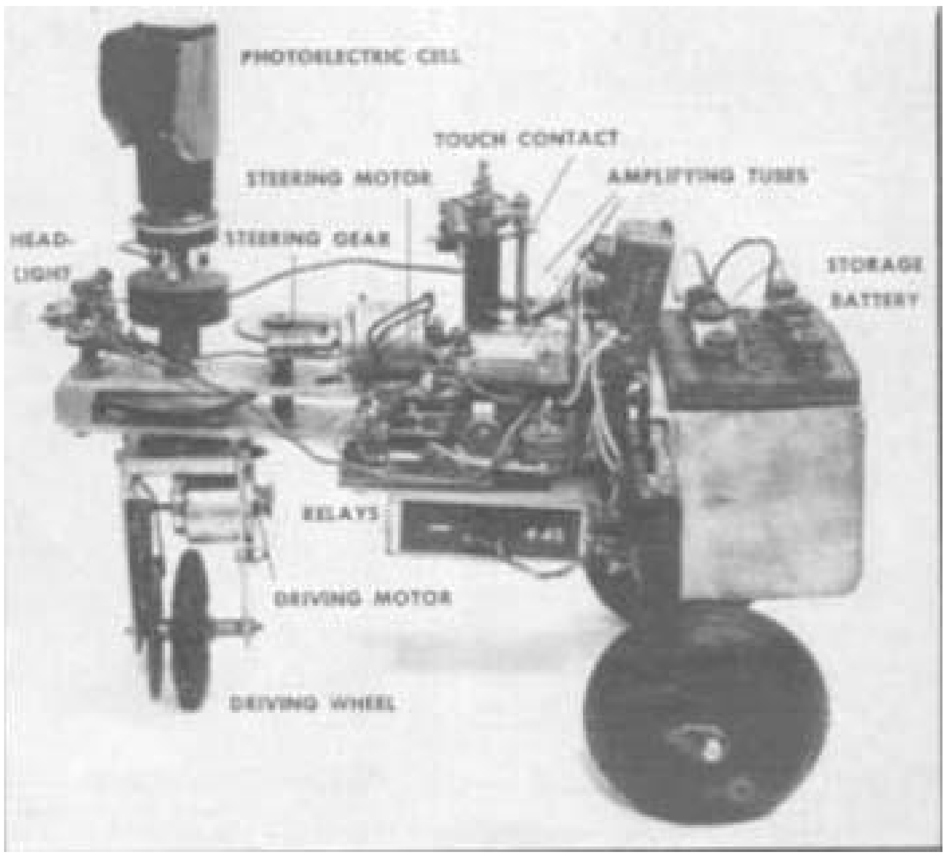


Figure 1.2: Elsie's original photo, with its components identified. The robot nicknamed "tortoise", due to its appearance had three wheels, each one with an independent motor. Power was supplied by a DC battery (right side, at the back of the system).

As generally called, “Braitenberg vehicles” consist of the simplest sort of wheeled robots in which sensors that are sensitive to different stimuli are coupled directly to motors that drive the wheels. In these models, we assume that the sensor generates a signal that is proportional to the stimulus. In the simplest vehicle (see Vehicle 1 in Figure 1.3), we imagine a sensor sensitive to light and assume that the more light that falls on face the sensor the faster the motor turns. The wire running from the sensor to the motor is labeled positive $(+)$ indicating that the sensor stimulates the motor to turn.

With the development in microelectronics and silicon technology first more complicated but compact analog circuits, then microcontrollers took their place in mobile robot control. Those controllers allowed researchers to use their robots for more advanced tasks relative to what is done until that time. Most of the researches used to be about moving to mobile robot toward a desired location while avoiding collision.

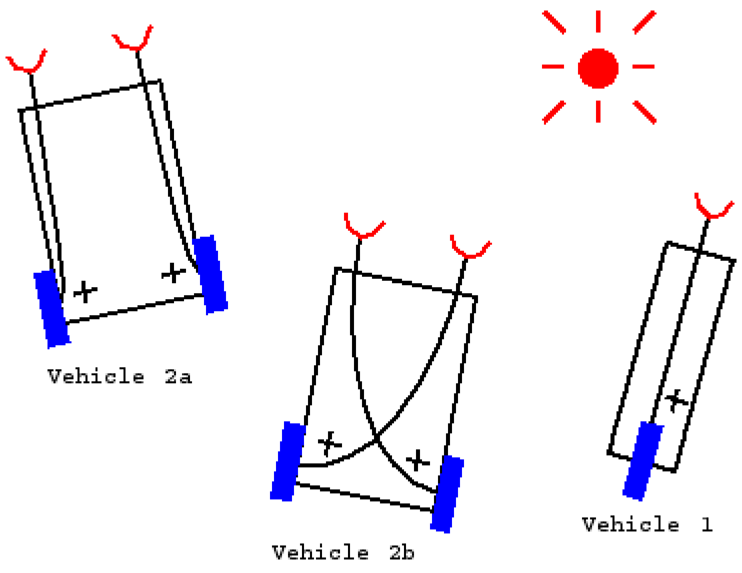


Figure 1.3: Simple Braitenberg vehicle examples. We can also imagine connections, which are inhibitory (negative labeled) and motor inputs that can sum the (perhaps weighted) contributions from different wires and sensors.

Often a robot will have multiple goals, some conflicting, that must be reached simultaneously. For a mobile robot, it may be avoiding obstacles while moving toward target point. Although the relative importance will be context dependent, each task must have a relative importance, or priority.

A robot will most likely have more than one sensor. Furthermore, those sensors are not necessarily single type; a mobile robot may have a TV camera, many infrared beacons and ultrasound range finders for obstacle avoidance, encoders on steering mechanism for speed control, and much different type of sensors for other purposes. Obviously not all sensors are used for each single task, as example encoders do not have a direct relationship with obstacle avoidance. A robot control system must support the use of various types of sensors.

A robot ought to be robust. It should be able to adapt in case of some sensors fails. When changes in the environment are drastic, it should still be able to function. Moreover, as more functionality is added, the rest of the system must not be changed radically.

Within the framework of mobile robotics, it is often difficult to establish a relationship between the data perceived by the robot and the behavior it must achieve according to this input. How to achieve to the target position (goal point), using range data from sensor measurements such as ultrasonic sensors and infrared sensors, to steer the robot safely through the uncertain terrain, which contains unpredictable obstacles?

Although, during more than 50 years, many different strategies, approaches and algorithms are investigated by researchers, there are only few major approaches to develop control algorithm for a mobile robot. One way is to look at the task to be performed, and appropriately generate the controls necessary to guide the robot to perform the task. The controls are typically steering angle and vehicle velocity for a mobile robot. The other way is sensing the environment, creating a model of the world surrounding the robot and planning the action on that model. This method is generally referred as “model based control” or more formerly as “deliberative control”. A third way is to view the task as a mixture or combination of more elementary tasks called behaviors. Programming the execution of a given task then reduces to finding the proper combination of those behaviors to produce the desired task. In such a case, the robot does not reason about its actions, but merely respond to external stimuli, therefore this final method is referred as “sensor based control” or as “reactive control”. Comparison of deliberative control and reactive control is summarized in Figure 1.4 below. Further details are studied in the following sections.

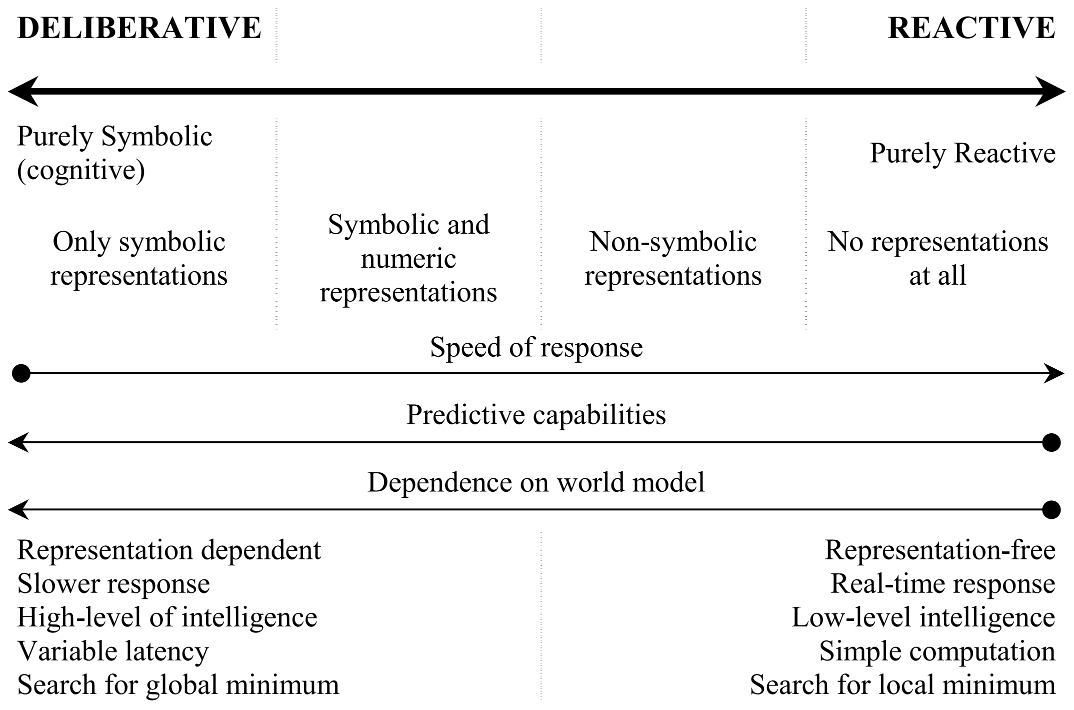


Figure 1.4: Robot control system spectrum.

There are many variations of both control methods. Moreover, some researchers used suitable combinations of those approaches in order to overcome various disadvantages. In fact, looking at both oldest and the most recent and advanced work in this field, one can see that two very different assumptions are made: either the robot has no a priory information about the environment, or, the complete knowledge of the environment is priori known to the system and is introduced by the operator. Those methods are also summarized briefly in the following sections.

1.2.1 Table Look-up Method (Preprogrammed Robots)

In a fixed environment where every change is known in advance, the desired actions of a robot can be planned and coded in terms of actuator values or controllers reference values. Then those values are read from a table and corresponding actions are generated by the robot. Some sensors such as limit switches may also be installed and included in the control.

This method is advantageous where repetitive and fixed tasks are done, such as factories, warehouses and similar industrial plants. Beside advantages like requirement for very small computational power and sensors, many drawbacks exist. Those includes non expandability, requirement for very stable environment where almost no change occurs an so on…

1.2.2 Deliberative Control

The development of mobile robots capable of autonomous navigation requires intelligent control strategies that can handle the uncertainties presented by the real world. At the same time, real-time performance must be maintained. Traditional approaches to mobile robot navigation have dealt with these requirements with the use of explicit pre-determined world models and computationally intensive planning algorithms. Those systems can be simplified as "sense-model-plan-act" cycles as shown in

Figure 1.5. According to this approach, perception is obtained from sensors that are collecting data about the world. Then according to those information world model, which is previously inserted into the controller, is updated. New actions of the robot are

planned according to the updated world model data. Finally, planned actions must be executed both on world model and on the real world. Then the algorithm is repeated.

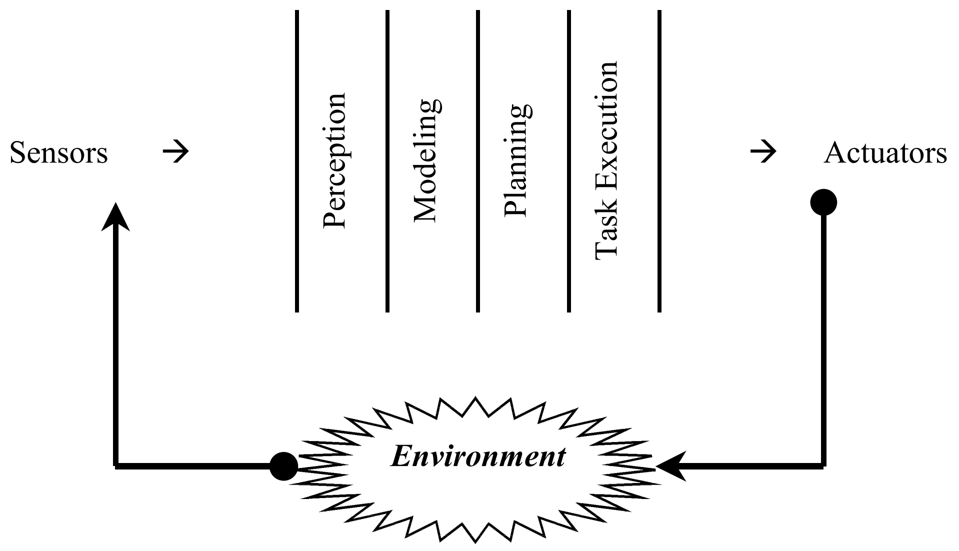


Figure 1.5: Decomposition of a mobile robot control system into functional modules in deliberative control method.

Deliberative control systems have been criticized on many drawbacks over the last decade. Nevertheless, the resulting systems exhibited acceptable performance in spite of many drawbacks of the architecture. Major problems with those systems are brittleness for environmental changes, extensive requirement of hardware, insufficient real-time performance and lack of expandability.

Deliberative control is model based, which requires the presence of a complete knowledge about the environment. This knowledge is obviously far beyond the possibilities offered by the sensors of the robot. Therefore, the robot uses preinstalled information about the environment and always assumes its correctness until the presence of a sensory data that is conflicting with the model. In such a case, that part of the model is updated and new action plan is prepared. Obviously, the change in the environment situated outside the sensors range of the robot is not detected. Moreover, it is assumed unchanged. Therefore, this control technique is classified as “open loop control” and it is failed when severe changes occur in the environment.

An autonomous mobile robot operates in real world environments where the boundary conditions are changing rapidly. Actually, those changes are so rapid that either the controller does not have enough computation time to respond or the robot does not have enough sensors to detect the whole area. One solution to that problem is using extensive computational power combined with large amount of sensor readings. However, the need for significant amounts of memory and processor power to store and maintain world models precludes the use of small, lightweight computing systems. Although this is not necessarily a drawback for stationary manipulators, it is a problem for autonomous mobile robots, for which computational resources must be carried onboard.

Furthermore, in such a design addition of new functionality to the robot requires changes of the sensor mapping, world model, and action planning. Clearly, this is not easier than designing a completely new mobile robot.

Finally, in this approach, all modules must be complete and working before an action takes place. If one module fails, either due to software or hardware problems, the entire system will break down.

1.2.3 Reactive Control: Behavior Based Approach

More flexible, robust and easily applicable approach to control algorithm development is first introduced by Brooks in 1985. This new approach, "behavior based control", rather than dividing the control system into serially connected functional units, decomposes the problem into task achieving behaviors. Accordingly, the controller is formed by those simple primitive behaviors that are running in parallel and interrupting each other when necessary as shown in Figure 1.6. In Brooks's implementation, behaviors are competence layers constructed as "Augmented Finite State Machines" (AFSM).

Primitive behaviors are, generally, open-loop transfer functions such as movement, servo-to-heading, turn-until-clear, wander/explore, and obstacle avoidance. Although these can function as stand-alone behaviors, they are used as building blocks for the more complex composite behaviors. Composite behaviors, on the other hand, are close-loop functionals that continuously sense and react to local changes in the environment. Examples are follow wall, follow boundary, and circumnavigate-obstacle.

This approach supports multiple goals, multiple sensors and brings a robust and additive control system in the picture. In principle, this new approach can also be applied in all situations where classical control theory is not feasible, usually due to tremendous modeling difficulties. Moreover, robustness of this approach is higher compare to the older methods since in case of failure of one of the modules then other behaviors can still produce meaningful actions for the robot.

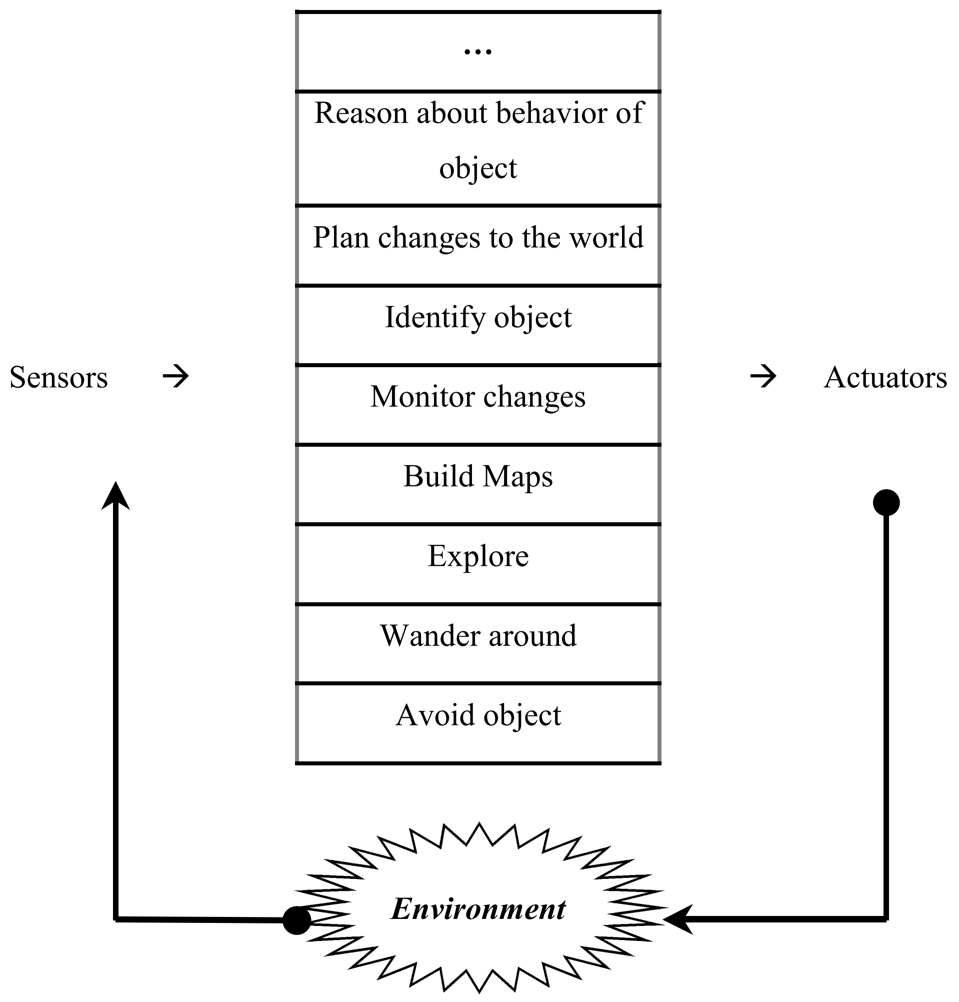


Figure 1.6: Decomposition of mobile robot control system into task achieving behaviors running in parallel.

The reaction of each layer to the obtained sensor stimuli produces an output. This output is generally in the form of a reference for the next controller. In mobile robotic, layers generally produces linear velocity and steering angle that will be used as a reference for the wheels velocity control unit. As an example if an obstacle is detected in the heading of the robot, obstacle avoidance layer produces a new reference steering angle to avoid this obstacle. Then wheel speed controllers will adjust the speed of the driving motors to follow the given reference.

One of key issues of behavior-based control is how to efficiently coordinate conflicts and competition among different types of reactive behavior (equivalently among layers) to achieve a good performance. However, in Brooks subsumption architecture, although each layer runs in parallel, the output of only one is executed in a specific time bringing the problem of action selection.

The output produced by each of those primitive behaviors passes through some arbitration and then realized. The arbitration, which is actually “action selection”, is realized in two distinct manners. Either Brooks’s hard-wired method is used or a stand-alone module that is collecting outputs of each layer and then deciding the reference for the next controller is designed. In the first case, each higher-level layer is assumed to have higher priority and always the output of the highest-layer behavior is used ignoring the lower layer’s outputs. Alternatively, different researchers applied a stand-alone module acting as an action selector. This new module, placed between horizontal layers and the motor control units, collect reference data produced by each behavior and according to a predefined judgment method select the best action to take. Here the best action can be defined as the action that will increase overall performance of the mobile robot. Different methods applied as behavior arbitration is studied in the following section.

There are many different extensions of Brooks’s architecture for much different applications. Parker [Parker 1992] has modified behavioral controls for multi-robot cases. Arkin [Arkin 1993] extended behavioral control architecture for multi-robot control. Moreover, Eustace [Eustace 1994] created “Behavioral Synthesis Model” specially designed for facilitate cooperation and coordination between multiple robotic devices for execution of complex tasks. In his architecture, there are four strategy levels: self, environment, species and universal. Each of those levels consist of different behavior pattern such as obstacle avoidance, battery monitoring etc. Each behavior pattern creates a motion response and another function called utility. The utility is a measure of importance of the created motion. Then responses created by individual pattern are resolved using a process of additive synthesis where utility is used to resolve conflicts. Moreover, communication between agents is also implemented as a behavioral pattern inside the species layer.

1.2.3.1 Action Selection for Behavior Based Robotics

There are already several implementations of primitive behaviors using variety of methods. Each of them is aiming intelligent behavior using simple reactions to the perceptions coming from sensors. Most of them show high performance under dynamically changing, complex environment when executed alone. However, once multiple goal realization, such as avoiding obstacle while reaching a target point, comes into picture then action selection gains importance.

Between 1985 and 1988, Brooks used subsumption architecture to develop several robots, all with wheels. In the summer of 1988, he built Genghis, one of the most famous six legged robots, that is shown in the Smithsonian Air and Space Museum now [Brooks, 1989]. In all of those robots, the action selection method was direct replacement (inhibition and suppression) of the control output from higher levels to lower levels. As an example, if an obstacle is detected, all lower level functions were stopped and only obstacle avoidance lever was working. Although the configuration works well in less crowded areas such as laboratory test beds, the multi goal realization in a real world was not so successful. Consequently, better action selection methods were needed.

Many researchers suggested and applied fuzzy logic based controllers [Tunstel and Jamshidi 1994, Mochiada et al. 1995, Chenyu et al. 1996] for this special purpose. The advantage of fuzzy logic is that potentially conflicting functions can be fused in a natural and smooth way, so that a reasonable decision can be made to serve both functions.

Moreover, Mochiada proposed an “Emotional Mechanism” similar to the human emotional mechanism [Mochiada et al. 1995]. In his method, first a frustration level is defined witch has two states: pleasantness and unpleasantness. Then this frustration level is continuously updated using some sensory inputs according to his artificial amygdaloidal nucleus model. In this model, a ”patience against unpleasantness” variable exists. With the help of this variable, robot may sometimes move opposite to the goal poison to handle some of the obstacles, or come very close to the obstacle in order to reach goal points very close to them. This feature is also a result of the applied Braitenberg’s action selection architecture (see Figure 1.7), which relates sensor data directly to the motor without any explicit calculation. Mochiada's final experiments showed high performance in both obstacle avoidance and goal point reaching.

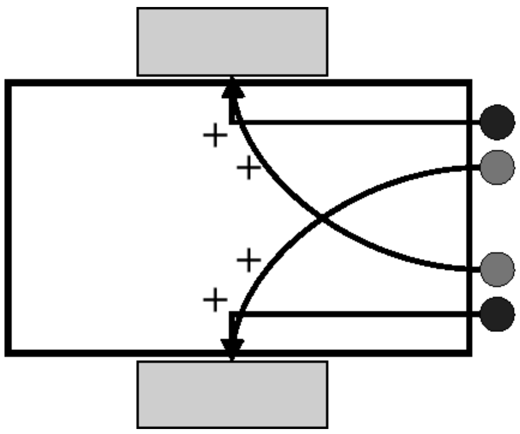


Figure 1.7: Braitenberg Vehicle 3c as example of Braitenberg action selection mechanism.

1.2.4 Multiagent Event Based Control Architecture (MEBCA)

Proposed by Chen [Chen and Luo 1998], MEBCA is an event based and multiagent structured control algorithm. In this approach, tasks are divided into simple behaviors in form of agents, as in the case of behavior-based approach. However, this time agents are grouped together to form classes. Proposed classes are planner, assistants, motion executors and control center. According to the agent groups and their priorities, it is easier to decide which agents should run in parallel and which should run sequentially for building a highly efficient multiagent system.

Example agents for such a system are obstacle avoidance, goal following, wall following, docking, behavior planning, path planning, self-referencing, error detection etc… Each agent of the system is capable of dealing with a specific assignment by its local intelligence. The cooperation of the all agents is through the communication handled by the control center, which is also an agent.

Various agent will need different data, e.g. some agents will need only sensor readings to model the world while some will need the complete world model for action planning. Therefore, the shared memory records both low and high level information as well as setting necessary for the whole system. Any agent can access that information to complete its own task, by communicating with the control center. The control center responds with the requested information according to the priority of the agents class.

Another important agent in the system is behavior arbitrator, which decides the activity of the agents. Once a new command, such as a new goal position, arrives from the application layer, control center will set the motion planner active to plan a sequence of behaviors or a path to goal. According to the plan motion assistants activates executors.

Main drawbacks of this proposed architecture is the difficulty to program, expensive computational power, and requirement for relatively large memory.

1.2.5 Other Methods

Since the existence of the robotics many different approaches are developed. It is obviously impossible to summarize all of them. However, we tried to cite major ideas that are either mostly used or formed basis for other approaches. Moreover, many researchers designed and tested combinations of methods to overcome disadvantages of them. Combination of model based and sensor-based algorithms are mostly studied.

One successful study is realized by Kweon [Kweon, at al. 1992]. Kweon applied three-clustered system architecture to overcome drawbacks of the behavior-based approach by using traditional approach as a secondary control. Three clustered system architecture consists of a manager (motion executer), and three-clustered behaviors: reflexive, purposive and adaptive level behaviors. The reflexive level group consists of agents, which maintain minimal safety of the robot. The role of the purposive level is to achieve global mission of the robot such as navigating the robot to the goal point safely. Finally, the adaptive level agents save the failure of the purposive level agents or deadlock situations.

Another successful example is realized by Ma, Li and Liu [Ma et al. 1996]. In their system, a global path-planning algorithm is used to create some “subgoal” points using priory knowledge of the environment if available. Moreover, a pure reactive algorithm is designed for local obstacle avoidance that is running when trying to reach subgoal points. By the way, the robot safely travels in crowded environments, responding to both stationary and mobile obstacles.

1.2.6 Local Obstacle Avoidance Methods

Serious researches are going on, to create an "everyday robot" that will operate in public areas. Obviously, safe operation of robots is becoming increasingly important as human-robot interaction increases. Moreover, areas where robots are operating are becoming more busy compare to standard industrial application.

Being one of the oldest problems obstacle avoidance is still a serious area of research especially in mobile robotics. There are many different methods, which are running robustly. Many well-known local obstacle avoidance methods are based on the computation on which the robot shall move. Prominent examples are potential field methods and vector field histograms. Typically, those approaches do not take into account that vehicles moves on arcs rather that rotation at a specific point.

Different modifications are applied to both methods in order to overcome existing drawbacks. As an example, Kweon and Watanabe [Kweon, et al, 1992] proposed a method, based on active contour models, to compute the position and direction the robot by fusing multiple sensor data. To determine the best position and orientation of the robot using the range readings from multiple sensors, Kweon make use of energy minimizing curves, which were introduced by Kass in 1987.

1.2.6.1 Edge-Detection Methods

One of the oldest but popular obstacle avoidance method is edge-detection. In this method, sensors are used to detect edges of the obstacle around the mobile robot. The line connecting consecutive edges are considered as the boundaries of the obstacle. Then an algorithm tries to steer the robot around the obstacle.

A disadvantage with the current implementations of this method is that the robot stops in front of the obstacles to gather proper sensor information. However, this is not an inherent limitation of edge-detection method; it may be possible to overcome this problem either with faster controllers or with suitable buffer algorithms that will collect sensor information while robot moves.

In a modified version of the edge detection method, the robot remains stationary while taking panoramic scan of the environment. After the application of certain linefitting algorithm, an edge based global path planning algorithm determines the robot's subsequent path.

Being mostly applied with ultrasonic sensor, the method generally suffers from shortcomings of the sensors. Mainly poor directionality, frequent misreadings and specular reflections are important error sources. Any of those errors can cause determination of an edge at a completely wrong direction, which will result strange paths for the mobile robot.

Another important drawback of this method is the inapplicability to moving obstacle avoidance not only because of the computational power requirements, but also because of long-term path planning structure. Robot retains old obstacle places, and use them in future path planning's. Therefore this method is mostly model based and far from being a reactive method.

1.2.6.2 Certainty Grids Method

Developed at Carnegie Melon University (CMU), certainty grid method uses probabilistic representation of obstacle locations in order to overcome inaccuracies coming from sensory data. In the certainty grid method the robot creates a two dimensional array representing its environment. Each element of the array is referred as cell, and contain certainty values, that indicate the measure of confidence that an obstacle exists within the cell area or not.

According to sensor readings, content of each cell is updated with a probabilistic method that takes into account the characteristics of the used sensor. As an example, ultrasonic sensors have a conical scanning area and the returning value of a scan is the radial distance to the nearest object in the cone, yet does not specify the angular location of the object. Statistically, detected object are more likely to be in the acoustic axis of the sensor rather than being in the angular limits. For this reason when updating a cell, using ultrasonic sensor readings, CMU's method uses a probabilistic function that has maximum value at the acoustic axis and that decreases toward minimum values at the angular limits of the scanning are.

A mobile robot with 24 sensors is used in CMU's applications, to take panoramic scan of the environment surrounding the mobile robot. After suitable update of the cell values, mobile robot move to a new position, and then a new scan is done. Finally, a fairly accurate map of the room is obtained.

Certainty grids method also assumes that there are no moving obstacles in the environment. In a multi robot application example, each robot will detect others in more than one place and more than one time. Since no distinction is made between stationary and moving obstacles, the resultant map will be full of obstacles without any space to move. This major drawback is caused by the model-based structure of the method. Nevertheless, some modifications can be made to overcome these disadvantages.

1.2.6.3 Potential Field Method

The idea of applying imaginary forces on the robot, caused either by obstacles or by the goal point, is first suggested by Khatib. In this method, obstacles exert a repulsive force on the mobile robot while the goal point toward where the robot is trying to reach applies an attractive force. The resultant force vector influences the motion of the robot as an accelerating force. Whenever the robot changes place the algorithm is repeated.

Researchers showed great interest to this idea. In a short time period, many variations are developed and used. Krogh, Thorpe, Newman and Hogan are the major researchers that worked to enhance this new approach. However, none of them used real sensory data, all assumed having a known and prescribed world model in which geometric shapes represent obstacles. Moreover, robot's path is generated off-line assuming ideal dynamics. And then followed by the robot.

By contrast, Brooks and Arkin equipped their robot with rings of ultrasonic sensors and used a potential field method in the feedback of the control. Brooks's robot treated measurement from each ultrasonic sensor as a separate repulsive force. In case the vectorial sum of those forces exceeds a certain threshold, the robot changes direction under the influence of the calculated force vector. In this implementation, only one set of readings is considered and past readings are lost. Arkin also programmed his robot in a similar fashion. This way, robots were totally reactive to the environment and capable of avoiding both stationary and moving obstacles. However, the lack of long term memory and global path planning results in inefficient paths.

This approach mostly suffers from “equilibrium of the forces”. Imagine that there is an obstacle just in the heading direction of the robot and it repels the robot. Imagine also that the goal point is hidden in the other side of the obstacle and it is just in the heading direction. It will be such a moment that the total force on the robot will be zero. In such a case, either the robot will not change direction (and hit the obstacle) or it will oscillate back and forth according to the size of the repulsive and attractive forces. Similar problems are encountered near to symmetrically placed obstacles. As examples, in a corridor or in the opening of a door, the robot will most probably experience a repulsive force in the heading direction.

One successful application is realized by Song and Chang [Song and Chang 1999]. They used a CCD camera and sixteen ultrasonic sensors to detect distances to the obstacles. Moreover, a neural network, which is trained off line, is used to predict the next position of the obstacle. Then potential field method is used to determine the direction and the velocity of the robot.

1.2.6.4 Virtual Force Field Method

Developed by Borenstein and Koren, “Virtual Force Field” (VFF) method can be seen as a combination of the CMU’s certainty grids method and potential field method of Khatib. In this approach the environment is divided into a two dimensional array, with each element called cell, containing certainty value of the presence of an obstacle. As the robot moves, a window of $w_S \times w_S$ cells, in the center of which the robot is assumed to be, accompanies it. In each reading the certainty values of contained in this window is updated. Similarly to CMU’s method, a probabilistic update is used. However, this time only one cell is updated in each sensor reading. This simplification yields faster update of the cell data resulting a probabilistic distribution with lower computational power.

Next, the potential field idea is applied to the obtained histogram grid. Each cell in the window accompanying the robot exerts a repulsive force with magnitude proportional to its value (certainty of having an obstacle) and inversely proportional to the square of the distance between the robot and obstacle’s position. Those forces are summed with the attractive force toward the target position. Finally, the resulting force influences the motion of the mobile robot.

In this approach, sensor inaccuracies are eliminated with the usage of the probabilistic values. Moreover, sensor data influences the steering control immediately whenever a new obstacle is determined. By the way, a fast reflexive behavior is obtained.

Main drawback of the system is caused by the size of the cells. As the size of the cells decreases, the computational power requirement increases. In addition, as the cell size increases drastic changes in the force is observed in each movement, resulting of a non-smooth trajectory. Another problem is encountered while traversing narrow corridors. While the robot move along the centerline, a stable and smooth trajectory is observed. However, if the robot comes close to one of the walls, it experiences a strong force resulting to a motion toward the other wall. Finally, an oscillatory motion is observed.

This method also suffers from equilibrium of forces as in the case of potential field method. However, this is less observable since distances to the obstacles are classified (using cells) and force values are not continuous but increases by steps.

1.2.6.5 Vector Field Histogram Method

Virtual force field method (VFF) using hundreds of data points produces a repulsive force vector, namely magnitude and direction. Since this transform is a single step reduction, detailed information about the local obstacle distribution is lost. To remedy this shortcoming, Borenstein and Koren developed “Vector Field Histogram” (VFH) method.

The VFH method uses two-step data reduction data technique rather than single-step technique used by VFF method. Therefore, three level of data representation exists.

The highest-level hold detailed information of the robots environment. With an identical method with VFF, a two-dimensional Cartesian histogram grid is continuously updated in real-time using sensor data.

The intermediate level is a one-dimensional polar histogram grid. This new histogram grid is constructed around the robot’s momentary location. This new histogram is composed by n different sectors each having angular width α. Each of those sectors holds a value representing the obstacle density in that direction.

Finally, the lowest level keeps the output of the whole VFH algorithm; the reference values for the drive and steer controllers of the vehicle.

VFH method is a local path planner, and it does not attempt to find an optimal path. Furthermore, a robot with VFH control can de trapped in dead-end situations. Being trapped for the robot refers the case where the robot shows cyclic behavior around one or more stationary points.

1.2.7 Goal Position Tracking (or Goal Position Reaching)

A mobile robot, before realizing its predefined task must reach to a specific position and sometimes via specific orientation. Alternatively, it may be required for the robot to wander around while realizing its primary task such as mapping of the environment by collecting sensor data. Whatever the context of the experiment is, reaching to a target point is essential in mobile robotics. In order to realize this subtask, researchers used two simple approaches.

First approach is the one, which is used, in deliberative control. The robot having the complete representation of the environment plans its path to follow as a function of tike. While making the plan, the maximum speed of the robot and other kinetics and dynamics constraints are also considered. Finally, the reference is followed by the robot until the presence of an unexpected event, such as the detection of an unknown obstacle. In such a care a new plan is built, and processed. The main drawback of this algorithm is obviously the extensive computational power requirement. Moreover, in some cases the robot may travel long time toward a goal point, then detect an obstacle in front of it, and then may plan a completely off direction path that will require it to travel back.

Another approach is the one that is called potential field method. In this method, the point in space where the robot tries to reach exerts an attractive force on the robot. This virtual force orients and accelerates the robot toward the goal point. The main disadvantage of this algorithm is caused by the local minimums of the potential field. Remarking that the virtual forces acting on the robot creates a potential field, it can be proven that the robot is always moving toward the minimum point of this imaginary potential field. In such a case, the existence of local minima will trap the robot at that specific point in the environment.

Moreover, there are some algorithms, which are combining those two methods. The aim is clearly overcoming drawbacks of both methods while benefiting advantages.

1.3 Comments

As seen in the above brief survey of literature, there are many different approaches for mobile robot control. However, we can group all of them in three different categories. In the first category we can place all the deliberative control approaches, in the second one all the reactive control approaches and finally in the third one composite solutions.

Deliberative control as a main disadvantage has the problem of what we will call knowledge. A mobile robot with such a control must know almost everything about the environment where it is asked to perform any task. However, this may not be always possible; consider a mobile robot working in space or in the ruins after a natural disaster.

Some of the researcher tried to leave the robot to the environment to learn it. However, robots needed not only long time but also had experienced many crashes before having an acceptable knowledge about the environment. Moreover, this learning procedure takes certain time, which may not be present in the situation, remembering the space and ruins examples…

In deliberative control, positions of all obstacles except moving and newly installed ones are known. Moreover, the appearance of an unknown obstacle will cause an update in the path plan, and consequently a new and free path will be generated. In this approach, obstacle avoidance is hidden in path planning.

Reactive control, on the other hand, does not require any knowledge about the environment. Consequently, no free path planning exists in this approach. Therefore, obstacles must be detected ant any instant of time and a reaction that will help the robot to avoid obstacles must be generated. The main disadvantage of this approach is the difficulty of ensuring the continuity of the robot motion. Although robots realized using this approach shows high performance with moving obstacles, they may sometimes be locked in relatively simple situations such as door openings and cavities.

Finally, to overcome drawback of the reactive control researchers studied a composite solution. In this case, local obstacle avoidance is handled by the reactive control, but globally looking, the robot is directed toward the subgoal positions one by one. Those subgoal positions are derived by the deliberative control to move the robot safely to the goal position. Still the approach requires knowledge about the environment. However, this time the knowledge is limited.

From our point of view, if a general-purpose mobile robot control will be developed this must have reactive navigation in the environment, which does not require any previous knowledge. Nevertheless, the system must also be open for addition of deliberative control in case it's needed.

1.4 Statement of the Problem

The development of satisfactory control method for an autonomous mobile robot that can be part of a multiagent system is still an open problem. For such a system, we can identify a number of requirements, which can put constraints on possible control method we are proposing. As a minimum, required control method must work in real-time and be able to respond to sudden changes that may occur in the environment.

Mobile robots are most of the time required to realize more than one goal at a time. In the simplest case, these goals may be reaching to a specific point while avoiding obstacles. The relative importance of the goals is context dependent, however they are never disregarded. Control of a mobile robot must take care about the support of multiple goals and must find the way to select the action that serves a maximum number of goals at the same time.

A mobile robot that is running a feedback control usually has more than one sensor. Those sensors may be for internal states of the agent, such as wheel velocities, or for measuring environmental states, such as the distance to the closes obstacle. Moreover, mentioned sensors will probably be selected from a wide range of possibilities, such as TV cameras, encoders, resolvers, infrared or ultrasonic distance detectors and so on. All sensors will have an error component together with their readings. Some sensor reading will overlap in the physical quantities they measure and will often give inconsistent readings due to sensor errors or environmental conditions. In the case of errors, the robot must still be showing meaningful behavior within the possible limits of the physical restrictions. The control for a mobile robot must be resistant to drastic changes in the environment.

A mobile robot control cannot be specific to a single type of mobile robot. Since mobile robots are designed for different purposes, physical shape and mechanics, such as driving mechanism, of the robots would be very different. For example, some robots may be circular shape while other may have rectangular one; some may be activated by step motors while others by AC machines. Obviously, mini and micro size mobile robot will be built beside big and giant size ones.

In many cases, robots are used as groups either to increase performance of the requested task while decreasing time consumption or to realize tasks that are beyond physical limits of the robot. For example, a forklift with load capacity 50kg cannot move 90kg block. However, two of them, working in coordination, can easily realize this task.A mobile robot control must be open for additional controls that will guide the robot to be a part of a multiagent system.

The goal of this work is to develop a new structure of the mobile robot motion control system, capable of being a building block of an intelligent agent. The algorithms should be platform independent and should solve at least the obstacle avoidance and the drive to target function. It has to be suitable for the layered control structure.

The idea to create imaginary repulsive forces on the mobile robot is almost the only approach in reactive obstacle avoidance. Although applications differ in creating the forces and in the way of influencing the robot's motion accordingly, the core idea is always the same. Consequently, robots using this obstacle avoidance method suffer from local minimums that appear in the environment. To pull robots from those minimums, an attractive force toward the goal point, or target point, is added to the system. However, this new force, removing some of the local minimums created other ones especially near symmetrical distributions such as door openings or corridors. Most of the time, external "noise" is used as solution to remove the mobile robot from the local minimums [Ferber 1999]. This noise is randomly created force acting on the robot. However, if this force is not strong enough to pull it from the local minimum then the robot is trapped forever.

Like other researchers, we implemented potential field method for obstacle avoidance. However, we always treated repulsive force of the obstacle and attractive force of the goal point separately. Then we "fused" two behaviors, obstacles avoidance and goal tracking. This way we eliminated this major drawback of the potential field method.

However, the problem is now reduced to the arbitration of the behaviors. The system should be designed such that the fusion of two behaviors must be possible. In many mobile robot reactive control examples, direct control of the wheel velocities is used. In such a case, each behavior will produce an output for the wheels, however the union of the whole outputs may be far away from the physical capacity of the robot. Moreover, direct control of the wheels may serve the originating behavior shortly but destroy another behavior for the rest of the time. For example, imagine an obstacle placed at a very close distance to a wall. A mobile robot that is approaching this target point will perceive the obstacle. Until that time, goal-tracking behavior used to be satisfied with the behavior of the robot and was not creating any command that will affect wheel speeds. With the perception of the obstacle, direct command of the obstacle avoidance behavior to the wheels will change moving direction of the robot. Then the robot will move once more toward the same goal point and then this will repeat infinitely. For the mentioned reason, the behavior arbitration must always be aware of the all behaviors running in the system and then decide about wheel velocities.

In this thesis, we will be concentrating on the design of a general reactive control scheme for mobile robots. During this work, the basic ides used are not much different of what is used until now. However, we will treat the system as a whole and design each individual behavior such that the rest of the system will not be destroyed.

2. PLANT AND METHOD

2.1 Plant

We define our plant as the combination of all the elements present in the experimental environment. Major elements in our plant are entities forming the multi agent system. Those entities are either mobile or stationary robots or any other abstract entities such as a software routine. Of course, if in an environment there are at least two robots then automatically there are obstacles in that environment since each robot is it-self is an obstacle for the other(s). Obstacles are elements, situated in an environment where they are either preventing or limiting motions of the agents.

For convenience, all of the other elements in our plant are referred as obstacles. However, we can always classify obstacles according to different criterions. Actually in our environment, we have two different groups of obstacles. First group is formed by stationary obstacles. Stationary obstacles do not change their position with respect to the origin of the system, or to a constant point in the world, in time. Elements of the second group of obstacles, in the contrary of stationary obstacles, move around the world. They are called mobile obstacles. Those can be any other agent, a human, an animal or even a ball that is rolling in the world.

In brief, we can divide all physical and abstract objects in the environment into two groups. The first one is composed of agents and the second of obstacles. Further details are defined and discussed in the following sections.

2.1.1 Agent

In this thesis, we are concentrating on collective robotics, which relates to the creation not of a single robot, but an assembly of robots, called agents, that cooperate to accomplish a mission. A mission or task is a collection of goals that must be reached one by one. Here goal refers to anything that is realizable by an agent, such as going to a specific point in the environment.

We can divide the properties owned by those agents into two groups, physical and abstract properties. Physical properties of the agents are those who are mostly related with the hardware used. Sensors owned, driving system, size, mass, inertia etc… all belong to this category. Abstract properties on the other hand define the ones coming from software and from the existence of some additional hardware since they add different skills to the owner agent. Those properties define both realizable tasks of the agent and the behavior(s) shown by the agent. Behaviors are series of actions performed by the robot when its controller is executed.

2.1.1.1 Abstract Properties of the Agent

First of all our agents are endowed with autonomy. Etymologically a system is autonomous if it is ruled by the laws which it self decreed. This means that they are not directed by commands coming directly from a user (or from another agent/shared controller), but by a set of tendencies. In most general form, a tendency is an internal state of the agent, which either impel an agent to act or tend to limit the actions. As an example, appetitive behavior is a tendency that may drive us toward a restaurant or, generally speaking, toward food… Another consequence coming from autonomy is that an agent must be able to control (or be aware of) its own resources like power, memory, skills and expertise's.

Agents forming our system may all be totipotent (each agent have all the necessary skills to complete the given task alone) all specialists (each agent has a limited number of skills required by the task) or a mixture of both types. Those agents may or may not interact with each other while running. However, each agent must at least have communication to be a part of a multi agent system.

Our agents are all reactive. A reactive agent is the one that do not use symbolic representations but merely responses to sensor inputs. The advantage of having reactive agents is that those agents are much more flexible to the changes in the environmental conditions such as the place or the number of the obstacles.

Each of our agents is controlled via reactive control scheme. Our controller is composed of many layers running in parallel. Examples are obstacle avoidance, drive toward goal, move enable/disable etc… Each of those layers can be seen as different modules that are controlling the agent. In fact, executed alone each layer will produce an output that is a meaningful behavior when observed on the robot. For convenience, we order these layers according to their priority. The topmost layer is the one that has highest priority and lowest priority layer is at the bottom. All of those layers use sensory information that they needs, and produces an output. Those outputs constitute the reference for the next controller. In a mobile robot, this next controller is generally either a position control or a velocity control module, which is generally referred as low-level motion controller. A general setup for our layered structure is shown in Figure 2.1 below.

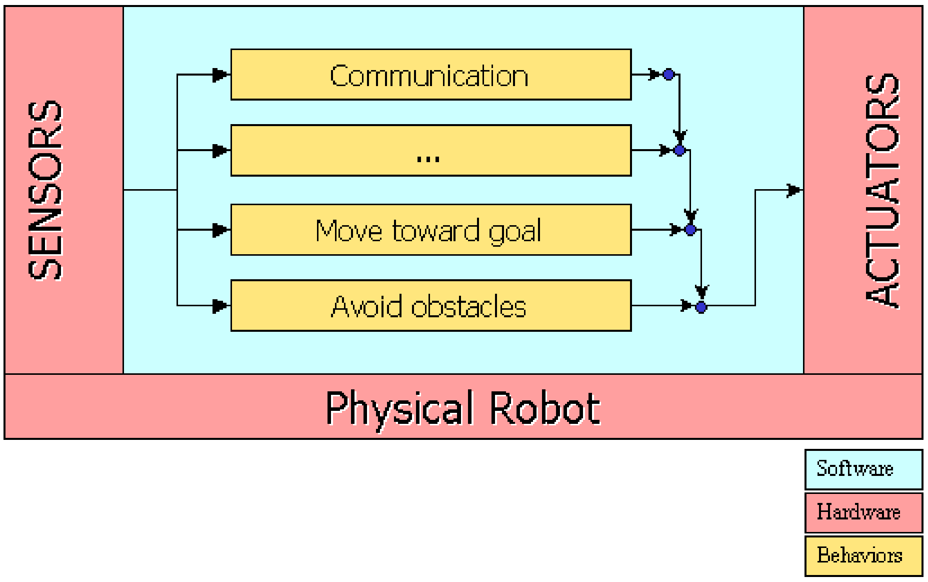


Figure 2.1: General setup of the layered structure as introduced by Brooks.

As seen on the figure (Figure 2.1) major tasks of a mobile robot are put as different layers. Each layer reacts to particular stimuli and prepares its output. Obviously all of those outputs cannot be executes at one time since some of them may conflict. Actually, they will be conflicting most of the time, an obstacle placed between the agent and the goal point will impede the agent while trying to move toward the goal point. Therefore an arbitration, action selection or combination method must be applied. The combination of the reference outputs received from layers can be subsumption, linear combination or any other suitable way. Once those reference outputs are combined, they are sent to the main controller. Finally, main controller follows that reference by generating necessary motor control signals such as PWM.

In the system, the output of the sensors may be anything possible as long as related layer(s) knows how to decode it. As an example, a vision system may return the exact coordinates of the object in space, or polar coordinates, while an infrared sensor may return just the distance to the object in its range. Similarly, the lower level motion controller of the agent may be controlling the caster wheel, a differential drive system or any other physicals system. While doing this, it can use sliding mode control, P, PI, PID or any other type of control. As long as this low-level motion controller accepts the output of our software as reference input this layer could be anything possible.

We can state that, this structure constitutes only a "buffer" between sensor outputs and the input to the main controller. The performance of the software level is hardware dependent, but still it is applicable to all configurations that are able to give the same or similar sensor outputs and that is capable of producing the given low level motion controller reference. This is where the major advantage is observed. Suppose one has three different types of agents, a differential drive wheel set type, a car like and a biped for example. As long as each of them are equipped with the same sensors giving the same type of sensory outputs and receives the same type of reference to the lover level motion controller, the same software where layered structure is coded will run in each of them exactly the same way. However, the performance may vary with the performance of the lower level motion controller, positioning of the sensors and physical limitations.

2.1.1.2 Physical Properties of the Agent

Brooks's layered architecture offers many advantages such as modularity, ease of programming, robustness and platform independence. As seen on the Figure 2.4 and as explained by Brooks in 1986, this layered structure is almost independent from the physical robot itself. As long as the data output from the sensors and the input to the main controller is the same by means of measure then the rest of the system does not affect control layers.

Exactly the same way, our layered control structure is also totally platform independent. On any system, which offers necessary sensory inputs and that can accept output of our controller as a reference for the lower level controller; the controller will run exactly the same way. Nevertheless, for consistency, details of the used mobile robot model are defined in the following paragraphs.

Our mobile agents are all differential drive type, nonholonomic robots generally referred as “wheel set”. This type of robots is nonholonomic since it has three degrees of freedom (x, y, ϕ; position and orientation in space) but only two controls (v_R, v_L; right and left wheel velocities).

Obviously one of the most wanted features in a mobile robot is the avoidance of stationary and mobile obstacles. Although the problem is very old and very clear by means of the definition and requirements, no perfect (or unique) solution is found until now. Many different ideas are applied since the existence of the mobile robotics. Major ones are summarized in the first chapter. Each idea has its own requirements depending on the control algorithm’s inputs and outputs. Those are generally sensory request since before avoiding an obstacle one must at least know that there is something over there.

Between many possible alternatives, we selected potential field method similar to [Khatib 1985] and applied it in a slightly different manner. The idea is simple; there are sensors placed around the mobile robot in a suitable way. Using measurements from those sensors and their orientation data one can calculate both the distance from the obstacle to the body of the agent and the direction perpendicular to the surface of the obstacle. Although some restrictions apply, we can assume that this information can be gathered for both stationary and mobile obstacles, since a mobile obstacle can be assumed as stationary within a small instance of time.

Once the system is executed, each sensor will measure the distance to the closest object in its range. A “decoder” can use the information coming from consecutive sensor pairs to yield a repulsive force vector from the obstacle to the agent. Then the sum of these vectors defines the total repulsive force on the agent created by sensed obstacles. Details of the algorithm are given in below. However, obviously to run this algorithm we are not limited with distance sensors. A vision system will also perfectly run as well as any other sensor system that is able to measure the distance between the agent and obstacles. Nevertheless, as the number of distance sensors around the agent increases with well-balanced distribution (such as homogenous distribution), better avoidance of obstacles will be obtained.

2.1.1.3 Kinematics Model of the Agent

The vehicle considered in this work as a mobile robot has a differential drive form. There are two wheels placed on a common axis passing from the center of a circular body (one at the left and one at the right). Both of them are powered using DC servomotors either by gear reduction or by direct drive. One or more caster type wheels are used to keep the balance of the vehicle. This form of physical vehicle is referred as "wheel set". A simplified form is shown in Figure 2.2 below.

Wheel sets are mobile platforms that are subject to nonholonomic constraints since they have three degrees of freedom: x, y, ϕ position in space and orientation, but only two controls: either ω_R and ω_L right and left wheel angular velocities or τ_R and τ_L, left and right wheel torques respectively.

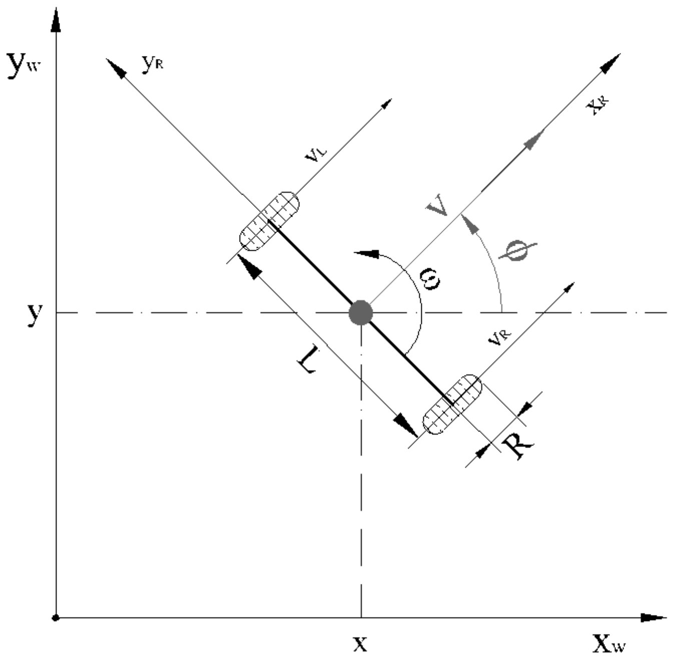


Figure 2.2: Wheel set with common fixed axis is used as physical agent. Wheels are constrained to longitudinal velocities v_R (right) and v_L (left) along robot axis x_R. Lateral motion along axis y_R is impossible. Differences in wheel velocities, v_R and v_L, result in translational robot motion v and rotational robot motion ω. Also shown is the robot poison (x, y, ϕ) in the world coordinate frame (x_w, y_w).

To determine the kinematics of the vehicle, consider a set of wheels with a common axle, but independent wheels as shown in Figure 2.2. Wheels are powered with two different motors. Assuming no slip at tires, the motion of each wheel is restricted to its longitudinal direction with linear velocities v_R and v_L, respectively, by a single nonholonomic constraint. In other words, no motion can occur along the lateral robot coordinate axis y_R. Also shown in Figure 2.2 is the robot configuration $q = (x, y, \phi) \in \Re^3$ that is composed by two parts. First part is the position of the robot as x and y coordinates in world coordinate frame (x_w, y_w). Second part is the orientation of the robot, which is measured as the angle between positive x-axis of the world coordinate frame (x_w, y_w) and the positive x-axis of the robot coordinate frame (x_R, y_R).

Controllable parameters in such a system are the two velocities v_R and v_L, which may be easily translated into the translational and rotational velocity variables $u = (v, \omega) \in \Re^2$ of the robot, for convenience. The motion of the wheel set in the world coordinate frame is given by the following first order differential equations

$$\begin{aligned} \dot{x} &= v \cdot \cos\phi \\ \dot{y} &= v \cdot \sin\phi \\ \dot{\phi} &= \omega \end{aligned} \tag{2.1}$$

where v is the velocity of the center of mass of the agent. This velocity can be expressed in terms of left and right wheels linear velocity, v_L and v_R respectively, as follows

$$v = \frac{v_R + v_L}{2} = \frac{R \cdot (\omega_R + \omega_L)}{2} \tag{2.2}$$

where R is the radius of the wheels, ω_L and ω_R are angular velocities of the left and right wheels respectively.

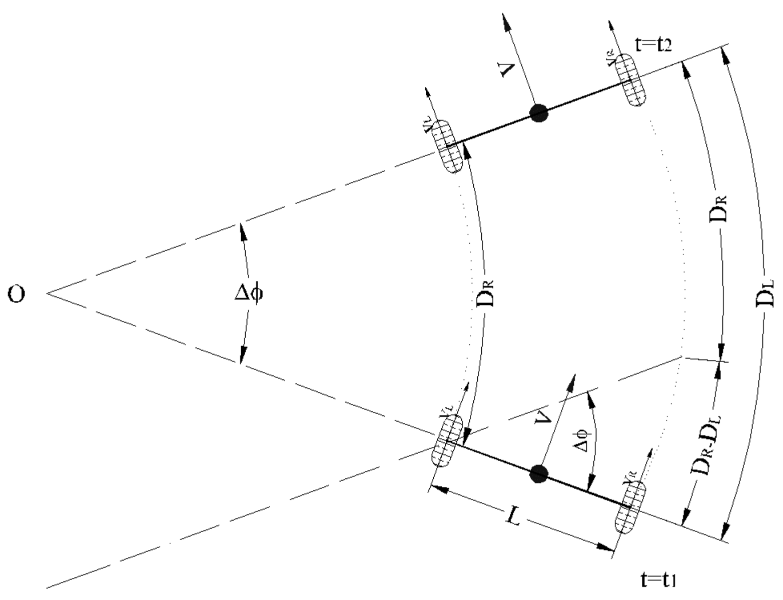


Figure 2.3: Instantaneous motion of the wheel set.

Moreover ω, velocity of the steering direction can also be expressed in terms of those wheel velocities. Consider the movement of the vehicle in a very short time interval Δt as shown in Figure 2.3. Using left and right wheel displacements, D_L and D_R respectively, we can determine ω as follows,

$$
\begin{aligned}
&\Delta t = t_2 - t_1 \\
&D_L \cong \Delta t \cdot v_L \\
&D_R \cong \Delta t \cdot v_R \\
&\Delta\phi \cdot L \cong D_R - D_L = \Delta t \cdot (v_R - v_L) = \Delta t \cdot R \cdot (\omega_R - \omega_L) \\
&\frac{\Delta\phi \cdot L}{\Delta t} \cong \frac{\Delta t \cdot R \cdot (\omega_R - \omega_L)}{\Delta t} \\
&\frac{\Delta\phi}{\Delta t} \cong \frac{R \cdot (\omega_R - \omega_L)}{L} \\
&\Rightarrow \frac{d\phi}{dt} \cong \frac{R \cdot (\omega_R - \omega_L)}{L}
\end{aligned}
\tag{2.3}
$$

where L denotes the length of the axis joining driven wheels. Finally

$$
\begin{aligned}
x &= \int v \cdot \cos\phi \cdot dt \\
y &= \int v \cdot \sin\phi \cdot dt \qquad (2.4) \\
\phi &= \int \omega \cdot dt
\end{aligned}
$$

forms the kinematics and dynamics model for this case.

As a summary, this model is given in Figure 2.4 below. In this figure position and orientation of the vehicle is obtained by the given wheel torques. However, it is always possible to disregard the first part of this picture and start from the calculations from the wheel velocities.

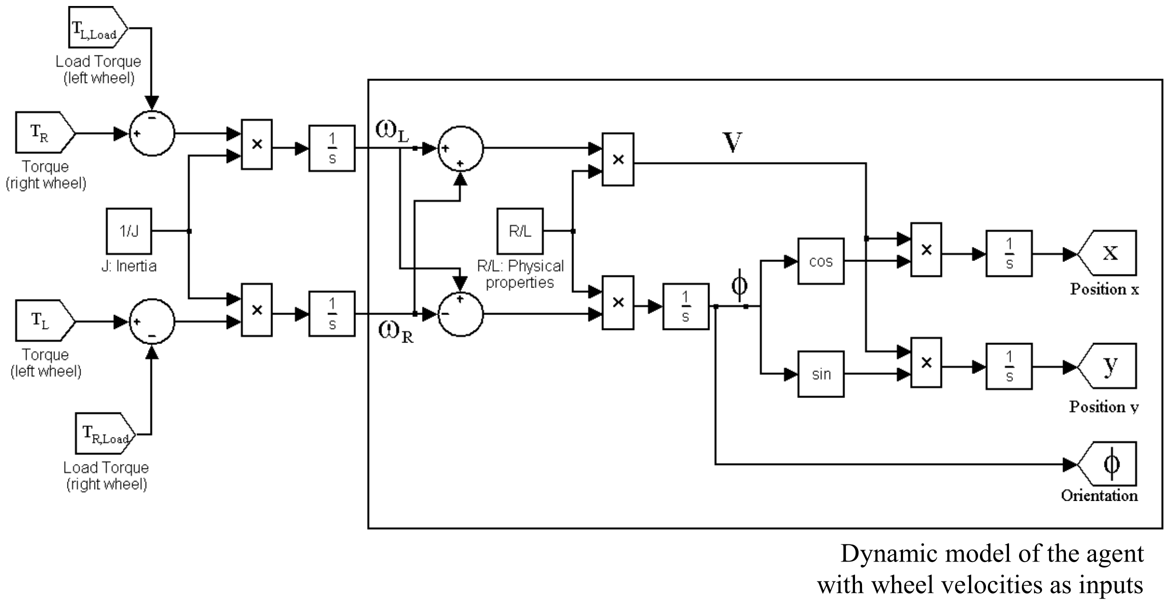


Figure 2.4: Simplified model of the vehicle kinematics and dynamics.

2.1.1.4 Augmented Dynamics of the Agent

We can also add complete motor, gearbox and wheel dynamics to the vehicle model and augment our model with the controller inputs as motor torques. In this case, this model will replace first and second parts in the simplified model of the vehicle (Figure 2.4 above), by directly producing velocity of the center of mass and steering angle of the robot.

Assuming that the differential type robot is driven by two DC motors with simple gearboxes, we can simplify the driving mechanism schematics as shown in Figure 2.5 below. The second driven wheel is assumed to be identically configured. The symbols used in the below figure as follows,

R_A	: Armature resistance
L_A	: Armature inductance
i_A	: Armature current
w_m	: Angular velocity of the motor
θ_m	: Angular displacement of the motor
T_m, T_1, T_2	: Torques
F_{c2}	: Coulomb friction coefficient
B_M, B_2	: Viscous friction coefficient
$\frac{N_1}{N_2}$	: Reduction Ratio
J_2	: Moment of inertia of the wheel
J_m	: Equivalent moment of inertia of the motor and motor load

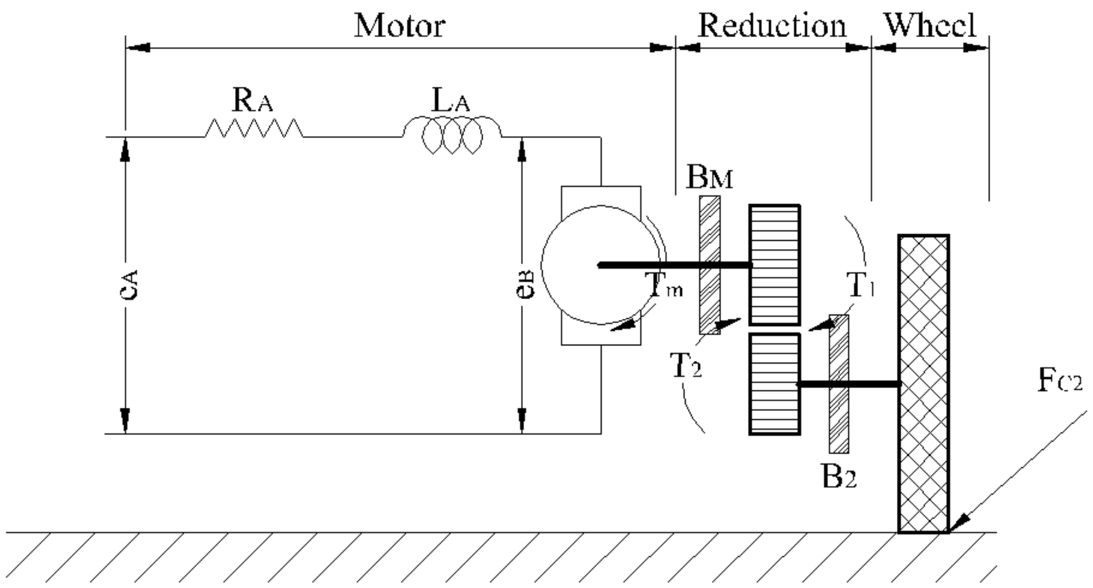


Figure 2.5: Simplified dynamics for a DC motor driven wheel with gearbox.

The DC motor dynamics can easily be found in many control books. By referring to [Kuo 1995], we can directly write the mechanical equations as given below,

$$\frac{d^2\theta_m}{dt^2} = -\alpha_1 \frac{d\theta_m}{dt} - \alpha_2 sign(\dot{\theta}_m) + \alpha_3 T_m \tag{2.5}$$

where

$$\begin{aligned} \alpha_1 &= \frac{B_{eq}}{J_{eq}} \\ \alpha_2 &= \frac{F_{ceq}}{J_{eq}} \\ \alpha_3 &= \frac{1}{J_{eq}} \end{aligned} \tag{2.6}$$

and

$$\begin{aligned} J_{eq} &= J_m + \left(\frac{N_1}{N_2}\right)^2 J_2 \\ B_{eq} &= B_m + \left(\frac{N_1}{N_2}\right)^2 B_2 \\ F_{eq} &= \frac{N_1}{N_2} F_{c2} \end{aligned} \tag{2.7}$$

Choosing state vectors as

$$\begin{aligned} x_1 &= v \\ x_2 &= \dot{\phi} \\ x_3 &= \phi \end{aligned} \tag{2.8}$$

then by combining equations 2.5 and 2.8, the equations describing the dynamics of the vehicle can be written in a more convenient matrix form as,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -\alpha_1 x_1 - \alpha_2 \frac{R}{2}(sign(g_1(x)) + sign(g_2(x))) \\ -\alpha_1 x_2 - \alpha_2 \frac{R}{2}(sign(g_1(x)) + sign(g_2(x))) \\ x_2 \end{bmatrix} + R\alpha_3 \begin{bmatrix} 1/2 & 1/2 \\ 1/L & 1/L \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_R \\ T_L \end{bmatrix} \tag{2.9}$$

where

$$\begin{aligned} g_1(x) &= \frac{1}{2}x_1 + \frac{L}{2R}x_2 \\ g_2(x) &= \frac{1}{2}x_1 - \frac{L}{2R}x_2 \end{aligned} \tag{2.10}$$

Moreover, our plant can be written in the nonlinear matrix form as:

$$\frac{dx}{dt} = f(x) + Bu \tag{2.11}$$

where $u = [T_R \;\; T_L]^T$ is the control matrix expressed in terms of the motor torques applied to the system, $f(x)$ and B are the matrices in equation 2.8

2.1.1.5 Simplified Model of Agents

Although the augmented model that contains both vehicle and all other driving mechanism dynamics is not difficult to simulate, we preferred not to include this part in our code for simplicity. The assumption is the following; agents have servo drives with velocity controller implemented as hardware. Therefore, the module called "low-level motion controller" produces necessary motor velocities, which are proportional to the wheel velocities actually, by using the given reference velocity and steering angle of the robot. Then those reference velocities are processed by velocity controllers of each motor, executed, and the outputs that are actual wheel velocities are delivered to the rest of the system. Finally, we can use those velocities to determine velocity, orientation, and position of the agent. The simplified agent model is shown Figure 2.6 below.

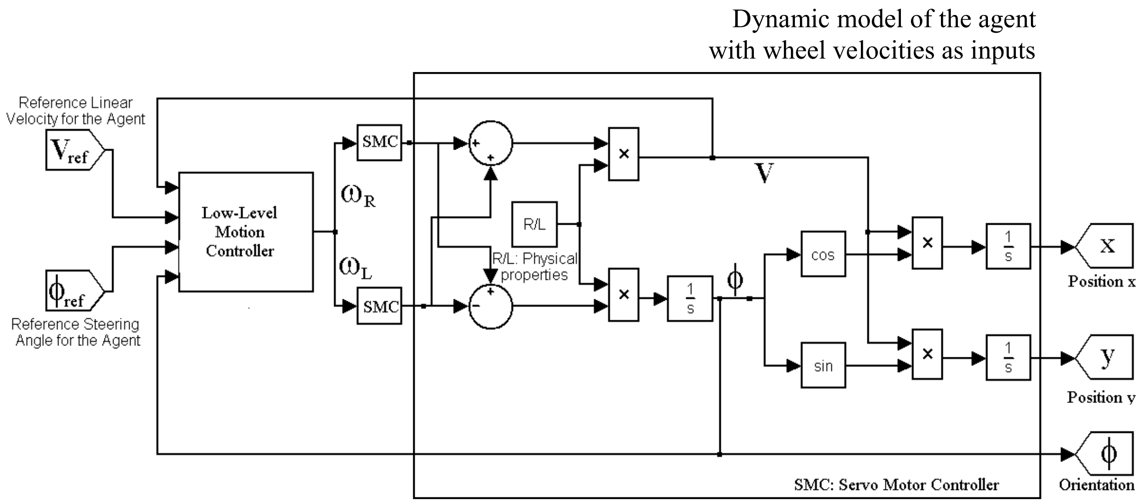

Dynamic model of the agent with wheel velocities as inputs
Reference Linear Velocity for the Agent
V_{ref}
ϕ_{ref}
Reference Steering Angle for the Agent
Low-Level Motion Controller
SMC
ω_R
ω_L
R/L
R/L: Physical properties
V
ϕ
cos
sin
x
Position x
y
Position y
ϕ
Orientation
SMC: Servo Motor Controller

Figure 2.6: Simplified model of the agent. Low-level motion controller and wheel velocity control blocks are also shown.

2.1.1.6 Sensors for Agents

While acting in a world, agents must “feel” their environment. This feeling is produced by the sensors of the agents. Many different sensors can be used in mobile robotics. Those sensors constituted important part in control since they allow us to have feedback control instead of open loop. This way, our agents are not simple, preprogrammed “repeaters” but reactive robots that are showing intelligent behaviors.

In our agents, we have two major types of sensors installed. The first type sensors are for internal usage. Encoders are used to detect the position and/or velocity of the motors. Using this data, we can estimate the position of the robot using kinematics model of the wheel set. The second type of sensors is for detecting external world. Ultrasonic and/or infrared distance measurement sensors are used to detect obstacle positions relative to the robot position.

Many different sensors such as gyros, compasses, resolvers, beacon detectors, camera vision, proximity sensors and other similar sensors can be used to improve both position estimation and obstacle detection. By better sensing the surrounding world, our robot will be given the opportunity to perform better.

On the other hand, some of those sensors may be essential for higher layer tasks. To have a temperature map of the room, we must obviously use a temperature sensor, which has no effect on position control.

2.1.2 Obstacles

Agents are clearly essential components of the plant. However, many different physical objects may also be present in an environment. For practical reasons we are referring all physical object, including other agents, that is out of control for an agent, as obstacle for that specific agent. Obstacles are entities that either are preventing the agent to move or limiting its actions.

Obstacles can be classified into two groups; stationary obstacles and mobile obstacles. Elements of the first group are the physical object that is somehow fixed in the environment. Those obstacles keep their position constant during the whole process time. In the contrary of the first group, elements of the second group do not have fixed position in an environment. A bouncing or a rolling cylinder, a human or an agent are all examples for mobile obstacles.

Physical contact between one or more obstacles and an agent is referred as collision. The control developed for the agent must avoid such contacts. For this purpose many studies are done. Some used the assumption that mobile obstacles are stationary during the discrete time interval. While, others tried to estimate next position of the mobile obstacles to avoid contacts with them. We are currently not concentrating on this specific problem. However, as a basic functionality of the mobile agent we are including simple control to avoid obstacles.

2.2 Method

In this section, methods used in the application of the basic ideas behind the proposed control are presented. The aim is not to cover every detail but to give basic information to better understand the rest of the text.

2.2.1 Variable Structure Systems and Sliding Mode Control

Since very long-time, discontinuous control systems have been studied as a high level in the course of the development of the control system theory. The study of discontinuous dynamical systems is a multifaceted problem, which covers mathematical, control theory and application aspects. As a result, the problem of discontinuous dynamical plant control has been studied by mathematicians, physicists and engineers to solve problems that arise in their own field of interest. However, sometimes, as in power electronics and motion control systems, the switching is a must and therefore discontinuity is explicitly introduced into system.

The discontinuity of control results a discontinuity in the differential equations describing the system motion. If such discontinuities are deliberately introduced on certain manifold in the system state space, the motion in a sliding mode may occur. This type of motion features some attractive properties and has long since been applied in relay system. Sliding modes are the basic motion in Variable Structure Systems (VSS) [Emelyanov 1970, Utkin 1974].

In the sliding mode, trajectories of the state vector belong to the manifold of lower dimension than of the state space. Therefore, the order of differential equations describing motion is reduced. In most of the practical systems, the motion in sliding mode is control-independent and determined by both plant parameters and the equations of the discontinuity surfaces. This allows the initial problem to be decoupled into two problems of lower dimension. The control is used to restrict the motion of the system on the sliding mode subspace. The desired character of the motion over the intersection of discontinuity surfaces is provided by an appropriate selection of the equations of the discontinuity surfaces. Moreover, under certain condition, the system with sliding mode may become invariant to variation of the dynamic properties of the control plant.

The interest to discontinuous control systems is enhanced by their effective application to solution of problems diverse in their physical nature and functional purpose is a basic element in favor of the importance of this area.

2.2.1.1 Sliding Mode in Continuous-Time Systems

Variable Structure Systems are originally defined in continuous-time for dynamic systems described by ordinary differential equations with discontinuity. In such a system, sliding mode motion can occur. This motion is represented by the state trajectories in the sliding mode manifold and high frequency changes in the control. The motion belongs to certain manifold in state space with a lower dimension than that of the system. This fact results order reduction in the motion equations, which enables simplification and decoupling design procedure to be employed. For sliding mode application, the equations of motion and the existence conditions must be solved first.

Further analysis will be restricted to the systems linearly dependent on control

$$\dot{x} = f(x,t) + B(x,t) \cdot u \tag{2.12}$$

where $B(x,t)$ is an $n \times m$ matrix, $x \in \Re^n$, $u \in \Re^m$, $f : \Re^1 \times \Re^n \to \Re^n$. For such a system boundary layer regularization enables substantiation of so called equivalent control method, which is used in deriving the sliding mode equations [Utkin 1981]. In accordance with this method, control in (2.12) should be replaced by the equivalent control, which is the solution to

$$\dot{\sigma} = Gf(x,t) + GB(x,t)u_{eq} = 0, \qquad G = \{\partial\sigma / \partial x\} \tag{2.13}$$

where $\sigma = 0$, $\sigma \in \Re^m$ is defining sliding mode manifold while $\sigma = 0$ describe so-called switching surfaces. For $\det(GB) \neq 0$, equivalent control for system (2.12) in manifold $\sigma = 0$ can be calculated as $u_{eq} = -(GB)^{-1} Gf$ and sliding mode equation is

$$\dot{x} = [I - (GB)^{-1} G] \cdot f \qquad \sigma = 0 \tag{2.14}$$

From $\sigma = 0$, m components ($x_2 \in \Re^m$) of the state vector x may be determined as functions of the rest ($n - m$) components ($x_1 \in \Re^{n-m}$) as $x_2 = \sigma_0(x_1), \sigma_0 \in \Re^m$ and the order of the sliding mode equation 2.13 may be reduced by m :

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, \sigma_0(x_1)) \\ x_2 &= \sigma_0(x_1) \end{aligned} \qquad x_1 \in \Re^{n-m} \tag{2.15}$$

2.2.1.2 Existence and stability of sliding modes

To derive sliding mode existence conditions in analytical form, the projection of the system motion on the manifold

$$S = \{x \in R^n : \sigma(x) = 0\} \qquad \sigma \in \Re^m \tag{2.16}$$

should be analyzed. The conditions of existence of multidimensional sliding modes are in close relation with the convergence of motion toward the manifold S. Hence the solution is based on the methods of the theory of stability, which are applied to the equations of projection of motion onto the subspace S expressed as

$$\dot{\sigma} = Gf(x,t) + GB(x,t)u_{eq} = 0 \qquad G = \frac{\partial \sigma}{\partial x} \tag{2.17}$$

The domain S is the domain of existence of the sliding mode if there exists such a continuously differentiable function $V(\mathbf{x},\sigma,t)$ that [Utkin 1981]:

i) For each x, σ and t that function is defined positively with respect to σ and,

$$\min_{\|\sigma\|=R}(V) = h_R \qquad \max_{\|\sigma\|=R}(V) = H_R \qquad h_R = 0 \quad for \quad R = 0$$

(where h_R and H_R depend only on R)

ii) The time derivative of the function V a negative upper boundary on the set of all points of the sphere $\|\sigma\| = R$, with the exception of the points on the surfaces of discontinuity, where this derivative is undetermined.

Since standard method for obtaining the Lyapunov function for non-linear systems does not exist, for arbitrary matrices in equation 2.17 no standard solution, V, is known.

2.2.1.3 Design

Equations of motion in the sliding mode (2.14) depend on the $m \times n$ elements of matrix G, which consists of the gradients of function σ. Therefore, this motion can be influenced by changing the positions of the switching surfaces within the system state space. The synthesis of control in the systems with sliding modes is performed in the following order; first step consists of the selection of switching surfaces so that the motion in the sliding mode has the required properties. Second step consists of the selection of such control input that the sliding mode exists on the entire domain S. Finally, the third task can be formulated as the finding of conditions for the state to reach the domain S from any initial position.

Considering that the right side of the sliding mode equations (2.14) is continuous, the first stage of the procedure is considered as the task of the synthesis of continuous control. The second and the third stages of the synthesis are near to the task setting, because they consist of providing the stability of the system origin (2.17). It is necessary to consider the method of selecting the controls, which provide the existence of the sliding mode for the intersection of all switching surfaces. The main difficulty in solving the formulated task is the lack of a universal method, which would permit the splitting of the domain S into subdomains in which the sliding mode exists and where this motion does not take place. At the same time, for a special form of the matrix GB it is possible to obtain the sufficient conditions for the existence of the sliding mode. In spite that reduction to such special cases can be done only through an appropriate selection of the matrix G, which has to provide for the required character of the sliding mode, these two tasks can be solved independently.

2.2.2 Sliding Modes in Sampled-Data Systems

Ideal sliding mode, characterized by the motion along the sliding mode manifold can rarely occur in real systems. In continuous time systems, real sliding mode motion is characterized by high frequency oscillations within a boundary layer of the sliding mode manifold, but still averaged motion is kept in the sliding mode manifold. For continuous time, design of the sliding mode with discontinuous control, requires the information about the upper bound of the equivalent control and the position of the system state with respect to the sliding mode manifold (the signs of all components of the switching function vector) [Utkin, 1992].

Due to the hold processes in the control loop and unpredictable changes of the external disturbances in discrete-time systems, the ideal sliding mode can occur rarely. Therefore, like in any discrete-time system, the state can be kept within a boundary layer of the sliding mode manifold. The motion in this boundary layer is accepted as a sliding mode motion [Utkin et al., 1987, Kotta, 1989, Furuta, 1990], and sometimes it is described as so-called quasi-sliding mode [Milosavljevic, 1985, Sira-Ramirez, 1991].

A considerable amount of work has been done analyzing discrete-time sliding modes [Su et al., 1993, Su et al., 1994, Young et al., 1992, Milosavljevic, 1985], studied quasi-sliding in the vicinity of the sliding manifold due to the discritization of continuous time signals. Utkin and Drakunov, 1987, proposed, for control law design, the *discrete-time equivalent control* that directs the state onto sliding mode manifold after a finite number of sampling intervals, due to the boundedness of the control input. The resulting control appears to be non-switching. In most of the other related works, the proposed control strategies use, in one or another way, the calculation or estimation of the discrete-time equivalent control explicitly, which requires the transformation of the plant model into a discrete-time form. That leads to calculation intensive requirements, and it is sensitive to the plant parameters change.

In some applications, like power electronics, switching is the way of life and motion in a boundary layer is unavoidable regardless of the technique used in the control system design. In these systems chattering (often called ripple) is the structural property of the system. In some other systems, like mechanical or process systems, discontinuity of the control action is not so desirable, or easy to achieve, from many points of view such as actuator wearing, excitement of unmodeled dynamics, etc. In these systems, properties that can be achieved by introducing sliding mode motion are very attractive, so a design procedure allowing attain to these properties while avoiding discontinuity of the control action is desirable. Further, the design procedure will be demonstrated in details along with stability proofs for systems without computational delays. The design procedure begins with a selection of the candidate Lyapunov function and the form, which the time derivative of the candidate Lyapunov function should satisfy. From those two selections, the control input is determined. In sampled data systems the satisfaction of the stability conditions is determined when renewed control is applied (usually the beginning of the sampling interval) and at the end of the sampling interval, in order to select the sampling interval and allowed computational delay.

2.2.2.1 Problem statement

Given a continuous nonlinear dynamical system

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x},t) + \mathbf{B}(\mathbf{x},t) \cdot \mathbf{u}(\mathbf{x},t) , \tag{2.18}$$

where all elements of vector $\mathbf{f}(\mathbf{x},t)$ and matrix $\mathbf{B}(\mathbf{x},t)$ are continuous and bounded having continuous and bounded first order time derivatives; $rank(\mathbf{B}(\mathbf{x},t)) = m$, $\forall \mathbf{x}, t > 0$; all components of the control input $\mathbf{u}(\mathbf{x},t)$ are bounded by known functions $u(\mathbf{x},t)_{i\max}$ and $u(\mathbf{x},t)_{i\min}$, $i = 1,2,...,m$.

Assume that the desired specification of closed loop system is achieved if the system state satisfies algebraic constraint $\sigma(\mathbf{x},t) = 0$, $\sigma \in \Re^m$ where all components of function $\sigma(\mathbf{x},t)$ are continuous. Then, the design goal is to stabilize the system motion on the smooth manifold

$$\mathbf{S} = \left\{ \mathbf{x} \in R^n : \sigma(\mathbf{x},t) = \mathbf{0}; \sigma \in R^m ; \mathbf{G}(\mathbf{x},t) = \frac{\partial \sigma}{\partial \mathbf{x}} \right\} \tag{2.19}$$

where all elements of vector $\partial\sigma/\partial t$ and matrix $\mathbf{G}(\mathbf{x},t)$ are continuous and bounded and $rank(\mathbf{G}(\mathbf{x},t)) = m \ \forall \mathbf{x}, t > 0$.

2.2.2.2. Design

For system (2.18), asymptotic stability of the solution $\sigma(\mathbf{x},t) = \mathbf{0}$ can be assured if one can find a control input such that the Lyapunov stability criteria are satisfied. Natural selection of the candidate Lyapunov function is a quadratic form $v = \sigma^T\sigma/2$. The design procedure can be started from the requirement that the time derivative of the Lyapunov function should have the following form $dv/dt = -\sigma^T \mathbf{D}\sigma$, $\mathbf{D} > 0$. Then

$v(\mathbf{x},t) > 0$, $dv(\mathbf{x},t)/dt < 0$, $\forall \mathbf{x} \notin \mathbf{S}$, $v(\mathbf{x}^*,t) = 0$, $dv(\mathbf{x}^*,t)/dt = 0$, $\mathbf{x}^* \in \mathbf{S}$, $\forall t > 0$ and solution $\sigma(\mathbf{x},t) = 0$ is asymptotically stable on the trajectories of the system (2.18). A control input that satisfies given requirements can be calculated from

$$\begin{aligned} &\sigma^T \frac{d\sigma}{dt} = -\sigma^T D\sigma \\ &\Rightarrow \frac{d\sigma}{dt} + D\sigma = 0 \end{aligned}, \qquad \forall \sigma \neq 0 \tag{2.20}$$

By substituting $\frac{d\sigma}{dt} = \frac{\partial \sigma}{\partial t} + Gf + GBu$ into $\frac{d\sigma}{dt} + D\sigma = 0$ and by solving this equation for an unknown control u, it is easy to obtain

$$\mathbf{u}(\mathbf{x},t) = -(\mathbf{GB})^{-1}\left(\frac{\partial \sigma(\mathbf{x},t)}{\partial t} + \mathbf{Gf}\right) - (\mathbf{GB})^{-1}\mathbf{D}\sigma \tag{2.21}$$

where $\mathbf{GB}$ is a regular matrix

Selected control input guaranties that the motion of system (2.18) satisfies the dynamical constrain $\frac{d\sigma}{dt} + D\sigma = 0$. That means all distances from the manifold $\mathbf{S}$ exponentially tend to zero and the system motion will remain in ε-vicinity of the manifold $\mathbf{S}$ after reaching it. Strictly speaking, control (2.20), being continuous, does not provide the sliding mode motion on manifold (2.19) because that manifold could be reached only for $t \to \infty$. Taking into account that the equivalent control, required to keep the motion on the manifold S if initial state belongs to this manifold, can be expressed as $u_{eq} = -(GB)^{-1}(\frac{\partial \sigma}{\partial t} + Gf)$, equation 2.20 can be rewritten as

$$\mathbf{u}(\mathbf{x},t) = \mathbf{u}_{eq}(\mathbf{x},t) - (\mathbf{GB})^{-1}\mathbf{D}\sigma \tag{2.22}$$

In (2.22) the resulting control action is continuous (the equivalent control is continuous and function $\sigma(\mathbf{x},t)$ is continuous by assumption) and $\mathbf{u}(\mathbf{x},t) = \mathbf{u}_{eq}(\mathbf{x},t)$ for $\sigma(\mathbf{x},t) = 0$. In the implementation of algorithms (2.21) or (2.22) full information about system dynamics and external disturbances is required (for equivalent control calculation). Because of this, these algorithms are not practical for application. They are used here as intermediate results to show the procedure in the development of simpler and more useful control strategies. From $\frac{d\sigma}{dt} = GB(\mathbf{u}(\mathbf{x},t) - \mathbf{u}_{eq}(\mathbf{x},t))$ equivalent control can be substituted into (2.22) to obtain

$$\mathbf{u}(\mathbf{x},t) \equiv \mathbf{u}(\mathbf{x},t) - (\mathbf{GB})^{-1}\left(\mathbf{D}\sigma(\mathbf{x},t) + \frac{d\sigma(x,t)}{dt}\right) \qquad (2.23)$$

This form of expressing the control input is very instructive. It shows that in order to force the system to reach ε-vicinity of sliding mode manifold (2.19) and to stay within ε boundary layer the control input should be modified by the term $(\mathbf{GB})^{-1}\left(\mathbf{D}\sigma(\mathbf{x},t) + \frac{d\sigma(x,t)}{dt}\right)$ at every instant of time. The control (2.23) takes the value of the equivalent control for $\sigma(\mathbf{x},t) = 0$.

2.2.2.1.1 Discrete-time realization

The procedure for discrete-time sliding mode design begins with a transformation of the plant description to the discrete-time form $\mathbf{x}(kT+T) = \mathbf{x}(kT) + \mathbf{f}^* + \mathbf{B}^*\mathbf{u}(kT)$ where T is sampling interval, $\mathbf{f}^* = \int_{kT}^{kT+T} \mathbf{f}(x,\tau)d\tau$; $\mathbf{B}^* = \int_{kT}^{kT+T} \mathbf{B}(x,\tau)d\tau$. Then the discrete-time equivalent control can be calculated from $\sigma(kT+T) = \mathbf{Gx}(kT+T) = 0$ as $\mathbf{u}_{eq}(kT) = -(\mathbf{GB}^*)^{-1}(\mathbf{Gx}(kT) + \mathbf{Gf}^*)$. It should be noted that in this case no computational delay is taken into account. The intersampling change for equivalent control is of $O(T)$ order. The equivalent control tends to infinity if $T \to 0$ for $\sigma \neq 0$, since $(\mathbf{GB}^*)^{-1} \to \infty$ while $(\mathbf{Gx}(kT) + \mathbf{Gf}^*)$ takes a finite value. That requires the introduction of limits for the control action. Taking into account that, control action is bounded by assumption, the control algorithm can be expressed as $\mathbf{u}(\mathrm{kT}) = \mathrm{sat}(\mathbf{u}_{\mathrm{eq}}(\mathrm{kT}))$ where $\min(\mathrm{sat}(\bullet)) = u_{\min}$ and $\max(\mathrm{sat}(\bullet)) = u_{\max}$. It has been proven [Su et al., 1993] that the selected control will force system state to stay in ε-vicinity of sliding mode manifold (2.19) with a thickness of the boundary layer of $O(T)$ order.

To avoid cumbersome explanations in discrete-time control systems, from now on, term “continuous control“ is used in the sense that intersampling change of the control input is of $O(T)$ order. Similarly, for the discrete-time systems, the motion within in ε-vicinity of sliding mode manifold (2.19) with a thickness of the boundary layer of $O(T)$ order will be accepted as the sliding mode motion in the sliding mode manifold.

Algorithm (2.22) can be easily modified for the application in the discrete time systems with no computational delay. In such a system, relations between measured and computed variables are depicted in Figure 2.7, where measurement taken before the calculation of new value of the control input are denoted as $\bullet(kT^{-})$ and all variables immediately after new value of the control input is applied are denoted by $\bullet(kT^{+})$, (from now on all variables will be written shorter so $\sigma(kT)$ means $\sigma(x(kT),kT)$. Note that all continuous functions and variables satisfy $\bullet(kT^{-}) = \bullet(kT^{+})$.

By taking into account the relationship depicted in Figure 2.7, algorithm (2.22) can be rewritten in the following form

$$\mathbf{u}(kT^{+}) = \mathbf{u}(kT^{-}) - (\mathbf{GB})^{-1}(\mathbf{D}\sigma(kT^{-}) + \frac{d\sigma(kT^{-})}{dt}) \tag{2.24}$$

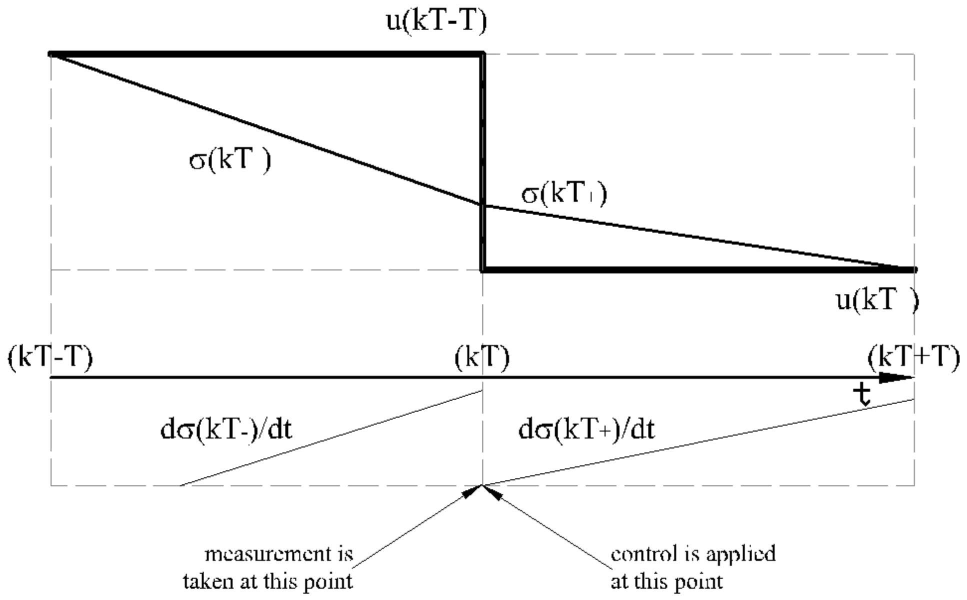

u(kT-T)
σ(kT)
σ(kT+)
u(kT)
(kT-T)
(kT)
(kT+T)
t
dσ(kT-)/dt
dσ(kT+)/dt
measurement is taken at this point
control is applied at this point

Figure 2.7: The relations between measured and calculated variables for discrete time systems without computational delay.

The satiability conditions should be analyzed at the moment the control input is applied to the system and at the end of the corresponding sampling interval. The value of the switching function and its derivative at k^{th} sampling interval e.g. at $t = kT$ and $t = kT + T$ should be determined and value of the selected Lyapunov function and its derivative should be calculated at both instants. By calculating, the derivative of function $\sigma(\mathbf{x},t)$ at $t = kT^+$ one can find

$$\frac{d\sigma(kT^+)}{dt} = \frac{\partial\sigma(kT^+)}{\partial t} + \mathbf{G}(\mathbf{f}(kT^+) + \mathbf{Bu}(kT^+)) \tag{2.25}$$

From (2.24) and (2.25) it follows

$$\frac{d\sigma(kT^+)}{dt} = [\frac{\partial\sigma(kT^-)}{\partial t} + \mathbf{G}(\mathbf{f}(kT^-) + \mathbf{Bu}(kT^-))] - \mathbf{D}\sigma(kT^-) - \frac{d\sigma(kT^-)}{dt}$$

and finally

$$\frac{d\sigma(kT^+)}{dt} = \frac{d\sigma(kT^-)}{dt} - \mathbf{D}\sigma(kT^-) - \frac{d\sigma(kT^-)}{dt} = -\mathbf{D}\sigma(kT^-) = -\mathbf{D}\sigma(kT^+) \tag{2.26}$$

The time derivative of the Lyapunov function, at $\mathrm{t = kT^+}$, can be expressed as $dv(kT^+)/dt = -\sigma(kT^+)^T \mathbf{D}\sigma(kT^+)$. This shows that at the moment immediately after new control is applied (the beginning of the sampling interval) the stability conditions are satisfied.

For $\mathrm{t} \in [kT, kT + T]$ the control input is constant. The change of the system state during an intersampling interval can be expressed as

$$\begin{aligned} \mathbf{x}(kT+t) &= \mathbf{x}(kT) + \int_{kT}^{kT+t} \mathbf{f}(x,\tau)d\tau + \left[\int_{kT}^{kT+t} \mathbf{B}(x,\tau)d\tau\right]\mathbf{u}(kT) \\ \mathbf{x}(kT+t) &= \mathbf{x}(kT) + \xi(t); \end{aligned} \tag{2.27}$$

and the change of the distance from the sliding mode manifold (2.19) is

$$
\begin{aligned}
\sigma(kT+t) &= \sigma(kT) + \int_{kT}^{kT+t} \mathbf{Gf}(x,\tau)d\tau + \left[\int_{kT}^{kT+t} \mathbf{GB}(x,\tau)d\tau\right]\mathbf{u}(kT) \\
\sigma(kT+t) &= \sigma(kT) + \varsigma(t)
\end{aligned}
\tag{2.28}
$$

by the assumption $\mathbf{f}$, $\mathbf{B}$ and $\mathbf{G}$ are continuous and bounded. The maximum change of state vector $\xi(t)$, and of distances from the sliding mode manifolds $\varsigma(t)$, inside the interval $\mathrm{t} \in [kT, kT+T]$ are of O(T) order. The change of the time derivative of the sliding mode function can be determined as

$$
\frac{d\sigma(kT+t)}{dt} = \mathbf{Gf}(kT+t) + \mathbf{GBu}(kT+t) + \frac{\partial\sigma(kT+t)}{\partial t}
\tag{2.29}
$$

2.2.3 Layered Structure and Behavioral Arbitration

Earliest approach for mobile robot control design was model based. In this approach, sensory data is evaluated to modify the model of the environment where the agent is acting. Then agent must prepare a plan before acting. While planning, agent assumes that the part of the environment that is not accessible by its sensors stays constant or unchanged. Finally, agent executes its plan until a difference between the model and the environment is detected. And then this loops runs infinitely.

This type of control needs to know everything on the environment before the agent is put in. However, in wide and crowded areas this is not feasible. Moreover, agents must be well thought before updating the model in its memory. As an example, a box that is preventing him to reach a certain space may cause an update in the model and consequently newer plan to execute. However, the same box may be removed ten seconds later. Similarly, if all moving obstacles are marked as stationary obstacles, each time perceived, in the model, then the agent will probably not find any place to move.

All those practical problems forced researchers to move a newer control that does not need to model and plan. Suggested first by Brooks [Brooks 1985], a new approach uses “reactions” to the instantaneous data coming from the environment (obtained by sensors) to obtain reasonable movements for the robot. By the way, rather than preparing a complex control for the robot, many independent and relatively small control layers that can run in parallel are designed. This new approach, reactive control, is actually based on an idea similar to the reflexes that exists in living organisms. A human or an animal that touches a hot surface quickly moves, without thinking, to break the contact. Similarly, in Brooks implementation the robot reacts to the obstacles that are sensible to his sensors.

However, while the robot senses an obstacle, it continuously reacts to it without thinking. Moreover, agent that is moving under the influence of the obstacles may be trapped easily between obstacles since it is continuously seeking for a free space rather than moving to the target point for example. This problem, bring the necessity to select the correct action that will be executed at that time. Or preferably combine actions to serve all of the layers at a time. Many researchers applied different techniques to solve this problem referred as action selection or behavior arbitration. However, at our best knowledge most of them preferred to alter wheel velocities directly using the outputs of the layers. Consequently, the opportunity to fuse different behaviors to serve more than one layer is lost.

In the proposed approach, we are never acting of the physical entities such as motors, motor drivers and similar entities. Nevertheless, we are changing the reference values of the low-level motion controller that deals with the control of physical entities. Those are velocity and steering direction of the mobile robot. Moreover, each layer rather than giving its own references for the low-level motion controller, return values that are representing deviations in those values. By the way, we can weight each layers outputs according to a constant or dynamical importance. Finally, add the resultant values to the current velocity and steering direction to obtain new reference values.

3. SOLUTION: AN AGENT INSTANCE

In our simulations, we created many identical agents to simulate a multi agent system. Physically all of them are wheel set type differential drive vehicles, and are equipped with the same sensor configuration. All of them are and totipotent, possesses all the necessary skills to realize given tasks.

As stated earlier we implemented a layered structure in the design of the control for the agents. Accordingly, our agents are formed by "competence layers". Those layers, individually performing their own tasks, produce an output. The output of each layer is the reference for the low-level motion controller, which is implemented as a sliding mode controller that generated necessary wheel velocities using velocity and steering direction reference.

Figure 3.1 shows the main structure of the agents. Details of each layer are studied in the following sections.

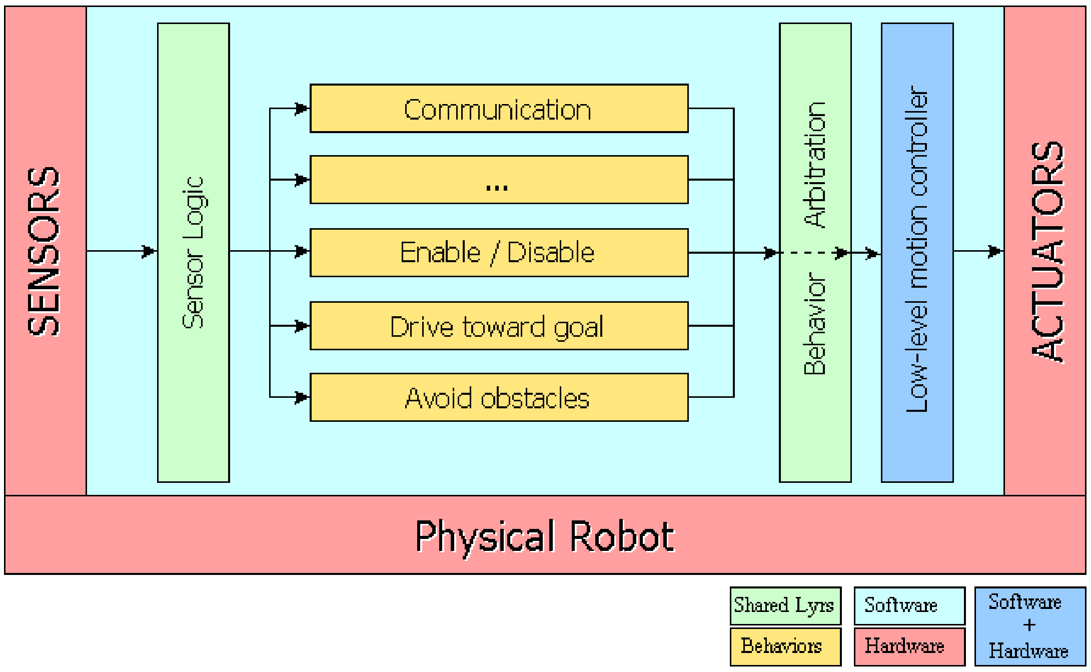


Figure 3.1: Structure of our agents, which are designed according to the behavior-based approach.

3.1 Low-Level Motion Controller and Physical Agent

Physically our agents are formed by two driving wheels aligned on the same line passing through the center of the circular body, and one or more caster wheels to keep the balance. This configuration, differential drive or wheel set, is well known example of nonholonomic robot. Since assuming no slip at tires, the motion of each wheel is restricted to its longitudinal direction with velocities v_R and v_L, respectively, by a single nonholonomic constraint. In other words, no motion can occur along the lateral robot coordinate axis y_R.

As shown in Figure 2.2, the robot configuration in the world coordinate frame (x_w, y_w) is $q = (x, y, \phi) \in \Re^3$, which covers both position and the orientation of the vehicle. Control inputs are right and left wheel velocities v_R and v_L respectively, which may be translated into the translational and rotational velocity variables, v and ω respectively, for convenience. The forward kinematics is already found. Moreover, simplified model of the agent is shown as a diagram in Figure 2.6, together with low-level motion controller.

The low-level motion controller is the part where the reference velocity and direction information coming from higher levels is converted to reference wheel velocities for servo motor controllers. First, reference velocity, v_{ref}, must be between minimum and maximum velocities, $v_{\min}$ and $v_{\max}$ respectively, permitted by the physical properties of the robot, such as the maximum speed of the driving motors, limits of the gearboxes if used, and size of the wheels. Similarly, we can trap reference orientation between 0 and 2π, however, such a modification may cause unexpected results in the control. For example, such a change may create a sign difference between reference and actual values, and then cause turnery motion.

$$v_{ref} = \min(v_{\max}, \max(v_{\min}, v_{ref})) \tag{3.1}$$

Then reference velocity and orientation of the robot are compared with the actual velocity and orientation to determine errors.

$$\begin{aligned} e_v &= v_{ref} - v \\ e_\phi &= \phi_{ref} - \phi \end{aligned} \tag{3.2}$$

From the measurement of angular velocities of left (ω_L) and right (ω_R) wheels, actual position and orientation of the robot can be determined easily,

$$\begin{aligned} x &= \frac{R}{2} \cdot \int_{\tau=0}^{t} (\omega_R + \omega_L) \cdot \cos(\phi) \cdot d\tau \\ y &= \frac{R}{2} \cdot \int_{\tau=0}^{t} (\omega_R + \omega_L) \cdot \sin(\phi) \cdot d\tau \\ \phi &= \frac{R}{L} \cdot \int_{\tau=0}^{t} (\omega_R - \omega_L) \cdot d\tau \end{aligned} \tag{3.3}$$

Moreover, with the information of the reference velocity and orientation, reference position of the robot can be obtained,

$$\begin{aligned} x_{ref} &= x + v_{ref} \cdot \cos(\phi_{ref}) \cdot dt \\ y_{ref} &= y + v_{ref} \cdot \sin(\phi_{ref}) \cdot dt \end{aligned} \tag{3.4}$$

In those two equations, we are using actual position of the robot (x, y) together with the reference velocity and orientation to determine the desired next position of the robot. Obviously, the control should be selected to keep the position error equal to zero, while changing the angular velocities of the powered wheels. The position errors are then defined as,

$$\begin{aligned} e_x &= x_{ref} - x \\ e_y &= y_{ref} - y \end{aligned} \tag{3.5}$$

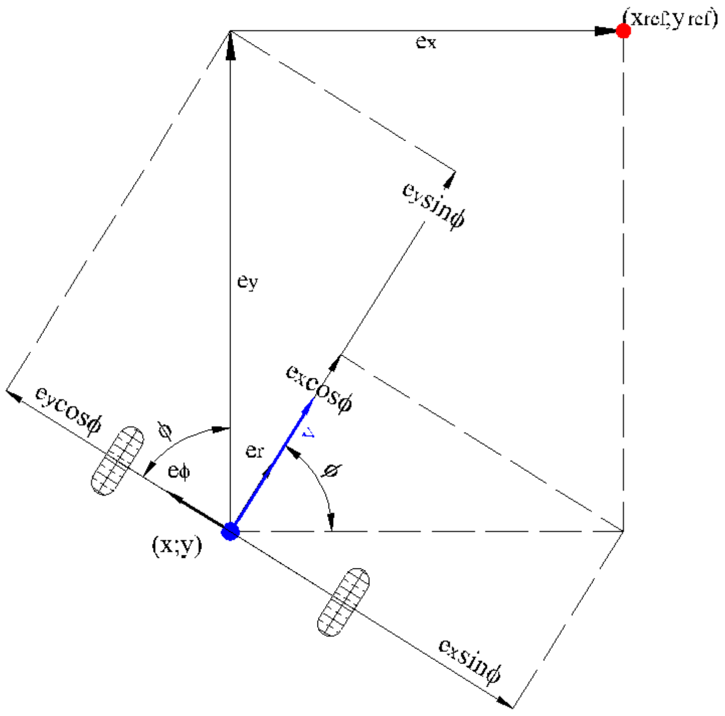


Figure 3.2: Projection of position errors e_x and e_y onto the velocity and its perpendicular direction.

We can use the geometry shown in Figure 3.2 to project those two position errors on to the axes: one that is collinear with velocity (referred as velocity direction, but denoted with subscript r rather than v to prevent conflicts) and the other that is perpendicular to it (phi direction or steering direction, noted with subscript ϕ). Note that both of those errors have the dimensions of distance.

$$\begin{aligned} e_r &= e_x \cdot \cos(\phi) + e_y \cdot \sin(\phi) \\ e_\phi &= -e_x \cdot \sin(\phi) + e_y \cdot \cos(\phi) \end{aligned} \tag{3.6}$$

Knowing the errors in velocity and steering direction we can calculate "corrected" values for the reference values, which will constitute reference for the rest of the system.

$$\begin{aligned} r_{ref}^{corr} &= r_{ref} + e_r \\ \phi_{ref}^{corr} &= \phi_{ref} + e_\phi \end{aligned} \tag{3.7}$$

where r_{ref} and r_{ref}^{corr} are the distances from the origin of the world coordinate frame to the robot's actual reference and corrected reference position respectively. Now we can calculate how far are our real values from those new references,

$$\begin{aligned} e_r^{corr} &= r_{ref}^{corr} - r \\ e_\phi^{corr} &= \phi_{ref}^{corr} - \phi \end{aligned} \tag{3.8}$$

which are actually errors in terms of distances, that are compensating both present and past velocity, direction and their control errors.

Now having the actual values and related errors, in velocity and orientation control, classical sliding mode controller can be used. For this purpose the following state vectors are used together with the state space equation,

$$\begin{aligned} \dot{x} &= v \cdot \cos\phi \\ \dot{y} &= v \cdot \sin\phi \\ \dot{\phi} &= \omega \end{aligned} \tag{3.9}$$

Or in terms of previously found wheel velocities this equation can be rewritten as,

$$\begin{aligned} \dot{x} &= \left(\frac{R}{2} \cdot (\omega_R + \omega_L)\right) \cdot \cos\phi \\ \dot{y} &= \left(\frac{R}{2} \cdot (\omega_R + \omega_L)\right) \cdot \sin\phi \\ \dot{\phi} &= \frac{R}{L} \cdot (\omega_R - \omega_L) \end{aligned} \tag{3.10}$$

Instead of choosing ω_R and ω_L as control inputs, we can select

$$\begin{aligned} u_1 &= \omega_R + \omega_L \\ u_2 &= \omega_R - \omega_L \end{aligned} \tag{3.11}$$

for simplicity. Using the given transformation above we obtain,

$$\begin{aligned}\dot{x} &= \frac{R}{2}\cdot u_1 \cdot \cos\phi \\ \dot{y} &= \frac{R}{2}\cdot u_1 \cdot \sin\phi \\ \dot{\phi} &= \frac{R}{L}\cdot u_2\end{aligned} \tag{3.12}$$

Note that first element of the control, u_1, is proportional to the velocity while second one, u_2, is proportional to the velocity of orientation change, $\dot{\phi}$. We can always combine first two states using polar coordinates where r denotes the distance to the origin of the world coordinate frame.

$$\begin{aligned}\dot{r} &= \sqrt{\dot{x}^2 + \dot{y}^2} = \frac{R}{2}\cdot u_1 \\ \dot{\phi} &= \frac{R}{L}\cdot u_2\end{aligned} \tag{3.13}$$

The first term above is the velocity of the vehicle and the second one is the velocity in the orientation change, which are both invariant under this transformation. Note that, although the above equations are independent from each other, both can be written in a matrix form that has the form of

$$\frac{dx}{dt} = f(x) + B\cdot u \tag{3.14}$$

where

$$\begin{aligned}x &= \begin{bmatrix} r \\ \phi \end{bmatrix} \\ f(x) &= 0 \\ B &= \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} R/2 & 0 \\ 0 & R/L \end{bmatrix} \\ u &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\end{aligned} \tag{3.15}$$

Now, using new controls, u_1 and u_2, together with the corresponding references, v_{ref}^{corr} and ϕ_{ref}^{corr} respectively, we can define the sliding manifolds as follows,

$$\begin{aligned} \sigma_v &= e_v^{corr} \\ \sigma_\phi &= e_\phi^{corr} \end{aligned} \tag{3.16}$$

since our aim is obviously converge errors e_v^{corr} and e_ϕ^{corr} to zero. The control should be chosen such that the positive definite candidate Lyapunov function

$$\gamma = \frac{\sigma^T \sigma}{2} \geq 0 \tag{3.17}$$

satisfies Lyapunov stability criteria. Since both equations are independent, we can use componentwise control, where components of the sliding surface are separately controlled to tend to zero, rather that vector control, where components of the sliding surface are dealt together. Separating the components of γ and taking its time derivative, we obtain,

$$\dot{\gamma}_i = \sigma_i \dot{\sigma}_i = -D_i \cdot \sigma_i^2 \qquad\qquad i = r, \phi \tag{3.18}$$

where components of the Lyapunov candidate function, γ, are all positive definite, and their time derivatives, $\dot{\gamma}$, are negative definite for some constant $D > 0$. Now by taking σ_i in a common factor,

$$\sigma_i \left(\dot{\sigma}_i - D_i \cdot \sigma_i \right) = 0 \qquad\qquad i = r, \phi \tag{3.19}$$

In the above equation either σ_i or $\left(\dot{\sigma}_i - D_i \cdot \sigma_i \right)$ is zero. In the first case, we don't have information about the second part of the equation, which means that we don't know the behavior of the sliding manifold σ_i. However, if the second part is zero, then obviously σ_i will tend to zero. We can now explicitly write those equations,

$$
\begin{aligned}
\dot{e}_v^{corr} &= -D_r \cdot e_r^{corr} \Rightarrow \dot{r}_{ref}^{corr} - \dot{r} = -D_r \cdot e_r^{corr} \\
\dot{e}_\phi^{corr} &= -D_\phi \cdot e_\phi^{corr} \Rightarrow \dot{\phi}_{ref}^{corr} - \dot{\phi} = -D_\phi \cdot e_\phi^{corr}
\end{aligned} \tag{3.20}
$$

We can replace both $\dot{r}$ and $\dot{v}$ using state space equation (3.13),

$$
\begin{aligned}
\dot{r}_{ref}^{corr} - b_{11} \cdot u_1 - f_r + D_r \cdot e_r^{corr} &= 0 \\
\dot{\phi}_{ref}^{corr} - b_{22} \cdot u_2 - f_\phi + D_\phi \cdot e_\phi^{corr} &= 0
\end{aligned} \tag{3.21}
$$

remembering that $b_{11} = \frac{R}{2}$, and $b_{22} = \frac{R}{L}$. Now controls u_1 and u_2 can be obtained,

$$
\begin{aligned}
u_1 &= \frac{1}{b_{11}} \cdot \left(\dot{r}_{ref}^{corr} - f_r + D_r \cdot e_r^{corr} \right) \\
u_2 &= \frac{1}{b_{22}} \cdot \left(\dot{\phi}_{ref}^{corr} - f_\phi + D_\phi \cdot e_\phi^{corr} \right)
\end{aligned} \tag{3.22}
$$

Once more using state space equation to replace f_r and f_ϕ,

$$
\begin{aligned}
u_1 &\equiv \frac{1}{b_{11}} \cdot \left(\dot{r}_{ref}^{corr} - \left(\dot{r} - b_{11} \cdot u_1 \right) + D_r \cdot e_r^{corr} \right) \\
u_2 &\equiv \frac{1}{b_{22}} \cdot \left(\dot{\phi}_{ref}^{corr} - \left(\dot{\phi} - b_{22} \cdot u_2 \right) + D_\phi \cdot e_\phi^{corr} \right)
\end{aligned} \tag{3.23}
$$

which can be simplified and reordered as,

$$
\begin{aligned}
u_1 &\equiv \frac{1}{b_{11}} \cdot \left(b_{11} \cdot u_1 + D_r \cdot e_r^{corr} + \dot{e}_r^{corr} \right) \\
u_2 &\equiv \frac{1}{b_{22}} \cdot \left(b_{22} \cdot u_2 + D_\phi \cdot e_\phi^{corr} + \dot{e}_\phi^{corr} \right)
\end{aligned} \tag{3.24}
$$

Here rather than using equality, identical sign is used since both sides of the equation have u_1 or u_2 and those control are not related with the future but past control. In a discrete time system, we can use this property as follows,

$$
\begin{aligned}
u_1(k \cdot dt) &= u_1((k-1) \cdot dt) + \left(D_r \cdot e_r^{corr} + \dot{e}_r^{corr}\right) \\
u_2(k \cdot dt) &= u_2((k-1) \cdot dt) + \left(D_\phi \cdot e_\phi^{corr} + \dot{e}_\phi^{corr}\right)
\end{aligned} \tag{3.25}
$$

which can further be evaluated as,

$$
\begin{aligned}
u_1(k \cdot dt) &= u_1((k-1) \cdot dt) + \frac{1}{dt \cdot b_{11}} \cdot \left[dt \cdot D_r \cdot e_r^{corr}(k \cdot dt) + e_r^{corr}(k \cdot dt) - e_r^{corr}((k-1) \cdot dt)\right] \\
u_2(k \cdot dt) &= u_2((k-1) \cdot dt) + \frac{1}{dt \cdot b_{22}} \cdot \left[dt \cdot D_\phi \cdot e_\phi^{corr}(k \cdot dt) + e_\phi^{corr}(k \cdot dt) - e_\phi^{corr}((k-1) \cdot dt)\right]
\end{aligned} \tag{3.26}
$$

where dt stands for discrete time interval, and k denotes the k^{th} time interval. Clearly, $k \cdot dt$ is the current time and $(k-1) \cdot dt$ represents the previous time interval.

Finally, actual references for the right and left wheel velocities that will be used by servo controllers are found as,

$$
\begin{aligned}
\omega_R^{ref} &= \frac{u_1 + u_2}{2} \\
\omega_L^{ref} &= \frac{u_2 - u_1}{2}
\end{aligned} \tag{3.27}
$$

Moreover, those values may be limited by the maximum and minimum permitted reference values of the servo motor driver if exists.

$$
\begin{aligned}
\omega_R^{ref} &= \max\left(\omega_{\min}^{ref}, \min\left(\omega_{\max}^{ref}, \frac{u_1 + u_2}{2}\right)\right) \\
\omega_L^{ref} &= \max\left(\omega_{\min}^{ref}, \min\left(\omega_{\max}^{ref}, \frac{u_1 - u_2}{2}\right)\right)
\end{aligned} \tag{3.28}
$$

where $\omega_{\min}^{ref}$ and $\omega_{\max}^{ref}$ denotes minimum and maximum possible reference values respectively.

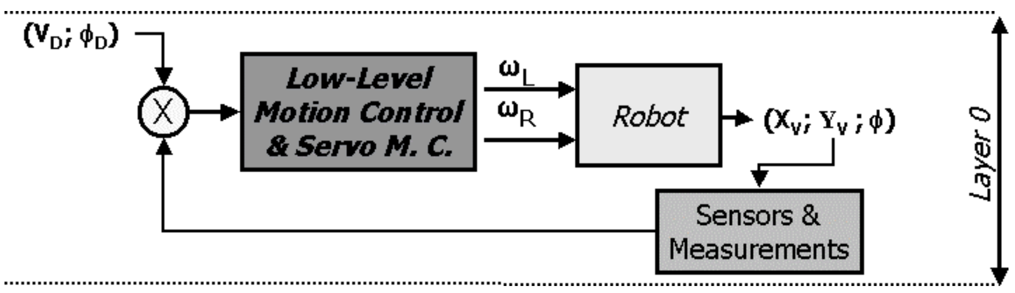


Figure 3.3: 0th level of the agent: position controller via feedback.

Now we can combine all those calculations under a block called "low level motion controller". Finally, schematize all those basic calculations as shown in Figure 3.3 above.

3.2 Obstacle Avoidance

Obviously one of the most wanted features in a mobile robot is the avoidance of both stationary and mobile obstacles. Although the problem is very old and very clear by means of the definition and requirements, no perfect (or unique) solution is found until now. Many different ideas are applied since the existence of the mobile robotics. Each idea has its own requirements depending on the control algorithm's inputs and outputs. Those are generally sensory request since before avoiding an obstacle one must at least know that there is something over there.

Between many possible alternatives, we selected potential field method and applied it in a suitable manner. The idea is simple; obstacles in the environment exert a repulsive force on the robot. Then the robot moves such that the force can be kept minimum, which ensures the clearance of the robot path.

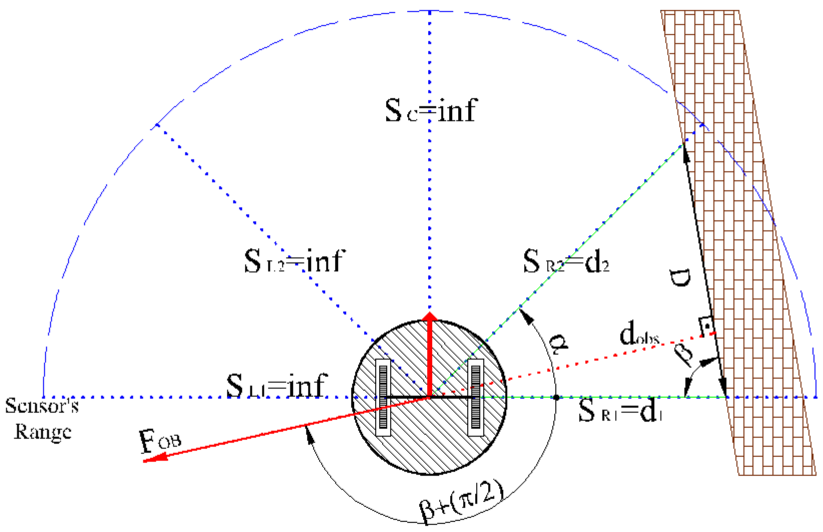
If for each point in the space, the force exerted by the obstacles is calculated, and show in $\hat{z}$ direction, the result will be a potential field map, where lowest potential field will ensure the clearest points by means of obstacles. If a robot move in this map, or in such an environment, by following lowest possible potential field lines, it will surely avoid obstacles. However, this do not ensure that the robot will travel to the requested point, since the robot is moving along the decreasing potential field only, which may not be in the direction of the goal point. With the addition of new forces which attracts the robot toward goal point, one may modify the potential field map such that while moving toward the goal point the potential field is either decreasing or is constant.

Clearly, the first step in the applied obstacle avoidance method is the determination of the repulsive force. Since agents must first detect the obstacles, we equipped them with many sensors placed around the body in a suitable way. Using measurements from those sensors one can calculate both the distance from the obstacle to the body of the agent and the direction perpendicular to the surface of the obstacle. Although some restrictions apply, we can assume that the same information can be gathered for both stationary and mobile obstacles, since a mobile obstacle can be assumed as stationary within a small instance of time. Then this information is used in building different repulsive force vectors that at the end will be summed up to form the composite repulsive force vector for that specific position and orientation of the robot. Obviously, the force experienced by the robot is orientation dependent since we assume that we have sensors placed at known angles, which may not cover the whole area around the robot.

In the simulations, we are using a simple sensor algorithm simulating many infrared sensors that has very narrow conical scanning areas with minimum and maximum measuring ranges defined. However, we are not exactly simulating real sensor readings that cover axial errors and noise. Instead, we place enough sensors around the robot, such that conical scanning areas of the sensors touches each neighbor sensor in order to cover whole area around the robot. This can also be assumed as having many laser range finders placed homogenously around the robot. Moreover, we are not using any sensor logic that will combine all the sensor information with correcting some of the known defects of the sensors but we assume that we have such a logic that works. Although those ideas are not applied in the simulations, in the following text we show that this sort of logic can be easily built.

The simplest solution to repulsive force calculation using distance information would be direct usage of a force with magnitude inversely proportional to the square of the distance measured by each sensor. Since the angles at which, sensor’s scanning axis are placed, are also known, directions of the force vectors are automatically known. Then one can calculate the sum of those vectors. In this approach when obstacles get closer, they will be detected by more and more sensors at the same time, such that the force caused by the same object when its closer will be much more strong since it is amplified two times. First amplification is done by inverse square proportionality, and second by the addition of more forces caused by the same obstacle but sensed by more sensors.

Another approach would be finding the distances using measurements from two neighbor sensors. Using the information from each consecutive sensor pair, we can define a wall type flat obstacle in the angular section limited by those two sensors axis. In addition, a vector perpendicular to that surface can be defined. An example configuration is given in Figure 3.4 and studied below.



Cosine theorem: $D^2 = d_1^2 + d_2^2 - 2 \cdot d_1 \cdot d_2 \cdot \cos\alpha$

Sine theorem: $\dfrac{D}{\sin\alpha} = \dfrac{d_1}{\sin(\pi - \beta - \alpha)} = \dfrac{d_2}{\sin\beta}$

Figure 3.4: An example of sensor configuration and the repulsive force obtained for an obstacle. How to calculate the force vector perpendicular to the obstacle is also shown.

As seen on the above figure we assume five infrared distance sensors placed around the agent with 45 degrees steps (axis of the conical scanning area are shown with dotted lines). Each having the same maximum sensing range, shown by dashed arc. In the given configuration, sensors at the left of the body, S_{L1} and S_{L2}, and sensor at the heading direction, S_C, returns infinity value since there is no obstacle in their detection range. However, sensors placed at the right of the body, S_{R1} and S_{R2}, return values d_1 and d_2 respectively, which are the distances to the obstacle in their range. Without having the knowledge of if it is the same obstacle or not, we can consider single, block type obstacle.

This assumption may create a problem in some cases. For example, imagine a wall with a door opening. And image that one sensor measures the distance to the wall at the left-hand side of the door, while its neighbor sensor is measuring distance to the wall on the other side. Then the robot will not feel the door opening, which may cause it to miss a possible optimal path. However, if we have enough sensors, with very narrow scanning area (such as laser range finders), then the sectors that can be covered by those sensors will intersect and form a complete circular area around the robot. By the way, the robot will not miss such relatively narrow cavities that it can use as a passage. Moreover, if those cavities are so narrow that the robot cannot pass through then the sides will create enough repulsive force to change its direction parallel to one of the sides of the cavity.

Continuing our above example, we know that the angle between sensors S_{L1} and S_{L2}, shown as θ in the figure, is likely to be 45^{o}. Having a triangle with one angle, θ, and two edges, d_1 and d_2, known we can use cosine theorem to find the length of the third edge, which is shown as T.

Cosine theorem: $$D^2 = d_1^2 + d_2^2 - 2 \cdot d_1 \cdot d_2 \cdot \cos\alpha \tag{3.29}$$

Once we have it, we can then use all those information to find β, inclination angle of the obstacle, using sinus theorem.

Sine theorem: $$\frac{D}{\sin\alpha} = \frac{d_1}{\sin(\pi - \beta - \alpha)} = \frac{d_2}{\sin\beta} \tag{3.30}$$

Finally, we can define $\vec{F}_{OB}$; the repulsive force acting on the body of the agent, using the direction information for direction, and using the distances d_1 and d_2 for magnitude.

$$\left\|\vec{F}_{OB}\right\| = \frac{d_{\max}^2}{d_{obs}^2} \tag{3.31}$$

where $d_{\max}$ is the range of the simulated sensors, and d_{obs} is the minimum distance between the center of mass of the robot and the “virtual” obstacle, shown with dotted lines in Figure 3.4. The distance to the obstacle, d_{obs}, can be easily calculated using d_1 and β,

$$d_{obs} = d_1 \cdot \cos\left(\frac{\pi}{2} - \beta\right) \tag{3.32}$$

In such a configuration, increasing the number of sensors will increase the detection resolution, and by the way, obstacles will generally be detected by more than one sensor. However, in case one sensor detects an obstacle with no neighbor sensor returning a measurement, we can directly use this measurement as d_{obs} and then calculate $\vec{F}_{OB}$ as before.

In both of the above methods, it is clear that the force magnitude is inversely proportional to the square of the distance. This is explicitly done in order to have increasing repulsion as the robot comes close to the obstacles. Moreover, the force tends to infinity when the obstacle is very close, which ensure the fast response if a very close obstacle is suddenly detected. The proportionality constant is chosen as the square of the range of the sensor to have a force from one to infinity as magnitude, even for very different classes of sensors.

In brief, using direction and distance information, we calculated imaginary repulsive force acting on the body of the agent. Then our control algorithm changes the direction of the agent such that the force along the heading direction of the agent is kept at certain threshold, usually zero, which ensures the clearance of the agents path. In addition using the force's magnitude, we can define a gradual deceleration and by the way, we can obtain a naturally decreasing velocity while the robot moves toward obstacles. The block diagram for the algorithm is shown in Figure 3.5. Details of the control scheme are studied in the following text.

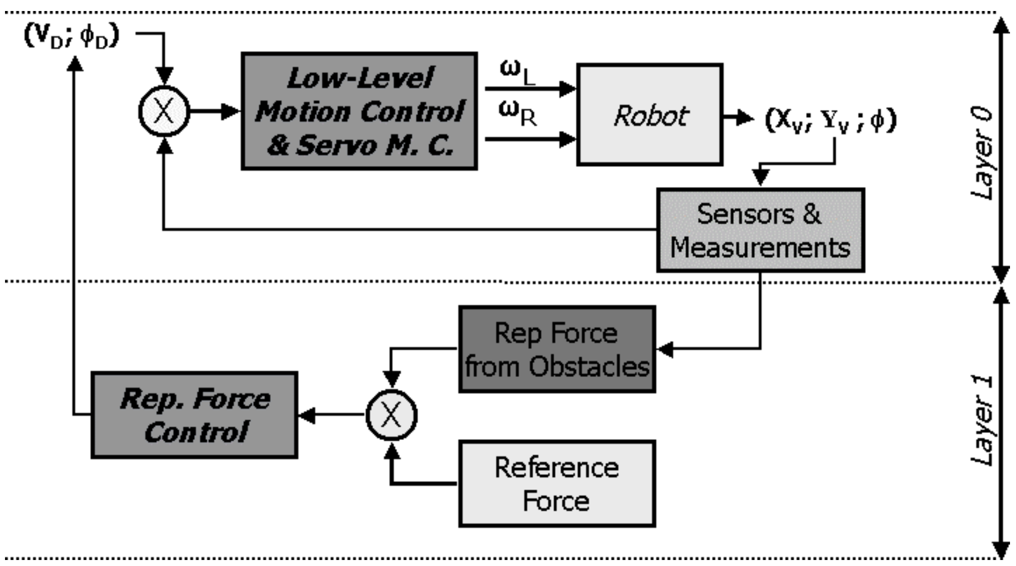


Figure 3.5: Layer 0 and layer 1 shown together. Block diagram for the repulsive force control algorithm is also shown.

The block, referred as "Repulsive Force Control", is the part that is adjusting the agents velocity and steering direction, by changing the references of the low-level motion controller, in order to minimize the repulsive force along this direction. If we denote this repulsive force as $\vec{F}_{OB}$, with components f_{obs} and θ_{obs} that are magnitude and direction respectively, then we can use any suitable control algorithm to "control" this experienced force by changing the heading direction of the agent. We preferred componentwise sliding mode controller to minimize the repulsive force acting on the agent. However, any other control could be used.

In the system, we have two inputs: magnitude and direction of the force vector, and two outputs: reference velocity and steering angle. Fist of all, using the given inputs and state of the agent, we can obtain the force along the heading direction and along the direction perpendicular to it F_r and F_ϕ are actually the states of the system that we want to control.

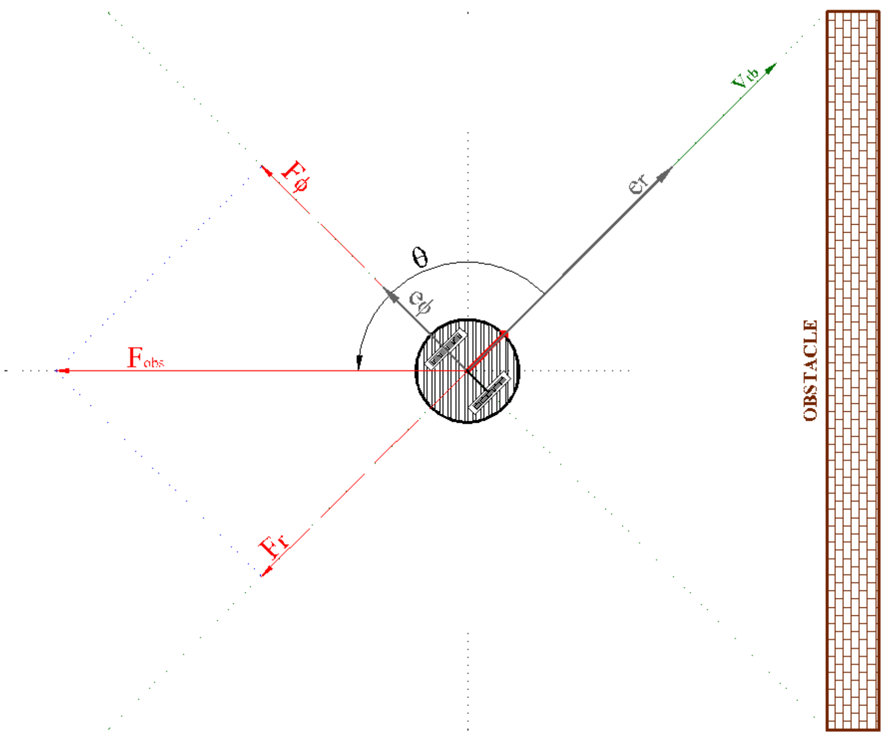


Figure 3.6: General structure for repulsive force control.

Assume repulsive force acting in the center of body of the agent $\vec{F}_{OB}(f_{obs}, \theta_{obs})$ to have components $f_{obs} \geq 0$ and $0 \leq \theta_{obs} \leq 2\pi$. Also, assume that our agent is in state (v, ϕ) as shown in Figure 3.6, where the components are denoting velocity and the steering direction angle respectively. Then we can project the repulsive force to the velocity direction of the agent (marked with subscript r) and to the perpendicular direction to it (marked with subscript ϕ), as follows:

$$\begin{aligned} &\theta = \phi - \theta_{obs} \\ &-\pi \leq \theta \leq \pi \\ &F_r = f_{obs} \cdot \cos\theta \\ &F_\phi = f_{obs} \cdot \sin\theta \end{aligned} \tag{3.33}$$

where θ denotes the angle between the velocity direction of the agent and the repulsive force direction. For practical reasons we trap this θ value between $-\pi$ and π.

We can assign our controls to be the derivatives of those forces. Then the state space equation can be written as,

$$\begin{aligned}\dot{F}_r &= u_r \\ \dot{F}_\phi &= u_\phi\end{aligned} \tag{3.34}$$

which have the form

$$\frac{dF}{dt} = f(F) + B \cdot u \tag{3.35}$$

where

$$\begin{aligned}
F &= \begin{bmatrix} F_r \\ F_\phi \end{bmatrix} \\
f(F) &= 0 \\
B &= \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
u &= \begin{bmatrix} u_r \\ u_\phi \end{bmatrix}
\end{aligned} \tag{3.36}$$

Both components of the output vector may be assigned its own sliding manifold and hence may be controlled independently since they are mathematically independent. Alternatively, both two components are dealt with as a vector. Now, using controls u_r and u_ϕ, which later will be related with v and ϕ, together with the references forces F_r^{ref} and F_ϕ^{ref}, we can define errors, which we are trying to tend to zero

$$\begin{aligned} e_r &= F_r^{ref} - F_r \\ e_\phi &= F_\phi^{ref} - F_\phi \end{aligned} \tag{3.37}$$

What are the reference values? We are trying to keep the force along the hading direction under certain threshold say f_r^{ref}, which is usually zero, to ensure the clearance of the path. Therefore, we can assume that the other component of the desired force will be such that, the sum of both components gives the original reading, if it is still sensible to agent's sensors. Therefore

$$\begin{aligned} F_r^{ref} &= f_r^{ref} \\ F_\phi^{ref} &= \left(|f_{obs}| - f_r^{ref}\right)\cdot \operatorname{sgn}(\pi - \theta) \end{aligned} \tag{3.38}$$

Here the term with sign is used to compare θ with π in order to decide the direction of the reference force in the perpendicular direction. Alternatively $F_\phi^{ref} = \left(|f_{obs}| - f_r^{ref}\right)\cdot \operatorname{sgn}(F_\phi)$ could be used.

Once we know references, and errors, we can then define the sliding manifolds for componentwise control,

$$\sigma_i = e_i \qquad\qquad i = r,\phi \tag{3.39}$$

since the aim is to converge both errors, e_r and e_ϕ, to zero. Each component of the control input vector u_i is responsible for ensuring sliding mode occurs along its respective manifold σ_i. Moreover, controls should be chosen such that the candidate Lyapunov function satisfies Lyapunov stability criteria. Consider the selected positive definite Lyapunov function candidate function for each component,

$$\gamma_i = \frac{1}{2}\sigma_i^2 \geq 0 \qquad\qquad i = r,\phi \tag{3.40}$$

with its time derivative along the system trajectories

$$\dot{\gamma}_i = \sigma_i\dot{\sigma}_i = -D_i\sigma_i^2 \qquad\qquad i = r,\phi \tag{3.41}$$

which are negative definite for some scalars $D_i > 0$. Consequently, finite convergence of each system components to the manifolds $\sigma_i = 0$ will be established. Now by taking σ_i into the common factor,

$$\sigma_i\left(\dot{\sigma}_i - D_i\sigma_i\right) = 0 \tag{3.42}$$

In the above equation either σ_i or $(\dot{\sigma}_i - D_i \cdot \sigma_i)$ is zero. In the first case, we don't have information about the second part of the equation, which means that we don't know the behavior of the sliding manifold σ_i. However, if the second part is zero, then obviously σ_i will exponentially tend to zero. We can now explicitly write those equations,

$$\begin{aligned} \dot{e}_v^{corr} &= -D_r \cdot e_r \Rightarrow \dot{F}_r^{ref} - \dot{F}_r = -D_r \cdot e_r \\ \dot{e}_\phi^{corr} &= -D_\phi \cdot e_\phi \Rightarrow \dot{F}_\phi^{ref} - \dot{F}_\phi = -D_\phi \cdot e_\phi \end{aligned} \tag{3.43}$$

We can replace both $\dot{F}_r$ and $\dot{F}_\phi$ using information from state space equation (3.34),

$$\begin{aligned} \dot{F}_r^{ref} - b_{11} \cdot u_1 - f_r + D_r \cdot e_r = 0 \\ \dot{F}_\phi^{ref} - b_{22} \cdot u_2 - f_\phi + D_\phi \cdot e_\phi = 0 \end{aligned} \tag{3.44}$$

remembering that $b_{11} = 1$, and $b_{22} = 1$. Now controls u_r and u_ϕ can be obtained,

$$\begin{aligned} u_r &= \dot{F}_r^{ref} - f_r + D_r \cdot e_r \\ u_\phi &= \dot{F}_\phi^{ref} - f_\phi + D_\phi \cdot e_\phi \end{aligned} \tag{3.45}$$

Once more using state space equation to replace f_r and f_ϕ,

$$\begin{aligned} u_r &\equiv \dot{F}_r^{ref} - (\dot{F}_r - u_1) + D_r \cdot e_r \\ u_\phi &\equiv \dot{F}_\phi^{ref} - (\dot{F}_\phi - u_2) + D_\phi \cdot e_\phi \end{aligned} \tag{3.46}$$

which can be simplified and reordered as,

$$\begin{aligned} u_r &\equiv u_r + (\dot{F}_r^{ref} - \dot{F}_r) + D_r \cdot e_r \\ u_\phi &\equiv u_\phi + (\dot{F}_\phi^{ref} - \dot{F}_\phi) + D_\phi \cdot e_\phi \end{aligned} \tag{3.47}$$

Here rather than using equality, identical sign is used since both sides of the equation have u_r or u_ϕ and those control are not related with the future but past control. In a discrete time system, we can use this property as follows,

$$\begin{aligned} u_r(k \cdot dt) &= u_r((k-1) \cdot dt) + \dot{e}_r + D_r \cdot e_r \\ u_\phi(k \cdot dt) &= u_\phi((k-1) \cdot dt) + \dot{e}_\phi + D_\phi \cdot e_\phi \end{aligned} \tag{3.48}$$

which can further be evaluated to obtain

$$\begin{aligned} u_r(k \cdot dt) &= u_r((k-1) \cdot dt) + \frac{1}{dt} \cdot \left[dt \cdot D_r \cdot e_r(k \cdot dt) + e_r(k \cdot dt) - e_r((k-1) \cdot dt)\right] \\ u_\phi(k \cdot dt) &= u_\phi((k-1) \cdot dt) + \frac{1}{dt} \cdot \left[dt \cdot D_\phi \cdot e_\phi(k \cdot dt) + e_\phi(k \cdot dt) - e_\phi((k-1) \cdot dt)\right] \end{aligned} \tag{3.49}$$

where dt stands for discrete time interval, and k denotes the k^{th} time interval. Clearly, $k \cdot dt$ is the current time and $(k-1) \cdot dt$ represents the previous time interval.

Now we have to relate our controls with the outputs. Since

$$\begin{aligned} F_r &\propto \cos\theta \\ F_\phi &\propto \sin\theta \end{aligned} \tag{3.50}$$

then

$$\begin{aligned} \dot{F}_r &= u_r \propto \sin\dot{\theta} \\ \dot{F}_\phi &= u_\phi \propto \cos\dot{\theta} \end{aligned} \tag{3.51}$$

Now its clear that

$$\tan\dot{\theta} \propto \frac{u_r}{u_\phi} \tag{3.52}$$

where $\dot{\theta}$ is the necessary change in the angle between the direction of the agent and the direction of the repulsive force acting on it. Since we cannot change the direction of the force, we have to change reference direction of the robot so that the force rotates relatively to the agent,

$$\phi^{ref} = \phi - \delta \cdot \dot{\theta} = \phi - \delta \cdot \arctan\left(\frac{u_r}{u_\phi}\right) \tag{3.53}$$

where δ is a constant number that is used to overcome jumps in the reference direction angle ϕ^{ref}. Obviously δ must be in the same order with dt which is the simulations discrete time constant. In the application of this control method, $\delta = 2.1 \cdot \mathrm{dt} \cdot \mathrm{sgn}(\theta)$ is used instead of $\delta = 2.1 \cdot \mathrm{dt}$, in order to ensure the rotation of the agent around it-self in the correct direction.

Although our agent changes direction while going toward an obstacle and avoid them, in some configurations it happens that the agent makes a sudden turn and sense an obstacle at so close distance that it does not have time to change direction one more time to avoid this new obstacle. One example configuration is shown below in Figure 3.7. In this configuration if the agent didn't decelerate before entering the circle marked as A, it could not turn once more to avoid the second part of the obstacle before reaching goal point G.

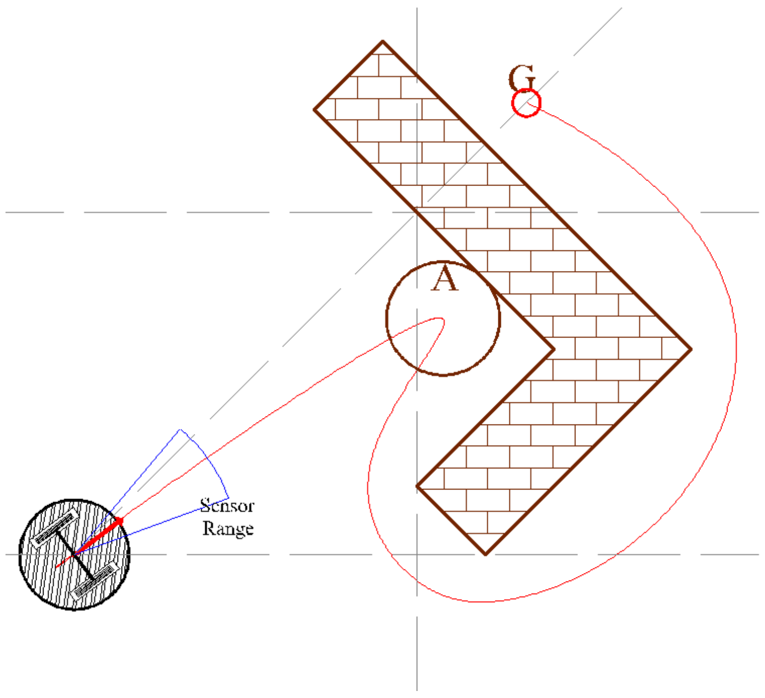


Figure 3.7: An example configuration requiring the agent to decelerate while changing the direction.

In such cases θ, the angle between the force and the direction of movement plays an important role. If θ is close to zero then the obstacle is close to the heading direction, and if θ is close to $\pm\frac{\pi}{2}$ then the obstacle is in one of the sides of the agent.

We can conclude that comparing θ value with $\frac{\pi}{2}$ we can decide to either decelerate or not. Necessary calculations are given below. Since $\theta = \phi - \theta_{obs}$, where $-\pi \le \theta \le \pi$ as already selected,

$$\chi = \frac{\left(\dfrac{|\theta|}{\pi/2}\right)}{2} = \frac{|\theta|}{\pi} \qquad (3.54)$$
$$0 \le \chi \le 1$$

Here we first take absolute value of θ, since obstacle at left or right hand side must have the same effect, and compare it with the value $\frac{\pi}{2}$ and then we scale it to fit between 0 and 1. We have to scale it since θ is between $-\pi$ and π, else the comparison will yield between 0 and 2. The desired behavior is maximum deceleration for $\theta \cong 0 \Rightarrow \chi = 0$, and no deceleration for $\theta \cong \pm\frac{\pi}{2} \Rightarrow \chi = 1$.

$$a^{ref} = a_0 \cdot \left(2 \cdot \left(\chi - \frac{1}{2}\right)\right) = a_0 \cdot 2 \cdot \left(\frac{|\theta|}{\pi} - \frac{1}{2}\right) \qquad (3.55)$$

and then

$$v^{ref} = v - a^{ref} \cdot dt = v - a_0 \cdot 2 \cdot \left(\frac{|\theta|}{\pi} - \frac{1}{2}\right) \cdot dt \qquad (3.56)$$

where a_0 is the scalar gain that defines the maximum deceleration amount in each cycle.

Normally, agents are asked to move to the target point as quick as possible. However, the controller algorithm is just decelerating the robot increasingly. To overcome this problem we implemented two different precautions.

First precaution is limiting the minimum velocity reference that force control module can assign for the low-level motion controller. The implementation is simple

$$v^{ref} = \max\left(v_{\min}, \left(v - a^{ref} \cdot dt\right)\right) \tag{3.57}$$

where $v_{\min}$ represents the limit.

Second precaution is implemented inside the low-level motion controller. The velocity of the robot is accelerated if no obstacle for is detected in the heading direction. In other words, if the force controller is not affecting the velocity reference, low-level motion controller accelerates the agent. Actually, this scheme is also used when any velocity reference higher than the actual velocity is given. Since that way, we are avoiding jumps in the reference velocity, causing smoother accelerations.

3.3 Drive Toward Goal Point (DTG)

Layer 2 is responsible to keep our agent moving toward the desired (x_G, y_G) coordinates, referred as the "Goal Point". In a similar fashion to the Layer 1, we can this time define an attractive force from the goal point to the agent and then try to control this force by changing the velocity and the direction of the robot.

Let our agent be at the position $P_V(x,y)$, and let the goal point, where we the robot tries to reach to be at position $P_G(x_G, y_G)$. Then we can defile an imaginary attractive force vector $\vec{G}$, as follows

$$\vec{G} = \frac{\left(\vec{P}_G - \vec{P}_V\right)^2}{B} \cdot \frac{\left(\vec{P}_G - \vec{P}_V\right)}{\left|\vec{P}_G - \vec{P}_V\right|} \tag{3.58}$$

where $\vec{P}_G$ and $\vec{P}_V$ are vectors from the origin to goal and to the vehicle (agent) respectively. B is the scalar gain used to adjust the magnitude of vector $\vec{G}$. In this configuration, the first term is the magnitude of $\vec{G}$ that is proportional to the square of the distance between goal and agent and the second term is unity vector that defines the direction of the vector.

That square law dominates when the agent is far away from the goal point and helps it to quickly orient itself toward that point. Then the force decreases. By the way, the robot follows a smooth trajectory while traveling to that special point.

Having the attractive force's magnitude and direction available, we can use another sliding mode controller to orient our robot.

In the system, we have two inputs: magnitude and direction of the attractive force vector, and two outputs: reference velocity and steering angle that will drive the robot toward the goal point. Fist of all, using the given inputs and state of the agent, we can obtain the force along the heading direction and along the direction perpendicular to it F_r and F_ϕ are actually the states of the system that we want to control.

Assume attractive force acting in the center of body of the agent $\vec{G}(g_{atr}, \theta_{atr})$ to have components $g_{atr} \geq 0$ and $0 \leq \theta_{atr} \leq 2\pi$. Also, assume that our agent is in state (v, ϕ), where the components are denoting velocity and the steering direction angle respectively. Then we can project the repulsive force to the velocity direction of the agent (marked with subscript r) and to the perpendicular direction to it (marked with subscript ϕ), as follows:

$$\begin{aligned} &\theta = \phi - \theta_{atr} \\ &-\pi \leq \theta \leq \pi \\ &F_r = g_{atr} \cdot \cos\theta \\ &F_\phi = g_{atr} \cdot \sin\theta \end{aligned} \tag{3.59}$$

where θ denotes the angle between the velocity direction of the agent and the attractive force direction. For practical reasons we trap this θ value between $-\pi$ and π.

We can assign our controls to be the derivatives of those forces. Then the state space equation can be written as,

$$\begin{aligned} \dot{F}_r &= u_r \\ \dot{F}_\phi &= u_\phi \end{aligned} \tag{3.60}$$

which have the form

$$\frac{dF}{dt} = f(F) + B \cdot u \tag{3.61}$$

where

$$\begin{aligned} F &= \begin{bmatrix} F_r \\ F_\phi \end{bmatrix} \\ f(F) &= 0 \\ B &= \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ u &= \begin{bmatrix} u_r \\ u_\phi \end{bmatrix} \end{aligned} \tag{3.62}$$

Both components of the output vector may be assigned its own sliding manifold and hence may be controlled independently since they are mathematically independent. Alternatively, both two components are dealt with as a vector. Now, using controls u_r and u_ϕ, which later will be related with v and ϕ, together with the references forces F_r^{ref} and F_ϕ^{ref}, we can define errors, which we are trying to tend to zero

$$\begin{aligned} e_r &= F_r^{ref} - F_r \\ e_\phi &= F_\phi^{ref} - F_\phi \end{aligned} \tag{3.63}$$

What are the reference values? We are trying to keep the force along the hading direction, as much as possible for that specific position of the robot, (x, y) to ensure the movement toward the goal point. Therefore, we can assume that the other component of the desired force will be such that, the sum of both components gives the original reading. Therefore

$$\begin{aligned} F_r^{ref} &= g_{atr} \\ F_\phi^{ref} &= 0 \end{aligned} \tag{3.64}$$

Once we know references, and errors, we can then define the sliding manifolds for componentwise control,

$$\sigma_i = e_i \qquad i = r, \phi \tag{3.65}$$

since the aim is to converge both errors, e_r and e_ϕ, to zero. Each component u_i of the control input vector u, is responsible for ensuring sliding mode occurs along its respective manifold σ_i. Moreover, controls should be chosen such that the candidate Lyapunov function satisfies Lyapunov stability criteria. Consider the selected positive definite Lyapunov function candidate function for each component,

$$\gamma_i = \frac{1}{2}\sigma_i^2 \geq 0 \qquad i = r, \phi \tag{3.66}$$

with its time derivative along the system trajectories

$$\dot{\gamma}_i = \sigma_i \dot{\sigma}_i = -D_i \sigma_i^2 \qquad i = r, \phi \tag{3.67}$$

which are negative definite for some scalars $D_i > 0$. Consequently, finite convergence of each system components to the manifolds $\sigma_i = 0$ will be established. Now by taking σ_i into the common factor,

$$\sigma_i \left(\dot{\sigma}_i - D_i \sigma_i \right) = 0 \qquad i = r, \phi \tag{3.68}$$

In the above equation either σ_i or $\left(\dot{\sigma}_i - D_i \cdot \sigma_i \right)$ is zero. In the first case, we don't have information about the second part of the equation, which means that we don't know the behavior of the sliding manifold σ_i. However, if the second part is zero, then obviously σ_i will exponentially tend to zero. We can now explicitly write those equations,

$$\begin{aligned}\dot{e}_v^{corr} &= -D_r \cdot e_r \Rightarrow \dot{F}_r^{ref} - \dot{F}_r = -D_r \cdot e_r \\ \dot{e}_\phi^{corr} &= -D_\phi \cdot e_\phi \Rightarrow \dot{F}_\phi^{ref} - \dot{F}_\phi = -D_\phi \cdot e_\phi\end{aligned} \tag{3.69}$$

We can replace both $\dot{F}_r$ and $\dot{F}_\phi$ using information from state space equation (3.60),

$$\begin{aligned}\dot{F}_r^{ref} - b_{11} \cdot u_1 - f_r + D_r \cdot e_r &= 0 \\ \dot{F}_\phi^{ref} - b_{22} \cdot u_2 - f_\phi + D_\phi \cdot e_\phi &= 0\end{aligned} \tag{3.70}$$

remembering that $b_{11} = 1$, and $b_{22} = 1$. Now controls u_r and u_ϕ can be obtained,

$$\begin{aligned}u_r &= \dot{F}_r^{ref} - f_r + D_r \cdot e_r \\ u_\phi &= \dot{F}_\phi^{ref} - f_\phi + D_\phi \cdot e_\phi\end{aligned} \tag{3.71}$$

Once more using state space equation to replace f_r and f_ϕ,

$$\begin{aligned}u_r &\equiv \dot{F}_r^{ref} - \left(\dot{F}_r - u_1\right) + D_r \cdot e_r \\ u_\phi &\equiv \dot{F}_\phi^{ref} - \left(\dot{F}_\phi - u_2\right) + D_\phi \cdot e_\phi\end{aligned} \tag{3.72}$$

which can be simplified and reordered as,

$$\begin{aligned}u_r &\equiv u_r + \left(\dot{F}_r^{ref} - \dot{F}_r\right) + D_r \cdot e_r \\ u_\phi &\equiv u_\phi + \left(\dot{F}_\phi^{ref} - \dot{F}_\phi\right) + D_\phi \cdot e_\phi\end{aligned} \tag{3.73}$$

Here rather than using equality, identical sign is used since both sides of the equation have u_r or u_ϕ and those control are not related with the future but past control. In a discrete time system, we can use this property as follows,

$$\begin{aligned}u_r(k \cdot dt) &= u_r((k-1) \cdot dt) + \dot{e}_r + D_r \cdot e_r \\ u_\phi(k \cdot dt) &= u_\phi((k-1) \cdot dt) + \dot{e}_\phi + D_\phi \cdot e_\phi\end{aligned} \tag{3.74}$$

which can further be evaluated to obtain

$$\begin{aligned} u_r(k \cdot dt) &= u_r((k-1) \cdot dt) + \frac{1}{dt} \cdot \left[dt \cdot D_r \cdot e_r(k \cdot dt) + e_r(k \cdot dt) - e_r((k-1) \cdot dt)\right] \\ u_\phi(k \cdot dt) &= u_\phi((k-1) \cdot dt) + \frac{1}{dt} \cdot \left[dt \cdot D_\phi \cdot e_\phi(k \cdot dt) + e_\phi(k \cdot dt) - e_\phi((k-1) \cdot dt)\right] \end{aligned} \tag{3.75}$$

where dt stands for discrete time interval, and k denotes the k^{th} time interval. Clearly, $k \cdot dt$ is the current time and $(k-1) \cdot dt$ represents the previous time interval.

Now we have to relate our controls with the outputs. Since

$$\begin{aligned} F_r &\propto \cos\theta \\ F_\phi &\propto \sin\theta \end{aligned} \tag{3.76}$$

then

$$\begin{aligned} \dot{F}_r &= u_r \propto \sin\dot{\theta} \\ \dot{F}_\phi &= u_\phi \propto \cos\dot{\theta} \end{aligned} \tag{3.77}$$

Now we can conclude

$$\tan\dot{\theta} \propto \frac{u_r}{u_\phi} \tag{3.78}$$

where $\dot{\theta}$ is the necessary change in the angle between the direction of the agent and the direction of the attractive force acting on it. Since we cannot change the direction of the force, we have to change reference direction of the robot so that it turns,

$$\phi^{ref} = \phi - \delta \cdot \dot{\theta} = \phi - \delta \cdot \arctan\left(\frac{u_r}{u_\phi}\right) \tag{3.79}$$

where δ is a constant number that is used to overcome jumps in the reference direction angle ϕ^{ref}. Obviously δ must be in the same order with dt which is the simulations discrete time constant. In the application of this control method, $\delta = 2.1 \cdot \mathrm{dt} \cdot \mathrm{sgn}(\theta)$ is used instead of $\delta = 2.1 \cdot \mathrm{dt}$, in order to ensure the rotation of the agent around it-self in the correct direction.

Moreover, a deceleration when the agent is moving in the opposite direction and acceleration when the movement is in the correct direction, can be defined. However, this is not seen suitable since to avoid an obstacle robot may have to move in the opposite direction for longtime. In such a case, the whole process will take much longer time. Therefore, the acceleration and deceleration is dominated only by the obstacle avoidance layer.

3.4 Behavior Arbitration

Once both Layer 1 and Layer 2 are added to the system, the general view takes the form shown in Figure 3.8, below. Now if the situation is carefully examined the fallowing is seen, Layer 2 that is goal point tracking is always working and producing

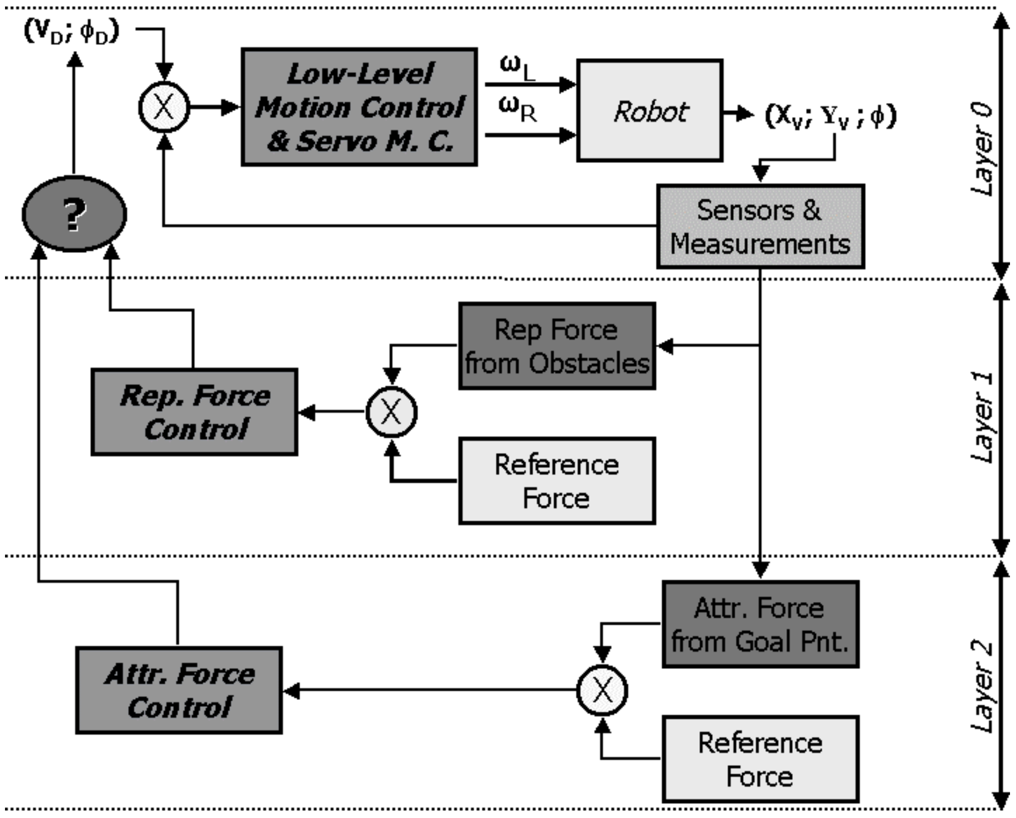


Figure 3.8: First three layers together, main controller, obstacle avoidance and drive toward goal.

an output for the low-level motion controller. In addition, once the robot sense an obstacle, Layer 1 that is obstacle avoidance will run and produce an output.

In such a case, where two outputs are present, both are guaranteed to conflict, since the obstacle layer do not takes into account all obstacles that is present but that is avoiding the agent to reach the target point. The agent may neglects the goal tracking and follow the trajectory drawn by the obstacle avoidance layer only. It may never get read of the obstacles and can be trapped. On the other hand, obstacle avoidance layer cannot be neglected since then collisions will occur.

Those two behaviors then must be combined. The low-level motion control receives the velocity and orientation as reference. If Layer 1 and Layer that are determining those two references, returns the change in the actual velocity and orientation, then according to some weight, which is the priority, those two references can be combined.

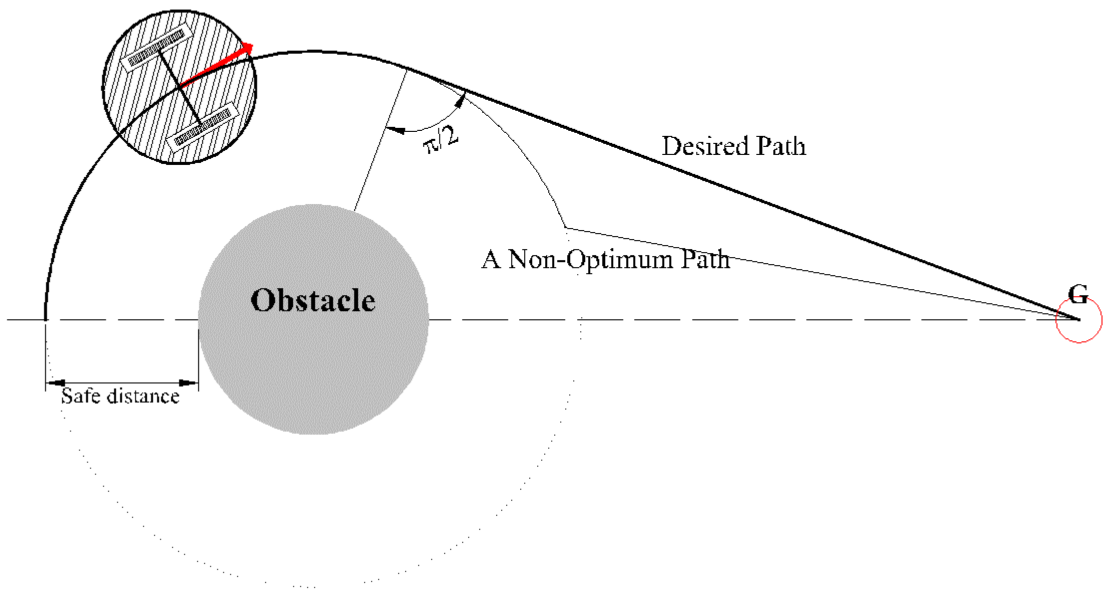


Figure 3.9: Optimum and non-optimum path example that can be observed when the agent avoids an obstacle.

Assume the agent that is moving toward the goal point while avoiding an obstacle in midway (Figure 3.9). Closely observing the situation, we can see that the agent must be under the "control" of the obstacle avoidance layer when the obstacle that is augmented by a safety distance, is on the line joining the agents position and the goal point. In other worlds, the robot must disregard the Layer 2 when the line joining its position and the goal point is tangent to the boundaries of the obstacle.

In the above formulation, the agent still uses one of the layers at a time disregarding the other. To have smoother motion and to direct the robot in the correct direction while avoiding obstacles instead of such a subsumption a linear addition such as

$$\begin{aligned} v^{ref} &= v + A_v \cdot \Delta v^{ref}_{Layer1} + B_v \cdot \Delta v^{ref}_{Layer2} \\ \phi^{ref} &= \phi + A_\phi \cdot \Delta \phi^{ref}_{Layer1} + B_\phi \cdot \Delta \phi^{ref}_{Layer2} \end{aligned} \tag{3.80}$$

can be used. Here A_v, A_ϕ, B_v and B_ϕ represents weight constants. Moreover, this linear addition is not forced to have constant coefficients. When the robot feels an obstacle just in the heading direction then this obstacle must have high priority since the agent is going on it with probably the maximum allowed velocity. On the other hand, an obstacle that is close to point where the robot will leave it must have relatively low priority since the agent will not hit it unless another obstacle appears at repels it strongly. Then we can formulate the behavior arbitration by comparing θ, the angle between repulsive force and agent's direction vectors, with $\frac{\pi}{2}$. Since $\theta = \phi - \theta_{obs}$, as already selected,

$$\chi = \frac{\left(\frac{|\theta|}{\pi/2} \right)}{2} = \frac{|\theta|}{\pi} \Rightarrow 0 \le \chi \le 1 \tag{3.81}$$

Here we first take absolute value of θ, since obstacle at left or right hand side must have the same effect, and compare it with the value $\frac{\pi}{2}$ and then we scale it to fit between 0 and 1. We have to scale it since θ is between $-\pi$ and π, else the comparison will yield between 0 and 2. The desired behavior is only obstacle avoidance for $\theta \cong 0 \Rightarrow \chi = 0$, and just goal tracking for $\theta \cong \pm\frac{\pi}{2} \Rightarrow \chi = 1$. Then

$$\begin{aligned} v^{ref} &= v + \chi \cdot \Delta v^{ref}_{Layer1} + (1-\chi) \cdot \Delta v^{ref}_{Layer2} \\ \phi^{ref} &= \phi + \chi \cdot \Delta \phi^{ref}_{Layer1} + (1-\chi) \cdot \Delta \phi^{ref}_{Layer2} \end{aligned} \tag{3.82}$$

In order to have much more natural motion, instead linearly changing function, an exponential form must be used,

$$\begin{aligned} v^{ref} &= v + \chi^2 \cdot \Delta v^{ref}_{Layer1} + (1-\chi)^2 \cdot \Delta v^{ref}_{Layer2} \\ \phi^{ref} &= \phi + \chi^2 \cdot \Delta \phi^{ref}_{Layer1} + (1-\chi)^2 \cdot \Delta \phi^{ref}_{Layer2} \end{aligned} \tag{3.83}$$

This way the agent will fully react obstacles, still moving toward the goal point with more smooth and natural movements.

3.5 Enabling and Disabling Features

The existence of a behavior in a robot does not require it to continuously work. Sometimes deliberately, mostly because of some restriction, a specific behavior may be disabled. For example, a mobile robot does not always have to wander around; it may sometimes stop at specific points and realize other tasks such as handling a box. For such reasons we also included a layer which is keeping all the enabling and disabling information. Drive mechanism, moving toward goal and obstacle avoidance are currently the layers that may be required to be disabled. Accordingly, the controller scheme is changed as in Figure 3.10. Mechanisms are explained in the following sections.

3.5.1 Drive Enable (Move or Stop)

Although mobile robots are designed to wander around, it is not always desired to keep them running all the time. Sometimes desired task from the user may require being stationary at a given position while keeping the orienting stable. For example a carrier robot that approaches its load, before lifting it must stop or at least slow down.

Since our agents may also carry similar tasks, we added a “Drive Enable” flag to the controller. Once this flag is off (have value zero) the agent decelerates quickly, and stops. While the desired velocity changes, the direction information is not affected by this flag.

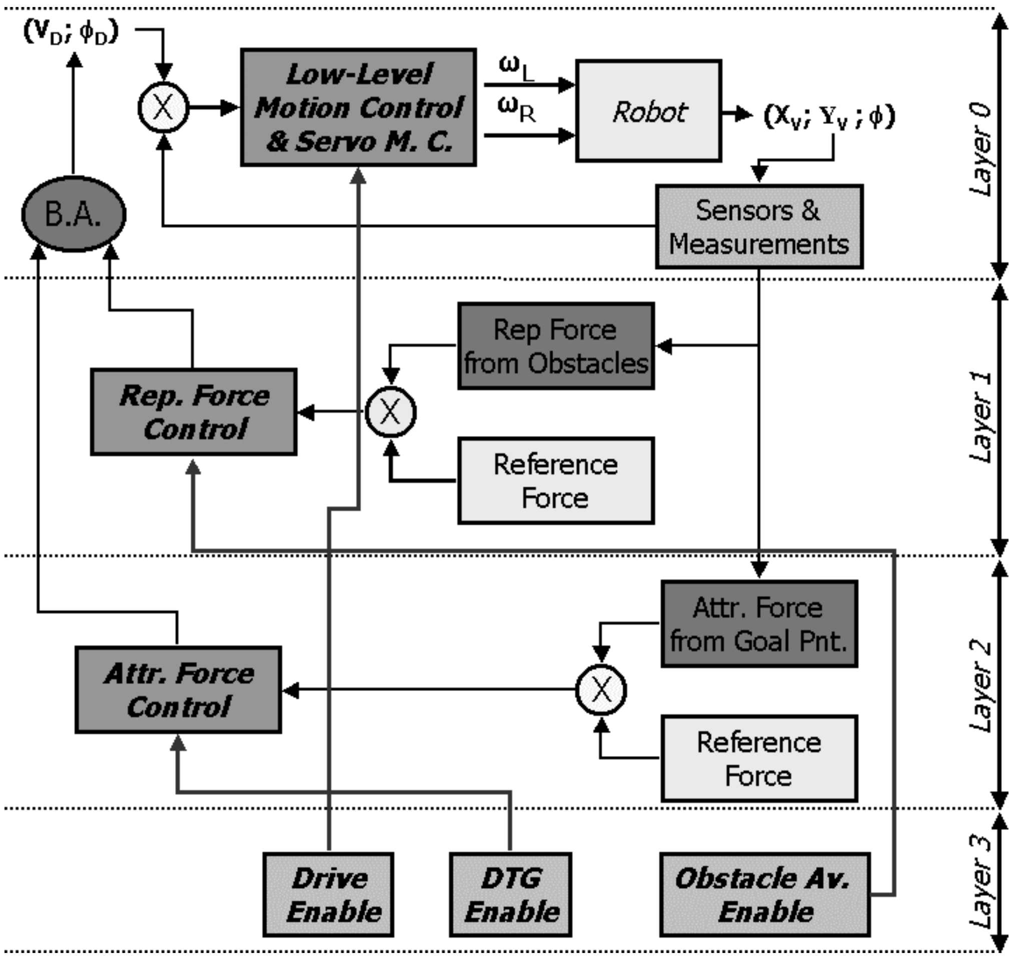


Figure 3.10: Lowest four layers together, enabling and disabling mechanism also shown.

Lower layers cannot effect the agents deceleration in this state. Nevertheless, both obstacle avoidance and drive toward goal layers are still active and they continuously change both the desired velocity and direction of the agent. However, only the desired direction is appearing in the inputs of the low-level motion controller, since the velocity reference is suppressed by higher level and replaced by zero.

However, if this flag is on (have a value different than zero) then this block has no effect on the agents movement, all of the lower and higher layers are running normally.

3.5.2 Drive Toward Goal Enable (DTG Enable)

In some extreme cases, it may be necessary to disable DTG layer (Layer 2). For example if the agent is blocked somewhere in the space and it cannot move simply because of the configurations of the vectorial forces, it may be useful to temporarily disregard the goal to handle obstacles.

For this purpose, a similar indicator to the drive enable flag is integrated into the controller. Once this “DTG Enable” flag is off (have a value zero), the agent neglect the attractive force of the goal point and moves under the influence of the repulsive force applied by obstacles only. If no force exist then the velocity and orientation is kept constant.

DTG layer cannot affect the agents motion in this state. Nevertheless, drive toward goal layer is still active and continuously produces a reference for both the velocity and direction of the agent. However, these references are suppressed by higher level.

However, if this flag is on (have a value different than zero) then this block has no effect on the agents movement, all of the lower and higher layers are running normally.

3.5.3 Obstacle Avoidance Enable

Although obstacle avoidance is essential in mobile robotics, there are cases where even this layer must be disabled. Imagine a forklift running this control. If it approaches to a specific box to hold it, it will immediately change direction by a command coming from Layer 1. Alternatively, in a laboratory experiment one may require moving obstacles that do not change direction when they detect an obstacle. In such applications disabling some of the sensors at this time may be a solution. However, remembering that this control shares all available resource to the entire layered structure, it appears that disabling hardware is not a correct choice. In the forklift example, a higher layer must determine if the carrier touches the box or not and this layer does not have to use tactile sensors if there are already installed distance sensors.

For this purpose, a similar indicator to the drive enable and DTG enable flags is integrated into the controller. Once this “Obstacle Avoidance Enable” flag is off (have a value zero), the agent neglects the repulsive force of the obstacles and moves under the influence of the attractive force applied by goal point only.

However, if this flag is on (have a value different than zero) then this block has no effect on the agents movement, all of the lower and higher layers are running normally.

3.6 Longer Term Memory Layers

We are proposing a reactive control for mobile robots. As stated before, mobile robots are mostly asked to move to a specific point while avoiding obstacles. However, this is part of the basic functionality of the robot. Such a vehicle moving around, without actually "doing anything" does not serve except experimental works. For which purposes a mobile robot may be used? Most of the researchers aim to create mobile robots that can work in hazardous environments where human may not survive. This can be bottom of a deep sea, nuclear plants, nuclear polluted areas or space… In each of those applications, the robot must remember some data and carry it to the base station where this data can be analyzed deeply to reason. This data may be temperature, nuclear radiation, altitude or similar data depending on the applications. Similarly, a robot that is working in a factory, as a carrier must also keep a log where what is transported together with source and destination points are marked.

Such kind of information are kept and processed in this layer, if necessary. However the structure of the layer is application specific. If any reaction must also be added to the mobile robot related with this data, it must be added to the upper layer. By the way, upper layer will take necessary data and manage the robot as it is requested by the user.

3.7 User Defined Layers

Briefly introduced in the previous section, mobile robots are mostly asked to work in different environments where humans may not work due to some physical limitations such as power, size or environmental temperature. Moreover, they are not only used to move around and avoid obstacles. Beside this basic functionality, they may use other task specific skills that they possess. A mobile robot may have a robot arm mounted on top of it together with a suitable end-effector to realize tasks such as collecting some pieces. In a different application, such as automated carrier, the robot must be able to lift the box, move it and suitably place to the desired location, and leave the box.

Clearly in all of those applications, there is external hardware, designed for the application, which must be executed. Moreover, such a control may require long-term memory data or any other instantaneous data that is coming from sensors. All the control related with such hardware or any modification to the existing control layers (such as changing reference values of the force control layers) should be done from this layer.

Control placed in this layer, as all other layers, have access to the sensor data, and to all lower level blocks. However, in some applications, this layer does not have to generate velocity and orientation reference that will be processed by the behavior arbitration. For example for wall following behavior, user defined layer must only change the repulsive force reference of the control, in the suitable form. Similarly changing the same reference value, agents may follow each other with predefined distance between them. Alternatively, they may be distributed over a circle.

3.8 Communication

Another indispensable feature in an agent is the communication. What communication may do in control of a mobile robot? Even an autonomous mobile robot requires high-level tasks commands, such as "move boxes at the production area to the warehouse". Or, an agent can be asked where is he, what he is doing, even what he did until now. May be some parameters, such as the controller gain in the low-level motion controller, needs to be updated. All of the mentioned requirements can easily be solve with simple communication. Moreover, such communication can safely be used in multi-robot collaboration where small time delays due to the transmission time are not important.

Since communication is the only gate of the agent to the user and to the other agents, this layer must be at the top layer from where it can reach and modify all the control elements. In our mobile robot control, this communication is the formed by incoming and outgoing messages however; any suitable communication method can be applied. Messages are composed of numbers only. The first number represents the command that is transmitted. Command may be anything like "go to (x, y)", "obstacle avoidance enable / disable", "stop" and so on. Other numbers that are inside the message represents arguments for the command. For example, "go to" command requires two arguments; x and y coordinates, "obstacle avoidance enable / disable" require only one that is either zero or one, "stop" command on the other hand does not require any argument.

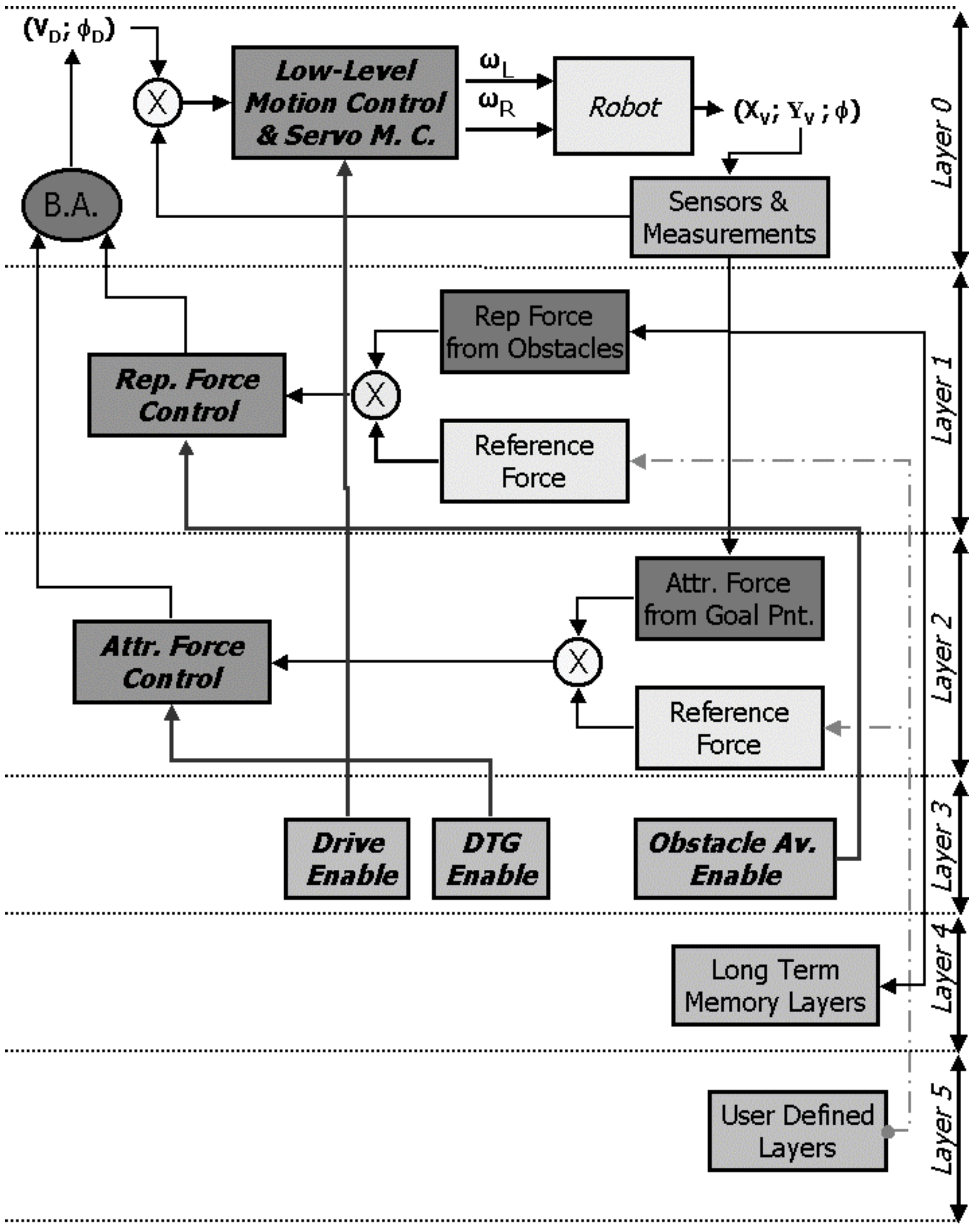


Figure 3.11: General view of the control showing where the user defined layers must be inserted.

Either the user or any other agent can transmit commands. Moreover, command may require the agent to send a response such as verification, log or any other command.

As seen in Figure 3.12, the communication layer has a bi-directional connection with all layers (not shown with all possible connection in the figure). This is simply a basic requirement since the agent must be able to send messages. In this system, any layer that has a message to send prepares it and sends to the message decoder in Layer 6. Then this block realizes what is necessary to send the message with the attached hardware, which may be practically any type.

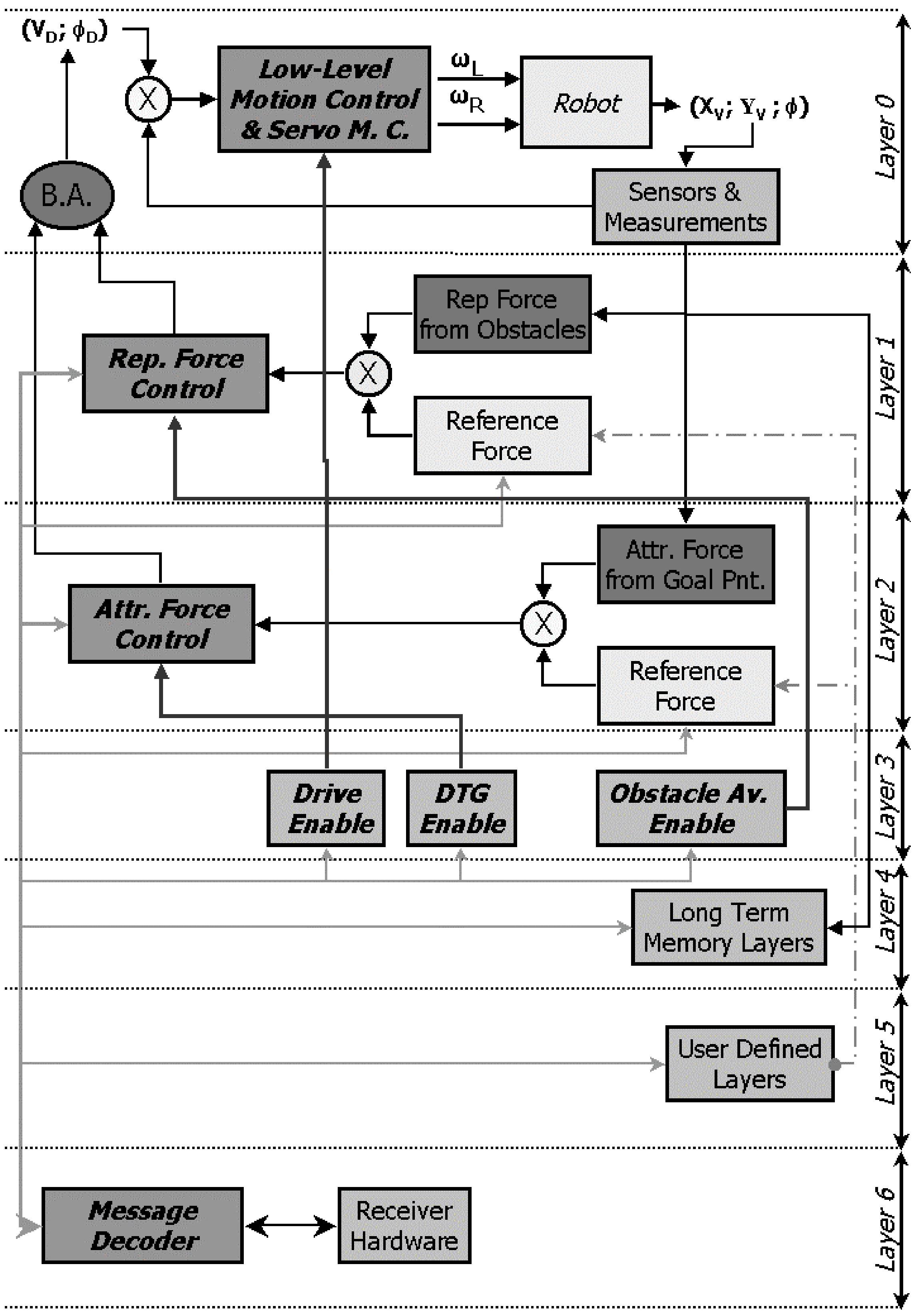


Figure 3.12: The complete structure of the mobile robot control. Note the bi-directional connection between the communication layer and other layers (all possible connections are not shown).

4. SIMULATION AND RESULTS

The proposed control method for mobile robots that can be used as a part of a multiagent system is tested on the developed simulation software written in C++ programming language. This simulation software is not a tool to create experiments and run them but a collection of routines that directly implements the control outlined in the previous chapter.

In this simulation, each experiment is it-self a C++ code that calls the necessary parts of the control routines in a suitable order. Those routines are all placed in different header files in order to simplify the experiment preparation process. Moreover, obstacles and agents are defined as classes in separate header files. This way, experiments are creating obstacles and agents as objects, and then manipulate them with simpler coding. Moreover, this type of coding gives the opportunity to have different parameters and properties for each element of the experiment.

Additionally, the code is portable to any platform such as UNIX, Linux or Macintosh as long as a C++ compiler supporting ANSI standards is present. The only part that is necessary to be updates in such a cross-platform transfer is the graphical user interface (GUI) routines if it's used.

In brief, once header files are present, necessary experiments can be prepared as different C++ codes that contain all the environment data such as position and size of the obstacles, robots etc… and execution information.

In our application, we preferred to use DOS platform, which offers simple GUI that is sufficient for our purpose. Nevertheless, the available memory and graphical functions under DOS is relatively small. In case of extensive memory requirements, one must reprogram the experiments in other platforms giving direct access to higher memory resources such as Windows or use dynamic memory allocation under DOS.

4.1 Algorithms

Main files used as experiments in this work are composed of six major parts. First part is where all initializations are done. Position of the obstacles and robots, their physical properties such as size, as well as controller parameters are all set here. All parameters including position, orientation and controller parameters have default values already defined. Therefore, all the parameters that are not set in this part are automatically loaded.

Then, in the second part of the code, message processing is done by each agent. All the parameters, reference values and similar data may be modified by the incoming messages. Both broadcast and personal incoming messages of each agent must be evaluated here. The messaging is based on the method described in the previous chapter.

Third part is where all calculations are done. Those calculations includes attractive and repulsive force control and other layers calculations, such as user defined layer, if exists. Of course, necessary sensor data is prepared in this section. However, note that almost all formulations in the previous chapter is derived for continuous time while in simulation we are running in discrete time. The assumption is the following; the discrete time interval, dt, is small enough that error caused by the mentioned transformation can be neglected.

Once all calculations are done, behavior arbitration must determine the velocity and orientation reference for the low-level motion controller. In this part, all information related to reference velocity and orientation change is collected from control layers. Then according to the defined behavior arbitration scheme, on previous chapter, they are combined to form the references of the low-level motion control.

Fifth part of the code is the low-level motion controller. According to the velocity and orientation reference coming either from the behavior arbitration or from the communication, this controller generated necessary right and left wheel velocities. And then, those values constitutes reference for the simulated servo motor drivers. When those simulated drivers run, they return simulated actual values of the wheel velocities. According to those values the position and orientation of the robot is also updated in this part.

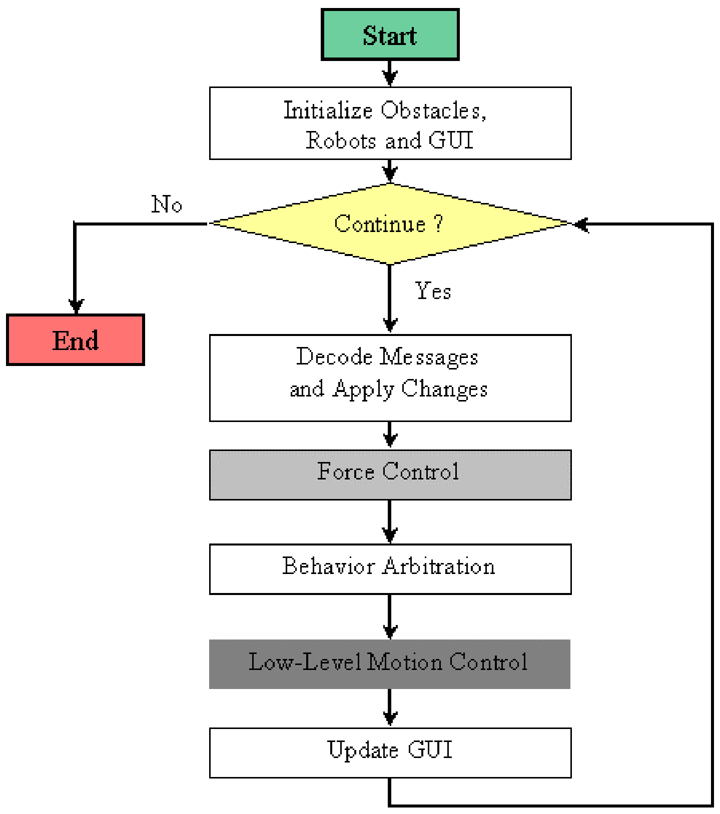


Figure 4.1: General workflow of the simulation.

Finally, the GUI is updated according to the new values. It's clear that all other updates, such as the control variables of the robot, simulation time, messages and so on, are done here.

A more detailed form of the algorithm is shown in Figure 4.2 below. In this new figure, force control and low-level motion controller blocks are drawn with their actual contents. Although in the algorithm, some of the blocks are drawn as working in parallel, this is actually not the case. Since the simulation is running in discrete time, no effort is needed to run those parts of the code in parallel. Similarly, in a real implementation, care must be taken implementing the repulsive and attractive force control procedures, as well as right and left wheel control routines.

The process explained above is our own implementation. This means that we prepared each part of the proposed control, such as obstacle avoidance, move toward goal and so on, as different routines. Moreover, we are calling those routines one by one for the control of the agent as we designed and described in the previous chapter.

Nevertheless, different algorithms and approaches can be tested using the same simulation files but calling routines in different order.

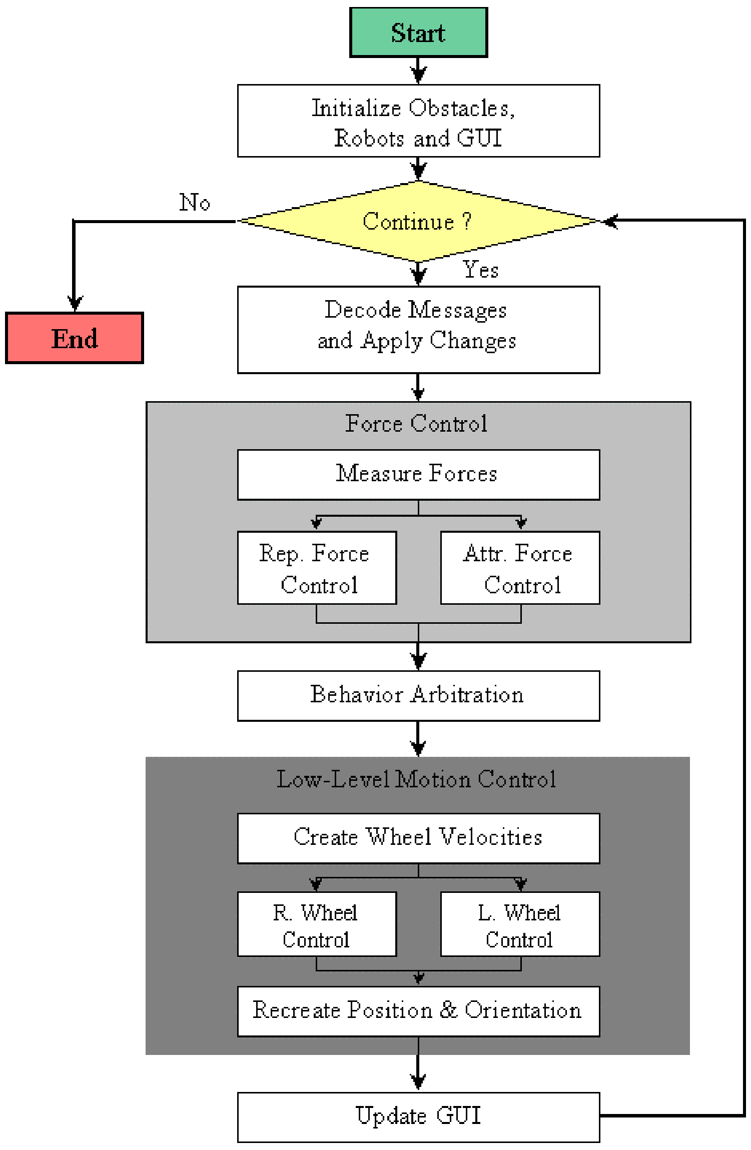


Figure 4.2: More detailed algorithm of the simulation.

4.2 Screen Layout and General Information on GUI

Our control is formed by the union of different algorithms, each having separate task such as goal tracking and obstacle avoidance. Those algorithms receive numbers, such as distances, passes through the designed processes, which produces other numbers such as position and orientation of the robot. In order to make result simpler to understand, a simple graphical user interface is developed. With the help of this interface, once an experiment is executed the screen looks like in Figure 4.3.

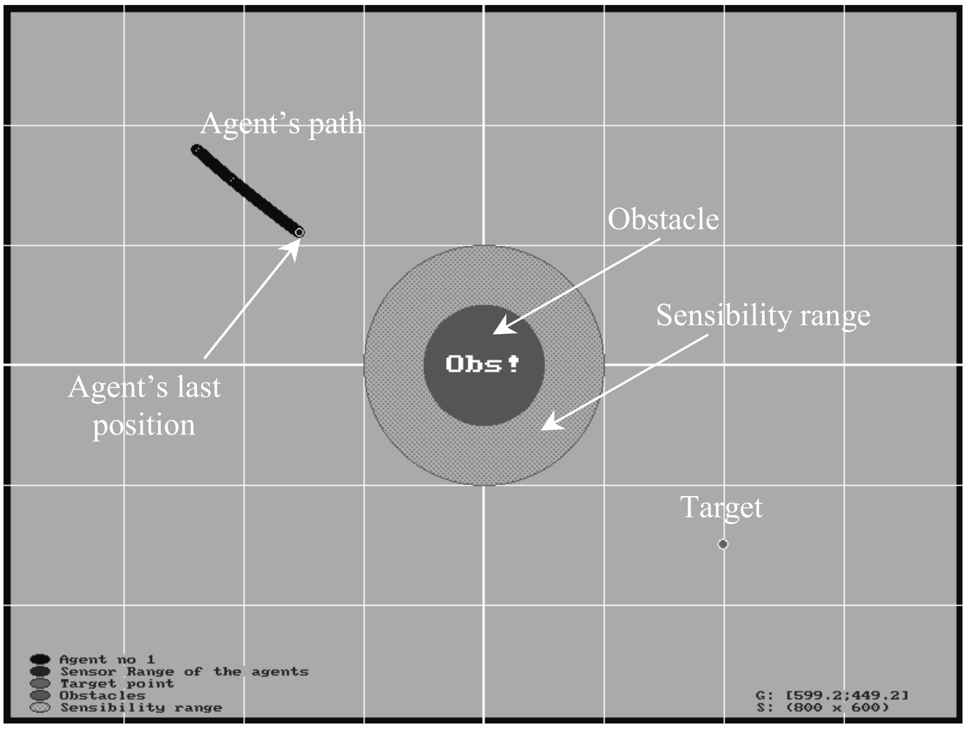


Figure 4.3: A sample screen for simulation that shows graphical user interface.

This screen represents the whole environment. The white grid serves as a ruler and it is drawn to give better feeling of the distances. The thick, dark rectangle on the borders represents solid walls present at the boundaries of the environment. Those walls constitute obstacles for the robot and must be detected and avoided. Then agents will always stay inside the visible area. In the given example screen, just at the center of the screen, a stationary obstacle is placed (marked with "Obs!"). Used obstacles are all circular shape with diameter greater than zero, solid and fully detectable by the sensors of the agents. A circle surrounds this obstacle. This ring represents "Sensibility Area", which means that agents cannot sense this obstacle unless they are inside this ring. In other words, the difference between the inner and outer radius of this ring is equal to the maximum range of the sensors carried by the agents. Moreover, we prefer to keep agents outside this are to have a "Safety Distance" between obstacles and agents. Target points are generally shown as small circles in the environment (see right-hand side, bottom in the figure).

The final and probably the most important entities in the screen are agents. Agents are shown as small disks with different colors. As the agent moves, it leaves a trace behind allowing the complete path followed by the agent to be clearly observable to the user. The last position of the agent on the other hand is surrounded by a small white circle.

4.3 Experiment 1

In this experiment, only one agent is placed to the environment together with two obstacles placed closely. Agent is told to go to the other side of the environment by setting its target point. However, both obstacles are in midway and are so closely placed that their sensibility ranges intersects (see Figure 4.4).

In such a case, the force on the line formed by equidistant points from the obstacles (roughly shown in the figure with a dotted white line), is in the direction of the agents motion. In addition, the change in the force magnitude, while the agent moves from this line to one of the obstacles, is weaker compare to one agent with one obstacle case. Therefore, expected reaction of the agent to the obstacles is relatively small in a standard potential field method. However, in our proposed control method we are not using intensities directly to influence the motion of the agent, as it is done by most of the researchers. We are using the direction information to orient the robot. Consequently, our agents react to even such small forces.

A weak point of many methods is so-called force equilibrium phenomenon that occurs in such symmetrical configurations. This means that at some point in the environment sum of repulsive force from the obstacle and attractive force of the goal returns zero. This point is referred as a local minimum in the space. Most of the robots that are using potential field method suffers from such points since the motion of the robot near them tend to stop. Door opening, corridors and similar symmetrical shapes generally create problems for mobile robots.

However, since in our method, obstacle’s repulsive forces and goal’s attractive force are treated separately and only their influence on the motion is combined, such minimas do not affect the motion of the agents.

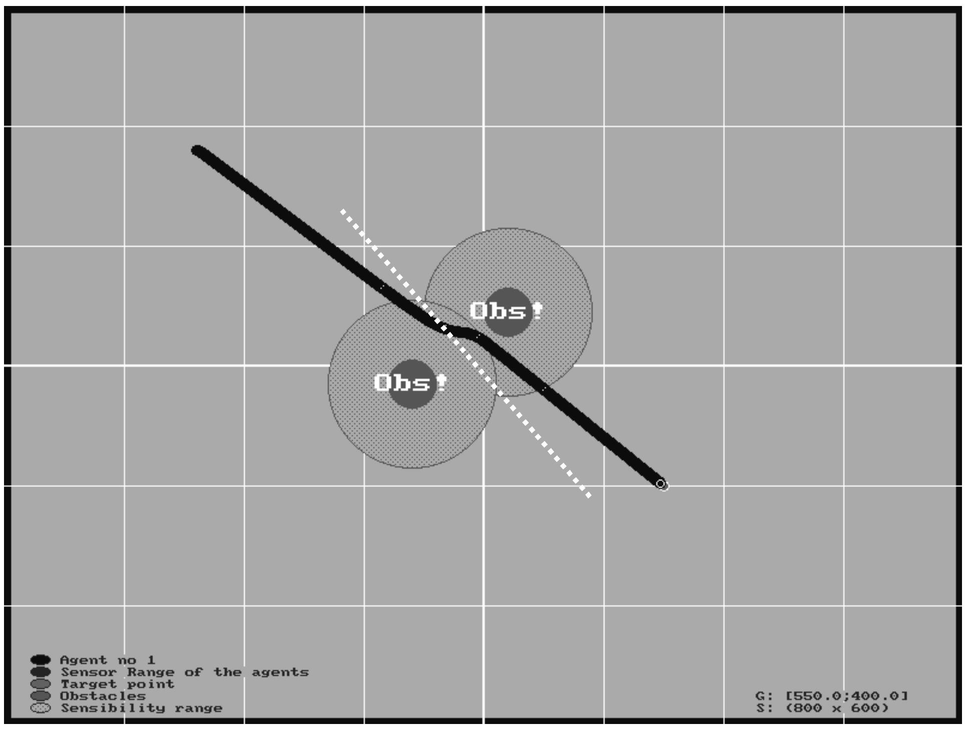


Figure 4.4: Experiment 1. An agent is placed into the environment and told to move to the other side by avoiding two obstacles placed in midway.

In standard potential field applications, the agent is expected to pass just from the centerline of the two obstacles since the forces are minimum at this location. If the agent is somehow disturbed from this position and gets closer to one of the two obstacles, it will quickly react to go in parallel to that line. In most of the applications robots crosses this line once more and then they get closer to the second obstacle. Moreover, this motion or series of reactions continuous and crates an oscillating path between two obstacles. However, in our application, even if the robot is close to one of the obstacles, it may not require to change its orientation since. This is actually a feature caused by the behavior arbitration; moving parallel to an obstacle will not cause a collision. Although such configurations look simple, they create remarkable problems in mobile robotics.

4.4 Experiment 2

In this experiment, an environment similar to the previous one is prepared. Only one agent is placed into the environment, but this time with four different obstacles. The agent must move to the opposite side of the environment while avoiding obstacles that it encounters (see Figure 4.5). In such a configuration, what is expected from the agent is the smooth and safe navigation through obstacles. The agent must direct itself toward the goal point unless an obstacle is sensed. If an obstacle is encountered then agent must change direction to move around it at a safe distance. This is exactly what is observed in the experiment.

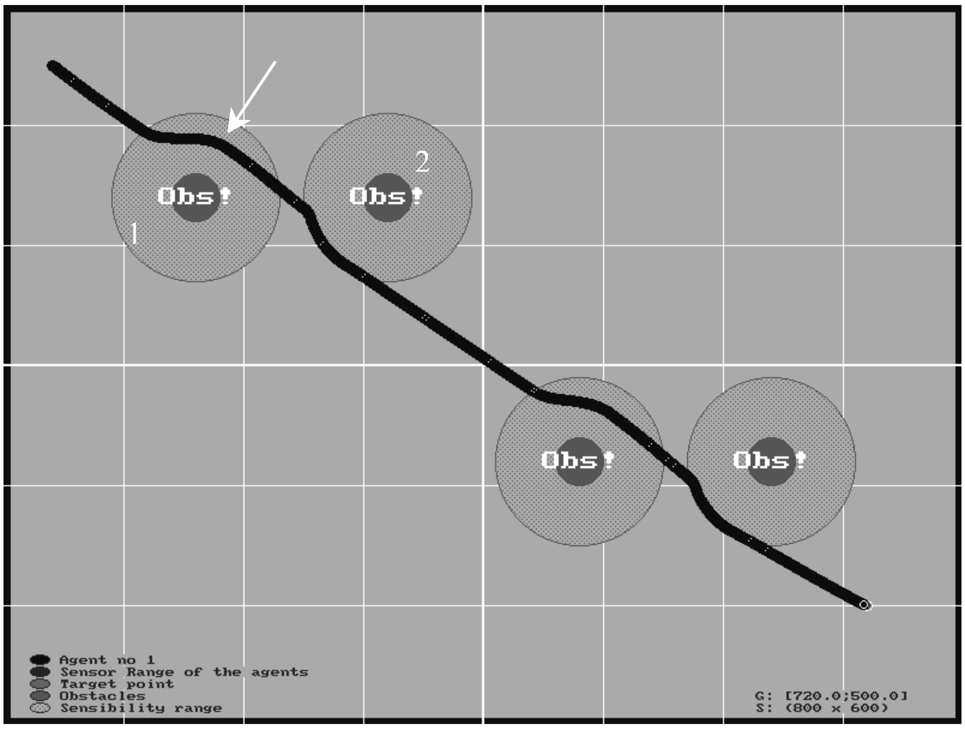


Figure 4.5: Experiment 2. An agent is placed into the environment and told to move to the other side by avoiding the obstacles.

Furthermore, in this figure, we also see clearly the work done by the behavior arbitration. When the agent gets into the sensibility range of the obstacle marked as “1”, it was moving directly to the obstacle. The force control algorithm influenced the robot to change its direction. Such that, the robot started to move around the obstacle 1. At the point shown with an arrow in Figure 4.5, obstacle 1 is not between the agent and the goal point anymore. Consequently the behavior arbitration inhibits the output from the obstacle avoidance layer until the agent reaches to the sensibility range of the obstacle marked as “2”. A similar behavior is observed with the two other obstacles.

4.5 Experiment 3

In this experiment, only one agent is placed into the environment and told to move toward a goal point. However, this goal point is a point inside a cavity or room. Walls of the room are defined by ten different disc shape obstacles. The resultant motion is shown in Figure 4.6 (sensibility ranges of the obstacles are not shown in the figure for clearance).

To ensure the motion of the agent around the cavity in counterclockwise it is initially installed to the point A as shown in the figure. Once the agent moves, it orients itself toward the goal point. However, with the first perception of the obstacle in the heading direction (point B in the figure) both obstacle avoidance and goal tracking behaviors produces conflicting references for their successor layers. However, behavior arbitration manages those conflicts and guides the robot safely toward the goal point.

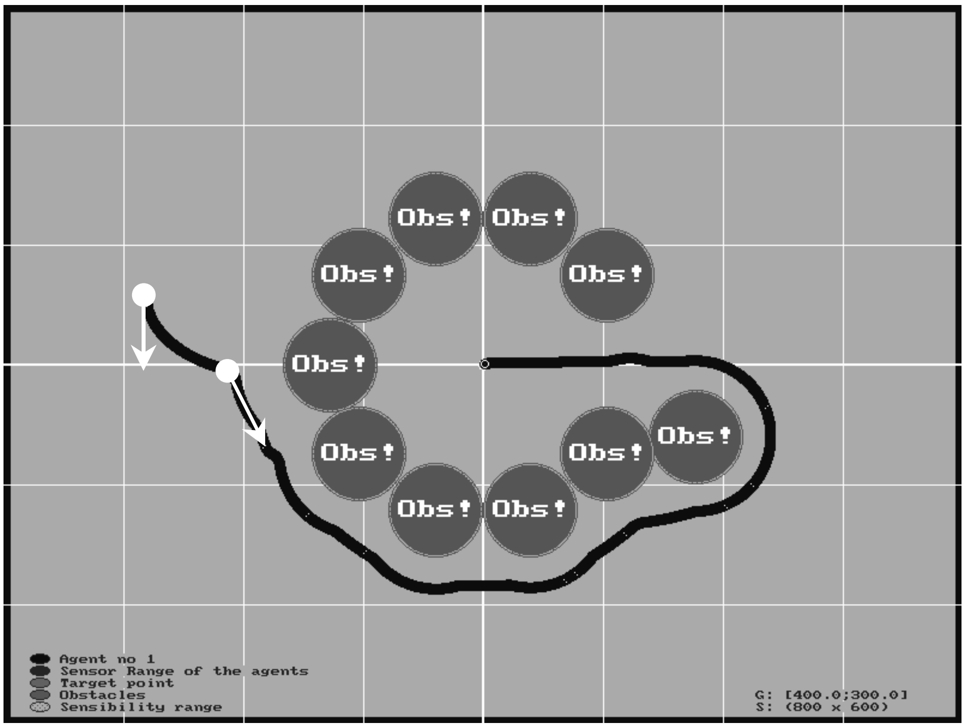


Figure 4.6: Experiment 3. An agent is placed into the environment and told to move toward the goal point hided inside a cavity formed by many obstacles.

4.6 Experiment 4

In this experiment, we tested the reaction of the agent to the moving obstacles. As shown in Figure 4.7, an agent is placed at the point S and told to move to the point T. Four other agents marked as MO 1, MO 2, MO 3 and MO 4, are also placed in the environment. However, their obstacle avoidance layers are disabled. Consequently, we obtained four moving obstacles.

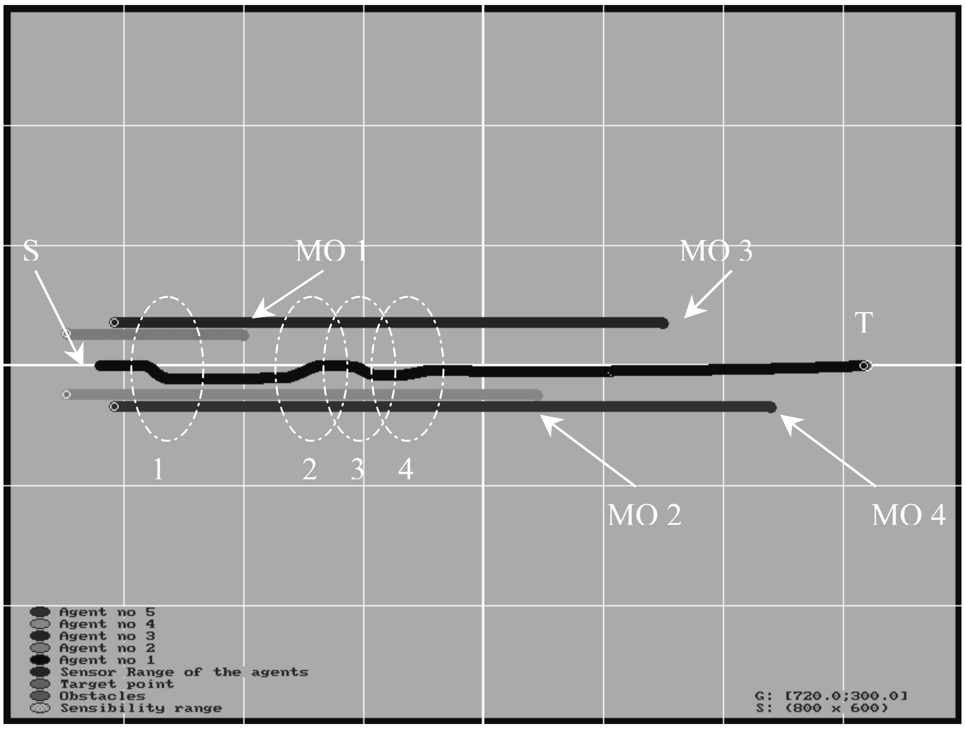


Figure 4.7: Experiment 4. An agent is placed into the environment (point S) and told to move to the other side (point T) by avoiding the moving obstacles.

During the experiment, the agent encountered four moving obstacles that are simulating humans or rolling balls. First confrontation happened between MO 1 and the agent (see circled area marked as 1 in Figure 4.7). The agent reacted quickly to avoid the obstacle. As expected, this reaction was fast since both the force and its derivative is used in control. Until next confrontation, the agent moved toward the target point T. Similar behavior is observed for MO 2, MO 3 and MO 4 confrontations. Moreover, we see clearly that the agent moves naturally and safely in the area where it encounters moving obstacles continuously.

4.7 Experiment 5

We stated that the proposed algorithm could be used for multiagent systems. Moreover, we showed in the previous experiments that the robot could safely move in areas where stationary or moving obstacles are present. In this experiment on the other hand, we will try to show that the agents can work in the same environment, each having different tasks, and still safely navigate.

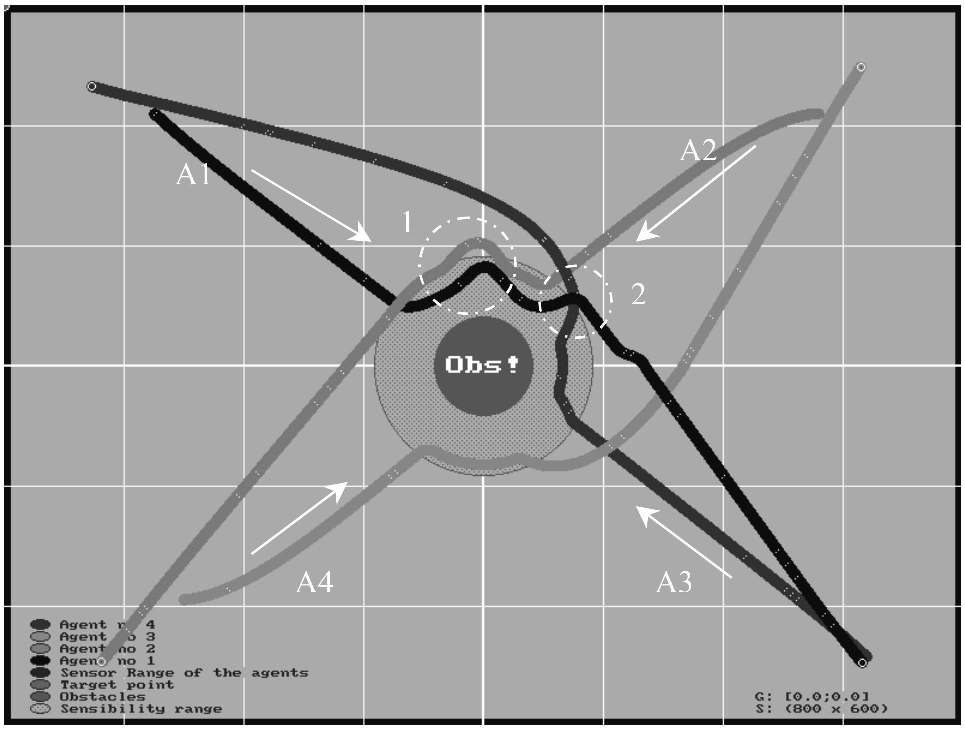


Figure 4.8: Experiment 5. Four agents are placed into the environment and told to move in different directions by avoiding obstacles.

The experimental setup and the results are shown in Figure 4.8 above. We placed four agents A1, A2, A3 and A4 to the environment where only one stationary obstacle exists. Each agent is told to move to the opposite side of the environment (shown with arrows).

First confrontation happened between A1 and A2 inside the sensibility range of the stationary obstacle. As clearly seen in the figure, although agents got close to each other, no collision occurred (white circle marked with 1). Then A1 encountered A3, again inside the sensibility range of the stationary obstacle (white circle marked with 2). Once more, both agents avoided collisions.

5. CONCLUSION

Although robots have shown their usefulness in industrial applications, a mobile robot operating in our everyday environment is a rare sight. This is caused by the strict requirements set for both the hardware and software components of the robot, especially for an autonomous mobile robot, together by the complex, dynamic environment and tasks of reasonable complexity.

In this work, we suggested a new approach for realization of reactive control of mobile robots. This new realization divides the control into layers. Each layer has its own task and goals to be accomplished. However, they do not rule over the robot directly. Each of them, processing available information and sensor data, produces a desired effect for robot motion. Then the outputs from each layer are collected in behavior arbitration. In the whole control, behavior arbitration is the only part that can directly influence the motion of the robot, except some possible user applications such as emergency stop.

In the contrary of similar methods, we did not integrated wheel speed controller to the layers but to the hardware. This way, any type and shape robot with necessary actuators and control that can use reference values of our control can be safely used.

Reaching to a specific point while avoiding obstacles and fallowing walls is a simple multi goal example for a mobile robot. The proposed approach for realization supports multi goals. Two basic goals for a mobile robot, goal tracking and obstacle avoidance, are already in the control, and working in harmony. Further goals can easily be defined in the appropriate layer of control. This way, the additions of the new layers to the mobile robot control will augment richness of the behaviors observed and with correct implementations will increase the performance observed.

Proposed approach for mobile robot control supports multi sensor perception. Each control layer uses only necessary sensor functions for its own process. Moreover, layers of control do not evaluate the signal coming directly from the sensors but get information of the environment from a preprocessor. This way the addition of new sensors does only affect the mentioned preprocessor that will convert the signals to the necessary form asked by the layers.

In the proposed control, the failure of a behavior results absence of a single behavior. Still, the rest of the control will be working. Nevertheless, as in all methods, if the input data, which is the perception, is corrupted either by a sensor or by preprocessor failure then most probably the whole system will fail. Similarly, in case of actuator, low-level motion controller or behavior arbitration failure also the system will tend to fail.

In the defined control, each part of the system is very transparent for other parts for information collection. Besides this transparency, the same parts of the system are all fully opaque for change in internal variables by other parts. This way, added layers to the existing system have access to the already calculated values. This is very advantageous when coordination between mobile robot's motion and additional hardware such as a manipulator. Moreover, the system saves time without calculating same data more than once. Consequently, requirement for computational power is relatively small.

Communication is also included in the proposed mobile robot control since it is a basic requirement for robots that are working as a part of a multiagent system. The aim is to facilitate the coordination between agents.

As a summary, the proposed control for general-purpose mobile robot is robust, expandable, supports multi sensor perception and multi goal realization. Moreover, robots do not have any handicap to be a part of a multiagent system.

Proposed control is tested on simulations, and different scenarios are studied. Especially, the cases that are problematic to many other approaches are investigated. Results are shown in the previous chapter. The simulation results confirmed the high performance of the method. Moreover, same results show that some of the drawbacks coming from the nature of the applied control are avoided. Furthermore, from these results, we can conclude that the proposed control is a potential alternative for mobile robots control operating in dynamic environments and as an agent in multiagent system.

REFERENCES

[1] R. C. Arkin, *"Behavior-based robotics"*. Cambridge, Mass.: MIT Press, 1998.

[2] R. C. Arkin, T. Balch, and E. Nitz, "Communication of behavioral state in multi-agent retrieval tasks" presented at IEEE Int. Conf. on Robotics and Automation, 1993.

[3] A. Attoui, *"Real-time and multi-agent systems"*. New York: Springer, 2000.

[4] J. Borenstein and K. Y., "The Vector Field Histogram-Fast obstacle avoidance for mobile robots", IEEE Transactions on Robotics & Automation, vol. 7, pp. 278-287, 1991.

[5] M. Bowling and M. Veloso, "Motion control in dynamic multi-robot environments" presented at IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA '99, 1999.

[6] V. Braitenberg, *"Vehicles: experiments in synthetic psychology"*, 7 ed: MIT Press, 2000.

[7] R. A. Brooks, "A robust layered control system for a mobile robot", MIT, Massachusetts A.I. Memo 864, Sep. 1985.

[8] R. A. Brooks, "A robot that walks; emergent behaviors from a carefully evolved network", Massachusetts Institute of Technology Artificial Intelligence Laboratory A.I Memo 1091, Feb. 1989.

[9] R. A. Brooks, *"Cambrian intelligence: the early history of the new AI"*. Cambridge, Mass.: MIT Press, 1999.

[10] L. Capozzo, G. Attolico, and G. Cicirelli, "Building low cost vehicles for simple reactive behaviors" presented at IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC'99, 1999.

[11] T. M. Chen and R. C. Luo, "Integrated multi-behavior mobile robot navigation using decentralized control" presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998.

[12] F. Davesne and C. Barret, "A reactive navigation method based on an incremental learning of tasks sequences" presented at First Workshop on Robot Motion and Control, RoMoCo'99, 1999.

[13] S. V. Emelyanov, *"Theory of variable structure systems"*. Moscow, Nauka, 1970.

[14] D. Eustace, D. P. Barnes, and J. O. Gray, "A behavior synthesis architecture for co-operant mobile robot control" presented at International Conference on Control, Control '94, 1994.

[15] J. Ferber, *"Multi-agent systems: an introduction to distributed artificial intelligence"*. Harlow, Eng.: Addison-Wesley, 1999.

[16] K. Furuta, "Sliding mode control of discrete-time systems", Systems and Control Letters, vol. 14, pp. 145-152, 1990.

[17] K. Furuta, A. Sano, and D. Atherton, *"State variable methods in automatic control"*: John Wiley & Sons, 1988.

[18] D. Gachet, M. A. Salichs, J. R. Pimentel, L. Moreno, and A. de la Escalera, "A software architecture for behavioral control strategies of autonomous systems" presented at International Conference on Industrial Electronics, Control Instrumentation and Automation, Power Electronics and Motion Control, 1992.

[19] H. Hexmoor, "Architectural issues for integration of sensing and acting modalities" presented at IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Intelligent Control (ISIC), 1998.

[20] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots" presented at IEEE International Conference on Robotics Automation, St. Louis, MO, 1985.

[21] U. Kotta, "Comments on the stability of discrete-time sliding mode control systems", IEEE Transactions Automatic Control, vol. 34, pp. 1021-1022, 1989.

[22] B. C. Kuo, *"Automatic control systems"*, 7 ed. New York: John Wiley, 1995.

[23] I. Kweon, Y. Kuno, M. Watanabe, and K. Onoguchi, "Behavior-based mobile robot using active sensor fusion" presented at IEEE International Conference on Robotics and Automation, 1992.

[24] I. D. Landau and C. B., "Design of multivariable adaptive model following control systems", Automatica, Pergamon Press, vol. 10, pp. 483-494, 1974.

[25] R. C. Luo and T. M. Chen, "Multiagent and event driven based dynamic collision avoidance for an autonomous mobile robot" presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998.

[26] C. Ma, W. Li, and L. Liu, "Mobile robot motion by integration of low- level behavior control and high level global planning" presented at IEEE International Conference on Systems, Man and Cybernetics, 1996.

[27] C. Milosavljevic, "General conditions for the existence of a quasisliding mode on the switching hyperplane in discrete variable", Automatic Remote Control, vol. 46, pp. 307-314, 1985.

[28] T. Mochiada, A. Ishiguro, T. Aoki, and Y. Uchikawa, "Behavior arbitration for autonomous mobile robots using emotion mechanisms" presented at IEEE/RSJ International Conference on Intelligent Robots and Systems 95, 'Human Robot Interaction and Cooperative Robots', 1995.

[29] J. Nielsen and G. Sandini, "Learning mobile robot navigation: a behavior-based approach" presented at IEEE International Conference on Systems, Man, and Cybernetics, 'Humans, Information and Technology', 1994.

[30] L. E. Parker, "A performance-based architecture for heterogeneous, situated agent cooperation" presented at AAAI Workshop on Cooperation Among Heterogeneous Intelligent Systems, 1992.

[31] P. Pirjanian, "Reliable reaction [mobile robot navigation]" presented at IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems, 1996.

[32] I. M. Rekleitis, G. Dudek, and E. E. Milios, "Multi-robot collaboration for robust exploration" presented at IEEE International Conference on Robotics and Automation, ICRA '00, 2000.

[33] M. Resnick, *"Turtles, termites, and traffic jams: explorations in massively parallel microworlds"*, 1st MIT Press ed. Cambridge, Mass.: MIT Press, 1997.

[34] R. M. E. Sabbatini, "W. Grey Walter: the Machina Speculatrix".

[35] C. Schlegel, "Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot" presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998.

[36] H. Sira-Ramirez, "Nonlinear discrete variable structure systems in quasi-sliding mode", International Journal of Control, vol. 54, pp. 1171-1187, 1991.

[37] K.-T. Song and C. C. Chang, "Reactive navigation in dynamic environment using a multisensor predictor", IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 29, pp. 870-880, 1999.

[38] A. Steinhage and R. Schoner, "The dynamic approach to autonomous robot navigation" presented at IEEE International Symposium on Industrial Electronics, ISIE' 97, 1997.

[39] W. C. Su, S. V. Drakunov, and U. Ozguner, "Sliding mode control in discrete time linear systems" presented at 12th IFAC World Congress, Sidney, Australia, 1993.

[40] W. C. Su, S. V. Drakunov, and U. Ozguner, "Implementation of variable structure control for sampled-data systems" presented at Workshop on Robust Control via Variable Structure & Lyapunov Techniques, Benevento, Italy, 1994.

[41] F. Tsuzuki and K. Sasaki, "A novel configuration of ultrasonic sensors for mobile robots" presented at IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94, 'Advanced Robotic Systems and the Real World', IROS '94, 1994.

[42] E. Tunstel and M. Jamshidi, "Fuzzy logic and behavior control strategy for autonomous mobile robot mapping" presented at IEEE World Congress on Computational Intelligence, Third IEEE Conference on Fuzzy Systems, 1994.

[43] V. I. Utkin, *"Sliding modes and their application in variable structure systems"*. Moscow, Nauka: (in Russian, English translation by Mir 1978), 1974.

[44] V. I. Utkin, *"Sliding modes in control and optimization"*. Berlin; New York: Springer Verlag, 1992.

[45] V. I. Utkin and S. Drakunov, "On discrete-time sliding mode control" presented at IFAC Symposium on Nonlinear Control Systems (NOLCOS), Capri, Italy, 1987.

[46] M. J. Wooldridge, *"An introduction to multiagent systems"*. New York: J. Wiley, 2001.

[47] C.-S. Wu, "Obstacle avoidance of autonomous mobile robots" presented at The Institute of Electrical and Electronics Engineers 31st Annual International Carnahan Conference on Security Technology, 1997.

[48] K. Young and S. Drakunov, "Sliding mode control with chattering reduction" presented at American Control Conference, Chicago, IL, 1992.