A BEHAVIOR BASED ROBOT CONTOL SYSTEM ARCHITECTURE FOR NAVIGATION IN ENVIRONMENTS WITH RANDOMLY ALLOCATED WALLS

A THESIS SUBMITED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

BERRİN ALTUNTAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

DECEMBER 2003

Approval of the Graduate School of Natural and Applied Sciences,

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ferda Nur Alpaslan
Supervisor

Examining Committee Members

Prof. Dr. Uğur Halıcı

Dr. Ayşenur Birtürk

Assoc. Prof. Dr. Ferda Nur Alpaslan

Assoc. Prof. Dr. Nihan Çiçekli

Assist. Prof. Dr. Bilge Say

ABSTRACT

A BEHAVIOR BASED ROBOT CONTOL SYSTEM ARCHITECTURE FOR NAVIGATION IN ENVIRONMENTS WITH RANDOMLY ALLOCATED WALLS

M.Sc. Department of Computer Engineering

Supervisor: Assoc. Prof. Ferda Nur ALPASLAN

December 2003, 62 pages

Integration of knowledge to the control system of a robot is the best way to emerge intelligence to robot. The most useful knowledge for a robot control system that aims to visit the landmarks in an environment is the enviromental knowledge. The most natural representation of the robot's environment is a map.

This study presents a behavior based robot control system architecture that is based on subsumption and motor schema architectures and enables the robot to construct the map of the environment by using proximity sensors, odometry sensors, compass and image. The knowledge produced after processing the sensor values, is stored in Short Term Memory (STM) or Long Term Memory (LTM) of the robot, according to the persistence requirements of the knowledge. The knowledge stored in the STM acts as a sensor value, while LTM stores the map of the environment. The map of the environment is not a priori information for the robot, but it constructs the map as it moves in the environment. By the help of the map constructed the robot will be enabled to visit non-visited areas in the environment and to localize itself in its internal world.

The controller is designed for a real robot Khepera equipped with the sensors required. The controller was tested on simulator called Webots version 2.0 on Linux operating system.

Keyword: Behavior-Based Robot Control System, Topological Map, And Visual Servo

ÖZ

RASTGELE YERLEŞTİRİLMİŞ DUVARLI ORTAMLARDA GEZİNİM İÇİN DAVRANIŞ TABANLI ROBOT KONTROL SİSTEM TASARIMI

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç.Dr. Ferda Nur ALPASLAN

Aralık 2003, 62 sayfa

Bir robota zeka eklemenin en iyi yolu robotun kontrol sistemine bilgi entegre etmektir. Bir ortamdaki dönüm noktalarını ziyaret etmeyi amaçlayan bir robot kontrol sistemi için en faydalı bilgi, çevresel bilgidir. Robotun çevresinin en doğal gösterimi haritadır.

Bu çalışma, bütünü-kapsama yapısını ve motor şema mimarilerini baz alarak, robotun, yakınlık sensörlerini, odometri sensörlerini, pusulayı, ve resimi kullanarak çevresinin haritasını oluşturmasını sağlıyacak sistem mimarisini sunar. Sensör değerlerini işleyerek oluşturulan bilgi, bilginin sürekliliği ile ilgili gereksinimlerine göre, Kısa Dönem Hafızada (KDH) veya Uzun Dönem Hafızada (UDH) saklanır. KDH'de saklanan bilgi sensör değeri gibi kullanılırken, UDH çevrenin haritasını saklar. Çevrenin haritası robota önbilgi olarak verilmemiştir, aksine robot, ortamda hareket ettikçe çevrenin haritasını oluşturur. Oluşturulan haritayı kullanarak, robotun, çevresindeki gidilmemiş alanları ziyaret etmesi sağlanacaktır.

Kontrol sistemi gerçek bir robot olan sensörlerle donatılmış Khepera için tasarlanmıştır. Kontrol sistemi Linux işletim sisteminde Webots Versiyon 2.0 benzeticisinde test edilmiştir.

Anahtar Kelimeler: Davranış tabanlı kontrol sistemleri, Topolojik haritalar, Görsel Gezinim.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Context and Motivation

A robot is a machine, that has the ability to extract information from its environment, and act in a meaningful and purposive manner [1]. Designing a robot has two aspects: The first one is the physical properties of the robot and the second one is the aim of the robot. The aim of the robot determines the physical properties of the robot. For example robots that need to move objects must be able to grasp them, robots that must function at night need sensors capable of operating under those conditions, and so on.

A robot control system should be designed and developed to achieve the aims of the robot. Behavior-based robotics and traditional approach of artificial intelligence are different approaches for the control systems of robots [1]. Traditional approach stands heavily on the representation of the world model and deliberative reasoning methods for robot control. On the other hand, behavior based approach supports reactive and independent units. Design architectures based on the behavior-based robotics approach are subsumption architecture and motor schema-based architecture.

A control system designed by using the deliberative methods stands heavily on the complete knowledge of its environment and uses this information to predict the outcome of its actions. This architecture assumes that the response time of the robot is not important and that the knowledge about environment of robot is consistent and reliable. Reactive behavior based robot controller is suitable for the systems for which timely robot action is important and the environment in which the robot will act is dynamic.

This study is inspired by behavior-based robotic architectures. Control system for a robot, that enables the robot not only to perform the simple tasks such as avoiding obstacles but also to perform complex tasks such as map construction and learning, is designed and implemented. The topological map constructed helps the robot to localize itself in its environment. The proposed control system exposes new behaviors as a result of interaction of simple behaviors. This is most attractive power of the behavior-based architecture.

1.2 Organization of Thesis

In Chapter 2, the foundation of behavior based robot control is given. The reactive and deliberative robot control systems are described and compared. Behavior based control system architectures that will be used in this study are described. The architectures studied are: subsumption and motor-schema based. Chapter 3 presents the definition of the problem, the solution to the problem by using the methodologies given in Chapter 2. Chapter 4 gives the conclusion and some future work to extend this study.

CHAPTER 2

BEHAVIOR-BASED ROBOTICS

2.1 Historical Background

The significant history associated with the origins of modern behavior-based robotics is important in understanding the current state of the art. In this section, important historical developments in three related areas: cybernetics, artificial intelligence and robotics are reviewed.

2.1.1 Developments in Cybernetics from Behavior Based Robotics Perspective

Cybernetics is combination of control theory, information science and biology that seeks to explain the common principles of control and communication in both animals and machines. [1] In 1953, W. Grey Walter developed a robotic design called Machina Speculatrix, which was used in the design of Grey Walter’s tortoise. Some of the principles captured in the design of tortoise are [2]:

1. Parsimony: Simple reflexes can serve as the basis for behavior.

2. Exploration: The system never remains still except when recharging.

3. Attraction: The system is motivated to move towards some environmental objects.
4. Aversion: The system moves away from certain negative stimuli.
5. Discernment: The system has the ability to distinguish between productive and unproductive behavior.

The robot exhibited simple behaviors such as seek for light, head toward weak light, back away from bright light, avoid obstacle and recharge battery. Recharge battery is emerged as result of coordination of other simple behaviors. When the charge of the robot is enough it seeks for light and moves towards weak light. However when the charge of the robot is low, it perceives the recharge station, which is a strong light, as weak light, and it moves towards the recharge station. As the robot is charged it perceives the recharge station as strong light and it moves away [2].

The behaviors of the tortoise are prioritized and this principle is called arbitration coordination mechanism. By the help of arbitration coordination mechanism, simple behaviors are combined to form complex behaviors such as moving safely around a room.

Braitenberg extended the principles of analog circuit behavior and designed systems, which uses inhibitory and exhibitory influences, directly coupling the sensors to motors. As a result of this simple design issue, complex behaviors are obtained from simple sesorimotor transformations. Assume the following configuration [3]:

- Two motors and two light sensors
- The effect of the left light sensor is connected to left motor and effect of right light sensor is connected to right motor
- The speed of the motor is proportional to the light received.

The robot with the given configuration will move away from light since the speed of motor near the light is greater then the speed of the motor, which is away from light.

By adding various nonlinear speed dependencies to a Vehicle, where the speed peaks somewhere between the maximum and minimum intensities, other interesting motor behaviors can be observed. This can result in oscillatory navigation between two different light sources or by circular or other unusual patterns traced around a single source [3].

2.1.2 Developments in Artificial Intelligence from Behavior Based Robotics Perspective

From artificial intelligence point of view, an intelligent robot would tend to build up within itself an abstract model of the environment in which it is placed. If it were given a problem it could first explore solutions within the internal abstract model of the environment and then attempt external experiments. This approach makes the artificial intelligence studies more dependent on the usage of representational knowledge and deliberative reasoning methods for robotics.

The inception and growth of distributed artificial intelligence paralleled these developments. In distributed artificial intelligence, it is assumed that simple agents through coordinated and concerted interaction, constructs highly complex and intelligent systems. Individual behaviors can be viewed as independent agents in behavior-based robotics, relating it closely to distributed artificial intelligence.

2.1.3 Developments in Robotics from Behavior Based Robotics Perspective

Many different techniques and approaches for robotic control systems have been developed. The spectrum for the robot control system is given in Figure 2-1.

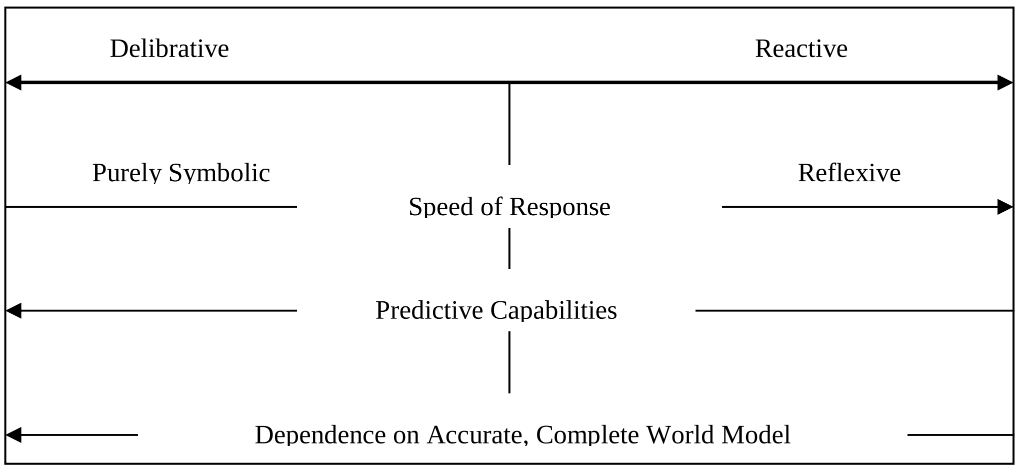


Figure 2-1 Robot Control Spectrum (Adapted from [1])

A robot employing deliberative reasoning requires relatively complete knowledge about the world and uses this knowledge to predict the outcome of its actions. This ability enables the robot to optimize its performance relative to its model of the world. Deliberative reasoning often requires strong assumption about this world model, primarily that the knowledge upon which reasoning is based is consistent, reliable and certain [1]. If the information the reasoner uses is inaccurate or has changed since obtained the result of the reasoning may be erroneous. Deliberative systems often uses hierarchical architecture. The subdivision of the layers in the hierarchy is based on funtionality and behaviors in the higher levels create subgoals for the behaviors in the lower levels. The representation of the world is shared in a global memory and the behaviors in any level reach the model of the environment when needed [4]. Deliberative reasoning systems aften have some common properties [1]:

- They have hierarchical structure.
- Communication and control occurs in a predictable and predetermined manner.

- Higher level in the hierarchy provides subgoals for lower levels.
- Time requirements are shorter and spatial considerations are more local at lower levels.
- They rely on symbolic representation of the wold.

Reactive systems tightly couples perception and action to produce timely respones in a dynamic and unconstructed world. In reactive systems an individual behavior is a stimulus response pair for a given environmental setting that is modulated by attention and determined by intention. Attention prioritizes tasks and focuses sensory resources and is determined by the current environmental context. Intention determines behaviors according to the robotic agent's internal goals. Key aspects of behavior based methodology include: situatedness, embodiment and emergence [5]. Situatedness stands for the fact that, the robot is an entity situated and surrounded by the real world. Embodiment says that the robot has a phsical presence and can not be simulated faitfully. Emergence suggects that intelligence is the result of the interaction of the robot with its environment.

2.2 Robot Behavior

A variety of approaches for behavioral choice and design have arisen. Some methods currently used for specifying and designing robotic behaviors are described below.

2.2.1 Behavior-Based Design Methodologies

The designer of the robot control system shall consider the following questions:

- What are the right behavioral building blocks for robotic system?
- What really is primitive behavior?
- How these behaviors are effectively coordinated?

- How are these behaviors grounded to sensors and actuators?

Unfortunately there are currently no universally agreed-upon answers to these questions. A variety of approaches for behavioral choice and design have arisen. The ultimate judge is the appropriateness of the robotic response to a given task and environment. Some methods are described below.

2.2.1.1 Ethologically Guided Design

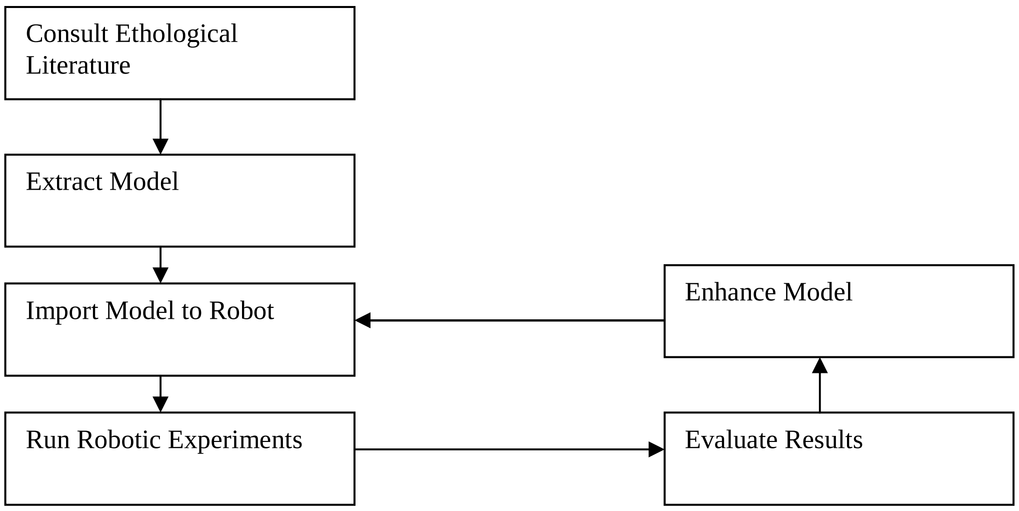


Figure 2-2 Design Methodology for Ethnologically Guided Systems (Adopted from [1])

Studies of animal behavior can provide powerful insight into the ways in which behavior can be constructed. In Ethologically Guided Design, a model is provided from scientific study, preferably with an active biolagical researcher. The animal model is then modified as necessary to realize computationally and is then grounded within robot's sensorimotor capabilies. The result from the robotic experiments are then compared to the results from the original biological studies, and the model is updated according to the results of the experiments [6]. All these process is depicted in Figure 2-2.

2.2.1.2 Situated Activity-Based Design

The design model is depicted in Figure 2-3.

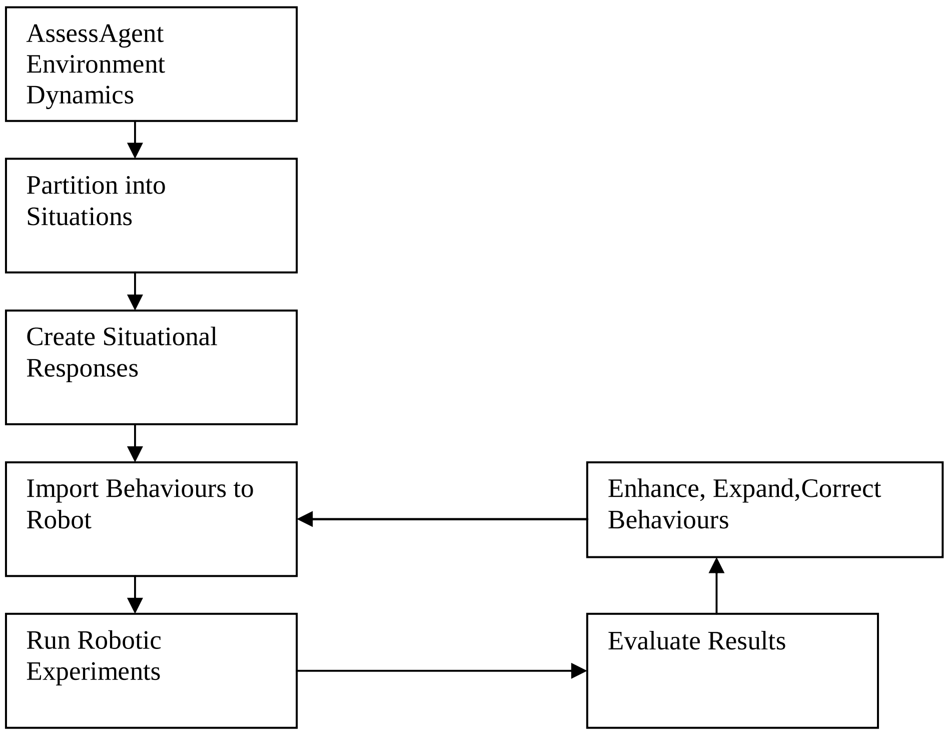

AssessAgent Environment Dynamics
Partition into Situations
Create Situational Responses
Import Behaviours to Robot
Run Robotic Experiments
Evaluate Results
Enhance, Expand,Correct Behaviours

Figure 2-3 Stituated Activity Design Methodology (Adopted from [1])

Situated activity based means that a robot’s actions are predicted upon the situations in which it finds itself. Hence perception problem is reduced to recognizing the situations the robot is in and then choosing one action to undertake. An important fact about this design is the number situations that the robot may be. If the probability of robot being in a situation is very low, then that situation should not be included in the system. Also the number of situations affects the performance of the system. Redundant situations shall be deleted from the system to increase the performance of the system [7].

The situations can be highly artificial and arbitrarily large in number. A coordination mechanism is needed to choose one of the candidate actions.

2.2.1.3 Experimentally Driven Design

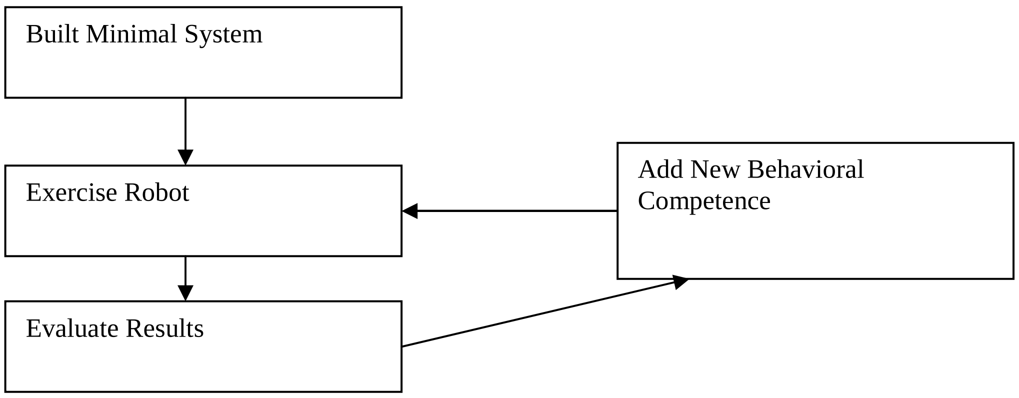


Figure 2-4 Experimentally Driven Design Methodology (Adopted from [1])

Experimentally driven behaviors are invariably created in a bottom-up manner. The basic operation premise is to endow a robot with a limited set of capabilities, run experiments in the real world, see what works and what does not, debug imperfect behaviors, and then add new behaviors iteratively until the overall system exhibits satisfactory performance [8]. The process for this design technique is depicted in Figure 2-4.

2.2.2 Expression of Robot Behavior

Most intuitive and least formal method of expressing the stimulus response relationship is Stimulus-Response (SR) Diagrams. SR diagram for a simple behavior consists of a stimulus, a behavior and a response pair. Any behavior can be represented as a generated response to a given stimulus computed by a specific behavior. A simple SR diagram is depicted in Figure 2-5.

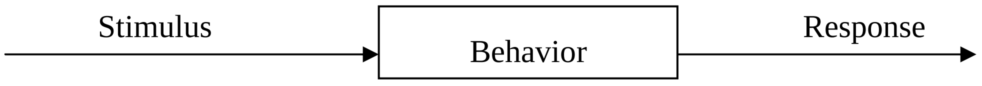


Figure 2-5 Simple SR Diagram (Adopted from [1])

Mathematical methods can be used to describe the same relationship using functional notation: b(s) = r, meaning behavior *b* when given stimulus *s* yields response *r*.[1]

Finite State Acceptor Diagrams (FSA) are useful when describing aggregations and sequences of behaviors. They make explicit the behaviors active at any time and the transitions between them. FSA are best used to specify complex behavioral control systems where entire sets of primitive behaviors are swapped in and out of execution during the accomplishment of some high level goal .[9] A sample FSA is depicted in Figure 2-6.

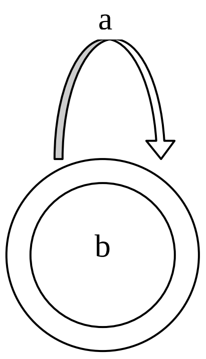


Figure 2-6 FSA For Simple Behavior (b = active behavior, a = all inputs and the arrow depicts the transition from one behavior to another when input a is received.) (Adopted from [9])

In this study, SR diagrams are used to represent behaviors.

2.2.3 Behavioral Encoding

To encode the behavioral response that the stimulus should evoke, we must create a functional mapping from stimulus plane to motor plane. An understanding of the dimensionality of a robot motor response is necessary in order to map stimulus on to it. For a response the two orthogonal components strength and orientation are important.

Strength denotes the magnitude of the response, which may or may not be related to the strength of a given stimulus. Even the stimulus having strength less then a threshold value does not cause a behavior to be active.

Orientation denotes the direction of action for the response. A behavior can be expressed as a triple *(S, R, β)* where S denotes the domain of all interpretable stimuli, R denotes the range of possible responses, and β denotes the mapping $\beta : S \rightarrow R$. The instantaneous response *R* of a behavior based reactive systen can be expressed as a six-dimensional vector consisting of six subcomponent vectors. Each of the subcomponent vectors encodes the magnitude of translational and rotational responses for each of the six degrees of freedom of motion of general mobile robot. If the robot is unconstraint, it has six Degrees Of Freedom (DOF), r – [x, y, z, α, β, ϕ] where the first theee components represent three translational DOF in cartesian coordinates. The last three components encode the three rotational DOF.

For ground based mobile robots, the dimensionality is often considerably less then 6 DOF. For example, a robot that moves on flat ground and can rotate only about its central axis has only three degrees of freedom, r = [x, y, α].

Another factor that limits the realization of a generated behavioral response is the robot’s non-holonomicity. A non-holonomic robot has restrictions in the way it can move, typically because of the kinematic or dynamic constraint on the robot, such as limited turning abilities or momentum at high velocities. The constraints imposed by non-holonomic systems can be dealt with either during the generation of response, by including them within the function β, or after *r* has been computed, translating the desired response to be within the limitaions of the robot itself.[10]

Each individual stimulus *s* is represented as binary tuple *(p, λ)* having both a particular type or perceptual class *p* and a property of strength λ.The stimulus strength λ can be discrete or real values and continuous. A thershold value is defined for a given perceptual class *p* above which a response is generated. Often the stregth of the input stimulus determines whether or not to respond and the magnitude of the response. Certain stimuli can provoke a motor response while some others which are called perceptual triggers change the behavioral configuration of the robot.[1]

Each individual active behavior produces a functional mapping between the stimulus domain and response range that defines a behavioral function β where $\beta(s) \to r$. β must be defined over all relevant *p* in *S*. The functional mapping between strength of stimulus and the magnitude and direction of robotic motor response defines the design space for a particular robotic behavior [1].

2.2.3.1 Discrete Encoding

In discrete encoding, the Behavioral Function β, is defined for all relevant perceptual class *p* and produce behaviors from discrete set of all possible behaviors. Situated action is an example for discrete encoding. For these cases, β consists of finite set of situation, response pairs. Sensing provides a clue for finding the appropriate situation. Rule-based systems are used in which the rules in *IF antecedent THEN consequent* form represents β. The *antecedent* consists of a list of preconditions that must be satisfied in order for the rule to be applicable and the *consequent* contains the motor response [11]. The finite set of rules corresponds to the discrete set of possible responses. More then one rule may be applicable for any given situation. A strategy should be designed for conflict resolution.

2.2.3.2 Continuous Functional Encoding

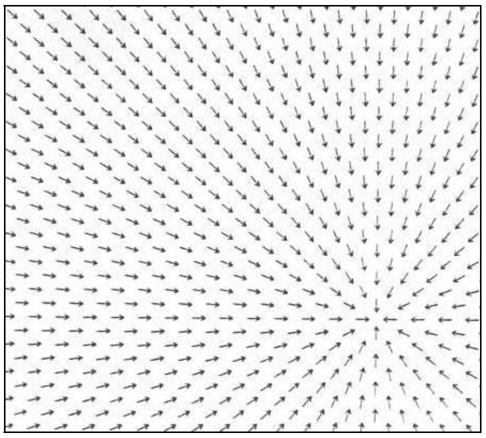
Continuous response allows a robot to have an infinite space of potential reactions to its world. Instead of having an enumerated set of responses, a mathematical function transforms the sensory input into a behavioral reaction. One of the most common methods for implementing continuous response is based on technique referred to as potential fields method.

This method generates a field representing a navigational space based on an arbitrary potential function. The classic function used is that of Coulomb’s electrostatic attraction; where the potential force drops off with the square of the distance between the robot and objects within the environment. Goals are treated as attractors and obstacles are treated as repulsors [12].

In Figure 2-7 potential fields for a goal and obstacle is depicted. Vector Summation calculates the path of the robot that operates in an environment with a goal object and an obstacle. The resultant path is depicted in Figure 2-8. The robot moves away from the obstacle but towards the goal object.

2.2.4 Assembling Behavior

Most of the robot control systems consist of multiple behaviors. In these systems, more then one behavior can be activated at a time. There shall be a mechanism to determine the output of the overall system when such abnormal situations arises. The mechanism to solve conflicts are coordination fuction. There are mainly two types of coordination function: Competitive Methods and Cooperative Methods [13].


(a)

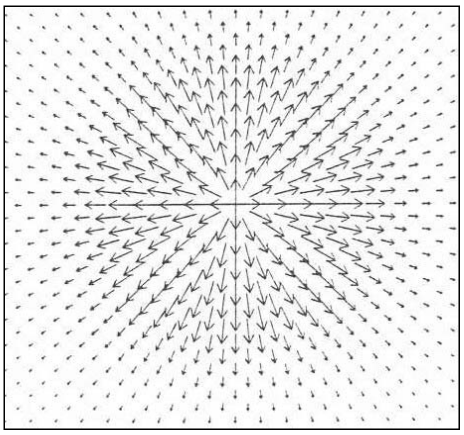
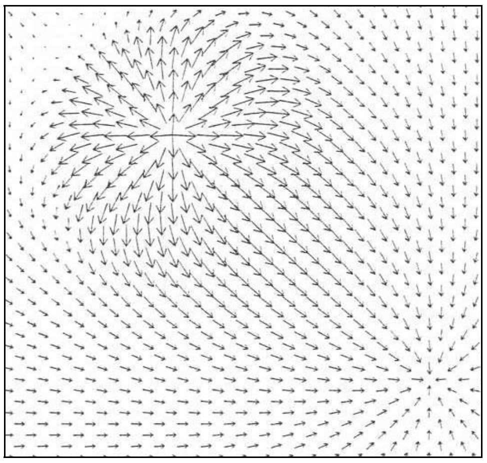

(b)

Figure 2-7 (a) Potential Field for a Goal (b) Potential Fields for an Obstacle (Adapted from [1])


(a)

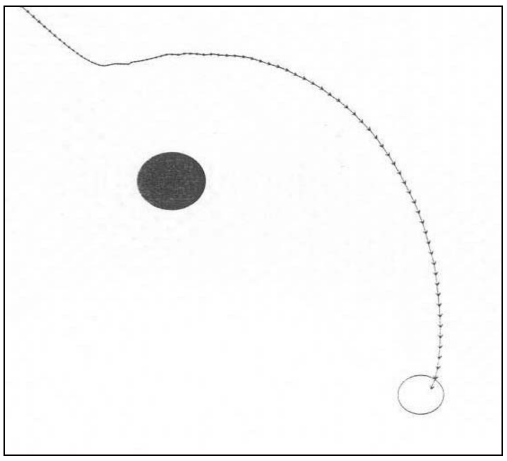

(b)

Figure 2-8(a) Vector summation of Vector Fields Given in Figure 2-7 (b) Path of the Robot (Adapted from [1])

2.2.4.1 Competitive Methods

Competitive methods provide a means of coordinating behavioral response for conflict resoulution. The coordinator can be viewed as a winner-take-all network in which a single response for the winning behavior out-muscles all others and is directed to the robot for execution. There are mainly three methods that can be studied under this title: Priority-Based Coordination, Action-Selection Coordination and Voting-Based Coordination.

In Priority-Based coordination, there exists an arbiter that selects a single behavioral response. The arbitration function can take the form of a fixed prioritization network in which a strict behavioral dominance hierarchy exists and, as a result if an action in higher prioritization level perceives stimulus then it is executed [14]. The structure of Prriority-Based Coordination is depicted in Figure 2-9.

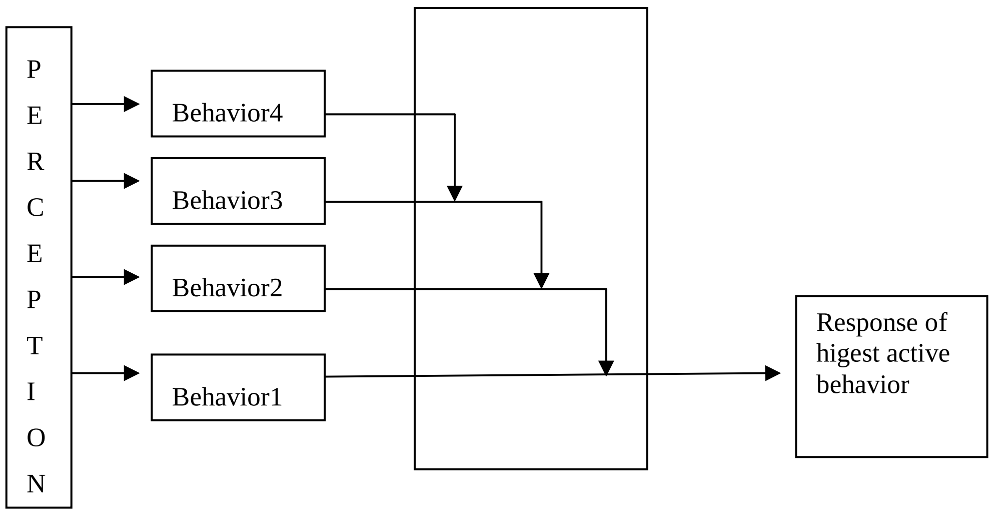


Figure 2-9 Prriority-Based Coordination (Adapted from [1])

In Action-Selection coordination method, which is depicted in Figure 2-10, all the behaviors send their response to the coordinator; the coordinator selects the behavior with the maximum response [15].

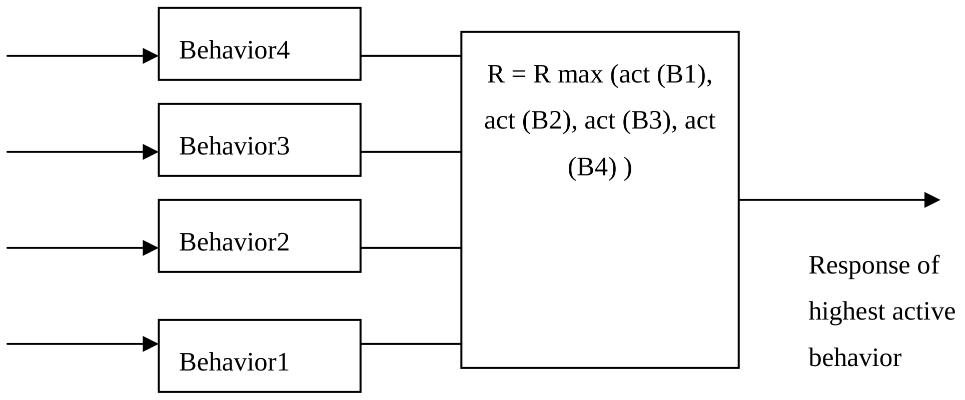

Behavior4
Behavior3
Behavior2
Behavior1
R = R max (act (B1), act (B2), act (B3), act (B4))
Response of highest active behavior

Figure 2-10 Action Selection Coordination (Adapted from [1])

In Voting-Based Coordination, every behavior gives a vote to a response and the response with the heights vote is executed. Here instead of each behavior's being encoded as s set of rule-based responses, each behavior cats a number of votes toward a predefined set of discrete motor responses. The structure of Voting-Based Coordination is depicted in Figure 2-11.

2.2.4.2 Cooperative Methods

Cooperative methods provide an alternative to competitive methods such as arbitration. Arbitration requires that a coordination function serving as an arbiter select a single behavioral response [1]. Behavioral Fusion via Vector Summation can be studied under this title.

In Behavioral Fusion via Vector Summation, which is depicted in Figure 2-12, the response of each behavior is received and averaged; the average value of the responses of all behaviors is returned as response.

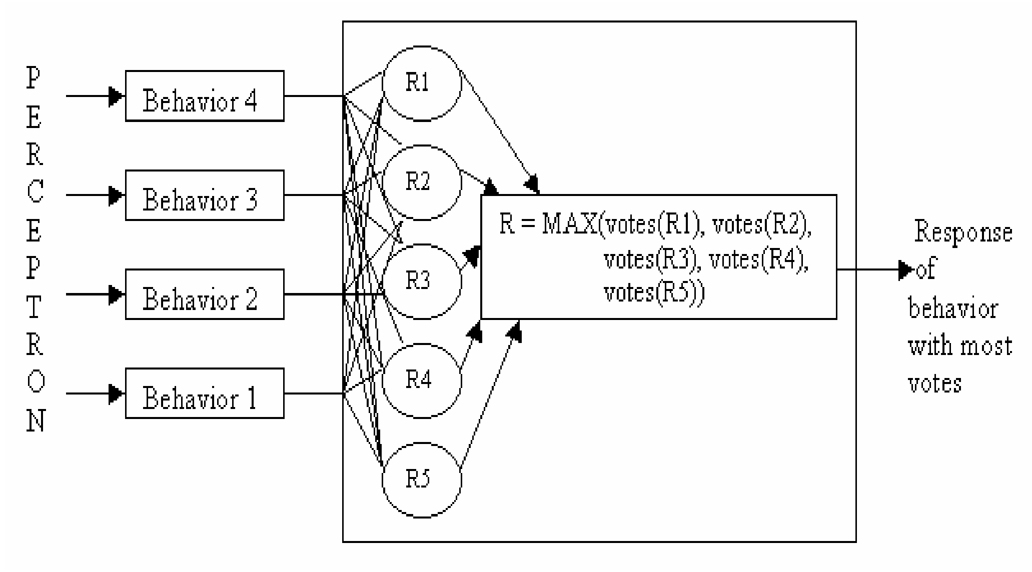

PERCEPTRON
Behavior 4
Behavior 3
Behavior 2
Behavior 1
R1
R2
R3
R4
R5
R = MAX(votes(R1), votes(R2), votes(R3), votes(R4), votes(R5))
Response of behavior with most votes

Figure 2-11 Voting-Based Coordination(Adapted from [1])

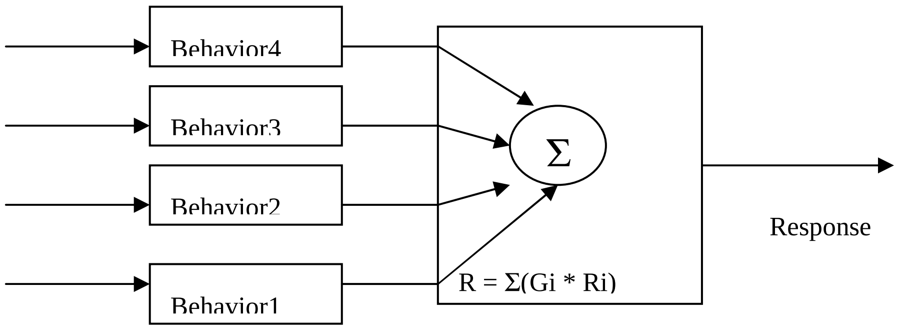


Figure 2-12 Behavioral Fusion via Vector Summation(Adapted from [1])

2.3 Behavior Based Architectures

Behavior based architectures can be seen as programming languages. There are many programming languages ranging from machine languages to very high-level languages. From the computability point of view, if a programming language has basic constructs for sequencing, conditional branching and iteration then it can compute the entire class of computable functions [1]. However Cobol programming language is not suitable for robot control problem. The problem determines the programming language that will be used.

Likewise, behavior based architectures, because of their different means of expressing behaviors and the set of coordination functions they offer, provide diversity to the robot control system designer. Each approach has its own strengths and weaknesses in terms of what it is best at doing or where it is most appropriately applied. All behavior-based architectures couple sensing and action, avoid representational symbolic knowledge of its environment and decompose the system into contextually meaningful units. They differ in the granularity level of the units, basis for behavioral decomposition, response encoding model and coordination method.

In this section Subsumption Architecture and Motor Schemas will be studied.

2.3.1 Subsumption Architecture

Subsumption architecture is developed against the traditional artificial intelligence's sense-plan-act paradigm. Task-achieving behaviors in the subsumption architecture are represented as separate layers. As opposed to the traditional paradigm, the actions in different layers don't take place sequentially but in parallel [14]. Augmented Finite State Automata (AFSM) represents each behavior. Because of the fact that the stimulus and response can be inhibited or suppressed, it is called augmented finite state automata. Each AFSM encapsulates a particular behavior transformation function, which transforms stimulus to response. The architecture of AFSM is given in Figure 2-13.

Each AFSM performs an action and is responsible for its own perception of the world. This model enables each behavioral layer to be mapped onto its own processor, which in turn supports parallel processing of each AFSM [16].

AFSM are connected to each other in a layered structure such that, simpler actions are in the lower levels of the architecture. The coordination between the actions on different layers is base on the fact that complex actions subsume simpler behaviors. Coordination in subsumption has two primary mechanisms:

- Inhibition: Signal is prevented to reach the actuators.
- Suppression: Current signal is replaced with suppressing message.

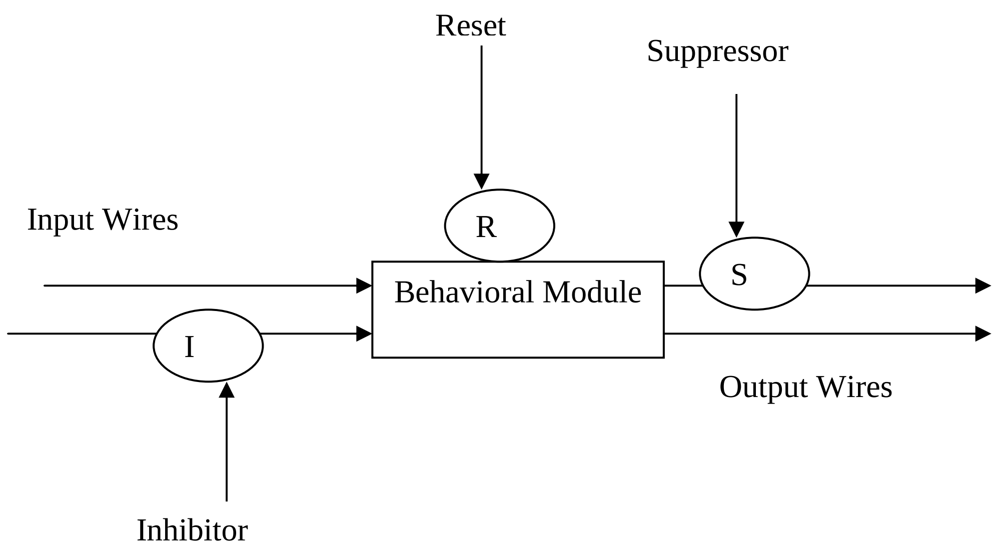


Figure 2-13 AFSM Used within Subsumtion Architecture (Adapted from [1])

The basic procedure to build systems by using subsumption architecture is as follows [17]:

- Qualitatively specify the behavior needed for the task
- Decompose and specify the robot's independent behaviors
- Determine behavioral granularity.

2.3.2 Motor Schemas

Motor Schema approach is strongly motivated by the biological sciences. A schema is the basic unit of behavior from which complex actions can be constructed; it consists of the knowledge of how to act or perceive as well as computational process by which it is enacted. Arkin [18, 19, 20], addressed the implication of the schema theory for autonomous robotics:

1. Schemas provide large grain modularity for expressing relationships between motor control and perception.

2. Schemas act concurrently as individual distributed agents in a cooperative or competing manner thus are suitable for distributed processing architectures.

3. Schemas provide a set of behavioral primitives, which can be combined to create more complex behaviors.

4. Cognitive and neuroscientific models support this approach.

The aim of the schema-based robotics is to provide behavioral primitives that can act in a distributed and parallel manner to yield intelligent robot action in response to environmental stimuli.

The motor schema method differs from other behavioral approaches in several significant ways:

1. Behavioral responses are represented as vector, which is generated by using potential field method.

2. Coordination is achieved by summing up the vectors.

3. There is not a hierarchical structure for coordination.

4. Pure arbitration is not used; instead, each behavior can contribute in varying degrees to the robot's overall response.

2.3.2.1 Schema Based Behaviors

Motor schema based behaviors are relatively large grain abstractions reusable over a wide range of circumstances.

A perceptual schema is embedded with each motor schema. The perceptual schemas provide the environmental information specific for a particular behavior. Perception is conducted in a need-to-know basis: individual perceptual algorithms provide the information necessary for a particular behavior. Perceptual schemas are defined recursively, that is, perceptual subschema can extract pieces of information that are subsequently processed by another schema in to more behaviorally meaningful unit. By this way multiple perceptual schemas can be used for a single motor schema. Each motor schema has as output an action vector, which defines the way the robot move in response to the perceived stimuli.

Figure 2-14 shows a sample architecture designed by using motor schema architecture approach:

Sample motor schemas are followings:

1. Move-ahead: move in a particular compass direction.
2. Escape: Move away from the projected intercept point between the robot and an approaching predator.
3. Avoid-past: Move away from recently visited areas.

2.4 Visual Servo Control

Vision is a useful robotic sensor since it mimics the human sense of vision. Visual feedback loop can be used to correct the position of a robot to increase task accuracy. The task in visual servoing is to use visual information to control the pose of the robot's motor relative to a target object or set of target features. Visual servoing architectures are divided into two according to the answer they give to the following question: Is the error signal defined in 3D (task space) coordinates or directly in terms of image features [21]. In position-based control, features are extracted from the image and used in conjunction with a geometric model of the target and the known camera model to estimate the pose of the target with respect to the camera. In image based servoing, control values are computed on the basis of image features directly.

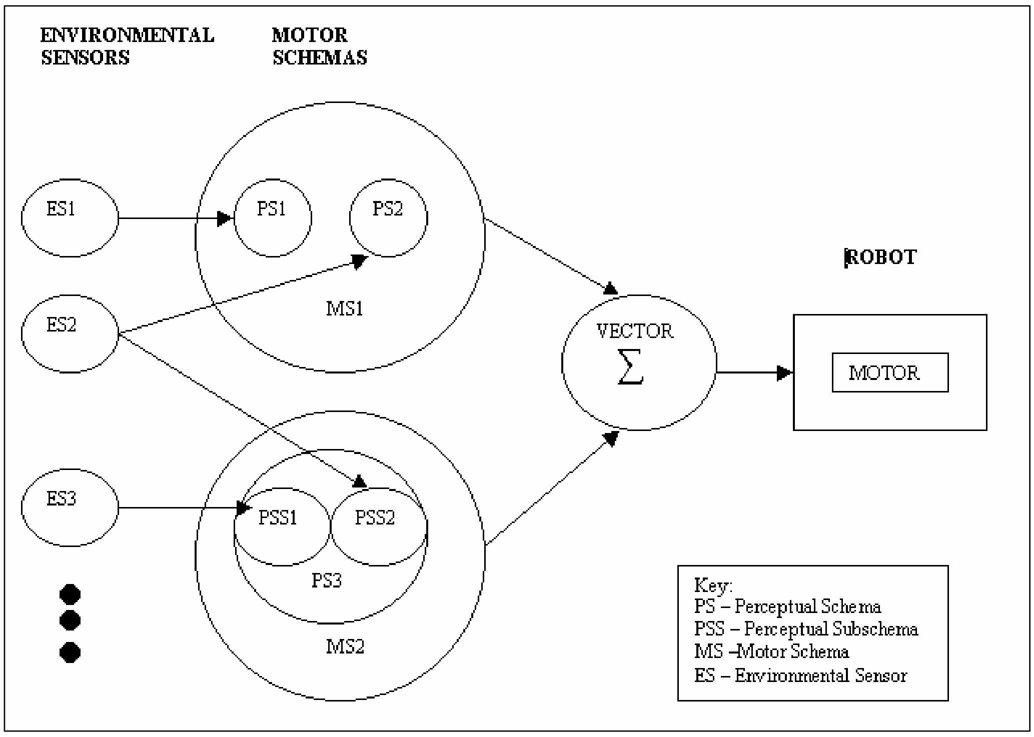


Figure 2-14 Perception Action Schema Relationship (Adapted from [1])

2.4.1 Position Based Visual Servo Control

In position based visual servoing, features are extracted from the image and used to estimate the pose of the target with respect to the camera. Using these values, an error between current and desired pose of the robot is defined in the task space. In this way, position-based control nearly separates the control issues, namely the computation of the feedback signal from estimation problems involved in computing position and pose from visual data [21].

The formal notion of a positioning task is given as: A positional task is represented by a function $E: \tau \rightarrow \Re$. [21] This function is referred to the kinematics error function. A positioning task is fulfilled with the end effectors in pose χ_c if $E(\chi_c) = 0$. Once suitable kinematics error function has been defined and the parameters of the function are instantiated from the visual data, a regulator is defined that reduces the estimated value of the kinematics error function to zero. This regulator produces at every time instant a desired end-effectors velocity screw *u* that is sent to robot control system. The process is to first determine the relative motion that would fulfill the task, and then to write a control low would produce the motion.

2.4.2 Image Based Visual Servo Control

In image based visual servo control, the error signal is defined directly in terms of image feature parameters. The changes in the image feature parameters are related to the changes in position of the robot. Image Jacobean Matrix captures these relationships.

Motion of the manipulator causes changes to the image observed by the vision system. Thus the specification of image-based visual servo task involves determining an appropriate error function *e*, such that when the task is achieved, $e = 0$. This can be done via a "teach by showing" approach in which the robot is moved to a goal position and the corresponding image is used to compute a vector of desired image feature parameters. [21]

Solving the image based motion planning problem is usually a two-step process: first the robot transfers the sensor features back to pose information, then it makes a motion plan in the pose space based on this information [22].

2.4.3 Hough Transform

In Visual Servo Control systems, image features are extracted from the image to be used to estimate the pose of the target with respect to the camera. In this study, the visual features used are the rectangles in the image that corresponds to walls in the environment. To extract rectangles from the image first the lines shall be identified.

A line can be represented by the following equation:

$$x * \cos\alpha + y * \sin\alpha = \rho$$

Consider a point (xi, yi) and the given equation. Infinitely many lines passes through (xi, yi), but they all satisfy $x*\cos\alpha + y*\sin\alpha = \rho$ equation for varying values of α and ρ. However αρ plane (also called parameter space) yields the equation a single line for a fixed pair (xi, yi). Furthermore, a second point (xj, yj) also has a line in parameter space associated with it, and this line intersects the line associated with (xi, yi) at αi, ρi. In fact all points contained on this line have lines in parameter space that intersects at (αi, ρi).

The computational attractiveness of the Hough Transform arises from subdivision of the parameter space in to so called accumulator cells. The cell at coordinate (i, j) , with accumulator value A(i,j) , corresponds to the square associated with parameter space coordinates (ai, bj). Initially, these calls are set to zero. Then, for every point (xk, yk) in the image plane, we let the parameter α equal each of the allowed subdivision values on the α axis and solve for the corresponding ρ using the $x*\cos\alpha + y*\sin\alpha = \rho$ equation. The resulting ρ's are then rounded of to the nearest allowed value in the ρ axis. If a choice of αi result in solution ρi, we let A(αi, ρi) = A(αi, ρi) +1. At the end of this procedure, a value of M in A(i,j) corresponds to M points in the xy plane. The cells with highest accumulator values corresponds to the parameters of the lines in the image.

In Figure 2-15 the normal representation of a line and the accumulator matrix for the line in pθ plane is depicted.

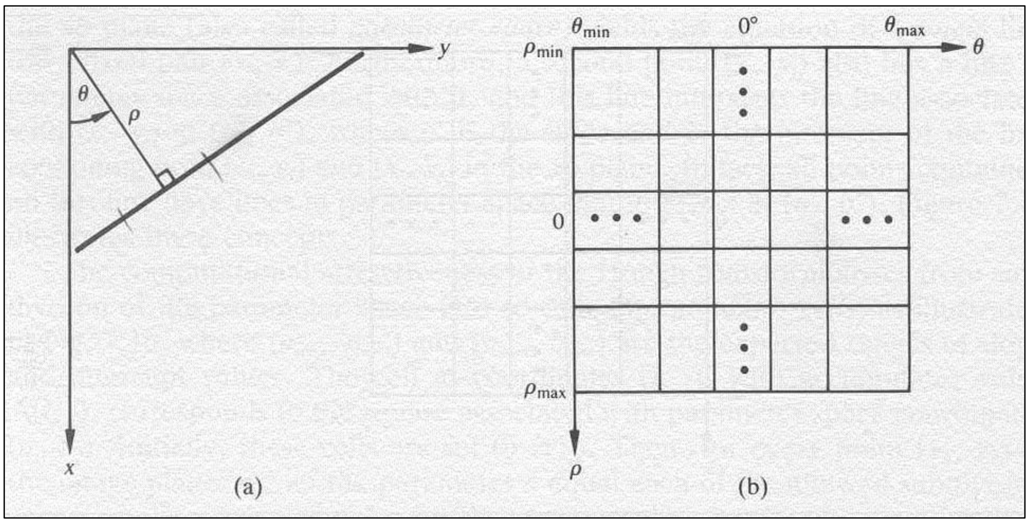


Figure 2-15(a) Normal Representation of line; (b) quantization of the pθ plane into cells (Adapted from [23])

2.5 Representational Issues in Behavior-Based Systems

Representational knowledge can be viewed as an impediment to the robot control. Knowledge representation of various forms can be introduced into reactive robotic systems at behavioral level. The most useful knowledge for the robot is the map of the environment in which it operates. The robot using a map can locate itself in its internal world and can have knowledge of its environment.

2.5.1 Map Representation Methods

The most natural representation of the robot's environment is a map [24]. In general, map representation methods can be divided into two main groups: those that rely primarily on an underlying metric representation and those that are topological.

The most straightforward spatial representation is to sample discretely the two- or three-dimensional environment to be described. The simplest method is to sample space in cells of a uniform grid. The main advantage of a regular lattice representation is its extreme generality: no strong assumptions are made regarding object type, and thus these grids can represent anything. The main disadvantage of this representation is that grid resolution is limited by the cell size.

One alternative to the storage problem is to represent space using cells of nonuniform shape and size. The most common of these methods is the quadtree, which is a recursive data structure for representing a square two-dimensional region. Cells that are neither uniformly empty nor full are subdivided into four equal subparts. Subparts are subdivided in turn until they are either uniformly empty or full or until a resolution limit is reached. In general, the number of cells varies roughly with the areas of the obstacles being described. Thus, for environments where most of the space is free or occupied, quadtree-like representations are very suitable.

Geometric maps are those made up of discrete geometric primitives: lines, polygons, points, polynomial functions and so forth. Such maps are characterized by two key properties: the set of basic primitives used for describing objects, and the set of composition and deformation operators used to manipulate objects. The primary shortcoming of geometric model-based representations relates to the fact that they can be difficult to infer reliably from sensor data.

Geometric representations rely on metric data as the core of the representation. Unfortunately these metric data are likely to be corrupted by sensor noise at the very least. To avoid reliance on error-prone metric data, a non-metric topological representation may be used. The key to a topological relationship is some explicit representation of connectivity between regions or objects. In its purest form this may involve a complete absence of metric data. A topological representation is based on an abstraction of the environment in terms of discrete places with edges connecting them. In some cases, the edges have length and the edges are oriented with respect to the nodes.

Typically, a robot’s environment is modeled as a graph whose vertices correspond to landmarks, which can be recognized using sensors, placed within the environment.

Each vertex corresponds to one of the unique landmarks, whereas edges correspond to known straight paths between landmarks. Each edge may be labeled with the distance that needs to be traveled along the edge to arrive at the next landmark. Although the robot has no real understanding of the geometric relationship between locations in the environment, the representation does encode sufficient information for the robot to conduct point-to-point navigation.

2.5.2 Localization

For numerous tasks a mobile robot needs to know where it is either on an ongoing basis or when specific events occur. This problem has many different connotations. In the strongest sense, knowing where the robot is involves estimating some global representation of the space. This is usually referred to as strong localization. The weak localization problem, in contrast, involves merely knowing if the current location has been visited before. Between the extremes of the weak localization and strong localization problems exist a continuum of different problem specifications that involve knowing where the robot is or estimating the robot's pose.

In certain circumstances it may be necessary to infer the robot's position without an a priori estimate of its location. This type of positioning is referred to as global localization. A more common version of the localization problem is the need to refine an estimate of the robot's pose continually. This task is known as pose maintenance or local localization. A key step in the process of performing either local or global localization involves matching the set of current observations to some established map. Standard matching methods can be broadly classified into the following categories:

- Data-data matching. Directly matching the current raw data with predicted raw data (extracted from the map either by predictive modeling or using stored data sets).

- Data-model matching. Matching the observed data to more abstract models stored in the map (based on model of how models and data are associated).

- Model-model matching. Matching models stored in the map to models generated from current observations.

In general matching with raw data can reduce dependence on a priori assumptions about the environment.

Most devices for measuring position and distance are relative measurement tools. By counting the number of rotations executed by a vehicle’s drive wheels, for example, and using knowledge of the wheel’s size and the vehicle’s kinematics, an estimate of the rate of position change can be obtained. Computing absolute coordinates thus involve the integration of such local differential quantities. This technique is known as dead reckoning, and when using odometers it is called odometry.

CHAPTER 3

DESIGN OF A BEHAVIOR BASED ROBOT CONTROL SYSTEM FOR ENVIRONMENTS WITH RANDOMLY ALLOCATED WALLS

3.1 Problem Definition

The problem can be stated as designing a robot control system that is based on the approaches described in Chapter 2. The robot with the designed behavior based control system should be able to:

- Move in environments with randomly allocated walls safely.
- Construct topological map of the environment.
- Learn topological map to avoid past.
- Localize itself in its internal representation of the world.

The controller should not violate the basic principles of behavior based robot control paradigm. The experiments were conducted in a simulator environment named

Webots 2.0 with the simulated robot called Khepera. The robot uses proximity sensors, odometry sensors, a color camera, and a compass to conduct its overall goal.

3.2 Environment Definition

The robot control systems cannot be designed without considering the properties of the world. The approaches used in the robot control system design is chosen according to what the robot will do and in what kind of environments the robot will operate. There are no dynamic objects in the environment. The only objects in the world are walls. The walls are located in the world at random positions, in random orientations. The colors or the intensities of the walls, which are neighbor, are supposed to be different than each other for edge detection algorithm to work properly. An example environment is shown in Figure 3-1.

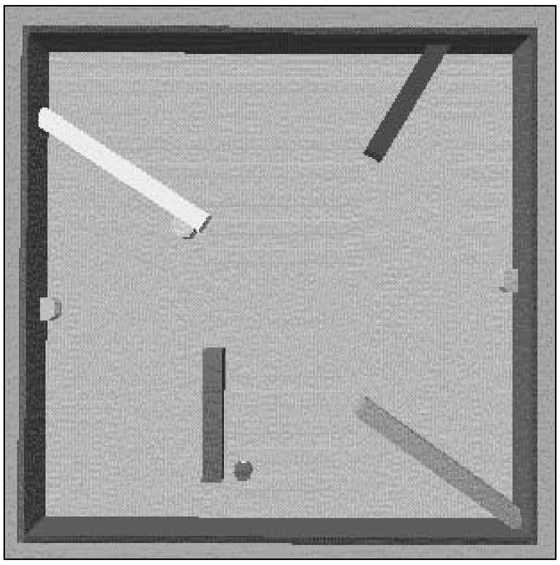


Figure 3-1 Example World Model

3.3 Simulated Robot’s Properties

Khepera is a miniature mobile robot with functionality similar to that of larger robots used in research and education. Khepera was originally designed as a research and teaching tool for a Swiss Research Priority Program at EPFL in Lausanne. It allows real world testing of algorithms developed in simulation for trajectory planning, obstacle avoidance, pre-processing of sensory information, and hypotheses on behaviour processing, among others [25].

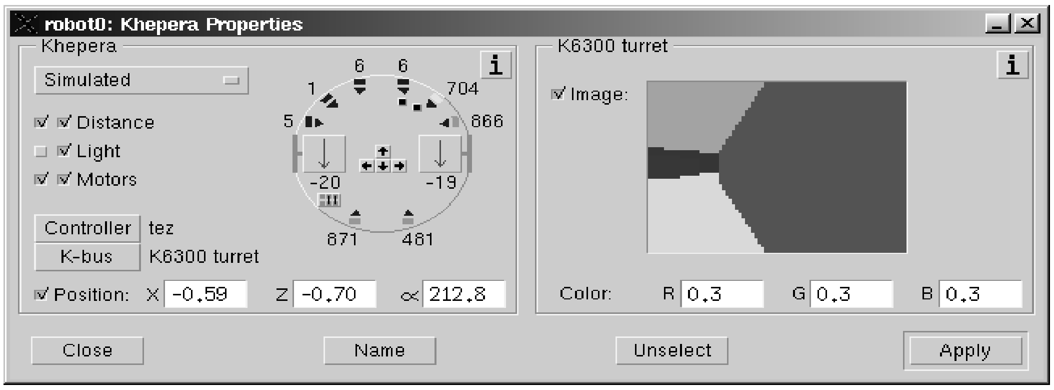
Very modular at both the software and hardware level, Khepera has a very efficient library of on-board applications for controlling the robot, monitoring experiments, and downloading new software. A large number of extension modules make it adaptable to a wide range of experimentation.

Basic Khepera robot has 8 infra-red sensors used to measure the distance from obstacles (distance ranges from 0 to 1023), the same 8 infra-red sensors used to measure the level of ambient light (light values range from 0 to 512), position sensor which is the value of the incremental encoder on each wheel (0.08mm for each pulse) and velocity sensor for each motor wheel (ranges from –20 to 20). Numerical values for the sensors is as given below:

Extentions can be plugged onto the robot. Extentions for Khepera robot can be a camera for vision or a gripper to hold objects in the environment. K6300 extention turret is used to obtain the vision of the environment in whcih the robot operates. This turret holds a 2D camere digital device. The features of K6300 turret is as follows [25]:

- 80x60 pixels resolution
- horizontal view of 60°
- vertical view of 60°
- 24 bit color depth

The properties of the robot can be viewed from WEBOTS simulator as given in Figure 3-2.


robot0: Khepera Properties

Khepera

Simulated

☑ ☑ Distance
☐ ☑ Light
☑ ☑ Motors

Controller tez
K-bus K6300 turret

☑ Position: X -0.59 Z -0.70 ∝ 212.8

6 6 1 704 5 866 -20 -19 871 481

K6300 turret

☑ Image:

Color: R 0.3 G 0.3 B 0.3

Figure 3-2 Khepera Properties Window

The image on the right side displays what the robot's camera receives from the environment, while the schema on the left displays the sensor values of eight proximity sensors.

3.4 System Overview

In the design of the behavior-based control system subsumption and motor-schema architectures are used to correlate some sensor data with actions and coordination. The highest level architectural view of the the control system is given in Figure 3-3.

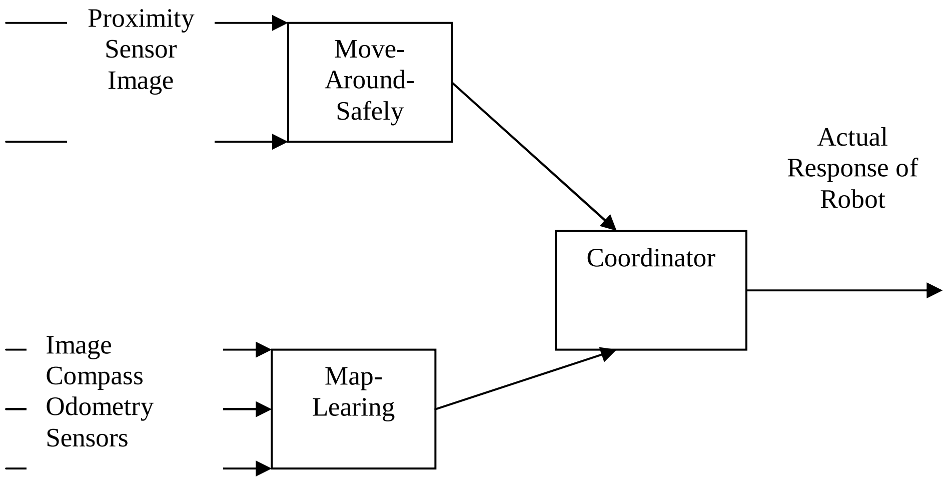


Figure 3-3 Highest Level System Architecture

3.4.1 Move-Around-Safely

This behavior provides the robot to move in the environment in a safe manner. It contains the a set of primitive behaviors. The primitive behaviors are coupled with and result in complex and useful behaviors. The primitive and complex behaviors observed are:

- **Avoid-Obstacle:** This behavior enables the robot to move in the environment by issuing the Braitenberg methodology. In Braitenberg methodology, the infrared values of the sensors are connected to the motors of the robot. When proximity sensor values on the left sensors become higher then the right sensor values, the speed of the left motor becomes higher than the right motor speed, and as result the robot moves away from the obstacles in a safe manner.

- **Move-Ahead:** This behavior enables the robot to move in a particular direction. This behavior gives each motor the same speed (20 mm/s) with the duration of 16 ms.. This behavior does not use any sensor information. It is basically coupled with other behaviors to obtain more complex behaviors.

- **Escape:** Move away from the projected intercept of point between the robot and walls. This behavior gives –20mm/s and –19mm/s speeds to left motor and right motor according to the position of the walls and previous direction of the robot.

The structure of Move-Around-Safely behavior is depicted in Figure 3-4.

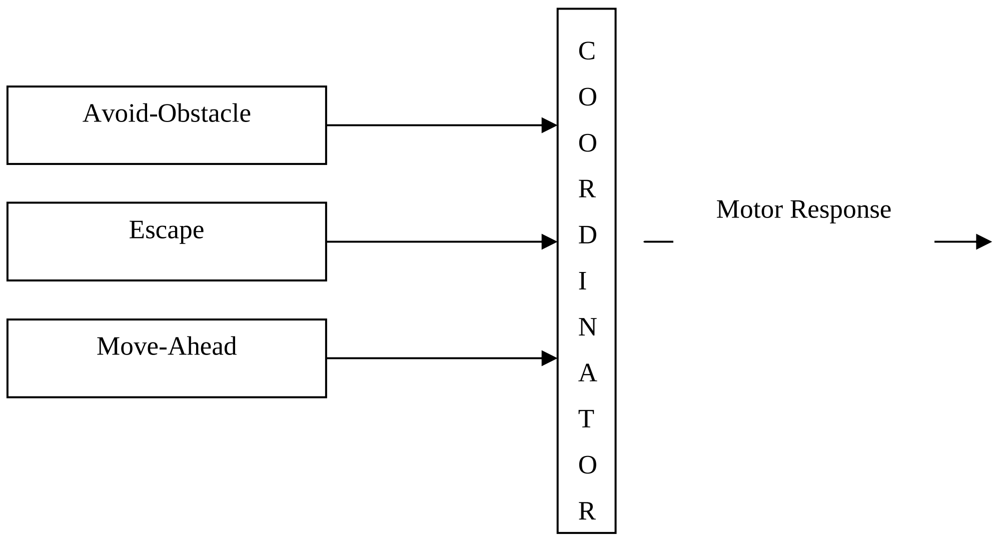


Figure 3-4 Organizational Structure of Move-Around-Safely Behavior

3.4.2 Map-Learning

As described in Chapter 2, a robot's environment is modeled as a graph. Each vertex corresponds to one of the unique landmarks, whereas edges correspond to known straight paths between landmarks. In map-learning behavior, the landmarks, which are the walls in the environment of this problem, are detected; the properties of the vertices are gathered; and navigation of the robot is encouraged to the places, which are not visited yet. Map navigation consists of following the boundaries of walls. When the robot, determines that it is in a particular region, it can move to different locations by traversing the graph representation that connects regions, effectively conducting path planning.

The behaviors covered in this part are:

- **Search-For-Corner:** This behavior uses the processed image of the environment. The perception mechanism used for this behavior is action oriented perception. The motor behaviors provide specifications for a perceptual process: what must be discerned from environmental sensing and constraints as to where it may be located.

  This behavior looks for the walls in the image. The robot is aligned according to the walls that can be perceived and previously visited wall. Search-For-Corner behavior uses the visual information to produce vectors towards the walls that can be seen while following the last wall. Map is used to produce vectors in opposite direction of the walls that were reached from the last wall. Consider the following case: While traversing a wall, it is detected that there exists a LEFT and FRONT wall. So the right motor speed is set to 20mm/s while left motor speed is set to 10mm/s. With this configuration, the robot moves towards the left wall.

- **Avoid-Past:** This behavior enables the robot to go towards the directions that are not searched from the current node. The robot traverses the graph to detect the relative orientation of the walls that are visited from the current wall. Consider the example given in Search-For-Corner behavior. If the robot visited the LEFT wall previously, then a vector in opposite direction is added to the vector summation of the current position. So the left motor speed becomes 10 mm/s while the right motor speed is also assigned to be 10 mm/s. With this configuration the robot moves towards the FRONT wall.

- **Corner-Approach:** This behavior enables the robot to move towards the corner detected as a result of Search-for-Corner behavior. The aim of Corner-Approach behavior is to keep the corner detected after Search-For-Corner behavior in the sight of view of the robot as it moves. The results of search-for-corner and avoid-past are combined to enable the robot to visit non-visited areas. If the wall approached was previously visited then an edge is added to the graph that connects the current node with the node previously visited. This behavior determines the properties (distance between the nodes and the direction of the robot to reach from one node to the other node) of the edges. The length of the edge is estimated by using the distance obtained from wheel encoders and the direction of the robot taken from compass.

- **Determine-Position:** This behavior determines whether the robot’s position is a new place in the environment or the position is a node in the map. This solves the localization problem. When the robot determines the first corner in the environment, the position of the corner is assumed to be the origin of the environment. The estimated position of the next corner is calculated from wheel encoders. Mean while if the action before detecting a corner was wall-following then the direction obtained from compass reading of the wall is assigned to the node representing the wall.

- **Wall-Following:** This behavior enables the robot move through the doorways. The direction of the closest wall is detected and the robot moves towards the wall and when the wall is on the right side or left side of the robot the robot moves ahead. From the definition of the wall following behavior, it is clear that it is a coupled behavior. It results from the Corner-Approach behavior and Move-Ahead behavior. The map of the environment is constructed while the robot is following the walls. Since the nodes of the graph are the walls, metric information about the walls are gathered while performing this behavior.

The structure of the Map-Learning behavior is given in Figure 3-5.

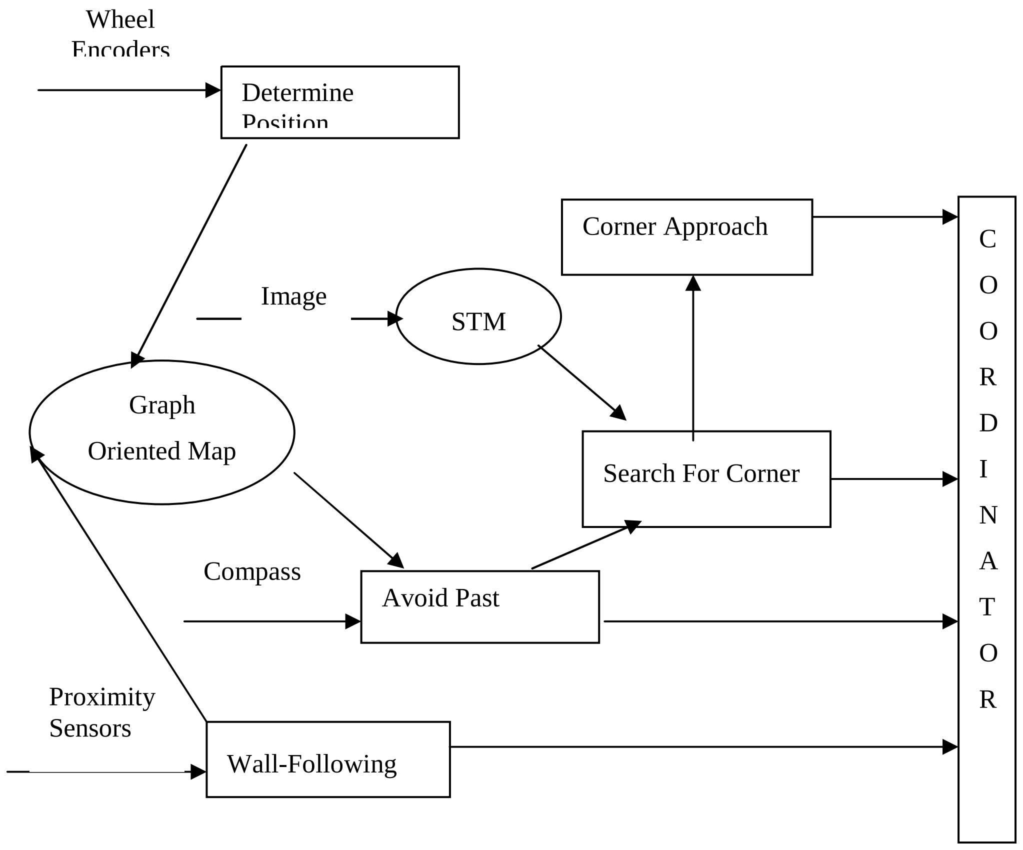


Figure 3-5 Structure of Map-Learning Behavior

As seen from the structure of Map Learning behavior, Determine Position behavior calcutes the estimated position of the robot by using the Wheel Encoders. The estimated position of the robot is searched through the map to determine whether the robot is visiting a node in the map or it is in a new place.

Avoid-Past behaviour uses the map to eliminate the walls already visited.

Search-For-Corner behavior uses the ouputs of Avoid-Past behavior and STM units to determine the direction of the corner to approach.

Corner-Approach behavior uses the outputs of Searc-For-Coner behavior to approach the corner detected.

Wall-Following behavior uses the proximity sensors to allign its position relative to the wall. This behavior feeds the map of the environment.

The coordinator selects the next behavior according to the sensor values received and processed sensor values.

3.4.3 Perceptual Processes

The hidden information in the image which are captured from the camera of the robot is very important to detect possible unvisited landmarks. The aim of the image analysis is:

- To detect the walls that can be reached from an other wall. The image is preprocessed to determine the list and orientation of the walls with respect to the current wall.
- To determine horizontal lines for Corner-Approach behaviors

3.4.3.1 Rectangle Detection

A rectangle is composed of lines. In rectangle detection algorithm, the edges in the image that the robot captures are detected, then the orientation and position of each line in the image is found by issuing Hough Transform. Then the connected lines are extracted to determine the orientation of each wall. The walls are identified to be LEFT_WALL, RIGHT_WALL or FRONT_WALL with respect to the wall currently being followed.

3.4.3.1.1 Line Detection

An edge is the boundary between two regions with relatively distinct Red-Green-Blue (RGB) properties. For each pixel in the image the orientation of the maximum change in RGB values of the pixel with respect to the pixels in its 8 neighbourhood are considered. The 8 neighbourhood of pixel X is given in Figure 3-6.

P0	P1	P2
P7	X	P3
P6	P5	P4

Figure 3-6 8-Neighbourhood of Pixel X

The following values are calculated for each pixel:

$rx = \frac{P3_{red} - P7_{red}}{2}$, $ry = \frac{P5_{red} - P1_{red}}{2}$ for red component of the pixel,

$gx = \frac{P3_{green} - P7_{green}}{2}$, $gy = \frac{P5_{green} - P1_{green}}{2}$ for green component of the pixel,

$bx = \frac{P3_{blue} - P7_{blue}}{2}$, $by = \frac{P5_{blue} - P1_{blue}}{2}$ for blue component of the pixel.

The eigen vectors and eigen values for q1, q2 and q3 are calculated as:

$q1 = rx^2 + gx^2 + bx^2$,

$q2 = rx * ry + gx * gy + bx * by$

$q3 = ry^2 + gy^2 + by^2$

Eigen Vectors give the direction of the changes in RGB values of the pixel with respect to its 8-Neighbours. Eigen values give the amount of changes in the RGB values in the direction of Eigen Vectors. The pixel is on an edge if the maximum eigen value for a direction is greater then a threshold value. Since the only information needed about a pixel for Hough transform, is whether it is on an edge or not, binary information is stored for each pixel insted of storing the color values of the pixel.

When the pixels are identified as whether they are on an edge or not, the lines can be found in the image by the help of Hough Transform. Hough Transform returns the start position, end position and orientation of each line in the image. The start position is the left most pixel and the orientation is the angle between the X-axis and the line that is perpendicular to the line detected. For example the orientation of a horizontal line determined by Hough Transform is 90 degrees.

3.4.3.1.2 Rectangle Detection

The lines determined by the help of Hough Transform is used to identify the rectangles in the image. The following facts determines the framework of rectangle detection algorithm:

- A horizontal line is an evidence for existance of at most 2 rectangles.
- Each line is connected to at most 2 lines to form a rectangle. The orientation of the connected lines must be greater then a threshold value to form a rectangle.

If the distance between start pixels of the parallel horizontal lines is greater than the distance between end pixels, then the rectangle is a LEFT_WALL. If the distance between end pixels of the lines are greater than the distance betwen start pixels, the rectangle is a RIGHT_WALL. If the distances are nearly same then the rectangle is FRONT_WALL. A sample image taken from the robot camera is given in Figure 3-7.

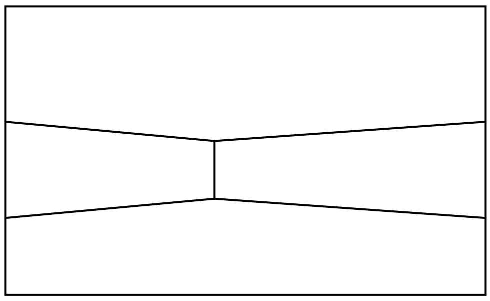


Figure 3-7 Sample Image Taken From the Robot Camera (LEFT_WALL nad RIGHT_WALL)

For each line *Li*, the lines found are traversed to determine the closest 2 lines (*Lj*, *Lk*) which are not paralell to the line. The lines are again traversed to determine the closest 2 lines (*Ljj, Ljk*) to the lines found in the previous step (*Lj or Lk*). If one of the closest line is the original line (*Ljj* == *Li* or *Lkj* == *Li*), then the two lines are said to be connected to form a rectangle. In this way the 4 edges of the rectangle is determined.

3.4.4 Represetational Processes

In this study, Short Term Units (STM) and Long Term Units (LTM) are utilized to store the processed sensor information. LTM is used like memory units of the human that stores learned knowledge, while STM stores the information needed just for a small time.

3.4.4.1 Short Term Memory Units

Short Term Memory (STM), provides recent information to guide the robot that is outside of its sensory range. STM are used in support of a single behavior, the representation directly feeds the behavior just like other sensory data, and the representations are constructed and used while the robot is in environment and then discarded. STM is utilized to store the detected walls while following a wall. The STM feeds the Search-For-Corner behavior. When the robot starts to follow another wall, the STM units created for the previous wall are eliminated.

Each unit of the STM stores the walls that can be seen while following another wall. When the robot finishes following the wall, STM units are used to determine the next wall to follow. According to the physical limitations of STM, the detection algorithm uses the last n elements of the STM.

3.4.4.2 Long Term Memory Units

The long-term memory (LTM) stores the landmarks, which are walls in this problem. The long-term memory is a topological map whose nodes represents the landmark of interest and each node is connected to others by edges augmented with the compass information.

Each node is represented by the estimated position of the start location, end location and the orientation of the wall. The location information is estimated by using the odometry information taken from the wheels of the robot. The orientation is taken from compass readings.

Wall-Following behavior initializes the new node and as the robot follows the wall the metric information is gathered and averaged to obtain more accurate estimates of the properties of the landmarks. While averaging the metric values of the landmarks, median is used instead of mean, because the deviation of the first metric values is very high.

Corner-Approach behavior initializes the edge that connects the last node followed and the new wall. The metric information of the edge is gathered as Corner-Approach behavior is being executed.

Each node of the LTM has an influence on the exploration task and localization problem. As the robot moves in the environment it searches the map to understand its estimated location with respect to the nodes in the map. The topological features of the current location of the robot can be identical to one of the nodes in the map. If it is the case, the connected nodes and the walls that can be seen from the current wall are taken into consideration and an orientation is estimated for the robot to move towards unvisited areas

If the topological features of the current location is similar to one of the nodes in the map, then the topological features of the node in the map is averaged according to the metric values of the new node.

If the current location of the robot is different than all the nodes in the map, then it is added to the map as a new node.

3.5 Flow of Execution

Flow of execution diagram displays the run time activities and control points that the application performs. The rectangles represent the activities, while the diamonds represent the control points. The conditions that are checked in each control point are written near the control point. Flow of Execution diagram is inspired from the Activity Diagram of UML modeling language [26]. In this section the main flow of the application and the details of some behavior’s flow of execution are given.

3.5.1 Main Flow of Execution

The application starts with preprocessing the image that the camera receives. Then the application determines the active behavior according to the sensor values. If the proximity sensor indicates that the robot is trapped then Escape behavior is executed. If there is no corner (horizontal line) in the image, then Search-For-Corner behavior is executed. If there is a corner in sight but the robot is not near to a wall to follow then Corner-Approach behavior is executed. If the robot is near a wall then Wall-Following behavior is executed.

If the behavior executed before Search-For-Corner behavior is Wall-Following, this implies that the robot was following a wall. If this is the case, the node that represents the wall is finalized and new node is searched in the map. If it is not in the map, it is added to the map.

If the behavior before Wall-Following behavior is Corner-Approach, this means that the robot was looking for a wall so a new node for the current wall will be created. If the behavior before Wall-Following is not Corner-Approach then the metric values for the wall that is being followed is corrected. While the metric values are collected to obtain more accurate information for each node, the walls that can be reached from the current wall is extracted from the image by using the Hough Transform algorithm.

The main flow of execution is depicted in Figure 3-8.

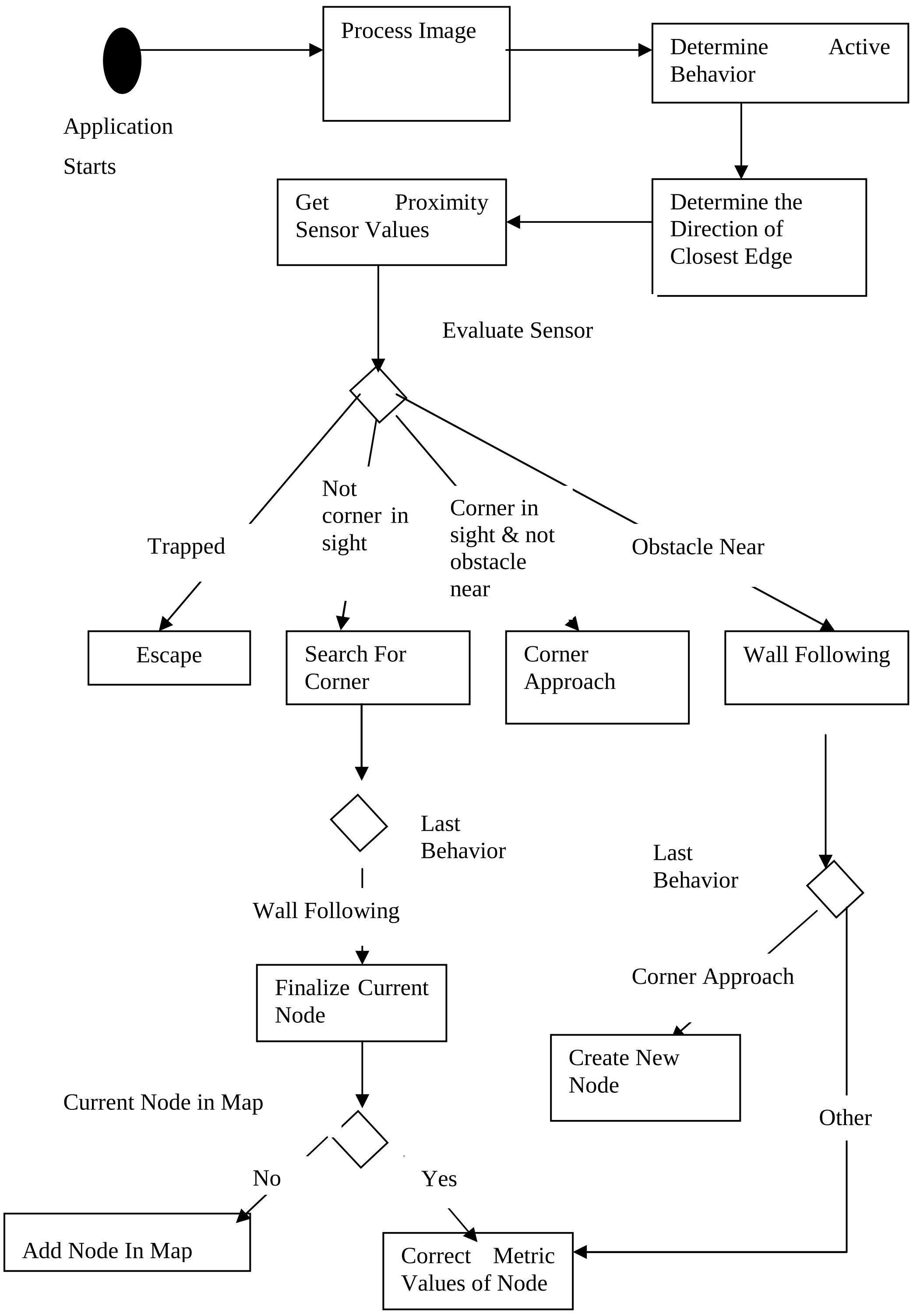


Figure 3-8 Main Flow of execution

3.5.2 Flow of Execution of Corner-Approach Behavior

Corner-Approach behavior stars with determining the speeds of the right motor and left motor according to the position of the horizontal line that the Search-For-Corner behavior orders to approach. The flow of execution is depicted in Figure 3-9.

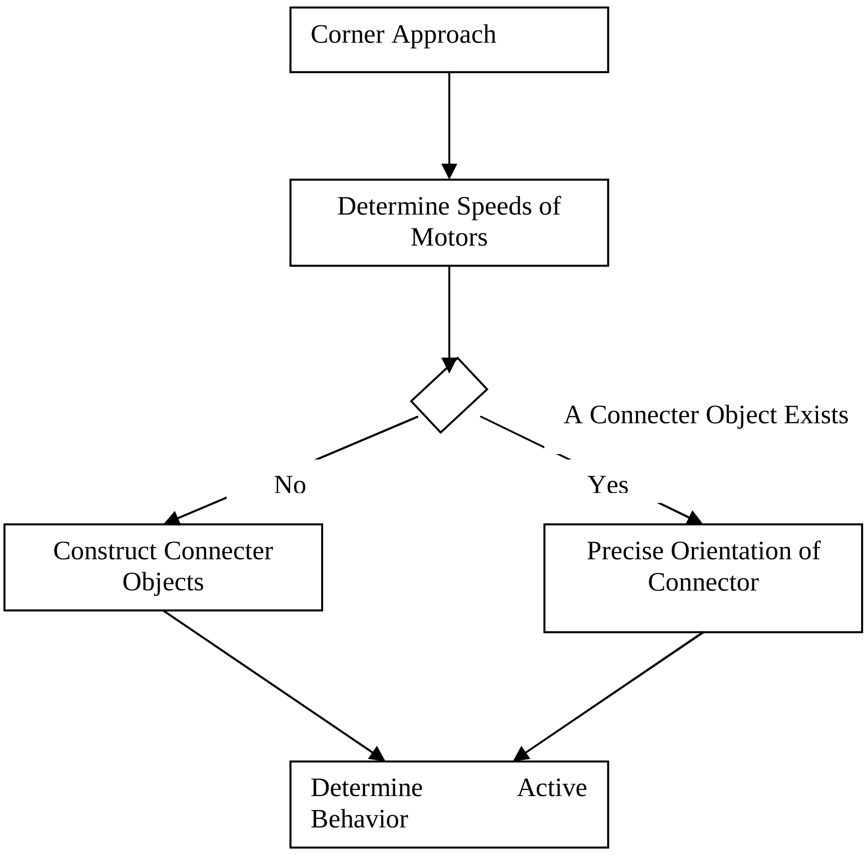

Corner Approach
Determine Speeds of Motors
A Connecter Object Exists
No
Yes
Construct Connecter Objects
Precise Orientation of Connector
Determine Active Behavior

Figure 3-9 Flow Of Execution For Corner Approach Behavior

One of the aims of Corner-Approach behavior is to determine the edges of the graph that represents the map of the environment in which the robot acts. When Corner-Approach behavior is initiated a new edge is created. As the robot performs Corner-Approach behavior the metrics of the edge becomes more precise. When the robot executes Wall-Following behavior after Corner-Approach behavior, the edge that is created during Corner-Approach behavior is used to connect the last wall and the recently found wall.

3.5.3 Flow of Execution of Search-For-Corner Behavior

Search-For-Corner behavior determines a direction for Corner-Approach behavior. The result of this behavior depends of the following items:

- The last wall followed
- The walls that can be reached from the last wall
- The walls previously visited from the last wall

While determining the direction for Corner-Approach behavior, this behavior considers the walls that can be reached from the last wall. It eliminates the walls that were previously visited after the last wall by using the map. When it determines the direction, it sets the speeds of the motor for selected direction. This behavior is performed until a wall that is not visited after the last wall is in sight of the robot. The flow of execution is depicted in Figure 3-10.

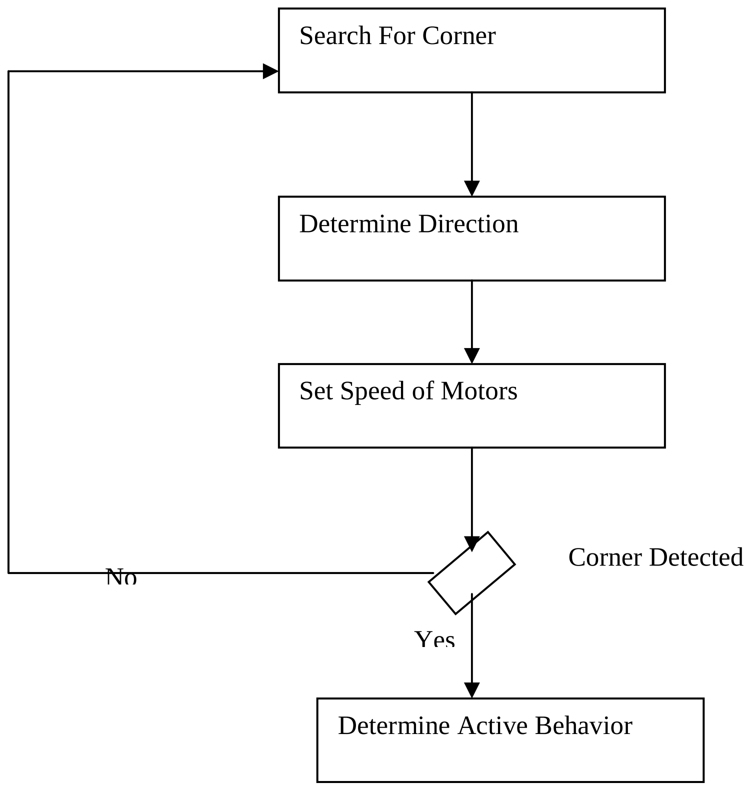


Figure 3-10 Flow of Execution of Search-For-Corner Behavior

3.5.4 Flow of Execution of Wall-Following Behavior

From the main flow of execution,Wall-Following behavior is initiated when a wall is detected from the proximity sensors. In Wall-Following behavior the side of the nearest wall is determined. A control point to determine whether there exists a front wall or not is executed. If there exist a front wall then there is no need to execute the activity to determine active behavior, because the robot is supposed to follow the wall in the front.

If there exists a front wall, then the robot turns until the front door is on the left or right side of the robot. When the robot turns because of the fact that there exists a front wall, it starts to follow a new wall. The front wall is searched from the map and if it is an unvisited wall, then last node is finalized and a new node is created for the front wall. If the front wall was reached from the last wall, then Search-For-Corner behavior is executed.

If there is not a front wall, then the robot moves ahead. The robot determines the walls that can be reached from the current wall while folowıing the wall.

3.6 Experimental Results

The robot control system is tested in environments prepeared in Webots 2.0 simulation tool. The robot passes through all paths that connects the walls. The paths going from a wall connects, the walls that can be seen while following the wall. When all the paths are identified, the map constructed by the robot was accurate and the robot located itself in its internal world successfully during run time.

The worlds in which the robot control system is tested is given in Figure 3-12, Figure 3-13, Figure 3-14, Figure 3-15, Figure 3-16 and Figure 3-17.

Each experiment is conducted 10 times since every time it starts, it choses a random orientation.

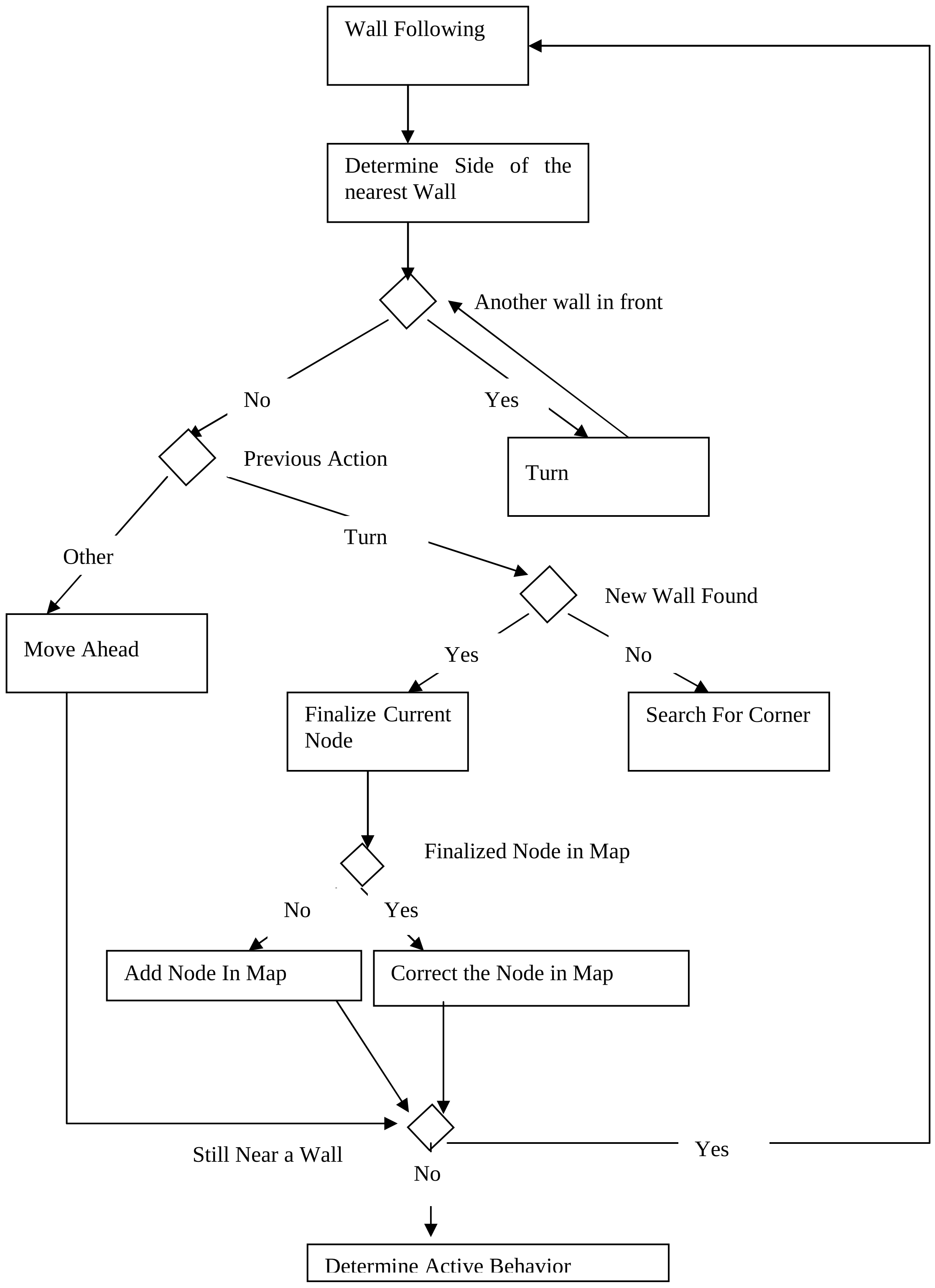


Figure 3-11 Flow of Execution of Wall-Following Behavior

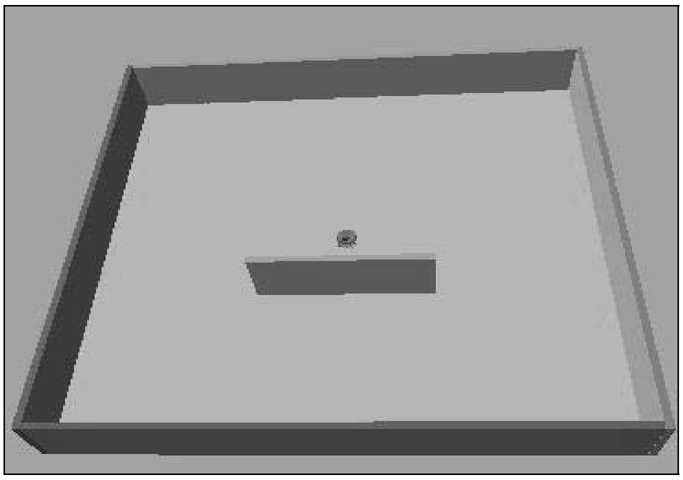


Figure 3-12 World of Experiment 1

The robot moves towars the small wall first, then according to its orientation it follows the rigth border wall or left border wall. When all the walls are visited the robot starts to move randomly in the environment.

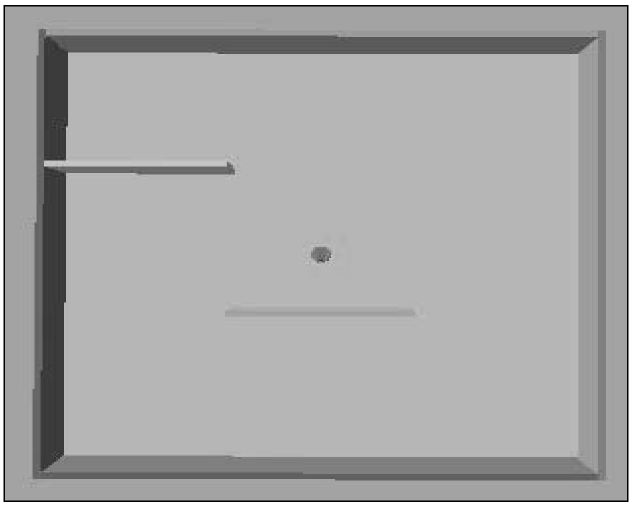


Figure 3-13 World of Experiment 2

Experiment 2 aims to test the performance of the robot with more then one Wall.

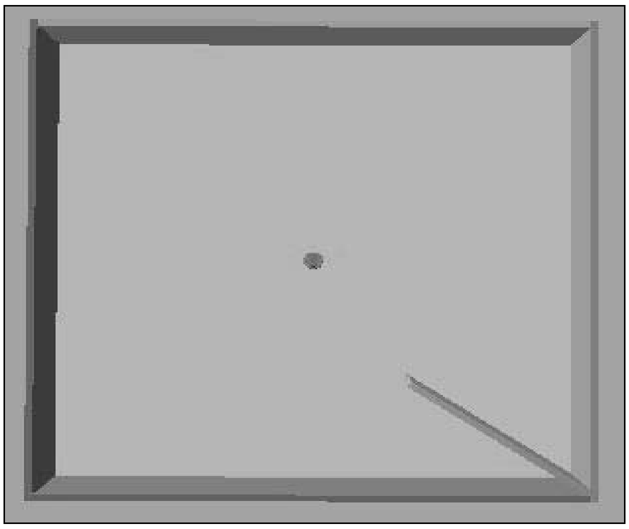


Figure 3-14 World of Experiment 3

Experiment 3 is conducted to test the performance of the robot in environments with walls in random orientations.

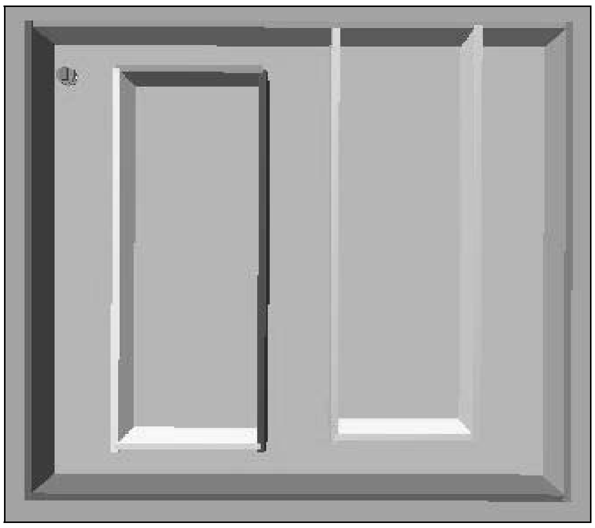


Figure 3-15 World of Experiment 4

The control system is tested in the maze like environments like the one in Figure 3-15. Since the walls are landmarks for the designed robot control system, the robot follows the walls that constitutes the hallways. So it passes through the same hallway twice to traverse the each wall.

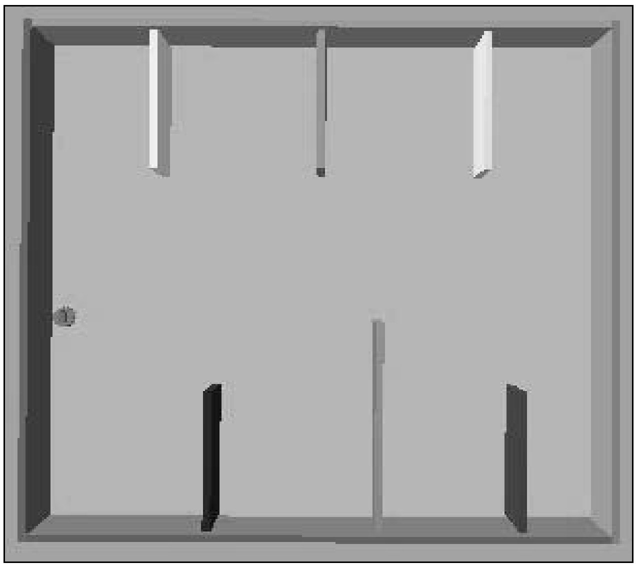


Figure 3-16 World of Experiment 5

Experiment 5 is conducted to test the robot with crowded environments.

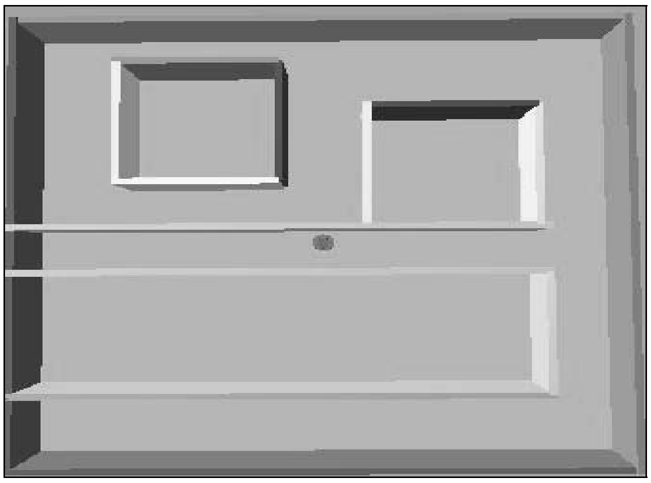


Figure 3-17 World of Experiment 6

Experiment 6 is conducted to test the robot in complex maze like environements.

Since this robot control system works in environments with wall, this control system is suitable for office like environments. Behaviors to handle objects in the environment other then walls can be added for robot to operate in more complex environments.

The performance of the robot gets slower as the number of the walls in the environment increments since the algorithmic complexity of rectangle detection and graph traversing is proportional to the number of walls. Memory usage depends on the number of distict walls and the possible ways between the walls. Map operations determines the memory usage since they are stored in LTM. The rectangles extracted from the image is stored in STM which has a fixed size and don't effect the memory usage. The performace of the robot Control system from memory and processor usage perspective is given in Figure 3-18.

Number of Walls in the Environment	**Memory Usage**	**Processor Usage**
4	239,196 KB	%18,24
5	289,764 KB	%17,16
6	296,620 KB	%19,01
8	315,104 KB	%23,10
9	317,452 KB	%23,86
12	324,104 KB	%26,02
17	335,214 KB	%31,57

Figure 3-18 Performace of Robot Control System

Emergent behaviours comes out in behavior besed system architectures as a result of coordination between simple behaviors. This fact is an advantage and sometimes a disadvantage. It is an advantage, because complex behaviors emerges although, the designer designs only simple behaviors. It is disadvantage because, unexpected behaviours can emerge and the designer shall design other behaviors to eliminate the unexpected behaviors.

One of the planned test case in the scope of this study was loading the robot control system to the Khepera and test the robot in real wold. However because of the lack of sensors of the robot currently in hand, the planned experiments can not be conducted. The sensors that are not supported are compass, and camera.

3.7 Contribution of This Thesis

There are many studies about map learing and navigation. For a mobile robot, the ability to navigate is one of the most important of all. Staying operational, for instance avoiding dangerous situations such as collisions, come first. But if any tasks, which relate to specific places in the robot's environment are to be performed navigation is a must. Navigation can be defined as the combination of three fundamental components:

- Map-building. This is the process involved in constructing a map from sensor readings taken at different robot locations. The correct treatment of sensory information and the reliable localization of the robot are fundamental points in the map-building process.

- Localization. This is the process involved in obtaining or knowing the actual robot's pose or location from sensor readings and the current map. An accurate map and reliable sensors are key points in achieving good localizations.

- Path-planning. This is the process involved in generating a feasible and safe trajectory based on the current map from the current robot location to a goal position. In this case it is very important to have an accurate map and a reliable localization.

One of the objectives of this thesis is to work on unknown environments, and the robot is provided with the three given competences. Often one or more of those components are not present in a given robot control system. Instead priori knowledge is provided.

There exist different methods for map representations as described in Chapter 2. These are grid based, topological maps and a combination of both to eliminate the disadvantages of both method and to combine the advantages of both methods. In this study robot's environment is modeled as a graph, whose vertices correspond to the landmarks while the edges correspond to known straight lines between landmarks. The edges are labeled with the distance and orientation of the line between landmarks.

The localization problem is solved since global estimated values of the landmarks are stored in the map and the estimated position of the robot is calculated from wheel encoders. The robot traverses the graph to determine its location in its internal representation of the world.

CHAPTER 4

CONCLUSION

This study presents a behavior based robot control system that enables the robot to construct map of the environment while moving safely and purposefully. This work is a demostration of the fact that complex behaviors can emerge from simple primitive behaviors. The reactive structure of the behavior based control systems emerges this propety.

Complex tasks, in this case map construction, can be achieved by utilizing representational knowledge.The map of the environment is utilized to remember the past actions of the robot with repect to the current location.

The study is tested on a simulation environment. A robot has physical presence. This spatial reality has consequences in its dynamic interactions with the world that can not be simulated faithfully. Future work will be to test the control system designed on a real robot.

In map construction, each node in the map has position information. This is an estimated position of the node with respect to the first corner detected.

For different behaviors, different types of sensor information is used. The behaviours can be updated to rely on image data only. The robot can be trained to learn behaviors that havily rely on proximity sensors. The map construction functionality can be updated since it uses positional information.

The robot control systems is designed with the assumption that the only objects in the environment are walls. To test the robot in office like environments other behaviors to handle objects other then walls can be added. When these kinds of behaviors are added, the map functionality shall be updated to determine the landmarks in the environment. Each object in office like environments can not be a landmark.

REFERENCES

[1] R.C. Arkin. Behvior Based Robotics. MIT Press, 1998.

[2] W.G.Walter. The Living Brain Norton. New York. 1953.

[3] V.Braitenberg. Vehicles: Experiments in Synthetic Psychology, MIT Press, Cambridge, MA. 1984.

[4] J. Albus, H.McCain and R. Lumia. NASA/NBS Standard Reference Model for Telerobot Control System Architecture(NASREM) NBS Technical Note 1235, Robot System Division, National Bureau of Standards.

[5] R. Brooks New Approaches to Robotics Science Vol 253 pp 1227-1232 September 1991.

[6] M. Arbib and D.House Depth and Detours : An Essay on Visually Guided Behavior , in Hand Function and the NeoCortex, eds. A. Goodman and I/Darian-Smith , Springer-Verlag, New York, pp 135-170 1987

[7] P.E. Agre, D. Chapman. Pengi: An Implementation of a Theory of Activity Proceedings of the American Association of Artificial Intelligence Conference pp 268-271, 1987

[8] R. Brooks A Robot That Walks: Emergent Behaviors from Carefully Evolved Network Proceedings of the IEEE International Conference on Robotics and Automation. Pp 692-694 May 1989.

[9] R.C.Arkin and D. MacKenzie Temporal Coordination of Percetual Algorithms for Mobile Robot Navigation. IEEE Transactions on Robotic and Automation, Vol. 10, No.3 June 1994

[10] A. Bellaiche, F. Jean, J.J. Risler Robot Motion Planning and Control Guidelines in Nonholonomic Motion Planning for Mobile Robots 1998.

[11] N. Nilsson Teleo-Reactive Programs for Agent Control Journal of Artificial Intelligence Research, Vol 1 pp 139-158 1994

[12] O. Khatip. Real-Time Obstacle Avoidence for Manipulators and Mobile Robots Proceedings of the International Conference on Robotic and Automation St Louise MO. Pp 500-505 1985

[13]R. Brooks. Intelligence Without Reason A.I Memo No 1293 MIT AI Laboratory. April 1991.

[14] R. Brooks. A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation. Vol. RA-2 No. 1. pp 14-23, 1986

[15] J.Rosenblatt. DAMN: A Distributed Architecture for Mobile Navigation Working Notes, AAAI 1995 Spring Symposium on Lessons Learned for Implemented Software Architecture for Physical Agents. Palo Alto, CA, March 1995.

[16] R.Brooks. A hardware Retargetable Distributed Layered Architecture for Mobile Robot Control. Proceedings of the IEEE International Conference on Robotics and Automation. Raleigh, NC, 1987. pp. 106-110

[17] M. Mataric. Behavior Based Control: Main Properties and Implications. Proceedings of Workshop on Intelligent Control Systems, International Conference on Robotic and Automation. Nice, France 1992.

[18] R.C. Arkin. Neuroscience in Motion: The Application of Schema Theory to Mobile Robotic. In Visumotor Coordination: Amphibians, Comparison, Models, and Robot. New York : Plenum Press, 1989. pp. 649-672

[19] R.C Arkin. The Impact of Cybernetics on the Design of a Mobile Robot System. IEEE Transactions on Systems, Man, and Cybernetics, Vol.20 No.6, November/December 1990. pp. 1245-1257

[20] R.C. Arkin. Modelling Neural Function at the Schema Level: Implications and Results for Robotic Control. Biological Neural Networks in Invertabrate Neurothology and Robotic 1993. San Diago, CA. pp. 383-410.

[21] S.Hutchinson, G.D.Hager & P.Corke. A Tutorial on Visual Servo Control. IEEE Transactions on Robotics and Automation, Vol. 12, No.5 October 1996.

[22] H.Zhang. Visual Motion Planning for Mobile Robots IEEE Transactions on Robotic and Automation, Vol. 18, No.2 April 2002

[23] Rafael C. Gonzales, Richard E. Woods. Digital Image Processing. Addison-Wesley Publishing Company, 1993.

[24] G. Dudek & M. Jenkin. Computational Principles of Mobile Robotics, Cambridge University Press, Cambridge, UK. 2000

[25] Cyberbotics S.a.r.l Webots 2.0 User Guide, 1999.

[26] Pierre-Alain Muller. Instant UML Wrox Pres Ltd. Canada. 1997