İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

BEHAVIOR GENERATION BY ARTIFICIAL EMOTIONS AND COGNITION FOR MULTI-GOAL ROBOT TASKS

M.Sc.Thesis by

Evren DAĞLARLI, B.Sc.

Department : Mechatronics Engineering

Programme: Mechatronics Engineering

Supervisor : Prof. Dr. Hakan TEMELTAŞ

DECEMBER 2006

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

BEHAVIOR GENERATION BY ARTIFICIAL EMOTIONS AND COGNITION FOR MULTI-GOAL ROBOT TASKS

M.Sc.Thesis by

Evren DAĞLARLI, B.Sc.

(518051011)

Date of submission : 14 December 2006

Date of defence examination: 30 January 2007

Supervisor (Chairman): Prof. Dr. Hakan TEMELTAŞ

Members of the Examining Committee Prof. Dr. İbrahim EKSİN

Assoc. Prof. Dr. Ata MUĞAN

DECEMBER 2006

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU AMAÇLARA SAHİP ROBOT GÖREVLERİ İÇİN KAVRAMA VE YAPAY DUYGULARLA DAVRANIŞ OLUŞTURULMASI

YÜKSEK LİSANS TEZİ
Evren DAĞLARLI
(518051011)

Tezin Enstitüye Verildiği Tarih : 14 Aralık 2006
Tezin Savunulduğu Tarih : 30 Ocak 2007

Tez Danışmanı : Prof.Dr. Hakan TEMELTAŞ
Diğer Jüri Üyeleri Prof.Dr. İbrahim EKSİN
Doç.Dr. Ata MUĞAN

ARALIK 2006

Preface

This thesis tries to combine of the recent developments in cybernetics and cognitive sciences with the viewpoint of control engineering and it is aimed to guide to autonoumus systems and robotics researchers about intelligent robot control architectures. As a student of the Mechatronics Engineering Masters Program in İTÜ, I would like to thank to all the instructors and coordinators of the program for giving an opportunity, their courage and enthusiasm.

Also thanks to my thesis advisor Hakan Temeltaş for his worthy opinion support and guidance in my researches, his patience and trust in my long term studies, gaining vision for my future scientific works and being open minded to new ideas through this thesis.

December 2006 Evren DAĞLARLI

Index

List of Tables

List of Figures

BEHAVIOR GENERATION BY ARTIFICIAL EMOTIONS AND COGNITION FOR MULTI-GOAL ROBOT TASKS

SUMMARY

In this study, the new approaches to the robotics subject, according to emotion and cognitive based robot control approach, behavior generation and self-learning paradigms are investigated for the real-time applications of multi-goal mobile robot tasks, Artificial Emotion and Cognitive Mechanism Based Robot Control Architecture is built up for a four-wheel driven and four-wheel steered mobile robot. Discrete stochastic state-space mathematical model is considered for behavioral and emotional transition processes of the autonoumus mobile robot in the dynamic realistic environment. Behavioral system is determined as evolutionary neural-fuzzy inference system for behavior generation and self-learning processes in the general robot control architecture. Motivational effect of synthetic emotional system act as behavioral gain coefficiens on the behavior sequences. The term of Cognitive mechanism system which is composed from rule base and reinforcement self-learning algorithm explain all of the deliberative events such as learning, reasoning and memory (rule spaces) of the autonoumus mobile robot to us. The kinematic and dynamic model of the mobile robot with non-holonomic constraints is used as present structure which is modeled in previous studies. The posture and speed of the robot and the configurations, speeds and torques of the wheels and all deliberative and cognitive events can be observed from the simulation plant and virtual reality viewer. The behaviors are investigated regarding their gains, fuzzy inference structures, real-time applicability and their coordination. This study constitutes basis for the multi-goal robot tasks and artificial emotions and cognitive mechanism based behavior generation experiments on a real mobile robot.

ÇOKLU AMAÇLARA SAHIP ROBOT GÖREVLERİ İÇİN KAVRAMA VE YAPAY DUYGULARLA DAVRANIŞ OLUŞTURULMASI

ÖZET

Bu çalışmada robotik alanında yeni yaklaşımlar olan duygu ve kavrama temelli robot kontrol yaklaşımına göre davranış üretimi ve kendiliğinden öğrenme konuları gerçek zamanda mobil robot uygulamaları bakımından incelenmiş, dört çekerli, dört yönlendirmeli bir mobil robot için yapay duygu ve kavrama mekanizması temelli robot kontrol mimarisi oluşturulmuştur. Dinamik gerçekçi ortamdaki otonom mobil robotun davranışsal ve duygusal geçiş işlemleri için ayrık rastlantısal durum uzayı matematiksel modeli düşünülmüştür. Davranış üretimi ve kendiliğinden öğrenme süreçleri için davranışsal sistem genel robot kontrol mimarisinde evrimsel sinirsel bulanık çıkartım sistemi olarak tanımlanır. Yapay duygusal sistemin motivasyonel etkisi davranış zincirinde davranışsal kazanç katsayısı olarak etkir. Kural tabanı ve zorlayıcı kendiliğinden öğrenme algoritmasından oluşan kavrama mekanizması terimi bize otonom mobil robotun öğrenme, muhakeme ve hafıza (kural uzayı) gibi tüm düşünsel olaylarını açıklar. Holonomik olmayan kısıtlara sahip mobil robotun kinematik ve dinamik modeli önceki çalışmalarda modellenen mevcut yapıyı kullanır. Simülasyon ortamından ve sanal gerçeklik gözlem penceresinden mobil robotun pozisyonu, tekerlek ve robot yönelimleri, tekerlek ve robot hızları, tekerlek momentleri ve tüm düşünsel, kavrama faaliyetleri gibi parametreler izlenebilmektedir. Davranışlar da, simulasyon ortamında kazanımları, bulanık mantık işleme yapıları, gerçek zaman uygulanabilirliği ve davranışların koordine edilmeleri bakımından incelenmiştir. Bu çalışma gerçek bir robotta yapılacak çoklu amaçlara sahip robot için yapay duygu ve kavrama mekanizması temelinde davranış oluşturma deneyleri için temel teşkil etmektedir.

1 Introduction

The multi-goal robot tasks menagement is one of the biggest challenges for robot researchers in the robot controller design. When the mobile robots are working in a realistic and dynamic environment, they have to overcome more behavioral tasks than singular task. Moreover most of this behaviors which include multiple objective complex tasks try to reach satisfaction situation.

The behaviors of the mobile robots have several features that pose serious difficulties to the learning and adaptation abilities. They have multiple objective which may confilict with each other. According to the temporary needs and goals of the robot, behaviors are dynamically changed in a realistic environment. The generated behaviors have short-term and long-term goals [1]. A sequence of different behaviors can trigger a related emotional state to accomplish a certain goals.

Having seen that the robotic and AI communities are progressing slowly in this area relative to the computer technology, a new approach has emerged in the mid 1980's. This approach redefines the intelligence [2] and tries to mimic the animal behaviors in basic, modular levels on the reactive robotics foundations [3, 4]. The name for this approach is Behavior Based Robotics and it is defined in chapter 3, starting from the biological inspirations [5-7], continuing on the first studies and theoretical foundations of the approach [2-4] and extended with Emotion Based Architecture [8-10]. The recent studies are on the hybrid architectures having interaction between the upper deliberative levels and lower level behaviors. The behaviors are used in the lower levels as reactive modules and some deliberative actions or long-term planning are done on the upper levels that are managing the behaviors [11]. Especially EMIB computational robot control architecture presented as a good example to this in 2002. This architecture was made of three main level. Following study about emotion based robot control architecture which is presented in 2003 is ALEC. Hidden Markov model based stochastic model was used for behavior selection in this controller. Also another work related with emotion based structure was appeared by Martin Buss in 2004. According to development of

cognitive sciences and control engineering, studies related with this subject will become in future.

Presented robot control architecture provides better solutions for achieving multi-goal tasks. Artificial emotions and cognition based architecture constitutes main framework of the robot control strategy. Also cognitive architecture term explains all of the deliberative events such as learning, reasoning and memory (rule spaces) of the mobile robot to us. It was constructed three main levels such as behavioral system, coordination level and emotion-motivation level for this proposed robot control architecture.

Behavioral system establishes relationship trough from sensors to actuators. Behavioral system can be considered as low-level control component of general architecture. This level generates different behaviors as relational fuzzy logic inference process. Generated behaviors are encoded as diploid chromosomal genetic string in self-organizing map neural network. Different type unique behaviors can be improved instead of certain predominated behaviors for investigation of the self learning paradigm in the intelligent autonoumus robot control architecture.

The coordination level provide interactive connection between behavioral system and emotion-motivation level. This level define several sub-modules such as instictual module, priority filter, behavior selection module and cognitive module. Instinctual module determine innate behaviors of autonoumus mobile robot. Also new behaviors can be added into this module when detected new behavioral states are learned in behavioral system. Priority filter is pre-processor for initial probability distribution. Thus created behaviors are prioritized such as Maslow hierarchy of needs theory. Behavior selection module employes a discrete stochastic state-space model which is called as hidden Markov model. Because the correct timing of the behavior-switching can be vital for multi objective task menagement. Cognitive module which is learning strategy of this layer implements a hybrid reinforcement self-learning algorithm. This module use Q-learning for training of state transition probability in SOM neural network.

The emotion-motivation level is upper control unit of artificial emotions and cognition based autonoumus robot control architecture. This level which is supervisor of autonoumus robot control architecture is divided to two sub-module such as emotion module and motivational module. Emotion module transition process is based on observation part of state-space hidden Markov model. When emotions are triggered, behavioral sequences are determined. Long-term behavioral action memory is

planned in emotions. For this module, it can be considered a fuzzy rule base and emotional learning module as second order cognitive system. Motivational module define intense of behaviors. Also intense of execution of the behaviors can directly act performance of reaching the goals. Motivation module apply a behavioral gain coefficient to the generated behaviors.

In the next section, artificial background information which is related with this thesis was given. In this part, hidden Markov model, reinforcement Q-learning, multi-chromosomal genetic programming and self-organizing neural networks were presented for integration into the thesis. In chapter 3, behavior and artificial emotions based robotics literature survey was presented by chronological order. Also several main architectures such as EMIB and ALEC were explained in chapter 3. In chapter 4, our proposed artificial emotions and cognition based general architecture was referred. This part contains three main level as general. These are behavioral system, coordination level and emotion-motivation level. Experimental results, simulink blocks and system components were explained in detail in chapter 5. Also 3D Virtual reality toolbox simulation environment and basic background related with simulation tools such as matlab simulink state flow toolbox, virtual reality toolbox were referred in this part. In chapter 6, conclusion and evalution of the experimental results were given. Also future works related with this thesis were determined in this section. In appendıx part, modeling of the 4x4x4 type mobile robot, kinematics and dynamic parameters are supervised under the non-holonomic constraints of the mobile robot, and the mathematical background is explained.

2 Artificial Intelligence Background : Learning, Classification And Reasoning

2.1 Hidden Markov Models

2.1.1 Description of Model

Hidden Markov models (HMMs) are latent variable models based on the Markov chains for sequential, stochastic data [12]. The sequences of data patterns are compactly represented as a state network. This approach can be resembled to finite state machine (FSM) representation. The main idea behind HMMs is to integrate a simple and temporal data models and the available statistical modeling tools for stationary signals into a sound mathematically tractable framework [13]. Majority of applications have been in pattern recognition and signal processing but succesive results are also reported in other fields, such as robotics, machine learning applications, text analysis, coding theory (criptology), ecology, and molecular biology. Also HMMs are widely used especially in speech recognition and there is an extensive literature on them.

The HMM and linear state-space model (SSM) are actually rather closely related. They can both be interpreted as linear Gaussian models [14]. The greatest difference between the models is that the SSM has continuous hidden states whereas the HMM has only discrete states. The nonlinear switching state-space model is a combination of two well known models, the hidden Markov model (HMM) and the nonlinear state-space model (NSSM) [12].

2.1.2 Markov Chains

A Markov chain is a discrete stochastic process with discrete states and discrete transformations between them. At each time instant the system is in one of the N possible states, numbered from one to N. At regularly spaced discrete times, the system switches its state, possibly back to the same state. The initial state of the chain is denoted M1 and the states after each time of change are M2,M3, . . [12]. Standard first order Markov chain has the additional property that the probabilities of the future states depend only on the current state and not the ones before it [15].

Formally this means that

$$p(M_{t+1} = j \mid M_t = i, M_{t-1} = k, ...) = p(M_{t+1} = j \mid M_t = i) \quad \textbf{(2.1)}$$

This is called the Markov property of the chain. Because of the Markov property, the complete probability distribution of the states of a Markov chain is defined by the initial distribution $\pi_i = p(M_t = i)$ and the state transition probability matrix

$$a_{ij} = p(M_{t+1} = j \mid M_t = i), \quad 1 \le i, j \le N. \quad \textbf{(2.2)}$$

Let us denote $\pi = (\pi_i)$ and A=(a_{ij}). In the general case the transition probabilities could be time dependent, i.e. aij = a_{ij}(t), but in this thesis only the time independent case is considered [12]. This allows the evaluation of the probability of a sequence of states M = (M1,M2, . . . ,MT), given the model parameters $\theta = (\mathrm{A}, \pi)$, as

$$p(M \mid \theta) = \pi_{M_t} \left[\prod_{t=1}^{T-1} a_{M_t, M_{t+1}} \right]. \quad \textbf{(2.3)}$$

2.1.3 Hidden States

Hidden Markov model is basically a Markov chain whose internal state can not be observed directly but only through some probabilistic function. That is, the internal state of the model only determines the probability distribution of the observed variables [12]. Let us denote the observations by X = (x(1), . . . , x(T)). For each state, the distribution p(x(t)|M_t) is defined and independent of the time index t. The exact form of this conditional distribution depends on the application. In the simplest case there is only a finite number of different observation symbols [12]. In this case the distribution can be characterised by the point probabilities;

$$b_i(m) = p(x(t) = m \mid M_t = i), \quad \forall i, m \quad \textbf{(2.4)}$$

Letting B = (bi(m)) and the parameters $\theta = (\mathrm{A}, \mathrm{B}, \pi)$ the joint probability of an observation sequence and a state sequence can be evaluated by simple extension to Equation (2.3)

$$p(X, M \mid \theta) = \pi_{M_t} \left[\prod_{t=1}^{T-1} a_{M_t M_{t+1}} \right] \left[\prod_{t=1}^{T} b_{M_t}(x(t)) \right]. \quad \textbf{(2.5)}$$

The posterior probability of a state sequence can be derived from this as

$$p(M \mid X, \theta) = \frac{1}{p(X \mid \theta)} \pi_{M_t} \left[\prod_{t=1}^{T-1} a_{M_t M_{t+1}} \right] \left[\prod_{t=1}^{T} b_{M_t}(x(t)) \right]. \quad \textbf{(2.6)}$$

Where

$$p(X \mid \theta) = \sum_M P(X, M \mid \theta). \quad \textbf{(2.7)}$$

2.1.4 Assumptions and Some Definitons

The basic assumption in the Markov models is that the system has a finite number of possible states, but it can occupy more than only one of them at a time. In each state characteristic signals are emitted and the signal features are observed independently for the time intervals t =1, …. ,T. For a sequence of observed feature vectors of the system. O=(O1,O2, ..., OT) There exists then the corresponding hidding sequence of states q=(q1,q2, ..., qT) that the system has visited starting from the initial state q0. Another basic assumption is Markov properity, i.e. the transition probability of the system to next state depends only on the provious state regardless of the earlier transition history [13]. The HMM consist of the state transition probability matrix A=[a_{ij}], where

$$a_{ij} = P(q_t = j \mid q_{t-1} = i), \quad \text{i,j =1, . . . , N} \quad \textbf{(2.8)}$$

And the set of state probability density models $B = \{b_i(O_t)\}_{i=1}^{N}$. The density model can be either a set of discrete probabilities for quantized observations $b_i(O_t) = P(O_t \mid q_t = i)$, or a continious probability density function [13].

The states and their interactions are the core of the HMM. Combining of the definitions above and the starting probabilities of the states $\pi_i = P(q_0 = i)$. Formally the HMM is defined as a triple $\lambda = (\pi, A, B)$ where A = [a_{ij}] is an N x N matrix of the state transition probabilities, B is a set of observation probability densities of N states, and π is an initial state probability vector. Different HMMs can be conveniently connected by including the transitions between models to the matrix A. There exist well-known optimization algorithms for the final parameter estimation task relying on different criteria such as maximum-likelihood (ML) principle, mutual information, or error-correctiveness [13].

However, a fundamental question before any parameter estimation algorithm can be applied, is how to define and parametrize an HMM state or a state sequence.

Usually prior knowledge is required for defining the sequence units to be modeled or deciding what kind of representations the states should have. There should be a balance between the specificity of the models and the amount of the training data [13].

2.1.5 Probability of the Sequences

The joint probability of the state sequence q being generated by the HMM and the observation sequence O being generated by that state sequence is

$$P(O, q \mid \lambda) = \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}, q_t} . b_{q_t}(O_t) \quad \textbf{(2.9)}$$

Since the generating state sequence is unknown the actual probability of the observation sequence for the model λ is

$$P(O \mid \lambda) = \sum_{q} \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}, q_t} . b_{q_t}(O_t) \quad \textbf{(2.10)}$$

The probability can be computed using the forward-backward procedure (Baum,1972). The dynamic programming by Viterbi algorithm (Forney,1973) is commonly used to decode most likely state sequence behind the observations by the maximizing recursively the probability $P(O, q \mid \lambda)$. The estimation of the HMM parameters using the maximum likelihood (ML) criterion i.e. maximization of $P(O \mid \lambda)$ over λ, is done using the Baum-Welch algorithm (Baum and Petrie, 1966). Anyhow a simpler ML training can be obtained by replacing the maximization of $P(O \mid \lambda)$ by the maximization of the likelihood of the most probable state sequence obtained by the Viterbi search [13]. The optimal model is

$$\overline{\lambda} = \arg\max_{\lambda} \max_{q} P(O, q \mid \lambda) \quad \textbf{(2.11)}$$

This latter method is called the segmental K-means or Viterbi training and in can be shown (Rabiner et al,1986) to have same asymptotic behavior as the Baum-Welch training, but with less numerical difficulties [15].

The idea in the present work is that the HMMs are not trained for pre-defined linguistic segments, but the SOM will find the segments from the feature sequences in an unsupervised manner. The resulting models may then be generalized sequences, or they may represent smaller and more detailed parts of the sequences. The specificity and characteristics of the models are determined by the

properties of the input data [13]. Earlier experiments with the data-driven segment units have been presented in [16, 17, 18]. Generative topographic mapping (GTM) [19, 20] also has resemblance to the current work.

2.2 Reinforcement Learning Approach with Q-Learning

2.2.1 Reinforcement Learning Terminology

Q-learning (Watkins, 1989) is a form of model-free reinforcement learning. It can also be viewed as a method of asynchronous dynamic programming (DP). It provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains [21]. Learning proceeds similarly to Sutton's (1984; 1988) method of temporal differences (TD): an agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty it receives and its estimate of the value of the state to which it is taken. By trying all actions in all states repeatedly, it learns which are best overall, judged by long-term discounted reward. Q-learning is a primitive (Watkins, 1989) form of learning, but, as such, it can operate as the basis of far more sophisticated devices. Examples of its use include Barto and Singh (1990), Sutton (1990), Chapman and Kaelbling (1991), Mahadevan and Connell (1991), and Lin (1992), who developed it independently. There are also various industrial applications. This section presents the proof outlined by Watkins (1989) that Q-learning converges [21].

2.2.2 Q-learning Algorithm Procedure

Consider a computational agent moving around some discrete, finite world, choosing one from a finite collection of actions at every time step. The world constitutes a controlled Markov process with the agent as a controller. At step *n,* the agent is equipped to register the state *xn* (€ X) of the world, an can choose its action *an* (€ 2)1 accordingly. The agent receives a probabilistic reward *rn,* whose mean value (*RXn (an)* depends only on the state and action, and the state of the world changes probabilistically to *yn* according to the law [21]:

$$\Pr ob[y_n = y \mid x_n, a_n] = P_{x_n y}[a_n] \qquad \textbf{(2.12)}$$

The task facing the agent is that of determining an optimal policy, one that maximizes total discounted expected reward. By discounted reward, it means that rewards received *s* steps hence are worth less than rewards received now, by a factor of *ys* (0 < *y* < 1) [21]. Under a policy π , the value of state *x* is

$$V^{\pi}(x) \equiv R_{\pi}(\pi(x)) + \gamma \sum_{y} P_{xy}[\pi(x)].V^{\pi}(y) \quad \textbf{(2.13)}$$

because the agent expects to receive *(Rx(π (x))* immediately for performing the action *I* recommends, and then moves to a state that is 'worth' $V^{\pi}(y)$ to it, with probability $P_{xy}[\pi(x)]$ [21]. The theory of DP (Bellman & Dreyfus, 1962; Ross, 1983) assures us that there is at least one optimal stationary policy π * which is such that

$$V^{*}(x) \equiv V^{\pi^{*}}(x) = \max_{a}\{R_{x}(a) + \gamma \sum_{y} P_{xy}[a].V^{\pi^{*}}(y)\} \quad \textbf{(2.14)}$$

It is as well as an agent can do from state *x.* Although this might look circular, it is actually well defined, and DP provides a number of methods for calculating *V*(y)* and one π*(assuming that *(Rx(a)* and Pxy[a] are known. The task facing a Q, learner is that of determining a π * without initially knowing these values [21]. There are traditional methods (e.g., Sato, Abe & Takeda, 1988) for learning *(Rx(a)* and Pxy[a] while concurrently performing DP, but any assumption of certainty equivalence, i.e., calculating actions as if the current model were accurate, costs dearly in the early stages of learning (Barto & Singh, 1990) [21]. Watkins (1989) classes Q-learning as incremental dynamic programming, because of the step-bystep manner in which it determines the optimal policy [21]. For a policy π, define Q, values (or action-values) as:

$$Q^{\pi}(x,a) = R_{x}(a) + \gamma \sum_{y} P_{xy}[\pi(x)].V^{\pi}(y) \quad \textbf{(2.15)}$$

In other words, the Q value is the expected discounted reward for executing action *a* at state *x* and following policy π thereafter. The object in Q-learning is to estimate the values for an optimal policy [21]. For convenience, define these as *Q*(x, a)* = *Q**(x, a), Vx, a.* It is straightforward to show that *V*(x)* = maxa *Q*(x, a)* and that if a* is an action at which the maximum is attained, then an optimal policy can be formed as π **(x)* = *a*.* Herein lies the utility of the Q, values—if an agent can learn them, it can easily decide what it is optimal to do [21]. Although there may be more than one optimal policy or a*,the Q* values are unique. In Q-learning, the agent's experience consists of a sequence of distinct stages or episodes [21]. In the nth episode, the agent:

• observes its current state *xn,*

• selects and performs an action *an,*

• observes the subsequent state *yn,*

• receives an immediate payoff *rn,* and

• adjusts its Qn-1 values using a learning factor an, according to:

$$Q_n(x,a) = \begin{cases} (1-a_n)Q_{n-1}(x,a) + a_n[r_n + \gamma.V_{n-1}(y_n)] & if \quad x=x_n \quad and \quad a=a_n \\ Q_{n-1}(x,a) & otherwise \end{cases} \quad \textbf{(2.16)}$$

Where

$$V_{n-1}(y) = \max_b \{Q_{n-1}(y,b)\} \quad \textbf{(2.17)}$$

is the best the agent thinks it can do from state *y.* Of course, in the early stages of learning, the *Q,* values may not accurately reflect the policy they implicitly define (the maximizing actions in equation 2.17). The initial Q, values, *Q,0(X, a),* for all states and actions are assumed given. Note that this description assumes a look-up table representation for the *Q,n(x, a).* Watkins (1989) shows that Q-learning may not converge correctly for other representations [21]. The most important condition implicit in the convergence theorem given below is that the sequence of episodes that forms the basis of learning must include an infinite number of episodes for each starting state and action [21]. This may be considered a strong condition on the way states and actions are selected—however, under the stochastic conditions of the theorem, no method could be guaranteed to find an optimal policy under weaker conditions [21]. Note, however, that the episodes need not form a continuous sequence—that is the *y of* one episode need not be the *x* of the next episode. The following theorem defines a set of conditions under which *Qn(x, a)* - *Q*(x, a)* as *n* - o. Define *ni(x, a)* as the index of the ith time that action *a* is tried in state *x* [21].

2.2.3 Discussions And Working Cases of Q-Learning

Two particular extensions to the version of Q-learning described above have been used in practice. One is the non-discounted case (γ = 1), but for a Markov process with absorbing goal states, and the other is to the case where many of the Q values are updated in each iteration rather than just one (Barto, Bradtke & Singh, 1991). The convergence result holds for both of these, and this section sketches the modifications to the proof that are necessary [21]. A process with absorbing goal states has one or more states which are bound in the end to trap the agent. This ultimate certainty of being trapped plays the role that γ < 1 played, in ensuring that the value of state *x* under any policy π, V^{π} *(x),* is bounded, that the difference between considering infinite and finite *(s)* numbers of actions tends to 0 as $s \to \infty$

[21]. This addition was pioneered by Sutton (1984; 1988) in his TD(λ) algorithm, in which a reward from a step taken *r* iterations previously is weighted by λ^r, where λ < 1. If the agent can remember the details of its learning episodes, then, after altering the learning rates, it can use each of them more than once [21]. This biases the Q-learning process towards the particular sample of the rewards and transitions that it has experienced. In the limit of re-presenting 'old' cards infinitely often, this reuse amounts to the certainty equivalence step of calculating the optimal actions for the observed sample of the Markovian environment rather than the actual environment itself [21].

2.3 Genetic Programming

Genetic programming is a stochastic search method suitable for addressing inductive learning tasks. Inspired by evolutionary process in natural living organisms, the genetic programming system maintains a population of programs to accomplish robust search. Genetic programming uses tree-structured chromosome inspired by the functional programming of LISP [22].

Tree-structured form represents the interpretation flow of input data. Genetic programming searches wanted solution using following procedure [21]. First, a population of chromosomes is randomly created to represent a pool of candidate solution. Second, they are assigned fitness value based on how close they come to the wanted function. Third, chromosomes are selected based on their fitness value and they are modified using genetic operators such as crossover and mutation. Fourth selected chromosomes comprise a new generation and are assigned new fitness values. This step is repeated until termination condition is satisfied [21]. More detailed explanation about each step of genetic programming is followed;

Table 2.1 : Pseudo Code and Algorithm EAs

Pseudo Code and Algorithm EAs :
INITIALIZE population randomly CALCULATE_FITNESS of each individual **while** *not* STOP_CRITERIA **do** SELECT parents RECOMBINE pairs of parents MUTATE offspring CALCULATE_FITNESS of new individuals REPLACE (some) parents by offspring **end_do**

2.3.1 Multi-Chromosomal Evolutionary Algorithms

There are several examples of genetic algorithms (GAs) which have described themselves as multi-chromosomal [23]. This approach, used multiple chromosomes to partition the search space and hence, unlike the other multi-chromosomal representations where the differing chromosomes represent different things and have different structures, this approach has identically structured chromosomes. The parameters of the functions to be solved were evenly distributed across varying numbers of chromosomes [23].

This early work not only uses multiple chromosomes, but also a diploid system and a bio-inspired crossover operator much like the one described here. Each gene in each chromosome codes for an exchange, a pair of numbers in the list to be sorted which will be compared and then exchanged if needed [23]. They use the diploidy inherent in their system to generate differing numbers of exchanges to alter the size of the sorting network they are evolving. If a pair of chromosomes differs at a particular point then both exchanges are expressed, otherwise only one exchange is expressed [23].

2.3.2 Dominance and diploidy in GA

In particular, standard GA problems in many fields are concerned with either optimizing an individual (or product) in a static fitness landscape, or solving a problem in which the fitness evaluation of different strategies is measurable against a schedule of static costs or benefits (either directly or indirectly, such as via a simulation) [24]. Meanwhile, problems with dynamic fitness evaluations have been only modestly approached with GA, because the "best" of the classic algorithms, or one that causes individuals to quickly converge on a solution, is not necessarily good at continuous adaptation in the face of a changing environment and fitness landscape [24].

Some attempts have been made to apply GA to changing-fitness problems, but all have had significant shortcomings. Bagley (1967) used the concept of diploidy, or pairs of homologous chromosomes, to model a population of individuals with a dominance map that determined which of two alleles would be expressed as the phenotype in a given situation, thereby presenting the opportunity of an individual to carry a hidden trait without expressing it [24]. However, the individuals were evaluated under static fitness conditions, the dominance maps converged too early,

and furthermore, the amount of information stored in the dominance maps was on the same order as all of the information stored in the genotypes of the individuals. In terms of memory storage and manipulation, Bagley effectively had to account for triploid individuals, instead of diploid ones [24]. Soon after, Hollstein (1971) used a dominance schedule with much better memory efficiency, but again used a static fitness landscape, and suggested that diploidy did not offer a significant advantage to fitness. Population diversity increased with diploidy, as latent recessive traits would emerge and create more variability between individuals [24].

More recently, Goldberg (1989) took a step toward solving dynamic fitness problems by using diploidy to create individuals in changing, often oscillating, fitness environments [24]. As predicted, the diploid populations were able to adapt to changing environments more readily than haploid populations, a result that many people attribute to the act of calling upon once-successful traits that were cached in combinations of recessive alleles that could re-emerge at random times and potentially exploit different environments [24].

Even when using a single chromosome in genetic programming, implementing a dominance system has been attempted [25]. In this work crossover is done by combining two subtrees to make one, then adding this to both parents to make two children [24]. The nodes which get to be expressed after the combination process are those with the highest dominance values. Dominance values are increased on nodes when a child produces a better result than its parents. However, they found there were problems integrating this type of system with the standard genetic programming representation [24].

The framework allowed for the creation of populations with varying numbers of subpopulations, and with migration of individuals between subpopulations. Mating pools were selected in tournament style, with tournament size based upon the desired certainty of the best-of-population individual reproducing [24]. In the reproduction stage, individuals in haploid populations could undergo single-point crossover, mutation, and reproduction. The handling of diploid populations, however, diverged from that of standard haploid GA [24].

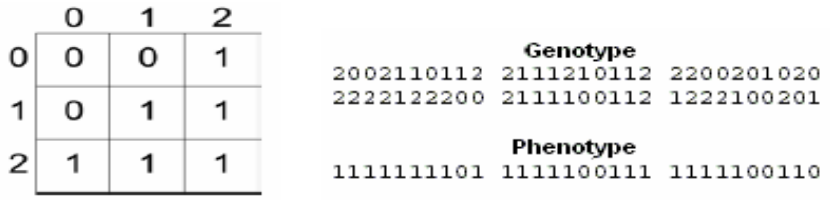

Figure 2.1 : Diploid chromosomal genetic string [24]

Dominance was determined in diploid populations according to the triallelic map created by Hollstein (1971) shown in Figure 2.1, in which binary bits at each locus were replaced with three choices: 0, 1, or 2. The phenotype of an individual, which represented the traits that the individual expressed in its phenotype, were determined in the following way: the '2' allele at a locus was globally dominant, and resulted in a '1' in the corresponding phenotype [24]. The '1' genotype allele also resulted in a phenotype of '1', but was recessive to the '0' allele, which returned a '0' in the phenotype. Because the probability of a '1' in the phenotype was twice as great as a '0' (six genetic combinations lead to a phenotype of '1' versus only three that lead to a '0'), the initial population creation during haploid runs was altered so that a '0' occurred in a haploid genotype half as often as a '1' [24].

2.3.3 Fitness Function

Fitness is the measure used by genetic programming during simulated evolution of how a program has learned to predict the outputs from the inputs - that is, the fitness of the learning domain [26]. Each chromosome has a corresponding fitness value and is selected for next generation using its fitness value. But, some problems such as robot control require at least two kind of fitness function. Existing genetic programming methods are practical enough to find an optimal solution in this domain [22]. To evolve this kind of complex behavior it is used a method called fitness switching [27]. Fitness switching is a method for evolving complex behaviors with genetic programming. It is based on the incremental learning procedure [22].

2.3.4 Selection Method

After the fitness of each chromosome has been determined by fitness function, it is determined whether to apply genetic operators to chromosome and whether to keep it in the population or allow it to be replaced [26]. The selector operator is used for this task. There are various selection operators. Fitness proportional selection specifies probabilities for chromosomes to be given a chance to pass offspring into the next generation [22]. A chromosome i is given a probability of equation 2.18 for being able to pass on traits [26].

$$P_i = \frac{f_i}{\sum_i f_i} \qquad \textbf{(2.18)}$$

Ranking selection is based on the fitness order, into which the individuals can be sorted. The selection probability is assigned to individuals as a function of their rank in the population. Tournament selection is not based on competition within the full

generation but in a subset of the population. A number of individuals, called the tournament size, is selected randomly, and a selective competition takes place [22]. The traits of the beter individuals in the tournament are then allowed to replace those of the worse individuals [26]. For our problem, we select fitness proportional selection operator.

2.3.5 Crossover

A binary variation operator is called recombination or crossover. As the names indicate such an operator merges information from two parent genotypes into one or two offspring genotypes [28]. Similarly to mutation, recombination is a stochastic operator: the choice of what parts of each parent are combined, and the way these parts are combined, depend on random drawings [29]. Again, the role of recombination is different in EC dialects: in Genetic Programming it is often the only variation operator, in Genetic Algorithms it is seen as the main search operator, and in Evolutionary Programming it is never used [28]. Perhaps this is why they are not commonly used, although several studies indicate that they have positive effects on the evolution. The principal behind recombination is simple that by mating two individuals with different but desirable features, it can be produced an offspring which combines both of those features [28].

Evolutionary Algorithms create a number of offspring by random recombination, accept that some will have undesirable combinations of traits, most may be no better or worse than their parents, and hope that some have improved characteristics [28]. Recombination is the superior form of reproduction, recombination operators in EAs are usually applied probabilistically, that is, with an existing chance of not being performed. It is important to note that variation operators are representation dependent [29]. That is, for different representations different variation operators have to be defined [28]. For example, if genotypes are bit-strings, then inverting a 0 to a 1 (1 to a 0) can be used as a mutation operator. However, if we represent possible solutions by tree-like structures another mutation operator is required [28].

2.3.6 Mutation

A unary variation operator is commonly called mutation [28]. It is applied to one genotype and delivers a (slightly) modified mutant, the child or offspring of it. A mutation operator is always stochastic: its output - the child depends on the out comes of a series of random choices. It should be noted that an arbitrary unary

operator is not necessarily seen as mutation [28]. A problem specific heuristic operator acting on one individual could be termed as mutation for being unary. However, in general mutation is supposed to cause a random, unbiased change. For this reason it might be more appropriate not to call heuristic unary operators mutation [28]. The role of mutation in EC is different in various EC-dialects, for instance in Genetic Programming it is often not used at all, in Genetic Algorithms it has traditionally been seen as a background operator to fill the gene pool with "fresh individuals", while in Evolutionary Programming it is the one and only variation operator doing the whole search work [28].

It is worth noting that variation operators form the evolutionary implementation of the elementary steps within the search space [29]. Generating a child amounts to stepping to a new point in this space. From this perspective, mutation has a theoretical role too: it can guarantee that the space is connected [28]. This is important since theorems stating that an EA will (given sufficient time) discover the global optimum of a given problem often rely on the property that each genotype representing a possible solution can be reached by the variation operators [28].

2.3.7 Replacement of Offsprings (survival selection)

The role of survivor selection or environmental selection is to distinguish among individuals based on their quality. In that it is similar to parent selection, but it is used in a different stage of the evolutionary cycle [28]. The survivor selection mechanism is called after having having created the offspring of the selected parents. As mentioned, in EC the population size is (almost always) constant, thus a choice has to to be made on which individuals will be allowed in the next generation [29]. This decision is usually based on their fitness values, favouring those with higher quality, although the concept of age is also frequently used. As opposed to parent selection which is typically stochastic, survivor selection is often deterministic, for instance ranking the unified multiset of parents and offspring and selecting the top segment (fitness biased), or selecting only from the offspring (age-biased) [28].

Survivor selection is also often called replacement or replacement strategy. In many cases the two terms can be used interchangeably [29]. The choice between the two is thus often arbitrary. A preference for using replacement can be motivated by the skewed proportion of the number of individuals in the population and the number of newly created children [28]. In particular, if the number of children is very small with respect to the population size, e.g., 2 children and a population of 100. In this case, the survivor selection step is as simple as to chose the two old individuals that are to

be deleted to make place for the new ones. In other words, it is more efficient to declare that everybody survives unless deleted, and to choose whom to replace [29].

2.4 Self Organizing Feature Map Neural network

Self-organizing feature maps (SOFM) learn to classify input vectors according to how they are grouped in the input space [30]. They differ from competitive layers in that neighboring neurons in the self-organizing map learn to recognize neighboring sections of the input space. Thus, self-organizing maps learn both the distribution (as do competitive layers) and topology of the input vectors they are trained on [31]. The neurons in the layer of an SOFM are arranged originally in physical positions according to a topology function. The functions gridtop, hextop or randtop can arrange the neurons in a grid, hexagonal, or random topology. Distances between neurons are calculated from their positions with a distance function. There are four distance functions, dist, boxdist, linkdist and mandist [32]. Link distance is the most common. These topology and distance functions are described in detail later in this section. Here a self-organizing feature map network identifies a winning neuron using the same procedure as employed by a competitive layer [31]. However, instead of updating only the winning neuron, all neurons within a certain neighborhood of the winning neuron are updated using the Kohonen rule. Specifically, we adjust all such neurons as follows [30].

$$ {}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) \qquad \textbf{(2.19)} $$

Here the neighborhood $N_i(d)$ contains the indices for all of the neurons that lie within a radius (d) of the winning neuron (i) [32].

$$ N_i(d) = \{j, d_{ij} \le d\} \qquad \textbf{(2.20)} $$

Thus, when a vector p is presented, the weights of the winning neuron and its close neighbors move toward p [31]. Consequently, after many presentations, neighboring neurons will have learned vectors similar to each other. To illustrate the concept of neighborhoods, consider the figure given below. The left diagram shows a two-dimensional neighborhood of radius d=1 around neuron 13. The right diagram (Figure 2.2) shows a neighborhood of radius d=2 [32].

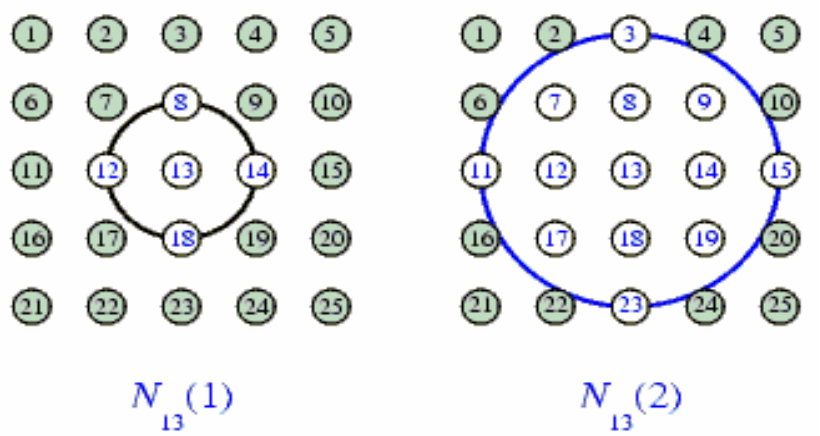

Figure 2.2 : Neighborhoods of the winnig neuron and best matching unit [32]

These neighborhoods could be written as ;

$$N_{13}(1) = \{8, 12, 13, 14, 18\} \quad \textbf{(2.21)}$$

$$N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\} \quad \textbf{(2.22)}$$

Note that the neurons in an SOFM do not have to be arranged in a two-dimensional pattern. It can be used a one-dimensional arrangement, or even three or more dimensions [32]. For a one-dimensional SOFM, a neuron has only two neighbors within a radius of 1 (or a single neighbor if the neuron is at the end of the line). Also it can be defined distance in different ways, for instance, by using rectangular and hexagonal arrangements of neurons and neighborhoods. The performance of the network is not sensitive to the exact shape of the neighborhoods [32].

2.4.1 Neural Network Architecture

The architecture for this SOFM is shown in figure 2.3. This architecture is like that of a competitive network, except no bias is used here [32]. The competitive transfer function produces a 1 for output element a1i corresponding to ,the winning neuron. All other output elements in a1 are 0. Now, however, as described above, neurons close to the winning neuron are updated along with the winning neuron. As described previously, one can chose from various topologies of neurons. Similarly, one can choose from various distance expressions to calculate neurons that are close to the winning neuron [32].

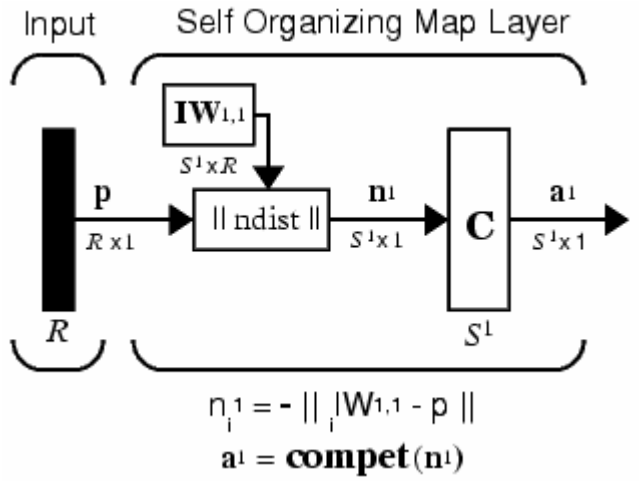

Figure 2.3 : Self-Organizing Map Neural Network [32]

2.4.2 Kohonen Learning Rule

The weights of the winning neuron (a row of the input weight matrix) are adjusted with the Kohonen learning rule. Supposing that the ith neuron wins, the elements of the ith row of the input weight matrix are adjusted as shown below [32].

$$ {}_i\mathbf{IW}^{1,1}(q) = {}_i\mathbf{IW}^{1,1}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{IW}^{1,1}(q-1)) \quad \textbf{(2.23)} $$

The Kohonen rule allows the weights of a neuron to learn an input vector, and because of this it is useful in recognition applications. Thus, the neuron whose weight vector was closest to the input vector is updated to be even closer [30]. The result is that the winning neuron is more likely to win the competition the next time a similar vector is presented, and less likely to win when a very different input vector is presented [31]. As more and more inputs are presented, each neuron in the layer closest to a group of input vectors soon adjusts its weight vector toward those input vectors [30]. Eventually, if there are enough neurons, every cluster of similar input vectors will have a neuron that outputs 1 when a vector in the cluster is presented, while outputting a 0 at all other times. Thus, the competitive network learns to categorize the input vectors it sees [32].

3 Behavior And Emotion Based Robot Control

3.1 Behavior Based Robot Control Approach

In the case of robotics, robot behavior is a direct connecting of the sensory information to the actuators. Behaviors do not have deliberative processing levels but may contain representational information as inputs or outputs [11]. Behaviors can be considered as unique agents which are basic control components of the mobile robot. The combination of this behavior modules constitute low-level component of the general architecture. Generally behavior modules are represented as parallel in this level.

Braitenberg vehicles give the basic understanding for the purely reactive behaviors. These robots were a thought experiment of Valentino Braitenberg [33], and later realized by some other scientists. These vehicles are a set of inflexible vehicles with direct connections of the sensors and motors (Figure 3.1). The sensors of "Fear" and "Aggression" robots are aversive to light (Vehicle 2). The motor closest to the light goes faster than the other. The "Aggression" robot hits the light source [11].

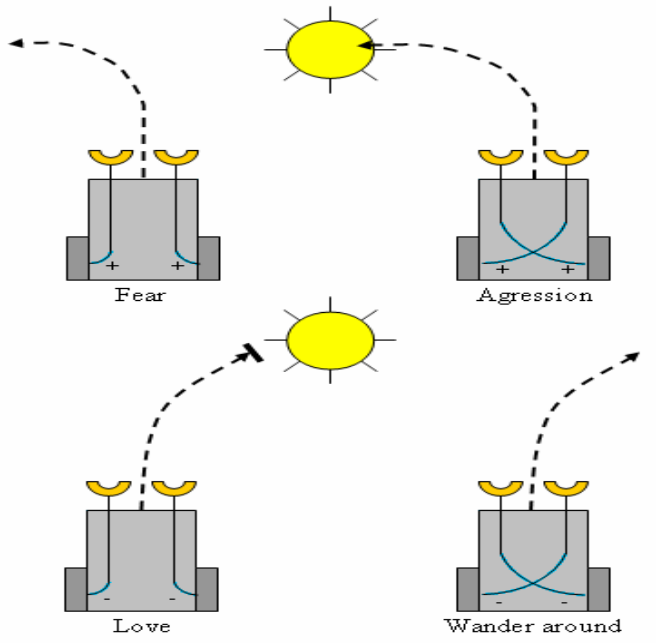

Figure 3.1 : Braitenberg Vehicles

The sensors of "Love" and "Wander around" robots are inhibiting the motor speed in case of strong light, and turns faster in weak light (Vehicle 3). The "Love" robot goes to the light source and stops at a close distance. The "Wander around" robot stays around the light source but also keeps traveling (Figure 3.1). By choosing some thresholds, nonlinear for the sensors and motors (Vehicle 4), adding some more light sources and sensors, different behaviors can be achieved [11].

3.1.1 Combining And Coordination Methodologies of Behaviors

The individual behaviors are dedicated to make certain jobs. These behaviors should be coordinated and used in parallel in order to have an intelligent system (Figure 3.2). The combination of these individual behaviors constitutes an undefined behavior to emerge that is called the Emergent Behavior [11].

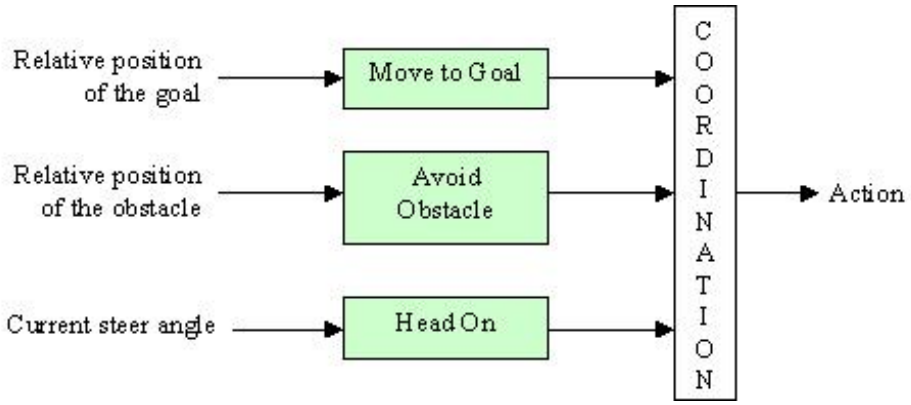

Figure 3.2 : Behavior Coordination

The resulting vectors of the behaviors are combined in a coordination mechanism. This coordination may be purely additive or may contain some strategies to coordinate the behaviors. Behaviors do not always give similar responses, sometimes there may be conflicts between the responses of different behaviors. There are two main types of coordination functions, competitive methods and cooperative methods [11].

3.1.1.1 Competitive methods

Different behaviors compete to be the only one whose response is used. The rules of competition are different in different approaches, but at last, only a single behavior's responses or a single response for all behaviors is applied to the motors, and the others are ignored (Figure 3.3) [11].

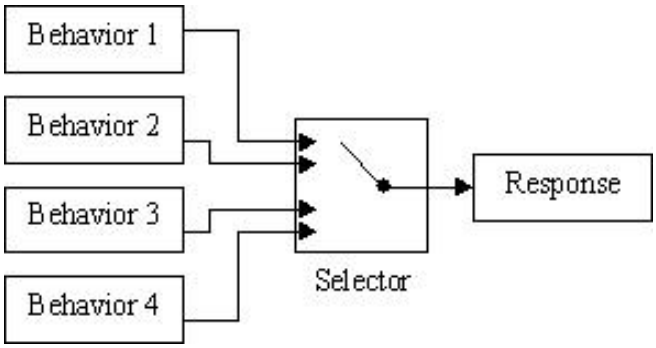

Figure 3.3 : Competitive methods

- Arbitration: There is a strict hierarchy between the behaviors. The behavior with a higher dominance competes over the one with a lower dominance [11].
- Action-selection: There is no predefined hierarchy; a behavior is selected according to the present situation or motivation of the robot [34].
- Voting: All the behaviors have predefined vote distribution for each response. The response with the most votes is selected to be executed, and other responses are ignored (Figure 3.4). This strategy selects one of the responses instead of selecting one of the behaviors [35].

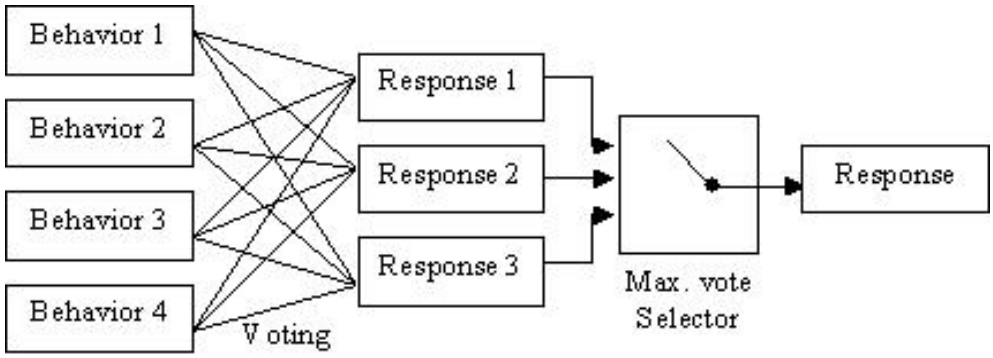

Figure 3.4 : Voting method

3.1.1.2 Cooperative methods

The advantage of cooperation is the ability to use the responses of different behaviors at the same time (Figure 3.5). Every behavior has some addition to the response at some orders. This order can be arranged by the gains of the behaviors. The resulting response may need to be limited, normalized or modified to avoid extreme conditions [11].

- Potential-fields: The simplest way to cooperate different behaviors is the vector calculation or superposition of the gained responses of each behavior [4].
- Fuzzy: The behaviors and responses are processed as fuzzy sets [36-38].

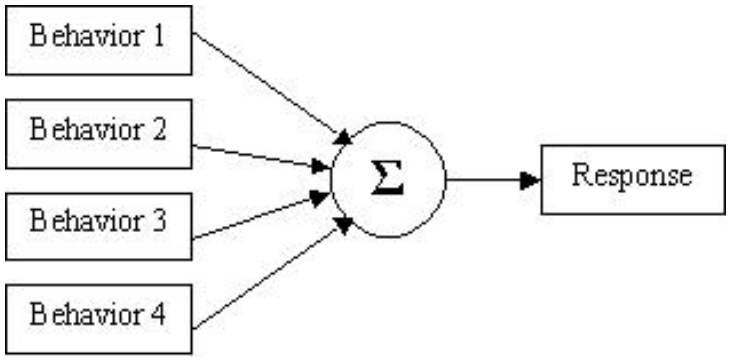

Figure 3.5 : Cooperative methods

3.1.2 Behavior-Based Architectures

The classical robot architectures lie on vertical “sense-plan-act” strategy. This property of the classical approach has some disadvantages in the real world applications especially because of their complexity, time consuming calculations, and costs [11]. Even this strategy is detrimental to the construction of real working robots and led robotics researchers in the wrong direction [2].

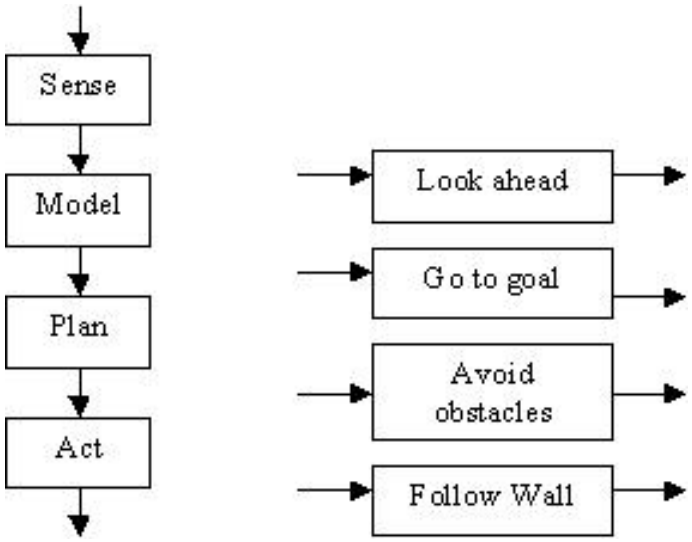

Figure 3.6 : Vertical (left) and horizontal (right) structures [11]

Behavior-based systems have a horizontal architecture (Figure 3.6). This type of reactive organization provides the behavior-based system to have several advantages in real-time applications because of its reactiveness and computation speed (Figure 3.7) [11].

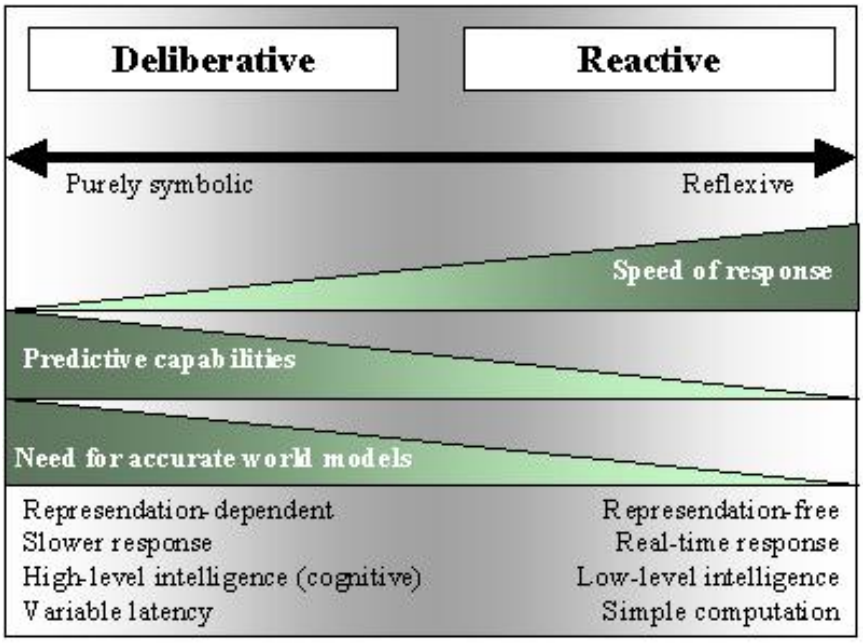

Figure 3.7 : Deliberative vs. Reactive [4]

3.1.2.1 Subsumption architecture

Subsumption architecture, developed by R. Brooks in mid-1980s, has a leveled organization of the behaviors. These levels have a hierarchical structure. This hierarchy is built by the coordination of the behaviors in that level (Figure 3.9). Lower levels never rely on the existence of higher levels [11]. Similar to competitive arbitration, the behaviors have two primary mechanisms for coordination:

- Inhibition: preventing a signal to be transmitted to the actuators
- Suppression: preventing and replacing a signal with a suppressing message.

The lowest level behaviors are called “Augmented Finite State Machine” (AFSM) and they may be reset, inhibited or suppressed by other active AFSMs (Figure 3.8) [11].

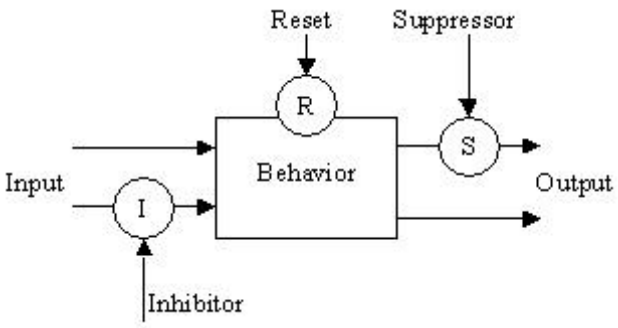

Figure 3.8 : AFSM

All AFSMs have different jobs and they perform their actions by their own perception. There is no global representation or model [11].

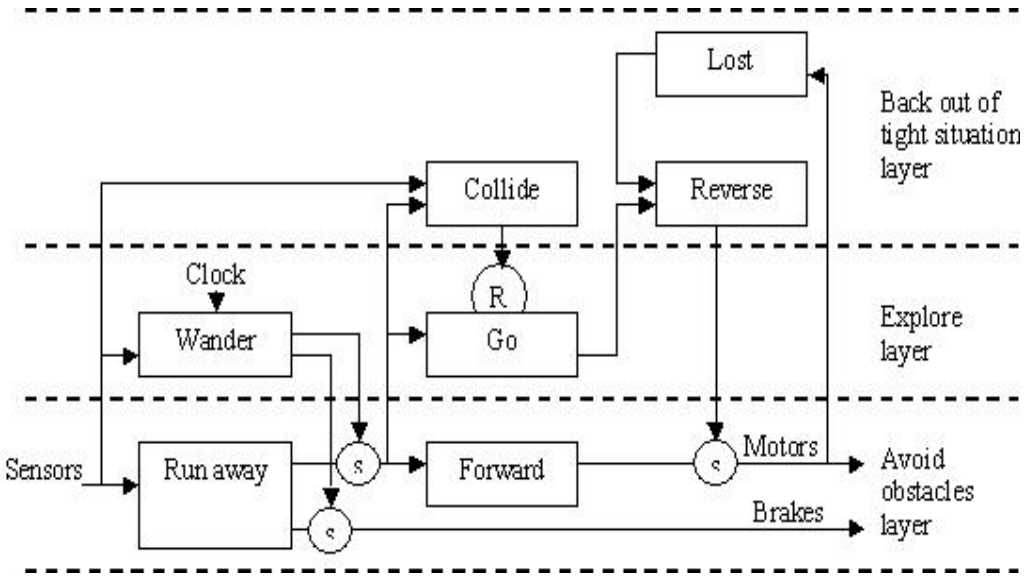

Figure 3.9 : Subsumption Architecture

One important drawback of the subsumption architecture is, the organization between the behaviors and levels gets more complicated by the increasing number of behaviors [11].

3.1.2.2 Motor schemas

Motor schema method [4] uses the potential fields method to coordinate the behaviors. Unlike the subsumption method, there is no predefined hierarchy between the behaviors, all the behaviors may contribute to the overall response of the robot. The software-oriented architecture makes it easy to modify [11].

There is a perceptual schema in each motor schema (MS). These perceptual schemas (PS) process the information for the motor schema and provide suitable stimuli (Figure 3.10). Each PS can use multiple sensors or outputs of other PS. This property enables the use of multiple sensors for a single sensoriomotor behavior [11].

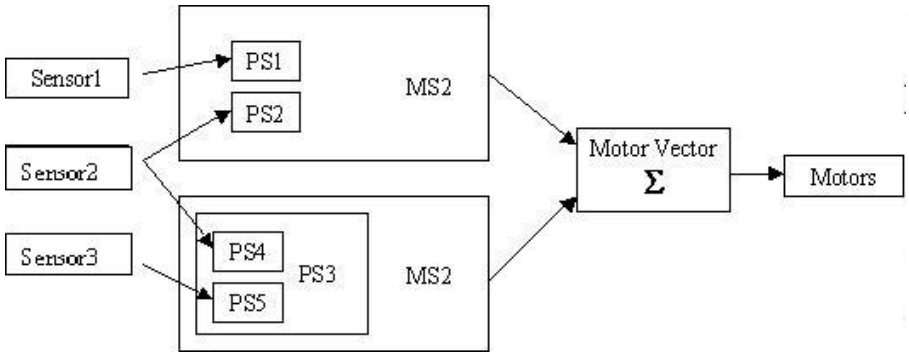

Figure 3.10 : Motor Schema Architecture

3.1.2.3 Other methods

There are several other methods used in behavior-based architecture, namely "Circuit Architecture" (Kaelbling and Rosenschein), "Action-selection" (Maes), "Colony Architecture " (Connel), "Animate Agent Architecture (Firby), "DAMN Architecture" (Rosenblatt), "Skill Network Architecture" (Zeltzer), and more [11].

The common subjects of all these architectures are their avoiding in using representations, using behaviors as building blocks and being reactive. The difference between the architectures is mainly the way they coordinate and manage the behaviors [11].

3.2 Emotion Based Architectures

Considering the nature as a model again, animals manage their behaviors through their motivations, or emotions [11]. The layering problem of multi-goal robot tasks in case of complex behaviors, can be partially solved by emotion-based architectures. Adding a new behavior becomes easier. This problem is solved by using emotions at a higher level that is organizing the behaviors.

For management of the multi-objective robot tasks, recent studies provide different and better solutions about artificial emotion based robot control approach to us. This paradigm serves several lemmas for behavioral selection policy and management of the behavioral sequences. also reinforcement of behavioral activations is realized as benefit from human behavioral and emotional processes by this methodology.

3.2.1 EMIB Architecture

EMIB computational architecture which is presented in 2002 can be example for emotion based control approach (Figure 3.11). This architecture is made of three levels such as behavior producing module, recommendation level and motivational level.

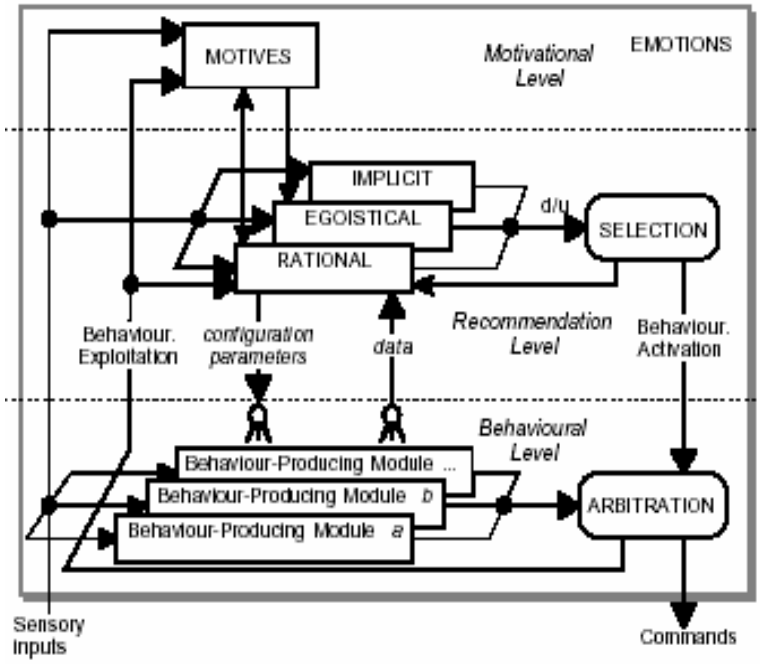

Figure 3.11 : EMIB Architecture [10]

3.2.1.1 Behavior producing module

The aim of behavior producing module is to construct relationship between sensors and actuators. This module determines behaviors which are predominated sub-modules as low level control units. It was considered that behavioral sub-modules run in parallel [10].

3.2.1.2 Recommendation level

Recommendation level is responsible from behavior selection and configuration of behavior producing module. Pre-dominated behaviors are located in the implicit module. Generated behaviors in the behavior producing module derived from instinctual module. Egoistical module realize pre-selection process for priority of the available behaviors such as Maslow's hierarchy of the needs theory (Maslow, 1954). The rational module is for behavioral recommendations based on innate or acquired knowledge about the world, to plan or to prepare the use of behavior producing modules exploiting such knowledge [10].

3.2.1.3 Motivational level

The motivational level monitors the goals and the overall states of the agent. Also it can be effective for the reinforcement or interruption of behaviors. In emotional module of this level, artificial emotions are determined. Artificial emotions are used to monitor how goals get satisfied or not [10].

3.2.2 ALEC Architecture

The ALEC (Asynchronous Learning by Emotion and Cognition) architecture which is presented in 2003 aims at a better learning performance by augmenting the emotion based architecture with a cognitive system which complements its current emotion-based adaptation capabilities with explicit rule knowledge extracted from the agent-environment interaction (Figure 3.12). The different learning capabilities of the two systems and their interaction have the potential to produce a more powerful adaptive system [1]. The cognitive system is based on the adaptive rule-decision system proposed within the CLARION model (Sun and Peterson, 1998) which allows learning the decision rules from the agent-environment interaction in a bottom-up fashion [1]. The ALEC architecture is made of several components such as behavior system, emotion system and cognitive system.

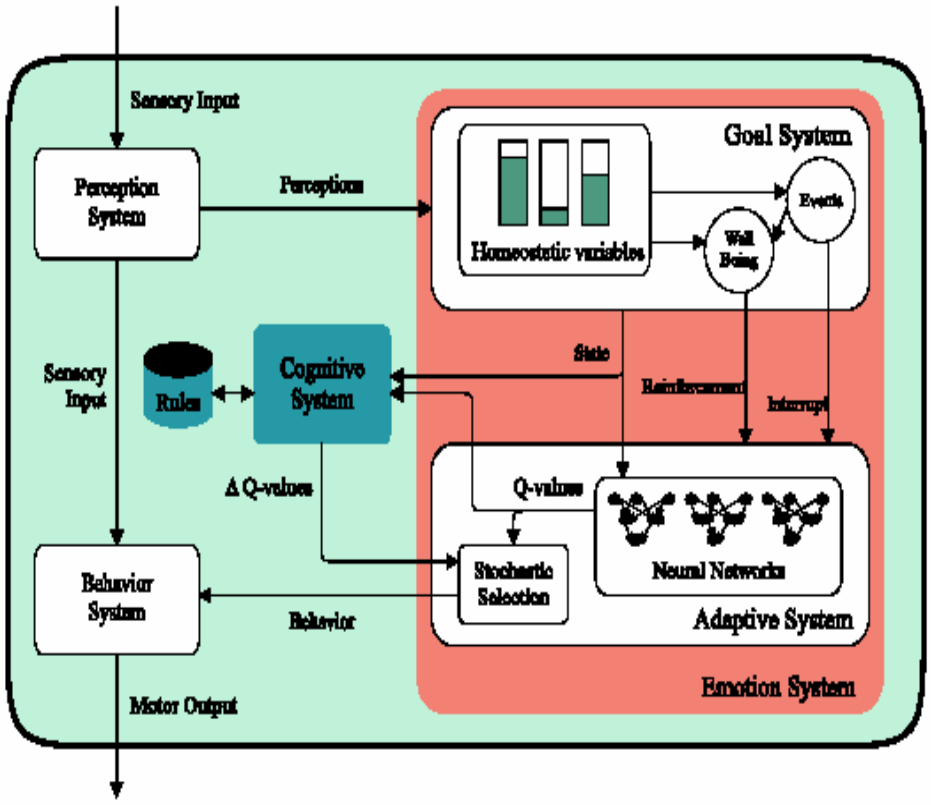

Figure 3.12 : ALEC Architecture [1]

3.2.2.1 Emotion system

The emotion system of ALEC is composed by the goal system and the adaptive system of the general architecture. The adaptive system implemented is a traditional reinforcement learning algorithm [1]. Through this algorithm, the agent learns

iteratively by trial and error the expected discounted cumulative reinforcement that it will receive after executing an action in response to a world state, *i.e.* the utility values. Also adaptive system is supported by generalization ability of the neural network. The Goal system is responsible for deciding when behavior switching should occur [1]. The goals are explicitly identified and associated with homeostatic variables. These homeostatic variables are associated with three different states: target, recovery and danger. The state of each variable depends on its continuous value, which is grouped into four qualitative categories: optimal, acceptable, deficient and dangerous. If a variable is in the target state it has a positive influence on the well-being, otherwise it has a negative influence which is proportional to its deviation from target values [1].

3.2.2.2 Cognitive system

Cognitive system maintains a dynamic collection of rules which allows it to make decisions based on past positive experiences. Each individual rule consists of a condition for activation and a behavior suggestion [1]. The activation condition is dictated by a set of intervals, one for each dimension of the input space. If a behavior is found to be successful in a particular state then the agent extracts a rule corresponding to the decision made and adds it to its rule set [1].

4 Artificial Emotion And Cognitive Based System Architecture

The architecture is made of three levels: the behavioural level, the coordination level and the motivational level (Figure 4.1). Proposed system was inspired by EMIB robot control architecture. However some contributions were considered on this system.

The behavioural level of the architecture is made of behaviour-producing modules, connecting sensory information to actions. However behavioral system of the proposed architecture does not have certain predominated behaviors. Unknown and unpredictible behaviors can be created in this behavioral system.

Thus the coordination level reconfigure behavioral parameters and different behaviors are derived as endless from available behaviors of the instinctual module. Priorities of the behavioral actions are determined by the priority filter module in the coordination level. Behavioral selection module of this level employ stochastic dynamic model based on hidden Markov model The coordination level is responsible for changing the selection of behaviours or to reconfigure them to make the system behave appropriately according to its goals and the situations it encounters in the world [10].

State - space hidden Markov model based Emotion-Motivation level compute and predict emotional observations as stochastic discrete events. The emotional module of this level store sequences of the behavioral states which represent emotions. Motivation module determine intense of performing behavioral tasks during time frame. According to goals and needs of the mobile robot, this idea supports that a certain behavioral gain coefficient is applied to each behavior of the sequence as adaptive.

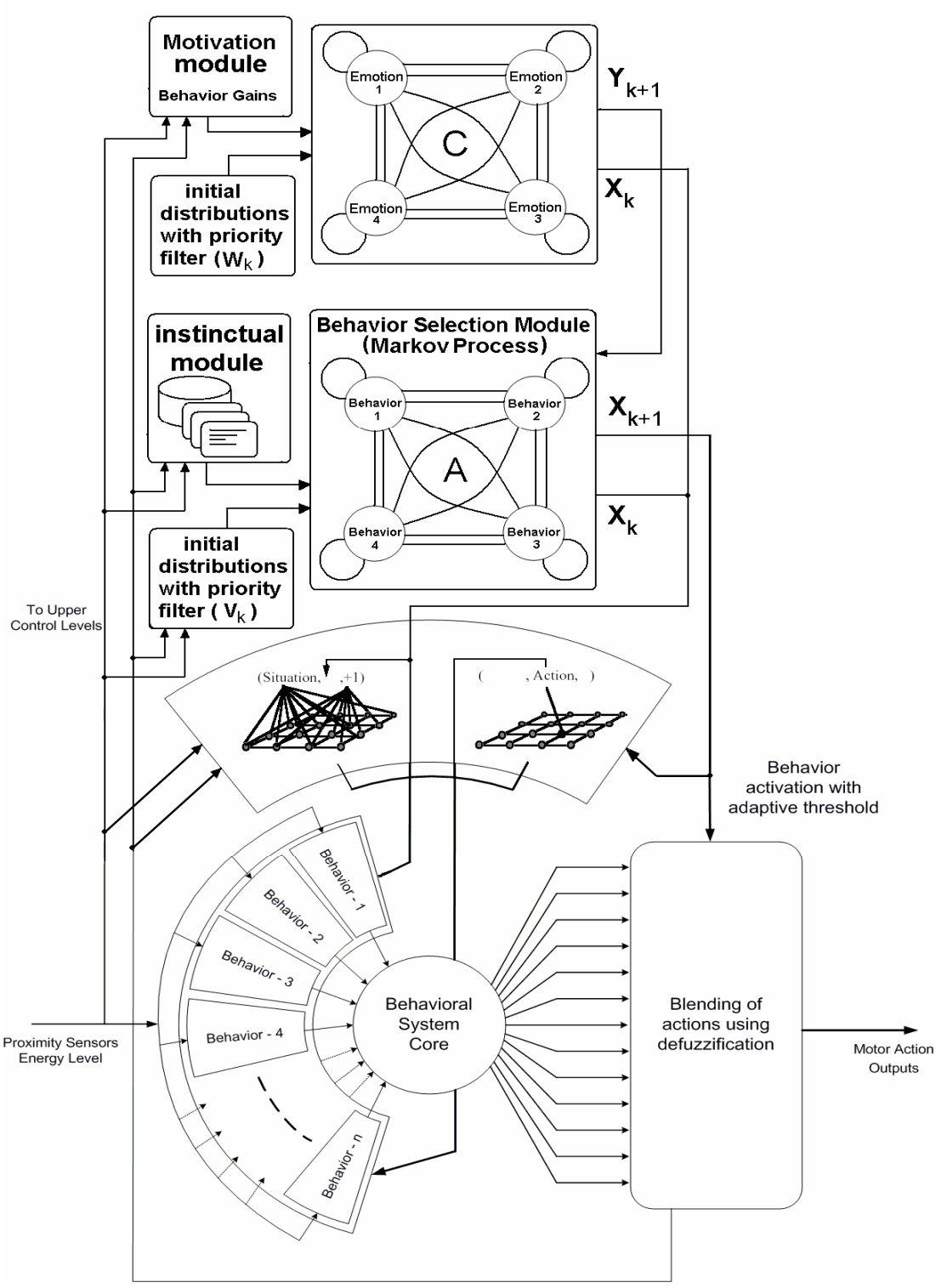

Figure 4.1 : Emotion And Cognition Based General Architecture

Working time frame of mobile robot determines duration of the behavioral activity according to short-term and long-term goals. In time domain, all of the cognitive events such as behaviors, emotions and others are located in this form;

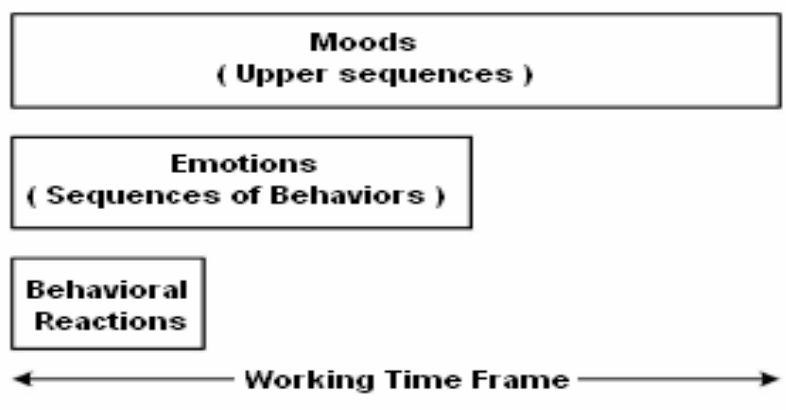

Figure 4.2 : Working time frame of the cognitive events [39]

The agent architecture has short-term and long-term goals. Low level primitive behavioral actions perform short term tasks [1]. Emotions realize as longer term task sequences than behaviors (Figure 4.2).

- it has multiple goals which may conflict with each other;
- there are situations in which the agent needs to temporarily overlook one goal in order to accomplish another successfully;
- a sequence of different behaviors may be required to accomplish a certain goal;
- the behaviors are unreliable: they may fail or they may lead the agent to undesirable situations;
- appropriate duration of the behaviors is undetermined, it depends on the environment and on their success [1].

Finally, SOFM Neural network can be considered as main root of whole architecture for designing robot control and cognitive system. According to this paradigm, all states of this system such as behaviours, emotional and motivational influences, instinctual or coordinational layer states composed into the proposed neural network architecture. Each layer or block element of robot cognitive system decomposition represents a neural layer of this architecture.

4.1 Behavioral System Module

Behavior-producing module is used as basic control component that is selected and modified according to the intentions (as derived by the higher decision levels) of the agent. These intentions are influenced by the situation perceived, the need to accomplish specific goals over time, and knowledge innate or acquired about the world [10].

A behavior pattern is a unique and independent piece of alive deliberative system having a complete adaptive function. Therefore behavior-producing modules allow the agent to respond in particular ways to situations encountered in the environment. These modules all run in parallel and their resulting commands are combined using an arbitration mechanism to generate the control actions of robot [10].

Behavioral search space is proposed as behavior producing module. This model is a black box approach for achieving to goals of robot. Candidate solutions (behaviors of mobile robot) are initialized randomly first. Behaviors of mobile robot have not been described previously. Sensor datas (stimulus) as input signal are applied to this module (behaviors). Components of each behavior have encoded into the diploidal chromosome structure. This structure is represented a unique transfer function of non-linear neural cell. Therefore responses of the robot behaviors can be trained and behaviors can be learned for transmitting feedback information to tuning first initialized behaviors and newly added (showing up oneself) .Determination of this search space of behavior module, selection of boundries is big problem what depends. The size of behavioral search space comprises to anticipated all probabilities for this behaviors. There is exact relationship between selection of search space boundries and input sensor range (perception limit).

There are three type sub-search space of behavior module. These are perception space, rule space and decision space. Perception space and decision space have continious characteristic as one component of whole behavior and decision making process. However rule space of behavior module has discrete feature. Behavioral system of mobile robot can be considered as fuzzy reasoning process. According to this approach perception search space corresponds to fuzzification phase of behavioral system. In the same way, decision search space corresponds to defuzzification phase of behavioral system. Rule base search space neural layer represents fuzzy inference engine mechanism. At the same time, it can be considered as core of behavior producing process. This section includes inner states which belongs to behavior producing module and accumulates rules of behaviors into the self memory.

In this module, main purpose is to creating best adaptive behavior learning environment and dynamic memory for mobile robot (Figure 4.3). Emotional states, instinctual, mood and other modules as upper control architectures play important role on the behavior producing module. Emotion based programming approach is helpful for tuning and learning behaviors. According to priority of agent needs,

instinctual module can act behavior producing module by sequental programming process.

Combining abstract reasoning processes with behavioral modules, must still allow emergence to take place at the behavioral level, and should also be present at the higher decision levels. For instance, by only using one module (like a planner) to reconfigure and modify behavioral modules over time, a strong dependency is created on the accuracy and the adequacy of the higher decision module [10].

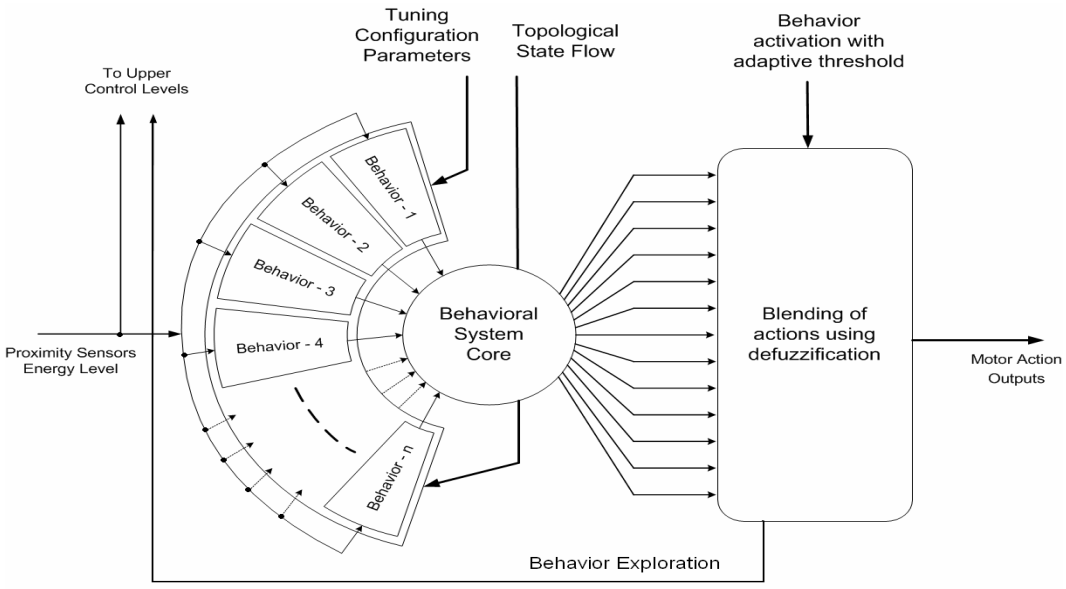

Figure 4.3 : Behavioral system

Behavior producing module allow to establish interaction of stimulus-response which can be called as relationship between sensor and motor (Figure 4.3) [10]. Another hard challenge arise from the need to couple the internal dynamics with the sensor-motor flow. Behaving agents receive a real time continous flow of sensor-motor states. From this huge amount of information they should be able to extract a limited number of states that capture useful features of the agent (behavior) / environment interaction and that can be used to later affect their behavior. Beside that, agents should be able to continously update their internal states by taking into account changes occuring in the agent/environment relation and/or in the environment itself. In other words, internal states should change according to a given dynamics that should be coupled with the dynamics of the sensor-motor flow [4]. Also this approach provide great emergence to us. The way of use of emergence would be in deriving knowledge about the world. Most deliberative approaches derive knowledge for reasoning about the world from sensor inputs and actions taken by the agent [10].

Since behavioral modules are the low-level control blocks and that their use are driven by what emerges from the interactions with the environment, they can also serve as an abstract representation of what is experienced in the world. The purpose associated with each behavioral module can then be exploited for reasoning, grounding intentions to what the agent is experiencing in the world [10].

4.1.1 Initializing Behavior Memory

According to robot needs and its inner states, behaviors of robot have dynamically changed. While robot is executing own tasks which can be complex and confilicting each other, some of these behaviors may disappear. Also new behaviors can be created. This behaviors are located in behavior producing module.

Network weights which are called as synaptic strenghts of neurons determine a topology of neural network (Figure 4.4). Weights of neural network and network topology introduce robot behaviors. Also weights of neural network of behavior module are held in the allocated memory for behavior generation. The synaptic weights of network are directly encoded on genotype such as a string of real values, a string of floating values or astring of binary values with a given precision [40].

Neural weights of behavior producing module can be assigned via different ways. Prior to training, each node's weights must be initialized. Typically these can be set to small standardized random values. The weights in the SOM from the accompanying code project are initialized so that 0 < w < 1. Generally, although this parameters can be assigned as randomly but this method is not very efficient approach. Networks with evolved initial weights learn significantly faster and better than networks with random initial weights [40].

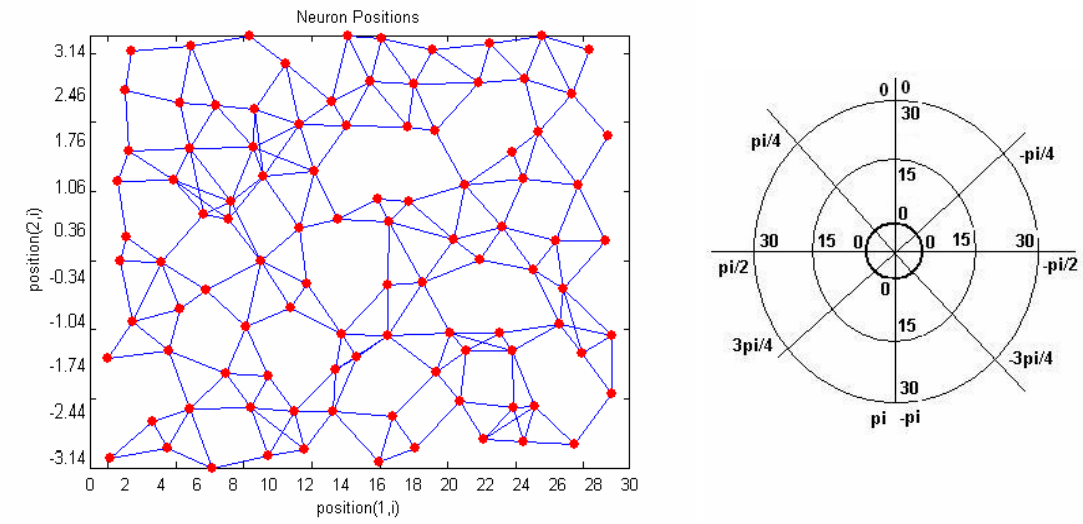

Figure 4.4 : initializing network weights

Perception layer network weights and its topology are constituted in a certain sensing limits (Figure 4.4). This sensing limit axes are represented as angular information (y axes) and distance information (x axes). Also sensing area which is called as perceptional search space describes the location of objects as fuzzy relational variable.

4.1.2 Network Architecture And Structure

The network is created from a 2D or 3D lattice of 'nodes', each of which is fully connected to the input layer. Figure 4.5 shows a very small Kohonen network of 4 X 4 nodes connected to the input layer (shown in green) representing a two dimensional vector (Figure 4.5) [41].

Every neuron i of the map is associated with an n-dimensional reference vector $m_i = [m_{i1}, \ldots m_{in}]^T$, where n denotes the dimension of the input vectors. The reference vectors together form a codebook. The neurons of the map are connected to adjacent neurons by a neighbourhood relation, which dictates the topology, or the structure, of the map [41].

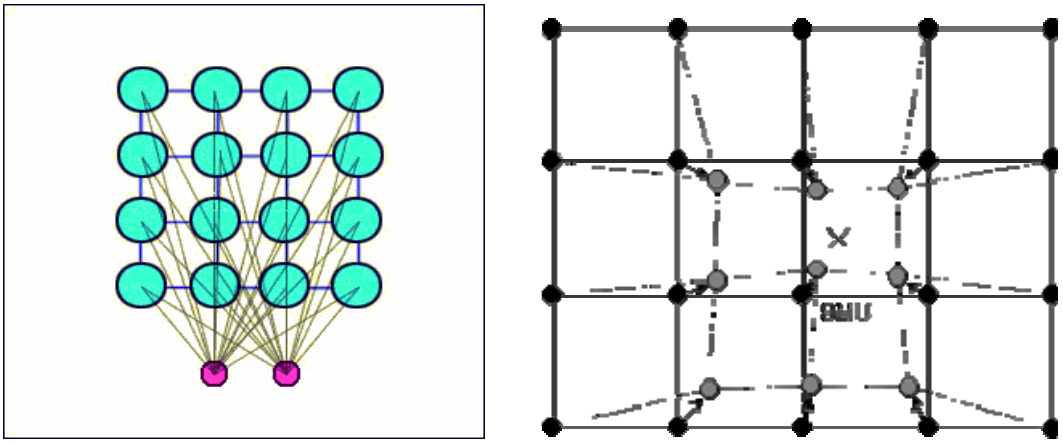

Figure 4.5 : Simple Kohonen network. And relationship with BMU [41].

Each node has a specific topological position (an x, y coordinate in the lattice) and contains a vector of weights of the same dimension as the input vectors [41]. That is to say, if the training data consists of vectors, *V*, of *n* dimensions:

$V_1, V_2, V_3 \ldots V_n$

Then each node will contain a corresponding weight vector *W,* of *n* dimensions:

$W_1, W_2, W_3 \ldots W_n$

The lines connecting the nodes are only there to represent adjacency and do not signify a connection as normally indicated when discussing a neural network (Figure 4.5). There are no lateral connections between nodes within the lattice [41].

There is only one layer of neurones. Each neurone is connected to the inputs. A neighbourhood is defined for each neurone. The neighbourhood size depends on the application [41].

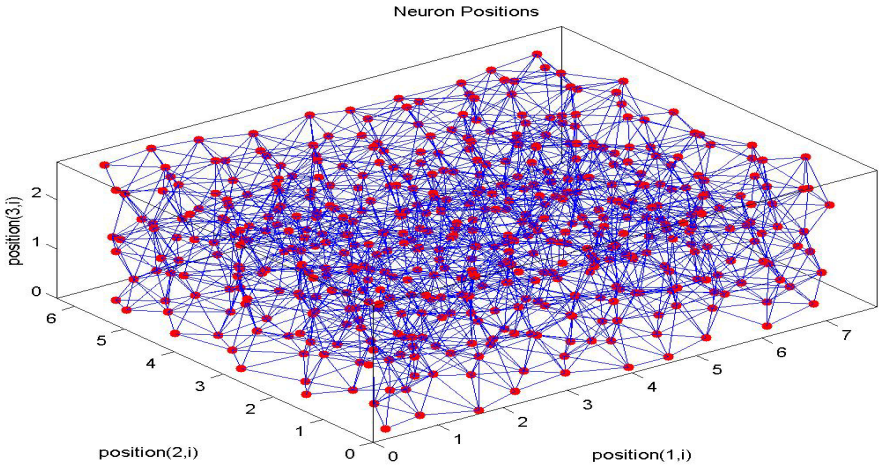

Figure 4.6 : Neural Network Architecture

Self-organising map model can constitute very different 3D fractal netwotk structure (Figure 4.6). During the learning phase, the network weights are tuned so as to reduce the difference between the input to the network and the selected neurone.

4.1.3 Adaptation And Learning Strategy of Behavioral System

The fitness function is computed using the residual error of the network before having being trained with self-organizing mapping (SOM) algorithm on given tasks [42]. The error function of neural network as fitness function is made minimization by evolution of fuzzy chromosomes. This operation provide to find best network weights.

4.1.3.1 Evolutionary operators : Crossover and Mutation

Crossover is a major operator of evolutionary methods. The traditional view is that crossover is primarily responsible for improvements in fitness [43]. Crossover modifies chromosome by exchanging a sub-tree of one parents tree with a sub-tree of another parents tree. Crossover is essential for generating a solution program in genetic programming [22].

Mutation modifies chromosome by changing one of node to another kind of nodes. But mutation is an necessary operator for finding hiding nodes, and mutation is often better for small populations, depending on the domain [43].

4.1.3.2 Self organized map neural network training algorithm

A SOM does not need a target output to be specified unlike many other types of network [31]. Instead, where the node weights match the input vector, that area of the lattice is selectively optimized to more closely resemble the data for the input vector is a member of the class. From an initial distribution of random weights, and over many iterations, the SOM eventually settles into a map of stable zones [41]. Each zone is effectively a feature classifier, so you can think of the graphical output as a type of feature map of the input space. Any new, previously unseen input vectors presented to the network will stimulate nodes in the zone with similar weight vectors [41].

Training occurs in several steps and over many iterations:

1. Each node's weights are initialized.
2. A vector is chosen at random from the set of training data and presented to the lattice. One sample vector x is randomly drawn from the input data set and its similarity (distance) to the codebook vectors is computed by using e.g. the common Euclidean distance measure:

$$\| x - m_c \| = \min_i \{ \| x - m_i \| \} \quad \textbf{(4.1)}$$

3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).
4. The radius of the neighbourhood of the BMU is now calculated. This is a value that starts large, typically set to the 'radius' of the lattice, but diminishes each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighbourhood.
5. After the BMU has been found, the codebook vectors are updated. The BMU itself as well as its topological neighbours are moved closer to the input vector in the input space i.e. the input vector attracts them. The magnitude of the attraction is governed by the learning rate. As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero

according to the specified learning rate function type. Along with the learning rate, the neighbourhood radius decreases as well.

The update rule for the reference vector of unit i is the following:

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t).[x(t) - m_i(t)], & i \in N_c(t) \\ m_i(t), & i \notin N_c(t) \end{cases} \quad \textbf{(4.2)}$$

6. Each neighbouring node's (the nodes found in step 4) weights are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered.

7. Repeat step 2 for N iterations. The steps 2 and 5 together consitute a single training step and they are repeated until the training ends. The number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighbourhood function and the learning rate is calculated accordingly [41].

Learning projects the N dimensional space represented by the training data on the M dimensional space of the self-organising map (usually two dimensions). This projection respects the relative distribution of the training data. The neighbourhoods allow us to generate a "continuous" mapping of the N dimensions space on the self-organising map. Close N dimensional data are close in the M dimensions of the self-organising map [41].

After the training is over, the map should be topologically ordered. This means that n topologically close (using some distance measure e.g. Euclidean) input data vectors map to n adjacent map neurons or even to the same single neuron [44].

4.1.3.3 Calculating the best matching unit

To determine the best matching unit, one method is to iterate through all the nodes and calculate the Euclidean distance between each node's weight vector and the current input vector. The node with a weight vector closest to the input vector is tagged as the BMU [41].

The Euclidean distance is given as:

$$Dist = \sqrt{\sum_{i=0}^{i=n} (V_i - W_i)^2} \quad \textbf{(4.3)}$$

where V is the current input vector and W is the node's weight vector. As an example, to calculate the distance between the vector for the colour red (1, 0, 0) with an arbitrary weight vector (0.1, 0.4, 0.5) [41].

distance = sqrt($(1 - 0.1)^2 + (0 - 0.4)^2 + (0 - 0.5)^2$)

= sqrt($(0.9)^2 + (-0.4)^2 + (-0.5)^2$)

= sqrt(0.81 + 0.16+ 0.25)

= sqrt(1.22)

distance = 1.106

4.1.3.4 Determining the best matching unit's local neighbourhood

Each iteration, after the BMU has been determined, the next step is to calculate which of the other nodes are within the BMU's neighbourhood. All these nodes will have their weight vectors altered in the next step [41].

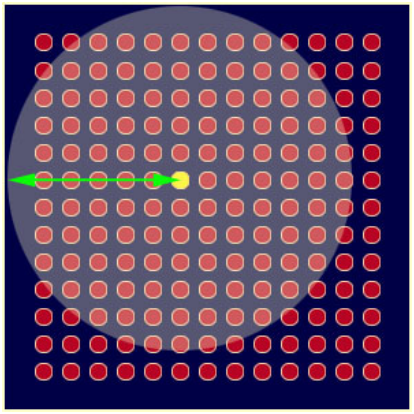

Figure 4.7 : The BMU's neighbourhood [41].

The neighbourhood shown above is centered around the BMU (coloured yellow) and encompasses most of the other nodes [41]. The green arrow shows the radius. A unique feature of the Kohonen learning algorithm is that the area of the neighbourhood shrinks over time (Figure 4.7). This is accomplished by making the radius of the neighbourhood shrink over time. To do this it is used the exponential decay function [41] :

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right) \qquad t = 1,2,3, \ldots \qquad \textbf{(4.4)}$$

Where the Greek letter sigma, σ_0, denotes the width of the lattice at time t and the Greek letter lambda, λ, denotes a time constant. *t* is the current time-step (iteration of the loop). In code name the value σ, radius of map, and it is equal to σ_0 at the commencement of training [41]. The value of λ is dependent on σ and the number of

iterations chosen for the algorithm to run. This value is set by the user. It can be used time constant and map radius to calculate the neighbourhood radius for each iteration of the algorithm using equation 4.4 [41].

The result shows how the neighbourhood decreases over time (the figure is drawn assuming the neighbourhood remains centered on the same node, in practice the BMU will move around according to the input vector being presented to the network) (Figure 4.8) [41].

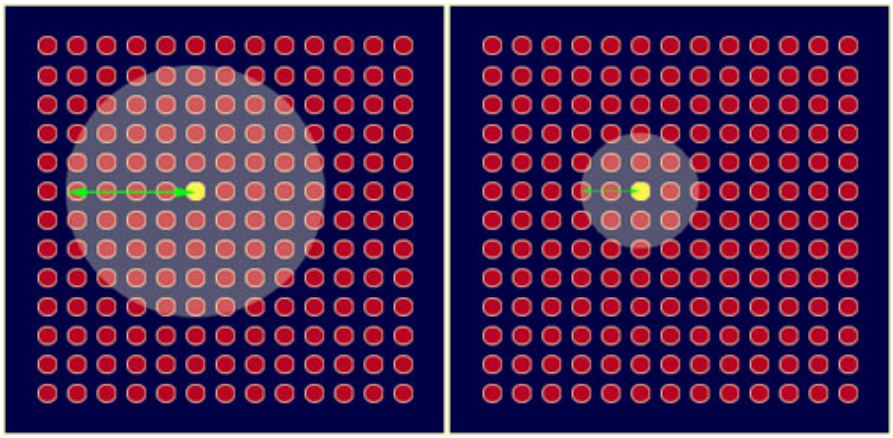

Figure 4.8 The ever shrinking radius [41].

Over time the neighbourhood will shrink to the size of just one node of the BMU. Now we know the radius, it's a simple matter to iterate through all the nodes in the lattice to determine if they lay within the radius or not. If a node is found to be within the neighbourhood then its weight vector is adjusted as follows [41].

4.1.3.5 Adjusting the weights

Every node within the BMU's neighbourhood (including the BMU) has its weight vector adjusted according to the following equation [41]:

$$W(t+1) = W(t) + L(t)(V(t) - W(t)) \quad \textbf{(4.5)}$$

Where *t* represents the time-step and *L* is a small variable called the learning rate, which decreases with time. Basically, what this equation is saying, is that the new adjusted weight for the node is equal to the old weight (W), plus a fraction of the difference (L) between the old weight and the input vector (V) [41].

The exponential decay function of the learning rate is calculated each iteration using the following equation:

$$L(t) = L_0 \exp\left(-\frac{t}{\lambda}\right) \quad t = 1,2,3, \ldots \tag{4.6}$$

The learning rate at the start of training, is set as 0,1. It then gradually decays over time so that during the last few iterations it is close to zero.

Actually at the edges of the BMUs neighbourhood, the learning process should have barely any effect at all. Ideally, the amount of learning should fade over distance similar to the Gaussian decay shown below (Figure 4.9) [41].

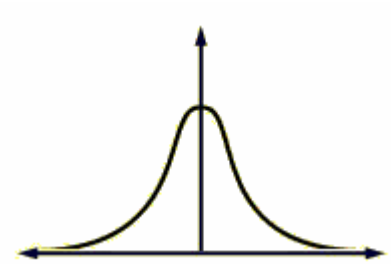

Figure 4.9 : gaussian decay learning rate function [41].

To achieve this, all it takes is a slight adjustment to equation 4.7 [41].

$$W(t+1) = W(t) + \Theta(t)L(t)(V(t) - W(t)) \tag{4.7}$$

Where theta, Θ, to represent the amount of influence a node's distance from the BMU has on its learning. Θ(*t*) is given by equation 4.8 [41].

$$\Theta(t) = \exp\left(-\frac{dist^2}{2\sigma^2(t)}\right) \quad t = 1,2,3, \ldots \tag{4.8}$$

Where *dist* is the distance a node is from the BMU and σ, is the width of the neighbourhood function as calculated by Equation (4.4). Additionally, please note that Θ also decays over time [41].

4.1.4 Perception System of Behavior Producing Module and Sensor Fusion

The fuzzification phase of behavior producing module converts input sensor datas to linguistic labels. But classical fuzzy control approach uses one dimension universe as membership functions of variables. Very different types of sensor information can involve in the mobile robot applications. These may be location, vision, energy level, thermal information, sound of environment, tactile (touch) perception knowledge. Therefore sensor informations need to be combined. It is convenient to combine

different sensory information for one purpose with membership functions which has two dimension universe (3D fuzzy surface) and rule based structure of Fuzzy Logic. Also the membership functions are helpful for combining different types of sensors because the distribution and universe of discourse of the membership functions can be arranged according to the output characteristics of each sensor [11].

If two dimension membership function (3D surface) can be expressed equation 4.9 (Figure 4.10);

$$Y(x_1, x_2) = \delta.(a.x_1 - b)^c + \gamma.(d.x_2 - e)^f - g \qquad \textbf{(4.9)}$$

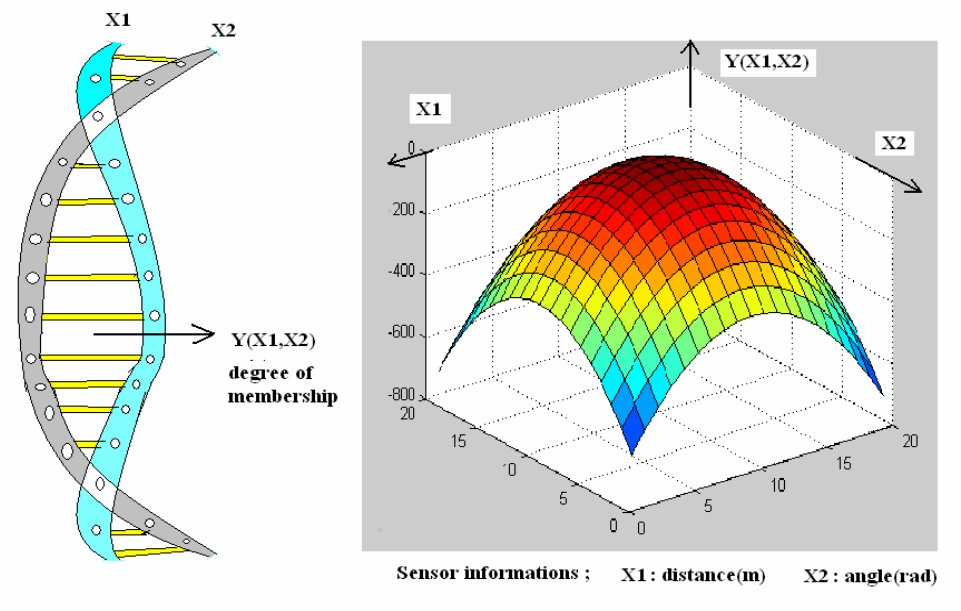

Figure 4.10: Diploid chromosomal genetic string corresponding 3D fuzzy relation surface

Such a membership function can be encoded as diploid chromosome string. Using genetic programming trees (GP trees) approach is suitable for encoding membership functions in the chromosome structure (Figure 4.11).

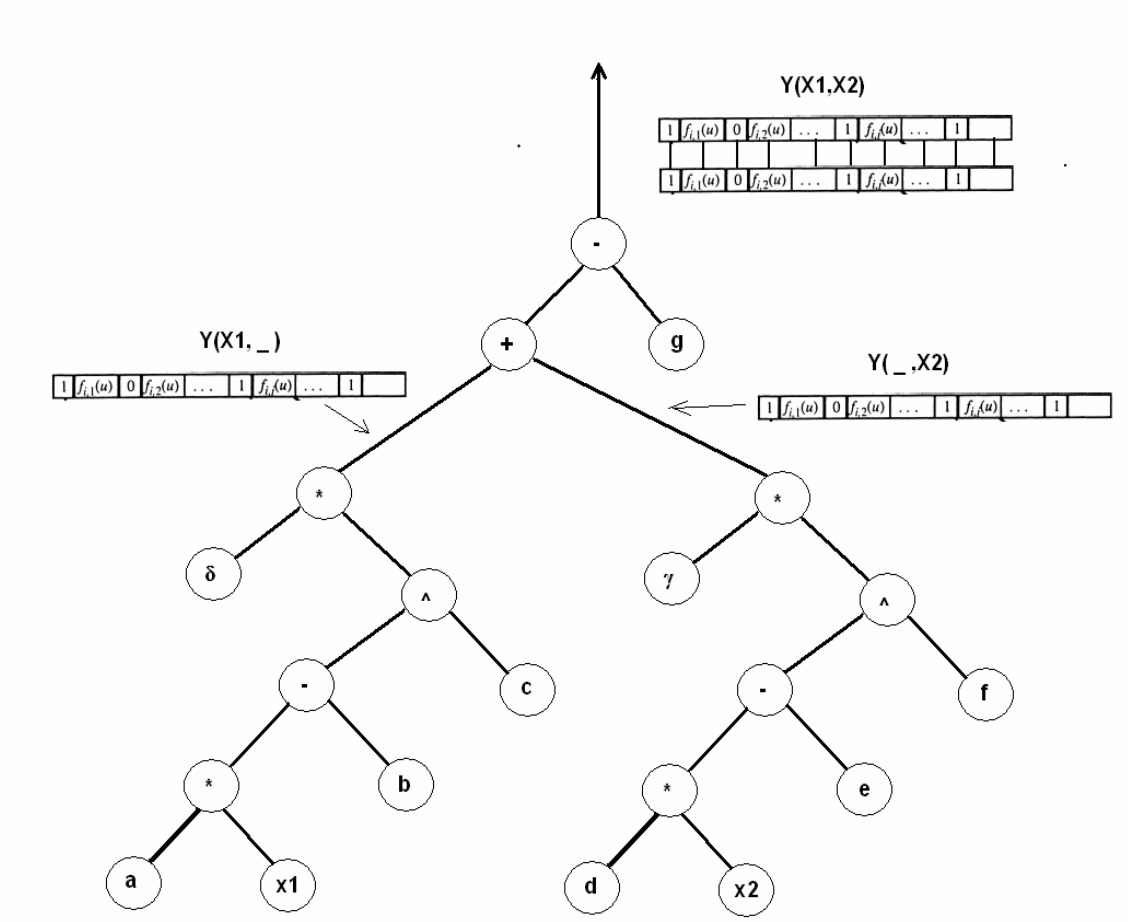

Figure 4.11: Genetic programming diploid chromosome tree

Each membership function belongs to related fuzzy variable. Many number of variables can be available for input (fuzzification) or output (defuzzification). For example these can be ;

Variable a : $Y_a(y_a1(x_1,x_2),\ y_a2(x_1,x_2),\ y_a3(x_1,x_2),\ y_a4(x_1,x_2), \ldots.,\ y_an)$

Variable b : $Y_b(y_b1(x_1,x_2),\ y_b2(x_1,x_2),\ y_b3(x_1,x_2),\ y_b4(x_1,x_2), \ldots.,\ y_bn)$

- -
- -

Variable m : m

4.1.4.1 Neuron model of perception system neural layer and network topology

Neurons of which belong to this section perform fuzzyfication process of robot behavior system. Neurons of perception system neural layers constitute a network topology which is called as fuzzy variable layer (Figure 4.12).

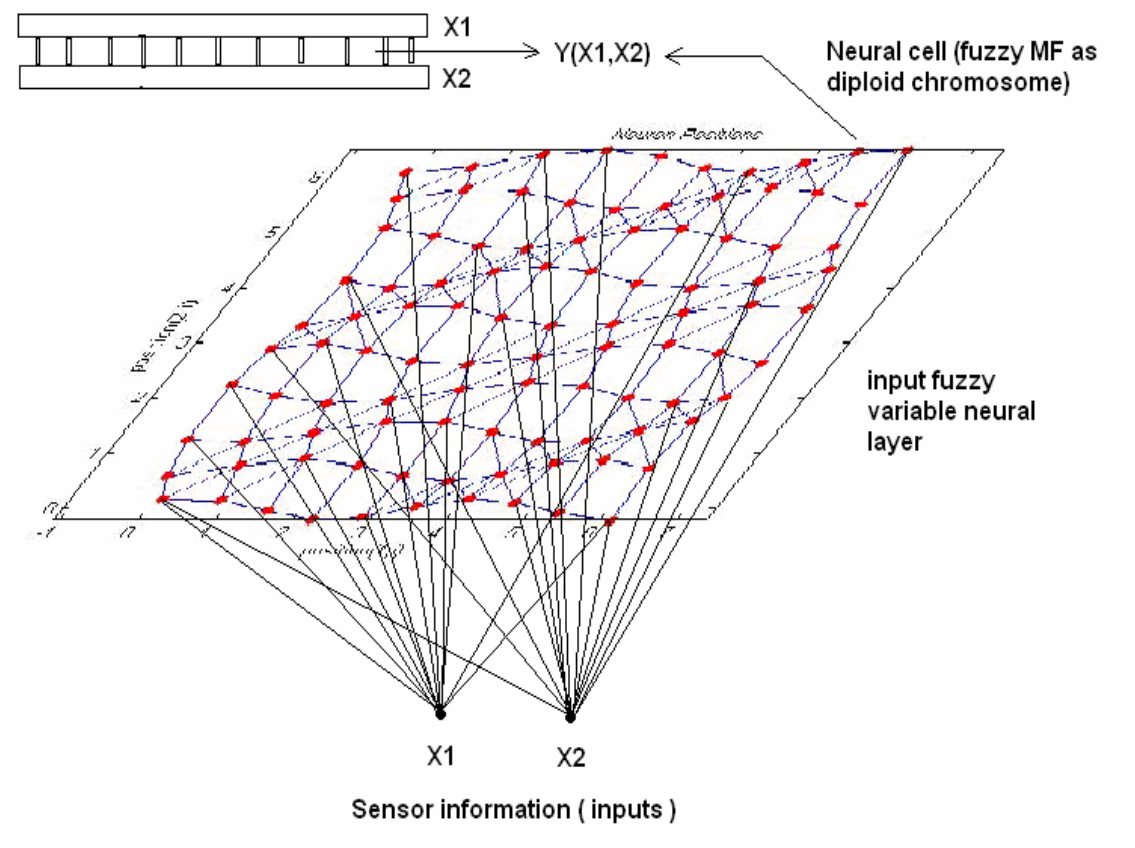

Figure 4.12 : Perception System Neural Layer

Each neuron determines a unique fuzzy membership function as genetic string. Someone of the input variables can be location of objects to mobile robot (Figure 4.12).

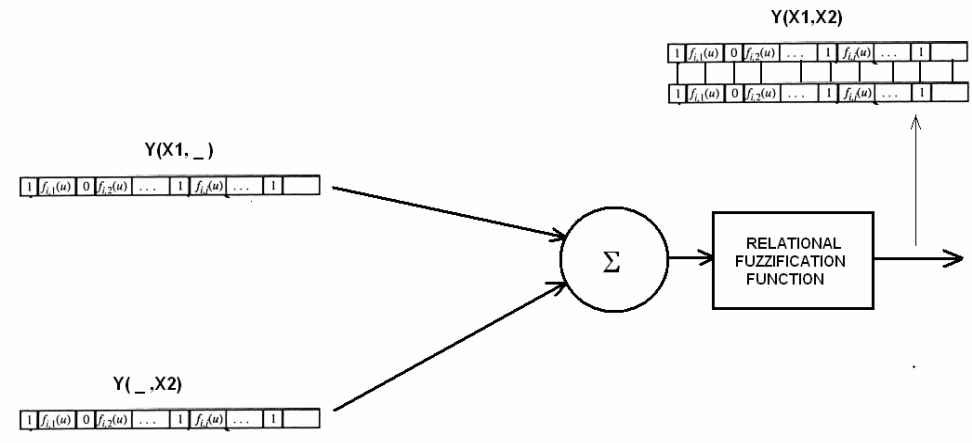

Figure 4.13 : Perception System Neuron model

Boundary of related sensor of corresponding x1 DNA string is from 0 to 30m. The boundary of related sensor of corresponding x2 DNA string is from –pi to pi (Figure 4.14). Such different sensor informations are distance to objects (may be obstacle,

goal, wall) information which relate with x1 and their orientation informations (angle) which relate with x2 (Figure 4.13).

Encoding of membership functions is realized in the diploid chromosome with Genetic programming representation. There are two type component for GP tree representation. Operators and terminals components contribute for representation non-linear membership functions.

Operators can be selected as { + , - , / , * , sin , cos , log , ^ }

Terminals can be selected as { x1 , x2 , [-1 1] }

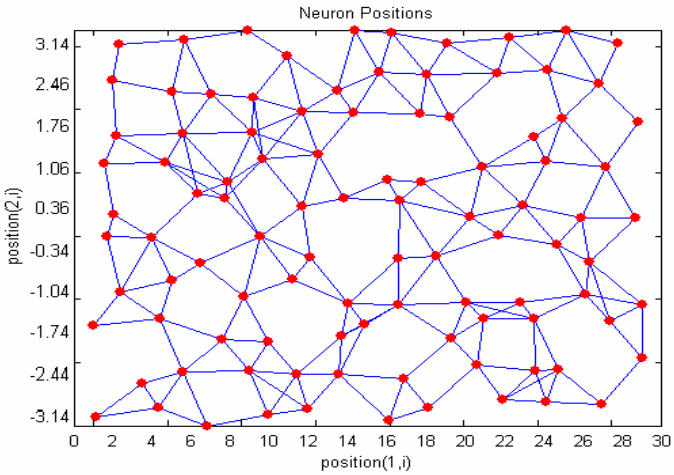

Figure 4-14 : Perception System Neural Layer Topology

4.1.5 Fuzzy Behavior Inference Engine System With Rule Space Neural Layer

Neurons in the rule space layer, perform fuzzy inference and reasoning process with signals which is coming from perception system. Basic logical expressions (”or”, ”and”, ”not”, ”xor”) and their combinations constitude complex recommendations. This recommendations apply to the 3D membership functions with diploid chromosome string which is constituted in the perception system neural layer (Figure 4.15).

Also this command components can be replaced into the ” IF-THEN” format. The way of using this format ;

IF (antecedent-1), (logical operator-1), (antecedent-2), (logical operator-2),

(antecedent-3), (logical operator-3) . . . . . THEN (consequent-1)

Inputs of the ” IF-THEN” format are expressed as antecedent label. Result of the ” IF-THEN” format is expressed as consequent label. Also this ” IF-THEN ” expression is called as rule. Rule space neural layer takes place from rule space neurons and

their network topology. Each rule space neuron of this section represents a " IF-THEN" rule. Rule space neurons constitute a network topology that can be called as rule base. All rule states of behaviors which are called as core of behaviors are expressed piece of a huge robot memory.

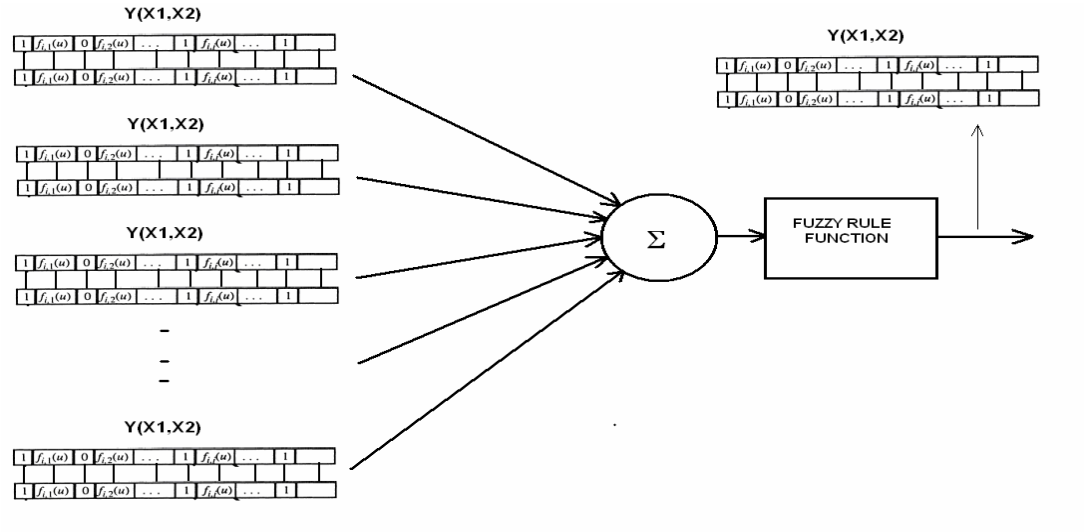

Figure 4.15 : Inference and Rule Base Neuron model

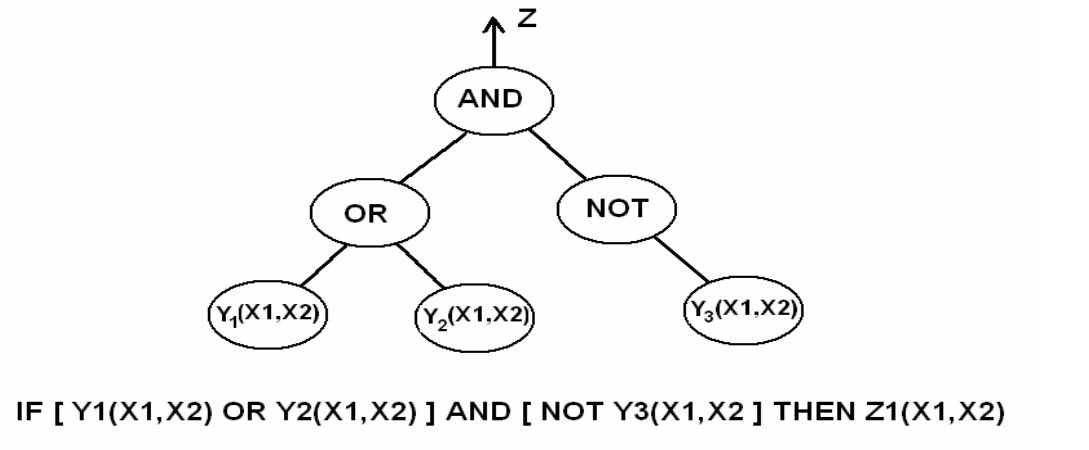

Figure 4.16 : Genetic programming chromosome tree of rule base neuron

Each neuron of fuzzy inference engine and rule base neural layer is encoded as IF-THEN rule by genetic programming approach (Figure 4.16). So this section of behavior producing module can be constituted a lot of unconstrained recommendations, it is required very large system memory storage (Figure 4.17).

Most of the actions in the behavior architecture are implemented using fuzzy rule-base or inference engine neural layer. Depending on the type of action, different neural informations of perception system layer which are chosen as input can be connected to different fuzzy rulebase neurons. However, the outputs from the fuzzy rule-bases for all actions are the same: the speeds of the wheels of the robot, which drive the robot to a desired posture.

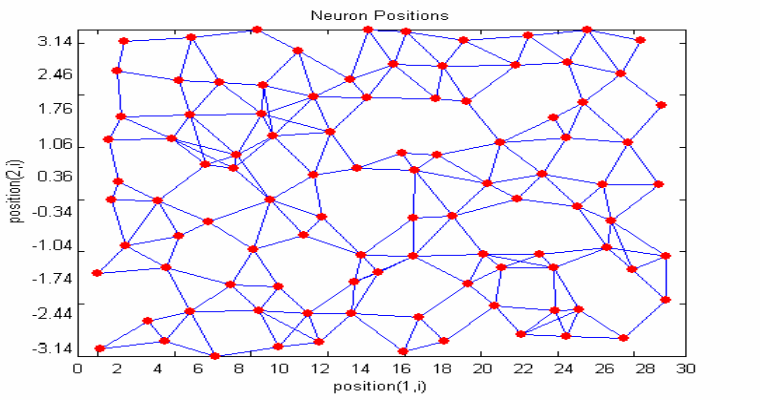

Figure 4.17 : Behavioral Core Inference And Rule Base System Neural Layer Topology

4.1.6 Decision Making Layer of Behavior Producing Module

Decision making neural layer realize two main duty. First of all, defuzzification process is implemented for obtain output motor action values by this layer. Also rules of different behaviors can be carried out in this layer. Desicion making neural layer provide combine multiple behaviors. Therefore this technique is used to get complex behaviors (Figure 4.18).

Defuzzification process provides the desired steering angle and velocity of robot data pairs for reaching target position, which is used to compute the shooting angle. The resulted shoot angle is used to activate the selected behavior.

Generally, two defuzzification methods are used for defuzzification. There are center of area (CoA) and max criterion [45]. For the CoA method, rules with positive strengths activate respective behaviors by recommending a pair of wheel steering angle and wheel speed. The center of area method is used to “merge” compatible behaviors and actions. The max criterion activates only one behavior at any time, which is determined by the rule with the highest strength. This method is used to coordinate behaviors and actions that are mutually exclusive [36].

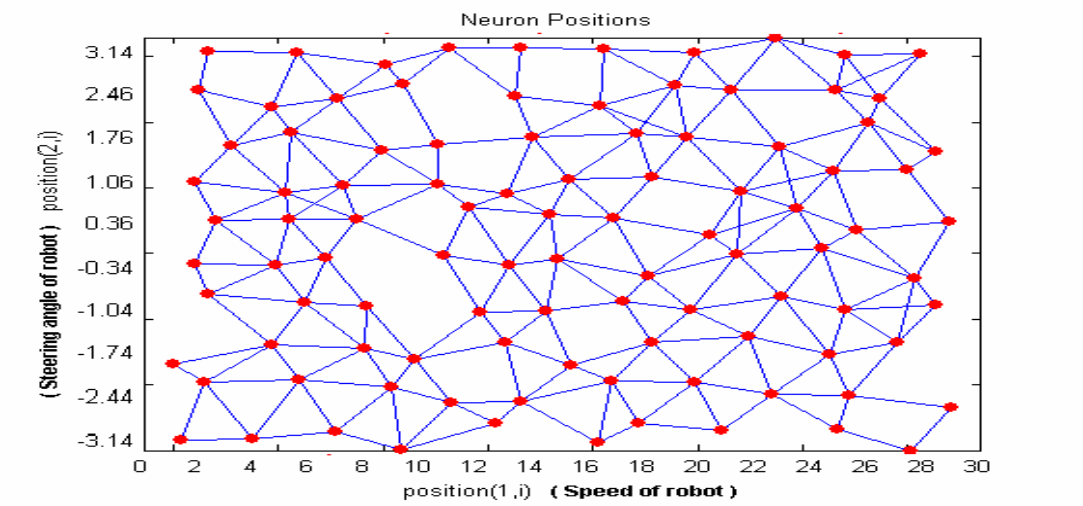

Figure 4.18 : Decision System Neural Layer Topology

Output fuzzy relations were devised as neurons in the lattice of neural network. Evolved weights of each neuron try to find to best candidate for reaching solution.

4.1.7 Created Behaviors and Case Studies

Too many different behaviors can be created in this behavior producing module. Behaviors which will be created in this module, are limited with system memory storages. Behavior producing module is responsible to learning of basic behavioral actions. Though this module perform self-organizing learning strategy, coordination layer tries to keep at certain boundries of emergent behavioral events with given reinforcement orders.

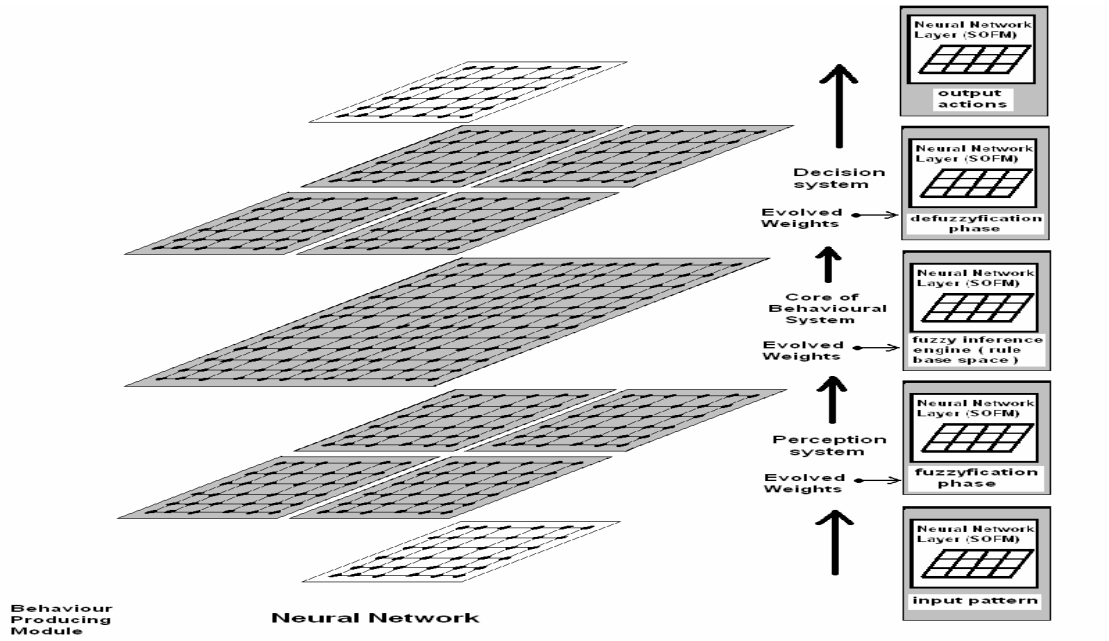

Figure 4.19 : Behavioral System General Neural Architecture and Data Flow

Also behavior producing module use relational fuzzy reasoning and inference process as embedded to behavioral system for performing behaviors in this proposed neural network architecture (figure 4.19). Because of using relational fuzzy inference process, fuzzy relations instead of membership fuctions (fuzzy sets) can be encoded as diploid chromosomal gene string representation by genetic programming procedure.

Therefore neural weights of the behavioral system are optimized by genetic programming aproach and obtained better weights for training and learning process.

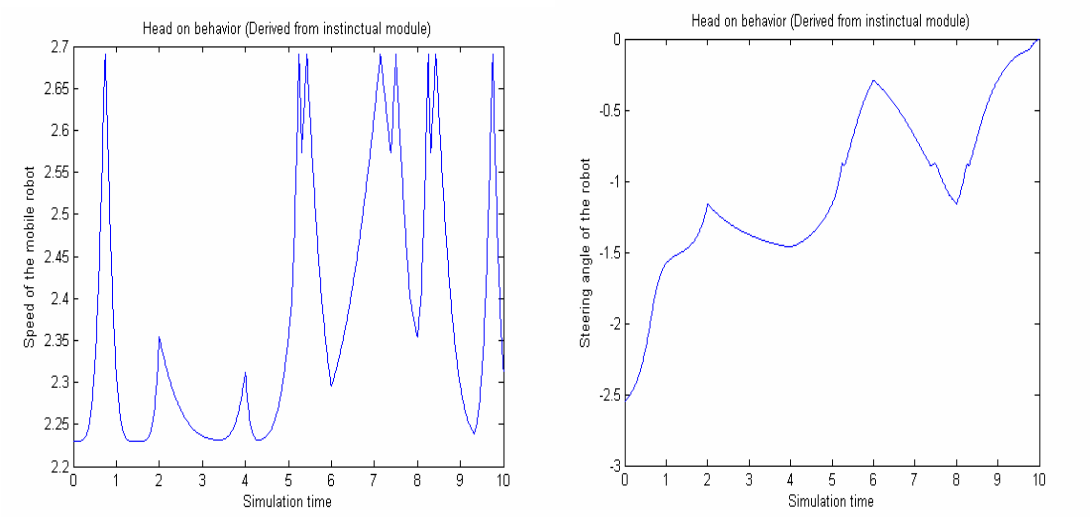

Figure 4.20 : The system response of the behavior

System responses which belong to an arbitrary behavior are presented in this section (Figure 4.20). Left figure express as speed information of motor output. And right figure represents as steering (angle) information of motor output.

According to physical and hardware structute of mobile robot, behavioral system can be improved. So the perception system of the behavior producing module can include more neural layer than one neural layer as parallel structure. In this section, one neural layer was used for perception system. This neural layer define locational information as relational fuzzy variable of the items according to mobile robot on the 2D grid network structure. Using different type sensors (vision-camera, thermal, laser, piezoelectic, mems. etc.) get needed to open new layers in the perception system. Also in the same way, the decision system of the behavior producing module can include more neural layer than one neural layer as parallel structure. In this section, one neural layer was used for decision system. Using different type actuators (different type motors, hydrolic-fluid actuators . etc.) get needed to open new layers in the perception system.

4.2 Coordination Level

Behaviour producing module can perform too many different action. However this module only does not order behaviours. Target vectors which is transmitted and received to behaviour producing module by coordination layer make a reinforcement effect upon behavioural system. Although behavioural system able to have self-learning ability, this self-learning process of behaviour producing module is tried and forced to keep at certain boundries by coordination layer.

Also target vectors can provide configuration of available behavioural system. This data flow between behaviour producing module and coordination layer tuning parameters of behaviours. Also state (data) flow which can be called as inner states of behavioural system modify neural topology information as morphological. Alteration of network morphology of behaviour producing module is effective for behaviour selection (transition) and modification (reinforcement) policy.

Coordination layer obtain rule orders of behaviours as this sensor datas are taken reference. Also input pattern can affect data orders of perception and decision system. Coordination level is monitored by the Emotion-Motivation level. Emotion-Motivation module give order behavior sequence information and their gain coefficients to the coordination level. Coordination layer is fed by input pattern (sensor informations).

There are three main duty of this layer. First of all, coordination layer must have provided configuration or recommendation of behaviour producing module. While this events are realized in the coordination layer, it should be protected emergence of this intelligent architecture. The second principal mission of this layer is to accommondate innate behaviour. This predominated behaviours can be resembled to instinctual process of alive.

Selection or transition of behavioural states is another significant mission.
Adaptive thresholds is applied to state transition probability matrix for switching of behavioural states by emotion layer. Relevant behavioural transition probability which remains under certain threshold is ignored.

Therefore relevant behaviour is considered to be inhibited (disactive). State transition probability matrix is used for prediction of behavioural inner states. This is a component of hidden markov model. For behaviour selection or transition policy, hidden markov model can be useful choosing. Hidden markov model is probabilistic sequental process as markov chains where states are not directly observed [13]. Multi tasking process is very important challenge for this proposed layer. Thus

implementation of multiple chains as behavioural state sequences can be accomplished to realization of menagement of multi objective behaviour selection process on the HMM.

Behaviour selection or transition policy can be supported by self organized reinforcement learning strategy. Therefore long term operation planning of robot is possible to provide for achieving multiple goals which is confilicting each other [1].

The developed learning architecture tries to maximize the reinforcement received by selecting between one of the available hand-designed or experience-gained behaviors [1]. At each trigger step, the agent may select between performing the behavior that has proven to be better in the past and therefore has the best utility value so far, or selecting an arbitrary behavior to improve its information about the utility of that behavior. The selection function used is based on the Boltzmann-Gibbs distribution and consists of selecting a behavior with higher probability, the higher its utility value in the current state [1]. According to robot needs, priority determination controller can make regulation effect on behavioural system. However priority of behaviours can be changed with the time by this layer.

The goal events, i.e. state change and prediction of state change, are also responsible for triggering the adaptive system for a new behavior selection [1]. The adaptive system only performs a reinforcement-learning step, i.e. evaluates the current behavior and makes a new behavior-selection, during trigger steps. On other steps, the previous selected behavior is kept and there is no policy update. The adaptive system is triggered when an event (one of the two described above) is detected [1].

4.2.1 Instinctual module

Instinctual module (Implicit) determine predominated behaviours and learning behavioural experimences derived from given stimulus. Several primitive behaviors such as obstacle avoid, move to goal, wander / search , head on, wall / trajectory following behaviors have been located in the instinctual module. This module aim that all other different behaviors are derived from this innate primitive behaviors. Also another mission of instinctual module is to eliminate uncertainities which are caused by unconstrained growing behaviors in the behavior producing module and to keep under control as converge to certain boundries. In the contents of this module, 25 number of basic fuzzy rule sets for each behaviors try to achieving offered tasks. Also for this, 25 number of input fuzzy relations (fuzzification phase) and 25 number of output fuzzy relations (defuzzification phase) were determined in

this module. All input and output fuzzy variables and their fuzzy relations is standart for all behaviors. Then rules of this predominated primitive behaviors collected in behavior determination matrix of instinctual module. For providing emergence, it was considered that matrix dimendions can be increased or decreased by adding or reducing behaviors.

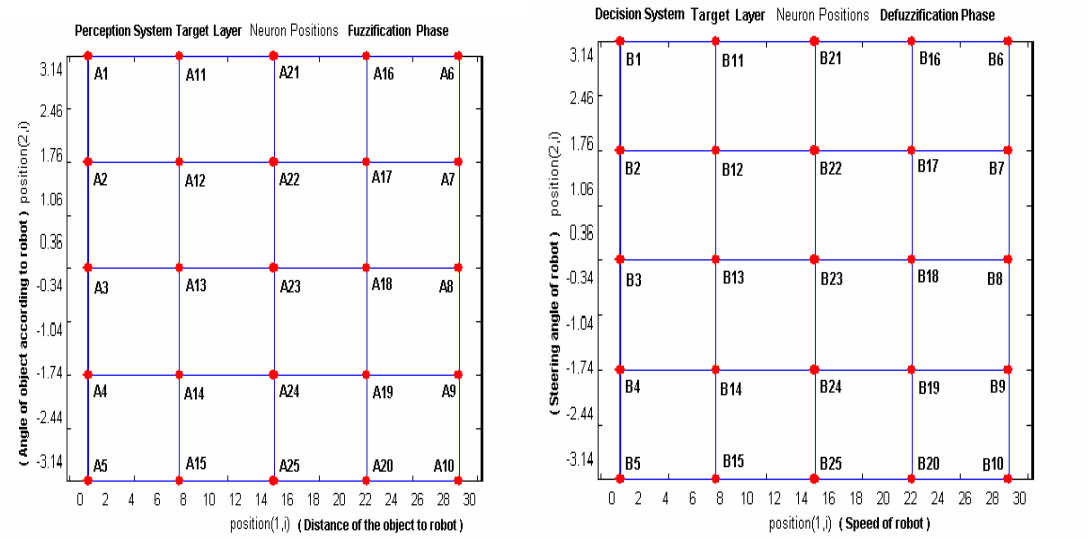

Figure 4.21: Ordered Neural Topologies For Perception and Decision System

In the above figures, it is shown components of the instinctual module. This units represent target layers of behaviors which give order to the behavior producing module for converge to ideal solutions (Figure 4.21).

4.2.1.1 Obstacle avoid

The input of the behavior is the sensory information about the relative position of the obstacle, and the output is the steer angle and speed [11]. This behavior forms a vector that is pushing the robot away from the obstacle. It can be considered that the reverse of the move to goal behavior. The magnitude of the vector increases, as the robot gets closer to the obstacle [11].

Table 4.1: Rule table for obstacle avoidance

Rule table for obstacle avoidance	
1) if input is A1 then output is B8	**14) if input is A24 then output is B22**
2) if input is A2 then output is B9	**15) if input is A25 then output is B23**
3) if input is A3 then output is B6 or B10	**16) if input is A16 then output is B13**
4) if input is A4 then output is B7	**17) if input is A17 then output is B14**
5) if input is A5 then output is B8	**18) if input is A18 then output is B11 or B15**
6) if input is A11 then output is B18	**19) if input is A19 then output is B12**
7) if input is A12 then output is B19	**20) if input is A20 then output is B13**
8) if input is A13 then output is B16 or B20	**21) if input is A6 then output is B3**
9) if input is A14 then output is B17	**22) if input is A7 then output is B4**
10) if input is A15 then output is B18	**23) if input is A8 then output is B1 or B5**
11) if input is A21 then output is B23	**24) if input is A9 then output is B2**
12) if input is A22 then output is B24	**25) if input is A10 then output is B3**
13) if input is A23 then output is B21 or B25	

4.2.1.2 Move to goal

This behavior forms a vector that is pulling the robot to the goal [11]. The position and orientation of this vector is determined according to the relative position of the goal. Either the original position of the goal and the robot, or the relative position of the goal should be known in order to determine the vector position and orientation. The magnitude of this vector is determined according to the distance between the robot and the goal. The move to goal behavior is fuzzified by using the relative distance and the relative angle of the goal as input to the fuzzy engine [11]. There are two outputs of the fuzzy engine, first is the speed and the second is the steer. The input fuzzy relations consist of two components which are constituted by combining the distance and the angle of the goal for determining location of the goal.

Table 4.2: Rule table for Move to Goal

Rule table for Move to Goal	
1) if input is A1 then output is B1	14) if input is A24 then output is B24
2) if input is A2 then output is B2	15) if input is A25 then output is B25
3) if input is A3 then output is B3	16) if input is A16 then output is B16
4) if input is A4 then output is B4	17) if input is A17 then output is B17
5) if input is A5 then output is B5	18) if input is A18 then output is B18
6) if input is A11 then output is B11	19) if input is A19 then output is B19
7) if input is A12 then output is B12	20) if input is A20 then output is B20
8) if input is A13 then output is B13	21) if input is A6 then output is B6
9) if input is A14 then output is B14	22) if input is A7 then output is B7
10) if input is A15 then output is B15	23) if input is A8 then output is B8
11) if input is A21 then output is B21	24) if input is A9 then output is B9
12) if input is A22 then output is B22	25) if input is A10 then output is B10
13) if input is A23 then output is B23	

4.2.1.3 Head on

This behavior provides the robot to look forward while moving. The behavior tries to decrease the steer angle down to 0 by changing orientation θ [11].

Table 4.3: Rule table for Head on

Rule table for Head on	
1) if input is A1 then output is B1	**14) if input is A24 then output is B24**
2) if input is A2 then output is B2	**15) if input is A25 then output is B25**
3) if input is A3 then output is B3	**16) if input is A16 then output is B1**
4) if input is A4 then output is B4	**17) if input is A17 then output is B2**
5) if input is A5 then output is B5	**18) if input is A18 then output is B3**
6) if input is A11 then output is B1	**19) if input is A19 then output is B4**
7) if input is A12 then output is B2	**20) if input is A20 then output is B5**
8) if input is A13 then output is B3	**21) if input is A6 then output is B1**
9) if input is A14 then output is B4	**22) if input is A7 then output is B2**
10) if input is A15 then output is B5	**23) if input is A8 then output is B3**
11) if input is A21 then output is B1	**24) if input is A9 then output is B4**
12) if input is A22 then output is B2	**25) if input is A10 then output is B5**
13) if input is A23 then output is B23	

4.2.1.4 Following trajectory (i.e. wall)

Generally this behavior is responsible from that the robot tracks a moving object or route. The behavior is a modified version of Move to Goal Behavior. This time, the Goal is moving and the goal attractor vector changes during the voyage [11].

By using this behavior, if the robot goes close to the walls, it starts to follow the wall by trying to keep the distance constant, and turn to the empty side if it faces a wall in front [11].

4.2.1.5 Wander / search

When this behavior is activated, the resulting fuzzy output relation is assigned as randomly whichever the input fuzzy relation is selected. Thus robot realizes to moving wander behavior as unconsiciously. This behavior can be useful provided that goal or interested items do not appear at the point of view of the robot. In this case search behavior try to find objects as moving wander [11].

Table 4.4: Rule table for Wander / Search

Rule table for Wander / Search	
1) if input is A1 then output is R	**14) if input is A24 then output is R**
2) if input is A2 then output is R	**15) if input is A25 then output is R**
3) if input is A3 then output is R	**16) if input is A16 then output is R**
4) if input is A4 then output is R	**17) if input is A17 then output is R**
5) if input is A5 then output is R	**18) if input is A18 then output is R**
6) if input is A11 then output is R	**19) if input is A19 then output is R**
7) if input is A12 then output is R	**20) if input is A20 then output is R**
8) if input is A13 then output is R	**21) if input is A6 then output is R**
9) if input is A14 then output is R	**22) if input is A7 then output is R**
10) if input is A15 then output is R	**23) if input is A8 then output is R**
11) if input is A21 then output is R	**24) if input is A9 then output is R**
12) if input is A22 then output is R	**25) if input is A10 then output is R**
13) if input is A23 then output is R	

Where R is represented as a random label value of the B class. Thus whenever input liguistic label come, output linguistic label take a random value.

4.2.2 Hidden Markov Models

HMM's are used in a wide field of signal processing, particularly in speech processing, communication systems, and biological signal processing applications as well as estimation and control. A HMM can be considered a partially observed stochastic dynamical system [46].

$$x_{k+1} = A.x_k + v_{k+1} \quad \textbf{(4.10)}$$

$$y_{k+1} = C.x_k + w_{k+1} \qquad k \in \mathrm{N}, \quad \textbf{(4.11)}$$

on the probability space (Ω, F, *P)* with set of measurable points Ω, complete filtration {Fk}, and probability measure *P* in discrete time. The system state is represented by the process x, while y represents the observable part of the system. The random numbers v and w are considered independed with identical distribution representing driving and measurement noise, respectively [46]. In this thesis we can considler the state space formed by sets of unit vectors *ei* and fj, so that (4.10) and (4.11) simplify to

$$X_{k+1} = A.X_k + V_{k+1} \quad \textbf{(4.12)}$$

$$Y_{k+1} = C.X_k + W_{k+1} \qquad k \in \mathrm{N}, \quad \textbf{(4.13)}$$

Where

$$X_k \in S_X = \{e_1, e_2, ..., e_N\} \quad \textbf{(4.14)}$$

$$Y_k \in S_Y = \{f_1, f_2, ..., f_N\} \quad \textbf{(4.15)}$$

with unity on indices i and. J of ei and fj, respectively,

$$A \in \mathbb{R}_0^{+N \times N}, \qquad C \in \mathbb{R}_0^{+M \times N},$$

and V, *W* are martingale increment processes. Further, *X* satisfies the Markov property, which implies the transition probability e.g. by linear combination [46].

$$P(X_{k+1} = e_j \mid F_k) = P(X_{k+1} = e_j \mid X_k), \quad \textbf{(4.16)}$$

so that *X* is a Markov process. The elements *aij, cij* of matrices *A* and *C* represent transition probabilities for a change of state and observable output generation,

respectively, according to

$$a_{ij} = P(X_{k+1} = e_j \mid X_k = e_j) \quad \textbf{(4.17)}$$

$$c_{ij} = P(Y_{k+1} = f_j \mid X_k = e_j), \quad \textbf{(4.18)}$$

whereby the column sums of *aij,* cij satisfy

$$\sum_{i=1}^{N} a_{ij} = \sum_{i=1}^{M} c_{ij} = 1. \quad \textbf{(4.19)}$$

Process *Xk,* corresponds to a digraph topology G consisting of vertices qi and edges or transitions t_{ij} as depicted (Figure 4-22). The states *Xk* of state space Sx correspond to vertices, while a_{ij} correspond to edges [46].

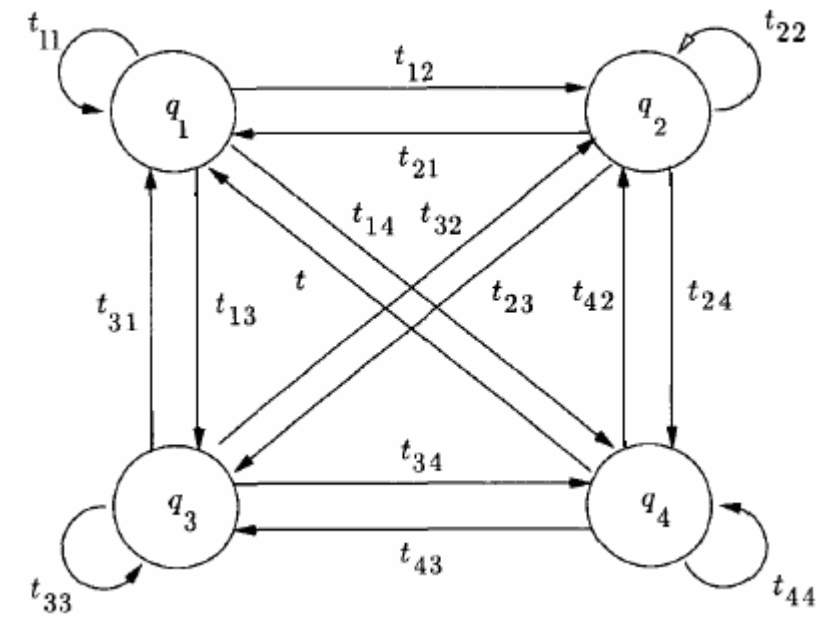

Figure 4.22: Graph of four state hidden Markov model [46].

Where Matrix *A* then represents the weighted adjacency matrix of G [46]. The HMM is augmented by a control input $u \in \{0, 1\}^L$ having an impact on the state transitions according to

$$X_{k+1} = A.X_k + B.u_{k+1} + V_{k+1} \quad \textbf{(4.20)}$$

where *B* represents the weighted incidence matrix of G or

$$X_{k+1} = A'(u_{k+1}).X_k + V_{k+1} \quad \textbf{(4.21)}$$

e.g. by linear combination

$$A'(u_k + 1) = \frac{1}{1 + K}\left[A + \begin{pmatrix} B^T u_{k+1}^T \\ B^T u_{k+1}^T \\ \vdots \\ B^T u_{k+1}^T \end{pmatrix} \right], \quad \textbf{(4.22)}$$

Where

$B \in \mathbb{R}_0^{+N \times L}$,

K amounts to

$$K = \sum_{i=1}^{L} u_{i,k}, \quad \textbf{(4.23)}$$

and condition (4.17) has to be met for A‘ so that

$$\sum_{i=1}^{N} b_{ij} = 1. \quad \textbf{(4.24)}$$

To further details on HMM’s the reader may e.g. refer to [47]

4.2.2.1 HMM Terminology

A HMM Model is specified by:

- The set of states S = {s1, s2, . . . , sNs}. This paramater set corresponds to the four possible innate behaviors in the instinctual module of coordination level, and a set of parameters $\Theta = \{\pi, A, B\}$ [48] :
- The prior probabilities π_i = P(q1 = si) are the probabilities of si being the first state of a state sequence. Collected in a vector π. (The prior probabilities were assumed equi-probable in the last example, π_i = 1/Ns.) [48].
- The transition probabilities are the probabilities to go from state i to state j:

  $a_{i,j} = P(q_{n+1} = s_j \mid q_n = s_i)$. They are collected in the matrix A [48].

- The emission probabilities characterize the likelihood of a certain observation x, if the model is in state si. Depending on the kind of observation x we have:

- For discrete observations, xn $\in$ {v1, . . . , vK}: $b_{i,k}$ = P(xn = vk|qn = si), the probabilities to observe vk if the current state is qn = si. The numbers bi,k can be collected in a matrix B. This would be the case for the behavior transition model, with K = 2 possible observations v1 and v2 [48].

- For continuous valued observations, e.g., xn $\in$ R^D : A set of functions $b_i(x_n) = P(x_n \mid q_n = s_i)$ describing the probability densities (probability density functions, pdfs) over the observation space for the system being in state si. Collected in the vector B(x) of functions. Emission pdfs are often parametrized, e.g, by mixtures of Gaussians [48].

The operation of a HMM is characterized by

- The (hidden) state sequence Q = {q1, q2, . . . , qN}, qn $\in$ S, (the sequence of the weather conditions from simulation time step 1 to N).

- The observation sequence X = {x1, x2, . . . , xN}.

A HMM allowing for transitions from any emitting state to any other emitting state is called an ergodic HMM. The other extreme, a HMM where the transitions only go from one state to itself or to a unique follower is called a left-right HMM [48].

Probability of a state sequence: the probability of a state sequence Q = {q1, q2, ... , qN} coming from a HMM with parameters Θ corresponds to the product of the transition probabilities from one state to the following [48]:

$$P(Q \mid \Theta) = \pi_{q_1} . \prod_{n=1}^{N-1} a_{q_n, q_{n+1}} = \pi_{q_1} . a_{q_1, q_2} . a_{q_2, q_3} . \ldots\ldots . a_{q_{N-1}, q_N} \qquad \textbf{(4.25)}$$

Likelihood of an observation sequence given a state sequence, or likelihood of an observation sequence along a single path: given an observation sequence X = {x1, x2, . . . , xN} and a state sequence Q ={q1, q2, . . . , qN} (of the same length) determined from a HMM with parameters Θ, the likelihood of X along the path Q is equal to [48]:

$$P(X \mid Q, \Theta) = \prod_{n=1}^{N} P(x_n \mid q_n, \Theta) = b_{q_1, x_1} . b_{q_2, x_2} . \ldots\ldots . b_{q_N, x_N} \qquad \textbf{(4.26)}$$

4.2.3 Behavior Selection Module and Transition / Selection Policy

Behavior selection module can be considered as hidden markov model which is represented as finite state machine (Figure 4.23). This approach provides several advantages to us. In alives, behavior selection process or transition from one behavior to another behavior is based on probabilistic statistical modeling. Mobile robot behavior planning organization corresponds to this reason. The constitution of behavioral sequences or loops realize on the behavior producing module which include HMM.

The priorities of behaviors depend on state transition probabilities which constitude on the state transition probability matrix A. This idea supports that the behavior which has higher state transition probability than another one has a higher priority then lower one. According to coordination level learning process, the state topology of behavior selection module modify when new state is added or an available state is deleted.

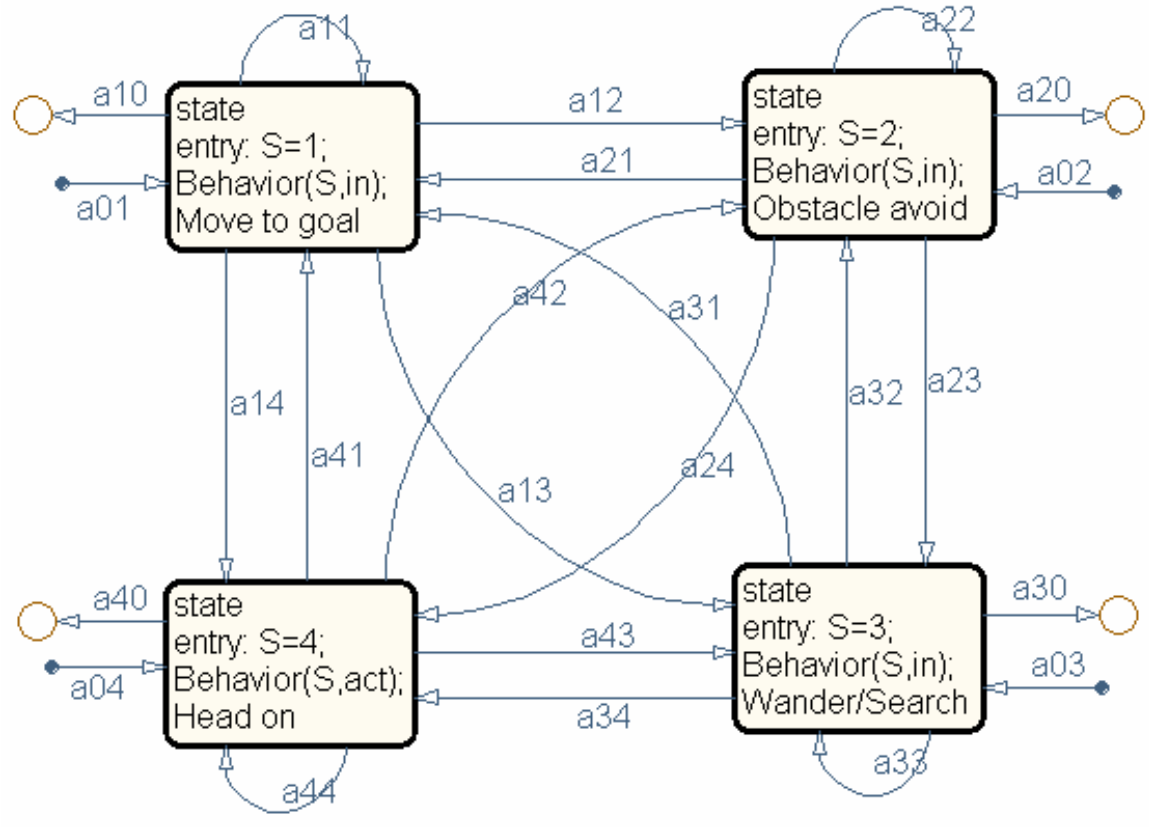

Figure 4.23: Behavioral Selection Module of Coordination Level Based on HMM

Behavior selection module perform below process according to the probability space $\Theta = (\Omega, F, P)$ on the proposed HMM (equation 4.27) [48].

$$X_{k+1} = A.X_k + V_{k+1} \quad \textbf{(4.27)}$$

$$Y_{k+1} = C.X_k + W_{k+1} \quad \textbf{(4.28)}$$

Ω = { move to goal, obstacle avoidance, head on, wander/search }

chosen according to model, where *Xk* represents the current behavioral state. Matrix *A* represents an behavioral state probability distribution satisfying (4.17), so that provided with a uniform random number a transition from state X_k, to X_{k+1} occurs with probability [48];

$$P(X_{i,k+1} \mid X_{j,k}) = a_{i,j} P(X_{j,k}) \quad \textbf{(4.29)}$$

The determination of the state transition probability matrix A directly affects the stochastic dynamics of hidden process *X ,* which can be considered behavioral state as inner emotion dynamics of the robot [48]. Different robot characters are modeled by tuning the elements of A. Stronger weighted main diagonal elements cause remaining in the current behavior state as inner emotional state rather than transition to a different state. Thus, a smoother course of *X* with longer constant behavioral phases is achieved [48]. Emphasizing one particular element of the main diagonal causes remaining more likely in one particular corresponding behavioral (ground) state of emotional model which is represented by state-space hidden markov model. Process *X* is initialized using a preferably uniform initial state distribution π. [48].

The behavior selection module of the coordination level selects behavioural states according to agents needs and goals which is influenced by motives module [48]. Thus when the behavioral state *Xk* is acted on behavior selection module of coordination level, emotional core model tries to reach corresponding emotional expression Y_{k+1} [48].

4.2.4 Cognitive System Module And Learning Methodology

4.2.4.1 Self-organized reinforcement learning

Reinforcement learning is the learning of a mapping from situations to actions so as to maximise a scalar reward or reinforcement signal [49]. A robot learns a given behaviour by being told how well or how badly it is performing as it acts in each given situation. As feedback, it receives a single information item from the world. By successive trials and/or errors, the robot determines a mapping function which is adapted through the learning phase [50]. For this purpose, numerous reinforcement

learning algorithms are available (e.g., TD(λ), AHC [51]), among which the Q-learning method is certainly the most used [50].

The Self-Organizing Map (SOM) was introduced by Teuvo Kohonen in 1982. The SOM (also known as the Kohonen feature map) algorithm is one of the best known artificial neural network algorithms. In contrast to many other neural networks using supervised learning, the SOM is based on unsupervised learning [50].

4.2.4.2 Q-learning algorithm

Reinforcement learning synthesises a mapping function between situations and actions by maximising a reinforcement signal [50]. Traditional Q-learning usually employs a table which stores the utility value of each possible action for every possible world state. In a real environment, the use of this table requires some form of partition of the continuous values provided by sensors. Q-learning provides the ability to learn, by determining which behavioural actions are most apropriate for a given stuation, the correct global response p for a given set of stimuli S presented by the world [50]. An update rule is used for the utility function Q(x,a), where x represents the states ans a the resulting actions. Q-learning [52] algorithms store the expected reinforcement value associated with each situation-action pair, usually in a look-up table. Three different functions (Figure 4-24) are involved: memorisation, exploration and updating [53]. Table 4-5 describes the Q-learning algorithm.

Table 4.5: Q-Learning Algorithm Procedure [50]

1. Initialisation of the robot memory: for all situation-action pairs, the associated Q value is 0 (i.e., $Q(i, a) = 0$).
2. Repeat :
   a - Let i be a world situation.
   b - The evaluation function select the action a to performed:
   $a = \text{argMax}(a)(Q(i, a'))$
   where a' represent any possible action in situation i.
   The exploration process modify the selected action to explore the situation-action space:
   $a^* = a + \Delta a$
   Δa is usually a randomly selected with a Gaussian distribution $N(0, \sigma)$, σ usually decreases as the learning proceed.
   c - The robot executes the action a in the world. Let r be the reward (r can be null) associated with the execution of the action a in the world.
   d - Update the robot memory:
   $Q_{t+1}(i, a) = Q_t(i, a) + ß(r + \gamma . \text{Max}(Q_t(i', a')) - Q_t(i, a))$. eq. 1
   where i' is the new situation after having carried out the action a in situation i,
   a'' is any action which is possible from state i' and $0 < ß, \gamma < 1$.

In response to the present situation, an action is proposed by the robot memory [50]. This action is the one that has the best probability of reward. Reward actions are also propagated across states so that rewards from similar states can facilitate their learning as well. However, this proposition may be modified by the evaluation function to allow an extensive exploration of the situation-action space. After the execution of the action by the robot in the real world, a reinforcement function provides a reinforcement value [50]. This value, here a simple qualitative criterion (+1, -1 or 0), is used by the updating algorithm to adjust the reward value (*Q*) associated with the situation action pair. The issue here is what constitutes a similar state. One approach uses the weighted Hamming distance as the basis for the similarity metric. The learning is incremental, because the acquisition of the examples is carried out in real situations [50].

4.2.4.3 Neural (SOM) Q-learning

An alternative to this method suggested by Lin (1993) is to use neural networks to learn by back-propagation the utility values of each action [50]. This method has the advantage of profiting from generalization over the input space and being more resistant to noise, but on the other hand the on-line training of neural-networks may not be very accurate [50]. The reason being that the neural networks have a tendency to be overwhelmed by the large quantity of consecutive similar training data and forget the rare relevant experiences. Using an asynchronous triggering mechanism, such as the one proposed by the current architecture, can help with this problem by detecting and using only a few relevant examples for training [50].

The SOM is quite a unique kind of neural network in the sense that it constructs a topology preserving mapping from the high-dimensional space onto map units in such a way that relative distances between data points are preserved [50]. The map units, or neurons, usually form a two-dimensional regular lattice where the location of a map unit carries semantic information. The SOM can thus serve as a clustering tool of high-dimensional data [50]. Because of its typical two-dimensional shape, it is also easy to visualize. Another important feature of the SOM is its capability to generalize. In other words, it can interpolate between previously encountered inputs [50].

During the learning phase, the neurones of the self-organising map approximate the probability density function of the inputs [50]. The inputs are situation, action and the associated Q value (Figure 4.24). The learning phase associates with each neurone of the map a situation-action pair plus its Q-value. It is a method of state grouping

involving syntactic similarity and locality [54]. The number of neurones equals the number of stored associations [50]. The neighbourhood property of the Kohonen map allows it to generalise across similar situation-action pairs. One of the most interesting aspects of SOMs is that they learn to classify data without supervision. With this approach an input vector is presented to the network (typically a multilayer feedforward network) and the output is compared with the target vector [50]. This is repeated many times and with many sets of vector pairs until the network gives the desired output. Training a SOM however, requires no target vector. A SOM learns to classify the training data without any external supervision whatsoever [50].

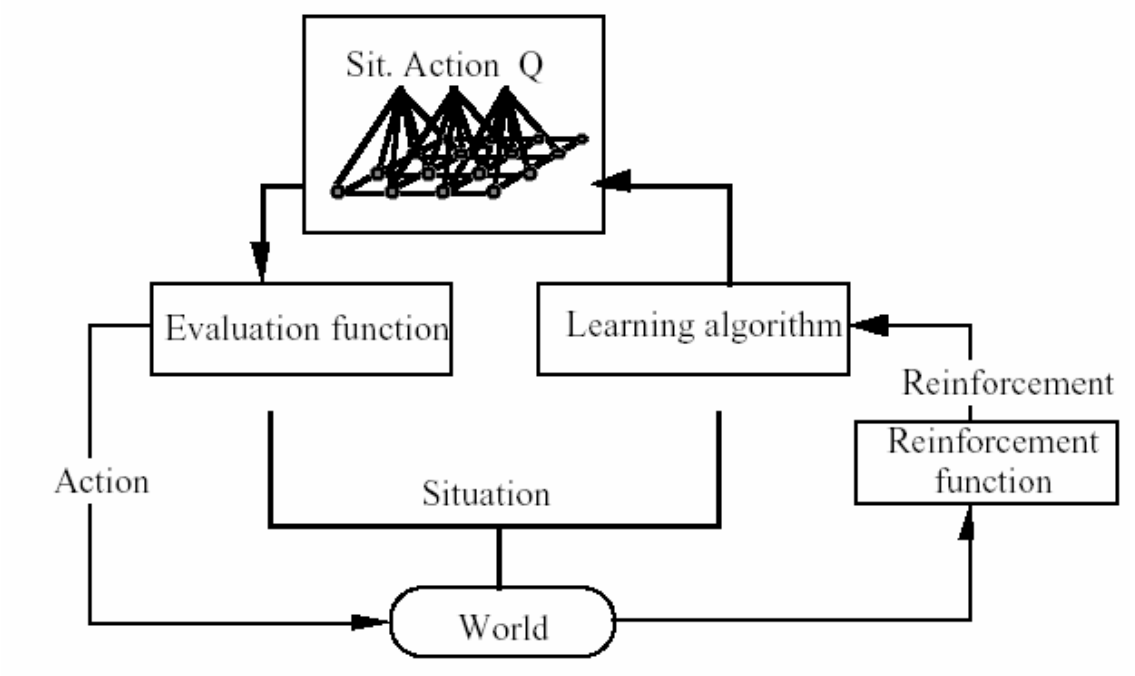

Figure 4.24 : The self-organising map implementation of Q-learning [50].

The Q-KOHON uses the self-organising map as an associative memory. This associative memory stored triplets [50]. Part of a triplet is used to probe the self-organising map in search of the corresponding information. Here, situation and *Q*value are used to find the action: the best action to undertake in a world situation is given by the neurone that has the minimal distance to the input situation and to a *Q* value of value +1 (Figure 4.25 a). To this end, equation 2 in the self-organising map learning algorithm (Table 4.6) has been changed to [50]:

$$d(i) = |world\_situation - W_{situation,\,i}| + |1 - W_{Qvalue,i}| \quad \textbf{(4.30)}$$

The selected neurone corresponds to a triplet (situation, action, *Q* value). It is this particular action that should offer the best reward in the world situation (Figure 4.25 b) [50].

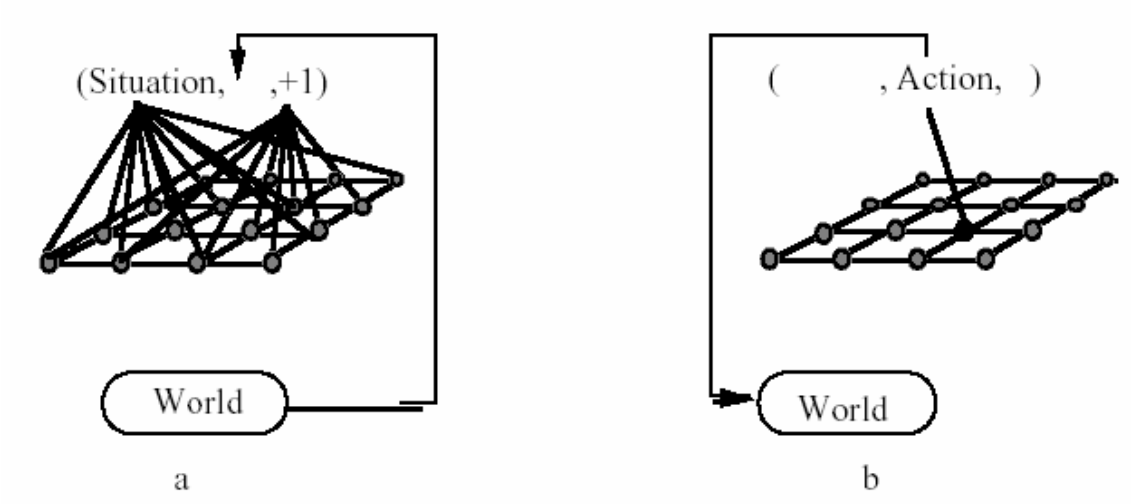

Figure 4.25 : Selection of the best action to perform in the world situation [50].

The Kohonen map is used as an associative memory: information is probed with part of it. a/ The world situation and a *Q* value of +1 are given as inputs [50]. The answer is a selected neurone which weights give situation, *Q* value and the associated action [50].

The learning base is built incrementally (as for the competitive multilayer implementation) [50]. Each action of the mobile robot is a learning example: the number of learning iterations is the total number of experiments (actions carried out in real world situations) [50].

Coding on a self-organising map is localised [50]. Each neurone represents a particular class (or cluster) of the inputs (Table 4.6). Competition occurs between all the neurones of the map [50]. During the learning phase, the neurones of the self-organising map approximate the probability density function of the inputs. The learning algorithm is presented in figure 4.25 [50].

Table 4.6 : Self-organising map learning algorithm [50]

Repeat

For each example E of the learning base, E = {x,y}

For each neurone i of the SOM,

$$d(i) = |x - w_{x,i}| + |y - w_{y,i}| \quad \text{(equation 2)}$$

Select the closest neurone: j = argMin (i) (d(i))

Modify the weights of j using a learning coefficient $0<\lambda<1$:

$$w_{j,x}(t+1) = w_{j,x}(t) + \lambda.(x - w_{j,x}(t)),$$

$$w_{j,y}(t+1) = w_{j,y}(t) + \lambda.(y - w_{j,y}(t))$$

Modify the weights of the neighbours of j with a learning coefficient μ using the same rule (usually $0<\mu<\lambda$)

The learning algorithm updates the *Q* value weight (using equation 1) and, also, the situation and action weights [50]. The neurone corresponding to the situation and the action effectively performed is selected. The distance used is different from the search process of the most rewarding action. It includes the situation and action vectors, but nothing concerning the *Q* value. The equation 2 is modified in the following manner [50]:

$$d(i) = |world\_situation - W_{situation,i}| + |1 - W_{action,i}| \quad \textbf{(4.31)}$$

Together with the selected neurone, the four neighbours are also updated. The learning coefficient is 0.9 for the selected neurone and 0.5 for the neighbourhood [50]. In conclusion, we note that, unlike the multilayer perceptron implementation, the interpretation of the weights is possible. Moreover, if a correct behaviour is learned (i.e., only positive or null reinforcement values are experienced), then all neurones will code positive *Q* values. This last fact results in the optimisation of the stored knowledge [50].

4.2.4.4 Mapping precision and topology preservation

The mapping precision measure describes how accurately the neurons 'respond' to the given data set [44]. For example, if the reference vector of the BMU calculated for a given testing vector xi is exactly the same xi, the error in precision is then 0. Normally, the number of data vectors exceeds the number of neurons and the precision error is thus always different from 0 [44].

A common measure that calculates the precision of the mapping is the average quantization error over the entire data set [44]:

$$E_q = \frac{1}{N}\sum_{i=1}^{N} \| x_i + m_c \| \quad \textbf{(4.32)}$$

The topology preservation measure describes how well the SOM preserves the topology of the studied data set [44]. Unlike the mapping precision measure, it considers the structure of the map. For a strangely twisted map, the topographic error is big even if the mapping precision error is small [44].

4.3 Artificial Emotion And Motivation Level

The presence of a stimulus is necessary but not enough to evoke a motor response in a behaviour based robot [4]. The level of the stimulus is a reference for the level of the reaction. There may be threshold level or a continuous path. The definition of the behaviour determines this path [10]. The reaction level can be controlled with behaviour gains. Behaviour gains are represented to emotional and motivational variables. The behaviour may be shut down with not enough behaviour gain (under level of motivational threshold) or the behaviour may be very poor [11]. According to level of behaviour gain (motivation) the force vector magnitude may be increased or decreased. The emotion and motivational level monitors the goals and the overall states of the agent [10].

4.3.1 Motivational Module

When operating in unpredictable and partially observable environments, an autonomous agent must examine the evolution of its general states, and try to capture what emerges from the interaction dynamics with its environment [10]. Works on motivational systems (Maes, 1991; Blumberg, Todd & Maes, 1996; Breazeal, 1998) have shown that a good balance between planning and reactivity for goal management can be achieved using internal variables that get activated or inhibited by different factors are concepts that are gaining importance in the design of autonomous agents [10].

The term ‘motive’ refers to something that prompts an agent to act in a certain way. Similarly, the Motives module is used to represent the behavioral gain coefficient of performing multi-goal robot tasks, making it decide how to behave in the world [10]. Motives can be influenced by the environment (from sensed conditions), the intentions of the robot (derived from behavioural recommendations), knowledge about the world (managed by the cognition module which is called as learning module of the coordination level) and by observing the effective use of the behaviour-producing modules. Motives are used to influence the recommendations of behaviour-producing modules and can also characterise special states in the cognition module (for example, motives experienced at particular locations can be memorised for future reference) [10]. Even though different representations can be used to implement the Motives module, we have been experimenting with one that tries to integrate temporal reasoning with a symbolic representation [10]. Each motive is associated with a particular goal (that can be accomplished using one or more behaviours) of the agent. A motive *m* is characterised by an energy level E

which is represented as probabilities of state transitions and a mapping function M that are used to determine its activation level as the behavioral gain coefficient [10].

The energy level and the activation level of a motive range between 0 and 100% [10]. The energy level can be influenced by various factors: sensory inputs, exploitation of behaviour-producing modules associated with the motive, activation of other motives, cognitive influences, emotions, and time. The energy level is computed by adding the influence of *n* factors affecting the motive, weighted by *w* [10].

Finally, mapping from E (motivational energy) to A (behavior activation gain) can be determined according to different methods [10]: directly transposed (*E*=*A*); determined using thresholds (*if E* > threshold then *A* = *x, …*); or triggered according to the energy level of other motives. The activation level of a motive can then influence other motives or be used by the other modules of the architecture [10].

4.3.2 Emotions Module

The concept of artificial emotion is increasingly used in designing autonomous robotic agents (Velásquez, 1998; Breazeal, 1998) [10]. In fact, psychological evidences suggest that emotion can serve three important roles (Michaud, Pirjanian, Audet & Létourneau, 2000). Emotions which can be classified as longer term cognitive activations can include different behavioral task sequences at different length [10].

- Emotion to adapt to limitations.
- Emotion for managing social behaviour. (hierarchy (*Anger/Fear*), territoriality (*Exploration/Surprise*), identity (*Acceptance/Rejection*) and temporality (*Joy /Sadness*).
- Emotion for interpersonal communication. In order for emotions to regulate behaviour in social interaction, emotion also has a communicative role.
- There are two main tasks of the emotional-motivational system [10]. Emotional system can obtain behavior selection sequence. According to this, behavioral transitions occur. Also motivational system can reinforce executing behavior. According to threshold level, some of behaviors may be interrupted or activated [10].

Artificial emotions are used to monitor how goals get satisfied or not [10]. Monitoring the energy level of motives makes the approach generic, since the emotions can be influenced by (an emerge from) different contexts (i.e., goals) according to the motives activated and their priority, attributed following the guidelines of Maslow's

Hierarchy of Needs Theory (Maslow, 1954) [10]. The emotional capability in the emotion and cognition based architecture which is inspired from the EMIB computational architecture is incorporated as a global background state, allowing emotions to influence and to be influenced by all of the architecture's modules [10].

Behaviour exploration parameters provide important feedback about the interactions the agent is having in its operating environment, interactions that emerge from the computation done in the three abstraction levels of the architecture. In the architecture, such feedback can be used to influence motives or to derive knowledge about the world as the agent experiences it [10].

4.3.3 Core of the Emotional system included HMM

A motive is defined as a reason for action [46]. In the viewpoint of robotics, motivation is a series of behavioral action through which mobile robot is continually going acting on the mobile robot. Observation part of the state-space HMM define emotional expressions and their transitions. The predicted behavior states in the coordination level are applied into the this procedure. Then below equation compute and obtain emotional expression corresponding to related behavioral state.

$$Y_{k+1} = C.X_k + W_{k+1} \tag{4.33}$$

Where *Y k* represents a currently observable emotional expression. The basic structure of the emotion core is shown in Figure 4.26.

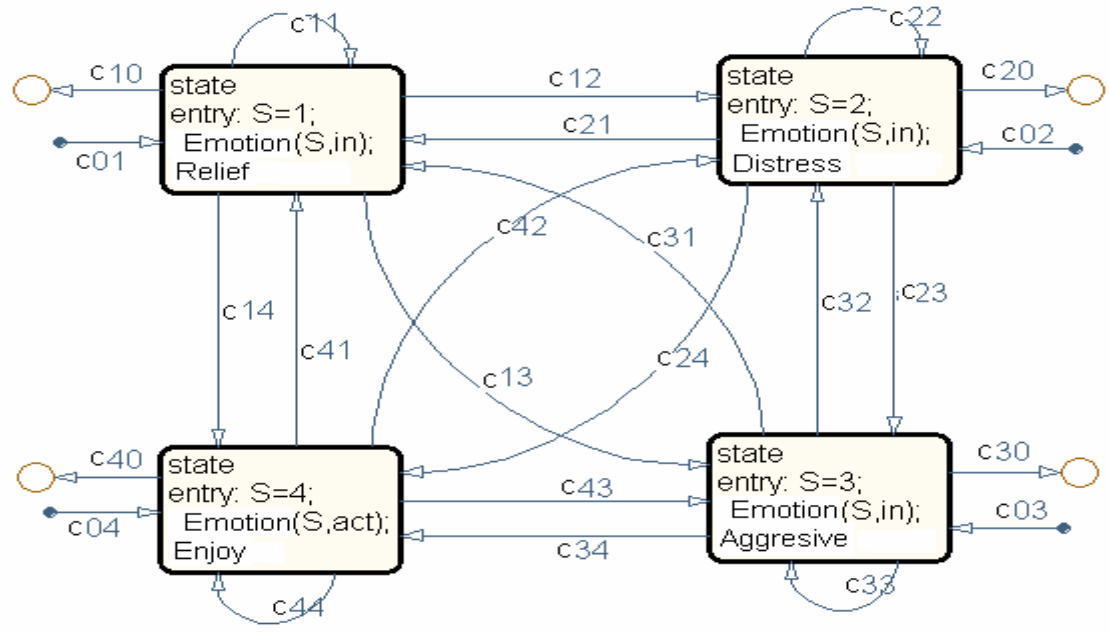

Figure 4-26 : Core of Emotional system

Given a current behavioral state *Xk* and provided with a second independently distributed uniform random number an emotional expression Y_{k+1} corresponding to a symbolic set [46]

SE = {Distress, Relief, Aggressive, Enjoy. . . }

is observable with probability

$$P(Y_{n,k+1} \mid X_{j,k}) = c_{n,j} P(X_{j,k}) \quad \textbf{(4.34)}$$

$$c_{n,j} = P(Y_{n,k+1} = f_n \mid X_{j,k} = e_j) \quad \textbf{(4.35)}$$

Matrix *C* of (4.36) represents an observation symbol probability distribution, where observable expressions are assigned to particular emotional states [46].

State-space representation is very useful model for core of the emotional system [46]. This Stocastic discrete model perform state transitions of between them. Artificial emotions are expressed in the observational part of the state space hidden Markov model. Also Core of the emotional system is at the case of interaction with behavior selection module of the coordination level. Behavior selection module include hidden state variables (behaviors) corresponding to observational part variables (emotional expressions) of the model [46].

5 Simulation and Results

5.1 Simulation Tools

All simulations of this thesis were realized in Matlab 7.0 and their products. Matlab and Simulink toolboxes are very useful for establishing mobile robot model, its emotion and cognition based general architecture and simulation environment.

5.1.1 Matlab 7.0 m-file

Program code of the emotion and cognition based general architecture was written as sub-routines in Matlab m-file. Each level of the general architecture use a Matlab function block and corresponding function m-file in simulink model.

5.1.2 Virtual Reality Toolbox

The Virtual Reality Toolbox is a solution for interacting with virtual reality models of dynamic systems over time [55]. It extends the capabilities of MATLAB and Simulink into the world of virtual reality graphics. The Virtual Reality Toolbox includes many features for you to create and visualize virtual reality models of dynamic systems [55].

The Virtual Reality Toolbox provides a flexible MATLAB interface to virtual reality worlds. After creating MATLAB objects and associating them with a virtual world, you can control the virtual world by using functions and methods [55].

5.1.3 State Flow Toolbox

Stateflow is a graphical design and development tool that works with Simulink [56]. Stateflow is a suitable environment for modeling logic used to control and supervise a physical plant modeled in Simulink. State flow toolbox visually models and simulates complex reactive control to provide clear, concise descriptions of complex system behavior using finite state machine theory, flow diagram notations, and state-transition diagrams all in the same diagram [56]. Stateflow brings system specification and design closer together. It is easy to create designs, consider various scenarios, and iterate until the Stateflow diagram models the desired behavior [56].

5.2 Visual Simulation Environment

Simulation environment and mobile robot were established on the Virtual Reality Toolbox. Visual observations of the mobile robot are realized by this Virtual Reality Toolbox application. Design of the 3D simulation environment include some objects such as obstacles, goal, walls, mobile robot. Simulation output informations are transfered to the sink block of the Virtual Reality toolbox. And simulation input informations are provided from the source block of the Virtual Reality toolbox. For example according to wheel angles, speeds of the wheels, displacement informations of the mobile robot and orientation informations of the mobile robot are transmitted to sink block. In the same way, location informations of the static and dynamic obstacles, location information of the green marker as target and Virtual Reality toolbox block sensor informations are connected to proposed system by the source block. So this tool is provide dynamic interaction between visual world and simulation blocks to us (Figure 5.1).

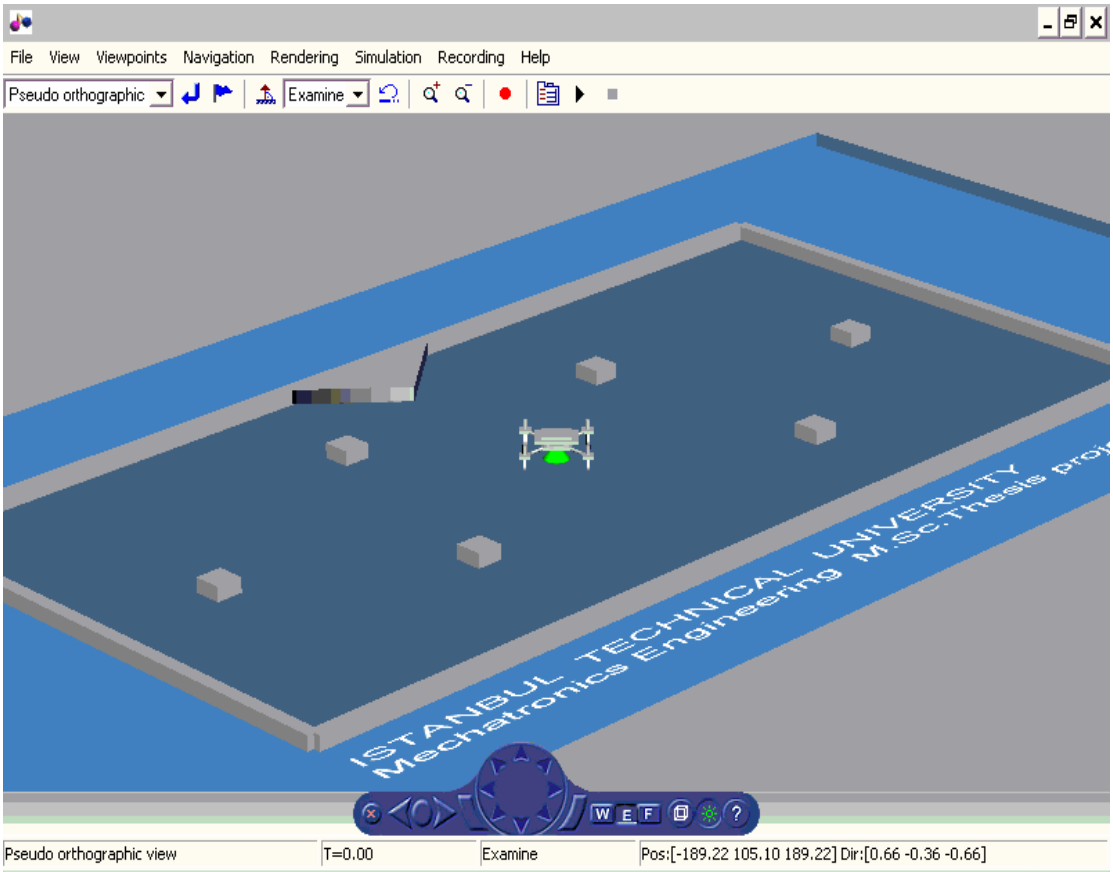

Figure 5.1 : Mobile robot simulation environment in Virtual Reality Viewer Window

5.3 Simulation Blocks And Architectural System Structure

General system architecture and model of the mobile robot are designed as blocks on the simulink. There are available many different type simulink tools in the general simulation window. Virtual Reality toolbox blocks need their conversion blocks for transformation simulation output informations to 3D Virtual Reality datas or transformation output 3D Virtual Reality datas as input to simulation output informations (Figure 5.2).

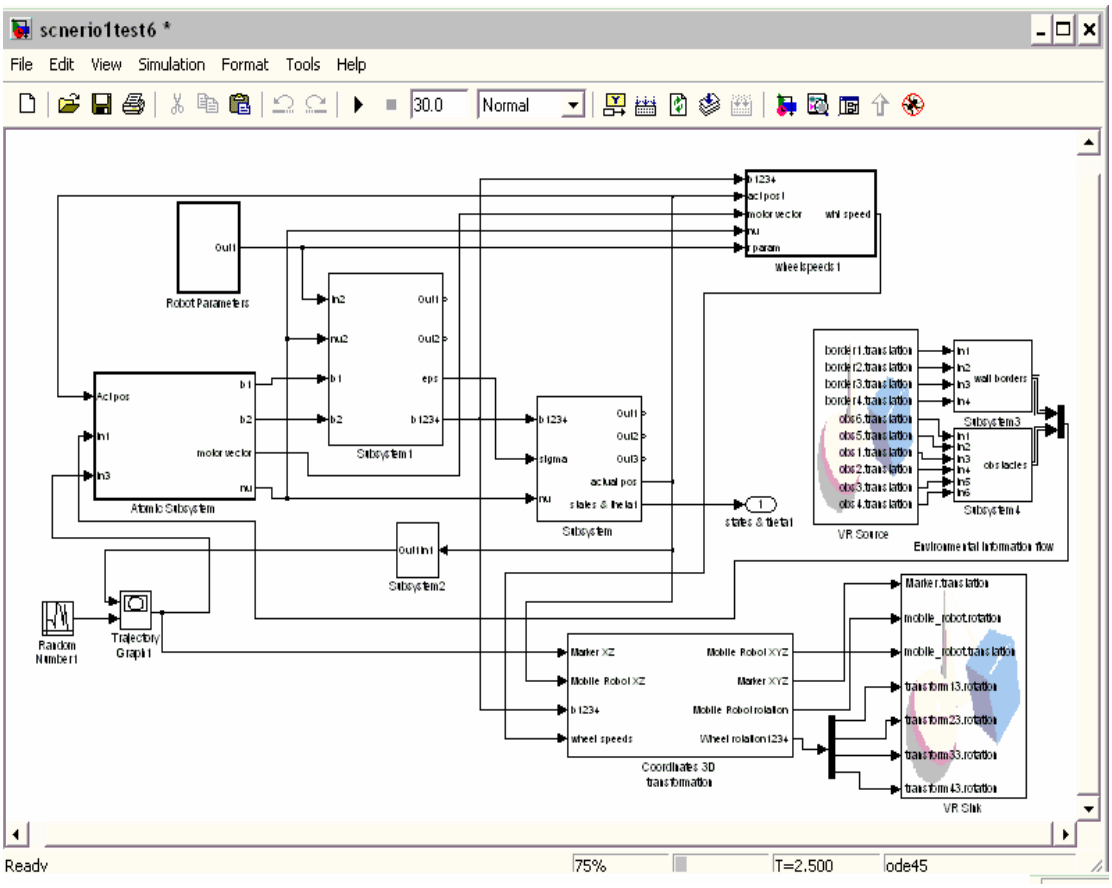

Figure 5.2 : Simulation Plant

As optional work, during the simulation can be dynamically given displacement ability to obstacles. But obstacles are considered as static object in this work. Trajectory graph block give displacement and orientation of the mobile robot as trajectory curve to us. Also trajectory graph transmitted target location information as sensor input to proposed system.

5.3.1 Artificial Cognitive System Simulink Block Structure

Proposed cognitive architecture of the mobile robot constitute three main levels. These are behavior producing module, coordination level and emotion-motivation level blocks. Also apart from several blocks are available. These are command fusion block, sensor data input module and environment block. Environment block prepare the datas of each items in the 3D environment as x-y pairs. According to actual position of the mobile robot, sensor data input module compute informations of eight ultra sonic sensors of the mobile robot. This informations contain distance of the an item to the mobile robot and angle difference between item and mobile robot.

Output datas of the sensor data input module enter to the behavior producing module for obtain output action information. Matlab function block and m-file which corresponds to behavior producing module are used for this module.

Also coordination level block is used for establishing relationship between emotion-motivation level and behavior producing module. This block is responsible from changing or switching behaviors, modify or tuning behaviors according to learning process. Emotion-motivation level block acts events of the coordination level. Another block is command fusion block. This block compute convenient motor datas such as wheel angles, speeds behavioral gaining procedure, difference between actual position and reference input.

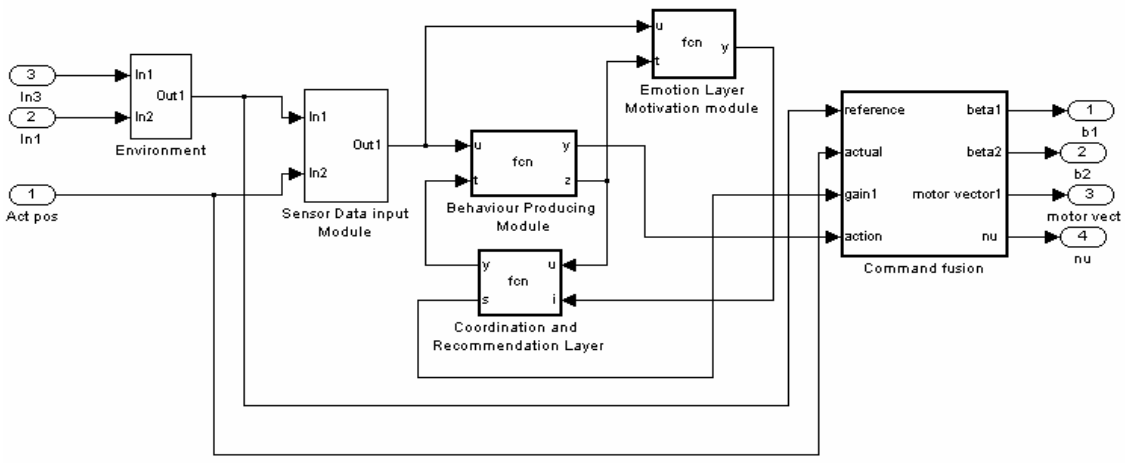

Figure 5.3 : Artificial Emotion and Cognition Based General Architecture

The gained steer and speed values of the behaviors are added in the Command Fusion Block (Figure 5.3). This addition is done separately on the speed, on the 1st wheel and on the 2nd wheel.

5.3.2 Robot Parameters

The parameters used in the simulation to define the robot are in Table 5-1

Table 5.l: Robot parameters [11]

Parameter	Yi(rad)	y_2(rad)	y_3(rad)	Y_4(rad)	/(m)	r_w(m)	M_b(kg)	Mw(kg)
Used value	0.69	2.44	-2.44	-0.69	0.21	0.2	25.	2

These parameters are entered in the "Robot Parameters" block in the main window [11]. By changing these variables, any 4x4x4 (4 wheel steered 4 wheel driven) mobile robot can be simulated. This parametric definition is an advantage of this simulation structure for further studies about this subject [11].

5.3.3 Kinematics Block

The Kinematics Block is calculating the actual posture of the mobile robot [11]. The initial posture is entered into this block by the GUI through the animation m-file, and the block calculates the actual posture using speed η and wheel angles β_1, β_2 (Figure 5.4) [11].

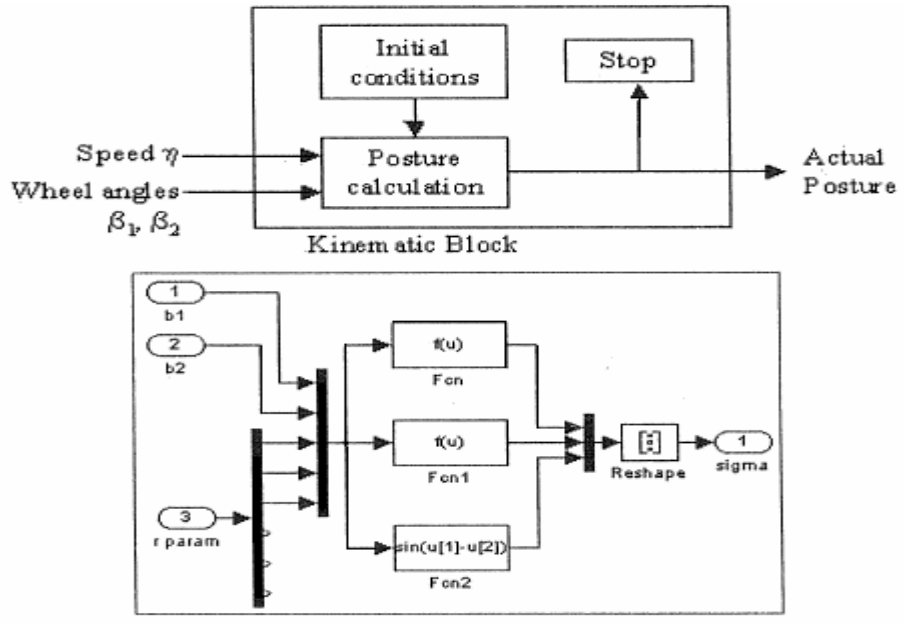

Figure 5.4a : Kinematics Block

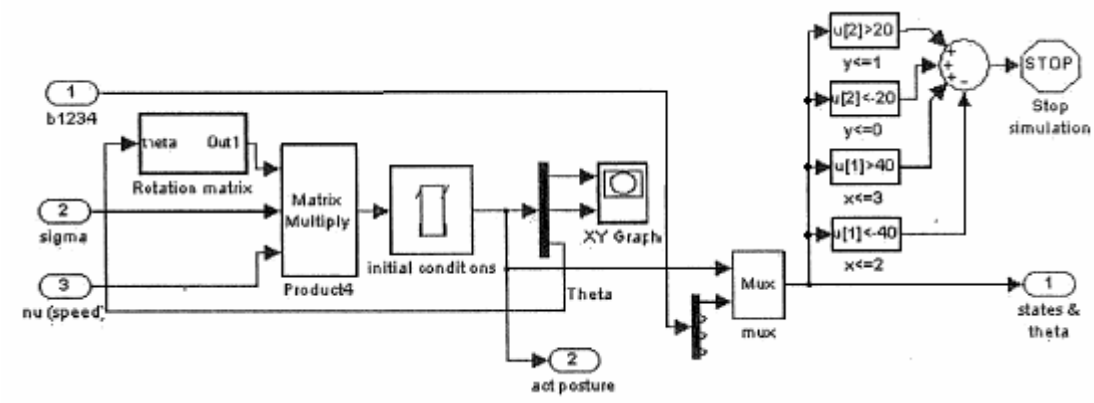

Figure 5.4b : Kinematics Block [11]

The kinematics block stops the simulation if the robot position coincides with the walls. This means that the robot hit the wall [11].

5.3.4 Torque Calculation

The torque for each wheel is calculated by using the posture information from the "Kinematics Block" and the "Robot Parameters" block [11]. Torque calculation block use standart simulink library blocks for this process (Figure 5.5). Also torque calculation procedure execute algebraic operations where was referred in the appendix A [11].

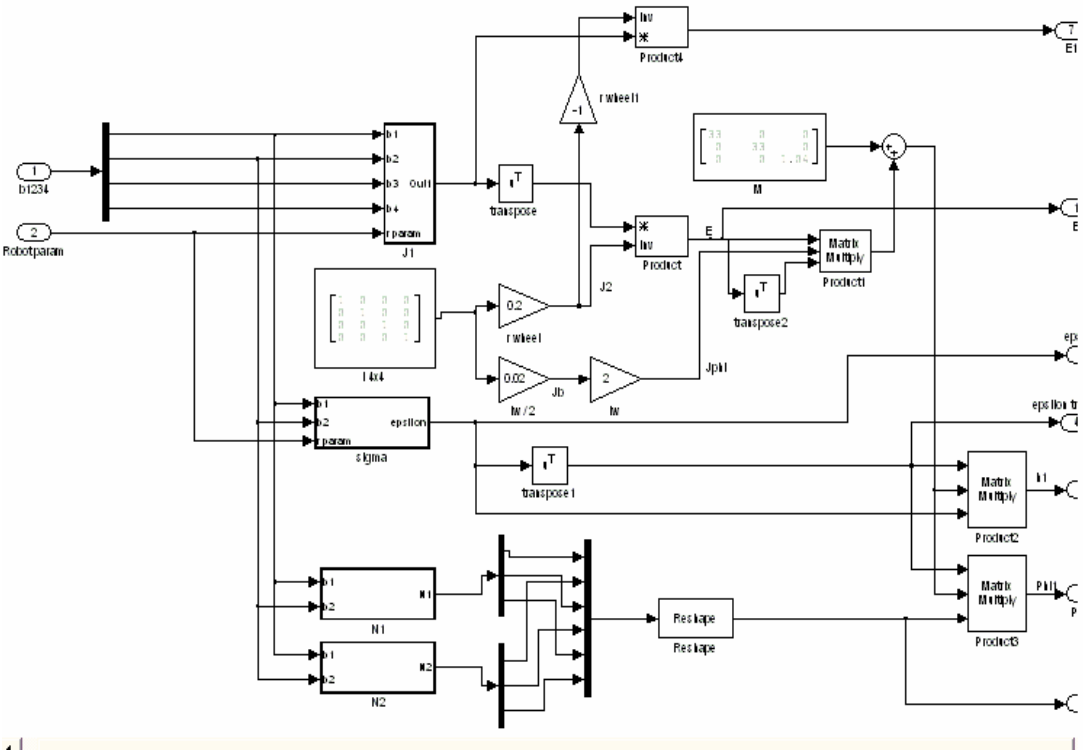

Figure 5.5 : Dynamic model of the mobile robot [11]

5.4 Simulation Results

Several case studies and their tests are realized according to proposed system architecture. Simulation results are presented as motor output data which belongs to related behavior.

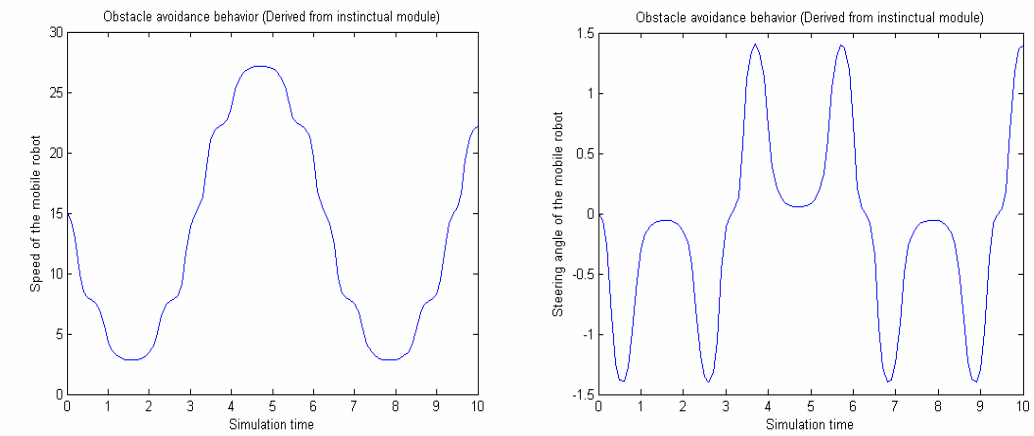

Figure 5.6 : Obstacle Avoidance behavior steering and speed of mobile robot

When the mobile robot come across with an obstacle, behavior transitions occur. Behavior producing module has generated so many behavior at very different characteristic. However in case of the switched current behavioral state is obstacle avoidance behavior by the behavior selection module of the coordination level, genetated behaviors have converged to obstacle avoidance behavior in the behavior producing module. In this behavior characteristic outputs such as speed and wheel steering angle are shown in figure 5.6. After move to goal behavior is activated and similar processes are realized for this new behavior (figure 5.7).

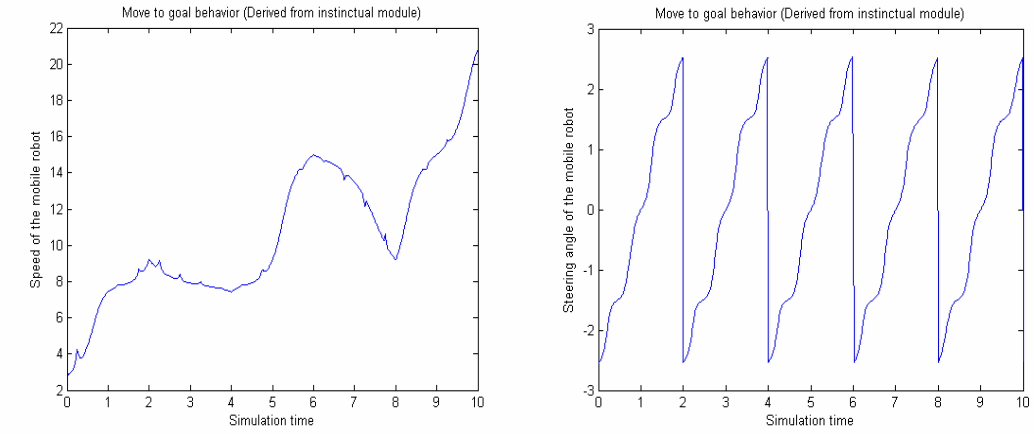

Figure 5.7 : Move to goal behavior steering and speed of mobile robot

5.4.1 Behavioral Selection Process Based on HMM

Behavioral state transitions in the coordination level can be observed at the each time step (Figure 5.8). Behavioral state variations as previous actions dynamically update by changing time. And actions are presented as next behavioral state. According to behavioral state sequences, resulting emotional transitions are observed (Figure 5.9). Matlab State Flow toolbox is very useful for performing this events. In the State Flow toolbox chart block, Hidden Markov model as finite state network was considered for behavioral selection and transition procedure of the coordination level. Thus menaging of discrete events and their sequences as stocastic process was provided to realize in this chart block.

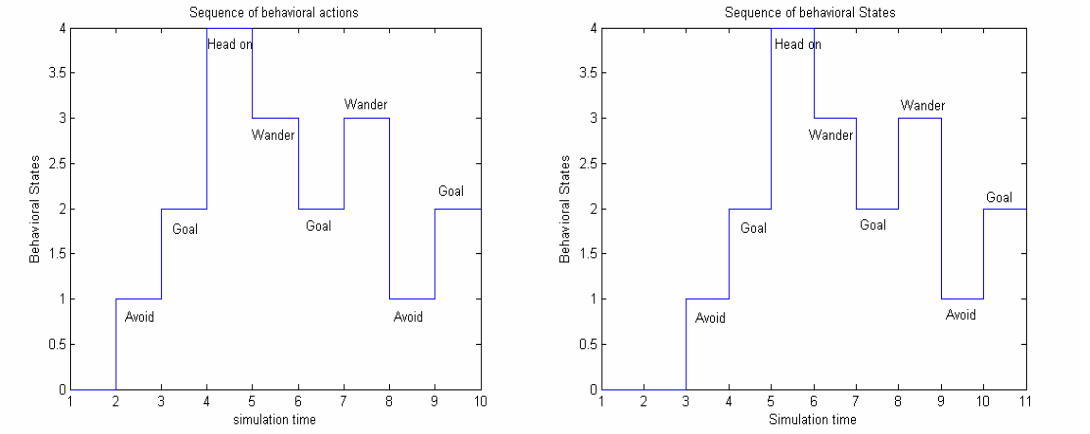

Figure 5.8 : Behavioral state (Previous action) and action sequences

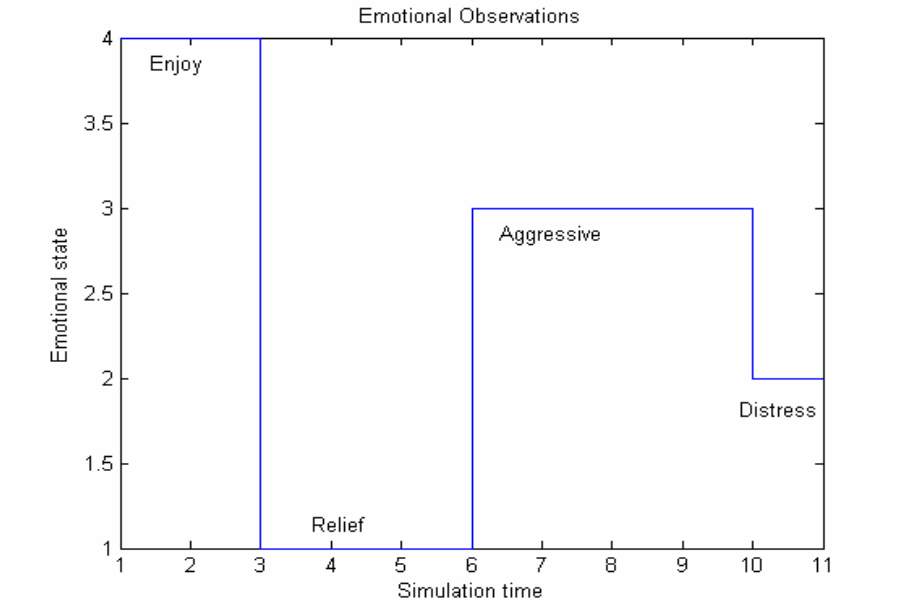

Figure 5.9 : Emotional transition sequences

5.4.2 Learning Process and Training Performances

Learning process of the SOM type neural network which belongs to behavior producing module try to adjust fuzzy logic control parameters such as rules, input and output fuzzy relational sets. Mean square error as training performance is struggled to minimize by neural network training algorithm (Figure 5.10).

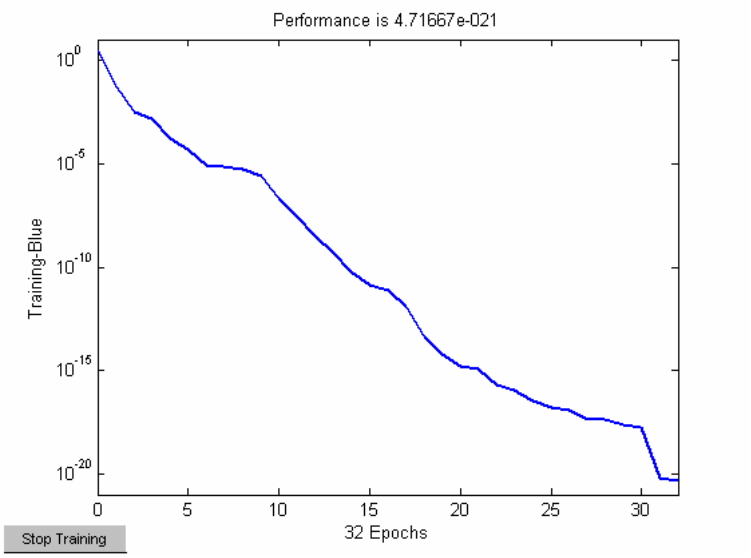

Figure 5.10 : Behavior producing module training performance

Coordination level for learning of the behavioral sequences use Q-learning algorithm based on SOM type neural network. State transition probability pattern of the Hidden Markov model as input is used in learning strategy of the coordination level. Training performance of the this learning process was presented in figure 5.11.

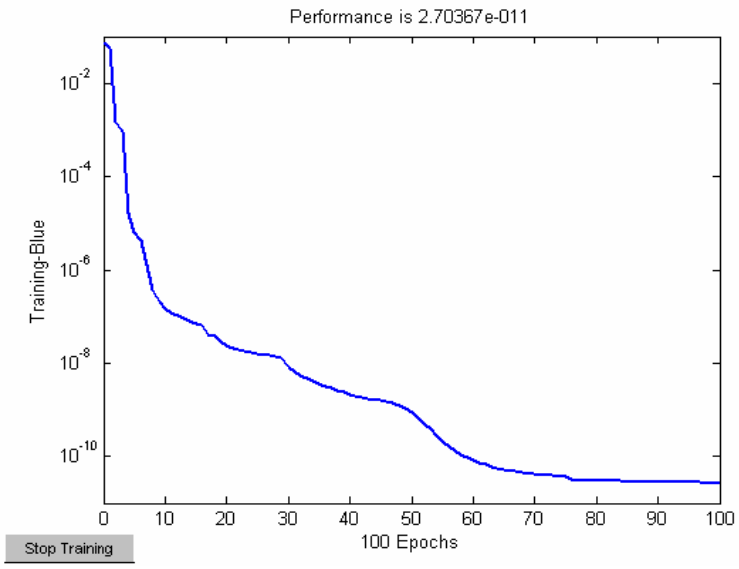

Figure 5.11 : Cognition module (Coordination level) Q-SOM network training performance

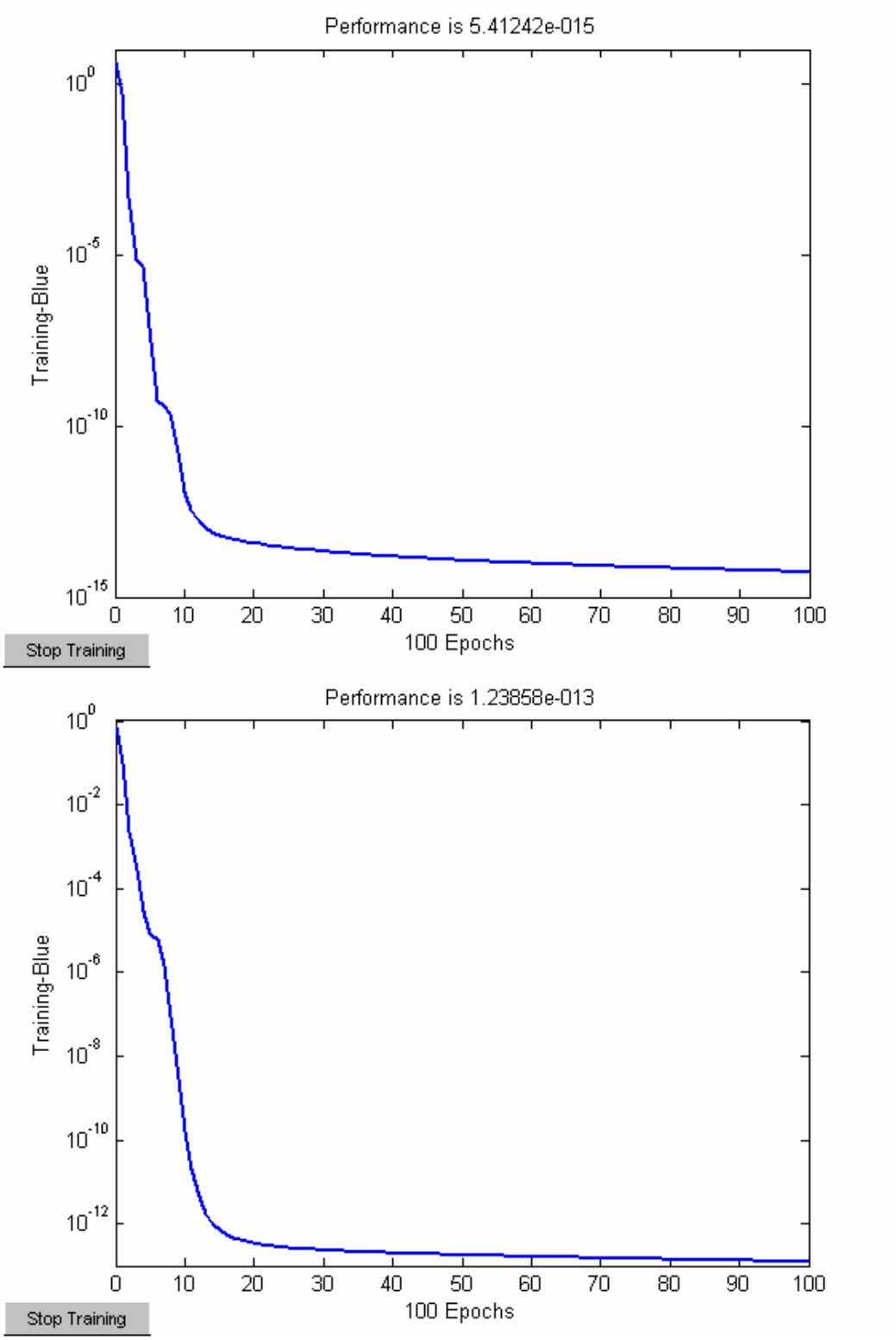

Figure 5.12 : Different type training samples

Several training results were presented in figures. Behavior producing module and learning module of the coordination level tried to improve adaptation performance according to training situations (Figure 5.12).

5.4.3 Genetic Algorithm Optimization Results

Behavior producing module use diploid chromosomal genetic programming for obtaining better network weights. Also behavioral parameters are encoded in diploid chromosomal genetic string. Simulation results can be shown in figure 5.13.

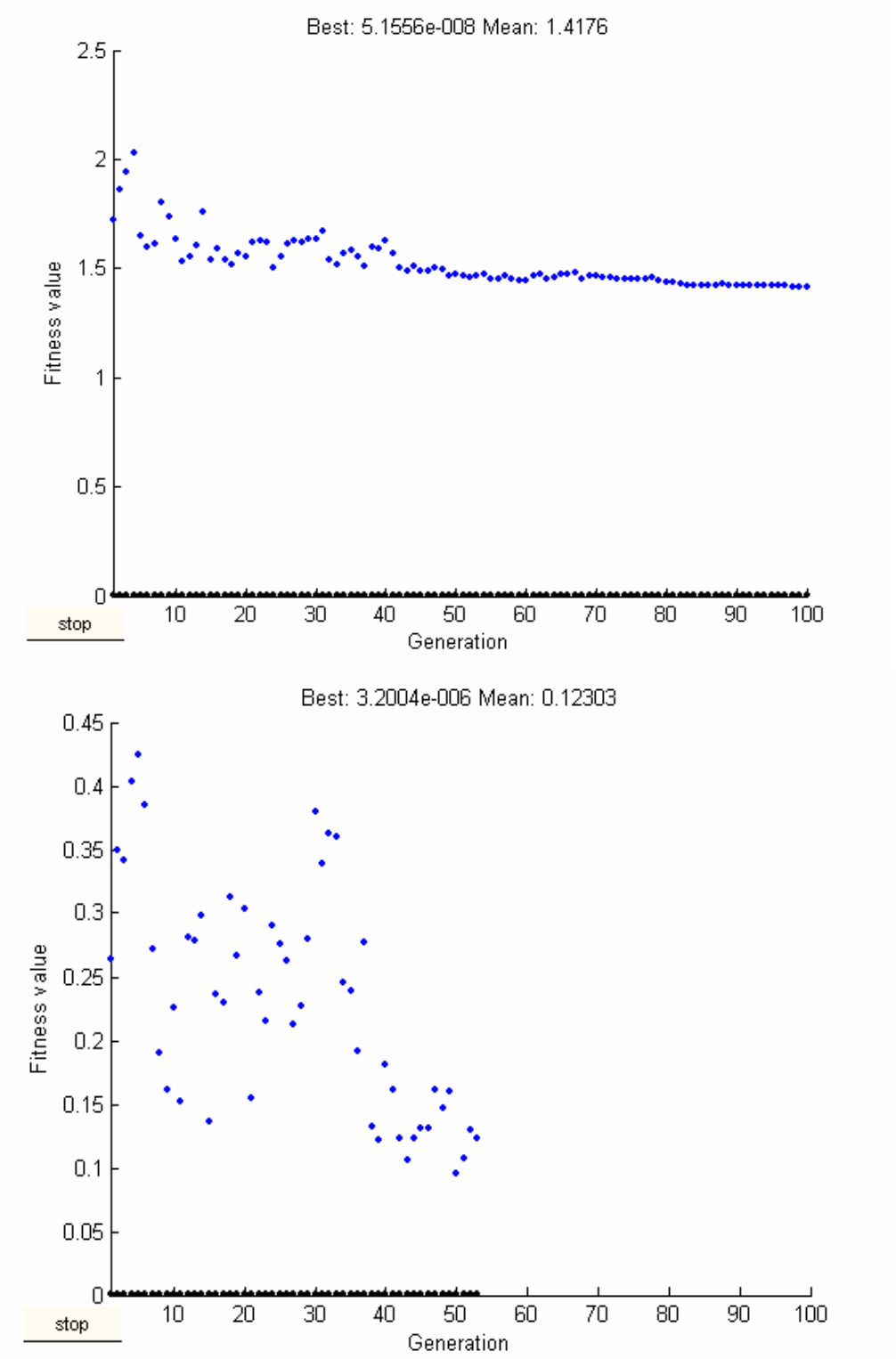

Figure 5.13 : Behavior producing module network weight optimisation with GA

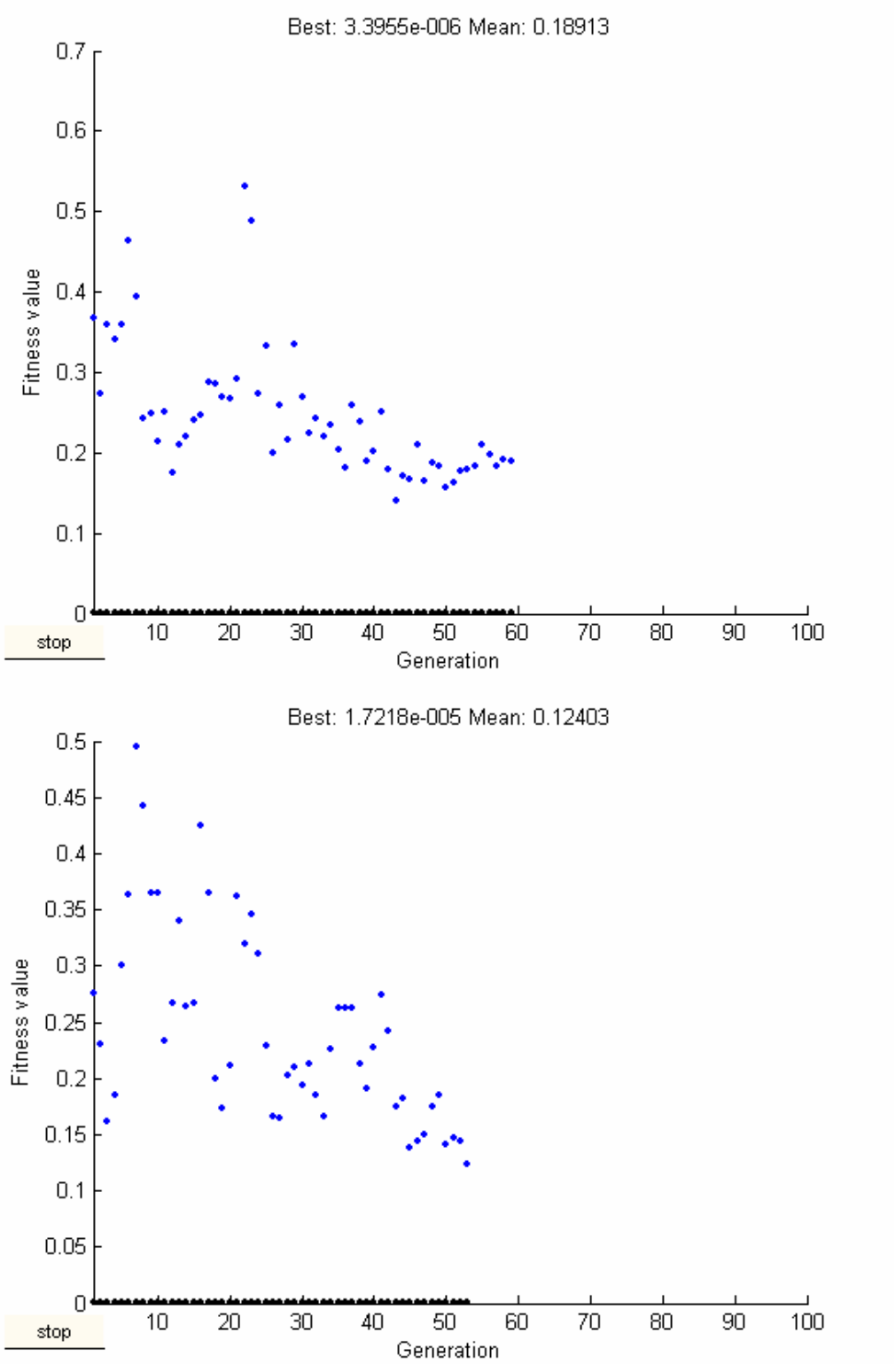

Figure 5.14 : Different type GA optimisation samples

Evolutionary optimization algorithm use the residual error as fitness function. Thus minimization of errors provide optimized population as neural network weights. Different optimization results was presented in this section. (Figure 5.14)

5.4.4 Emerging Network Topology and Fuzzy Logic Analogy

SOM type neural network layer topology was considered as fuzzy variable. According to this approach neurons of the network are represented as 3D relational fuzzy sets. Also rule base neural layer neurons are located in 2D network lattice. According to learning process, network topology modify dynamically (Figure 5.15)

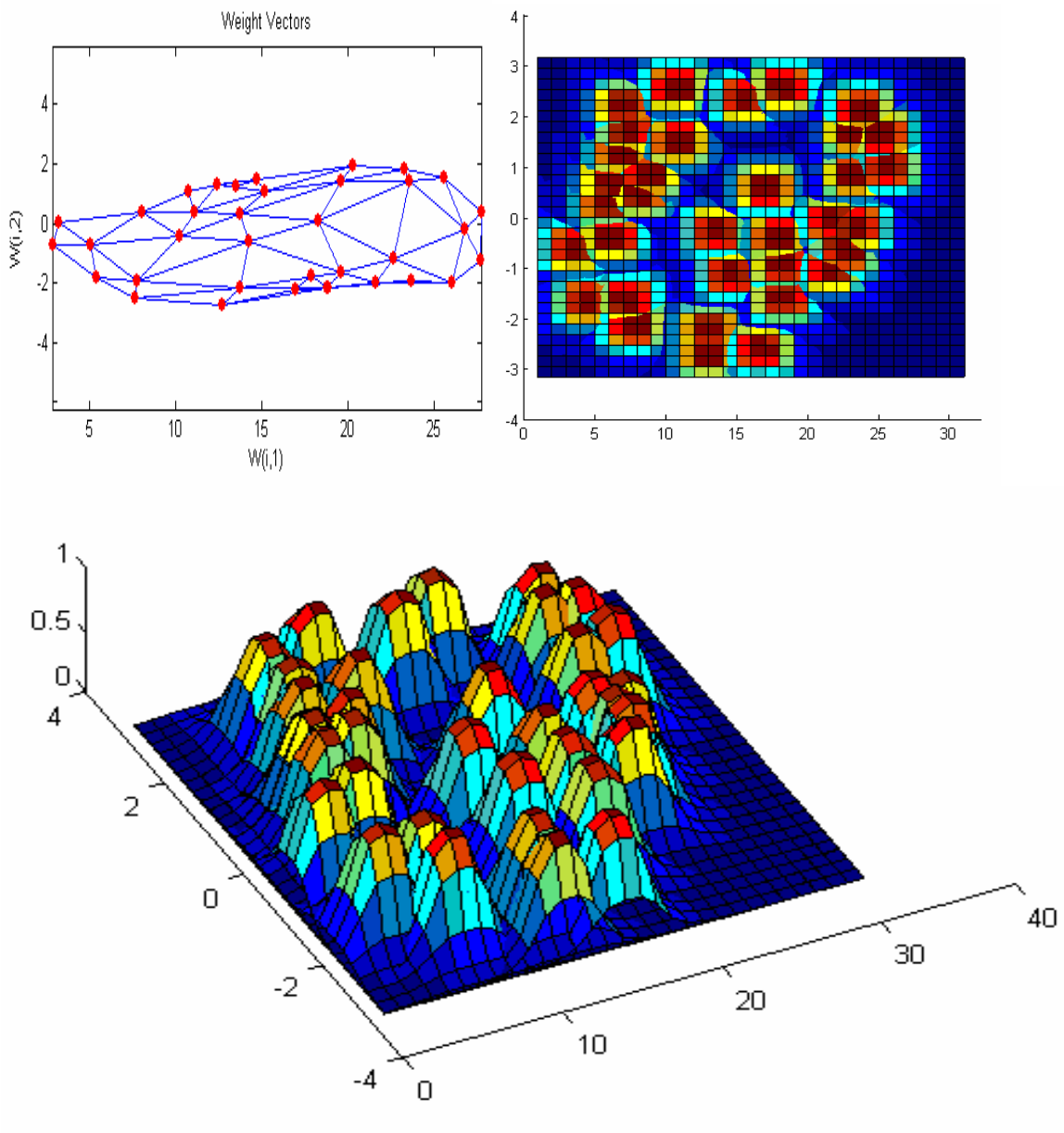

Figure 5.15: Network layer topology, fuzzy relational sets with corresponding neurons

Contour plot of relational fuzzy variable neural layer and its 3D view are shown in figure 5.15. This relational fuzzy set distributions are directly related with SOM type neural network layer neurons.

5.4.5 Case Study Scenarios

Mobile robot trajectories and and behaviors of the mobile robot can be visually observed in 3D Virtual Reality toolbox animation window. In figure 5.15, a sample scenario was animated (Figure 5.16). According to this, mobile robot avoided from obstacles while robot was moving to goal (target point).

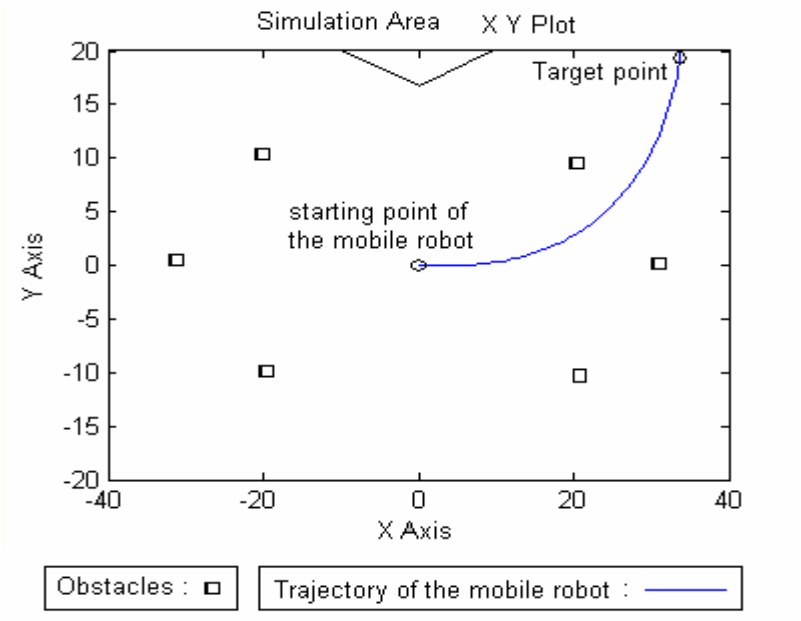

Figure 5.16 : The mobile robot and its trajectory in devised environment

The autonoumus mobile robot can be execute very complex behaviors which include multiple objective robot tasks (Figure 5.17). The complex behavior in figure explain that the robot tries to avoid from obstacles while it move to goal as wander.

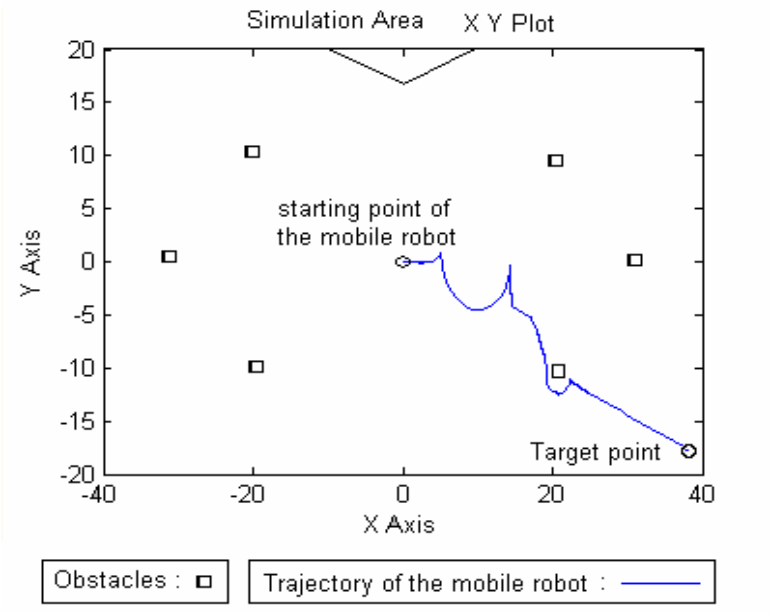

Figure 5.17 : A complex behavior of the mobile robot and including multi-goal robot tasks.

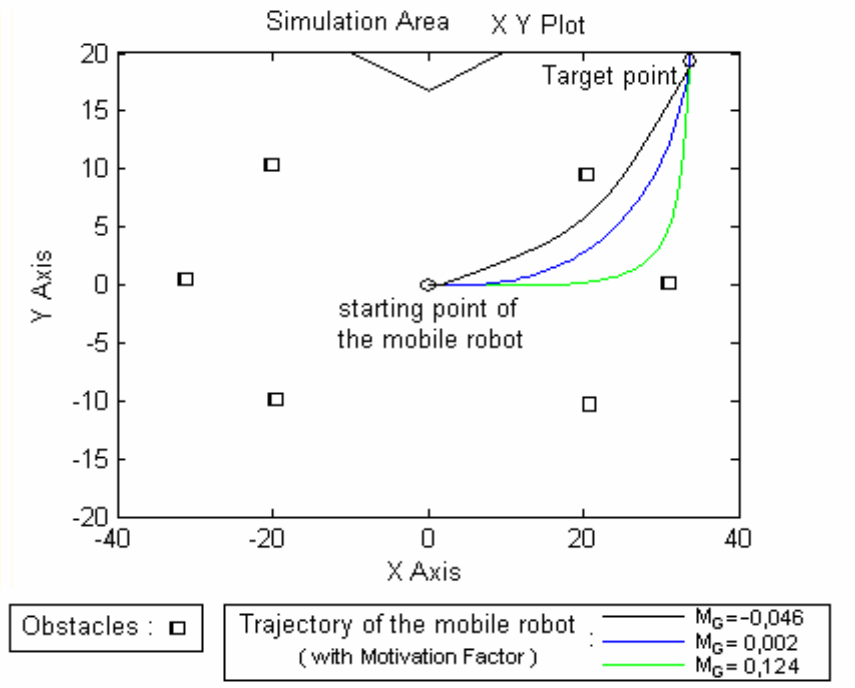

Figure 5.18 : Behavioral gain effects of Motivation factor

Motivational gain effects on artificial emotions are presented in figure 5.18. When artificial emotions which include a behavioral sequence are executed during simulation, motivation factor is effective as behavior gains which take scaled values between -1 and 1 on artificial emotion. Same behaviors of same artificial emotion appear in different form. For instance motivation gain coefficient of the behavioral transition from ”move to goal” to ”head on” is obtained by 2*a(1,4)-1=0,124.

According to the environmental events, the priority filtler of robot control architecture as an event trigger mechanism attempt to influence state transition probabilities. İn this condition, autonoumus robot controller has to make a decision related with matrix A between available or generated behavioral states. Also multiplicative motivation factor as behavioral gain coefficient supports emotional system (Figure 5.19).

$$A = \begin{bmatrix} 0{,}0211 & 0{,}4572 & 0{,}2312 & 0{,}5602 \\ 0{,}1431 & 0{,}9843 & 0{,}0740 & 0{,}3251 \\ 0{,}3846 & 0{,}1153 & 0{,}7282 & 0{,}4820 \\ 0{,}2267 & 0{,}5162 & 0{,}8419 & 0{,}1293 \end{bmatrix} \quad \textbf{(5.1)}$$

$$C = \begin{bmatrix} 0{,}3221 & 0{,}7105 & 0{,}1213 & 0{,}5116 \\ 0{,}2141 & 0{,}0918 & 0{,}1072 & 0{,}6032 \\ 0{,}5382 & 0{,}2171 & 0{,}8720 & 0{,}5148 \\ 0{,}4126 & 0{,}9230 & 0{,}3244 & 0{,}1872 \end{bmatrix} \quad \textbf{(5.2)}$$

When the robot faces obstacles as continuous and periodically, certain behavioral task sequences related with this condition occur and distress state as an artificial emotional expression is activated according to matrix C. For example while the robot is approximating to the obstacles or walls, if the executed behavioral action (current state) is obstacle avoidance corresponding second row of matrix A then it can be repeated because A(2,2) which is diagonal element has highest priority in the second row of matrix A. In this condition, probabilities in the second column of matrix C increase and become bigger than other columns of matrix C.

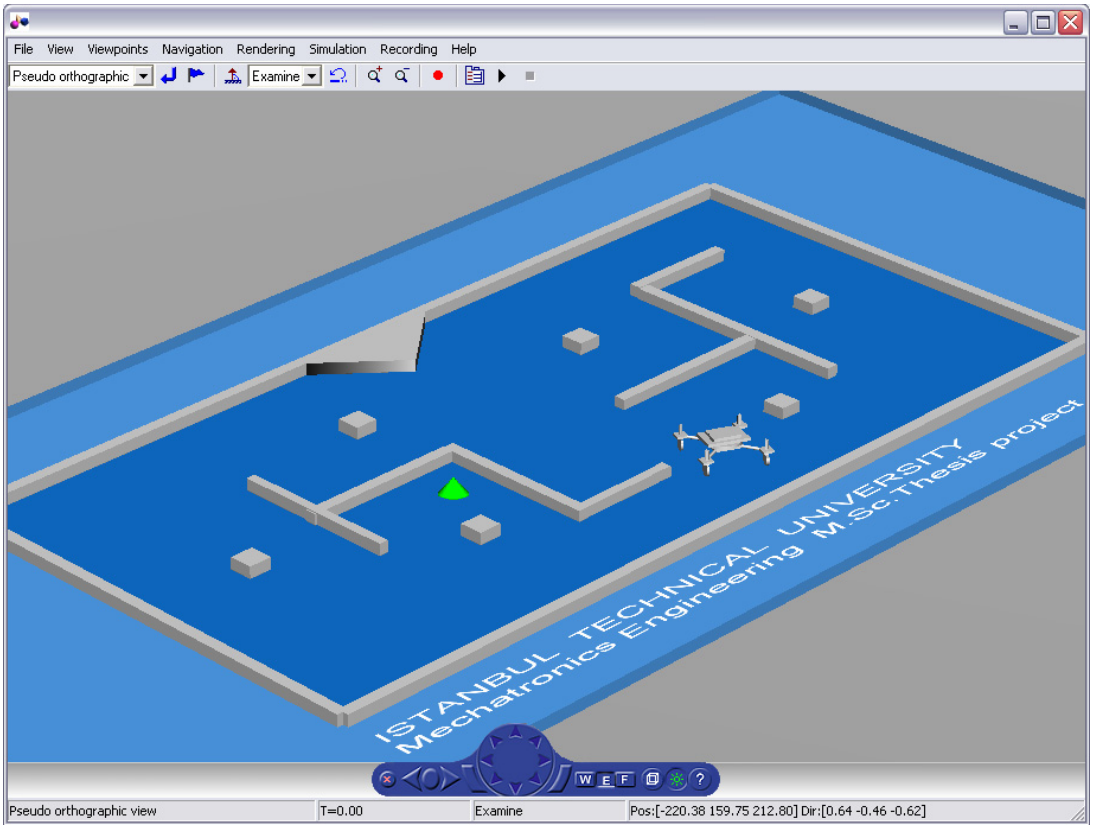

Figure 5.19 : Mobile robot simulation environment with walls

Provided that the robot can not perceives the target due to obstacles and walls, certain behavioral task sequences related with this condition occur and aggressive state as an artificial emotional expression is activated according to matrix C. Another example if the executed behavioral action (current state) is head on corresponding fourth row of matrix A then it can be changed to wander/search behavior because A(4,3) has highest priority in the fourth row of matrix A. In this condition, probabilities in the third column of matrix C increase and become bigger than other columns of matrix C for aggressive emotional expression (Figure 5.20).

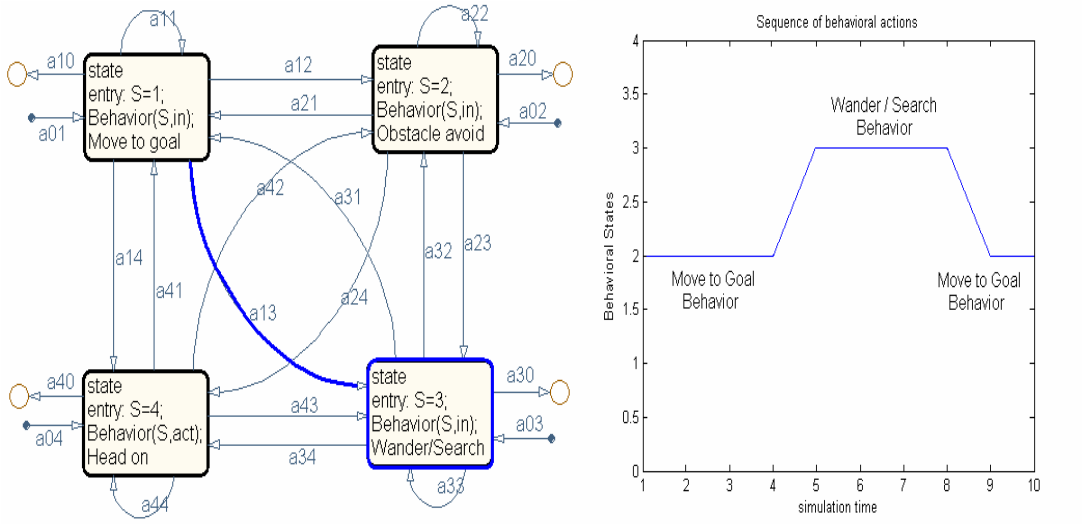

Figure 5.20 : Behavioral state transitions

While the robot can perceives the target point, certain behavioral task sequences related with this condition occur and enjoy state as an artificial emotional expression is activated according to matrix C. For example while the robot is approximating to the target point, if the executed behavioral action (current state) is obstacle avoidance corresponding second row of matrix A then it can be changed to ”move to goal” behavior. Thus A(2,1) must be highest priority in the second row of matrix A. In this condition, probabilities in the fourth column of matrix C increase and become bigger than other columns of matrix C.

If the robot can not see obstacles or when it overcomes obstacles, certain behavioral task sequences related with this condition occur and relief state as an artificial emotional expression is activated according to matrix C. For example while the robot is going away from the obstacles or walls, if the executed behavioral action (current state) is obstacle avoidance corresponding second row of matrix A then it can be switched to ”wander/search”. Therefore A(2,3) must be highest priority in the second row of matrix A. In this condition, probabilities in the first column of matrix C increase and become bigger than other columns of matrix C.

Behavioral learning performances of robot on artificial emotions and comparision between them are presented with different learning rate coefficients in figure 5.21. An artificial emotion based trained working provide best results such as shortest time consumption and success of make a decision for reaching target point. The mean square error of decision making is struggled to reducing minimum.

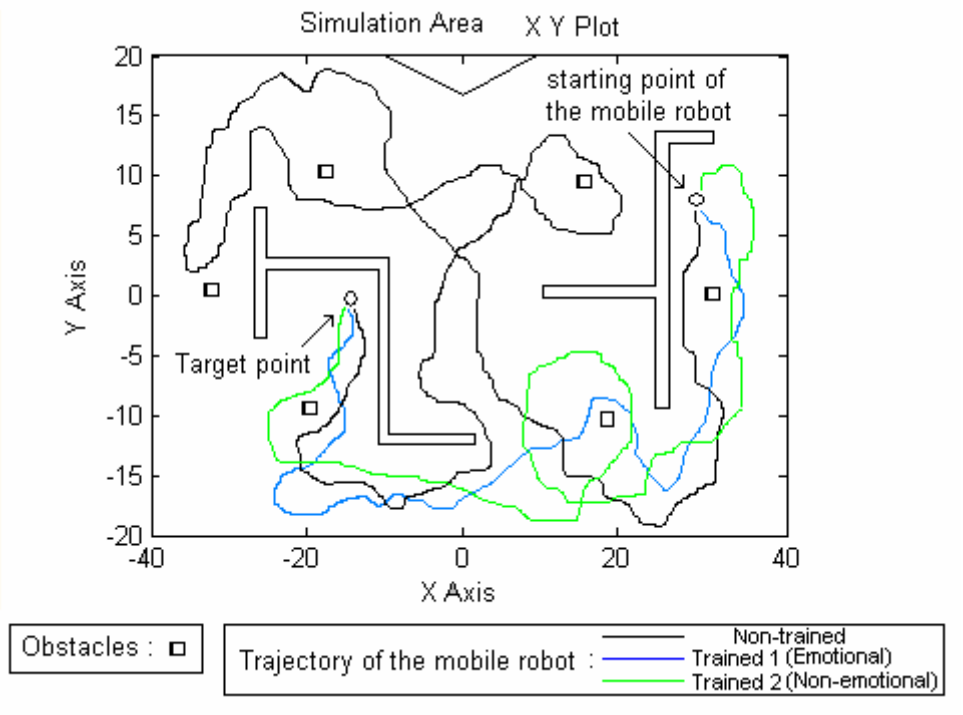

Figure 5.21 : The trained and emotional working trajectories

Also contributions of the artificial emotional system can be easily seen better rather than other behavioral learning performance of robot without artificial emotions for achieve to reaching target point. Results with comparision between them show that multiple task performance of the reaching target point without training (trail represented with black line) succeeds finding target point even though robot is forced and makes a lot of decision mistakes, however trained samples (Blue and green lines) provide better solutions in figure 5.22.

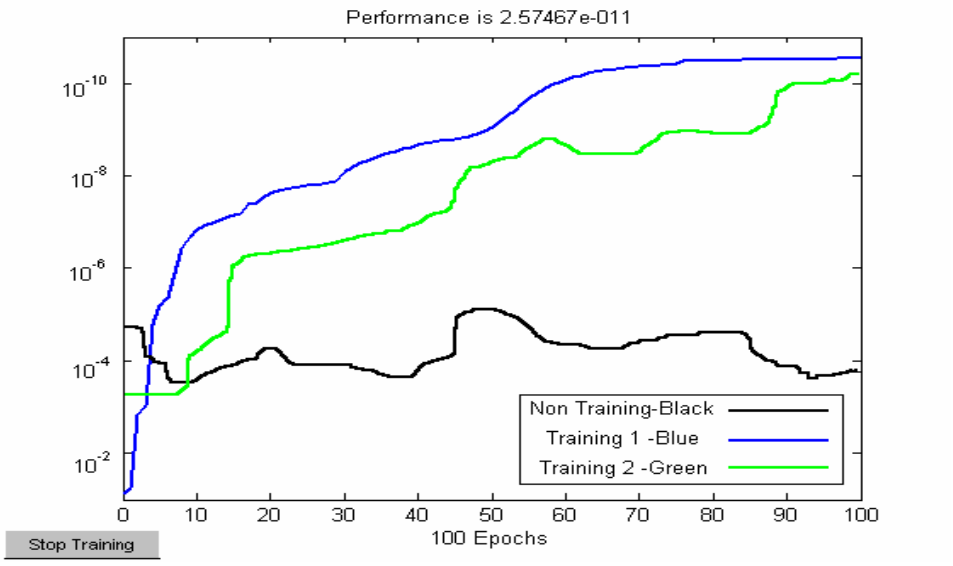

Figure 5.22 : The comparision between training and emotional working

6 Conclusion

In this study, general control architecture was tried to construct for multiple objective robot task menagement problems. This robot control architecture put forward emotion and cognitive based modeling approach. It is composed from three main levels. These are behavioral system level, coordination level, emotion-motivation level.

The behavioral system level is responsible from behavior generation operations. Behavior generation process observed and their results presented depend upon self-reinforcement learning approach by this level. Training performances presented in chapter 5. According to this, training errors were tried to minimize for behavior learning and adaptation. Behavior generation process of the behavioral system level is very complex event in dynamic realistic environment. So weight optimization of behavioral system neural network should be realized by diploid chromosomal genetic programming. Residual error determined as fitness function. However optimization performances is not very efficient.

The main duty of coordination level is to establish relationship between behavior system level and emotion-motivation level. There are several components and sub-modules in this level. These are behavior selection module, cognitive module, instinctual module and priority filter. Behavior selection module and behavioral state transions were tested in a realistic simulation situations based on hidden Markov model. And it is observed by State flow charts. Pre-dominated innate behaviors of the autonoumus mobile robot are located in the instinctual module. According to learning process this module can be improved by adding or removing behaviors. All generated behaviors are derived from instinctual module. Cognitive system is responsible from collecting behavioral rules and learning of the behavioral state transitions. This system employs Q-learning which is a reinforcement learning methodology based on the SOM neural network. Training performance of the Q-SOM and learning status of the behavioral transition pattern observed and presented in chapter 5. According to emotional states, priority filter constitutes priorities of the behaviors as related with behavioral state transition probabilities.

Emotion and motivational level is highest control level of the autonoumus robot control architecture. There are two main part of this level. Emotional system core

which is observational output relation of the discrete state-space hidden Markov model performs artificial emotion state transions and appearing emotional expressions. As a trigger of behavioral event, artificial emotions constitute from sequences of behaviors. This emotional events can be observed during simulation. According to behavioral state transition probabilities, motivational module apply behavioral gain coefficients to the generated or available behaviors.

The simulation environment does not contain the exact robot and world models, and there are approximations and some constraints due to the mathematical background. Making the experiments in a simulation environment gives a good understanding [11]. Also visual simulation environment devised on Virtual Reality developer. So animational events can be observed by Virtual Reality viewer window.

All the information from the environment is Virtual Reality toolbox sink block as sensory data. There are also internal sensors or dynamics of the robot. All these information should be processed and fed into the behaviors [11]. Fuzzification of the behaviors and processing the sensory information as fuzzy sets bring ease and support the reactivity of the behavior. The configurations of the sensors are important on deciding the membership functions of the fuzzy engines [11].

Present architecture can be improved by some contributions in the future. Second order cognitive system is helpful for menagement of the multi-goal robot tasks by fuzzified of artificial emotions. Also second order cognitive system should include emotional learning algorithm. An upper control level can be considered for longer term behavioral working time and deliberative events activation planning. This level should monitors emotional sequences based on second order disrete stochastic state-space model. In motivational module, better and more rational behavioral gain distribuition method can be investigated for intense of the generated or available behaviors. According to improving of the robot hardware, deeper study on behavioral model should be achieved. All of the free parameters which belong to the proposed architecture should be tried to design as more adaptive. Predominated behaviors of the instictual module can be able to gain new behavior by behavioral experience of the autonumus mobile robot. Finally, dynamic model organizer module can be considered as a mathematical model which use such as Kalman filter on the state-space model and Lyapunov stability approach for stability of this proposed system and establishing relationship between events.

References

[1] **Sandra, C.G**., 2003. Learning Behavior-Selection by Emotions and Cognition in a Multi-Goal Robot Task, Institute of Systems and Robotics, Portugal.

[2] **Brooks, R.A**., 1987. Intelligence without representation, MIT Artificial Intelligence Laboratory, USA.

[3] **Holland, O.**, 2002. The first biologically inspired robots, University of Essex, Colchester, UK.

[4] **Arkin, R. C.**, 1999. Behavior Based Robotics, The MIT Press, Cambridge, Massachusets, London, England.

[5] **Bizzi, E., Mussa-Ivaldi, F., Gistzer, S.**, 1991. Computations Underlying the Execution of Movement: A Biological Perspective, Science, Vol. 253, July, pp. 287-291.

[6] **Srinivasan, M. V., Zhang, S. W., Lehrer, M., Collett, T. S**., J. Exp. Biol. 199, 237.

[7] **Srinivasan, M. V.**, 1992. Distance perception in insects. Current Directions in Psychological Science, 1, 22-26.

[8] **Arkin, R. C.**, 2003. Moving up the Food Chain: Motivation and Emotion in Behavior-Based Robots, Georgia Institute of Technology, USA.

[9] **Svetlicic, I.**, 2004. Emotion Based Subsumption Architecture for Mobile Robotics, Masters Thesis, Dept. of Computer Science and System Analysis, Miami University, Ohio, USA.

[10] **Michaud, F.**,2002. EMIB — Computational Architecture Based on Emotion and Motivation for Intentional Selection and Configuration of Behavior-Producing Modules, Hermes Science Publications, Université de Sherbrooke, Canada.

[11] **Şirin, M.**, 2005. Behavior-Based Fuzzy Control For a Mobile Robot With Non-Holonomic Constraints, Istanbul Technical University, Turkey.

[12] **Honkela, A.**, 2001. Nonlinear Switching State-Space Models, Department of Engineering Physics And Mathematics, Helsinki University

[13] **Kurimo, M.**, 1997. Using Self-Organizing Map and Learning Vector Quantization For Mixture Density Hidden Markov Models, Neural Networks Research Centre, Helsinki University of Technology.

[14] **Katagiri, S., Lee, C.H.**, 1993. A New Hybrid Algorithm For Speech Recognition Based on HMM Segmentation and Learning Vector Quantization, IEEE Transactions on speech and audio processing, 1(4) : 421-430

[15] **Lawrence R. Rabiner.,** 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257-286

[16] **Bahl, L., Brown, P., de Souza, P., Meroer, R.**, 1986. Maximum Mutual Information Estimation of Hidden Markov Model Parameters For Speech Recognition. In proceedings of the IEEE international conference on Signal and Speech Processing, pages 49-52.

[17] **Baker, J. M.** , 1975. The Dragon System – an overview, IEEE Transactions on Acoustics, Speech and Signal Processing, 23(1):24-29

[18] **Bellegarda, J., Nahamoo, D.** , 1990. Tied Mixture Continuous Parameter Modeling for Speech Recognition. IEEE Transactions on Acoustics, Speech and Signal Processing, 38(12):2033-2045

[19] **Bahl, L., Brown, P., de Souza, P., Meroer, R.**, 1988. A new algorithm for the estimation of hidden Markov model parameters. In proceedings of the IEEE international conference on Signal and Speech Processing, pages 493-496.

[20] **Amari, S.**, 1967. The theory of adaptive pattern classifiers, IEEE Transactions on Electronic and Computer, 16(3):299-307.

[21] **Watkins, J.C.H., Dayan, P.**, 1992. Q-Learning Technical Note. Machine Learning, 8, 279-292, Netherlands.

[22] **Kwee-Bo, S., Lee, D.W., Zhang, B.**, 2002. Behavior Evolution of Autonoumus Mobile Robot Using Genetic Programming Based on Evolvable Hardware. International Journal of Fuzzy Logic and Intelligent Systems, vol. 2, no. 1, page 20-25.

[23] **Cavill, R., Smith, S., Tyrrell, A.**, 1992. Multi-Chromosomal Genetic Programming. Machine Learning, 8, 279-292, Netherlands.

[24] **Schafer, R.**, 1992. Using a Diploid Genetic Algorithm to Create and Maintain a Complex System in Dynamic Equilibrium . Machine Learning, 8, 279-292, Netherlands.

[25] **K. Vekaria and C. Clack**, 1997. Haploid genetic programming with dominance. Technical Report RN/97/121,

[26] **Banzhaf W., Nordin P., Keller R. E., Francone F. D.**, 1998.in Genetic Programming an Introduction, Morgan Kaufmann Publishers, Inc,

[27] **Byoung-Tak Zhang, Dong-Yeon Cho**, 1998. 1997. “Fitness Switching: Evolving Complex Group Behaviors Usign Genetic Programming,” Proceedings of Genetic Programming 1998, pp. 431-438,

[28] **Eiben, A.E., Jelasity, M.**, 2002, A Critical Note on Experimental Research Methodology in EC, Netherlands.

[29] **Eiben, A.E.**, 2002, Evolutionary Computing : The most powerful solver in universe, Duch Mathematical Archive, 5/3(2): 126-131.

[30] **Kohonen, T.**, 1987, Self-Organization and Associative Memory, 2nd Edition, Berlin: Springer-Verlag.

[31] **Kohonen, T.**, 1997, Self-Organizing Maps, 2nd Edition, Berlin: Springer-Verlag.

[32] **Neural Networks Toolbox**, 2005. The MathWorks inc. Version 4, The MathWorks, Inc. Mail 3 Apple Hill Drive Natick, MA 01760-2098

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

[34] **Maes, P.**, 1990. Situated Agents Can Have Golas, Robotics and Autonomous Systems, Vol.6, pp. 49-70.

[35] **Rosenblatt, J., Payton, D.**, 1989. A Fine Grained Alternative to Subsumption Architecture for Mobile Robot Control, Proceedings of the International Joint Conference on Neural Networks, June, pp. 317-323.

[36] **Vadakkepad, P., Miin O.C., Peng X., Lee T. H.**, 2004. Fuzzy Behavior-Based Control of Mobile Robots, IEEE Transactions on Fuzzy Systems, Vol. 12, No. 4, Auust pp. 559-564.

[37] **Rusu, P., Petriu E. M., Whallen T. E., Cornell, A., Spoelder, H. J. W.**, 2003. Behavior-Based Neuro-Fuzzy Controller for Mobile Robot Navigation, IEEE Transactions on Instrumentation and Measurement, Vol. 52, No. 4, August, pp. 1335-1340.

[38] **Thongchai, S., Suksakulchai, S., Wilkes, D. M., Sarkar, N.**, 2000. Sonar Behavior-Based Fuzzy Control for a Mobile Robot, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Nashwille, Tenessee, 8-11 October.

[39] **Kenyon, S.H.**, 2003. The Need For Emotional Architectures in Practical Robots, Cambridge, MA: MIT Press,

[40] **Nolfi, S., Floreano, D.**, 2000. Evolutionary Robotics, MA: The MIT Press, Cambridge, London

[41] **De Lisle, R.,** 2006. Kohonen's Self Organizing Feature Maps, www.ai-junkie.com/ann/som, access : 04.04.2006.

[42] **Duro, J.R., Santos, J., Grana, M.**, 2001. Biologically Inspired Robot Behavior Engineering, Physica-Verlag Heidelberg, Germany

[43] **Luke S., Spector L.**, 1997. “A Comparison of Crossover and Mutation in Genetic Programming,” Proceedings of Genetic Programming , pp. 240-248,

[44] **Luke,** http://koti.mbnet.fi/~phodju/nenet/NeuralNetworks/NeuralNetworks.html, Access date 04.04.2006

[45] **C. C. Lee**, 1990. “Fuzzy logic in control systems: Fuzzy logic controller–Part II,” *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 419–435, Mar./Apr.

[46] **Kuhnlenz, K., Buss, M.,** 2004. Towards an Emotion Core Based on a Hidden Markov Model, Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication Kurashiki, Okayama Japan September 20-22

[47] **R.J.Elliott, L. Aggoun, and J.B. Moore**, 1997. “Hidden Markov Models: Estimation and Control”, Applications of Mathematics: Stochastic Modeling and Applied Probability, Vol. 29, Springer, Berlin,

[48] **Resch, B.**, Hidden Markov Models : A Tutorial for the Course Computational Intelligence, Signal Processing and Speech Communication Laboratory, Graz

[49] **L. Kaelbling**, Learning in embedded systems, MIT Press, 1993.

[50] **Touzet, C.**, 1997. Neural Reinforcement Learning for Behaviour Synthesis, Robotics and Autonomous Systems, Special issue on Learning Robot: the New Wave, N. Sharkey Guest Editor,

[51] **O. Holland, M. Snaith**, "Extending the adaptive heuristic critic and Q-learning: from facts to implications," Artificial Neural Networks, 2, I. Alexander and J. Taylor (Eds.) Elsevier Science Publishers, 599-602, 1992.

[52] **C. J. C. H. Watkins**, "Learning from Delayed Rewards," Ph.D. thesis, King's College, Cambridge, England, 1989.

[53] **S. Sehad & C. Touzet**, "Reinforcement Learning and Neural Reinforcement Learning," ESANN 94, Editor M. Verleysen, D-Facto publication, Brussels, April 1994.

[54] **R. A. Mc Callum**, 1992. Using transitional proximity for faster reinforcement learning, Proc. of the Ninth International Conference on Machine Learning, Morgan Kaufman (GB).

[55] **Virtual Reality Toolbox**, 2005. The MathWorks Inc. 3 Apple Hill Drive Natick, 01760-2098

[56] **State Flow Toolbox**, 2005. The MathWorks Inc. 3 Apple Hill Drive Natick, 01760-2098

Appendix A: Modeling of 4 Wheel Steered 4 Wheel Driven Mobile Robot

A.1 Kinematics

A.1.1 Robot Posture

Robot posture is the position and orientation of the robot frame according to a reference frame O. The robot frame is the x-y frame fixed on the body of the robot as in figure A.1.

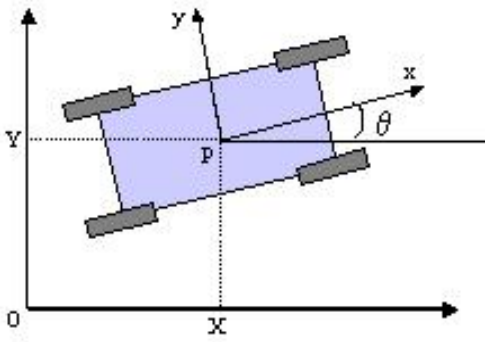

Figure A.1 : Robot posture

The posture is a vector composed of position "X,Y" and orientation "θ" of the robot as;

$$\xi = \begin{bmatrix} X & Y & \theta \end{bmatrix}^T \quad \textbf{(A.1)}$$

The rotation matrix for these two frames orthogonal to the plane is;

$$R(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \textbf{(A.2)}$$

A.1.2 Wheels

A.1.2.1 Pure rolling, no-slip

There are four centered orientable wheels, each of which is able to rotate about the vertical axis passing through the contact point of the wheel (steer) and horizontal

axis passing through the center of the wheel (drive). The steer angle is "β_i" and the drive angle is "φ_i" for i^{th} wheel (Figure A.2).

The velocity of the contact point of the wheel to the ground should be zero in order to satisfy the pure rolling no-slip assumption (Figure A.3). The equations constraining the motion of the wheel are;

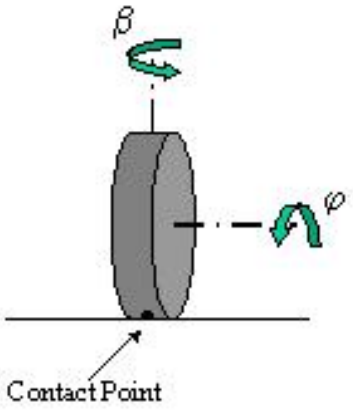

Figure A.2 : Off-centered wheel

Pure rolling;

$$\begin{bmatrix} -\sin(\gamma+\beta) & \cos(\gamma+\beta) & l\cos(\beta) \end{bmatrix} R(\beta)\dot{\xi} + r\dot{\varphi} = 0 \qquad \textbf{(A.3)}$$

No-slip;

$$\begin{bmatrix} \cos(\gamma+\beta) & \sin(\gamma+\beta) & l\sin(\beta) \end{bmatrix} R(\beta)\dot{\xi} = 0 \qquad \textbf{(A.4)}$$

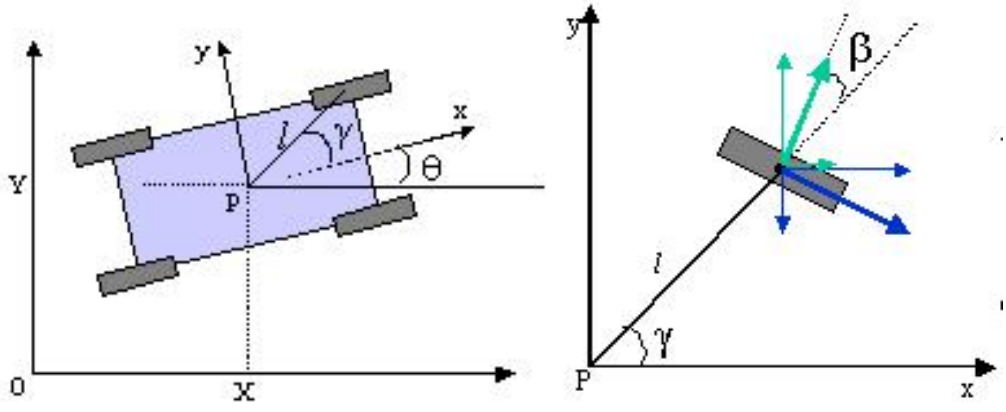

Figure A.3 : Pure rolling, no-slip constraints

A.1.2.2 ICR

According to Descartes' principle of instantaneous motion; at each instant, the motion of a planar rigid body coincides either with a pure translation, or with a pure rotation about some point, termed instantaneous center of rotation.

This principle is also applies to any point on the robot which is assumed to be rigid and moving on a horizontal plane. The Instantaneous Center of Rotation (ICR) may be anywhere between the moving point itself and infinity. If we apply this rule to the wheel-centers, we got the following figure (Figure A.4 : ICR).

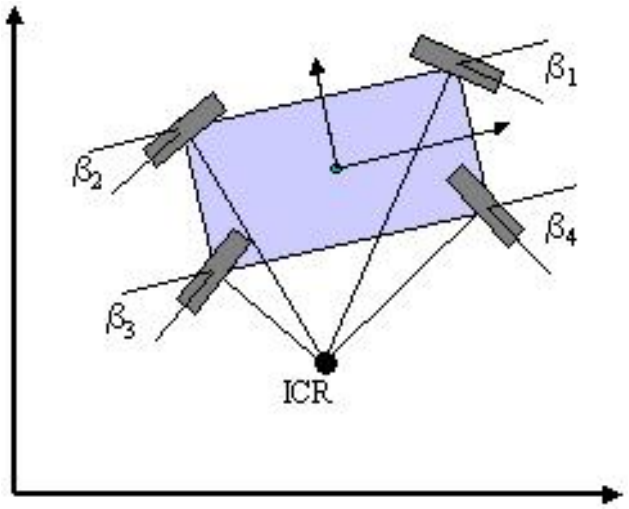

Figure A.4 : ICR

This phenomenon implies that the orientations of the wheels have some constraints, i.e. the horizontal axes of the wheels should coincide at ICR. These axes do not coincide if and only if the ICR is at infinity i.e. the wheels are parallel and the robot has only transverse motion.

Since two coinciding lines are enough to describe a point, orientations of any two wheels (β_c), for instance β_1 and β_2, are enough to describe the position of the ICR. Once the ICR is located, the orientations of the other two wheels can be calculated as;

$$\beta_3 = \tan^{-1}\left(\frac{\cos(\beta_1)\sin(\beta_2)}{\cos(\beta_1)\cos(\beta_2)+\frac{d_{14}}{d_{12}}\sin(\beta_{2-}\beta 1)}\right) \quad \textbf{(A.5)}$$

$$\beta_4 = \tan^{-1}\left(\frac{\sin(\beta_1)\cos(\beta_2)}{\cos(\beta_1)\cos(\beta_2)+\frac{d_{14}}{d_{12}}\sin(\beta_{2-}\beta 1)}\right) \quad \textbf{(A.6)}$$

A.1.2.3 Motion

Considering equations 4.3 and 4.4 for all 4 wheels;

$$J_1(\beta)=\begin{bmatrix}\cos(\beta_1) & \sin(\beta_1) & l_1\sin(\beta_1-\gamma_1)\\ \cos(\beta_2) & \sin(\beta_2) & l_2\sin(\beta_2-\gamma_2)\\ \cos(\beta_3) & \sin(\beta_3) & l_3\sin(\beta_3-\gamma_3)\\ \cos(\beta_4) & \sin(\beta_4) & l_4\sin(\beta_4-\gamma_4)\end{bmatrix} \quad \textbf{(A.7)}$$

$$J_2=\begin{bmatrix}r_1 & r_2 & r_3 & r_4\end{bmatrix}I_{4\times4} \quad \textbf{(A.8)}$$

$$C_1(\beta)=\begin{bmatrix}-\sin(\beta_1) & \cos(\beta_1) & l_1\cos(\beta_1-\gamma_1)\\ -\sin(\beta_2) & \cos(\beta_2) & l_2\cos(\beta_2-\gamma_2)\\ -\sin(\beta_3) & \cos(\beta_3) & l_3\cos(\beta_3-\gamma_3)\\ -\sin(\beta_4) & \cos(\beta_4) & l_4\cos(\beta_4-\gamma_4)\end{bmatrix} \quad \textbf{(A.9)}$$

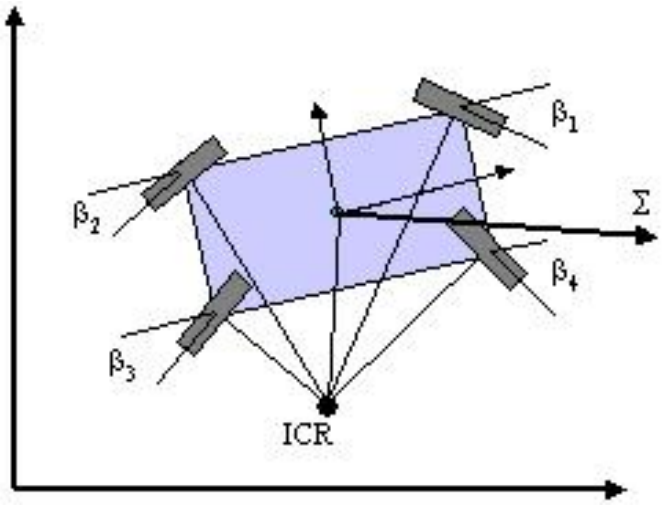

Figure A.5 : Instantaneous direction of motion

ICR and pure rolling, no-slip constraints describe an instantaneous direction of motion $\Sigma(\beta_c)$ for the robot. (Figure A.5)

$\Sigma(\beta_c)$ is perpendicular to the space spanned by C_1, thus;

$C_1(\beta)\Sigma(\beta_c)=0$ and,

$$\Sigma=\begin{bmatrix}l_1\cos(\beta_2)\cos(\beta_1-\gamma_1)-l_2\cos(\beta_1)\cos(\beta_2-\gamma_2)\\ l_1\sin(\beta_2)\cos(\beta_1-\gamma_1)-l_2\sin(\beta_1)\cos(\beta_2-\gamma_2)\\ \sin(\beta_1-\beta_2)\end{bmatrix} \quad \textbf{(A.10)}$$

On this direction, there is a velocity η(t).

So, the motion of the robot can be described as,

$$\dot{\xi} = R^T(\theta)\Sigma(\beta_c)\eta. \qquad \textbf{(A.11)}$$

This means that, the posture of the robot can be manipulated with a velocity input $\eta(t)$ at the instantaneous direction of $\Sigma(\beta_c)$.

A.2 Dynamics

A.2.1 Type of the mobile robot

Wheeled Mobile Robots can be classified into 5 groups. These groups are formed with two parameters; *degree of mobility* δ_m and *degree of steeribility* δ_s, and named as "mobile robot of Type(δ_m, δ_s)". Without going into the mathematical descriptions, these parameters can be described as,

Degree of mobility: The number of planar movements (x, y, θ) that a robot can go without changing its wheel configurations. This number is between 1 and 3.

Degree of steeribility: The number of conventional centered orientable wheels that can be oriented independently to steer the mobile robot. This number is between 0 and 2.

According to these explanations, the mobile robot studied in this thesis is a degenerate mobile robot of Type(1,2).

The nondegenerate form of the robot is stated in Figure A.6.

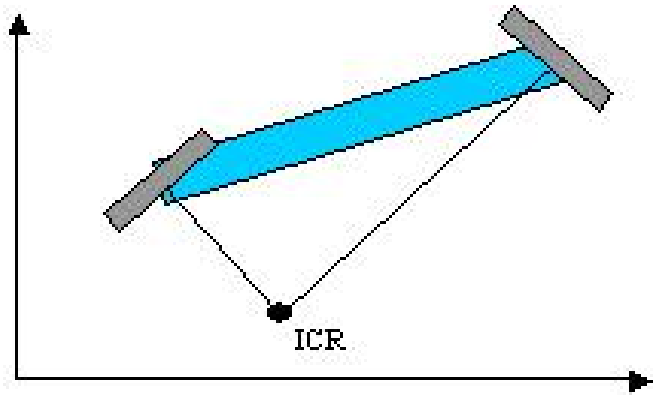

Figure A.6 : non-degenerate form; bicycle model

A.2.2 Torque calculation

If it is assumed that the mass distribution on the robot frame is symmetric about the x and y axes, the off-diagonal terms of the mass matrix vanishes and the matrix becomes;

$$M = \begin{bmatrix} m_f + 4m_w & 0 & 0 \\ 0 & m_f + 4m_w & 0 \\ 0 & 0 & I_f + m_w \sum_{i=1}^{4} l_i^{\,2} \end{bmatrix} \quad \textbf{(A.12)}$$

Where m_f is the mass of the robot frame and m_w is the mass of the wheel and the moment of inertia of the robot frame is I_f and for each wheel is I_w where $J_\phi = I_w I_{4x4}$

Using the Lagrange undetermined coefficients, the general dynamical model can be written as,

$$h_1(\beta)\dot{\eta} + \Phi_1(\beta)\zeta\eta = \Sigma^T E \tau_\phi \quad \textbf{(A.13)}$$

where

$$h_1(\beta) = \Sigma^T \left(M + E J_\phi E^T\right)\Sigma > 0 \quad \textbf{(A.14)}$$

and

$$\Phi_1(\beta) = \Sigma^T \left(M + E J_\phi E^T\right) N(\beta_c) \quad \textbf{(A.15)}$$

Where we define

$$E = J_1^T J_2^{-1} \quad \textbf{(A.16)}$$

and

$$N(\beta_c) = \begin{bmatrix} N_1 & N_2 \end{bmatrix} \quad \textbf{(A.17)}$$

such that,

$$N_1 = \begin{bmatrix} -l_1\cos(\beta_2)\sin(\beta_1-\gamma_1)+l_2\sin(\beta_1)\cos(\beta_2-\gamma_2) \\ -l_1\sin(\beta_2)\sin(\beta_1-\gamma_1)-l_2\cos(\beta_1)\cos(\beta_2-\gamma_2) \\ \cos(\beta_1-\beta_2) \end{bmatrix}$$

$$N_1 = \begin{bmatrix} -l_1\sin(\beta_2)\cos(\beta_1-\gamma_1)+l_2\cos(\beta_1)\sin(\beta_2-\gamma_2) \\ l_1\cos(\beta_2)\cos(\beta_1-\gamma_1)+l_2\sin(\beta_1)\sin(\beta_2-\gamma_2) \\ -\cos(\beta_1-\beta_2) \end{bmatrix} \quad \textbf{(A.18)}$$

A.2.3 Torque distribution

Having the non-holonomic constraints and the degree of mobility of the robot as 1, theoretically it will be enough to drive only one wheel to move the robot. This phenomenon can be understood better with a “train and railway” analogy.

The degree of mobility of a train is also 1, which means that a train can only move on the direction of the railway (i.e. forward or backward). Any one of the wheels or any group of wheels can drive the train supposed that the total torque is enough to move the train. The subject is not the distribution of the total torque, but the total of the torques.

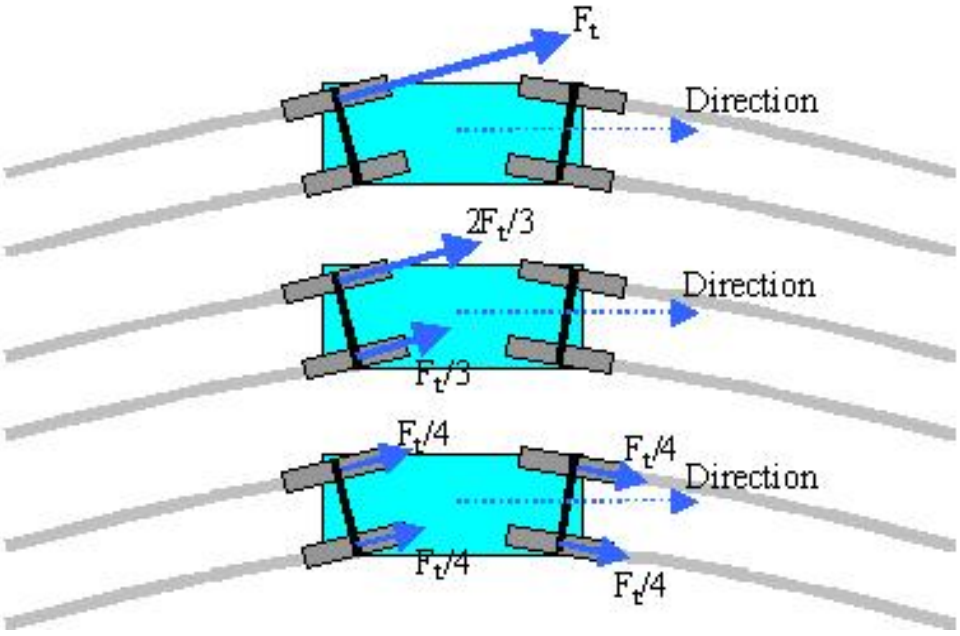

Figure A.7 : Railway analogy

By dividing the torque to the wheels evenly, the torque value of each wheel is limited in a nominal value (Figure A.7).

τ_ϕ cannot be extracted from eq. 10 directly. After linearizing eq.10 using computed torque approach, the torques are evenly distributed to each wheel as;

$$\Sigma^T E \tau_\phi = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \end{bmatrix}^T = h_1(\beta)\dot{\eta} + \Phi_1(\beta)\varsigma\eta = L \qquad \textbf{(A.19)}$$

We set $\tau_\phi = H\tau_0$ and H_i=Lsign(a_i)/σ where σ is the sum of the four elements of the vector Σ^TE.

Now τ_0 can be obtained from

$$\tau_0 = \frac{1}{\Sigma^T EH}\left(h_1(\beta)\dot{\eta} + \Phi_1(\beta)\varsigma\eta\right) \qquad \textbf{(A.20)}$$

A.3 Steering Strategy

A.3.1 Translation

Unlike the car-like vehicles, a 4x4x4 vehicle does not have to be parallel to its track. The vehicle may have a yaw angle between the body orientation and the direction of motion (Figure A.8).

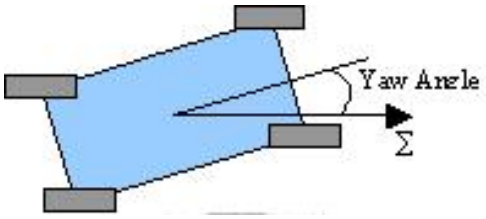

Figure A.8 : Yaw angle

In order to steer the robot to a point in the plane, simply β_1 and β_2 angles are set to the angle of the vector that is connecting the robot to the point (Figure A.9).

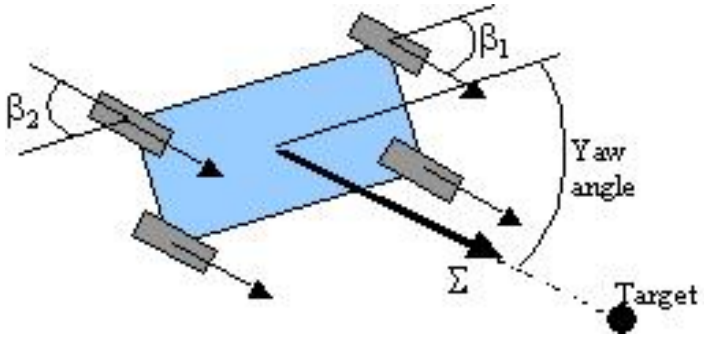

Figure A.9 : Transverse motion

A.3.2 Rotation

The yaw angle of the robot gives the advantage to control the orientation θ. For a stationary situation, i.e., the ICR is at the center of the robot, the directions of motion for the 4 wheel orientation points on the robot are perpendicular to the line connecting the points to the body center.

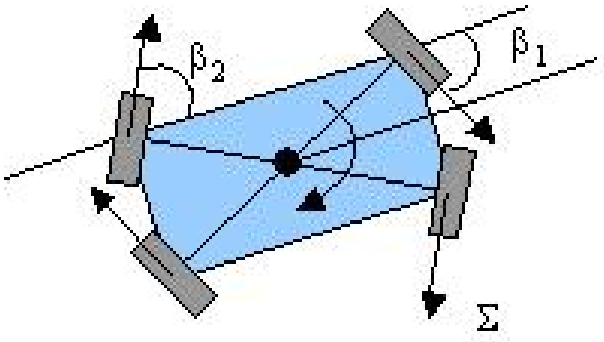

Figure A.10 : Rotation

The wheel speeds are the same and the wheel angles β_1 and β_2 are the same in magnitude but opposite in sign (Figure A.10).

A.3.3 Superposition

These two motions can be independently controlled. The resulting robot configuration is the superpositioning of the speed vectors of each wheel. This configuration also satisfies the ICR (Figure A.11).

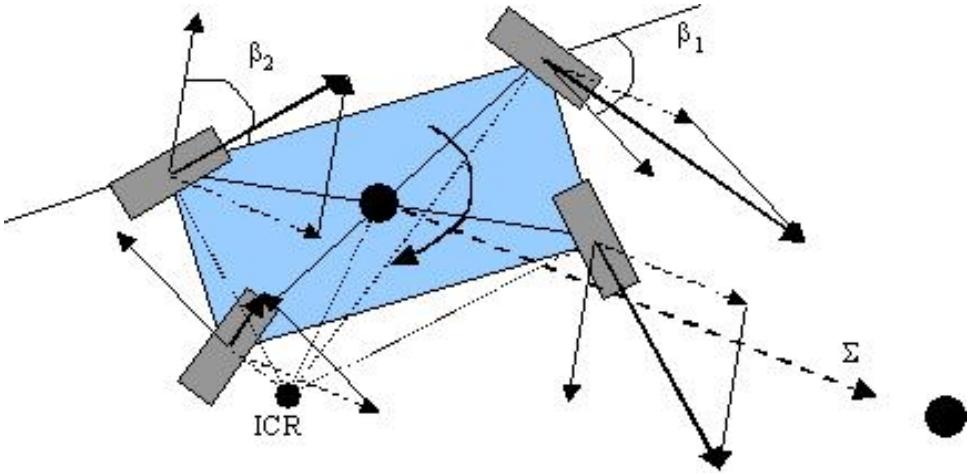

Figure A.11 : Superpositioning of transverse and rotational motions

The solid arrows show the vectors for rotation, and the dashed arrows show the vectors for translation. For superpositioning, simply by vector calculation, the solid arrows are added to the dashed arrows. The resulting solid bold arrows are the speed vectors of each wheel. This vector superpositioning gives the ICR and speeds of each wheel together with the instantaneous direction of motion Σ.

RESUME

He was born in Ankara, 1981. He completed the primary and secondary education in Kütahya Atatürk high school. After he graduated from İzmir Buca Technical high school in 1999. Then he started university education at Marmara University Technical Education Faculty, Electrical education department and graduated in 2004. He studied Advanced Technologies graduate program at Gazi University in Ankara, 2005. Meanwhile he worked as research and teaching assistant in Electrical and Electronics Engineering department at Atilim University. He started Mechatronics Engineering master program at İstanbul Technical University in 2005 and graduated in 2007.

Özgeçmiş

1981'de Ankara'da doğdum. İlk ve orta öğretimimi Kütahya'da 50.yıl İlkokulu ve Atatürk Lisesi'nde tamamladıktan sonra liseyi İzmir'de Buca Teknik Lisesi Elektrik Bölümü'nde okuduktan sonra 2000'de Istanbul'da Marmara Üniversitesi Teknik Eğitim Fakültesi Elektrik Öğretmenliği bölümüne girdim. 2004'de mezun olup Ankara'da Gazi Üniversitesi Ileri Teknolojiler Anabilim dalı'nda yüksek lisans eğitimime devam ettim. Aynı zamanda Atılım Üniversitesi Elektrik Elektronik Mühendisliği Bölümünde araştırma görevlisi olarak görev yaptım. 2005'de İstanbul Teknik Üniversitesi'nde Mekatronik Mühendisliği Yüksek Lisans Programına başladım ve 2007'de bitirdim.