

# Modification of Kohonen Rule for Vehicle Path Planning by Behavioral Cloning

Ranka Kulić  
*Faculty of Maritime of Studies  
Montenegro*

## Abstract

The problem of path generation for the autonomous vehicle in environments with infinite number obstacles is considered. Generally, the problem is known in the literature as the path planning. This chapter treated that problem using the algorithm, named MKBC, which is based on the behavioral cloning and Kohonen rule. In the behavioral cloning, the system learns from control traces of a human operator. Kohonen rule connected with the weighting coefficients, while the MKBC algorithm does not use the weighting values as values from the previous time, but permanently uses the training values as weighting values. That is something which enables an intelligent system to learn from the examples (operator's demonstrations) to control a vehicle in the process of the obstacles avoiding, like the human operator does. Like that, the very important MKBC characteristic is the simplicity. The MKBC simplicity is something which is so obviously, specially according to the RBF neural network and the machine learning algorithm which is used the previously. Following the MKBC given context the problem narrow passage avoiding and the goal position reaching fundamentally is observed. Namely, defining if - then rule, according to the named cases is treated as destroying of the consistency of the methodology. In that sense, using MKBC neural network the solution was found. At the end, the autonomous vehicle mathematical model which is given by nonlinear equations describing a 12 state dynamical system is used and in that case the MKBC algorithm is applied successfully. Eventually, as it has been illustrated the previously, the advantage of the entire methodology lies in the fact that a complete path of the vehicle can be defined off-line, without using sophisticated symbolical models of obstacles. These are facts that MKBC algorithm and the given methodology substantially differ from the others. In the next phase it is expected to confirm results in on - line simulation process.

Key words: vehicle path planning, behavioral cloning, cloning success, obstacle avoiding, machine learning, Kohonen rule, neural network, Shark dynamical model.

## 1. Introduction

In the last years an increasing interest in mobile robots has appeared, notably in aeronautical space exploration, automatized agriculture, collective mobile robot games, and so on (P. Vadakkepat, X. Peng et al., 2007). These application require the mobile robot to move in partially known environments with the high amount of uncertainty. Moving - obstacle

avoidance with unknown obstacle trajectory remains remarkable challenge and has opened a research area in the control of the mobile robots, but it is treated in our research using the methodology which is connected with unmoving obstacles. Many approaches are available to study this research area. In its simplest form, the motion planning problem can be defined as follows (C. Latombe, 1991; J. Schwartz, M. Sharir, J. Hopcroft, 1987). Let  $B$  be the autonomous vehicle consisting of collection of rigid subparts having a total  $k$  degrees of freedom, and let  $B$  be free to move in two - or three - dimensional space  $V$ , avoiding obstacles whose geometry is known. For a given initial position  $S$  and a desired target position  $G$  of  $B$ , the task is to determine whether there exist a continuous obstacle - avoiding motion of  $B$  from  $S$  to  $G$ , and if so, to find such a motion. The simplest collision avoidance algorithm fall into the generate and test paradigms. A simple path from  $S$  to  $G$ , usually a straight line, is hypothesized and then it is tested for potential collisions between  $B$  and obstacles. If collision is detected, a new path is proposed using information about detected collision. This process repeats until no collision is detected. But in spite of its simplicity these methods have not found significant application. They have several fundamental drawbacks. One of these is inability to propose a radically different and better path from local information about potential collision. Another is that collection methods are based on a configuration space approach (J. Reif, 1987; C. Latombe, 1991; J. Schwartz, M. Sharir, J. Hopcroft, 1987). The configuration of rigid body is set of independent parameters that characterize the position of every point of it. For the vehicle  $B$  some regions represent illegal configuration space because there are obstacles. So the find path (the vehicle motion planning) approach means that the vehicle have to be shrank to dimension of a reference point and to grow obstacles, i.e. to compute forbidden regions for the reference point. Finding the path of the vehicle is in this way transformed in finding the path of the reference point, moving in configuration space and avoiding obstacles. For the vehicle  $B$  moving from position  $S$  to position  $G$  the desired path is the shortest path which takes into account all constraints of the position of the reference point of  $B$ . It is possible to obtain this path by generating an appropriate graph (visibility graph, connectivity graph,...) and finding a path from graph node  $S$  to graph node  $G$ . Fundamental problem arising during the implementation of these methods is concerned with the obstacle growing and graph searching. For both cases the problem complexity is very large. For example, while in the planar case the shortest path can be found in time that is in the worst case the quadratic in the number of obstacle vertices and edges, finding the shortest path between two points in three dimensions, which avoids a collections of polyhedral obstacles is NP - hard (J. Schwartz, M. Sharir, and J. Hopcroft, 1987; J. Reif, 1987). This is a specially very large problem if the world model changes. Other classes of approaches are developed as alternative to the traditional ones. A typical such approach (O. Khatib, 1986) regards the obstacles as the sources of repelling potential field, while the goal position  $G$  of the vehicle is considered as a strong attractor. The vehicle  $B$  follows potential gradient vector field. These approaches try to find the local minimum only. As the next, we can consider the direction which assumes problem solution capability of the vehicle motion planning based on the transfer of skill into controllers of the vehicle (D. Michie, R. Camacho, 1994; C. Sammut, S. Hurst, D. Kedzier, D. Michie, 1992).

## 2. Motion planning based on the behavioral cloning

Skill is often defined as an ability to perform a high quality sensory-motor coordination and control in real time. Humans exhibit such a skill as a result of training over a period of time. It would be especially useful if we can also provide systems with the capability of acquiring such a skill. In this sense two approaches have been known. The former treats the skill as something that could be acquired in a dialogue with an operator. In that process it is expected from the operator to describe skills he has been governed over the control of the vehicle. Here arises some difficulties because the skill is human subconscious action and so cannot be completely consciously and reliably described. An alternative approach is to start from the assumption that the skill can be reconstructed, using learning algorithms, from the manifestation trace of it (D. Michie, R. Camacho, 1994; C. Sammut, S. Hurst, D. Kedzier, D. Michie, 1992). Sammut, Hurst, Kedzier and Michie give a description of the solution belonging to the flight control area. Our idea (R. Kulic, Z. Vukic, 2006) is to enable an intelligent system to learn from the examples (operator's demonstrations) to control a vehicle avoiding obstacles, like the human operator does.

In section III the intelligent controller concept is given. In section IV the results in controller development are presented. In section V the conclusion is given and possibilities of further development are discussed.

## 3. Elements of concept controller development

### 3.1 Learning problem

Suppose that is given a data set giving living area and price of  $m$  houses in some place. How can it be possible to learn to predict the prices of other houses in that place, as a function of living areas? To denote the input variables or input features (living areas in this case)  $x_{1i}$  is used. And  $y_i$  is used to denote the output or target variable which is needed for training to predict. The dataset  $\{(x_i, y_i), i=1, \dots, m\}$  that will be used to learn is called a training set. If  $X$  denote the space input values and  $Y$  denote the space output values the goal is for given a training set to learn a function  $h: X \rightarrow Y$ . The function  $h(X)$  is called a hypothesis and it has to be a good predictor for the corresponding values of  $Y$ . When the target variable is continuous, the learning problem is called a regression problem. When the target variable takes on only a small number of discrete values, the learning problem is called a classification problem. Let's consider a slightly richer dataset with a number of bedrooms in each house. The  $X$  is two-dimensional vector, with  $x_{1i}, x_{2i}$  features. In general, when designing a learning problem, it is up to us to decide what features to choose. To perform learning it is necessary to decide how we are going to represent the hypothesis  $h$ . A choice can be to approximate it as a linear function of  $x = \{x_{1i}, x_{2i}, \dots, x_{ni}\}$ :

$$h(x) = \theta_0 + \theta_1 x_{1i} + \theta_2 x_{2i} + \dots + \theta_n x_{ni}$$

The  $\theta_i$ 's are the parameters (or weights) parameterizing the space of linear functions mapping from  $X$  to  $Y$ . To simplify notation it is introduced  $x_{i0}=1$ . So that

$$h(x) = \sum_{j=1}^n \theta_j x_{ji},$$

where  $n$  is the number of input variables, without  $x_0$ . For given training set, how it is possible learn the parameters  $\theta_j$ ? One reason is to make  $h(X)$  close to  $Y$ , at least for the training example. To formalize this, it can be defined a function that measures for each value of the  $\theta_j$  how close  $h(x_i)$  to the corresponding  $y_i$ . So, the least-squares cost function is defined:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h(x_i) - y_i)^2.$$

That function is a special case of a much broader family of algorithms. Now, it is needed to choose vector  $\theta$  to minimize  $J(\theta)$ . To do so, let us a search algorithm that starts with some initial guess for  $\theta$  and repeatedly changes  $\theta$  to make  $J(\theta)$  smaller, until it is reached the value of  $\theta$  that minimize  $J(\theta)$ . Let's consider the gradient descent algorithm, which starts with some initial  $\theta$ , and repeatedly perform the update:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

This update is simultaneously performed for all values of  $j=0, \dots, n$ . The learning rate  $\alpha$  repeatedly changes in the direction of steepest decrease of  $J(\theta)$ . For one training example we have:

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{n} (h(x) - y)^n = n \frac{1}{n} (h(x) - y) \frac{\partial}{\partial \theta_j} (h(x) - y) = \\ &= (h(x) - y) \frac{\partial}{\partial \theta_j} \left( \sum_{j=0}^n \theta_j x_{ji} \right) = (h(x) - y) x_j \end{aligned}$$

For single training example that gives the rule:

$$\theta_j = \theta_j + \alpha (y - h(x)) x_j.$$

This is called LMS (least mean squares) update rule or Widrow - Hoff learning rule. For instance the magnitude of the update is proportional to the error  $(y_i - h(x_i))$ . For  $m$  training example the corresponding update rule

Repeat until convergence exist {

$$\theta_j = \theta_j + \alpha \sum_{i=1}^m (y_j - h(x_j)) x_{ji} \text{ for every } j$$

}.

The optimization problem have been given here for linear regression has only one global optima. Assuming the learning rate not too large this batch gradient descent method always converges to the global minimum.

### 3.2. Bihevioral cloning process

The idea of controller development by cloning the human operator (D. Michie, R. Camacho,1994) is illustrated in Figure 1.

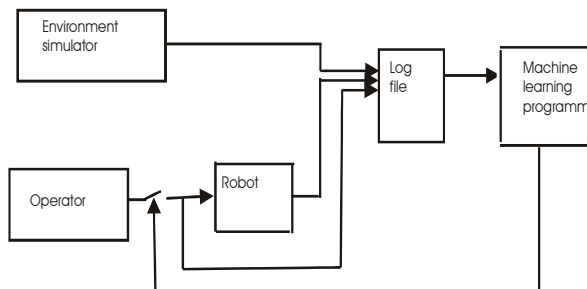


Figure 1. Behavioral cloning process. The autonomous underwater vehicle is named as the robot

In the obstacles avoiding problem, this idea could be interpreted in the following way. During one of the simulation phases, called the training phase, operator guides the vehicle avoiding unmoving obstacles located in its working space. During this phase variables that are evaluated as relevant are written into LOG FILE. In the second simulation phase, called the learning phase, the machine learning program, takes data from the LOG FILE, generates differential equations, that define the operator’s trajectory. In the third phase, called the verifying phase, operator is excluded from the vehicle control process and the vehicle is controlled solely by a clone induced in the learning phase. This development of process phases are needed for the repeated changing of both problem domain representation and/or learning system regarding cloning success criterion. Using “several” vehicle models (problem domain representations) and “several” machine learning systems, we attempt to find an appropriate domain model and an appropriate machine learning system that will enable the vehicle to avoid obstacles according to cloning success criterion.

### 3.3 The vehicle kinematical model

The following kinematical model of the vehicle is used:

$$\begin{aligned} \psi(n) &= \psi(n-1) + \Delta t r(n), \\ x(n) &= x(n-1) + \Delta t v \cos(\psi(n)), \\ y(n) &= y(n-1) + \Delta t v \sin(\psi(n)), \end{aligned}$$

where:  $\psi$  is the heading angle of the vehicle ( $\psi=0$  if the vehicle is oriented parallel to x-axis);  $r$  and  $v$  are control variables i.e. desired rotation speed and translation speed respectively;  $x, y$  are position coordinates,  $\Delta t$  is the sampling time and  $n$  is the time index. The vehicle is represented as a geometrical figure. Its dimensions are not neglected and we should point out that this is a very important fact. The selection of the vehicle model is inspired by conventional methodology that is used in control systems for a given path (R. Stojic, R. Kulic, M. Zivanovic, 1990).

### 3.4 Environment models

Environment model amounts to the distances of the vehicle gravity center from the goal position ( $dx_G$  and  $dy_G$ ) and from the obstacles ( $d_i$ ).  $dx_G$  and  $dy_G$  are calculated as:

$$dx_G = x - x_G,$$

$$dy_G = y - y_G,$$

where  $x_G, y_G$  are the goal position coordinates. Obstacles are represented by its characteristic values, as illustrated in Figure 2. Obstacle area is divided into sub-areas.

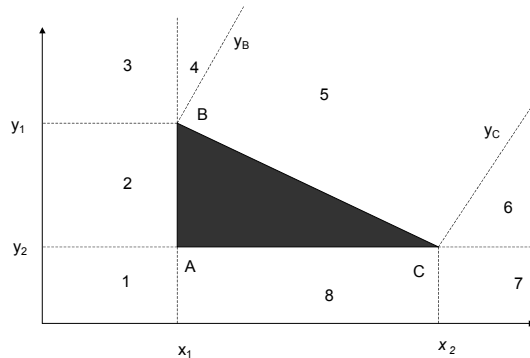


Figure 2. Triangular obstacle as the environment

A procedure, for the simulation purpose, calculating the vehicle distance  $d_i$  from  $i$ -th obstacle is explained for a triangle obstacle as:

SubArea-4:

*if*(( $x \geq x_1$ )*and*( $y \geq y_B$ )) *then*  $d_i = ((x-x_1)^2 + (y-y_1)^2)^{1/2}$ , SubArea5:

*if*(( $x \geq x_1$ )*and*( $y < y_B$ )*and*(( $y \geq y_C$ )) *then*

$$d_i = |fy + [(y_1 - y_2) / (x_2 - x_1)]x + [(y_2 - y_1) / (x_2 - x_1)]x_1 - y_1| / \sqrt{[(y_1 - y_2) / (x_2 - x_1)]^2 + 1}^2|,$$

$y_B$  and  $y_C$  are lines that are normal onto the line BC at points B and C.

### 3.5 Cloning success criterion

Performance error is very important for the evaluation of the quality of clone that was constructed. Regarding the ideal case the goal concept and the approximation concept of the vehicle trajectory are identical and the performance error is equal to zero. Ideal trajectory in  $x$ - $y$  plane without obstacles is, for example, a straight line between the start S and the goal G positions of the vehicle. Operator, in a training phase, mostly does not manage to realize this trajectory. Position error  $E_{xy}$  is based upon a distances  $d_{op(i)}$  and  $d_{cl(i)}$  of operator and clone trajectories, respectively, from the named straight line. Our problem is to avoid obstacle and so we can consider only  $E_{perf}$  as:

$$E_{xy} = \frac{\sum_{j=1}^N \frac{|d_{op}(j) - d_{cl}(j)|}{\max(d_{op}(j) - d_{cl}(j))}}{N} \tag{1}$$

For avoiding  $n$  obstacles we have to find  $(d)_m = \min\{d_i, i=1,..n\}$  in order to define:

$$E_{xy} = \frac{\sum_{j=1}^N \frac{|(d_{cl}(j))_m - (d_{cl}(j-1))_m|}{\max(d_{cl}(j))_m - (d_{cl}(j-1))_m}}{N-1} \quad (2).$$

Eventually, we can say that between two clones more successful is the one which produces lower performance error regarding equations (1) and (2).

### 3.6 The vehicle goal position as the obstacle

The vehicle goal position can be treated as the obstacle, but it has not been applied earlier in [12], [13]. In that sense the distance from the vehicle goal position and the vehicle gravity centre was taken into account, as  $d_{goal}$ . Now the attribute number is extended and "minimum distance" is determined by relation:  $d_{min} = \min \{ d_{goal}, d_1, d_2, \dots, d_k \}$ , where  $d_1, d_2, \dots, d_k$  are the minimal distance for avoiding K obstacles. Eventually, when we consider the vehicle goal position as the obstacle, we enable the vehicle path generation to become simple, but not to become optimal.

### 3.7 Learning systems

#### 3.7.1 Radial basis function (RBF) of neural network

The model is commonly referred to as the radial basis function (RBF) network. The most important attribute that distinguishes the RBF network from earlier radial based models is its adaptive nature. It generally allows to utilize a relative small number of locally tuned units. RBF network were independently proposed by several authors (D. S. Broomhead, D. Lowe, 1988; S. Lee, R. Kill, 1988; M. Niranjan, F. Fallside, 1988). The following is a description of the basic RBF architecture Figure 3. The RBF network has a feedforward structure consisting of a single hidden layer of Q locally tuned units, which are interconnected to an output layer of L linear units. All hidden units simultaneously receive R dimensional real-valued input vector p. Notice the absence of hidden layer weights. Each hidden unit output  $a_j$  is obtained by calculating the closeness of the input p to n dimensional parameter vector  $\mu_j$  (IW in Figure 2). This parameter is associated with jth hidden units. The response characteristics of the jth hidden units are given by

$$a_j[p] = K\left(-\frac{\|p - \mu_j\|}{2\sigma_j^2}\right),$$

where K is a strictly positive radially symmetric function (kernel) with a unique maximum at its center  $\mu_j$  and which drops off rapidly to zero away from the center. The parameter  $\sigma_j$  is the width of the receptive field of the input space for unit j. This means that  $a_j$  has an appreciable value only when distance  $\|p - \mu_j\|$  is smaller than the width  $\sigma_j$ . A specially but commonly used RBF network assumes a Gaussian basis function for the hidden units, i.e.:

$$a_j[p] = \exp\left(-\frac{\|p - \mu_j\|^2}{2\sigma_j^2}\right),$$

where  $\sigma_j$  and  $\mu_j$ , are the standard deviation and mean of the  $j$ th unit. The norm is the Euclidian norm. The output of the RBF network is the  $L$  dimensional vector  $a_2$ , which is given by:

$$a_2 = \sum_{j=1}^L LW_{ij} a_j .$$

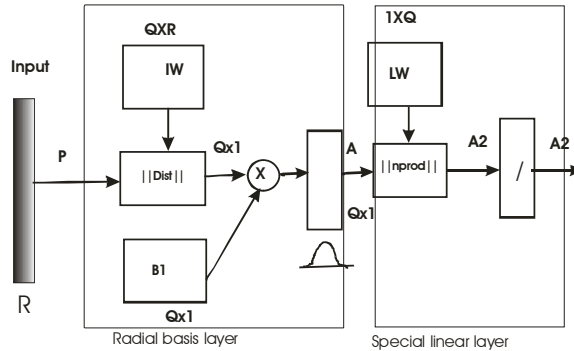


Figure 3. A radial basis function neural network consisting of a single hidden layer of locally tuned units that is fully interconnected to an output layer of linear units

RBF networks are best suited for approximating continuous real valued mappings  $f : R^n \rightarrow R^l$ , where  $n$  is sufficiently small. According to the previously named equations the RBF network may be as approximating a desired uncton  $f(p)$ . The degree of accuracy can be controlled by three parameters the number of basis functions to be used, their location and their width. The RBF networks are considered as universal approximators (T. Poggio, F. Girosi, 1990). The training of RBF network is addressed. Consider training set of  $m$  labeled pairs  $\{x_j, y_j\}$  which are represent samples of a continuous multivariate function. The criterion function is an error function  $E$  to be minimized over the given training set. It is desired to develop a training method that minimizes  $E$  by updating the free parameters of the RBF. These parameters are  $\sigma_j, \mu_j$  and  $w_{ij}$ . One of the first training methods that comes to mind is a fully supervised gradient descent methods over  $E$ , as it is given in section 3.1.

**3.7.2 RBF neural network algorithm and behavioral cloning**

The algorithm RBF neural network algorithm is given below for kinematical model which is given in 3.3.

```

for m:=1 to N do                                     % N is number of training examples
  begin
    dist[m,1] := (0.8326 / spread)2 * (IW[m,1] - d_min)2;    % d_min = d_min ;
    a[m,1] := e-dist[m,1];                                     % IW[m,1] is the training examples set of the distance d_min
  end;
for j:=1 to N do
  begin
    a2 := a2 + LW[1,j] * a[j,1];
    am := am + a[j,1];
  end;                                               % LW[m,1] is the training examples set of the angle ψ

```



*if ((am > 0) or (am < 0)) then a2 := a2/am;*  
*psides := a2;*

*r := (psides - psi) / dt;*

$$r = \frac{\psi_{des} - \psi}{\Delta t}, \quad dt = \Delta t = 0.05 \text{ s}$$

Control strategy was  $r = (\psi_{des} - \psi) / \Delta t$ ,  $v=0.1$ , where  $\psi_{des}$  is the desired value of angle  $\psi$ .

### 3.7.3 Modification Kohonen rule and behavioral cloning - MKBC algorithm

Thus, as it is given in Figure 3, the neuron whose weight vector was closest to the input vector is updated to become even more closer. Suppose that the  $i^{th}$  neuron wins, then the elements of the  $i^{th}$  row of the input weighting matrix are adjusted as shown:

$${}_iLW^{1,1}(q) = {}_iLW^{1,1}(q-1) + \alpha(p(q) - {}_iLW^{1,1}(q-1)),$$

where  $p(q)$  is input vector,  $q$  is the time index and  $\alpha$  must be specially tuned according to the domain. With regarding to the original Kohonen rule connected with the weighting coefficients, as the neural network, *this algorithm does not use the weighting values as the values from the previous time, but permanently uses the training values as the weighting values. This enables an intelligent system to learn from the examples (operator's demonstrations) to control a vehicle in avoiding obstacles, like the human operator does.* This seems, at least for this case, to be better according to the original idea of the Kohonen rule. The algorithm is given below.

```

for i:=1 to M do
begin
AS(1,1) := 0;
end;
alfa(i,n);           % Must be specially tuned
                    % according to the domain;
                    % n is the time index;
for i:=1 to M do    % M is the number of inputs;
begin
for j:=1 to N do   % N is the number of the
                    % training examples;
begin
LW[i,j,n] := alfa[i,n] * ( p[i,n0] - p[i,n]); % p[i,n] is the input
                    % vector;
                    % p(i,n0) are the training example set
AS(i,n) := AS(i,n) + ( LW[i,j,n0] - LW(i,j,n))^2; % LW[i,j,n0] is the training examples set
end;
a(i,n) := sqrt (AS(i,n));           % a(i,n) is the output
                    % vector;
end;
end;

```

The algorithm means that by using  $N$  training examples the weight vector  $LW [i,j,n0]$ ,  $\{i=1,M; j=1,N\}$  can be formed<sup>1</sup>. Then  $LW[i,j,n0]$  need to be modified according to: 1) the actual values of the input vector<sup>2</sup>  $p[i,n]$ ,  $\{i=1,M\}$  and 2) the actual value of the tuned vector  $\alpha[i,n]$ ,  $\{i=1,M\}$ , where  $n$  is the time index. The resulting weighting vector is  $LW[i,j,n]$  and it has deviation according to the input vector  $p[i,n]$ . The output vector is  $a[i,n]$  and it is connected with the weight vector as is illustrated by two last lines of the algorithm.

### 3.8 The MKBC algorithm and the motion planning

The MKBC algorithm is needed to be adapted in order to be used in the motion planning domain. The motion planning with the *weight vector* uses  $LW$  (Table 1), the training values of the learned variable which is the heading angle  $\psi$  (Section 3.3). Firstly, the algorithm requires the modification of the training vector  $LW$ . The modification executes, as is given below, using: 1)  $N$  training examples in  $IW$  and  $LW$ , where  $IW$  are the training values of the minimal distance  $d_{min}$  (section 3.6), 2) the current input value  $d_{min}$  and 3) tuning factor  $\alpha$ . Secondly, the algorithm finds the desired output value using, in some sense, least – square cost function of the heading angle. The algorithm output value enables the obstacle avoidance and it is the desired value of the heading angle ( $\psi_{des}$ ). The algorithm is given below.

$AS := 0;$

$alfa = K_c - K_a * d_{min};$  % The values  $K_c$  and  $K_a$  must be tuned;

for  $j := 1$  to  $N$  do %  $N$  is the number of the training examples;  
begin

$LW[2,j] := alfa * (IW [1,j] - d_{min});$

$AS := AS + (LW[2,j] - LW[1,j])^2;$

end;

$psides := sqrt(AS);$  %  $\psi_{des} = \sqrt{AS};$

%  $r := (psides - \psi) / dt;$  ,  $dt = \Delta t = 0.05$  s;

On the other hand, when the modified Kohonen rule with the weighting parameters is used, then the algorithm does not use the weighting values  $IW$  as the distance  $d_{min}$  from the previous time, but always uses  $IW$  as the training values of the distance  $d_{min}$  to determine the training values  $LW$  of the heading angle  $\psi$ . This enables an intelligent system to learn from the examples (operator's demonstrations) to control a vehicle in avoiding obstacles, like the human operator does. The very important MKBC characteristics are the operator cloning and simplicity, the simplicity specially according to the RBF neural network. The coefficient  $\alpha$  has values that are changed from time to time. Firstly, when it is tuned this factor enables the heading angle  $\psi$  to increase when the vehicle distance from the obstacles edges is small (i.e.  $d_{min} \approx 0.08$ ). The performance error  $E_{xy}$  has a minimum for that  $\alpha$  value. While  $\alpha$  factor increases, the total time  $T$  of the autonomous vehicle moving without touching the obstacle also increases.

<sup>1</sup>  $n0$  means the training example set.

<sup>2</sup> The training value of the input vector is  $p[i,n0]$ .

### 4. Experiments and results

In order to form training instances the task was only to avoid obstacles. The robot start position was its goal position. The minimal number of robot traveling from the start to the goal was to be one. It is the framework for selecting appropriate training instances. The idea is to combine this control strategy with control strategy without obstacles in order to form a full controller.

When an obstacle was included, the robot trajectory for training scene is illustrated in Figure 4.

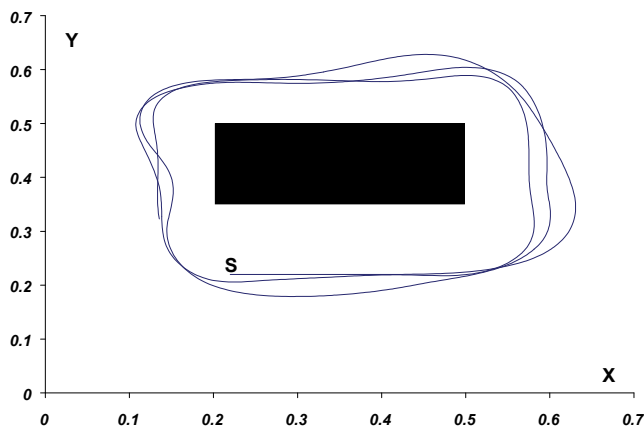


Figure 4. The robot trajectory used in order to gather training example

Control strategy was  $r = (\Psi_{desired} - \Psi) / \Delta t, v=0.1$ .

#### 4.1 Experiments with RBF neural network

In the following is described application of RBF network in mobile robot motion planning. A relatively small corresponding training set is given in Table 1.

dex J	Distance or $d_{min}$	Angle $\Psi$ or LW[1,j]	Angle modified $\Psi_m$ or LW[1,j]	Factor $f_m$
1	0.04	10.906	10.906	1
2	0.0494	13.8025	13.8025	1
3	0.0591	11.425	11.425	1
4	0.08	0.0035	81.025	23151
5	0.08	9.513	85.617	9
6	0.081	12.070	12.070	1
7	0.0894	12.692	12.692	1
8	0.0923	12.556	12.556	1
9	0.1006	7.6405	7.6405	1
10	0.12	0.6895	3.4475	5

Table 1. The original end the modified training examples according to the movement of the vehicle

The firstly by tuning spread factor (section 3.7.1 and 3.7.2) is tuned  $\sigma_j$ . For step  $k=1$  spread =1, and then spread decreased in 0.01. Process was stopped for spread=0.002. The performance error  $E_{xy}$  has minima for that spread value. While spread factor decreased, total time  $T$  of the mobile robot moving without touching of obstacles increased: (spread=0.005,  $T=534$  s; spread=0.003,  $T=600$  s; spread=0.002,  $T=637$  s). In order to decreased performance error for small the robot distance from obstacles edges ( $d_{min} \approx 0.08$ ),  $LW[1,j]$  is multiplied by  $f_m [j]$  as it is given in Table 1 and in Figure 5.

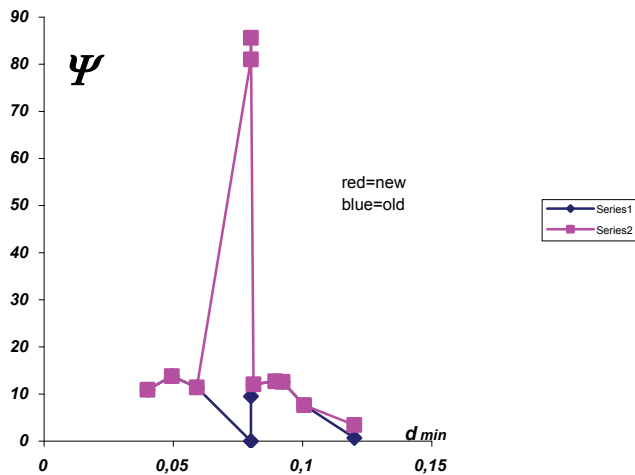
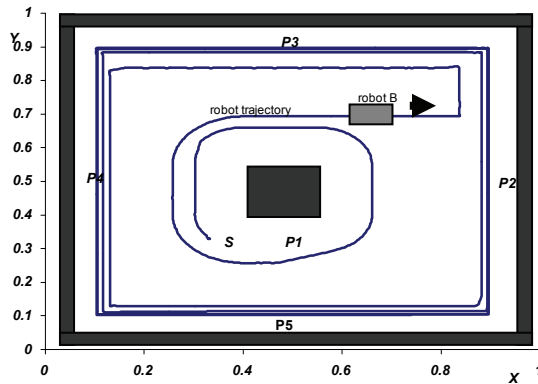
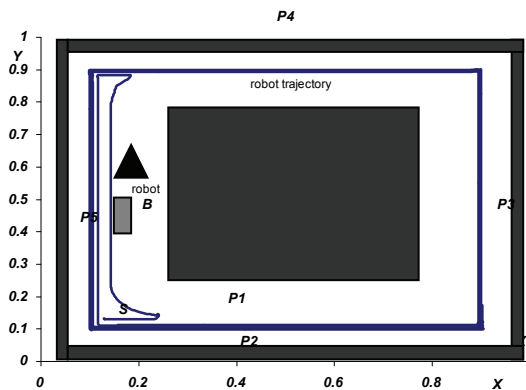


Figure 5. Modifying the  $LW_{ij}$  parameters or  $\Psi$  training instances for the RBF network in order to decrease performance error  $E_{xy}$

For  $f_m [4]=23151$ ,  $f_m[5]=6$  and another  $\{f_m [j]=1, j=1,2,3,6,7,8,9,10\}$ , it was  $T=1248$  s. But for  $f_m [j]$  which have values as it was given in Table 1 performance error was  $E_{xy} = 0.00382$  and robot moving was not time limited. Two scenes with five static obstacles is given in Figure 6. In that case we have situation when the mobile robot moves away from P1 obstacle to be close, until some critical distance and without touching, to obstacles P2,P3,P4 and P5. The robot safely avoids the obstacles touching. Changing  $f_m[j]$  repeatedly takes a step in the direction of steepest decrease of  $E_{xy}$ , like the cases described for function J in section 3.1. But some changes are inappropriate. Accordingly generally speaking we wished, but we could not manage, to synthesize the controller to guide the mobile robot to “oscillate” between obstacle P1 and set of obstacles P2,P3,P4 and P5. Exists the tendency of the clone to guide the robot in such a fashion that its distance from obstacle edges enhance gradually. The problem is when the controller solves turning to the left, i.e. it have to conclude that the robot distance, for example, from vertices of obstacle P4 is greater than from the robot distance from vertices of obstacle P5. Until that the robot avoids obstacle P4 attempting to turn, not to the left, but to the right. It is reason that the robot more and more is closed to obstacles P2, P3, P4 and P5. For small distance of the robot from obstacle vertices we attempt to increase reaction time when turns around a square.



a)



b)

Figure 6. The mobile robot trajectory in xy plane for five static obstacles . The mobile robot is controlled by RBF neural network controller

#### 4.2 MKBC algorithm as advance according to RBF neural network algorithm

The training instances Table 1 (selected 10 from 1801) the previously were tuned using RBF neural network (Section 4.1), i.e. an appropriate  $f_m$  factor was selected. In order to form training instances the task was to avoid rectangle obstacle and achieve the minimal number of one vehicle travels from the start to the goal position. On the other hand, the goal is the number of instances to be as small as possible. The MKBC algorithm (section 3.8) was tested as it is illustrated by Figure 7 for  $K_c = 1000$  and  $K_a = 50$  and

$$r = \frac{3.1623 * \psi_{des} - 0.9 * \psi}{\Delta t} \quad \text{control strategy, for } \Delta t = 0.05.$$

The vehicle start position and obstacles are given by relations:

Start  $S = (0.15, 0.15)$ ,

Position 1  $P1 = (0.3, 0.3, 0.7, 0.7)$ ,

Position 2  $P2 = (0.0, 0.95, 1.0, 1.0)$ ,

Position 3  $P3 = (0.95, 0.0, 1.0, 1.0)$ ,

Position 4  $P4 = (0.0, 0.0, 1.0, 0.05)$ ,

Position 5  $P5 = (0., 0.0, 0.05, 1.0)$ , for  $v = 0.2 \text{ [m/s]}$ .

For training instances given in Table 1 and for  $f_m [4] = 23151$ ,  $f_m [5] = 6$  and another  $\{f_m [i] = 1, j = 1, 2, 3, 6, 7, 8, 9, 10\}$ , as it is given in Table 1 the simulation time  $T$  using the exposed algorithm was greater then about 6000 [s].  $LW[1, i]$  is multiplied by  $f_m [i]$  as is given in Table 1. For  $f_m [i]$  with values given in Table 1 performance error was  $E_{xy} = 0.00180$  and the vehicle movement was time unlimited. So we take that  $f_m$  is exactly as it is given in Table 1. In this case we have situation when the autonomous vehicle moves away from  $P1$  obstacle to be closed, until some critical distance and without touching obstacles  $P2, P3, P4$  and  $P5$ . Here, the tendency of the clone exists to guide the vehicle in such a fashion that its distance from the obstacle edges enhance gradually. The problem appears when the controller solves the turn to the left. Namely, it should conclude that the vehicle distance, for example, from vertices of obstacle  $P4$  is greater than from the vehicle distance from vertices of obstacle  $P5$ . The vehicle avoids obstacle  $P4$  attempting to turn, not to the left, but instead to the right. This is the main reason why the vehicle is closer and closer to obstacles  $P2, P3, P4$  and  $P5$ . Changing  $f_m [j]$  repeatedly takes a step in the direction of steepest decrease of  $E_{xy}$ , like in the cases described early. Like that, if we make the possibility that the coefficient  $\alpha$  changes according to the  $d_{min}$  changing, we will going to induce a strougly encahement with regard to the simulation time which is the touch free.

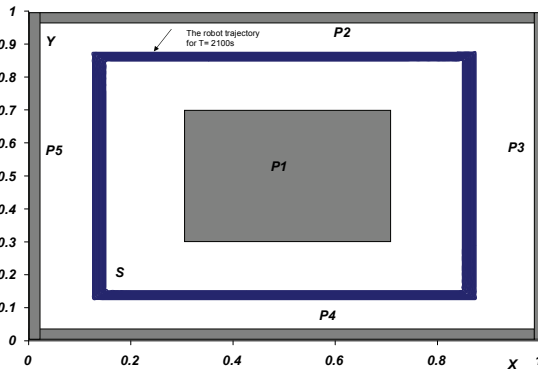
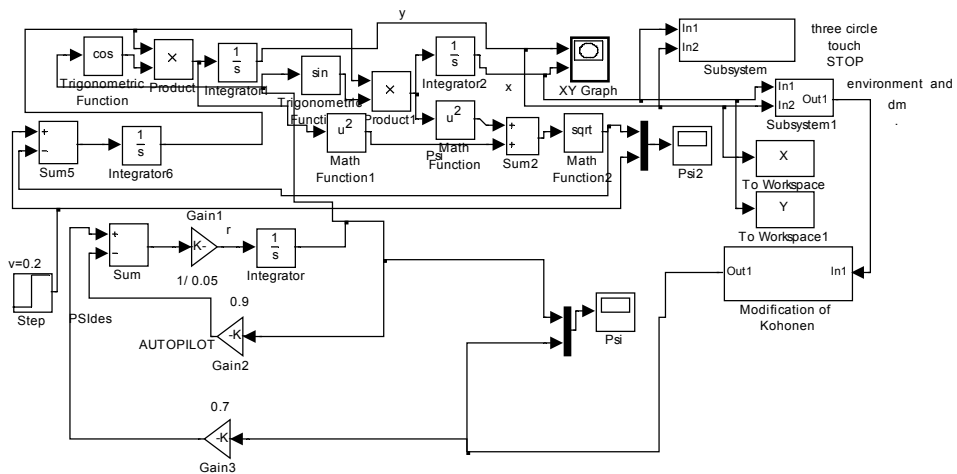
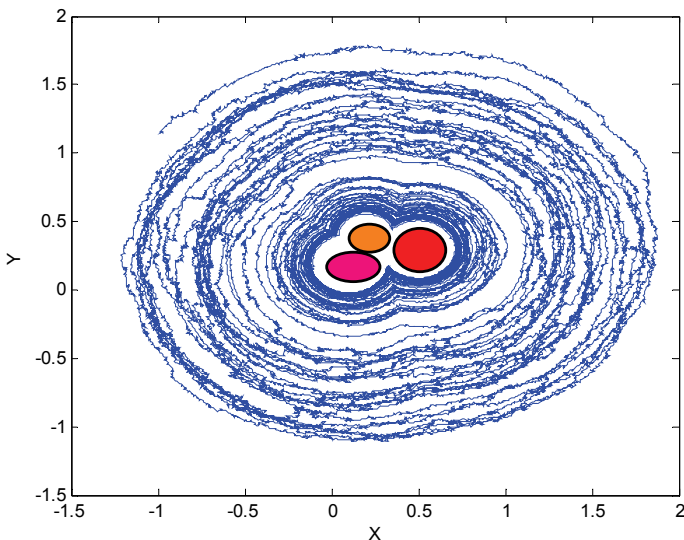


Figure 7. XY trajectory of the vehicle controlled by clone based on modified Kohonen rule

For  $K_c = 1000$  and  $K_a = 50$  and for the named control strategy we have practically infinite the time which is touch of free. The simplicity and that is reason to accept the MKBC algorithm as an advancement according to the RBF neural network. At the end, Simulink implementation MKBC for the vehicle kinematical model is very simple and it is illustrated in Figure 8, but according to Matlab integration method constants  $K_{cte}$ ,  $K_a$  and  $K_{psidesired}$  must be corrected as  $K_{cte} = 500, K_a = 100, K_{psidesired} = 0.7$ .



a) Simulink model



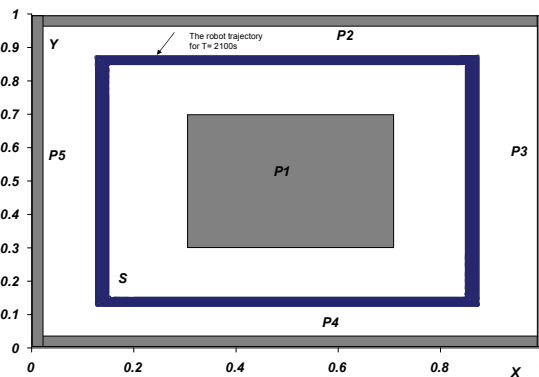
b) 2D trajectory for three obstacles when  $K_{ct}=500, K_a=100$

Figure 8. Simulink model for MKBC algorithm and an appropriate result

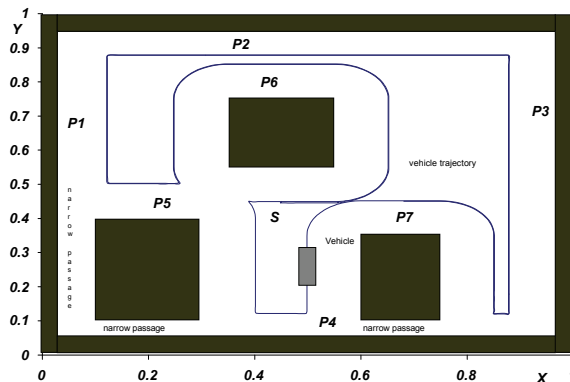
### 4.3 The narrow passage avoiding by MKBC algorithm

In literature (R. Kulić, 2004) is treated the narrow passage avoiding. In order to avoid the robot traveling through narrow passages between obstacles is needed to define a rule regarding dimension of the autonomous vehicle. In this section MKBC algorithm, as an advancement according to the RBF, is used. That algorithm is applied in order to solve the situation which is given in Figure 9b) -9h). In different situations which are illustrated by

Figure 9b) – 9h) the vehicle moves from different start positions to different goal positions avoiding obstacles and narrow passages also. All experiments are quite successful. The narrow passages are avoided using the reason which is described the previously. In Figure 9b), for example, the autonomous vehicle moves away from P6 obstacle to be close P7, until some critical distance, and so on, to be close until some critical distance from obstacles P4, P3, P2, P1, ..., and always changes  $\Psi$  rapidly in order to be safely avoided the obstacle touch for long time. Changes  $\Psi$  rapidly is connected with the changing  $f_m[j]$  also rapidly as it illustrated in Figure 5. It is the manner repeatedly takes a step in the direction of steepest decrease of  $E_{xy}$ . According to the different situations the vehicle managed to avoid obstacles again and again, and never did not have the chance to stop, i.e. that process is has not time limiting and we observed that all experiments are quite successful. It was found that the obstacle avoiding and also the narrow passage avoiding is enabled by the same MKBC algorithm or by the methodology which is the previously described and it means that the solution consistency is saved.

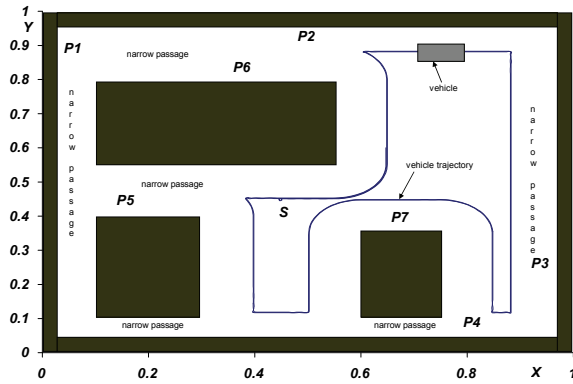


a)

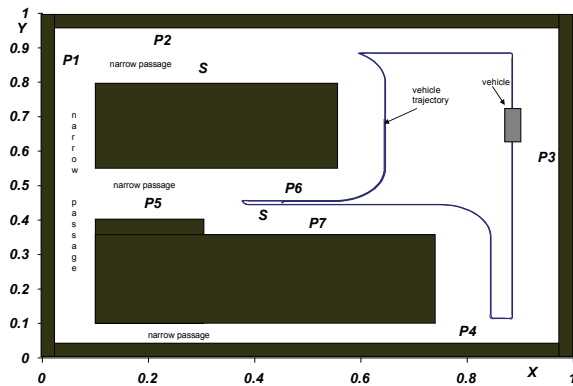


b)

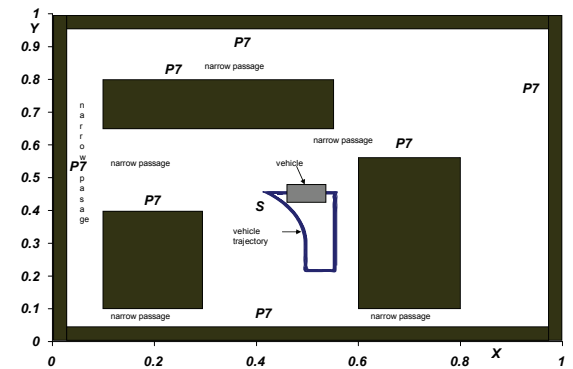




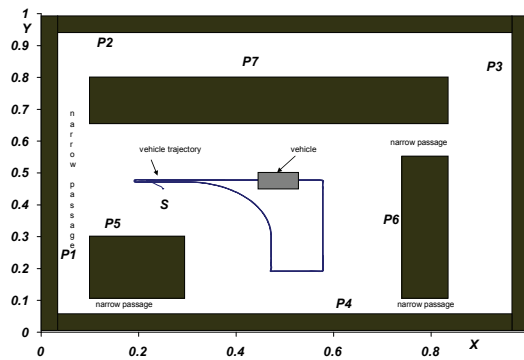
c)



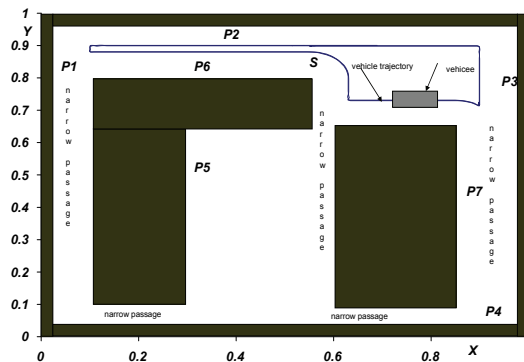
d)



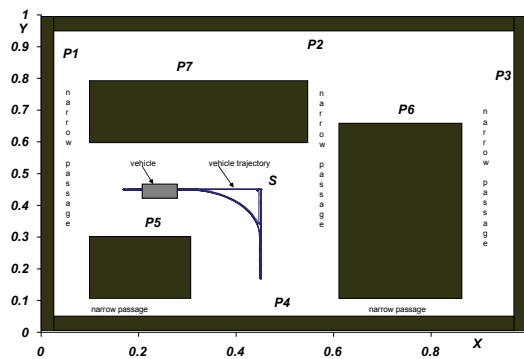
e)



f)



g)



h)

Figure 9. The autonomous vehicle trajectory in xy plane for 5 and for 7 static obstacles

The previously, in literature (R. Kulić, 2006), it is concluded, according to the given machine learning system, to avoid the vehicle traveling through the narrow passages

between obstacles it is needed to define a rule regarding their dimensions and now it could be added that rules can be escaped by changing in structure of the given learning system in sense that it is to be able to change  $\Psi$  rapidly.

#### 4.4 The goal position reaching

We interested in the goal reaching as it is given in (R. Kulic, 2004) and many disadvantages of if - then rules are named according to this problem. Situation is more and more difficult when the number of obstacles are increasing.

##### 4.4.1 The virtual obstacle based on line connecting the goal and the start position

In the next experiments the line which connect the goal and the start position serves to be formed a virtual obstacle as it is illustrated in Fig. 10. According to the Fig. 10 it is possible to form relations (3 - 8):

$$\begin{aligned} X1 &= XS + 0.05, Y1 = YS; \\ X2 &= XS, Y2 = YS + 0.05; \\ X3 &= XG - 0.05, Y3 = YG; \\ X4 &= XG, Y4 = YG - 0.05; \end{aligned} \tag{3}^3$$

$$YP1 = Y1 + \frac{Y2 - Y1}{X2 - X1}(X - X1) \tag{4}$$

$$YP2 = Y2 + \frac{Y3 - Y2}{X3 - X2}(X - X2) \tag{5}$$

$$YP3 = Y3 + \frac{Y4 - Y3}{X4 - X3}(X - X3) \tag{6}$$

$$YP4 = Y1 + \frac{Y4 - Y1}{X4 - X1}(X - X1) \tag{7}$$

It is known that the normal distance of the point M(X,Y) from the line  $aX+bY+c=0$  is:

$$d = \frac{|aX + bY + c|}{\sqrt{a^2 + b^2}} \tag{8}$$

A procedure, for the simulation purpose, calculating the vehicle distance from the virtual obstacle is given as: SubArea-1: *if*(( $Y \leq YP1$ )*and*( $Y \leq YP2$ )) *then*  $d_v = ((X-X1)^2 + (Y-Y1)^2)^{1/2}$ , SubArea2: *if*(( $Y \geq YP4$ )*and*( $Y < YP2$ )*and*( $Y \leq YP3$ )) *then*  $d_v = |Y + [(Y1-Y2)/(X2-X1)] X + [(Y2-Y1)/(X2-X1)] X1 - Y1| / [((Y1-Y2)/(X2-X1))^2 + 1]^{1/2}, \dots$  and so on for all of from the

<sup>3</sup> 0.05 \*2 means that the virtual obstacle is not only the line but has dimensions which is randomly determined. Equations (3) - (8) define the line which is connected with two determined points.

eight areas. This approach enables the vehicle moving from the start to the goal position expanding distance set by defining:  $(d)_m = \min\{d_v, d_i, i=1,..n\}$ , using only the equation according to the obstacle avoiding.

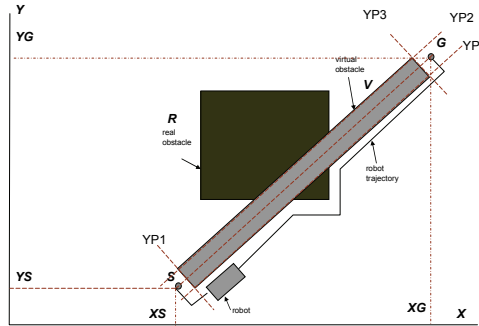


Figure 10. The virtual obstacle based on the start - the goal line

**4.4.2 Experiments using the virtual obstacle**

As the first, the methodology which is exposed in section 4.1 is tested using GoldHorn machine learning system and (A. Karalic,1991) and equation (9)

$$\psi_{desired} = \frac{-d_m + 0.13735}{0.0064} \tag{9}$$

and control strategy:  $r = (\psi_{desired} - \psi) / \Delta t, v=0.1$ . The result is given in Figure 11.

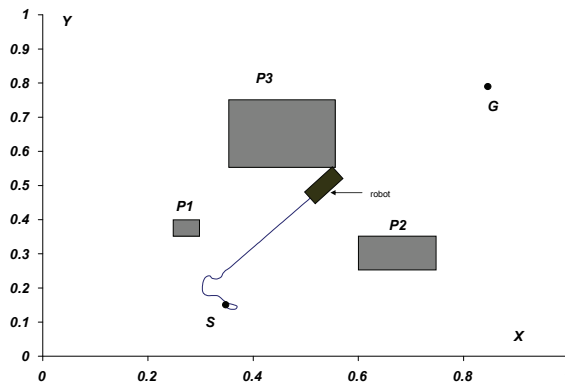


Figure 11. The virtual obstacle based on the start - the goal line and GoldHorn clone

The virtual obstacle is treated and according to all of obstacles obviously the touch is detected because reaction time of GoldHorn clone is inappropriate, as it was observed the previously.

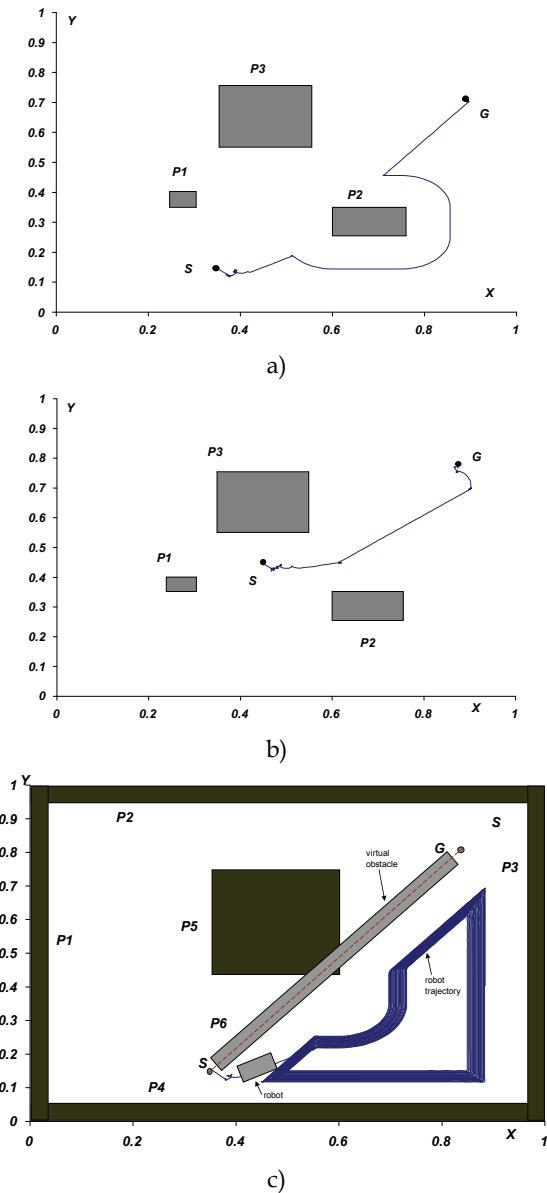


Figure 12. The autonomous vehicle trajectory in xy plane for three and for five static obstacles. The autonomous vehicle is controlled by neural network clone

A neural network MKBC algorithm as an advancement according to the GoldHorn machine learning system (A. Karlič, 1991) and to the RBF is used. MKBC algorithm is applied in order to solve the situation which is given in Fig. 12a), b), c). In situation illustrated by Fig. 12a) for three obstacles the vehicle moves from the start position  $S(0.35, 0.15)$  to the goal

position  $G(0.85, 0.8)$  avoiding the virtual obstacle also. The experiment is successful. Like that, the experiment is successful in Fig. 12b), where the virtual obstacle is treated again and where the vehicle reaction time is quite appropriate. In Fig. 12c) is illustrated situation with five static obstacles P1 – P5, and virtual obstacle P6 which is illustrated also. After a great number of obstacles avoiding circles the vehicle attempts to move to the goal G, but some oscillations appear and the vehicle decides to avoid obstacles and reach the goal again and again, and has not the chance to stop. That process has not been time limited, but it means that experiment is quite successful: we have obstacle avoiding and the goal position reaching using the same algorithm, i.e. the same relations (R. Kulić, 2004).

**4.5 Results end experiments in 3D space**

**4.5.1 Kinematikal 3D model**

We have applied behavioral cloning and machine learning and got some results, e.g., equations, which were suspected: this results could not suit us if we had dynamical model or if we had real time dynamical system. Generally we expected, equations could be wrong in some sense, but the methodology had to be an appropriate. It is illustrated in the next section. But for some class of trajectories vehicle moving can be decomposed on moving in the vertical and moving in the horizontal plain (R. Stojić, et al., 1990), and for some class of trajectories we can take that  $\psi_{desired}(n)=\theta_{desired}(n)$ . In that sense the next 3D kinematical model of the autonomous vehicle is observed:

$$\begin{aligned} \psi(n) &= \psi(n-1) + \Delta t r(n), \\ \theta(n) &= \theta(n-1) + \Delta t q(n), \\ x(n) &= x(n-1) + \Delta t v \cos(\psi(n)), \\ y(n) &= y(n-1) + \Delta t v \sin(\psi(n)), \\ z(n) &= z(n-1) - \Delta t v \sin(\theta(n)), \end{aligned}$$

where  $\theta$  is pitch angle,  $q$  is the appropriate control variable in vertical plane. To avoid obstacle in 3D the next control rule is observed, also:

**if** ( $n = even$ ) **then**  $r(n) = r(n-1)$  **else**  $r(n) = (\psi_{desired}(n) - \psi(n)) / \Delta t$ ;  
**if** ( $n = odd$ ) **then**  $q(n) = q(n-1)$  **else**  $q(n) = (\theta_{desired}(n) - \theta(n)) / \Delta t$ .

A simulated scene is illustrated by Figure 13, where the obstacles were avoided successfully.

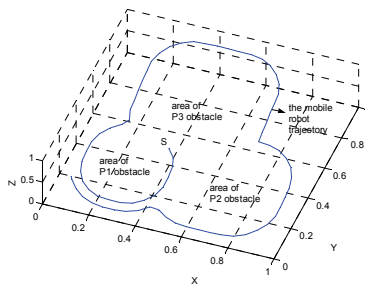


Figure 13. The autonomous vehicle 3D trajectory

It can be concluded that the proposed method gives results that can easily be applied in 3D space.

#### 4.5.2 The complete 3D mathematical model of the vehicle

In literature (G. Campa, M. Innocenti, 1999) is presented development of a mathematical model of an vehicle and its translation under Matlab environment. At the same time the report presents and explain both the equations and the Matlab code that implements those equations. The used approach is independent on which kind of vehicle is modeled, on which environment the vehicle moves (air or water). It means that the report could be easily taken as a reference for a wide class of vehicle modeling problems. In the final chapters several examples on how to use the given nonlinear model for simulation, analysis and control synthesis purposes are given. In Figure 14 is rigid body in 6 DOF with two coordinate frames: body fixed referenced frame and earth fixed reference frame.

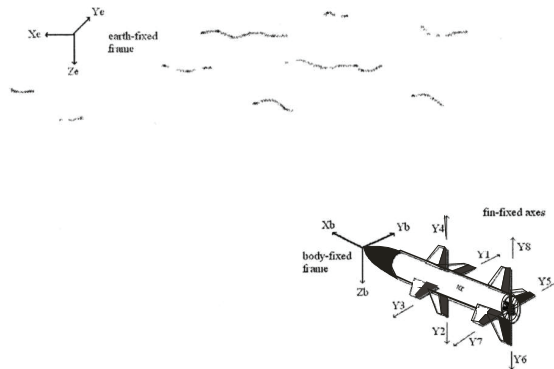


Figure 14. The observed vehicle

The considered vehicle has structure a cylindrical body, an aerodynamic prow and two sets of cross -configured mobile fins, numbered as in Figure 14. Each fin has its own fixed frame in order to rotate about its y -axis. The rotation angle is in counter clockwise. The 4 fins belonging to the front set (numbered from 1 to 4) refer as "wings", while the others 4 fins (numbered from 5 to 8) refer as "tails". Matlab function *vxdot* is the Matlab code equivalent of equation (62) (G. Campa, M. Innocenti , 1999) at equation describe a 12 state dynamic system. It takes as input a 38 - dimensional vector containing: 1) the 12 system states (generalized position and velocity ); 2)fin angles vector , 8 elements; 3) external force and moment with respect to the body frame, (e. g. thrusters), 6 elements; 5) external force and torque with respect to the earth frame ( e. g. attached cable or other force due to the contact with external objects), 6 elements; 6) marine current speed and acceleration with respect to the earth frame, 6 elements. Matlab *vxdot* function uses the named input and the information about vehicle structure ( G. Campa, M. Innocenti,1999) stored in special global variable "veh" ) to compute derivatives of the system states as described in (62) . The named nonlinear model is linearized about the operating point determined by  $[x,y,z]=[0,0,0]$ ,  $[u,v,w]=[3m/s, 0, 0]$ , the thrust is 416N, there is no sea current and there is no other external forces.

**4.5.1. The classical regulator design**

The Linear Quadratic Regulator is one of the regulators which is given in (G. Campa, M. Innocenti, 1999). Its appropriate scheme is given in Figure 7.

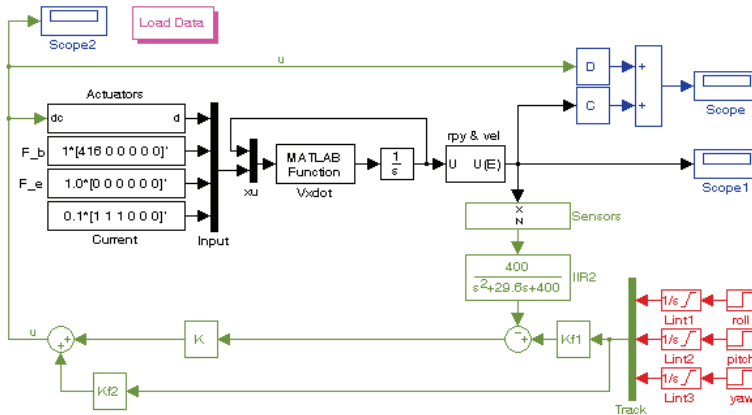


Figure 15. LQR closed loop scheme

Authors found the named regulator as a very good choice for testing the closed loop behavior of the used nonlinear model and treated it as the best in term of both performance and robustness. Having in the workspace the matrices a,b,c,d of the linear model , the feedback regulator is given in Matlab by:

```
>> Q=diag ([20 20 20 ones(1,6)]); R=eye(size(b,2));
```

That command line gives Q and R weight matrices meaning that the attitude states are weighted a value of 20 compared to the value 1 of the remaining states and inputs. Matrix K is given as:

```
>> K=lqr(a,b,Q,R);
```

The nonlinear model starts from the same input and initial conditions the previously considered for the linearization. As disturbance, a slow and constant sea current is considered, and some noise is added to each available measurements. To increase of realism of the simulation a model of the actuators with delays, position limiter and rate limiter is included as it is given by Figure 16.

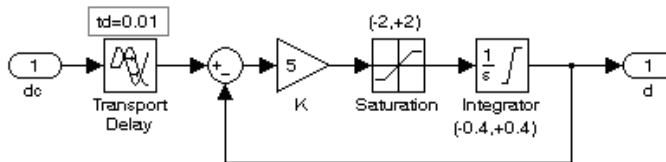


Figure 16. Actuators model



A sensors model is given in Figure 17.

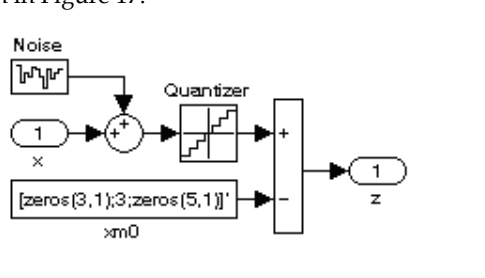


Figure 17. Sensors model

The command to be tracked consists in 3 independent steps of angle 0.1 rad and maximum amplitude 0.3 rad. The inputs or channels: yaw, pitch and roll are appear with time delay from 5s. Sample time is 0.1s. These inputs are used in experiments described in next section.

#### 4.5.2 To avoid obstacles regulator design

As existed our good experience according learning algorithm GoldHorn (R. Kulic, Z. Vukic 2006), in training phase we decided to use it again, i.e. use it clone based on equation

$$\psi_{desired} = -0.08005d_m - 1.54286 \tag{10}$$

$$\theta_{desired} = -0.08005d_m - 1.54286 \tag{11}$$

which GoldHorn managed to find. In order to form training instances system in Figure 3 is changed adding Simulink subsystems containing: 1) GoldHorn clone as operator, 2) virtual simulation environments, 3) the fashion of calculating<sup>4</sup>  $d_m$ , 4) the fashion of detecting touch between the vehicle and the obstacles and 5) the fashion of gathering the training instances in the Matlab workspace. Part of that model is given in Figure 18.

Situation with the vehicle start position  $S[0,0,0]$  and three unmoving sphere obstacles:  $O1[40,10,0,30]$ ,  $O2[30,40,20,20]$ ,  $O3[10,20,40,20]$ <sup>5</sup> is observed. In the training phase the vehicle is controlled by GoldHorn clone in order to avoid sphere obstacle from 0s to 500s. Sample time was 0.1s. The clone did not manage to realize the trajectory which mean that the vehicle start position is equal the vehicle goal position. It was difficult: the previously operator has not been a skilled worker and GoldHorn machine learning controller was similar to him. From 5122 examples were selected 10, i.e. 500<sup>th</sup>, 1000<sup>th</sup>, 1500<sup>th</sup>, ..., 5000<sup>th</sup>, as it is given in Table 2. It is important to expose that using training instances and Matlab polyfit instruction we get some unsatisfactory results.

<sup>4</sup>  $d_i$  is distance of the vehicle gravity center from  $i^{th}$  obstacle and  $d_m$  is defined as:

$d_m = \min(d_i, i=1,n)$ , where  $n$  is number of obstacles.

<sup>5</sup> Sphere is given as:  $O(a,b,c,R)$ , where  $a,b,c$  are coordinates of center and  $R$  is the radius.

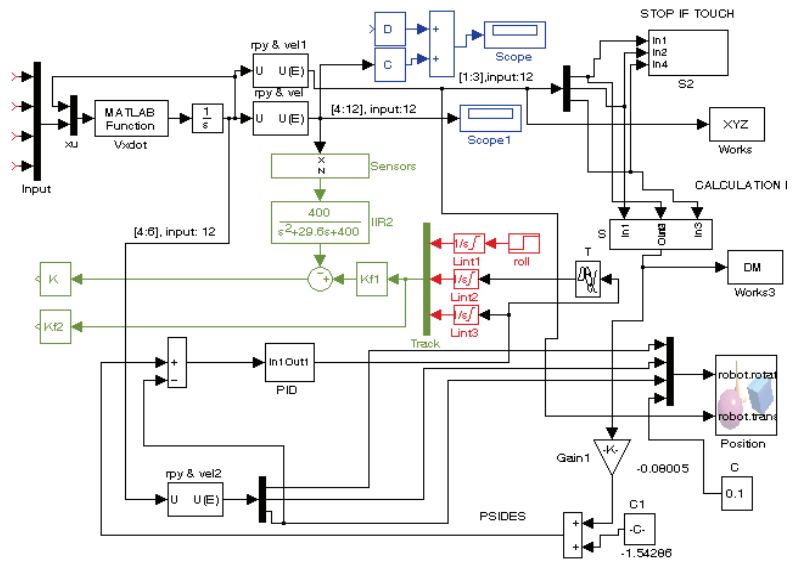


Figure 18. The part of Simulink model connected with the training phase

No.	$\Psi$ Angle	$d_m$ distance
1	-0.8162	2.1189
2	-1.5639	7.1298
3	-2.4173	19.0685
4	-3.4051	27.7691
5	-4.2736	28.9622
6	-4.6729	20.8607
7	-4.359	12.796
8	-3.7437	17.9315
9	-3.463	29.6801
10	-13.3025	150.2197

Table 2. The training example for 3D nonlinear dynamical model

### 4.5.3 The experiments and results

In verifying phase it is needed to connect  $\psi_{\text{desired}}$  with  $\psi$  and  $\theta$  inputs as it is illustrated in Figure 7. But it was not enough. Namely, it was observed that is needed to be synthesized additional regulator regarding obstacle avoiding process and to be formed the closed loop system according  $\psi$  and  $\theta$ , or according minimal distance  $d_m$  between the vehicle and obstacles. To require that kind of correction is a consequence of the fact that the learner outputs the hypothesis that has the minimum error over the training examples, but has the error (T. Mitchell, 1997). So, a PID regulator was added and its founded constants are:  $k_p=0.005$ ,  $k_d=0.0009$ ,  $k_i=0.0001$ . According to MKBC algorithm given in Section 3.8 tuned constants are:  $k_a=0.00005$ ,  $k_{cte}=-0.1$ . Between  $\theta$  and  $\psi$  time delay was  $T_\theta=1s$  and roll angle was  $\varphi=0.1rad$ . Maximum amplitude was: for  $\theta$   $i \varphi$  (-0.3, 0.3) rad., and for  $\psi$   $(-\infty, +\infty)$ rad. Part of Simulink model connected with verifying phase is illustrated by Figure 19. Situation with the vehicle start position  $S[0,0,0]$  and three unmoving sphere obstacles, with different positions and dimensions according to the training phase, are observed:  $O1[20,30,40,10]$ ,  $O2[30,40,50,20]$ ,  $O3[40,10,350,10]$ . The results are illustrated by Figure 20. As illustrated in figure 20b) the tendency of the regulator to guide the vehicle in such a fashion that its distance from obstacle edges increases was not observed. That fact is a good reason for the conclusion that these results, with sophisticated kinematical and dynamical model of the vehicle, quite confirm the results exposed the result the previously exposed when, when was used only kinematical model of the vehicle.

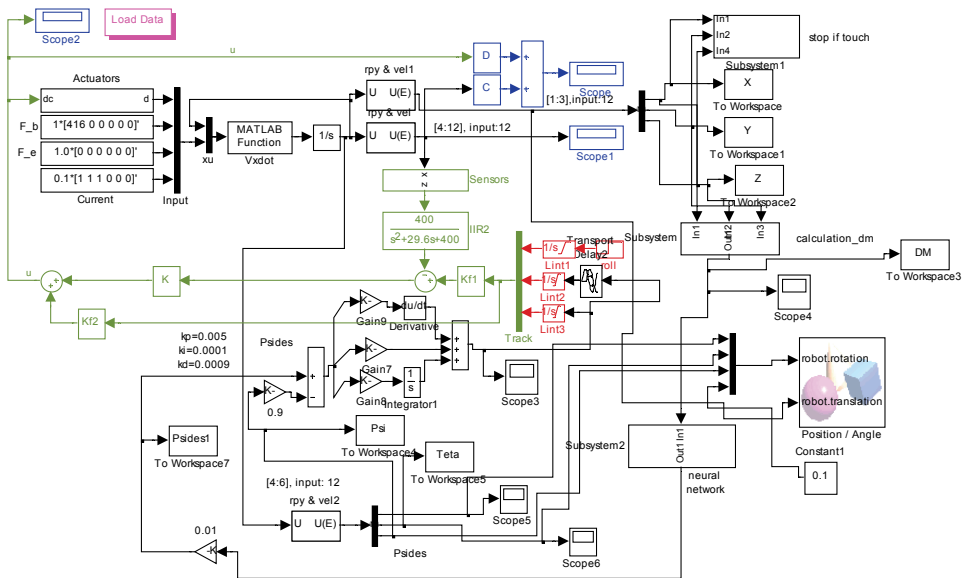


Figure 19. The part of Simulink model connected with the verifying phase

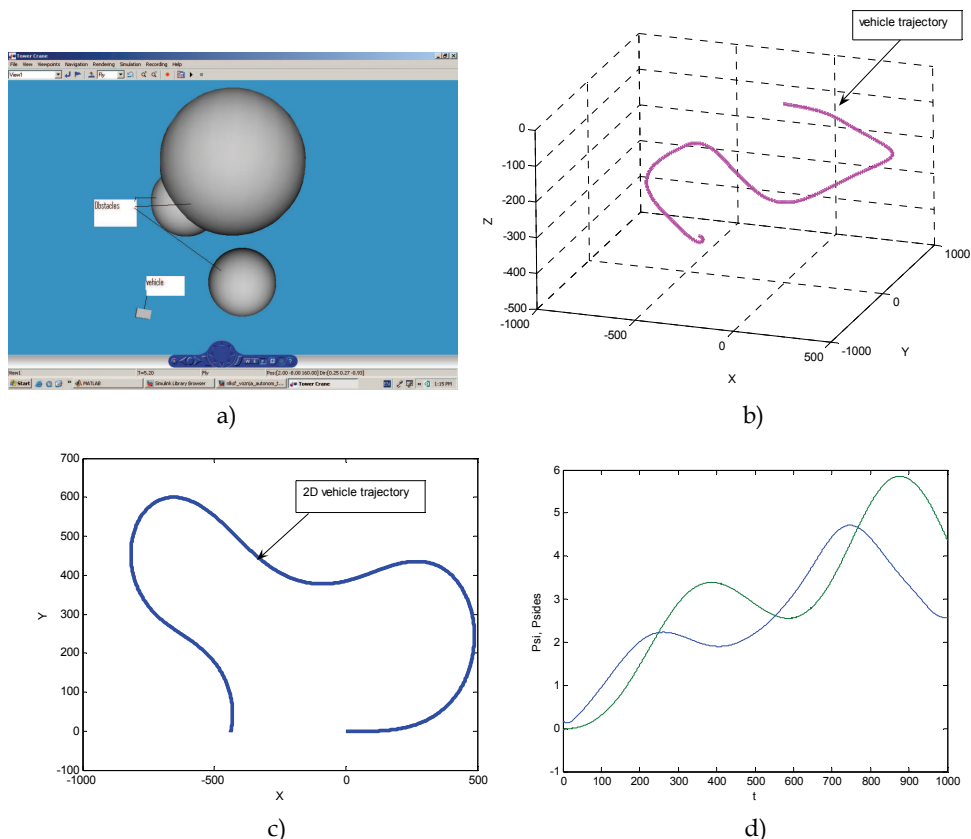


Figure 20. The simulation results of the autonomous vehicle moving and the obstacles avoiding

## 5. Conclusion

As it is exposed the obtained MKBC clone is tuned to produce the results that are more appropriate than those given in (R. Kulic, Z. Vukic, 2006). This is especially important when the vehicle turns around the square. The very important MKBC characteristics are the operator cloning and simplicity, the simplicity specially according to the RBF neural network. The reaction time substantially decreases for small distance of the vehicle from the obstacle vertices and edges according to the RBF neural network. According to the different situations the vehicle managed to avoid obstacles again and again, and never did not have the chance to stop, i.e. that process is has not time limiting in 2D space specially and we observed that all experiments are quite successful. Following the given context the problem narrow passage avoiding and the goal position reaching fundamentally is observed. Namely, defining if - then rule according to the problem is observed as destroying of the consistency of the reached methodology. MKBC algorithm gives chance that the obstacle avoiding, the narrow passage avoiding and goal position

reaching is enabled by the same algorithm or by the methodology which is the previously described and it means that the solution consistency is saved. The algorithm complexity is about  $O(n)$ , where  $n$  is the number of obstacles. At the end, the autonomous vehicle mathematical model is given by nonlinear equations describing a 12 state dynamical system and in that case the exposed methodology is applied successfully. The advantage of this approach lies in the fact that a complete path can be defined off-line, without using sophisticated symbolical models of obstacles. By this fact the named methodology substantially differs from the others. In the next phase it is expected to confirm results in on-line simulation process.

## 6. References

- J. C. Latombe, *Vehicle Motion Planning*, Kluwer Academic Publishers, Boston 1[1]991.
- J. Reif, Complexity of the generalized mover's problem. Editors J. Schwartz, M. Sharir, and J. Hopcroft: *Planning, Geometry, and Complexity of Vehicle Motion*, Ablex, Norwood, Nj, 1987. [2]
- J. Schwartz, M. Sharir, and J. Hopcroft, *Planning, Geometry, and Complexity of Vehicle Motion*, Ablex, Norwood, Nj, 1987. [3]
- J. Schwartz, M. Sharir, On the piano mover's problem: I. The case of twodimensional rigid polygonal body moving amidst polygonal barriers, Editors J. Schwartz, M. Sharir, and J. Hopcroft: *Planning, Geometry, and Complexity of Vehicle Motion*, Ablex, Norwood, Nj, 1987. [4]
- O. Khatib, Real - time obstacle avoidance for manipulators and autonomous vehicles, *International Journal of Vehicle Research*, 5, 1986. [5]
- D. Michie, R. Camacho, Building symbolic representation of intuitive real - time skills from performance data . In: K. Furukawa, S. Muggleton (eds.) *Machine Intelligence and Inductive learning*, Oxford University Press, Oxford, 1994. [6]
- C. Sammut, S.Hurst, D. Kedzier, D. Michie, Learning to Fly, D. Sleeman, P. Edwards, (eds), *Proceeding 9<sup>st</sup> International Workshop on Machine Learning* , Morgan Kaufmann, 1992. [7]
- D. Michie, Knowledge, learning and Machine Intelligence In:L.S. Sterling (ed), *Intelligent System*, Plenum Press, New York 1993; [8]
- R. Kulić, Z. Vukić, Methodology of concept control synthesis to avoid unmoving and moving obstacles (II), (*accepted, April 11, 2006; Online published June 14,2006* ) *Journal of Intelligent and Robotic Systems*, Publisher Springer Netherlands, ISSN 0921-0296 (paper) 1573-0409 (Online), DOI: 10.1007/s 10846-006-9035-7, Issue: Volume 45, Number 3, Date:March 2006, Pages: 267 - 294. [9]
- R. Kulić, Trajectory design for the autonomous vehicle by operator cloning, PhD thesis, Faculty of Electrical Engineering and Computing, Zagreb, 2004. (in Croatian) [10]
- A. Karalič, Regression trees learning from noise data, *Master thesis*, Univerzity of Ljubljana, Ljubljana, 1991. (in Slovenian); [11]
- V. Križman, The consideration of noise data using automatic modeling of dynamical systems, *Master thesis*, Univerzity of Ljubljana, Ljubljana, 1993. (in Slovenian). [12]
- L. Breiman, J.H. Friedman, R.A. Olshen, C.J Stone, *Classification and Regression Trees*, Wadsworth Int. Group, Belmont, California, USA, 1984; [13]
- C, A. Michelli, Interpolation of scattered data: Distance and conditionally positive definite functions, *Constructive Approximation*, 2, 11-22, 1986. [14]

- M. J.D. Powell, Radial basis functions for multivariate interpolation: A review, in *Algorithms for the Approximation of Functions and Data*, J. C. Mason and M. G. Cox, eds. Clarendon Press, Oxford, England, 1987. [15]
- R. Stojić, R. Kulić, M. Živanović, Flight control for the given trajectory, *Science-Technical observer*, Vol. XI, No. 8-9, Belgrade, 1990 (in Serbian). [16]
- M..H. Hassoun, *Foundamental of Artificial Neural Networks*, MIT Press, 1995. [17]
- D. S. Broomhead, D. Lowe, Multivariate functional interpolation and adaptive networks, *Complex Systems*, 2, 321-355, 1988. [18]
- S. Lee, R. Kill, Multilayer feedforward potential function networks, *Proceedings of the IEEE Second International Conference on Neural Networks*, San Diego 1988, vol. I,, pp. 161-171, IEEE, New York. [19]
- M. Niranjan , F. Fallside, Neural Networks and Radial Basis Functions in Classifying static Speech Patterns, *Technical Report CUEDIF -INFENG17R22*, Engineering Department, Cambridge University, 1988. [20]
- T. Poggio, F. Girosi, Networks for approximation and learning, *Proceeding of the IEEE*, 78(9), 1481-1497, 1990. [21]
- E. J. Hartman, et al., Layered neural networks with Gaussian hidden units as universal approximators, *Neural Computation*, 2(2), 210-215, 1990. [22]
- G. Campa, M. Innocenti, *Model of an underwater vehicle*, University of Pisa October 1999. [23]
- P. Vadakkepat, X. PengQ. B. Kiat, L. T. Heng, Evolution of fuzzy behaviors for multi - robotic system, *Robotics and Autonomous System*, vol 55(2). February 2007, pp 146 - 161. [24]
- T. Mitchell, *Machine Learning*, McGraw - Hill Companies, Inc., 1997. [25]



## **Motion Planning**

Edited by Xing-Jian Jing

ISBN 978-953-7619-01-5

Hard cover, 598 pages

**Publisher** InTech

**Published online** 01, June, 2008

**Published in print edition** June, 2008

In this book, new results or developments from different research backgrounds and application fields are put together to provide a wide and useful viewpoint on these headed research problems mentioned above, focused on the motion planning problem of mobile ro-bots. These results cover a large range of the problems that are frequently encountered in the motion planning of mobile robots both in theoretical methods and practical applications including obstacle avoidance methods, navigation and localization techniques, environmental modelling or map building methods, and vision signal processing etc. Different methods such as potential fields, reactive behaviours, neural-fuzzy based methods, motion control methods and so on are studied. Through this book and its references, the reader will definitely be able to get a thorough overview on the current research results for this specific topic in robotics. The book is intended for the readers who are interested and active in the field of robotics and especially for those who want to study and develop their own methods in motion/path planning or control for an intelligent robotic system.

### **How to reference**

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ranka Kulic (2008). Modification of Kohonen Rule for Vehicle Path Planing by Behavioral Cloning, Motion Planning, Xing-Jian Jing (Ed.), ISBN: 978-953-7619-01-5, InTech, Available from:  
[http://www.intechopen.com/books/motion\\_planning/modification\\_of\\_kohonen\\_rule\\_for\\_vehicle\\_path\\_planing\\_by\\_behavioral\\_cloning](http://www.intechopen.com/books/motion_planning/modification_of_kohonen_rule_for_vehicle_path_planing_by_behavioral_cloning)

**INTECH**  
open science | open minds

### **InTech Europe**

University Campus STeP Ri  
Slavka Krautzeka 83/A  
51000 Rijeka, Croatia  
Phone: +385 (51) 770 447  
Fax: +385 (51) 686 166  
[www.intechopen.com](http://www.intechopen.com)

### **InTech China**

Unit 405, Office Block, Hotel Equatorial Shanghai  
No.65, Yan An Road (West), Shanghai, 200040, China  
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元  
Phone: +86-21-62489820  
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](#), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.