International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:8, No:5, 2014

Neural Network in Fixed Time for Collision Detection between Two Convex Polyhedra

M. Khouil, N. Saber, M. Mestari

***Abstract*—**In this paper, a different architecture of a collision detection neural network (DCNN) is developed. This network, which has been particularly reviewed, has enabled us to solve with a new approach the problem of collision detection between two convex polyhedra in a fixed time (O (1) time). We used two types of neurons, linear and threshold logic, which simplified the actual implementation of all the networks proposed. The study of the collision detection is divided into two sections, the collision between a point and a polyhedron and then the collision between two convex polyhedra. The aim of this research is to determine through the AMAXNET network a mini maximum point in a fixed time, which allows us to detect the presence of a potential collision.

***Keywords*—**Collision identification, fixed time, convex polyhedra, neural network, AMAXNET.

I. Introduction

THE identification of a collision between two objects in space represents one of the main areas of interest in many fields of computer graphics, robotics, CAC (conception assisted by computer) and DAC (design assisted by computer) systems. This identification is done through various softwares that can model, simulate and plan the movement of objects in the presence of obstacles. Our study will focus on that.

Collision detection is the most difficult part to implement because we face many optimization problems that depend on various constraints (space and time). However, the most expensive part in the planning of a path is the calculation of the number of collision constraints for each iteration.

There are several efficient algorithms to identify a collision and calculate an appropriate distance. They represent a major research in this field ([2], [9], [12]).

As pointed out in [9], the Euclidean distance is the most natural way to measure a collision between two objects. The calculation of the Euclidean distance between two objects is part of the field of computational geometry [10]. For problems in two dimensions (2-D), the time of calculation is asymptotically O (log^2M) by Schwartz algorithm [11]. Wang and Chin [6] and Edelsbrunner [8] have reported more efficient algorithms. When polyhedra involve three dimension (3-D), the algorithm by Dobkin and Kirkpatrick [7] require O (M) operations.

In this study, a neural network for collision detection (DCNN) is designated to identify collisions between two convex polyhedra in a fixed time (O (1) time). The AMAXNET network is used for the generation of minimum and maximum distances, which are used as collision indicators. In this network, we will use only two neurons (a linear and threshold logic neurons) to minimize costs and facilitate the implementation.

This paper is organized as follows; the first part will be devoted to the presentation of artificial neural networks used. In the second part we will study the collision detection between a point and a polyhedron and then collision detection between two polyhedra. Finally we will explain the construction of the network DCNN and then conclude with a summary of all the topics discussed in this study.

M. Khouil is with the Laboratoire SSDIA, ENSET, Mohammedia Maroc (phone: +212-670-733-193; e-mail: meryem.khouil@gmail.com).
N. Saber is with the Laboratoire SSDIA, ENSET, Mohammedia Maroc (phone: +212-664-848-356; e-mail: saber.nadia.89@gmail.com).
M. Mestari is with the Laboratoire SSDIA, ENSET, Mohammedia Maroc (phone: +212-662-669-308; e-mail: mestari@enset-media.ac.ma).

II. Artificial Neural Networks Used

A. Neurons Used

The DCNN (Detection Collision Neural Network) and AMAXNET networks use two types of neurons, which are commonly used in the work of Fukushima [13], Kohonen [14] Mestari [4] Mestari et al [5] Mestari [1], Mestari [3]. The only difference between these two types of neurons is their activation function; one employs a linear activation function, and the other a logic threshold activation function.

These two types of neurons sum the weighted n entered and transmits the result according to the following (1):

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i - \Theta\right) \quad (1)$$

where ϕ is called activation function, Θ ($\Theta \in \mathbb{R}$) is the threshold or external bias, w_i are weights or synaptic forces, x_i are inputs (i = 1, 2, ..., n), n the number of inputs and y is the output.

The linear activation function is defined as follows:

$$\phi_L(x) \stackrel{\text{def}}{=} x \quad (2)$$

The threshold logic activation function:

$$\phi_T(x) \stackrel{\text{def}}{=} \begin{cases} 1 & if\ x \geq 0 \\ 0 & Otherwise \end{cases} \quad (3)$$

B. AMAXNET Network

A neural network AMAXNET, which with an array of real numbers as inputs, outputs the k^{th} largest element of the array proposed. Overall processing time is constant, and does not depend upon the size of the input array, being just six times the processing time for a single neuron. Two definitions, which are essential for construction of AMAXNET, are introduced.

- Definition 1: The order in the input array X=$(x_1, x_2, \dots, x_n)$ of any element x_i $(1 \le i \le n)$ is defined as being the number of elements of the input array less than x_i .
- Definition 2: Let $x_i$and x_q be two elements of the same value. If (i<q), then $x_i$is considered smaller than x_q, otherwise x_i is considered larger than x_q.

AMAXNET principle may be summarized thus: Overall, AMAXNET is composed of n order networks, $ON_i(1 \le i \le n)$, and a selection network SN. Also, AMAXNET is equipped with a control input called an adjustment input. This adjustment input allows choice of the order of the element to be transferred to output.

The function of the order networks $ON_i(1 \le i \le n)$ is to compute the order of each element x_i in the input array. The selection network's function, however, is to select and transfer to output the element of the input array corresponding to the order desired or chosen via the adjustment input.

1) Order Network

Let x_i and $x_q$be two elements of the input array X. The comparison function of x_i and $x_q$is defined as follows:

For i$\in \{1, \dots., q-1\}$,

$$comp(x_i, x_q) \stackrel{def}{\equiv} \begin{cases} 1 & if\ x_q \ge x_i \\ 0 & Otherwise \end{cases}$$

$$\stackrel{def}{\equiv} \phi_T(x_q - x_i) \quad (4)$$

For $i \in \{q+1, \dots, n\}$,

$$comp(x_i, x_q) \stackrel{def}{\equiv} \begin{cases} 1 & if\ x_q > x_i \\ 0 & Otherwise \end{cases}$$

$$\stackrel{def}{\equiv} \phi_T(-\phi_T(x_i - x_q) \quad (5)$$

The order of any element x_q in the input array X may be given by the following order function.

$$ord(x_q, X) \stackrel{def}{\equiv} \sum_{i<q} comp(x_i, x_q) + \sum_{i>q} comp(x_i, x_q) + 1$$
$$\stackrel{def}{\equiv} \sum_{i<q} \phi_T(x_q - x_i) + \sum_{i>q} \phi_T(-\phi_T(x_i - x_q) + 1 \quad (6)$$

The network for computing the order function is shown in Fig. 1 this network is denoted as ON_q.

2) Construction of Selection Network

The function of the selection network is to select from among the elements of input array X the element corresponding to the order fixed by the adjustment input and to transfer it to output. This network is composed of n equality sub-networks, ESN, followed by a linear neuron.

a) Equality Sub-Network

The equality sub-network ESN determines whether the order of an element is equal or not to a given number K. The number K (1$\le K \le n$) is fixed via the adjustment input.

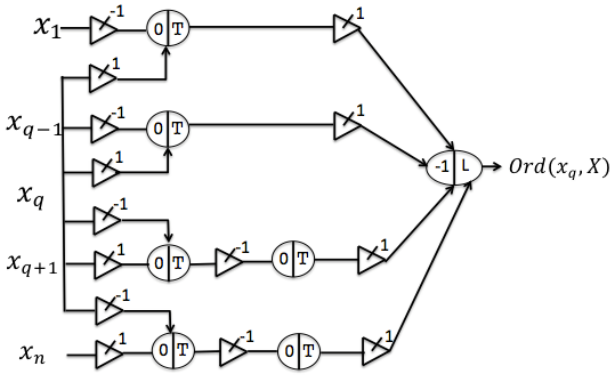

Fig. 1 The Order Network

The function computed by the equality sub-network ESN is defined as follows:

For all $x_i$and 1$\le K \le n$,

$$eq[ord(x_i, X), K] \stackrel{def}{\equiv} \begin{cases} 1\ if\ Ord(x_i, X) = K \\ 0 \qquad Otherwise \end{cases} \quad (7)$$

$$eq[ord(x_i, X), K] = \phi_T\big(-\phi_T(ord(x_i, X) - n - 1) - \phi_T(n - ord(x_i, X) - 1)\big) \quad (8)$$

where $\phi_T$is defined in (3).

The equality network EN consists of three threshold logic neurons as shown by the diagram in Fig. 2.

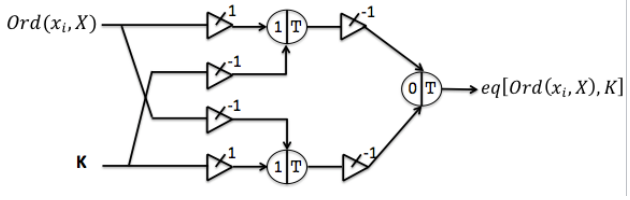

Fig. 2 Equality sub-network ESN

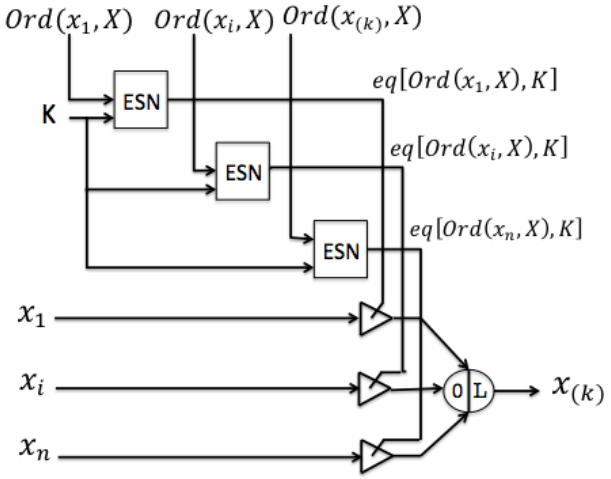

Fig. 3 Selection Network SN

b) Selection Network

The selection network is shown in Fig. 3 and is denoted as SN, it detects and transfers only the appropriate element whose Order equals K to output.

$x_{(K)}$is the K^{th} largest element of the input array X:

$$x_{(K)} = \sum_{i=1}^{n} x_i . eq[Ord(x_i, X), K] \quad (9)$$

AMAXNET is shown in Fig. 4, where the adjustment input determines which order statistic is to appear at the output. The network illustrated in Fig. 4 consists of two kinds of neurons arranged in six layers.

The number of neurons in AMAXNET for input size n is $\frac{3}{2}n^2 + \frac{5}{2}n + 1$.

There are six layers of neurons in AMAXNET, thus the total processing time is six times the processing time of a single neuron. As the number of elements of the input array increases, only the number of neurons in each layer increases, not the number of layers themselves. Therefore, AMAXNET's total processing time remains constant irrespective of the number of elements in the input array.

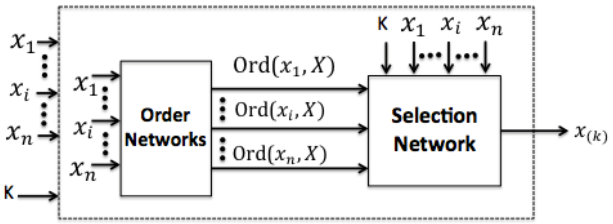

Fig. 4 The adjustable MAXNET, AMAXNET

III. Point-To-Polyhedron Collision Detection

This section will be devoted to the collision detection between a point and a polyhedron.

In this research we are interested in a class of objects which can be approximated by convex polyhedra, the objects in question may have the shape of digital images (composed of thousands of pixels), material objects (vehicles, Robots ...) or virtual objects...

To better view the illustrations on convex polyhedra, it is easier to consider convex polygons, taking into account that an arbitrary transverse plane of a convex polyhedron is a convex polygon.

All the illustrations put into this paper on convex polyhedra are drawn as convex polygons.

A convex polyhedron is the intersection of n half planes bounded by corresponding field.

We assign to each of the n planes, which limit the polyhedron, a Cartesian coordinate system (Fig. 5).

The x-y coordinate system of a plane corresponds to the boundary; the z-axis is oriented in the direction normal to the surface (toward the outside of the object).

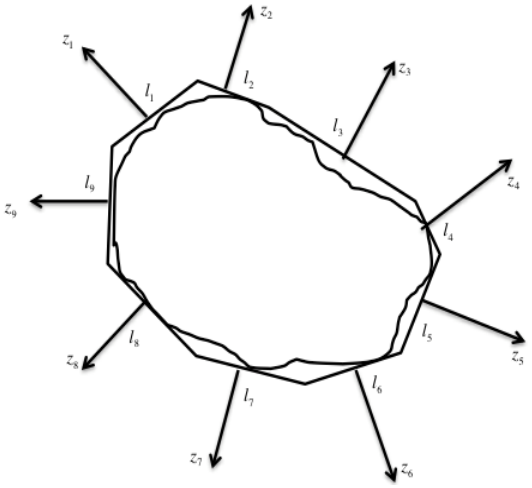

Fig. 5 Polygon bounded by N surfaces

Any arbitrary point is represented as a 3-D vector as:

$$P=[p_x, p_y, p_z]^T \quad (10)$$

It can be expressed either in the global Cartesian coordinates or in the boundary coordinate frames. To distinguish the representation of a point relative to a coordinate system, one may use the subscript "n" to indicate that the vector is expressed in the nth frame (attached to the nth boundary plane).

When the point is expressed with respect to the global coordinate frame, the subscript "n" will be omitted by default. According to the theory of geometric transformation, p_n is connected to p by the following relationship:

$$p_n = R_n p - t_n \quad (11)$$

where $R_n \in \mathbb{R}^3 \times \mathbb{R}^3$ is a rotation matrix, $t_n \in \mathbb{R}^3$ is the distance between the two origins of the two coordinate frames [2].

The x-y plane of a coordinate frame cuts the whole space into two parts. One half of the space is characterized by a positive z coordinate. This half-space does not contain the given polyhedron. The other half of the space is characterized by negative z coordinate, wherein the given polyhedron lies. Thus, only the z-coordinates are of interest in this study.

Equation (11) can be simplified as follows:

$$p_{nz} = z_n^T p - t_{nz} \quad (12)$$

where$z_n^T$is the last row of the rotation matrix R_n.

It is assumed that the n polyhedron has given shots, $p_{nz}$is obtained for 1 ≤ n ≤ N.

$p_{nz}$is the expression of p in the coordinate system connected to the boundary plane 'n'. In order to better mimic the relationship p to p_{nz}, we will proceed to the construction of p_p_{nz}network in order to transform p to p_{nz}.

This network will have as input the vector p=$[p_x, p_y, p_z]$and as output the p_{nz} measure.

Below the diagram of the network p_p_{nz} proposed:

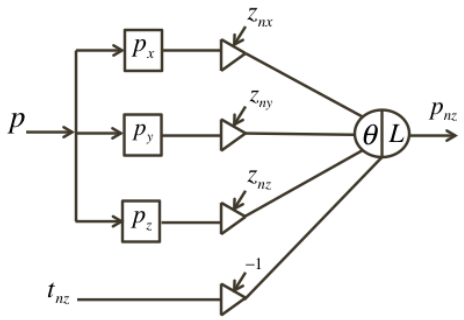

Fig. 6 Architecture of the network p-pnz

The following observations are essential for the rest of the study:

- If any one of $\{p_{nz}\}_{n=1}^{N}$is positive, in this case there is at least one boundary plane between the point P of the convex polyhedron. Thus the point P lies outside the polyhedron.
- On the other hand, if P is inside the convex polyhedron, then $p_{nz} \leq 0$ for all p ∈ [1, N].

On this basis, we define the following indicator:

$$p_{n*z} = \max_{1\leq n\leq N}\{p_{nz}\} \quad (13)$$

The collision between the point P and N surfaces of the polyhedron may be identified as follows:

- If$p_{n*z} > 0$, then collision free
- If$p_{n*z} \leq 0$, then collision occured, return direction vector z_{n*}.

IV. Collision Detection between two Polyhedra

Compared to the identification of a collision between a point and a polyhedron, it is more complex to identify a possible collision between two polyhedra. On basis of the previous section, a new measure of collision detection is introduced in the following.

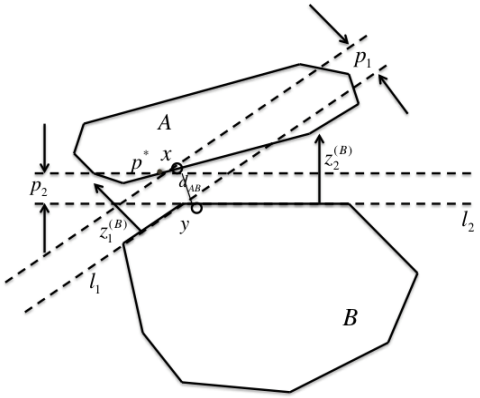

Fig. 7 Collision detection between polygon A and B

Fig. 7 illustrates perfectly the principle of our study, where two convex polygons are labeled A and B.

The numbers of boundary surfaces for the two polygons are denoted by N_A andN_B, and the Z axes of the boundary frames are denoted by $\left\{z_n^{(A)}\right\}_{n=1}^{N_A}$ and $\left\{z_n^{(B)}\right\}_{n=1}^{N_B}$, respectively. The corresponding thresholds sets are given by $\left\{t_n^{(A)}\right\}_{n=1}^{N_A}$ and $\left\{t_n^{(B)}\right\}_{n=1}^{N_B}$. A neural network can be programmed to identify a collision between a moving point P and the polygon B. The indicator p_{n*z} and its direction index n * in Eq. 13 are functions of vector P. For this reason, they should be denoted explicitly as $p_{n*z}(P)$ and n*(p). The new measure of collision detection is given by a constraint mini-maximum.

$$P_{n*z}^{*}=\inf_{p\in A}\{p_{n*z}(P)\}=\inf_{p\in A}\left\{\max_{1\leq k\leq N_B}\left[p^T z_k^{(B)} - t_k^{(B)}\right]\right\} \quad (14)$$

P_{n*z}^{*} Differs from $p_{n*z}$by the superscript "*". It is the indicator of $p^* \in A$, which has the smallest value in relation to indicators of any other points belonging to polygon A. In this paper, p^* is called the 'mini-maximum' point.

The positive value of P_{n*z}^{*}tells us about the fact that there is no point of the polyhedron A belonging to polyhedron B. Therefore, the two polygons do not contact nor collide with each other. If any part of the polygon A overlaps with a part of polygon B, as illustrated in Fig. 8.

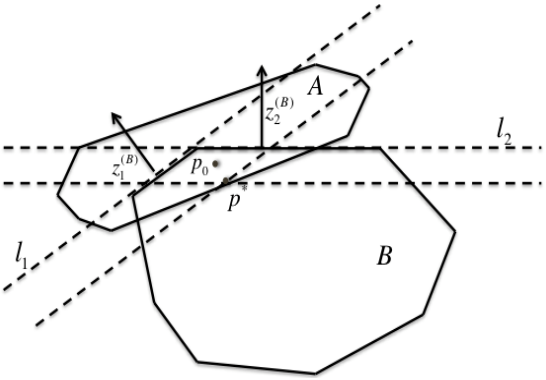

Fig. 8 Collision between two convex polyhedral A and B

There must exists at least one point$p_0$which belongs to both polygons. Since$p_0$is inside polyhedron B therefore:

$$p_{n*z}(p_0) = \max_{1\leq k\leq N_B}\left[p_0^T z_k^{(B)} - t_k^{(B)}\right] \leq 0 \quad (15)$$

A collision between polygons A and B may be defined as follows:

- If $P_{n*z}^{*} > 0$, then collision free
- If $P_{n*z}^{*} \leq 0$, then collision occured

The constraint mini-maximum $P_{n*z}^{*}$is called a pseudo-distance between polygons A and B. Its extension to polyhedra is rather straightforward. Mathematically, $P_{n*z}^{*}$is different from the Euclidean distance d_{AB}measuredbetween x-y as shown in Fig. 7. The difference between P_{n*z}^{*} andd_{AB}does

not affect the identification of collision or the generation of a collision free path.

V. Construction Of Collision Detection Network

The cornerstone of this study is the construction of the neural network DCNN of identifying a possible collision between two convex polyhedra through an algorithm to calculate the minimum distance between two polyhedra. To take advantage of parallelism and to minimize the complexity of the algorithm, we choose to use neural networks. After having explained in the preceding sections all networks, subnets, and neurons that are related to the construction of DCNN network, we will show below the architecture of the network in question:

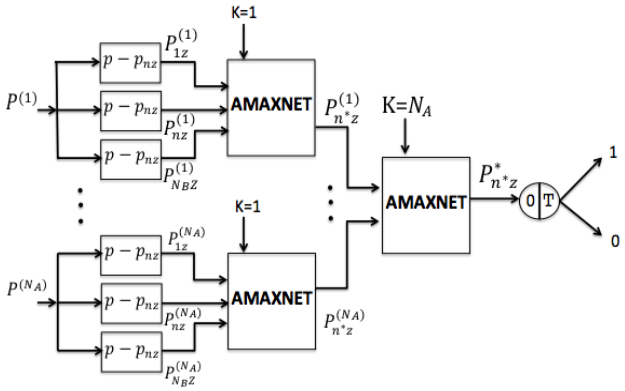

Fig. 9 Architecture of the network DCNN

DCNN has as input the vectors$P^{(n)}$ (P belong to polygon A), using the obtained network p_p_{nz}, we obtain p_{nz} for n $[1, N_B]$ which means the expression of P in the coordinate system connected to the plane defining the boundaries of the polygon B. Then through the AMAXNET with the adjustment input K=1, we receive$p_{n*z}{}^{(n)}$point representing the distance between the n^*th plane and the point P. The AMAXNET with the adjustment input K=N_A which has as entry the indicator $p_{n*z}{}^{(n)}$ for $n \in [1, N_A]$ give us as output the quantity P^*_{n*z}which denotes a pseudo distance between polygons A and B. Finally, by using the threshold logic neuronϕ_T, we obtain as output a binary result that informs us about the fact that there is a collision or not (1 means that there is no collision between A and B, 0 means that there is a collision between A and B).

VI. Conclusion

Through this study, we proposed a neural network in a fixed time to resolve the problem of collision detection between two convex polyhedra. Our network uses two types of neurons, linear and threshold logic ones. Among all neurons proposed in the literature, they are probably the easiest to implement.

The objective of this research is to determine the directions of the collision forces and the points on which the collision forces act. Instead of defining a potential function, a new collision detection measurement is inserted with a simple algorithm in fixed time. The proposed algorithm is suitable for implementation of neural networks using primitive elements of circuits, such as analogue integrators or linear operational amplifiers. The neural system does not require traditional CPU, memory chips or clock synchronization.

This network provides a very simple configuration, which avoids problems caused by poor precision in the implementation of connection weight values.

In addition, the network architecture is proper and simple, the connection strengths between neurons are all fixed, and most of them are equal to +1 or -1. This will greatly facilitate the actual hardware implementation of the project using the VLSI technology.

References

[1] M.Mestari, 'An analog Neural Network implementation in Fixed Time of Adjustable Order Statistic Filters and Applications' IEEE transactions on Neural Networks vol 15.No 3 pp 766-785, May2004.
[2] J.Yuan, 'Collision Identification between Convex Polyhedra Using Neural Networks', IEEE, April 1995.
[3] M. Mestari and A. Namir, "AMAXNET: A neural network implementation of adjustable MAXNET in fixed time", IFAC-IFIP-IMACS Proc. Internat. Conf. on Control of Industrial Systems, 20-22 May, Belfort, (France), vol. 2, 543-549, 1997.
[4] M. Mestari, A. Namir, K. Akodadi and A. Badi,"Θ(1) Time Neural Network Minimum Distance Classifier and its Application to Optical Character Recognition Problem", Applied Mathematical Sciences, Vol. 2, 2008, no. 26, 1253 - 1282, November 2007.
[5] M. Mestari and A. Namir, "AOSNET: A Neural Network Implementation of Adjustable Order Statistic Filters In Fixed Time", SAMS, 2000, Vol. 36, pp. 509-535.
[6] F. Chin and C. A. Wang, "Optimal algorithms for the intersection and minimum distance problems between planar polygons", IEEE Trans. Comput., vol. C-32, pp. 1203-1207, 1984.
[7] D. P. Dobkin and D. G. Kirkpatrick, "A linear algorithm for determining the separation of convex polyhedra", J. Algorithms, vol. 6, pp. 381-392, 1985.
[8] H. Edelsbrunner, "On computing the extreme distances between two convex polygons", J. Algorithms, vol. 6, pp. 515-542, 1985.
[9] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles", IEEE J. Robotics Automation, vol. RA-1, no. 1, pp. 21-30, 1985.
[10] D. T. Lee and F. P. Preparata, "Computational geometry, A survey", IEEE Trans. Comput., vol. C-33, pp. 1072-1101, 1984.
[11] J. T. Schwartz, "Finding the minimum distance between two convex polygons", inform. Professor. Lett., col. 13, pp. 168-170, 1981.
[12] C. Y. Liu and R. W. Mayne, "Distance calculations in motion planning problems with interference situations", in Proc. 1990 ASME Design Tech. Conf., vol. 23, no. 1, 1990, pp. 145-152.
[13] K. Fukushima, ."A neural network for visual pattern recognition", IEEE Computer, pp. 65-75, Mar., 1988.
[14] T. Kohonen, "Correlation matrix memories", IEEE Trans. Computers, vol.C-21, 353-359, 1972.