International Journal of Pattern Recognition and Artificial Intelligence
Vol. 16, No. 7 (2002) 797–815

A NEURAL NETWORK MODEL FOR MINIMUM SPANNING CIRCLE: ITS CONVERGENCE, ARCHITECTURE DESIGN AND APPLICATIONS

AMITAVA DATTA* and S. K. PARUI†

*Computer and Statistical Service Centre,
†Computer Vision and Pattern Recognition Unit,
Indian Statistical Institute, 203 B. T. Road, Calcutta 700 108, India
*amitava@isical.ac.in

A self-organizing neural network model that computes the smallest circle (also called minimum spanning circle) enclosing a finite set of given points was proposed by Datta.[3] In the article,[3] the algorithm is stated and it is demonstrated by simulation that the center of the smallest circle can be achieved with a given level of accuracy. No rigorous proof was given in support of the simulation results. In this paper, we make a rigorous analysis of the model and mathematically prove that the model converges to the desired center of the minimum spanning circle. A suitable neural network architecture is also designed for parallel implementation of the proposed model. Time complexity of the algorithm is worked out under the proposed architecture. Extension of the proposed model to higher dimensions is discussed and demonstrated with some applications.

Keywords: Minimum spanning circle; one-center; neural networks; self-organization; facility location; outlier detection; maximum likelihood estimator.

1. Introduction

This paper deals with the problem of finding the *minimum spanning circle* (MSC) or the *smallest enclosing circle* (SEC). The issue is to find the radius and the center of the smallest circle that encloses n given points in the Euclidean plane. In location theory this is the unweighted Euclidean *one-center* problem in the plane.

The MSC has applications in optimization, pattern recognition, image analysis, statistical estimation, etc. It has applications in transmission and transportation problems. Suppose a community of users needs the service of some facility, for example, radio/TV receivers with signals from a single transmitter. The problem is to find the optimum location of the transmitter (facility). In other words, we need to find where the facility should be located so as to minimize the maximum distance from the facility to any user. It is clear that, if we imagine the users as points in the plane, then the center of the MSC for this set of points is the solution

*Author for correspondence.

of the problem. In a different kind of application, for example in transportation, the center of the MSC can be the optimal location for a service station if we want to minimize the maximum distance that a customer would have to travel from his place to the service station. Here the locations of the residences specify the set of points.

The MSC problem has a long history. After it was posed by Sylvester[20] in 1857, it attracted several researchers and as a result many solutions[4–6,9,11,12,16–19,21] have been suggested. The (worst-case) time complexities for the above solutions range from $O(n^3)$ to $O(n \log n)$, where n is the number of input points. Megiddo[9] formulated this problem as a linear programming problem which could be solved in $O(n)$ time. A randomized algorithm is available for computing the MSC that takes the expected $O(n)$ time.[22] Apart from the time complexities, many of these algorithms suffer from implementation complexities. That is, they are not simple from the point of view of actual programming and implementation.

An algorithm, based on a *self-organizing neural network*, was proposed by Datta[3] to find the MSC for a given set of points. The author demonstrated, by computer simulation, how the algorithm finds the optimum position of the center of the MSC iteratively (Figs. 1 and 2). Against each presentation of input signals (here points) the network adapts without any supervision and finally converges to the optimum solution. In this paper, we mathematically analyze Datta's MSC algorithm and give a rigorous proof of convergence. We also propose a multilayer architecture for an efficient and parallel implementation of Datta's self-organizing algorithm. The article is organized as follows. Section 2 briefly describes the MSC

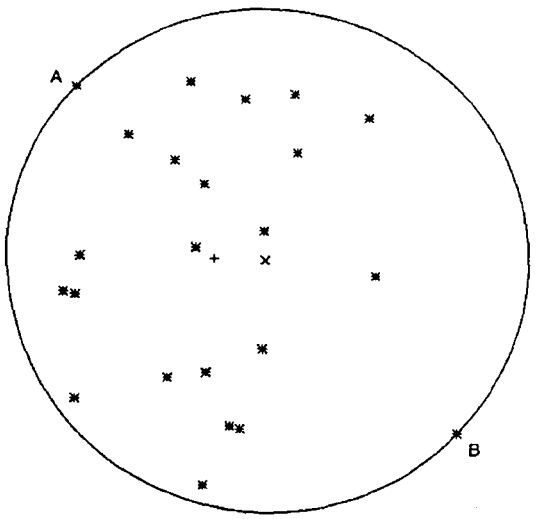


Fig. 1. A result obtained by Datta[3] when the minimum spanning circle is determined by two points. "+" represents the initial position of the weight vector and "x" represents the final center of the circle ($t = 50$).

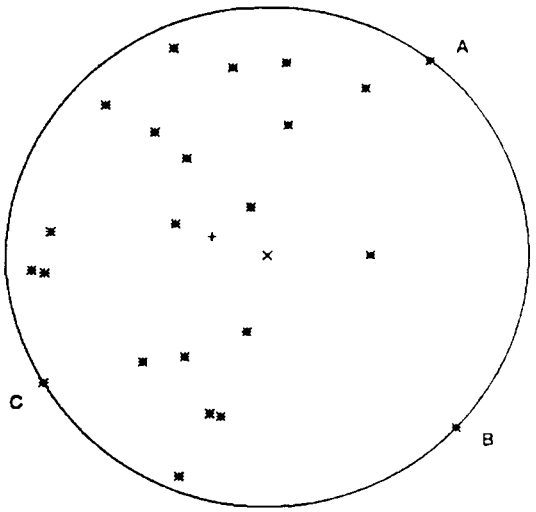


Fig. 2. A result obtained by Datta[3] when the minimum spanning circle is determined by three points ($t = 70$).

algrithm. Section 3 contains the mathematical analysis and the proof of convergence of the algorithm. The proposed architecture for the MSC model is presented in Sec. 4 where the time complexity is also worked out. Intuitively, the proposed model is straightaway extendable to higher dimensions. This has been demonstrated by some applications in Sec. 5.

2. The MSC Algorithm

The formal description of the MSC problem is as follows:

Let $S = \{P_1, P_2, \ldots, P_n\}$ be a set of n given points in the Euclidean plane. The problem is to find the center and radius of the smallest circle such that no point of S falls outside the circle. Such a circle is called the minimum spanning circle. The problem can be stated as:

Find the point $W = (w_1, w_2)$ so that

$$\max_j \|P_j - W\| \tag{1}$$

is minimized over all choices of W in the plane. In other words, we compute

$$r = \min_{(w_1, w_2)} \max_j \{((w_1 - a_j)^2 + (w_2 - b_j)^2)^{\frac{1}{2}}\} \tag{2}$$

where (a_j, b_j) is the co-ordinates of $P_j, j = 1, 2, \ldots, n$. The radius of the MSC is r and the optimum (w_1, w_2) is the center.

Assign one processor to each point of S. Another processor π stores a weight vector $W = (w_1, w_2)$. This weight vector will be updated iteratively in the self-organization process. The weight vector here represents a position in the plane where the processor π can be thought to be located.

The initial value, $W(0)$, of the weight vector W is set at random. At iteration t we find the farthest point (w.r.t. Euclidean distance) from $W(t)$. Let $P_k = (a_k, b_k)$ be the farthest point, $k \in \{1, 2, \ldots, n\}$. That is,

$$\|P_k - W\| = \max_j \|P_j - W\| . \tag{3}$$

Then the weight vector W is updated as follows :

$$W(t+1) = W(t) + \alpha(t)(P_k - W(t)) \tag{4}$$

where $\alpha(t)$ is a decreasing function of iteration number t satisfying some condition as stated in the next section.

The idea is as follows. Find the farthest point P_k from $W(0)$. Since we desire to minimize the distance $\text{dist}(P_k, W(0))$, we try to move W toward P_k. With a suitable value of α (as discussed later) we update the weight vector W according to Eq. (4). This process is repeated enabling W to gradually move towards P_k. After each weight update the farthest point from W is recalculated. It is clear that during this process, some new point $P_{k'} (k' \neq k)$ becomes the farthest point from the weight vector W and, as soon as it happens, W starts moving toward $P_{k'}$ instead of moving toward P_k. The weight update process is continued repeatedly with $\alpha(t) \to 0$ as $t \to \infty$ enabling the process to stabilize when the difference between the weight vectors at two successive iterations is negligible.

The Algorithm MSC

Step 1. Initialize iteration number $t = 0$; Initialize $W(t) = (w_1(t), w_2(t))$ with random values.

Step 2. Calculate $d_j = \text{dist}(W(t), P_j)$ for all $j = 1, 2, \ldots, n$.

Step 3. Find the maximum d_j, and the point P_k such that $\max_j d_j = \text{dist}(W(t), P_k)$.

Step 4. Adjust weight vector $W(t)$ according to Eq. (4).

Step 5. Set $t = t + 1$ and repeat from Step 2 until the weight vectors of two successive iterations are sufficiently close.

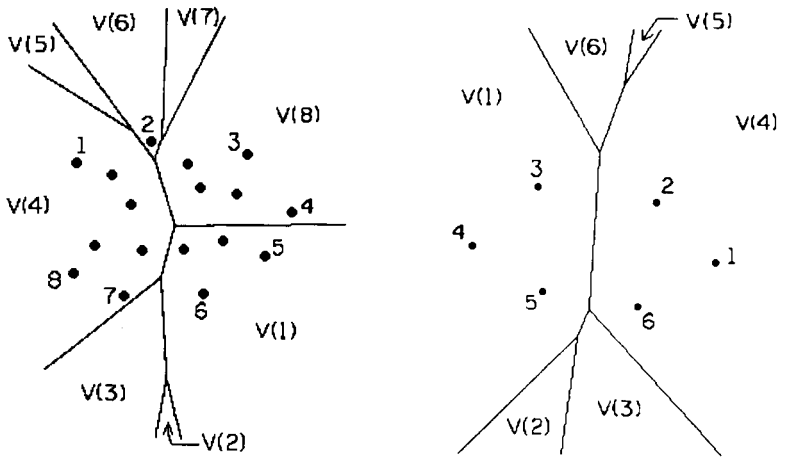
3. Analysis and Convergence of the MSC Algorithm

In this section, we do mathematical analysis of the **Algorithm-MSC** and study its convergence properties. Before going into the details of these we need the following definition to be stated.

Definition 1. For the set S, the *farthest point Voronoi diagram* (FPVD) is a partition of the plane defined by the convex regions V_r, $1 \leq r \leq n$ where

$$V_r = \{P : \|P - P_r\| > \|P - P_t\| \ \forall t \neq r\} \tag{5}$$

V_r is the farthest point Voronoi polygon corresponding to P_r, $1 \leq r \leq n$. Farthest point Voronoi diagrams are shown in Fig. 3.



(a) (b)

Fig. 3. Farthest point Voronoi diagrams for two planar sets. MSC is determined by (a) three points, (b) two points.

We shall now prove that in **Algorithm-MSC**, the weight vector $W(t)$ converges to the center (say, W^*) of the minimum spanning circle. The major cases of the proof are as follows. We prove

Case-1. If $S = \{P_1\}$ then $W^* = P_1$ where W^* is the limiting value of $W(t)$.

Case-2. If $S = \{P_1, P_2\}$ then $W^* = \frac{1}{2}(P_1 + P_2)$.

Case-3. If $S = \{P_1, P_2, P_3\}$ then W^* is either (a) the midpoint of one of the sides of the triangle $P_1P_2P_3$ or (b) the common FPV vertex of the three FPV polygons V_1, V_2, V_3.

Case-4. If $S = \{P_1, P_2, \ldots, P_n\}$ then

(a) If the MSC is determined by two points P_i, P_j then $W^* = \frac{1}{2}(P_i + P_j)$.

(b) If the MSC is determined by three points P_i, P_j, P_k then W^* = the common FPV vertex of the three FPV polygons V_i, V_j, V_k.

Before going into the proof, we state below a few known results.[11,14,17]

Result 1. If there is a circle C passing through three points, P_i, P_j and P_k such that they do not lie on any open semicircumference of C and if C contains all the points, then C is the MSC.

Result 2. If the MSC passes through exactly two points P_i and P_j, then the line segment P_iP_j forms a diameter of the MSC.

Result 3. If no two points of S form a diameter of the MSC, then the MSC passes through at least three points which do not lie on any open semicircumference of the MSC.

Result 4. The MSC is unique.

Let $X(t) \in S, t = 0, 1, 2, \ldots$ be such that $\|X(t) - W(t)\| = \max_i \|P_i - W(t)\|, 0 < \alpha(t) < 1$ for all t and $\alpha(t)$ decreases to 0 as t tends to ∞ such that $\sum_{t=0}^{\infty} \alpha(t) = \infty$.

Note that the points of S come from a bounded region. Hence, $\|W(0) - P_i\| < K$ for all i, for some $K > 0$. It is easy to see that $\|W(t) - P_i\| < K$ for all i, for all t.

Lemma 1. $\prod_{t=0}^{\infty}[1 - \alpha(t)] = 0$.

Proof. Since $0 < \alpha(t) < 1$ for all t, $\sum_{t=0}^{\infty} \alpha(t) = \infty$ if and only if $\prod_{t=0}^{\infty}[1-\alpha(t)] = 0$ (see Ref. 1). □

We shall now prove a result on $\alpha(t)$. For any given t_0, let $S(0) = \alpha(t_0)$ and $S(t+1) = [1 - \alpha(t_0 + t + 1)]S(t) + \alpha(t_0 + t + 1)$ for $t = 0, 1, 2, \ldots$.

Note that

$$S(1) = \alpha(t_0)[1 - \alpha(t_0 + 1)] + \alpha(t_0 + 1)$$

$$S(2) = \alpha(t_0)[1 - \alpha(t_0 + 1)][1 - \alpha(t_0 + 2)] + \alpha(t_0 + 1)[1 - \alpha(t_0 + 2)] + \alpha(t_0 + 2)$$

$$\begin{aligned} S(3) = {} & \alpha(t_0)[1 - \alpha(t_0 + 1)][1 - \alpha(t_0 + 2)][1 - \alpha(t_0 + 3)] \\ & + \alpha(t_0 + 1)[1 - \alpha(t_0 + 2)][1 - \alpha(t_0 + 3)] \\ & + \alpha(t_0 + 2)[1 - \alpha(t_0 + 3)] + \alpha(t_0 + 3)\,. \end{aligned}$$

In general,

$$\begin{aligned} S(t) = {} & \alpha(t_0)[1 - \alpha(t_0 + 1)][1 - \alpha(t_0 + 2)] \cdots [1 - \alpha(t_0 + t)] \\ & + \alpha(t_0 + 1)[1 - \alpha(t_0 + 2)] \cdots [1 - \alpha(t_0 + t)] \\ & + \alpha(t_0 + 2)[1 - \alpha(t_0 + 3)] \cdots [1 - \alpha(t_0 + t)] + \cdots \\ & + \alpha(t_0 + t - 2)[1 - \alpha(t_0 + t - 1)][1 - \alpha(t_0 + t)] \\ & + \alpha(t_0 + t - 1)[1 - \alpha(t_0 + t)] + \alpha(t_0 + t)\,. \end{aligned}$$

Lemma 2. *$S(t)$ goes to 1 as t goes to ∞.*

Proof. It is easy to see that

$$\begin{aligned} 1 - S(1) &= [1 - \alpha(t_0)][1 - \alpha(t_0 + 1)] \\ 1 - S(2) &= [1 - S(1)][1 - \alpha(t_0 + 2)] \\ &= [1 - \alpha(t_0)][1 - \alpha(t_0 + 1)][1 - \alpha(t_0 + 2)] \\ 1 - S(3) &= [1 - S(2)][1 - \alpha(t_0 + 3)] \\ &= [1 - \alpha(t_0)][1 - \alpha(t_0 + 1)][1 - \alpha(t_0 + 2)][1 - \alpha(t_0 + 3)]\,. \end{aligned}$$

In general,

$$1 - S(t) = [1 - \alpha(t_0)][1 - \alpha(t_0 + 1)][1 - \alpha(t_0 + 2)] \cdots [1 - \alpha(t_0 + t)]\,.$$

From Lemma 1, $\prod_{t=t_0}^{\infty}[1-\alpha(t)] = 0$. Hence Lemma 2. □

Applying Eq. (4) repeatedly we get

$$\begin{aligned}W(t_0+t+1) &= [1-\alpha(t_0)][1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]W(t_0)\\ &\quad+\alpha(t_0)[1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]X(t_0)\\ &\quad+\alpha(t_0+1)[1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]X(t_0+1)\\ &\quad+\alpha(t_0+2)[1-\alpha(t_0+3)]\cdots[1-\alpha(t_0+t)]X(t_0+2)+\cdots\\ &\quad+\alpha(t_0+t-2)[1-\alpha(t_0+t-1)][1-\alpha(t_0+t)]X(t_0+t-2)\\ &\quad+\alpha(t_0+t-1)[1-\alpha(t_0+t)]X(t_0+t-1)\\ &\quad+\alpha(t_0+t)X(t_0+t)\,.\end{aligned}$$

Case 1. $n = 1$.

Here, $X(t) = P_1$ for all t. Hence, from above, we get

$$W(t_0+t+1) = [1-S(t)]W(t_0) + S(t)P_1\,.$$

Lemma 3. *For Case* 1, $W(t_0+t)$ *goes to* P_1 *as* t *goes to* ∞.

Case 2. $n = 2$. The center of the MSC is $\frac{1}{2}(P_1+P_2)$.

Lemma 4. *For Case* 2, $W(t_0+t)$ *goes to* $(P_1+P_2)/2$ *as* t *goes to* ∞.

Proof. It is based on Propositions 1 and 2 below.

Proposition 1. $W(t_0+t+1)$ *can be made arbitrarily close to the straight line* L *passing through* P_1 *and* P_2.

Note that

$$\begin{aligned}&P_1 - W(t_0+t+1)\\ &\quad= [1-\alpha(t_0)][1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)][P_1-W(t_0)]\\ &\qquad+\alpha(t_0)[1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)][P_1-X(t_0)]\\ &\qquad+\alpha(t_0+1)[1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)][P_1-X(t_0+1)]\\ &\qquad+\alpha(t_0+2)[1-\alpha(t_0+3)]\cdots[1-\alpha(t_0+t)][P_1-X(t_0+2)]+\cdots\\ &\qquad+\alpha(t_0+t-2)[1-\alpha(t_0+t-1)][1-\alpha(t_0+t)][P_1-X(t_0+t-2)]\\ &\qquad+\alpha(t_0+t-1)[1-\alpha(t_0+t)][P_1-X(t_0+t-1)]\\ &\qquad+\alpha(t_0+t)[P_1-X(t_0+t)]\end{aligned}$$

where X is either P_1 or P_2.

Now,

$$\begin{aligned}
&\|P_1 - W(t_0+t+1)\| \\
&\quad \le [1-\alpha(t_0)][1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_1 - W(t_0)\| \\
&\qquad + \alpha(t_0)[1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_1 - X(t_0)\| \\
&\qquad + \alpha(t_0+1)[1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_1 - X(t_0+1)\| \\
&\qquad + \alpha(t_0+2)[1-\alpha(t_0+3)]\cdots[1-\alpha(t_0+t)]\|P_1 - X(t_0+2)\| + \cdots \\
&\qquad + \alpha(t_0+t-2)[1-\alpha(t_0+t-1)][1-\alpha(t_0+t)]\|P_1 - X(t_0+t-2)\| \\
&\qquad + \alpha(t_0+t-1)[1-\alpha(t_0+t)]\|P_1 - X(t_0+t-1)\| \\
&\qquad + \alpha(t_0+t)\|P_1 - X(t_0+t)\| .
\end{aligned}$$

Similarly,

$$\begin{aligned}
&\|P_2 - W(t_0+t+1)\| \\
&\quad \le [1-\alpha(t_0)][1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_2 - W(t_0)\| \\
&\qquad + \alpha(t_0)[1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_2 - X(t_0)\| \\
&\qquad + \alpha(t_0+1)[1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_2 - X(t_0+1)\| \\
&\qquad + \alpha(t_0+2)[1-\alpha(t_0+3)]\cdots[1-\alpha(t_0+t)]\|P_2 - X(t_0+2)\| + \cdots \\
&\qquad + \alpha(t_0+t-2)[1-\alpha(t_0+t-1)][1-\alpha(t_0+t)]\|P_2 - X(t_0+t-2)\| \\
&\qquad + \alpha(t_0+t-1)[1-\alpha(t_0+t)]\|P_2 - X(t_0+t-1)\| \\
&\qquad + \alpha(t_0+t)\|P_2 - X(t_0+t)\| .
\end{aligned}$$

Hence,

$$\begin{aligned}
&\|P_1 - W(t_0+t+1)\| + \|P_2 - W(t_0+t+1)\| \\
&\quad \le [1-\alpha(t_0)][1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)][\|P_1 - W(t_0)\| \\
&\qquad + \|P_2 - W(t_0)\|] \\
&\qquad + \alpha(t_0)[1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)][\|P_1 - X(t_0)\| \\
&\qquad + \|P_2 - X(t_0)\|] \\
&\qquad + \alpha(t_0+1)[1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)][\|P_1 - X(t_0+1)\| \\
&\qquad + \|P_2 - X(t_0+1)\|] \\
&\qquad + \alpha(t_0+2)[1-\alpha(t_0+3)]\cdots[1-\alpha(t_0+t)][\|P_1 - X(t_0+2)\|
\end{aligned}$$

$$
\begin{aligned}
&+\|P_2 - X(t_0+2)\|] + \cdots \\
&+\alpha(t_0+t-2)[1-\alpha(t_0+t-1)][1-\alpha(t_0+t)][\|P_1 - X(t_0+t-2)\| \\
&+\|P_2 - X(t_0+t-2)\|] \\
&+\alpha(t_0+t-1)[1-\alpha(t_0+t)][\|P_1 - X(t_0+t-1)\| + \|P_2 - X(t_0+t-1)\|] \\
&+\alpha(t_0+t)[\|P_1 - X(t_0+t)\| + \|P_2 - X(t_0+t)\|] \\
=\ &[1-\alpha(t_0)][1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)][\|P_1 - W(t_0)\| \\
&+\|P_2 - W(t_0)\|] \\
&+\alpha(t_0)[1-\alpha(t_0+1)][1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_1 - P_2\|] \\
&+\alpha(t_0+1)[1-\alpha(t_0+2)]\cdots[1-\alpha(t_0+t)]\|P_1 - P_2\| \\
&+\alpha(t_0+2)[1-\alpha(t_0+3)]\cdots[1-\alpha(t_0+t)][\|P_1 - P_2\| + \cdots \\
&+\alpha(t_0+t-2)[1-\alpha(t_0+t-1)][1-\alpha(t_0+t)][\|P_1 - P_2\| \\
&+\alpha(t_0+t-1)[1-\alpha(t_0+t)]\|P_1 - P_2\| \\
&+\alpha(t_0+t)\|P_1 - P_2\| \\
=\ &[1-S(t)][\|P_1 - W(t_0)\| + \|P_2 - W(t_0)\|] + S(t)\|P_1 - P_2\| .
\end{aligned}
$$

Now, $S(t)$ goes to 1 as t tends to ∞. So, $W(t_0+t+1)$ gets arbitrarily close to the line segment joining P_1 and P_2. Hence Proposition 1. □

Proposition 2. *$W(t_0+t+1)$ can be made arbitrarily close to the perpendicular bisector L_1 of the line segment P_1P_2.*

Let H_1 and H_2 be the two half planes defined by L_1 such that P_i belongs to H_i $(i = 1, 2)$. Without loss of generality, let $W(t_0)$ lie in H_1. From Lemma 3, we know $W(t_0+t)$ will belong to H_2 for some t. This is because $W(t_0+t)$ moves towards P_2 until it falls in H_2 [Fig. 4(*a*)]. Suppose $W(t_0+1), W(t_0+2), \ldots, W(t_0+t_1-1)$ belong to H_1 and $W(t_0+t_1)$ belongs to H_2. Now, the perpendicular distance of $W(t_0+t_1)$ from L_1 is less than or equal to $\|W(t_0+t_1-1) - W(t_0+t_1)\|$ [Fig. 4(a)].

Note that $\|W(t_0+1) - W(t_0)\| = \alpha(t_0)\|X(t_0) - W(t_0)\| \leq \alpha(t_0)K$.

Since $\alpha(t) \to 0$ as $t \to \infty$, for any $\delta > 0$, we can take t_0 to be sufficiently large so that $\|W(t_0+1) - W(t_0)\| < \delta$ and in general, $\|W(t_0+t) - W(t_0+t-1)\| < \delta$ for all $t > 0$.

Now, $W(t_0+t_1+t)$ will belong to H_1 for some t. Suppose, $W(t_0+t_1), \ldots, W(t_0+t_1+t_2-1)$ belong to H_2 and $W(t_0+t_1+t_2)$ belongs to H_1. From Fig. 4(a), it is clear that the distance between $W(t_0+t_1+t)$ and L_1 is less than δ for all $t > 0$. Hence Proposition 2.

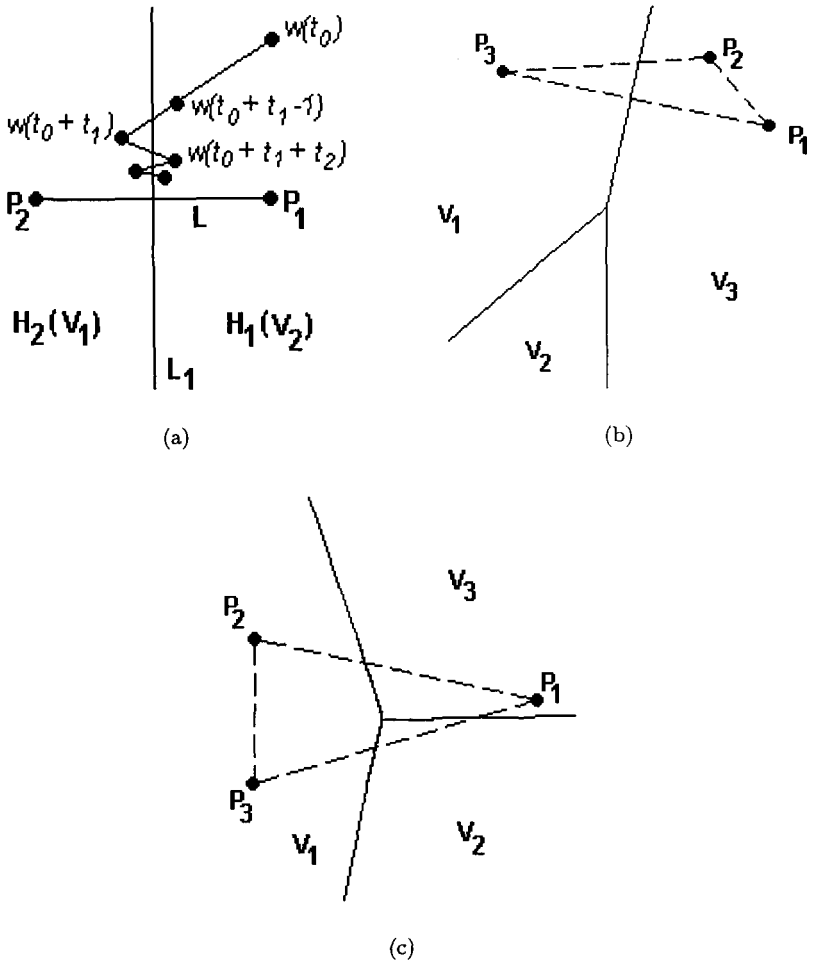


Fig. 4. (a) The trajectory of $W(t)$ for Case 2; (b) the FPVD for Case 3(a); (c) the FPVD for Case 3(b). The FPVD is denoted by solid lines and the triangle formed by the input points is denoted by dashed lines.

Case 3. Suppose $n = 3$. Consider the farthest point Voronoi diagram (FPVD) of the points. The only vertex of the FPVD of P_1, P_2, P_3 may lie either (a) outside [Fig. 4(b)] or (b) inside [Fig. 4(c)] the triangle $\Delta P_1P_2P_3$ formed by P_1, P_2, P_3. The FPV polygon corresponding to the point P_i is V_i $(i = 1, 2, 3)$.

Case 3(a). In the former case, for some large t_0, $W(t)$ will lie outside V_2 for all $t > t_0$. From Case-2, it is then clear that $W(t)$ converges to $(P_1 + P_3)/2$ which is the center of the MSC.

Case 3(b). In the latter case, consider the three perpendicular bisectors L_{ij} of the line segments P_iP_j $(1 \leq i \neq j \leq 3)$. From the arguments given in the proof of Proposition 2 in Case-2, $W(t)$ will be arbitrarily close to each of the three L_{ij}'s.

Thus $W(t)$ converges to the intersection point of the three bisectors which is the FPV vertex where the FPV polygons V_1, V_2, V_3 meet. Here, the FPV vertex is the center of the MSC. Note that for any large t_0 and for any i, there will be a $t_i > t_0$ such that $W(t_i)$ lies in V_i. In other words, each V_i will be visited by $W(t)$ infinitely many times.

Denote the limiting value of $W(t)$ by W^*.

Lemma 5. *For Case-3, $W(t_0+t)$ goes either to the midpoint of a side of $\Delta P_1P_2P_3$ or to the vertex of the FPVD of P_1, P_2, P_3 as t goes to ∞.*

Case 4. Suppose $n > 3$.

We claim that W^* will lie either (a) on an open edge (an edge excluding the vertex points) or (b) on a vertex of the FPV diagram of the point set S. Suppose not. Suppose, W^* lies in the interior of some V_p. Then, after some time, $W(t)$'s will always lie in the interior of V_p in which case, the limiting value of $W(t)$ will be P_p (from Case 1). It is a contradiction since V_p cannot contain P_p. Hence we have two subcases:

Case 4(a). The limiting value W^* lies on an open FPV edge. Suppose this edge is the common edge of the two FPV polygons V_i and V_j. That means, for sufficiently large t_0, $W(t)$ will lie either in V_i or in V_j for all $t > t_0$. Moreover, for any t' there will exist $t_i, t_j > t'$ such that $W(t_i)$ and $W(t_j)$ are in V_i and V_j, respectively. In other words, both V_i and V_j (and only they) will be visited by $W(t)$ infinitely many times. This case is similar to Case 2 and hence $W^* = \frac{1}{2}(P_i + P_j)$ [see Fig. 3(b)]. Now construct a circle C, centred at W^* passing through P_i, P_j. Obviously, P_iP_j forms the diameter of the circle C and since P_i, P_j are the farthest points from W^*, C contains all the points of S. Hence, by Results 1–4, C is the MSC.

Case 4(b). The limiting value of W^* lies on an FPV vertex. Suppose W^* lies on the common FPV vertex v_{ijk} of three FPV polygons say, V_i, V_j and V_k. That means, for sufficiently large t_0, $W(t)$ will lie either in V_i or V_j or V_k for all $t > t_0$ and each of V_i, V_j, V_k will be visited by $W(t)$ infinitely many times. Then it will be similar to Case 3(b). Note that W^* will be equidistant from P_i, P_j and P_k. Construct a circle C, centred at W^* and passing through P_i, P_j and P_k. We claim that P_i, P_j and P_k do not lie on an open semicircumference of C. Suppose not. Then v_{ijk} falls outside the triangle $\Delta P_iP_jP_k$. Hence, from Case 3(a), W^* cannot lie on v_{ijk} [see Fig. 4(b)]. This is a contradiction since W^* lies on v_{ijk}. Moreover, since P_i, P_j and P_k are the farthest points from v_{ijk}, C contains all the points of S. Hence C is the MSC from Results 1–4.

Lemma 6. *For Case-4, as t goes to ∞, either (a) $W(t_0+t)$ goes to $\frac{1}{2}(P_i+P_j)$ for some P_i, P_j (when the MSC is determined by the two points P_i, P_j) or (b) $W(t_0+t)$ goes to the vertex of the FPVD of P_i, P_j, P_k for some P_i, P_j, P_k (when the MSC is determined by the three points P_i, P_j, P_k).*

Summarizing the above we have:

Case-1. If $S = \{P_1\}$ then $W^* = P_1$.
Case-2. If $S = \{P_1, P_2\}$ then $W^* = \frac{1}{2}(P_1 + P_2)$.
Case-3. If $S = \{P_1, P_2, P_3\}$ then W^* is either (a) the midpoint of one of the sides of the triangle $P_1P_2P_3$ or (b) the common FPV vertex of the three FPV polygons V_1, V_2, V_3.
Case-4. If $S = \{P_1, P_2, \ldots, P_n\}$ then

(a) If the MSC is determined by two points P_i, P_j then $W^* = \frac{1}{2}(P_i + P_j)$.
(b) If the MSC is determined by three points P_i, P_j, P_k then W^* = the common FPV vertex of the three FPV polygons V_i, V_j, V_k.

Definition 2. The points by which the MSC is determined are called the *contact points* of the MSC.

4. The Architecture Design

It can be seen that the above model works as a self-organizing process. We present a set of input vectors to a network of processors against which a competition takes place and the network generates some feedback (the maximum d_j and the point P_k in Step 3 of the algorithm). Depending on the feedback the processor π adapts and moves accordingly. It should be kept in mind that the processor never moves physically. The movement is in terms of the changes in the weight vector. If the process is repeated iteratively, the process converges and the processor π positions itself to the desired center of the MSC. Moreover, the above learning is unsupervised. In brief, from the inputs the network learns without any supervision and finally outputs the MSC. Similar things essentially happen in self-organization.[7]

We now describe a neural network architecture to implement the MSC model. The network consists of three layers (Fig. 5) where the bottom and middle layers contain n (n = input size) neurons each and the top layer contains only one neuron. The input vectors are stored into the n neurons of the bottom layer and the neuron in the top layer stores the weight vector W. The jth neuron in the bottom layer is connected to the jth neuron in the middle layer. All the neurons in the bottom and middle layers are connected to the single neuron in the top layer. Additionally, the middle layer is a fully connected network, that is, every neuron is connected to every other neuron. Similar types of multiple layers of neurons communicating via feedforward and feedback connections have been used by several researchers[2,13,15] in self-organizing models.

Every neuron in the middle layer is composed of a building block (Fig. 6) as used by Pal *et al.*[13] in their RANKNET model. The RANKNET model ranks all input values presented to the neurons in a single iteration where the neuron with the maximum input value outputs rank n. Thus the maximum input value can be obtained in a single clock time.

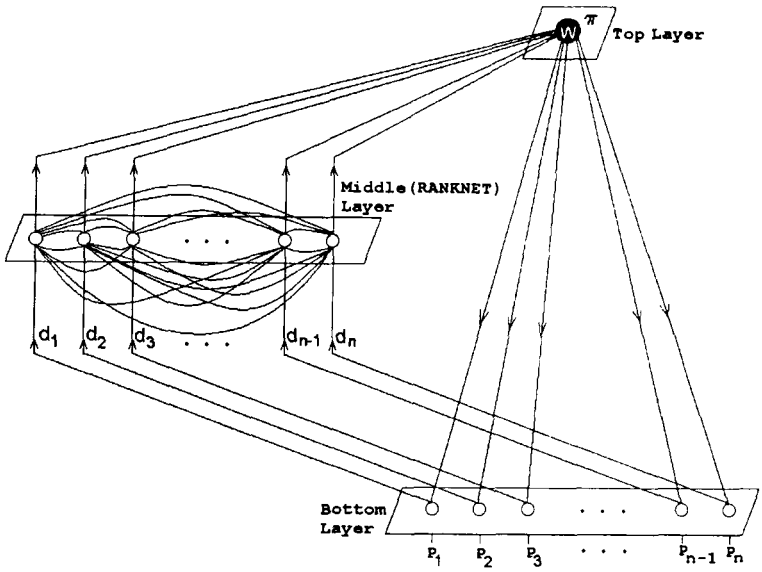


Fig. 5. The architecture of the MSC model. Empty circles denote fixed processors and the solid circle denotes floating processor.

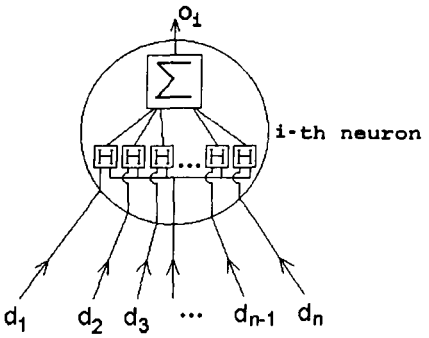


Fig. 6. The architecture for a neuron in the middle layer (Fig. 5). The internal threshold (H) of the ith neuron is its input value d_i.

The RANKNET is a fully connected network composed of n neurons, where n is the number of input values. Every neuron is connected to every other neuron and every input value is associated with a neuron. Let d_i denote the input value to the ith neuron. Denote the output of the ith neuron by o_i. The ith neuron, $i = 1, 2, \ldots, n$, computes

$$o_i(0) = \sum_{j=1}^{n} H(d_j, d_i) \tag{6}$$

where

$$H(d_j, d_i) = \begin{cases} 1 & \text{if } d_j \geq d_i \\ 0 & \text{otherwise.} \end{cases}$$

Thus, H is a hard-limiter that uses the respective neuron's input as the internal threshold. The ith neuron computes $H(d_j, d_i), j = 1, 2, \ldots, n$ and then adds them to get the output o_i. Hence o_i gives the rank $r_i \in \{1, 2, \ldots, n\}$ of the input value d_i. All the neurons compute their outputs o_i's simultaneously. Thus, the ranks of all input values are obtained in a single iteration.

The hardware implementation of the proposed network is much simpler than the existing MAXNET (or MINNET) models. Every neuron in Fig. 5 (middle layer) is composed of a building block as given in Fig. 6. The difference between our neuron architecture and the same in the MAXNET model proposed by Lippman *et al.*[8] is that each neuron in the MAXNET model first computes the weighted sum of the input values and then uses a threshold logic function to compute the output while every neuron in the RANKNET model first uses hard-limiters and then computes the sum.

Computational Cost

The present algorithm works in parallel with the help of multiple processors (see Fig. 5). The input points are assigned to each processor in the bottom layer. Initially, the weight vector W in the processor π in the top layer is assigned to a random value. The value of W is passed to every neuron at the bottom layer and the neurons in the bottom layer simultaneously compute $d_i, i = 1, 2, \ldots, n$. This computation is done in constant time. Now the ith neuron of the bottom layer passes the computed d_i value, along with the corresponding P_i, to the ith neuron of the middle layer. The middle layer, by the RANKNET algorithm,[13] computes the ranks (and hence selects the *winner* neuron, that is, the neuron with rank n) in constant time. The P_i value of the winner neuron is passed to the neuron π in the top layer which updates the value of W by Eq. 4. Thus one complete cycle (top layer $\rightarrow$ bottom layer $\rightarrow$ middle layer $\rightarrow$ top layer) takes constant time. The cycle (iteration) is repeated until $W(t)$ converges, that is, $W(t+1)$ and $W(t)$ are sufficiently close. Thus, in the whole process, complexity does not depend on the size of the input.

5. Applications

5.1. *Outlier detection*

We have proved, for the two-dimensional case, that the MSC model[3] converges and yields at the center of the minimum spanning circle. Although the general case is not proved, it is not difficult to conjecture that the model can be extended to any higher dimensional case. This can be done by replacing the 2D weight vector by d-dim ($d > 2$) weight vector and computing the d-dim Euclidean distance. Other

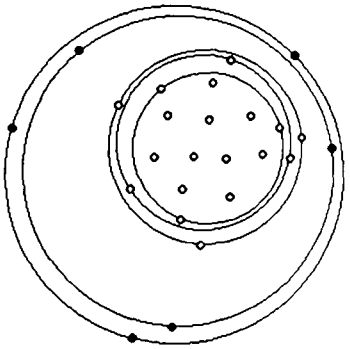


Fig. 7. A planar point set with outliers. Empty dots (tiny circles) represent valid points and solid dots represent outliers.

computations remain the same. Thus a similar architecture, as proposed in the 2D case, can be used in higher dimensions also.

With the above notion, an experiment is carried out that detects the outliers from an input set in a high dimensional space. We first explain the problem on a 2D input set. In Fig. 7, a set of planar points with a few outliers are shown. The MSC of the point set is computed and the contact points are removed from the input set. This is repeated for several iterations. In this process, after all the outliers get removed from the input set, it is expected that the radius of the (newly) computed MSC will become relatively steady.

Outlier detection in a high dimensional space is a challenging problem. We have taken data points from a very high dimensional space ($d = 153$) and have successfully detected the outliers using the higher dimensional version of the MSC model. The outlier detection algorithm goes as follows.

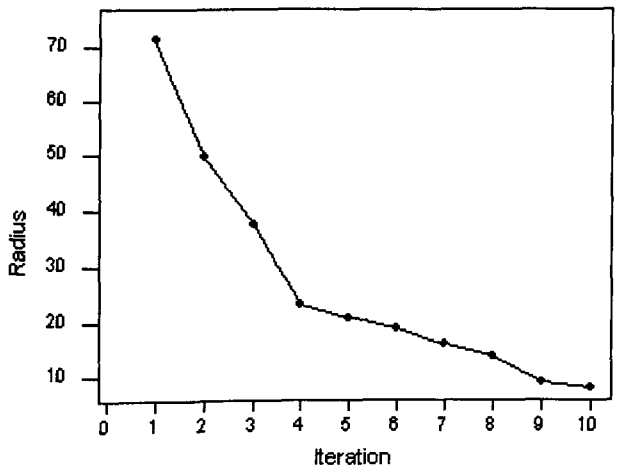


Fig. 8. The graph of successive radii.

Table 1. Average errors in estimating the radius and the center.

n	$\lvert r - 100\rvert$	$\lVert C - (100, 100, 100)\rVert$
5	25.1572	28.3163
10	12.1049	21.5907
20	4.6939	7.3968
50	1.3422	2.8645
100	0.9841	2.0011
500	0.1011	0.3753

Outlier Detection Algorithm

Step 1. Compute the MSC of the input set and its radius.
Step 2. Remove the contact points (see Definition 2) from the input set.
Step 3. Repeat from Step 1 until two successive radii are relatively close.

Figure 8 is the graph showing the successive radii. It is found that there is a significant fall of the value of the radius upto iteration 4 after which it becomes relatively steady. Hence after iteration 4, one can reasonably conclude that the remaining data set has no outliers.

5.2. *Statistical estimation*

The algorithm proposed in this paper can be used for statistical estimation of parameters of a certain type of statistical distributions. Suppose, f is the probability density function of the uniform distribution over a hyper-sphere in d dimensions. The unknown parameters in f here are the radius and the center of the hypersphere.

Let $P_1, P_2, \ldots, P_n$ (each a d-dimensional point) be a random sample of size n drawn from this distribution. The maximum likelihood estimators of the parameters here are the radius and the center of the smallest hyper-sphere containing $P_1, P_2, \ldots, P_n$. We made an empirical study of how these estimators perform when $d = 3$. The radius and the center are taken as 100 and (100, 100, 100). Six different values of n are considered, namely, 5, 10, 20, 50, 100, 500. For each of these six values, the algorithm has been tested for 20 random seeds for the starting vector. In each such case, the difference $|r - 100|$ and the distance $\sqrt{(c_1 - 100)^2 + (c_2 - 100)^2 + (c_3 - 100)^2}$ are computed where r and $C = (c_1, c_2, c_3)$ are the estimates for the actual radius and center. For each value of n, the averages of the two error quantities are computed (Table 1). It can be seen that both these errors decrease as n increases and the maximum likelihood estimators based on the smallest enclosing hyper-sphere perform satisfactorily.

6. Conclusions

During the last few years, there has been a great interest in applying neural networks technology in various fields of conventional computing. This is due to the facts that neurocomputing provides parallelism, it is adaptive and it sometimes simplifies a problem. A number of processors, each performing simple computations, when work collectively, can solve a complex problem. Datta[3] proposed a self-organizing model to compute the minimum spanning circle for a given set of points. The present work is a theoretical study of the model that proves the convergence of the algorithm. It discusses an efficient architectural design and some applications of the model.

It is shown that the MSC algorithm, proposed by Datta,[3] converges to the center of the desired smallest circle. There exist several algorithms[4–6,11,12,16–19,21] for the MSC problem. The proposed algorithm is iterative in nature and we have suggested an architecture for parallel implementation of the problem. The worst-case time complexities of these algorithms range from $O(n^3)$ to $O(n \log n)$, excepting the work due to Megiddo[9] which has $O(n)$ worst-case time complexity. Using randomized algorithm expected $O(n)$ complexity is achieved by Welzl.[22] The worst-case time complexity of the proposed implementation does not depend on the input size when $O(n)$ processors are used. As can be seen, the individual processors perform very simple computations like finding the Euclidean distance (d_j) here, which is an important phenomenon of neural network models.

It can be conjectured that the present work can be straightaway implemented for any higher dimension. For a similar problem in d dimensions ($d > 2$) one has to simply calculate the Euclidean distances in d dimensions. The complexity of the algorithm remains the same. The extendability of the model in high dimensions is empirically demonstrated here with the help of two applications. Among the above cited algorithms only a few (for example, Welzl's model[22]) are extendable to high dimensions.

The present model is self-organizing in that it learns the center and the radius of the MSC in an unsupervised manner. In Kohonen's self-organizing model,[7] all the processors in the network take part in the weight update process. The present model uses a network of processors where all the processors have fixed weights excepting one i.e. all other processors are fixed (in the bottom and middle layers) while a single floating processor (in the top layer) takes part in the process of positioning itself in the proper location. The fixed processors are used for parallel computations. In Kohonen's model, the "winner" processor moves itself toward the input vector presented while in the present model, the winner processor draws the processor π toward itself (toward its associated input point).

Acknowledgment

The authors thank the reviewers for their valuable comments.

References

1. T. M. Apostol, *Mathematical Analysis*, 2nd edition, Addison-Wesley, 1979.
2. G. A. Carpenter and S. Grossberg, "ART2: Self-organization of stable category recognition codes for analog input pattern," *Proc. IEEE Int. Conf. Neural Networks*, San Diego, CA, II, 1987, pp. 727–736.
3. A. Datta, "Computing minimum spanning circle by self-organization," *Neurocomputing* **13** (1996) 75–83.
4. J. Elzinga and D. E. Hearn, "Geometrical solutions for some minimax location problems," *Transport. Sci.* **6** (1972) 379–394.
5. R. L. Francis and J. A. White, *Facility Layout and Location*, Prentice-Hall, 1974.
6. D. W. Hearn and J. Vijay, "Efficient algorithms for the (weighted) minimum circle problem," *Operat. Res.* **30** (1982) 777–795.
7. T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, 1989.
8. R. P. Lippmann, B. Bold and M. L. Malpass, "A comparison of Hamming and Hopfield neural nets for pattern classification," Technical Report 769, Lincoln Laboratory, MIT, May 1987.
9. N. Megiddo, "Linear time algorithms for linear programming in R^3 and related problems," *SIAM J. Comput.* **12** (1983) 759–776.
10. K. Mehrotra, C. K. Mohan and S. Ranka, *Elements of Artificial Neural Networks*, MIT Press, 1997.
11. R. C. Melville, "An implementation study of two algorithms for the minimum spanning circle problem," *Computational Geomerty*, ed. G. T. Toussaint, North-Holland, 1985, pp. 267–294.
12. K. P. K. Nair and R. Chandrasekaran, "Optimal location of a single service center of certain types," *Nav. Res. Logist. Quart.* **18** (1971) 503–510.
13. S. Pal, A. Datta and N. R. Pal, "A multi-layer self-organizing model for convex-hull computation," *IEEE Trans. Neural Networks* **12**, 6 (2001) 1341–1347.
14. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, NY, 1985.
15. T. Scrrano-Gotarredona, B. Linares-Barranco and A. G. Andreou, *Adaptive Resonance Theory Microchips-Circuit Design Techniques*, Kluwer Academic Publishers, USA, 1998.
16. M. Shamos and D. Hoey, "Closest-point problems," *Proc. 16th Ann. IEEE Symp. Foundations of Computer Science*, Los Angeles, 1975, pp. 151–162.
17. M. Shamos, *Problems in Computational Geometry*, Carnegie-Mellon University, 1977.
18. R. Skyum, "A simple algorithm for smallest enclosing circle," *Inform. Proc. Lett.* **37** (1991) 121–125.
19. R. D. Smallwood, "Minimax detection station placement," *Operat. Res.* **13** (1965) 636–646.
20. J. J. Sylvester, "A question in the geometry of situation," *Quart. J. Math.* **1** (1857) 79.
21. G. Toussaint and B. Bhattacharya, "On geometric algorithms that use the farthest-point Voronoi diagram," Technical Report SOCS-81.3, School of Computer Science, McGill University, 1981.
22. E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," *New Results and New Trends in Computer Science*, ed. H. Maurer, Lecture Notes in Computer Science, Vol. 555, 1991, pp. 359–370.

Amitava Datta received his Master's degree in statistics from the Indian Statistical Institute in 1977. After working for a few years in industries, he joined the Indian Statistical Institute as a system analyst in 1988. Since then he has been working in the fields of image processing and pattern recognition. From 1991 to 1992, he visited GSF, Munich, as a Guest Scientist and worked on a query-based decision support system. He received his Ph.D. degree from the Indian Statistical Institute in 2000.

His current research interests are in neural network based image processing and pattern recognition.

S. K. Parui received his Master's degree in statistics and his Ph.D. from the Indian Statistical Institute in 1975 and 1986, respectively. From 1985 to 1987, he held a post-doctoral position in Leicester Polytechnic, England, working on an automatic industrial inspection project, and from 1988 to 1989, he was a visiting scientist in GSF, Munich, working on biological image processing. He has been with the Indian Statistical Institute from 1987 and is now a Professor. He has published over 60 papers in journals and conference proceedings.

His current research interests include shape analysis, statistical pattern recognition, neural networks and computational geometry.