MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1390 January, 1993

C.B.C.L. Paper No. 079

Some Extensions of the K-Means Algorithm for Image Segmentation and Pattern Classification

Jose L. Marroquin and Federico Girosi

Abstract

In this paper we present some extensions to the k-means algorithm for vector quantization that permit its efficient use in image segmentation and pattern classification tasks. It is shown that by introducing state variables that correspond to certain statistics of the dynamic behavior of the algorithm, it is possible to find the representative centers of the lower dimensional manifolds that define the boundaries between classes, for clouds of multi-dimensional, multi-class data; this permits one, for example, to find class boundaries directly from sparse data (e.g., in image segmentation tasks) or to efficiently place centers for pattern classification (e.g., with local Gaussian classifiers). The same state variables can be used to define algorithms for determining adaptively the optimal number of centers for clouds of data with space-varying density. Some examples of the application of these extensions are also given.

This report describes research done within CIMAT (Guanajuato, Mexico), the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory. This research is sponsored by grants from the Office of Naval Research under contracts N00014-91-J-1270 and N00014-92-J-1879; by a grant from the National Science Foundation under contract ASC-9217041; and by a grant from the National Institutes of Health under contract NIH 2-S07-RR07047. Additional support is provided by the North Atlantic Treaty Organization, ATR Audio and Visual Perception Research Laboratories, Mitsubishi Electric Corporation, Sumitomo Metal Industries, and Siemens AG. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014-91-J-4038. J.L. Marroquin was supported in part by a grant from the Consejo Nacional de Ciencia y Tecnologia, Mexico.

This scheme suffers from some limitations: in the first place, it is difficult to analyze (except in some particular cases [25]), and thus to understand its performance in a precise way; besides, the neighborhood structure is *imposed* rather than found from the data (although some modifications have been proposed to this end [21]), which limits its usefulness in unsupervised clustering tasks.

The above considerations provide the motivation for the present work: it is our purpose to extend the LKMA so that some of its limitations are overcome; specifically, we will propose extended versions of the algorithm that:

i) Allow for a rigorous analysis of its convergence properties.

ii) Work well for clustered data.

iii) Will, if desired, find the centers of the inter–class boundary set.

iv) Adapt the number of centers to the local spatial density of the data.

v) Find the "natural" neighborhood structure for the centers of a data set (i.e., may be used for unsupervised clustering).

The plan of the presentation is as follows: in section 2, we introduce a family of algorithms that include the LKMA as a limiting case, and give a general convergence theorem for this class; also in this section, we present some basic extensions and generalizations needed for finding the centers of the inter–class boundary set, and for adapting the number of centers to the data density. In section 3 we give examples of the application of the extended scheme, specifically, to image segmentation and pattern classification, and finally, in section 4, we present some conclusions and open problems.

2 Extended Local K-Means Algorithms

First, we will introduce a generalization of the LKMA that will help us to understand its convergence properties. Consider again a set $X = \{x_1, \ldots, x_N\}$ of points in $\mathbf{R}^D$, a set of centers $\{m_1, \ldots, m_M\}$, and the weighted error measure:

$$\mathcal{E}_\beta(m) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{M} \|x_i - m_k\|^2 w_{ik}^\beta \qquad (6)$$

where

$$w_{ik}^\beta = \frac{\exp[-\beta\|x_i - m_k\|^2]}{\sum_{j=1}^{M}\exp[-\beta\|x_i - m_j\|^2]} \qquad (7)$$

It is clear that the error measure (2) may be obtained as the limit of (6) as $\beta \to \infty$. Now, $\mathcal{E}_\beta$ is differentiable, and its gradient with respect to m_k is given by:

$$\nabla_k \mathcal{E}_\beta = \sum_{i=1}^{N}(m_k - x_i)W_{ik}^\beta \qquad (8)$$

where

$$W_{ik}^\beta = w_{ik}^\beta(1 + \beta(\sum_j w_{ij}^\beta\|x_i - m_j\|^2 - \|x_i - m_k\|^2))$$

A standard gradient descent procedure for minimizing $\mathcal{E}_\beta$ would therefore take the form:

$$m_k^{(t+1)} = m_k^{(t)} - a_t\sum_{i=1}^{N}(m_k^{(t)} - x_i)W_{ik}^\beta \ . \qquad (9)$$

for some sequence $\{a_t\}$ converging to zero. However, it is often more convenient to adopt a *stochastic gradient descent* algorithm for minimizing $\mathcal{E}_\beta$, that is equivalent to approximating, at each step, the sum on the right side of eq. (9) with just one term, randomly drawn among the N terms. In formula:

$$m_k^{(t+1)} = m_k^{(t)} - a_t(m_k^{(t)} - x_{\xi_t})W_{\xi_t k}^\beta \qquad (10)$$

where $\{\xi_t\}$ is a sequence of random variables which take values on $\{1, 2, \ldots, N\}$. This minimization technique is especially convenient when the data points x_i come one at a time, and it has been extensively used by the neural network community, as a part of the so called "back-propagation" procedure for neural networks training. In the limit as $\beta \to \infty$, (10) becomes precisely the LKMA (3). The convergence of (10) to a local minimum of $\mathcal{E}_\beta$ follows from the following lemma (the proof is given in the appendix; see also [28] for closely related results):

Lemma 2.1 *Let $F(y) : \mathbf{R}^k \mapsto \mathbf{R}$ be of the form:*

$$F(y) = f_0(y) + \frac{1}{N}\sum_{i=1}^{N} f_i(y)$$

where f_i are differentiable functions whose gradient is bounded and satisfies the following Lipschiz condition:

$$\|\nabla f_i(y) - \nabla f_i(y')\| \le M\|y - y'\| \ \ i = 0, \ldots, N$$

for some positive number M. Let $\{y_n\}$ be the sequence

$$y_{n+1} = y_n - a_n(\nabla f_0(y_n) + \nabla f_{\xi_n}(y_n)) + a_n\eta_n \qquad (11)$$

where $\{\xi_n\}$ is a sequence of random variables that take values in $\{1, \ldots, N\}$ with uniform probability distribution, $\{a_n\}$ is a sequence of scalars satisfying:

$$\sum_{i=1}^{\infty} a_n = \infty \ \text{ and } \ \sum_{i=1}^{\infty} a_n^2 < \infty$$

and $\{\eta_n\}$ is a sequence of random variables that is bounded and converges to zero with probability one. Assume that $\{y_n\}$ is bounded and that S is a locally asymptotically stable point of the ordinary differential equation:

$$\dot{y} = -\nabla F \qquad (12)$$

with domain of attraction A_S. Then, if $y_n \in G$ for all n, for some compact domain $G \subset A_S$, $\{y_n\}$ converges to S with probability one.

To satisfy the conditions of the lemma, the sequence $\{a_t\}$ in (10) must be chosen appropriately. One possible choice is, for example,

$$a_t = 1/t \ .$$

else if $h_k > \theta_s N$, generate a new center at a location corresponding to one the data points inside the current Voronoi polytope of center k.

where $\theta_i, \theta_s \in [0,1]$ are two suitably chosen thresholds such that $(\theta_s - \theta_i)N > 1$. Note that this choice for the location of the new centers is necessary to ensure convergence (see below); in practice, however, one may simply locate them at $m_k + \epsilon r$ where r is a random unit vector, and ϵ a positive number small enough, so that the new center is attracted by at least one data point inside S_k.

If the total number of centers change after a sweep, one should reset h_k to zero for all k, and a_t to a_{t-N}, and effect a new sweep until the number of centers stabilize.

It is not difficult to show the convergence of (18) when the lower threshold θ_i is set to 0: in this case, one starts with one processor (center) and successively generate new ones until the Voronoi polytopes corresponding to all centers contain less than $\theta_s N$ data points. Since in this case the number of centers increases monotonically and it is bounded above, (e.g., by the total number of data points), it will necessarily converge to a fixed number M^*.

θ_s is a free parameter that controls the expected average number of points per center, while θ_i controls the variance of this number (a small variance is obtained if $(\theta_s - \theta_i)$ is small). The fact that one has control over the variance means that one can generate more uniform center distributions with this method than with the standard k-means scheme. This, in turn, will usually improve significantly the performance of other procedures that may use these centers, for example, for vector quantization or for function approximation (see section 3).

In practice, it is convenient to set the lower threshold to a positive value to prevent centers to be attracted to single outlier data points, as well as the existence of centers with empty Voronoi polytopes (which may happen due to random initialization, if one starts with more than one center). Note, however that convergence cannot be guaranteed in this case; consider the following example: suppose we have $N = 7$ data points; $N\theta_s = 5$ and $N\theta_i = 4$. It is clear that the number of centers generated by (18) will always oscillate between 1 and 2.

This kind of pathological situations are unlikely to occur in practice though, specially if $(\theta_s - \theta_i)$ is large enough (say, if $\theta_s > 3\theta_i$). A practical way of ensuring convergence in any case is to let θ_i go to zero after a fixed number of iterations.

2.2.2 Boundary Finders

In the case of multi–class data, the augmented state may be used to find the inter–class boundaries directly from sparse data.

Let us assume that we have class–specific centers for classes 1 through $M-1$. Consider the 2–class case first: the augmented state will contain, for each center k, the vector $(m_k, h_k, \hat{h}_{1k})$ (we only have one type of centers in this case). The idea is to constrain the update rule so that a center position is updated approximately the same number of times by data points belonging to each class, so that at all t,

$$\hat{h}_{1k}^{(t)} \approx \hat{h}_{2k}^{(t)} \approx \frac{1}{2} h_k^{(t)}$$

The boundary–finding update rule may thus take the form:

$$\begin{aligned} m_k^{(t+1)} &= m_k^{(t)} + a_t(x_i - m_k^{(t)}), \\ &\quad \text{if } x_i \in S_k^{(t)} \text{ and } \hat{h}_{1k}^{(t)} \leq \frac{1}{2} h_k^{(t)} \\ &= m_k^{(t)} \text{ , otherwise} \end{aligned} \tag{19}$$

where x_i is the data point chosen at time t.

It is clear that with this rule we will have, at any time t, $\mid h_k^{(t)} - 2\hat{h}_{1k}^{(t)} \mid \leq 1$, so that there will be approximately the same number of data points belonging to each class inside the Voronoi polytope of each center, provided that the data density is uniform. In this case, upon convergence, every center k will be located at about the midpoint of the centroids of the sets $\{\mathcal{C}_0 \cap S_k\}$ and $\{\mathcal{C}_1 \cap S_k\}$ where $\mathcal{C}_n$ is the set of data points of class n.

To see why this is true, note that when the update rule (20) reaches its steady state, we must have that

$$E[(x_i - m_k)] = \sum_{c=1}^{2} \sum_{x_i \in S_k} (x_i - m_k) P_k(i,c) = 0$$

where $P_k(i,c)$ is the probability of selecting an example i that is in S_k and belongs to class c and $E[\cdot]$ denotes the expected value. Now,

$$\begin{aligned} P_k(i,c) &= \Pr(\text{select } i \mid i \in \mathcal{C}_c \cap S_k) \Pr(\text{select } \mathcal{C}_c) \approx \\ &\approx \frac{1}{\mid \mathcal{C}_c \cap S_k \mid} \cdot \frac{1}{2} \end{aligned} \tag{20}$$

where $\mid \mathcal{C}_c \cap S_k \mid$ denotes the number of points of class c inside S_k, so that

$$m_k \approx \frac{1}{2} \sum_{c=1}^{2} \frac{1}{\mid \mathcal{C}_c \cap S_k \mid} \sum_{i \in \mathcal{C}_c \cap S_k} x_i$$

It is in this sense that one may say that the centers are representative samples of the inter–class boundary set.

This procedure may be generalized to $Q > 2$ classes, by sampling, for each class $\{1, \ldots, Q-1\}$, the boundary between itself and all the other classes. This however, is not very efficient, since many parts of the boundary will be sampled several times. A more economical sampling may be obtained by defining the sets:

$$T_0^k = \mathcal{C}_k$$

$$T_1^k = \mathcal{C}_{k+1} \cup \mathcal{C}_{k+2} \cup \ldots \mathcal{C}_Q$$

for $k = 1, \ldots Q-1$, and finding, for each k, the centers $\{m_{k1}, \ldots, m_{kM_k}\}$ that sample the boundary between the sets T_0^k and T_1^k using algorithm (20).

It is of course possible (and desirable) to combine this procedure with the one for finding the number of centers in an adaptive way. Figure 3 shows the performance of

Figure 4 around here

A more efficient way of achieving the same result is to define a "pyramid" of processes that operate from coarse to fine scales, and that increase the number of centers at each refining step: one may start with a 3×3 lattice , which after a few iterations of (22) may be refined by adding intermediate centers whose initial positions correspond to the midpoints of existing ones (see figure 5), and repeat the whole procedure recursively until the desired number of centers is obtained. Note that with this procedure, the potentials are always of the form (23) with $k = 1$.

Figure 5 around here

The final configurations obtained in this way (see figure 4–b) are very similar to those obtained with Kohonen's algorithm [15] (which also incorporates long range interactions), but since the neighborhood size remains fixed, the computational complexity is lower, and since the new centers are already close to their correct (globally ordered) positions, the convergence rate is significantly faster.

It is convenient, as in the case of Kohonen's scheme, to mantain a fixed, relatively large a_t in (22) until the final number of centers is reached. At this point, the use of an appropriately decreasing sequence guarantees the final convergence to a local minimum of (21) (see lemma 1). Other examples of the use of this approach will be given in the next section.

3 Applications

In this section we present some applications that illustrate the power of the techniques we have developed

3.1 Edge Detection from Sparse Data

Suppose we have a set of sparse data points $X = \{x_1, \ldots, x_N\}$ inside the unit square Ω in $\mathbf{R}^2$, which may belong to either one of two classes $\{0, 1\}$, and suppose there is a closed region $A \subset \Omega$ whose boundary is a closed, smooth curve, and such that

$$C(x) = 1 \Leftrightarrow x \in A$$

(i.e., all the data points in class 1 are inside A). The problem now is to find a polygonal line (i.e., a *sequence* of points $\{m_1, \ldots m_M\}$) that lies close to the smooth curve that defines the boundary of A.

This may be achieved by combining the adaptive boundary–finding scheme of section 2.2.2 with a prior MRF constraint on the configuration of centers that corresponds to a circular lattice (i.e., a closed polygonal line). In particular, to every clique of 3 neighboring sites (i, j, k) we associate the potential:

$$V_{ijk}(m) = \frac{1}{2}|| - m_i + 2m_j - m_k||^2$$

(note that m_1 and m_M are considered neighbors in a circular lattice).

The combined update rule takes the form:

$$\begin{aligned} m_k^{(t+1)} &= m_k^{(t)} + a_t \left[(x_i - m_k^{(t)}) - \lambda \sum_{C:k \in C} \frac{\partial V_C(m^{(t)})}{\partial m_k} \right], \\ &\quad \text{if } x_i \in S_k^{(t)} \text{ and } \hat{h}_k^{(t)} \leq \frac{1}{2} h_k^{(t)} \\ &= m_k^{(t)} - a_t \left[\lambda \sum_{C:k \in C} \frac{\partial V_C(m^{(t)})}{\partial m_k} \right], \\ &\quad \text{if } x_i \in S_j^{(t)} \text{ and } \hat{h}_j^{(t)} \leq \frac{1}{2} h_j^{(t)}, j \neq k \\ &= m_k^{(t)} \text{ , otherwise} \end{aligned} \tag{24}$$

where x_i is the data point chosen at time t.

An example of the performance of this algorithm is shown in figure 6. Note that since in the final configuration the centers are ordered, this scheme is in fact finding the (discrete) boundary curve from sparse data without interpolating the corresponding surface. In this sense, it may be said that this algorithm finds the initial position, the number of knots and the final configuration of a "snake" [12] that approximates the inter–class boundary.

Figure 6 around here

Figure 7 around here

With straightforward modifications, this scheme may be used for finding: multiple closed boundaries (fig. 7–a and 7–c); open curves that go from one border of the image to another (fig. 7–b and 7–d), etc.

3.2 Local Gaussian Classifiers

Gaussian Classifiers [5] are a well known class of procedures for the segmentation of multi–class, multi–dimensional data. The classification procedure for each specimen $x \in \mathbf{R}^n$ involves the computation of Q quadratic discriminant functions (one for each class) :

$$\begin{aligned} D_k &= (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \mid \Sigma_k \mid \\ &\quad \text{for } k = 1 \ldots, Q \end{aligned} \tag{25}$$

where μ_k is the estimated location of the centroid of class k; $\Sigma_k = [\sigma_{ij}^{(k)}]$ is the estimated $(n \times n)$ covariance matrix and $\mid \Sigma_k \mid$ is its determinant. The specimen is then assigned to the class with the lowest value of D_k.

The "learning" phase consists in the computation of μ and Σ from a set of examples $\{x_1, \ldots, x_N\}$ with known classes $\{c_1, \ldots, c_N\}$:

$$\mu_k = \frac{\sum_{r=1}^N x_r \delta(C(x_r) - k)}{\sum_{r=1}^N \delta(C(x_r) - k)} \tag{26}$$

$$\sigma_{ij}^{(k)} = \frac{\sum_{r=1}^N x_{ri} x_{rj} \delta(C(x_r) - k)}{\sum_{r=1}^N \delta(C(x_r) - k)} - \mu_{k_i} \mu_{k_j} \tag{27}$$

so that it takes only one pass through the data.

and Poggio in [4].

In both cases the training and test error were less than 5% with 3 centers (3 Gaussians per class), and less than 8% with one Gaussian per class.

Figure 8 around here

3.3 Image Segmentation

As a final example, we consider an image segmentation problem that arises in the processing of certain biomedical images: scintigraphic images [11, 19], which are obtained by counting the number of radioactive particles that incide on each cell of a receptor array. The goal of the processing step is to obtain from these measurements an estimate of the radioisotope distribution in specific organs within the human body.

Particle count and radioisotope concentration are related by the Poisson distribution formula; therefore, the processing step consists in the restoration of a piecewise smooth function corrupted by Poisson noise. If it were possible to find the boundaries of the organ in question (e.g., the heart), the problem would reduce to filtering a smooth function within a given domain, for which effective methods are available (for example, Bayesian estimation methods with MRF priors and quadratic potentials to model the smoothness constraint [20]). In the example that we give here, we show that it is possible to adapt the methods that we have presented to classify the pixels of a scintigraphic image of the heart in such a way that one class corresponds approximately to the interior of the organ.

To do this, we will use the following concepts:

We assume that the two classes are characterized solely by the intensity level of the image, i.e., the interior (class 1) has high intensity with respect to the background (class 2). It is assumed that the classes are fuzzy sets [29] with membership functions of the form:

$$\begin{aligned} \phi_1(z) &= \frac{1}{1+\exp[-\beta(z-\theta)]} \\ \phi_2(z) &= 1-\phi_1(z) \end{aligned} \tag{28}$$

where β and θ are positive parameters.

The formulae for the parameters of the discriminant functions of the local Gaussian classifier c are modified in the obvious way:

$$\mu_k^c = \frac{\sum_r x_r \phi_k(z(x_r))}{\sum_r \phi_k(z(x_r))} \tag{29}$$

$$\sigma_{ij}^{(c,k)} = \frac{\sum_r x_{r_i} x_{r_j} \phi_k(z(x_r))}{\sum_r \phi_k(z(x_r))} - \mu_{k_i}^c \mu_{k_j}^c \tag{30}$$

where the sums are taken over the learning domain of the classifier; x_r denotes the coordinates of pixel r of the image, and $z(x_r)$ denotes the value of the observed intensity.

The learning domain of each local classifier is taken as before, as the Voronoi polygon of a center that samples the image in an appropriate way. In this particular example, we are interested in segmenting the left ventricule of the hearth taken from a left anterior oblique projection. From this viewpoint, the ventricule appears as a high intensity "donut" over a dark background (see figure 9–a). Therefore, it is desirable that the centers are located uniformly along a closed, smooth curve that is attracted towards the higher intensity region of the image (note that the quadratic decision surface of each local classifier may be an hyperbola, and therefore, it can adequately segment a region that looks like a band within its domain).

Since a scintigraphic image is actually representing particle counts, we may use update rule (24) directly, considering that at each location x_r there are $z(x_r)$ data points. To get an appropriate behavior for this update rule, however, it is necessary that all the data points are visited in a random order (see lemma 1). To obtain this condition, it is not enough to visit the sites of the lattice randomly; it is also necessary, when, each site i is visited, to "flip a coin", and only update the corresponding center location with probability $P_{update} = z(x_i)/z_{max}$, where $z_{max} = \max_i z(x_i)$.

Figure 9 around here

The results of this procedure applied to the real scintigraphic image of figure 9–a are shown in figure 9–b. The white squares indicate the center locations, and the white line the final compound decision boundary. The threshold θ was obtained as the minimum between the two largest peaks of the global histogram of the image; in the experiments we performed, however, we found that the final results are not very sensitive to the precise value of neither this nor the other parameter (β).

4 Conclusions

In this paper we have analyzed the local K-Means algorithm, and have presented some extensions that increase its range of applicability. Our main contributions are the following:

i) We have established sufficient conditions for the convergence of this algorithm to a (local) minimum of a quadratic distorsion measure (lemma 1). In doing so, we showed that it can be obtained as the limit (as the parameter β becomes large) of a family of algorithms which are closely related to those obtained by minimizing an information distorsion measure. For moderate values of β, we showed that these algorithms can be used for unsupervised learning (clustering) tasks, since they can find a neighborhood structure for the centers that reflects the structure of the data.

ii) We showed that by varying the parameter β and adding a noise term to the update equation, it is possible to improve significantly the performance of the algorithm for clustered data.

iii) We introduced a modification to this algorithm that consists in augmenting the state of each processor (center) so that it keeps track of its own dynamic behavior. With this modification, it is

Theorem A.2 *Let $\{z_n\}$ be a sequence of independent random variables with zero mean. Then, if*

$$\sum_{n=1}^{\infty} E[z_n^2] < \infty$$

the series $\sum_{n=1}^{\infty} z_n$ converges with probability 1.

In order to apply this theorem to the random variable s_i, we first need to check that s_i has zero mean. The probability distribution of the random variables ξ has been assumed to be uniform, and therefore $P(\xi_i) = \frac{1}{N}$. We then have:

$$\begin{aligned} E[s_i] &= a_i E[\nabla F - \nabla f_{\xi_i} - \nabla f_0] = \\ &= a_i [\nabla F - E[\nabla f_{\xi_i}] - \nabla f_0] = \\ &= a_i \nabla F - \frac{1}{N} \sum_{n=1}^{N} \nabla f_i - \nabla f_0] = 0 \ . \end{aligned}$$

Similarly we have

$$E[s_i^2] = a_i^2 E[(\nabla F - \nabla f_{\xi_i} - \nabla f_0)^2] = a_i^2 C(y)$$

for some function $C(y)$. If we assume that

$$\sum_{n=1}^{\infty} a_n^2 < \infty$$

then $\sum_{n=1}^{\infty} E[s_n^2] = C(x) \sum_{n=1}^{\infty} a_n^2 < +\infty$. Therefore the series $\sum_{i=1}^{n} s_i$ converges with probability 1, assumption A4 holds, and lemma 2.1 is proved.

Note that the assumptions about the boundedness of the sequence $\{y_n\}$, and the boundedness of the gradient of the f_i are not as restrictive as they seem, and in practice may be dropped. In fact, we can restrict the sequence $\{y_n\}$ to a compact region $G \subset R^r$, considering, instead of eq. (31), the following sequence:

$$y_{n+1} = \Pi_G [y_n - a_n (\nabla f_0(y_n) + \nabla f_{\xi_n}(y_n)) + a_n \eta_n] \quad (33)$$

where Π_G is the projection onto G. In this case, lemma 2.1 now holds if we substitute ∇F in eq. (12) with $\Pi_G \nabla F$. Choosing G sufficiently large to contain the fixed points of interest, the conclusion of the theorem is practically unchanged. Moreover, since now the sequence is restricted to the compact set G, the derivatives of the f_i are certainly bounded on G, being continuous, and therefore the assumpions of boundedness of the gradient of the f_i can be dropped.

Acknowledgements We are grateful to Tomaso Poggio and John Harris for useful discussions and for a critical reading of the manuscript. We also thank Roberto Brunelli for providing us the gender classification data set.

References

[1] M.R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[2] M. Benaim and L. Tomasini. Competitive and self-organizing algorithms based on the minimization of an information criterion. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 391–396. Elsevier Science Publishers B.V. (North-Holland), 1991.

[3] J. Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *J. Royal Statist. Soc.*, 36(B):192–326, 1974.

[4] R. Brunelli and T. Poggio. HyperBF networks for gender recognition. In *Proceedings Image Understanding Workshop.* Morgan Kaufmann, San Mateo, CA, January 1992.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[6] R. Durbin and G. Mitchison. A dimension reduction framework for understanding cortical maps. *Nature*, 343:644–647, February 1990.

[7] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, 1984.

[8] A. Gersho. On the structure of vector quantizers. *IEEE Transactions on Information Theory*, 28(2):157–166, 1982.

[9] A. Gersho and R.M. Gray. *Vector quantization and signal compression.* Kluwer Academic Publishers, Boston, 1991.

[10] McLachlan G.J. and Basford K.E. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker, New York and Basel, 1988.

[11] T. Inouye. Computer processing of scintillation camera gas. *Nucl. Instrum. Meth.*, 124:215–219, 1975.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

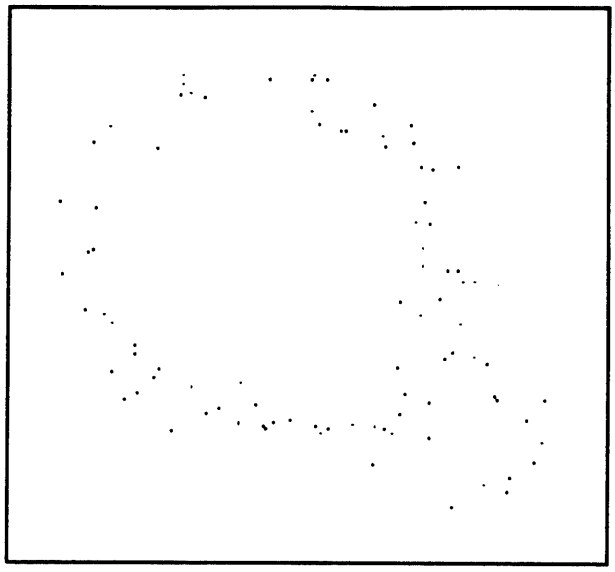
[13] R. Kinderman and J.L. Snell. *Markov Random Fields and their applications.* Amer. Math. Soc., Providence, RI, 1980.

[14] T. Kohonen. *Self-organization and associative memory.* Springer-Verlag, Berlin, 1984.

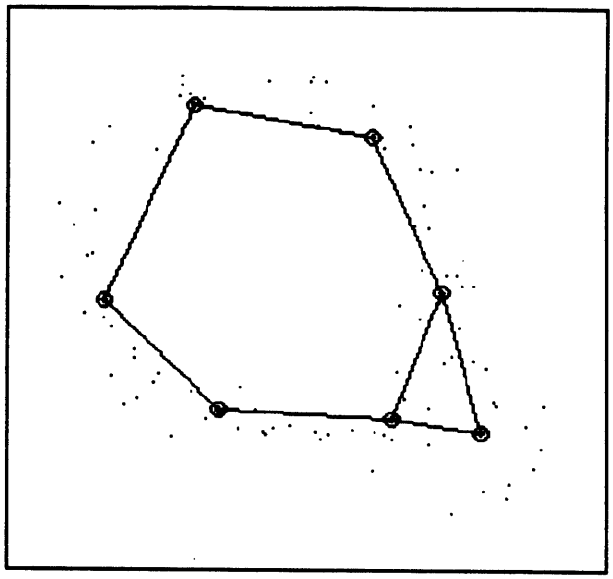
[15] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[16] H. Kushner and D. Clark. *Stochastic approximation methods for constrained and unconstrained systems*, volume 26 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 1978.

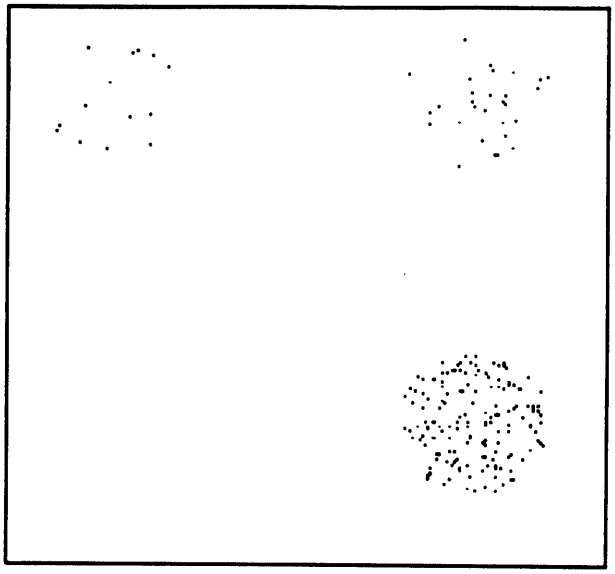
[17] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Proceedings of the IEEE*, Com-28(1):84–95, January 1980.



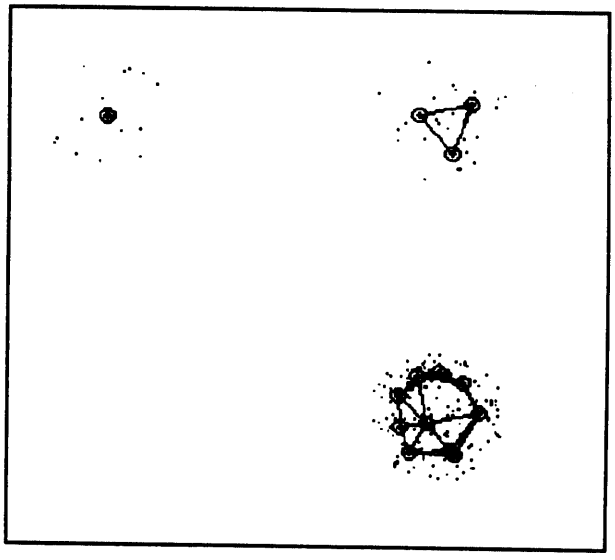
(a)



(c)



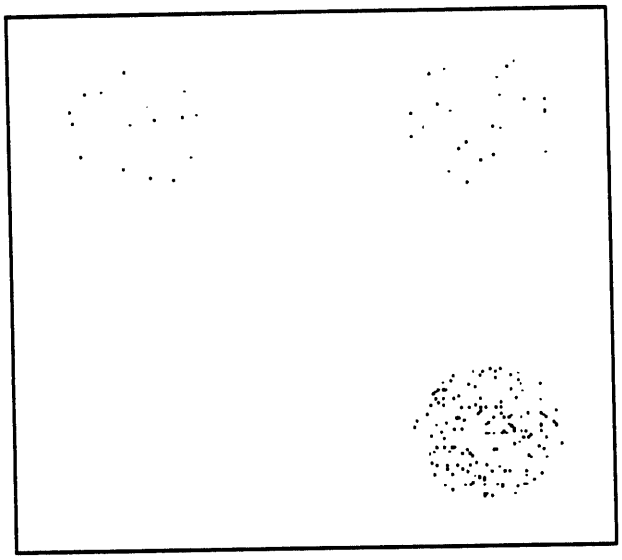
(b)



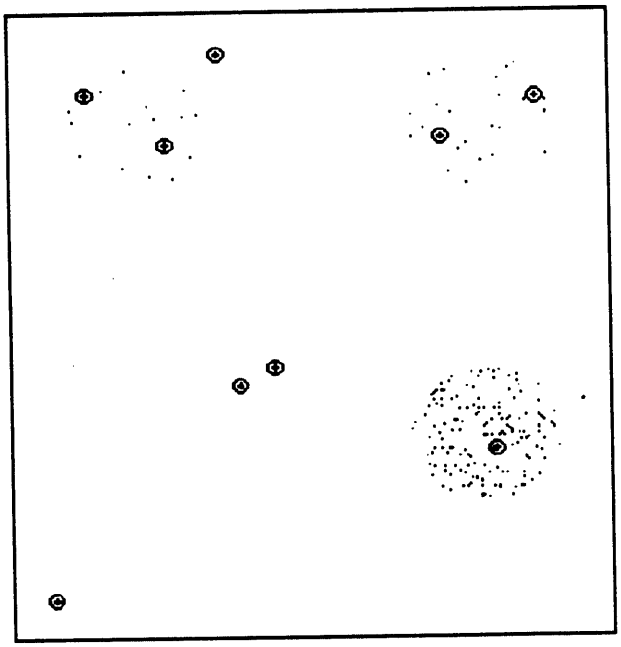
(d)

Fig. 1

(a)



(b)



(c)

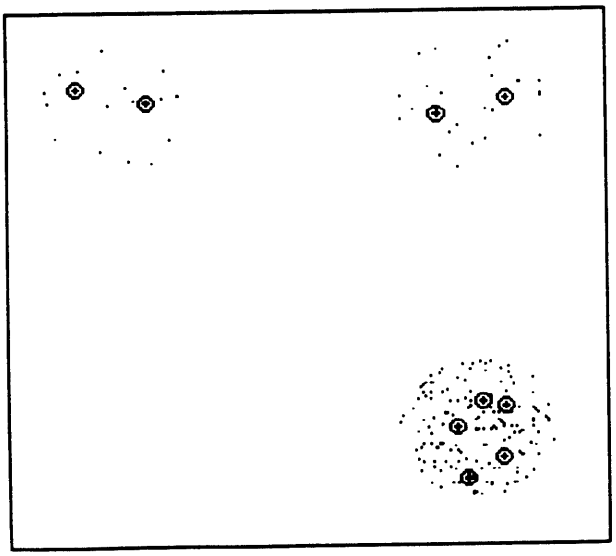
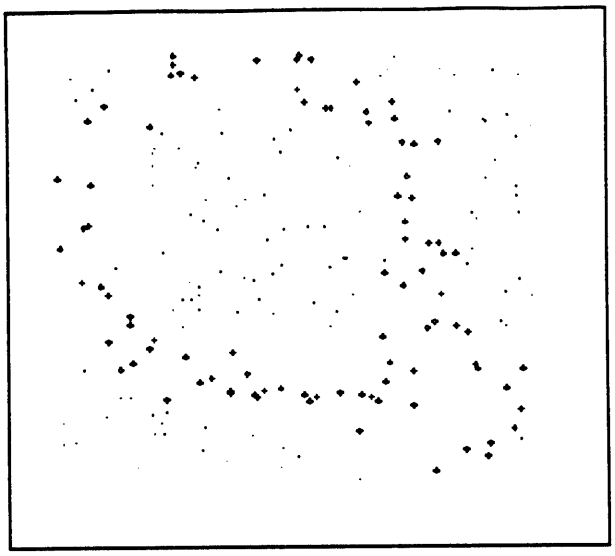
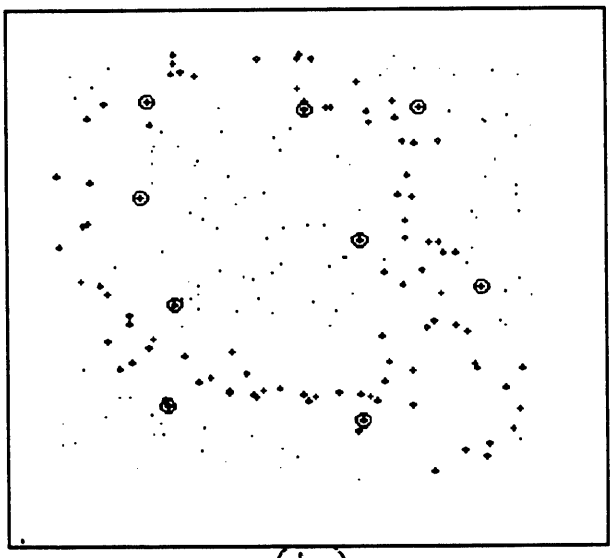


Fig. 2

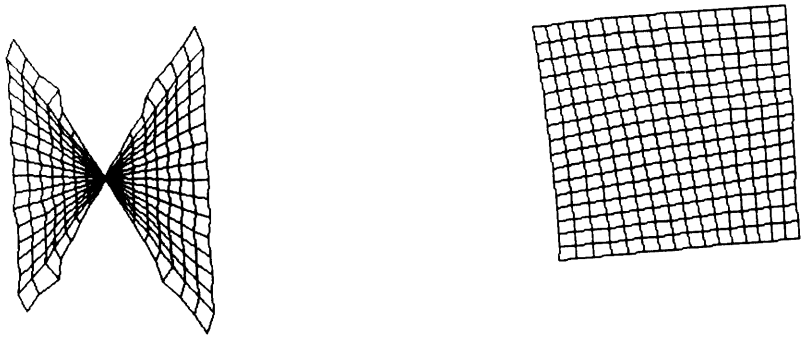


(a)



(b)

Fig. 3



(a) (b)

Fig. 4

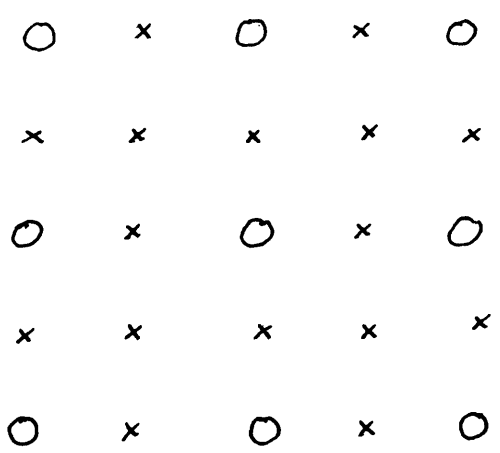
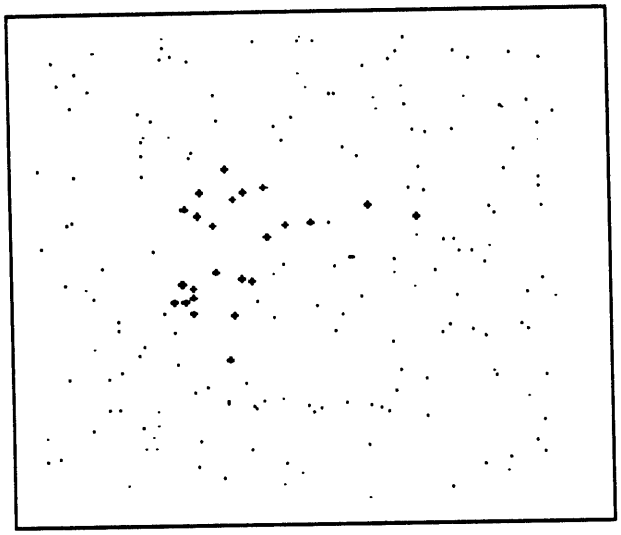
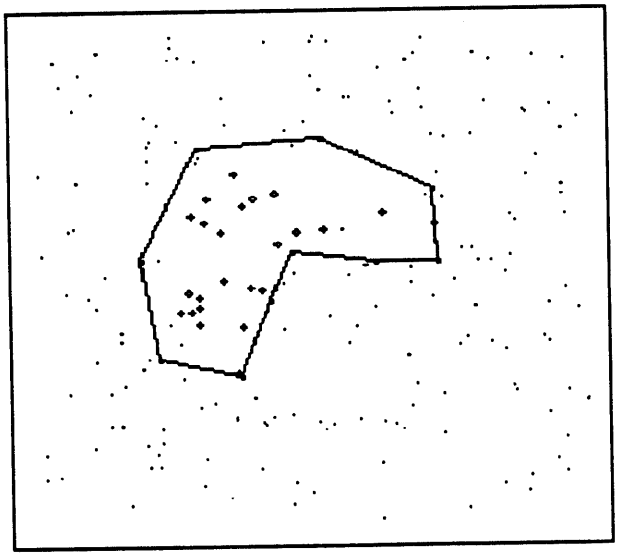


Fig. 5

(a)



(b)



(c)

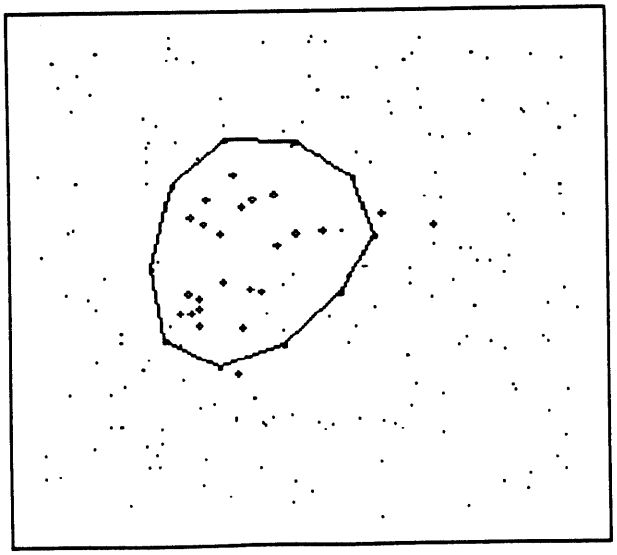
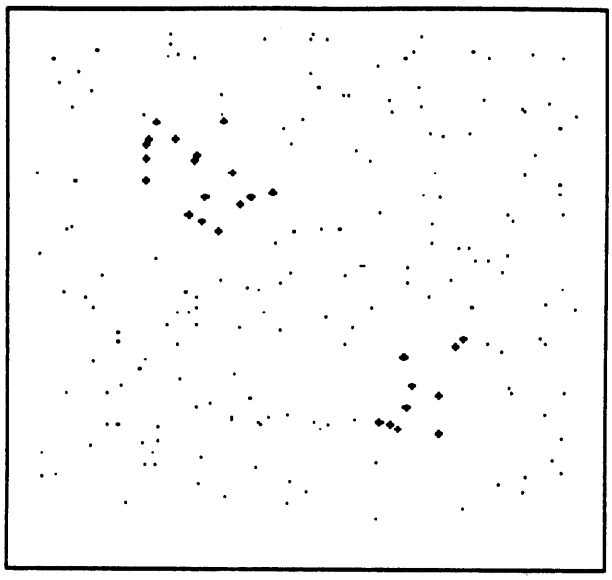
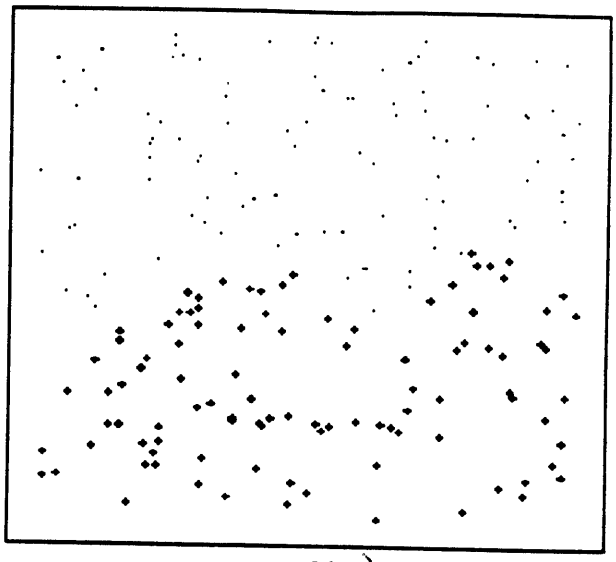


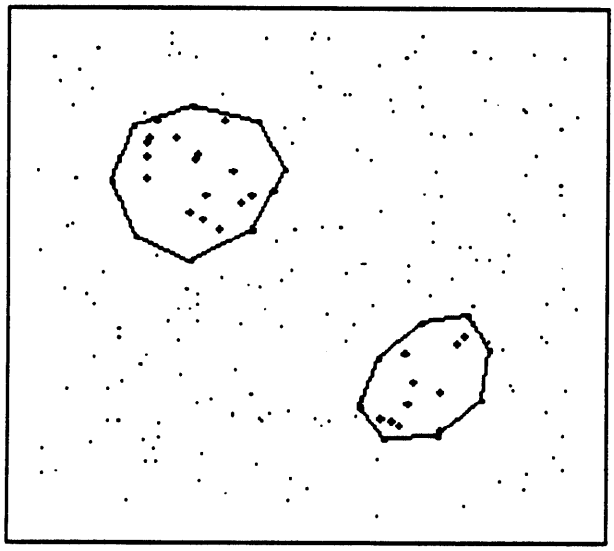
Fig. 6



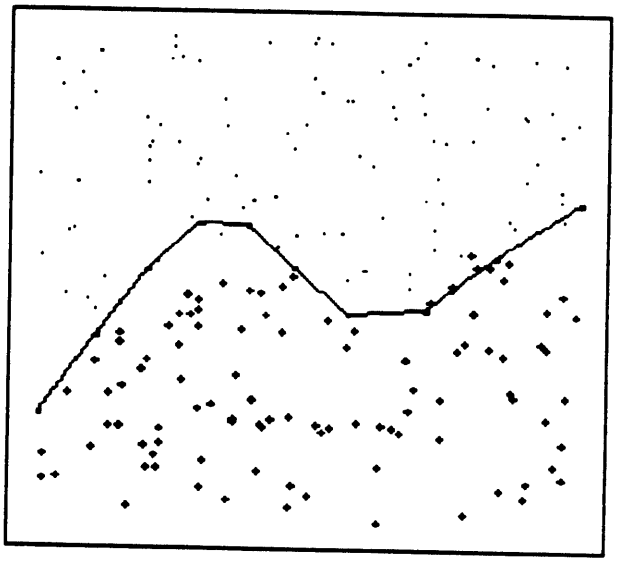
(a)



(b)



(c)



(d)

Fig. 7

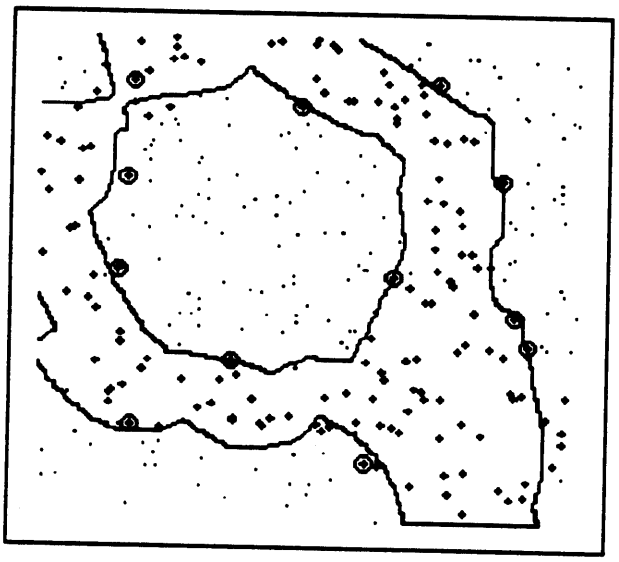
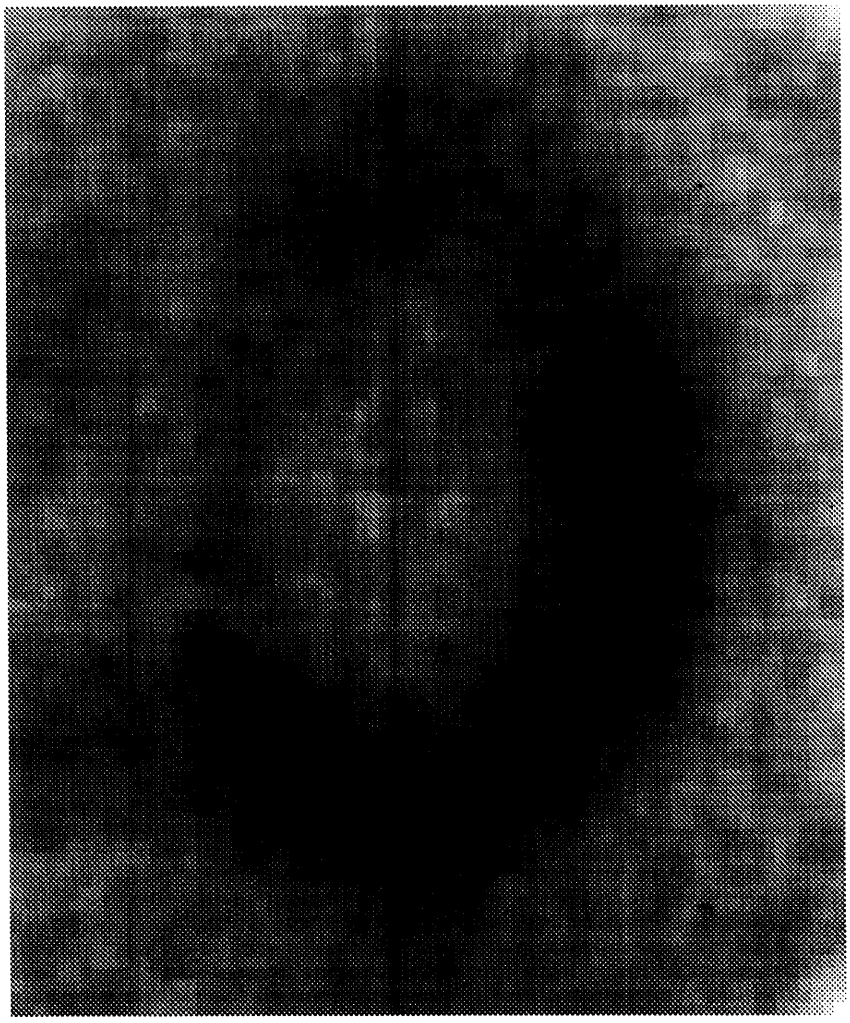
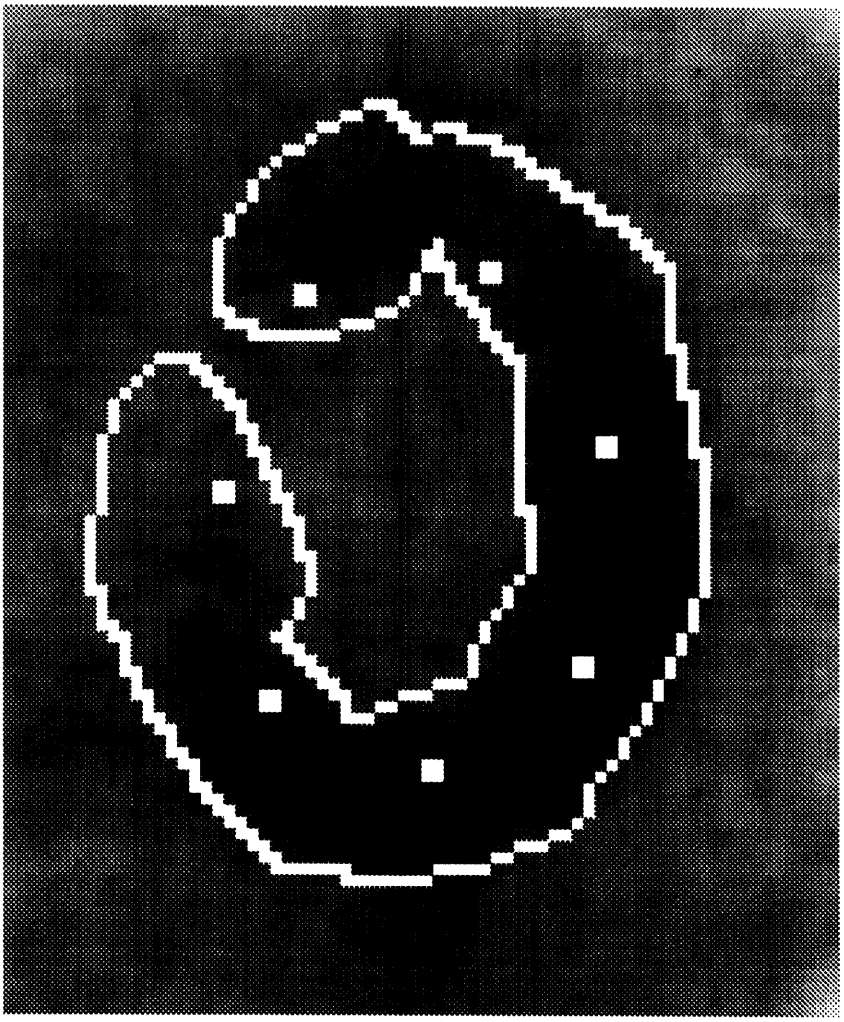


Fig. 8



(a)

Fig. 9



(b)

Fig. 9