TITLE:	Pattern recognition letters [electronic re
VOLUME/ISSUE/PAG	12/7 409-412
DATE:	1991
AUTHOR OF	D. Chetverikov
TITLE OF	Fast neighborhood search in planar point s
ISSN:	0167-8655
CALL NUMBER:	http://rapidill.org/redir.ashx?id=MzA0OTk(
DELIVERY:	Ariel: 129.82.28.195
REPLY:	Mail:

Pattern Recognition Letters 12 (1991) 409–412
North-Holland

July 1991

Fast neighborhood search in planar point sets

Dmitry Chetverikov

Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, P.O. Box 63, H-1302 Hungary

Received 14 November 1989
Revised 30 January 1991

Abstract

Chetverikov, D., Fast neighborhood search in planar point sets, Pattern Recognition Letters 12 (1991) 409–412.

A simple and practical method is proposed that facilitates fast spatial neighborhood operations in a non-ordered point list containing coordinates of a planar point set. Given a rectangular neighborhood, $O(Nq)$ operations are needed to scan the neighborhoods of N points in the list, where q is the average number of points in the neighborhood. The method uses a data structure that requires $O(N)$ bytes of storage. The complexity of the algorithm that structures the data is $O(N)$.

Keywords. Spatial neighborhood search, planar point sets.

1. Introduction

The problem of spatial neighborhood search in a planar point set given by a list of the point coordinates arises in various tasks of computer vision and pattern recognition. For many kinds of local operations, one has to find those points in the list that are within a window centered on a given point. A typical example is matching of two consecutive frames for motion analysis when the feature point in the second frame corresponding to a point in the first frame is searched within a neighborhood determined by the maximum speed (e.g., Ballard and Brown (1982)).

A number of methods for neighborhood search have been published in literature. Preparata and Shamos (1988) present a selection of sophisticated algorithms with excellent asymptotical behavior. In practice, however, one often needs a simple and practical solution that offers a reasonable execution time. In this paper, a way of structuring the data is introduced which provides a simple and fast algorithm that is particularly suitable for operations in rectangular neighborhoods of points in a spatially non-ordered point list. If an upper estimate of the maximum distance between neighboring points is available, the proposed method can be used to find the nearest neighbor.

2. Basic idea

Let A be a point belonging to a planar point set S. For the sake of simplicity, consider a square window W_2 centered on A. W_2 is the neighborhood of A that contains those points of S that should be accessed. The method is readily generalized to rectangular neighborhoods as well. Let the size of W_2

0167-8655/91/$03.50

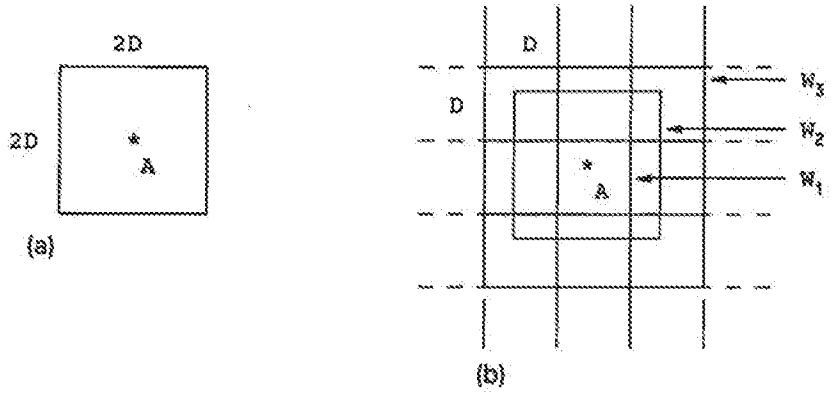

Figure 1. (a) Point A and its $2D\times 2D$ size neighborhood W_2. (b) W_1 is the box containing point A; W_3 is the $3D\times 3D$ size window that contains W_2.

be $2D\times 2D$. (See Figure 1a.) Consider then a set of $D\times D$ size square windows covering S. These windows will be called *boxes*. Assume there are M rows of boxes with M boxes in each row. Denote by W_1 the box containing point A. (See Figure 1b.) Let the $3D\times 3D$ size window formed by W_1 and its eight $D\times D$ size neighbors be denoted by W_3. The points lying within W_2 must be within W_3 as well. Thus, by restructuring the original non-ordered point lists so as to conform to the partition of the plane into the boxes one can reduce the search set to those points that lie within those boxes that form W_3. All one has to do is to organize the data in a way suitable for placing the points into the appropriate boxes. This latter operation will be called *boxing*. The neighborhood search then simplifies to the retrieval of the points stored in the corresponding nine boxes and selecting those of them that are within W_2.

3. Boxing

Let points $A_i\in S$, $i=0,1,\ldots,N-1$, form the original spatially non-ordered list of points L_1. Denote by (X_i, Y_i) the coordinates of A_i. The boxing data structure consists of a rearranged point list L_2 and an index matrix $I_{m,n}$ whose elements correspond to boxes: $m,n=0,1,\ldots,M-1$. To be able to treat all boxes in a uniform manner, $I_{m,n}$ is formally extended by adding one more element $I_{M-1,M}$. The items of L_2 are the N coordinate pairs placed in the order of boxes, i.e., those points that are within the first box are followed by those lying within the second box, etc. $I_{m,n}$ contains integers indicating the beginnings of the boxes in L_2 so that $I_{m,n+1}-I_{m,n}$ is the number of points in the (m,n)th box. Figure 2 illustrates the boxing data structure. The following simple algorithm inputs L_1 and fills $I_{m,n}$ and L_2.

Boxing Algorithm

Pass 1. Computing $I_{m,n}$.

Step 1. Allocate an $M\times M$ size accumulator array $B_{m,n}$ which is to contain the number of points in each box.

Step 2. Scan L_1 and fill $B_{m,n}$. A point A_i is within the box whose indices are as follows:

$$m_i=\left\lfloor\frac{Y_{\max}-Y_i}{D}\right\rfloor,\qquad n_i=\left\lfloor\frac{X_i}{D}\right\rfloor, \tag{1}$$

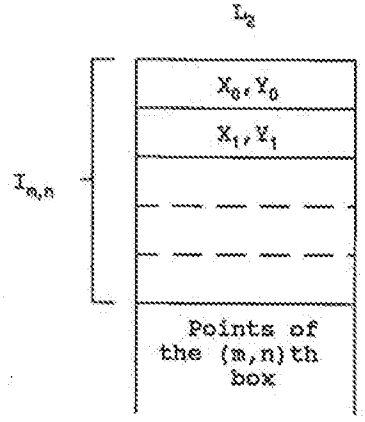

Figure 2. The boxing data structure.

where Y_{max} is the maximum value of Y and $\lfloor \cdot \rfloor$ means truncation.

Step 3. Fill $I_{m,n}$ using the following recursive formula:

$I_{0,0}=0$.

For all $(m,n)\neq(0,0)$,

$$I_{m,n}=\begin{cases} I_{m,n-1}+B_{m,n-1}, & \text{if } n>0; \\ I_{m-1,M-1}+B_{m-1,M-1}, & \text{otherwise.} \end{cases} \quad (2)$$

Then formally extend $I_{m,n}$ by setting $I_{M-1,M}=N$.

Pass 2. Filling L_2.

Step 1. For all m and n, set $B_{m,n}=0$.

Step 2. Scan L_1 again. Use (1), $I_{m,n}$, and $B_{m,n}$ to fill L_2. In L_2, the first point of the (m,n)th box is indexed by $I_{m,n}$ while the address of the subsequent points is controlled via $B_{m,n}$ whose value is incremented each time a new point enters the box.

The complexity of the boxing algorithm is $O(N+M^2)$. The storage requirement is $O(N)$ bytes for L_2 and $O(M^2)$ bytes for $I_{m,n}$. Normally, $N>M^2$ and both the complexity and the storage required are proportional to the number of points N.

4. Neighborhood search

Once the point set S has been boxed, accessing the points lying within the $2D\times 2D$ size neighborhood W_2 of an arbitrary point A_i means retrieving the points stored in W_3 and selecting those of them that are within W_2. The steps of the access procedure are given below.

Access Procedure

Step 1. Compute by formula (1) the indices m_i, n_i of the box that contains A_i.

Step 2. Use the boxing data structure to retrieve the points B_j, $j=0,1,\ldots,q_i-1$ lying within nine boxes (m_i,n_i), $(m_i\pm 1,n_i\pm 1)$, $(m_i,n_i\pm 1)$, and $(m_i\pm 1,n_i)$. (Here q_i denotes the number of points in the nine boxes.) In L_2, $I_{m,n}$ indexes the first point in the (m,n)th box, while the number of points in the box is given by

$$\text{or} \quad \begin{cases} I_{m,n+1}-I_{m,n}, & \text{if } n<M-1, \\ I_{m+1,0}-I_{m,M-1}, & \text{if } n=M-1. \end{cases} \quad (3)$$

Step 3. Select those points B_j^* in the nine boxes that are within the $2D\times 2D$ size neighborhood of A_i:

$$|X_j^*-X_i|<D, \qquad |Y_j^*-Y_i|<D. \quad (4)$$

The access procedure requires $O(q_i)$ operations. Therefore, $O(Nq)$ operations are needed to access neighborhoods of all N points in S, where q is the average number of points in the neighborhood.

Concluding remarks

The boxing technique can be generalized not only to rectangular neighborhoods, but to neighborhoods of arbitrary shape as well. The boxing procedure does not change. The steps 2 and 3 of the access procedure, however, should be modified in accordance with the definition of the neighborhood.

The proposed technique is appropriate only for point sets of a low dimensionality. If d is the dimensionality of point sets, both the accumulator array B and the index array I require $O(M^d)$ space, while the access procedure requires 3^d boxes to be searched.

The presented way of structuring the data is a general tool for spatial neighborhood search in point lists. In particular, it can be used to find the nearest neighbor of a point if an upper estimate of the maximum distance between a point and its nearest neighbor is available. In this case, the computational requirement of the approach as compared to the one of conventional NN-classifiers depends on the spatial distribution of points and the accuracy of the maximum distance estimate. For example, the fast NN-classifier by Sethi (1981) needs $O(N)$ distance calculations and $O(N^2)$ comparisons during the preprocessing, while our approach needs $O(N)$ arithmetic operations. To classify an unknown point, Sethi's algorithm requires 3 distance calculations and $O(\log N)$ comparisons. In the proposed method, $O(q)$ distance calculations and $O(q)$ comparisons are necessary.

Since $q \ll N$, in certain circumstances application of the boxing technique may be preferable.

Acknowledgement

The work was supported by the Hungarian National Science Foundation Grant OTKA 1067.

References

Ballard, D.H. and C.M. Brown (1982). *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ.

Preparata, F.P. and M.I. Shamos (1988). *Computational Geometry*. Springer, New York.

Settii, I.K. (1981). A fast algorithm for recognizing nearest neighbors. *IEEE Trans. Syst. Man Cybernet.* 11, 245–248.