An Approximate Nearest Neighbours Search Algorithm Based on the Extended General Spacefilling Curves Heuristic

Juan-Carlos Pérez[1] **Enrique Vidal** [2]

[1] Dpt. de Informática de Sistemas y ComputAdores (DISCA), Universidad Politécnica de Valencia. C. de Vera s/n 46071, Spain. jcperez@disca.upv.es

[2] Dpt. de Sistemas Informáticos y Computación (DSIC) Universidad Politécnica de Valencia. C. de Vera s/n 46071, Spain. evidal@iti.upv.es

Abstract

In this paper, an algorithm for approximate nearest neighbours search in vector spaces is proposed. It is based on the Extended General Spacefilling Curves Heuristic (EGSH). Under this general scheme, a number of mappings are established between a region of a multidimensional real vector space and an interval of the real line, and then for each mapping the problem is solved in one dimension. To this end, the real values that represent the prototypes are stored in several ordered data structures (e.g. b-trees). The nearest neighbours of a test point are then efficiently searched in each structure and placed into a set of candidate neighbours. Finally, the distance from each candidate to the test point is measured in the original multidimensional space, and the nearest one(s) are chosen.

1. Introduction

The search for the nearest neighbours to a given point from a set of N prototypes in a real vector space of dimension n, is a task common to a large number of applications. In the field of Pattern Recognition this problem appears frequently, for instance in the estimation of probability densities using the k-nearest neighbours rule [Fukunaga,90]; in some methods of multivariate function learning [Poggio,90] [Omohundro,90]; in most clustering techniques in vector spaces, also known as vector quantization [Jain,88]; etc. A large (and increasing) number of other problems inside and outside that field can benefit from an efficient method for the search of nearest neighbours.

The temporal cost of the trivial solution to this problem, an exhaustive search in the set of prototypes, is in $O(nN)$ if we assume, as we will from now on, that the temporal cost of the distance computation is proportional to the dimension of the space, n. The spatial cost is also in $O(nN)$, since this method does not require the construction of any special data structure. When N is very large, the exhaustive search method can be impractical.

A number of algorithms able to perform more efficiently than the exhaustive search have been proposed. The ideas applied to obtain lower temporal costs are diverse. Most algorithms for vector spaces [Friedman,77], [Kim,86] , [Yao,85] are directly based on the construction of *kd*-trees. Another populated class of techniques is composed by those which take advantage of certain properties of the metrics, particularly the Triangle Inequality, to readily discard prototypes that cannot possibly be the nearest to the test point, therefore limiting the extension of the

search. Many of these methods do not even require that the objects be represented in a real vector space. Among them, we can cite the ones presented in [Sethi,81], [Kalantari,83], [Vidal,86] and [Shasha,90]. Finally, some algorithms cannot be classified into any of these groups and propose methods based on different concepts [Friedman,75], [Yunck,76], [Bentley,80] and [Murphy,90].

In many cases, an absolute guarantee of finding the real nearest neighbour to the test point is not necessary. Owing to that, some algorithms of *approximate* nearest neighbour search have been proposed [Miclet,83], [Bern,93]. The method presented here falls into this category. In [Perez,94] and [Perez,97], a review of all the cited algorithms is presented, with special emphasis on the costs involved and in the applicability of the different techniques.

2. The Extended General Spacefilling Heuristic

In [Bartholdi,88], the General Spacefilling Heuristic is presented as a generic procedure, useful in many combinatorial optimization problems where the data-set is composed by points in a multidimensional space and the solution involves the preservation of some notion of proximity among them. The method consists on mapping each point into the real line (using a n-D to 1-D mapping based on the notion of spacefilling curve) to get an easier or more efficiently tractable problem in one dimension. Once solved the reduced problem, the points, along with the solutions found, are returned to the original space.

Spacefilling Curves, also known as Peano Curves, are a class of fractal curves originally described by G. Peano and D. Hilbert more than a century ago [Peano,1890], [Hilbert,1891]. Simply stated, a Spacefilling Curve is a recursive structure that completely fills a region S with positive content in $\mathbb{R}^n$. This means that every point in S belongs to the curve. In [Bartholdi,88] or in [Perez,94] and [Perez,97], formal definitions of a spacefilling curve are presented, along with the details of the computation of Sierpinski's Spacefilling Curve, $\psi(t)$, introduced in [Sierpinski,1912].

The continous nature of spacefilling curves confer quite remarkable properties to mappings based on them. Of particular interest here is the *preservation of a notion of proximity,* such that any pair of points which are close in the Real Line are guaranteed to come from nearby points in the original (multidimensional) space. Unfortunately, the converse is not true in general, since neighbours in the space do not always end as neighbours in the Real Line (and, additionally, distant points in the space are not necessarily mapped into distant points in the Real Line). Details on this and other properties of spacefilling curves, as well as a brief survey of their applications appear in [Perez,94] and [Perez,97].

The first instantiation of the General Spacefilling Heuristic is proposed by its authors [Bartholdi,83], and deals with finding an approximate solution to the Travelling Salesman Problem. In that work and in [Imai,86], the same method is used to obtain an (also approximate) solution to the Minimum-Weight Matching Problem on the plane, defining the pairs of matching points as the consecutive pairs of points in the Travelling Salesman Path.

The methodology described is simple and extremely fast in general but, as a consequence of the fact that the spacefilling mapping function, ψ^{-1}, cannot map consistently any connected region of the space into a (connected) interval of the Real Line, often gives rise to intolerable discontinuities in the resultant mapping. These

discontinuities are specially remarkable in certain areas which we will call *conflictive regions.*

As a solution to that problem, the **Extended General Spacefilling Heuristic** was proposed in [Perez,94], [Perez,97] and [Perez,98]. It consists, in essence, on applying the spacefilling mapping r times, where r is a constant to be defined later. Each application of the mapping is fed with the original data affected by a distinct transformation. As a consequence, the areas introducing major distortions (the most conflictive regions) can affect points in different regions of the original representation space. Each tandem transformation-mapping will establish a distinct unidimensional submodel of the problem. All these submodels, adequately combined, will configure the final model.

The sequence of operations that should be used to apply this method to a set of N n-dimensional points is shown in the following algorithm:

1- Normalize the set of points to fit into the unit Hypercube $[0,1]^n$.

2- For each submodel i:

  2.1- For each n-dimensional point of the problem, apply transformation T_i.

  2.2- For each transformed point, map it into $[0,1]$, applying a spacefilling mapping (for instance, ψ^{-1}).

  2.3- Solve the unidimensional problem.

4- Combine the solutions of the different submodels to obtain a global solution. In some cases, this combination can include a weight factor for each partial solution, according to a measure of its quality.

The number of submodels r can be as high as necessary to obtain a satisfactory solution to the problem. It seems reasonable to tie proportionally this parameter to the dimension of the input space n. Using, for instance, $r=n$ submodels, n real values are obtained (one per submodel) from input vectors with n components. This suggests a certain degree of conservation of the original information. Values of $r=2n$ or $r=3n$ are also typical in cases where better results are needed and values of $r=n/2$, $r=n/3$, etc. can be used if lower costs are more important.

In [Perez,98], an application of the heuristic to learning of multivariate function mappings based on examples is presented, and other possible applications are suggested.

3. Approximate Nearest Neighbour Search Algorithm

The algorithm proposed is a direct application of the Extended General Spacefilling Heuristic. The instantiation of that general scheme is as follows:

- The vectors received by the first stage of the extended heuristic (normalization and transformations) are directly the prototypes.

- The unidimensional values that correspond to each submodel of the extended heuristic are sorted and stored into a vector, inserted into a b-tree, or inserted into an indexed table.

- In the operating phase (when a test point p is presented to the system), the usual method of mapping the point onto the Real Line is employed, with a different transformation for each of the r submodels. The b nearest neighbours of the unidimensional value *in the Real Line,* for each submodel, can be readily found using a dichotomic search in the vector, a conventional search in the b-tree, in $O(\log N + b)$, or even a direct access operation to a table indexed by those values, in $O(b)$. The union of the r sets of b neighbours obtained from the r submodels produces a set of size $\leq rb$ wich will be exhaustively searched to find the nearest neighbour to p in the original multidimensional space using the desired distance function in that space.

More formally, given a set of N prototypes $x_i \in \mathbb{R}^n$ $1 \leq i \leq N$, the preprocessing is as follows:

- Defined a number of submodels, r, compute for each submodel the N transformed and normalized vectors:

$$x'_{ij} = T_j(x_i), \quad 1 \leq i \leq N, \quad 1 \leq j \leq r.$$

where $T_j(\cdot)$ represents a sequence of operations *rotation-normalization-transformation* defined in the heuristic.

- Compute the unidimensional values associated to those vectors, via a Spacefilling Mapping:

$$p_{ij} = \psi^{-1}(x'_{ij}), \quad 1 \leq i \leq N, \quad 1 \leq j \leq r.$$

- For j in $[1, 2, \ldots r]$, sort each set of real values $\{p_{1j}, p_{2,j}, \ldots, p_{Nj}\}$ in ascending order or insert them into a suitable data structure with efficient search capabilities.

Given a test vector $x \in \mathbb{R}^n$, which nearest neighbour is to be found, the operating phase consists on the following steps:

- Normalize and transforrm x for each submodel:

$$x'_j = T_j(x), \quad 1 \leq j \leq r.$$

- Compute the corresponding unidimensional values using the chosen spacefilling curve:

$$p_j = \psi^{-1}(x'_j), \quad 1 \leq j \leq r.$$

- For j in $[1, r]$, locate via a dichotomic search or oher suitable method the b nearest neighbours of p_j in $\{p_{1j}, p_{2,j}, \ldots, p_{Nj}\}$. The distance in the line is defined as a cyclic metric: $\Delta(p,p') = min\{|p-p'|, 1-|p-p'|\}$. Actualize the set B,

initially empty, with the union of all the sets of b points found for each submodel j.

- Compute the distance in the multidimensional space using the originally considered metric between p and each element in B, and select the nearest neighbour.

Obviously, an extension to the search of the k nearest neighbours is trivial. It is only necessary to modify the last step of the operating phase to find the k nearest prototypes.

The preprocessing cost is in $O(nN + N\,logN)$ per submodel, where the first term corresponds to the normalization and the transformations, and the second one is the cost of the spacefilling mapping and the sort of the values in the Real Line or the insertion into a b-tree. The number of submodels r has been considered proportional to the dimension, n. Therefore, the total preprocessing cost is in $O(n^2N + nN\,logN)$. The asymptotic spatial cost is not higher than that necessary to store the original points, namely in $O(nN)$. If an indexed table is used instead to store the points in each submodel, using the real value as an index, then the spatial cost depends on the number of pre-allocated buckets necessary to avoid collisions, which cannot be easily estimated in general, but the average preprocessing cost is reduced to be in $O(n^2N)$, since insertion can be done in constant average time.

The search of the nearest neighbour to a previously unknown point (operating phase) involves a cost per submodel in:

- $O(n)$ for the normalization and transformation of the point.
- $O(1)$ for the computation of the spacefilling mapping (given a suitable limitation to a constant in the maximum dimension of the computation).
- $O(logN)$ for the search if a sorted vector or a b-tree is used, or $O(1)$ if an indexed table is employed.
- $O(nb)$ for the computation of the distances from the point to the b nearest neighbours in the Real Line, and the selection of the nearest one.

The temporal cost is therefore, considering a number of submodels r proportional to n and a constant b, in $O(n^2 + n\,logN)$ for a sorted vector or a b-tree, or $O(n^2)$ for an indexed table. Of course, the search in the different submodels can be very easily performed in parallel, even using weakly coupled processors such as networked machines, obtaining then a search cost in $O(n + logN)$ for a sorted structure or in $O(n)$ for an indexed one.

In applications with very large data-sets, where the need for prototypes to be inserted and deleted at any time exists and the use of disk-based storage is mandatory, the best data structures for the submodels are the b-trees. In those cases, the retrieval of the b nearest real values in each submodel can be done very efficiently even for large values of b, and the search in all submodels can be performed in parallel if each tree is stored in a different disk. This can be true even for single processor configurations, since the seek times of the succesive leaves of the b-trees are the dominant factor in the total search time.

4. Experiments

In order to establish the influence of the number of prototypes N and the dimensionality of the space n in the results of the method, a number of experiments have been performed. In the first experiments, a set of points drawn from a uniform random distribution in $[0,1]^2$ was used, with values of N in *[8,000 ... 700,000]* and a test set size of $N'=1000$ points. In figure 1, the results are shown. No decrease of the search effectiveness due to the size of the set of prototypes is evidenced.

For each test point, a list of prototypes ordered from smaller to larger distance to the test point has been constructed (through exhaustive search) and the sequence number, in that list, of the point selected as the nearest neighbour by the approximate method, has been determined. If repeated values existed in the sequence, the number chosen would be, obviously, the smallest. We will call this number *rank number* of the approximate point. The *average rank number*, as well as the *worst rank number*, for the N' test points, have been calculated for each experiment. The ranges and mean values of these terms, for sets of 4 experiments with different random initializations, have been represented in the figures.

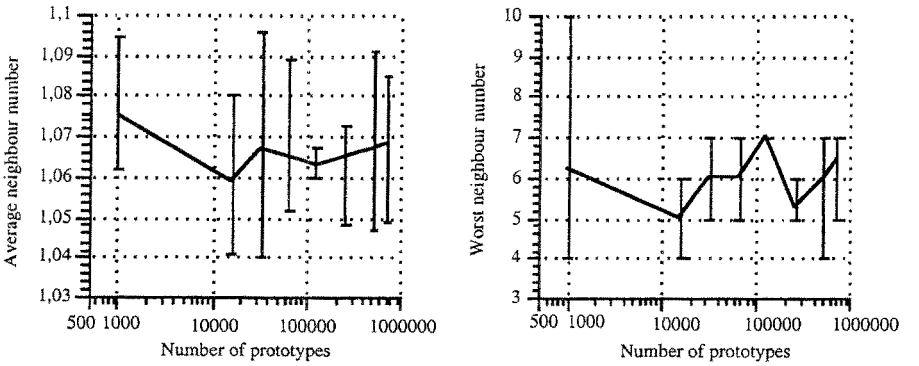

Figure 1: Average and worst rank number of the approximated nearest neighbour found with the proposed method (a value of *1* means finding the nearest neighbour, *2* means finding the second, etc.) against the size of the set of prototypes, in a uniform random distribution in $n=2$ dimensions, with $r=2$ and $b=4$. The average values are shown, along with the maximum and minimum among four experiments performed using different random parameters for the transformations.

In the first experiment, the value assigned to b was *4* in order to make the error rates larger, since in such a low dimension the errors obtained using larger values of b are too low to be useful when represented in the graphs. In the rest of the experiments, a value of $b=16$ was used.

To check if this behaviour is retained in spaces of higher dimensionality, the previous experiments have been repeated in an 8-dimensional space. The values of N are in this case in *[8000,128000]*. As can be seen in figure 2, the average and worst rank numbers increase linearly with a low slope as the number of prototypes grows (note the high resolution of the ordinate axis and the *logarithmic* scale of the abscises).

A key factor in the cost of many fast nearest neighbour search algorithms is the dimensionality of the space where the elements are defined, n. The average cost of most of them grows steeply with the dimension n. It has also been noted that the accuracy of the approximate search algorithms drops very quickly as n gets larger.

To evaluate the effect of the dimensionality in our method, a number of experiments were performed. The number of prototypes was fixed at $N=25000$. The test set was composed of $N'=1000$ points.

In figure 3 the results are shown, along with linear least squares fits (notice that, since the controlled variable is represented in a logarithmic scale, a linear fit appears as an exponential function). The linear functions obtained are defined by the expressions $Neighb_{avg}(n) = 0.94 + 0.042\ n$, for the average neighbour, and $Neighb_{worst}(n) = 4.16 + 0.53\ n$, for the worst neighbour. These equations show small slopes, which means that the dimension of the space has a only a moderate influence on the quality of the results.

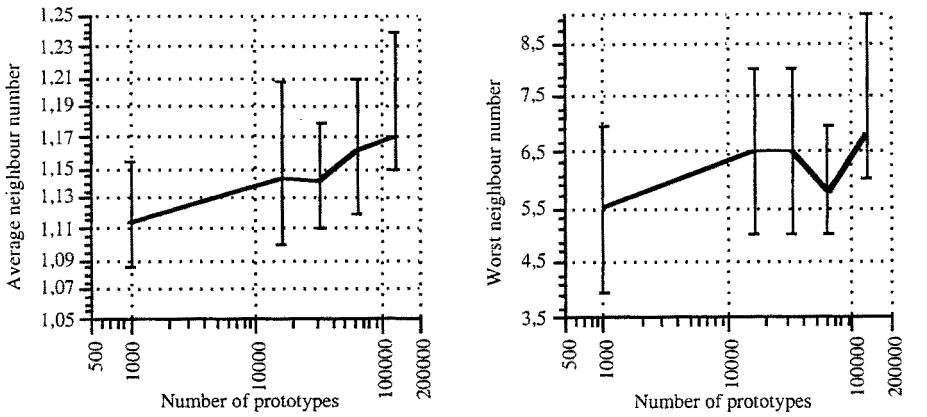

Figure 2: Average and worst rank number of the approximated nearest neighbour for a uniform random distribution in $n=8$ dimensions, with $r=8$ and $b=16$. The average values are shown, along with the maximum and minimum among four experiments performed using different random parameters for the transformations.

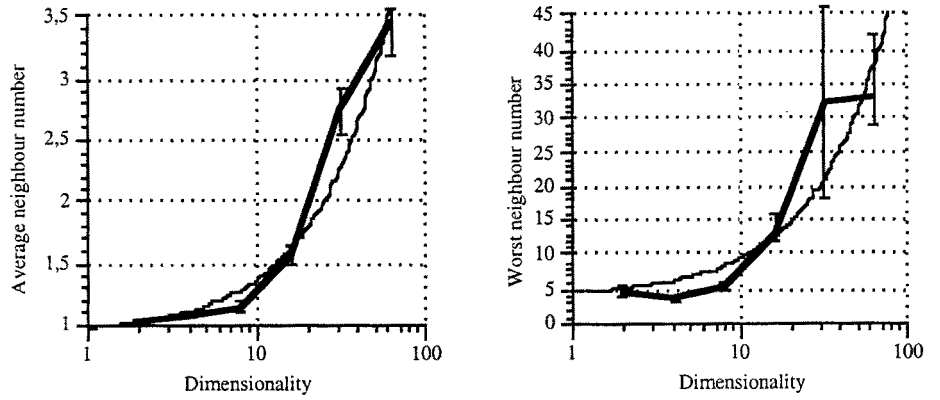

Figure 3: Average and worst rank number of the approximated nearest neighbour in uniform random distributions with $r=n$ and $b=16$. The average values are shown, along with the maximum and minimum among four experiments performed using different random parameters for the rotations and transformations. The thin line corresponds to a *linear* least squares fit of the data.

An alternative to a fixed b (value of the number of neighbours retained along the real line), is making it depend linearly on the dimension n. In this case, the asymptotic cost of a search operation increases from $O(n^2 + n\ logN)$ to $O(n^3 + n\ logN)$ or from $O(n^2)$ to $O(n^3)$. The results taking $b=n$, and keeping the same conditions as in the previous experiments are shown in figure 4.

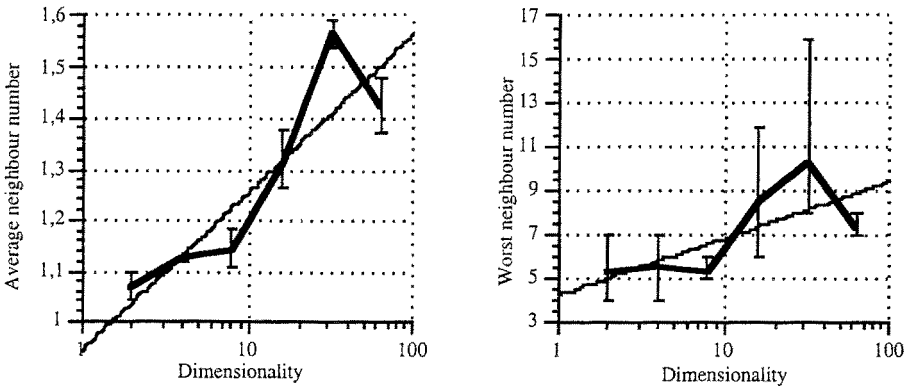

Figure 4: Results in the same conditions of figure 3, but taking $b=n$. The thin line corresponds to a *logarithmic* least squares fit of the data.

In this case, the growth of the average and worse rank numbers is no longer linear, but logarithmic: $Neigh_{avg}(n) = 0.94 + 0.135\ ln(n)$ and $Neigh_{worst}(n) = 4.25 + 1.13\ ln(n)$.

To illustrate the performance of the method in a real task, the times of a nearest neighbour search using the approximate algorithm are compared to the exahustive search in figure 5. Four submodels were used in an 8-dimensional space, with $b=16$.

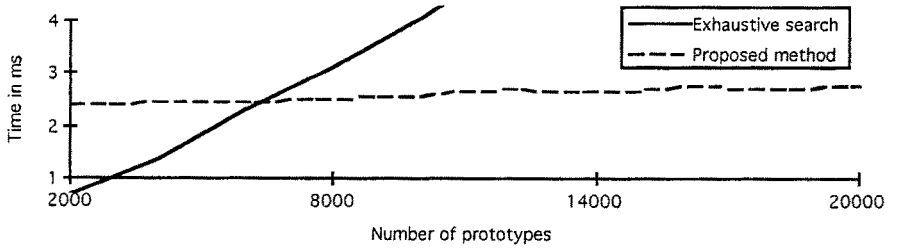

Figure 5: Times of exhaustive and approximate searches in a 300 Mhz Pentium II computer.

A more complete set of experiments can be found in [Perez,94] and [Perez,97], where some tests comparing the classification errors incurred using the exhaustive k-nearest neighbours rule and the approximate search algorithm do not show significative differences for a surprisingly wide range of parameter settings. This is in agreement with the recent results of [Skubalska,96] where a single spacefilling mapping is used in several k-nearest neighbour classification tasks, with error rates similar to the exhaustive search method.

5. Conclusions

The search of the nearest neighbours to a point in a set of prototypes in a vector space is a key problem in many disciplines, due to its theoretical appeal and its practical applications. The results presented here are very promising, and tests on the performance of the proposed method on real applications are being carried out, showing a real practical value. These applications include optical character recognition, and identification using fingerprints in a large database. In every case, a huge reduction of the actual search time has been achieved at the expense of a small loss of precision.

The method proposed is easy to adapt to the demands of most tasks, whether disk-based or memory resident, with a fixed data-set or with a need of frequent insertions and deletions, etc, and a specially good behaviour is being observed in classification applications where very large sets of prototypes are involved. In those cases, the time savings are especially important and the size of the manageable data-sets are enlarged many orders of magnitude.

6. References

Bartholdi, J.J. ; Platzman, L.K. (1983) "A Fast Heuristic Based on Spacefilling Curves for Minimum-Weight Matching in the Plane", *Information Processing Letters,* 17. pp. 177-180.

Bartholdi, J.J. ; Platzman, L.K. (1988) "Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space", *Management Science,* 34. pp. 291-305.

Bentley, J.L. ; Weide, B.W. ; Yao, A.C. (1980). "Optimal Expected Time Algorithms for Closest Point Problems", *ACM Transactions on Mathematical Software,* Vol. 6, pp. 563-580.

Bern, M. (1993). "Approximate Closest-Point Queries in High Dimensions", *Pattern Recognition,* Vol. 45, pp. 95-99.

Fukunaga, K. ; Narendra, P.M. (1975). "A Branch and Bound Algorithm for Computing *k*-Nearest Neighbors", *IEEE Transactions on Computers,* Vol. 24, No. 7, pp. 750-753.

Friedman, J.H. ; Baskett, F. ; Shustek, L.J. (1975). "An Algorithm for Finding Nearest Neighbors", *IEEE Tr. on Computers,* Vol. 24, No. 10, pp. 1000-1006.

Friedman, J.H. ; Bentley, J.L. ; Finkel, R.A. (1977). "An Algorithm for Finding Best Matches in Logarithmic Expected Time", *ACM Transactions on Mathematical Software,* Vol. 3, No. 3, pp. 209-226.

Fukunaga, K. (1990). "Introduction to Statistical Pattern Recognition", *Academic Press,* San Diego, CA.

Hilbert, D. (1891). "Ueber die steitge Abbildung einer Linie auf ein Flaechenstueck", *Math. Ann,* Vol. 38, pp. 459-460.

Imai, H. (1986). "Worst-Case Analysis for Planar Matching and Tour Heuristics with Bucketing Techniques and Spacefilling Curves", *Journal of the Operations Research Society of Japan,* Vol. 29, No. 1, pp. 43-67.

Jain, A.K. ; Dubes, R.C. (1988). "Algorithms for Clustering Data", *Prentice Hall.*

Kalantari, I. ; McDonald, G. (1983). "A Data Structure and an Algorithm for the Nearest Point Problem", *IEEE Trans. on Software Engineering,* Vol. 9, No. 5, pp. 631-634.

Kim, B.S. ; Park, S.B. (1986). "A Fast *k*-Nearest Neighbor Finding Algorithm Based on the Ordered Partition", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 8, No. 6, pp. 761-766.

Miclet, L. ; Dabouz, M. (1983). "Approximative Fast Nearest Neighbor Recognition", *Pattern Recognition Letters,* Vol. 1, No. 5/6, pp. 277-285.

Murphy, O.J. ; Selkow, S.M. (1990). "Finding Nearest Neighbors with Voronoi Tessellations", *Information Processing Letters,* Vol. 34, pp. 37-41.

Omohundro, S.M. (1990). "Geometric Learning Algorithms", *Physica D, 42.* pp.307-321.

Peano, G. (1890). "Sur une Courbe qui Remplit Toute une Aire Plane", *Math. Ann.,* Vol. 36, pp. 157-160.

Pérez, J.C. ; Vidal, E. (1994). "Métodos Geométricos de Aprendizaje Supervisado",.*Ph.D. Thesis* (In spanish) DSIC. Univ. Politécnica de Valencia.

Pérez, J.C. ; Vidal, E. (1997). "The Extended General Spacefilling Curves Heuristic" Technical Report, Dept. DISCA. Universidad Politécnica de Valencia. ***http://www.disca.upv.es***

Pérez, J.C. ; Vidal, E. (1998). "The Extended General Spacefilling Curves Heuristic" Submitted to ICPR-98.

Poggio, T. ; Girosi, F. (1990). "Networks for Approximation and Learning", *Proceedings of the IEEE,* Vol.78, no.9. pp.1481-1497.

Sethi, I.K. (1981). "A Fast Algorithm for Recognizing Nearest Neighbors", *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 11, No. 3, pp. 245-248.

Shasha, D. ; Wang, T. (1990). "New Techniques for Best-Match Retrieval", *ACM Transactions on Information Systems,* Vol. 8, No. 2, pp. 140-158.

Sierpinski, M.W. (1912). "Sur une Nouvelle Courbe Continue qui Remplit Toute une Aire Plane", *Bull. Acad. Sci. de Cracovie,* pp. 462-478.

Skubalska, E. ; Krzyzak, A. (1996). "Fast k-NN Classification Rule Using Metric on Space-Filling Curves", *Proceedings of ICPR-96,* pp.121-125.

Vidal, E. (1986). "An Algorithm for Finding Nearest Neighbours in (Approximately) Constant Average Time", *Pattern Recognition Letters,* Vol. 4, pp. 333-344.

Yao, A.C. ; Yao, F.F. (1985). "A General Approach to *d*-dimensional Geometric Queries", *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing,* 163-168.

Yunck, T.P. (1976). "A Technique to Identify Nearest Neighbors", *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 6, No. 10, pp. 678-683.