Pattern Recognition Letters 13 (1992) 471-480
North-Holland

July 1992

An efficient approximation-elimination algorithm for fast nearest-neighbour search based on a spherical distance coordinate formulation

V. Ramasubramanian and K.K. Paliwal

Computer Systems and Communications Group, Tata Institute of Fundamental Research, Homi Bhabha Road, Bombay 400005, India

Received 5 November 1991

Abstract

Ramasubramanian, V. and K.K. Paliwal, An efficient approximation-elimination algorithm for fast nearest-neighbour search based on a spherical distance coordinate formulation, Pattern Recognition Letters 13 (1992) 471-480.

An efficient approximation-elimination search algorithm for fast nearest-neighbour search is proposed based on a spherical distance coordinate formulation, where a vector in K-dimensional space is represented uniquely by its distances from $K+1$ fixed points. The proposed algorithm uses triangle-inequality based elimination rules which is applicable for search using metric distances measures. It is a more efficient fixed point equivalent of the Approximation Elimination Search Algorithm (AESA) proposed earlier by Vidal [2]. In comparison to AESA which has a very high $O(N^2)$ storage complexity, the proposed algorithm uses only $O(N)$ storage with very low approximation-elimination computational overheads while achieving complexity reductions closely comparable to AESA. The algorithm is used for fast vector quantization of speech waveforms and is observed to have $O(K+1)$ average complexity.

Keywords. Fast nearest-neighbour search, triangle-inequality based elimination, distance coordinate representation, fast vector quantization, approximation-elimination search.

Introduction

Nearest-neighbour search consists in finding the closest point to a query point among N points in K-dimensional space. This search is widely used in several areas such as pattern classification, non-parametric estimation and data compression using vector quantization. Reducing the complexity of nearest-neighbour search is of considerable interest in these areas, and particularly in vector quantization encoding where the complexity increases exponentially with dimension K [1]. In this paper, we discuss fast nearest-neighbour search in the context of vector quantization. Vector quantization encoding is the minimum-distortion quantization of a vector

$$x=(x_1,\ldots,x_K)$$

(referred to as the *test vector*) using a given set of N K-dimensional *codevectors* called the *codebook*

Correspondence to: V. Ramasubramanian, Computer Systems and Communications Group, Tata Institute of Fundamental Research, Homi Bhabha Road, Bombay 400005, India.

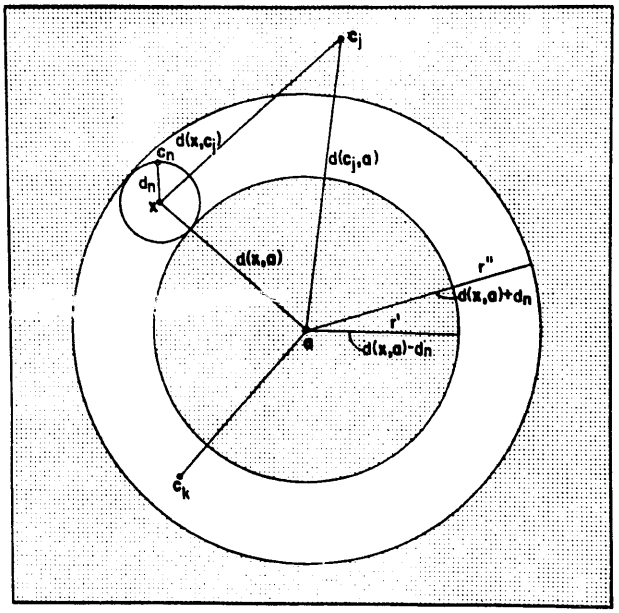

Figure 1. Triangle-inequality based elimination.

$$C=\{c_j\}_{j=1,\ldots,N},$$

of size N, under some distance measure $d(x, y)$. This involves finding the nearest-neighbour of x in C, given by

$$q(x)=c_k: \quad d(x,c_k)\leqslant d(x,c_j), \quad j=1,\ldots,N,$$

which requires N vector distance computations $d(x,c_j)$ using the exhaustive full-search.

One of the common approaches to fast nearest-neighbour search is the use of inexpensive elimination rules to eliminate codevectors which cannot be nearer to the testvector x than the current-nearest neighbour c_n. Among these, the triangle-inequality based elimination rule, which is applicable when the distance measure is a metric, is a popular one [2]. Applying the triangle inequality between x, a codevector c_j and some fixed point a yields (see Figure 1)

$$d(c_j,a)\leqslant d(x,a)+d(x,c_j)$$

and

$$d(c_j,a)\geqslant d(x,a)-d(x,c_j).$$

If c_n is the current-nearest neighbour of x with the current-nearest neighbour distance $d_n=d(x,c_n)$, then, for any codevector c_j to be a successor of c_n (i.e., to be inside the current-nearest neighbour ball of radius $d(x,c_n)$ and center at x) it is required that $d(x,c_j)<d(x,c_n)$. This c_j should then satisfy

$$d(c_j,a)\leqslant d(x,a)+d(x,c_n)$$

and

$$d(c_j,a)\geqslant d(x,a)-d(x,c_n).$$

Conversely, any c_j not satisfying either of these two conditions cannot satisfy $d(x,c_j)<d(x,c_n)$ and hence can be rejected without computing $d(x,c_j)$. This results in the triangle-inequality based elimination rule: Reject c_j if

$$d(c_j,a)>d(x,a)+d(x,c_n)=r''$$

or

$$d(c_j,a)<d(x,a)-d(x,c_n)=r'.$$

This elimination corresponds to a hyperannulus constraint of the search space, where code-vectors lying outside the hyperannulus region formed between the two concentric hyperspheres of radius r' and r'', centered at a (henceforth referred to as 'anchor point') are eliminated. This is illustrated in Figure 1. The hyperannulus has a width

$$r''-r'=2d_n,$$

and encloses the current nearest-neighbor ball of radius d_n. This elimination requires the N codevector-anchor point distances

$$\{d(c_j,a)\}_{j=1,\ldots,N},$$

and the test vector-anchor point distance $d(x,a)$. The efficiency of this triangle-inequality (or hyperannulus) elimination increases with the use of more distinct anchor points, $\{a_m\}_{m=1,\ldots,M}$, the codevectors retained after elimination being those present inside the intersection volume of the multiple hyperannulus corresponding to $\{a_m\}_{m=1,\ldots,M}$. However, use of M anchor points requires the precomputation and storage of the MN codevector-anchor point distances

$$\{\{d(c_j,a_m)\}_{m=1,\ldots,M}\}_{j=1,\ldots,N}$$

and the computation of the M test vector-anchor point distances $\{d(x,a_m)\}_{m=1,\ldots,M}$.

The AESA (Approximation Elimination Search Algorithm) proposed earlier by Vidal [2] employs this multiple anchor point hyperannulus elimination by using the codevectors themselves as the

anchor points. Since the test vector-anchor point distances are then the test vector to code vector distances, only those codevectors whose distance to the test vector is known can be used as an anchor point. The codevectors are introduced as anchor points in an approximation-elimination framework, where the search consists of successive approximation to the actual nearest neighbour using repeated application of two main steps: (i) Approximation: selecting a codevector using an approximation criterion and (ii) Elimination: eliminating codevectors using the triangle inequality rule. The role of approximation step is to select codevectors as close to the test vector as possible using some computationally inexpensive approximation criterion and serves as an efficient alternative to choosing codevectors at random or in a prespecified sequence. In this scheme, the codevectors which form the anchor point set and the number of such codevectors are dependent on the test vector, and consequently, any subset of the full codebook can play the role of the anchor point set. Therefore all the codevector-codevector distances

$$\{d(\boldsymbol{c}_i, \boldsymbol{c}_j)\}, \quad i,j = 1,\ldots,N$$

have to be precomputed and stored in a triangular array of size $N(N-1)/2$. This exhorbitant $O(N^2)$ storage complexity of AESA severely limits its practical use for large codebook sizes [2] and in DTW-based fast isolated word recognition for vocabulary sizes up to about only 400 [3].

In this paper, we propose an efficient equivalent algorithm of AESA which uses only $O(N)$ storage with $O(K+1)$ average search complexity and worst case complexity closely comparable to AESA. The proposed algorithm is motivated from the discussion in [2] which analyses the efficiency of AESA and poses general questions directed towards obtaining possible improvements over AESA. The algorithm proposed here uses only $K+1$ fixed anchor points and is based on the representation of a vector in K-dimensional space by its $K+1$ distances to some $K+1$ fixed points. This representation and the approximation-elimination procedure based on this is given in the following sections.

Spherical distance coordinate representation

Let $\{\boldsymbol{a}_m\}_{m=0,\ldots,M-1}$ be M points in $\mathbb{R}^K$. For any $\boldsymbol{x}$ in $\mathbb{R}^K$, let $d_m^x = d(\boldsymbol{x}, \boldsymbol{a}_m)$, where d_m^x is the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{a}_m$, given by

$$(d_m^x)^2 = \sum_{j=1}^{K} (x_j - a_{mj})^2.$$

Denoting d_m^x as d_m, the M distance specification corresponds to M equations e_m:

$$\sum_{j=1}^{K} (x_j - a_{mj})^2 = (d_m)^2 \quad \Rightarrow$$

$$e_m: \ \|\boldsymbol{x}\|^2 + \|\boldsymbol{a}_m\|^2 - 2\sum_{j=1}^{K} x_j a_{mj} = (d_m)^2,$$

$$m = 0,\ldots,M-1.$$

Subtracting expression e_m from e_0 gives

$$\sum_{j=1}^{K} x_j a'_{mj} = \tfrac{1}{2}[(d_0^2 - d_m^2) - (\|\boldsymbol{a}_0\|^2 - \|\boldsymbol{a}_m\|^2)] = D_m^x \tag{1}$$

where $a'_{mj} = a_{mj} - a_{0j}$. Given only

$$\{\boldsymbol{a}_m\}_{m=0,\ldots,M-1} \quad \text{and} \quad \{d_m\}_{m=0,\ldots,M-1},$$

this can be viewed as a set of $M-1$ linear equations in the K unknowns $(x_j)_{j=1,\ldots,K}$ and can be expressed as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{D}^x$, where $\boldsymbol{A}$ is a $(M-1)\times K$ matrix with the 'difference anchor vector' $(\boldsymbol{a}_m - \boldsymbol{a}_0)^T$ as row m, for $m = 1,\ldots,M-1$ and $\boldsymbol{D}^x = (D_1^x,\ldots,D_{M-1}^x)^T$. This set of $M-1$ equations in K unknowns will have a unique solution for $(x_j)_{j=1,\ldots,K}$ (i.e., $\boldsymbol{x}$) only for $M-1=K$ (i.e., $M=K+1$) and the rank of $\boldsymbol{A}$ is K—which requires linear independence of its row vectors

$$\{(\boldsymbol{a}_m - \boldsymbol{a}_0)^T\}_{m=1,\ldots,M-1}.$$

Thus, a vector $\boldsymbol{x}$ in $\mathbb{R}^K$ can be uniquely represented by its $K+1$ distances

$$\boldsymbol{d}^x = \{d_m^x = d(\boldsymbol{x}, \boldsymbol{a}_m)\}_{m=0,\ldots,K}$$

(henceforth referred to as the (*spherical*) *distance coordinates* of $\boldsymbol{x}$) from $K+1$ fixed points

$$\{\boldsymbol{a}_m\}_{m=0,\ldots,K}$$

(called the anchor points, henceforth) which are such that the K difference vectors

$$\{(\boldsymbol{a}_m - \boldsymbol{a}_0)^T\}_{m=1,\ldots,K}$$

are linearly independent. When either $M<K+1$, or $M=K+1$ but the anchor points do not satisfy the linear independent condition, the solution will not be unique and the M distances of x from the M anchor points will not locate x uniquely. We refer to these anchor points as 'sub-optimal' (with respect to unique point representation). The locus solution of x for sub-optimal anchor points then corresponds to (1) being essentially underdetermined.

Based on this, the N codevectors $\{c_j\}_{j=1,\ldots,N}$ are represented by their $K+1$ distance coordinates

$$d^{c_j}=\{d_m^{c_j}=d(c_j,a_m)\}_{m=0,\ldots,K}$$

using some specific optimal anchor point configuration. Given a test vector x, its $K+1$ spherical distance coordinates d^x are also computed from the given optimal $K+1$ anchor points. Then, the approximation criterion, as used by Vidal [2] for selecting candidate codevectors c_a close to the test vector x, given for the general case of M anchor points here will be:

$$c_a=\arg\min_{c_j\in C}\alpha_M(d^x,d^{c_j}),$$

where $\alpha_M(d^x,d^{c_j})=\sum_{m=0}^{M-1}|d_m^x-d_m^{c_j}|$ is essentially an M-dimensional L_1 distance between d^x and d^{c_j}. For optimal anchor point configuration, when $M=K+1$, these coordinates locate c_j and x uniquely and $\alpha_{K+1}(d^x,d^{c_j})$ will be a good approximation of the actual distance between x and c_j with a high degree of correlation. With respect to elimination, when $M=K+1$, the $K+1$ hyperannulus formed for a given current nearest-neighbour ball will have a convex intersection volume locally inscribing the current nearest-neighbour ball. However, when $M\leqslant K+1$, the sub-optimal number of distance coordinates of x and c_j do not represent them uniquely. Consequently, $\alpha_M(d^x,d^{c_j})$ will be a poor approximation of the actual distance and the approximation criterion can select codevectors which are actually very far from x, though having smaller α_M values than the codevectors which are closer to x. With respect to elimination, since x is not represented uniquely, the current nearest-neighbour ball volume specified in terms of the M perturbations in d^x and the corresponding M hyperannulus intersection volume will not be localized around x. During elimination, this will result in retention of codevectors which are actually far from x, but contained within the M hyperannulus intersection volume which now encloses the sub-optimal locus solution of x.

Anchor point configurations

For the representation of the codevectors and test vector by their distance coordinates in the proposed approximation-elimination algorithms, we use the following two specific anchor point configurations which satisfy the optimal conditions for unique point location:

A1: a_0 at the origin and $(a_1,\ldots,a_K)$ along the coordinate axes at equal distances ϱ from the origin.

A2: a_0 at the origin and $(a_1,\ldots,a_K)$ at equal distances ϱ from the origin along the K principal component directions of the test vector data obtained as the directions of the eigenvectors of the covariance matrix of the test vector data. Here, anchor point $a_m=\varrho e_m$, is located along the direction of the eigenvector e_m whose corresponding eigenvalue λ_m is the mth maximum among the K eigenvalues $(\lambda_1,\ldots,\lambda_K)$.

The principal component directions represent the orthogonal directions of maximum variance in the data, proportional to the corresponding eigenvalues. Moreover, among $\{e_i\}_{i=r,\ldots,K}$, $1\leqslant r\leqslant K$, e_r is the direction of maximum variance and is also the solution for the best perpendicular least-square error (or eigenvector) fit, with minimum cross-sectional volume orthogonal to this direction. Therefore, an hyperannulus with its center placed well away from the origin along e_1 will yield the minimum intersection (cross-section volume) with the data and the codevector distribution, implying high approximation-elimination efficiency, and this efficiency decreases gradually from e_1 to e_K. Thus, the A2 configuration will be more efficient than A1 in approximation and elimination, particularly in the case of speech waveform vector quantization where the vectors are highly correlated across their components. The A2 spherical distance coordinate representation implicitly achieves a principal component rotation of the test

vector and codebook and is equivalent to their explicit principal component rotation with the use of anchor point configuration A1. In this context, it should be noted that the principal component rotation has no effect on the performance of AESA since its anchor points are the codevectors themselves and rotation does not provide any relative advantage in reducing the extent of intersection between the hyperannulus and the codevector distribution.

Fixed anchor point — Approximation elimination search algorithm (FAP-AESA)

The FAP-AESA described here uses a constant number of M fixed anchor points all at the same time. This uses the approximation criterion given above for M anchor points to select the codevector at each approximation step.

Given $\boldsymbol{C}=\{\boldsymbol{c}_j\}_{j=1,\ldots,N}$, and $A_M=\{\boldsymbol{a}_0,\ldots,\boldsymbol{a}_{M-1}\}$, $\{\{d^{c_j}\}_{m=0,\ldots,M-1}\}_{j=1,\ldots,N}$ are precomputed and stored with NM storage. Given a test vector $\boldsymbol{x}$, its M distance coordinates $\boldsymbol{d}^x=\{d_m^x\}_{m=0,\ldots,M-1}$ and the full approximation values

$$\alpha_M(\boldsymbol{d}^x,\boldsymbol{d}^{c_j})=\sum_{m=0}^{M-1}|d_m^x-d_m^{c_j}|,$$

for $j=1,\ldots,N$, are computed. Approximation-elimination search in FAP-AESA then involves repeating Steps 1 to 3 until the codevector set $\boldsymbol{C}$ is empty:

Step 1. Approximation:

$$\boldsymbol{c}_a=\arg\min_{\boldsymbol{c}_j\in\boldsymbol{C}}\alpha_M(\boldsymbol{d}^x,\boldsymbol{d}^{c_j});$$

$$\boldsymbol{C}=\boldsymbol{C}-\{\boldsymbol{c}_a\}$$

Step 2. Distance computation and nearest-neighbour update

$$d_a=d(\boldsymbol{x},\boldsymbol{c}_a);$$

If $d_a\geqslant d_n$ then continue at Step 1
else $\boldsymbol{c}_n=\boldsymbol{c}_a$ and $d_n=d_a$

Step 3. Elimination:

$$\boldsymbol{C}=\boldsymbol{C}-\{\boldsymbol{c}_j:\ d_m^{c_j}<d_m^x-d_n \text{ or } d_m^{c_j}>d_m^x+d_n \text{ for } m=0,\ldots,M-1\}$$

The main computational overhead in FAP-AESA is the full approximation carried out in the first step for all the N codevectors incurring N M-dimensional L_1 distance computation costs and the full M anchor point elimination step on all the N codevectors. In order to reduce this overhead cost incurred due to the use of all M anchor points, we describe in the following section, an algorithm which uses the anchor points incrementally as done in AESA [2].

Incremental FAP-AESA (IFAP-AESA)

The incremental realization of FAP-AESA, termed IFAP-AESA, is similar to AESA with the main difference that only a prespecified number of anchor points M is used. The anchor points are introduced at each approximation step up to a maximum of M anchor points (or until the codevector set is exhausted, whichever happens earlier), after which the number of anchor points used is held constant at M, and the approximation-elimination search proceeds until the codevector set is exhausted.

The IFAP-AESA differs from FAP-AESA mainly in the approximation step with the introduction of an 'increment' control step which increments the anchor point set up to the specified maximum number of anchor points. Given $\boldsymbol{x}$, the anchor point set and approximation values are initialized as $A_{-1}=\{\}$ and $\alpha_0(\boldsymbol{d}^x,\boldsymbol{d}^{c_j})=0$ for all $\boldsymbol{c}_j\in\boldsymbol{C}$; step count $p=0$. The modified approximation step with the increment control is:

Increment: $p=p+1$; if $p\not>M$, then
$q=p-1$;
$d_q^x=d(\boldsymbol{x},\boldsymbol{a}_q)$ and $A_q=A_{q-1}\cup\{\boldsymbol{a}_q\}$;
$\alpha_{q+1}(\boldsymbol{d}^x,\boldsymbol{d}^{c_j})=\sum_{\boldsymbol{a}_m\in A_q}|d_m^x-d_m^{c_j}|$ is computed incrementally as

$$\alpha_{q+1}(\boldsymbol{d}^x,\boldsymbol{d}^{c_j})=\alpha_q(\boldsymbol{d}^x,\boldsymbol{d}^{c_j})+|d_q^x-d_q^{c_j}|;$$

Approximation:

$$\boldsymbol{c}_a=\arg\min_{\boldsymbol{c}_j\in\boldsymbol{C}}\alpha_{q+1}(\boldsymbol{d}^x,\boldsymbol{d}^{c_j});$$

The distance computation and nearest-neighbour update are as in FAP-AESA, and for every update the elimination is done using the current anchor

point set A_q with $q+1$ anchor points. The incremental accumulation of α_q is done up to $q=M-1$, i.e., M anchor points have been introduced. Prior to the incrementing of the α_q value, elimination with the new anchor point a_q is done by rejecting c_j if $|d_q^x - d_q^{c_j}|$ is greater than d_n. For $p>M$, the approximation value of the remaining codevectors is saturated at α_M — the full M approximation value — as α_q is not incremented further and the approximation involves only finding the codevector with the minimum full-approximation value α_M.

Here, the incremental accumulation of the approximation value at each step p requires only one coordinate difference computation for the codevectors in the current codevector set. As this codevector set size decreases progressively at each elimination step and for every addition of a new anchor point, the single component approximation cost decreases rapidly as the search progresses and the effective approximation overhead cost is much less than for FAP-AESA. Moreover, since the anchor point a_0 introduced in the first step is known a priori, the first approximation candidate c_a is found by a binary search on the ordered a_0 codevector coordinates after computing $d(x, a_0)$. Elimination is also done by a binary search based truncation on this ordered list using the lower and upper hyperannulus bounds corresponding to the current nearest-neighbour radius $d(x, c_a)$. This cuts down the O(N) initial approximation-elimination cost to O(log N) resulting in significant computational overhead cost savings, while requiring only an additional N scalar storage for the ordered a_0 coordinates of the N codevectors.

Experiments, results and discussion

Here we present results characterizing the performance of FAP-AESA and IFAP-AESA and compare it with AESA [2] in the context of vector quantization of speech waveform data. The algorithms FAP-AESA and IFAP-AESA are studied for number of anchor points $M \leqslant K+1$ as well as $M>K+1$. For $M>K+1$, the $K+1$ optimal configurations A1 and A2 are extended with the anchor points $\{a_m\}_{M>K+1}$ located in separate quadrants at equal distances from the origin.

First we characterize the approximation efficiency as a function of the number of anchor points used for the optimal anchor point configurations A1 and A2. The approximation efficiency is measured by using the *initial* candidate codevector selected by the approximation criterion from the full codebook. Here, we use $(\overline{ns_i}, \widehat{ns_i})$ — the average and maximum number of codevectors within the current nearest-neighbour ball of the initial candidate codevector. This serves as a measure of the volume of the current nearest-neighbour ball of the first approximation candidate and will characterize the efficiency of approximation in terms of how close the codevector it selects is to the test vector. $(\overline{ns_i}, \widehat{ns_i})$ are plotted in Figure 2 for M varying from 1 to 15 for the anchor point configurations A1 and A2. It can be clearly seen that the approximation efficiency increases drastically as M reaches $K+1$, indicating efficient localization by the approximation. For $M>K+1$, the efficiency saturates or gradually worsens.

Figure 3 shows the performance efficiency of FAP-AESA for dimension $K=10$ and codebook size $N=1024$ using 5000 test vectors. This is shown for M, the number of anchor points used, varying from 1 to 15. The basic elimination efficiency of M

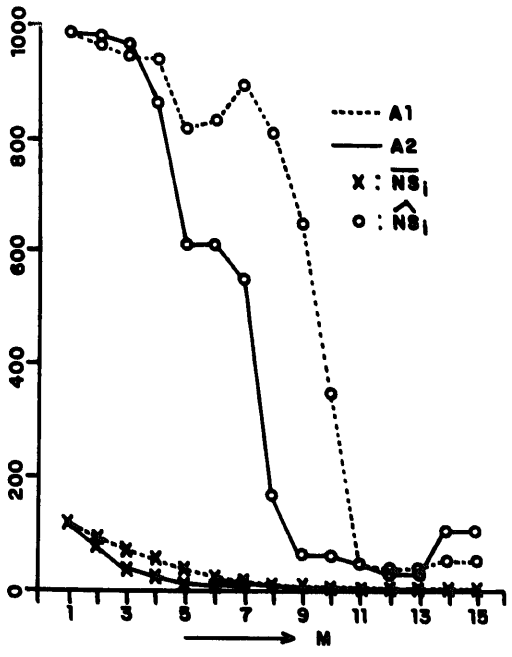

Figure 2. Approximation efficiency of FAP-AESA.

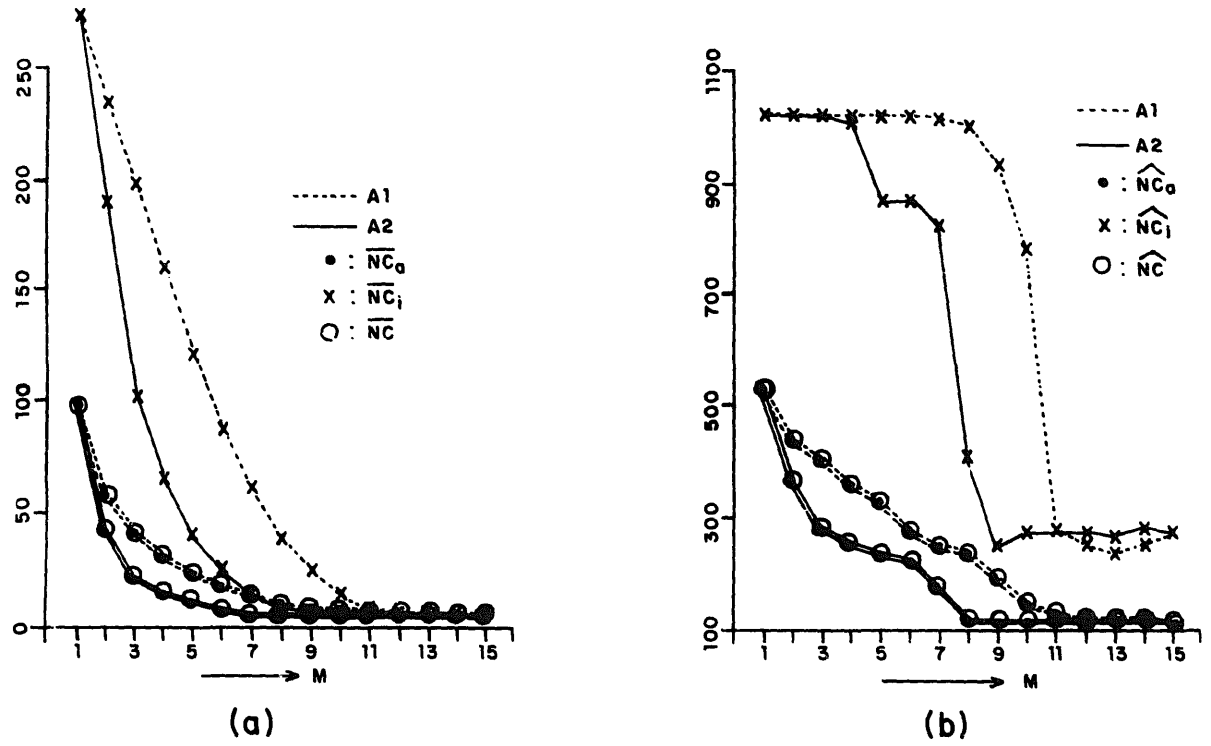

Figure 3. Performance efficiency of FAP-AESA.

anchor points is measured by $(\overline{nc_a}, \widehat{nc_a})$ — the (average, maximum) number of codevectors retained after elimination using the *actual* nearest-neighbour ball radius $d(x, q(x))$ to define the hyperannulus. These are the average and maximum number of codevectors inside the smallest M hyperannulus intersection volume achievable using the given M anchor points for a given test vector. These can be seen to decrease with increase in M indicating better localization of the intersection volume around the current nearest-neighbour ball with addition of anchor points. The elimination efficiency saturates for $M > K+1$, showing that use of optimal $K+1$ anchor points is quite sufficient to produce a saturating localization of the intersecting volume.

The approximation efficiency in using M anchor points is shown using $(\overline{nc_i}, \widehat{nc_i})$ — the (average, maximum) number of codevectors retained after elimination with the current nearest-neighbour as the *initial* codevector c_a selected by the approximation criterion in the first step. Given that use of the actual nearest-neighbour $q(x)$ achieves the complexity $(\overline{nc_a}, \widehat{nc_a})$, the difference between $(\overline{nc_i}, \widehat{nc_i})$ and $(\overline{nc_a}, \widehat{nc_a})$ is an indirect measure of the approximation efficiency as it indicates how close the codevector c_a selected by the approximation criterion is to $q(x)$. This is quite high for $M < K+1$ and indicates poor approximation efficiency. However, as M increases towards $K+1$, $(\overline{nc_i}, \widehat{nc_i})$ shows a drastic fall close to $(\overline{nc_a}, \widehat{nc_a})$, indicating the excellent approximation efficiency at $K+1$. The approximation efficiency however saturates for $M > K+1$. It can also be noted that configuration A2 performs significantly better than A1 with higher approximation and elimination efficiency, particularly for $M < K+1$, due to its principal component advantage in having smaller intersections with the speech data.

The actual complexity of number of distance computations carried out in FAP-AESA is given by $(\overline{nc}, \widehat{nc})$. This is very close to $(\overline{nc_a}, \widehat{nc_a})$ complexity for all M, indicating the high efficiency of the combined approximation-elimination steps in reducing the first step complexity of $(\overline{nc_i}, \widehat{nc_i})$ very close to $(\overline{nc_a}, \widehat{nc_a})$ — the best complexity achievable by FAP-AESA if it starts with the actual nearest-neighbour itself.

In Figure 4, we show the performance of IFAP-AESA for M (the maximum number of an-

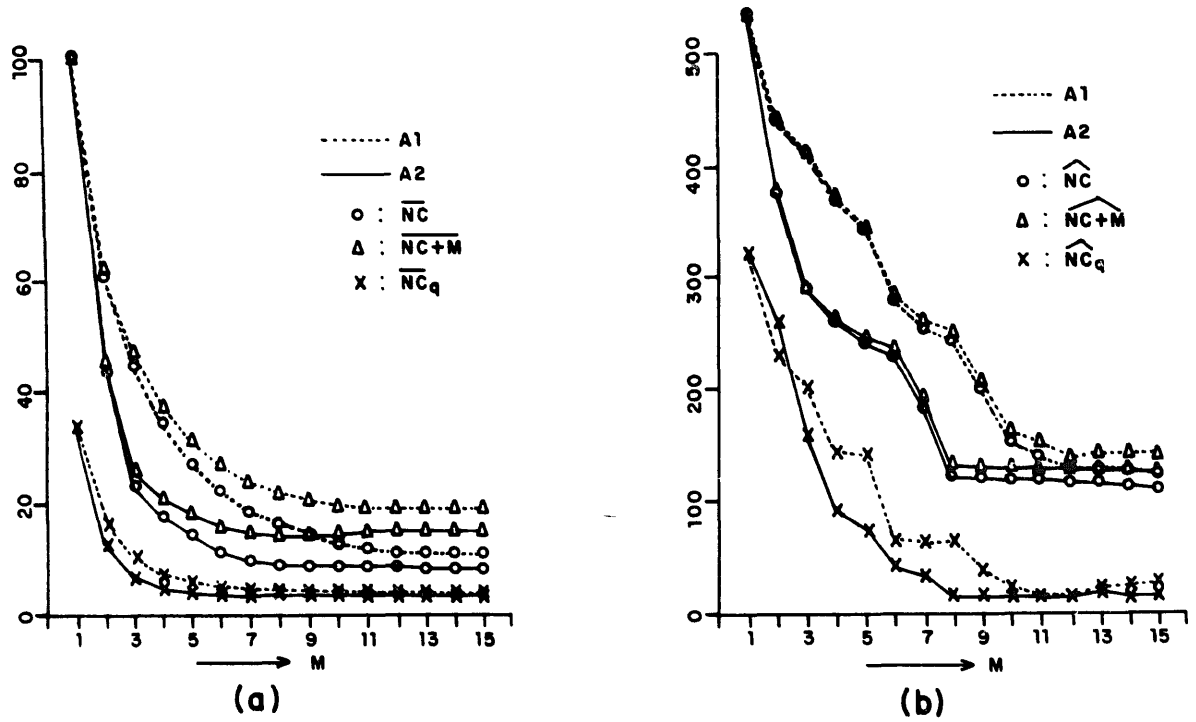

Figure 4. Performance efficiency of IFAP-AESA.

chor points used) varying from 1 to 15 for $K=10$ and $N=1024$. In Figure 4, $(\overline{nc}, \widehat{nc})$ gives the average number of distances computed and is the basic complexity of IFAP-AESA. The net average complexity is $\overline{nc}+\bar{M}$ — including $\bar{M}$, the average number of test vector-anchor point distances computed. The net worst case complexity is $\widehat{nc}+M$ — including the maximum number of test vector-anchor point distances computed. Also shown is $(\overline{nc_q}, \widehat{nc_q})$ — the (average, maximum) number of approximation steps required to locate the actual nearest-neighbour. This characterizes the effective approximation efficiency of the incremental scheme with smaller $(\overline{nc_q}, \widehat{nc_q})$ indicating more effective approximation. $(\overline{nc}, \widehat{nc})$ and $(\overline{nc_q}, \widehat{nc_q})$ decrease as M increases towards $K+1$ and saturates for $M>K+1$, indicating the saturation of both the elimination and approximation efficiency. It can also be noted that anchor point configuration A2 performs significantly better than A1 for all M.

The average number of anchor points $\bar{M}$ actually used in the search is shown in Figure 5 for various M. $\bar{M}$ is much less than M, indicating that the search terminates usually well before all the M anchor points are used. Configuration A2 uses a less number of anchor points on average than A1 demonstrating its higher approximation-elimination efficiency.

In Table 1, we compare the performances of FAP-AESA, IFAP-AESA (for A1 and A2 configurations) with AESA [2] for dimension $K=10$ and codebook size $N=1024$ using $M=K+1=11$ anchor points in terms of computational and storage complexity and overhead costs. Here, the following can be noted: For FAP-AESA and IFAP-AESA, A2 performs better than A1 in terms of complexity reduction and overhead costs. For a given configuration, IFAP-AESA performs very closely to FAP-AESA, but with considerably reduced approximation-elimination overhead costs. FAP-AESA and IFAP-AESA have an additional storage of $N(K+1)$ and $N(K+1)+N$ respectively (i.e., only (O(N) storage complexity)

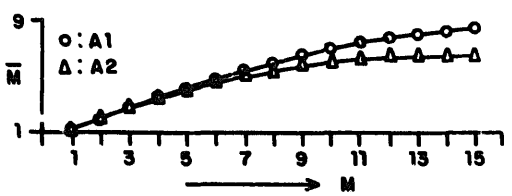

Figure 5. Average number of anchor points used in IFAP-AESA for anchor point configurations A1 and A2.

Table 1
Performance of FAP-AESA, IFAP-AESA and AESA (dimension $K = 10$, codebook size $N = 1024$)

Algorithm	AC	M		nc		nc+M		Additional Storage	Approx-imation Cost	Elimin-ation Cost	m	a	c
		ave	max	ave	max	ave	max						
FAP-AESA	A1	11	11	7.9	137	18.9	148	11264	1118	1956	18.9	1982	2066
	A2	11	11	5.9	117	16.9	128	11264	1081	1699	16.9	1978	1805
IFAP-AESA	A1	7.4	11	11.8	142	19.2	153	12288	101	339	19.2	239	441
	A2	6.2	11	8.7	120	14.9	131	12288	67	213	14.9	163	281
AESA		11.4	116	11.4	116	11.4	116	523776	172	741	11.4	366	914
FULL-SEARCH		-	-	1024	1024	-	-	0	-	-	1024	1946	1023

AC: anchor point configuration; M: number of anchor points used; nc: number of codevector distances computed (m, a, c): average number of (multiplications, additions and comparisons)

as against the $N(N-1)/2$, $O(N^2)$ storage complexity of AESA. In addition, the overhead costs are much lower for IFAP-AESA than AESA. Both FAP-AESA and IFAP-AESA (with the more efficient A2 configuration) perform very closely to AESA while using only a maximum of 11 anchor points as compared to a much larger (116) number of maximum anchor points required by AESA. The standard *macs* measure also shows FAP-AESA and IFAP-AESA to have less overall complexity than AESA. The significant complexity reduction offered by FAP-AESA and IFAP-AESA over full-search can be noted.

Table 2 shows the average and worst case complexities of FAP-AESA and IFAP-AESA using anchor point configuration A2, for codebook sizes $N = 256$, 512, 1024 with dimension $K = 10$, and $K = 8$, 9, 10 with $N = 1024$. The constant complexity performance of these algorithms can be observed across the dimensions and codebook sizes, with the additional overhead of $K+1$ distance computations between the test vector and $K+1$ anchor points resulting in their $O(K+1)$ average complexity.

Table 2
Performance of FAP-AESA and IFAP-AESA (anchor point configuration: A2)

K = 10					N = 1024				
	FAP-AESA		IFAP-AESA			FAP-AESA		IFAP-AESA	
N	$\overline{nc}$	$\widehat{nc}$	$\overline{nc}$	$\widehat{nc}$	K	$\overline{nc}$	$\widehat{nc}$	$\overline{nc}$	$\widehat{nc}$
256	4.5	76	6.5	79	8	4.0	85	6.5	86
512	5.2	88	7.6	91	9	4.8	126	7.5	129
1024	5.9	117	8.7	120	10	5.9	117	8.7	120

$(\overline{nc}, \widehat{nc})$: (average, maximum) number of codevector distances computed

We have shown here that fast nearest-neighbour search in a K-dimensional space can be done efficiently using only $K+1$ optimal anchor points, the distances from which represent any vector uniquely. This scheme is an efficient alternative to AESA as it achieves comparable complexity reductions using significantly less storage. However, the proposed scheme does not extend readily as an alternative to AESA for fast DTW-based IWR search [3], as this involves the use of M optimal anchor points such that every word in the utterance in the given vocabulary is represented uniquely by its distances to these anchor points. This however raises the issue of finding M words from the given vocabulary set of size N (with $M \ll N$) to serve as the optimal anchor points under the criterion of unique representation. This also leads to the more interesting possibility of representing words in a regular M-dimensional space with the M distances to the M optimal anchor points as the coordinates of a word. This has important implications in terms of fast search and in applying standard pattern recognition techniques of regular vector spaces to words and word-like units.

Conclusions

An efficient approximation-elimination search algorithm has been proposed for fast nearest-neighbour search with metric distance measures using triangle-inequality based elimination rules. The

algorithm is based on a spherical distance coordinate formulation, where a vector in K-dimensional space is represented uniquely by its distances from $K+1$ fixed points. The proposed algorithm uses only $O(N)$ storage with very low approximation-elimination computational overheads while achieving complexity reductions closely comparable to the recently proposed AESA which has high $O(N^2)$ storage complexity. The algorithm is studied in the context of fast vector quantization of speech waveform and is observed to have $O(K+1)$ average complexity. Moreover, the proposed algorithm allows free choice of anchor point configurations and uses a fixed point configuration along the principal component directions of the speech data to exploit the high correlation in speech waveform vectors for more efficient approximation-elimination search.

References

[1] Makhoul, J., S. Roucos and H. Gish (1985). Vector quantization in speech coding. *Proc. IEEE* 73, 1555–1588.

[2] Vidal, E. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time complexity. *Pattern Recognition Letters* 4, 145–157.

[3] Vidal, E., H.M. Rulot, F. Casacuberta and J.M. Benedi (1988). On the use of a metric-space search algorithm (AESA) for fast DTW-based recognition of isolated words. *IEEE Trans. Acoust. Speech Signal Process.* 36 (5), 651–660.