Distinctiveness-Sensitive Nearest-Neighbor Search for Efficient Similarity Retrieval of Multimedia Information

Norio Katayama and Shin'ichi Satoh

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
E-mail: {katayama,satoh}@nii.ac.jp

Abstract

Nearest neighbor (NN) search in high dimensional feature space is widely used for similarity retrieval of multimedia information. However, recent research results in the database literature reveal that a curious problem happens in high dimensional space. Since high dimensional space has high degree of freedom, points could be so scattered that every distance between them might yield no significant difference. In this case, we can say that the NN is indistinctive because many points exist at the similar distance. To make matters worse, indistinctive NNs require more search cost because search completes only after choosing the NN from plenty of strong candidates. In order to circumvent the harmful effect of indistinctive NNs, this paper presents a new NN search algorithm which determines the distinctiveness of the NN during search operation. This enables us not only to cut down search cost but also to distinguish distinctive NNs from indistinctive ones. These advantages are especially beneficial to interactive retrieval systems.

1. Introduction

Nearest-neighbor (NN) search in the feature space is widely used for the similarity retrieval of multimedia information. Each piece of multimedia information is mapped to a vector in a multi-dimensional space where the distance between two vectors (typically, Euclidean distance between the heads of vectors) corresponds to the similarity of multimedia information. These vectors and the space are called feature vectors and feature space respectively. Once the feature space is obtained, the similarity retrieval of multimedia information is reduced to NN search in the feature space. One of the important properties of the feature space is high dimensionality. It is very common to use 10 or higher dimensional space for the feature space.

Recent results in the literature reveal that a curious problem happens in high dimensional space, which is not imagined in low dimensional space. Since high dimensional space has high degree of freedom, points could be so scattered that every distance between them might yield no significant difference. A typical example appears in the uniformly distributed points, i.e., points distributed uniformly in a unit hypercube. For example, Berchtold et al.[1] reported that most of the points are located near the surface of the cube, and Katayama et al.[2] reported experimentally that the distances among the data points are very similar for any combination of them. Recently, Beyer et al.[3] showed that under a broad set of conditions (broader than uniform distribution) the distance to the nearest data point approaches the distance to the farthest data point as dimensionality increases. Thus, in high dimensional space, the distance to the nearest neighbor may not yield significant difference compared with the distances to the other points. In this case, we can say that the nearest neighbor is *indistinctive* because many points exist at the similar distance with the nearest neighbor. To make matters worse, indistinctive nearest neighbors require more search cost than distinctive ones because search completes only after choosing the nearest neighbor from plenty of strong candidates.

Indistinctive nearest neighbors are troublesome with respect to the similarity retrieval. They are less informative to users since we do not have significant difference between the nearest neighbor and the others. In addition, indistinctive nearest neighbors degrade the system response because they incur high search cost. As mentioned above, high dimensional space is more likely to have indistinctive nearest neighbors. This means that the similarity retrieval based on high dimensional feature space is more vulnerable to the harmful effect of indistinctive nearest neighbors. However, this does not mean that high dimensional feature space is useless for the similarity retrieval of multimedia information. Since the point distribution of the feature space is skewed, the intrinsic dimensionality (effective dimensionality) in a local region can be much less than the dimensionality of the feature space. Hence, we might have indistinctive nearest neighbors in one region but might have distinctive ones in another. The distinctiveness of the nearest neighbor differs from one local region to another.

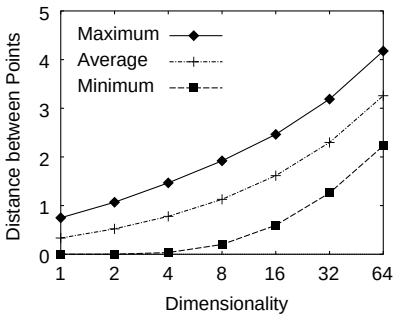


Figure 1. Distances among 100k points generated at random in a unit hypercube [2].

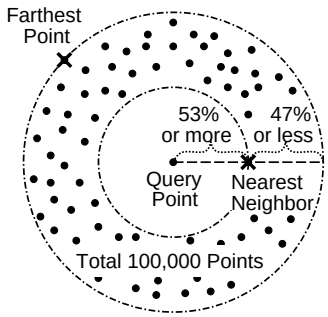


Figure 2. Case of 64 dimensions of Figure 1.

From the above perspective, we noticed that we should distinguish indistinctive nearest neighbors from distinctive ones in order to achieve efficient similarity retrieval of multimedia information. If we detect the distinctiveness of the nearest neighbor, we can change the behavior of the NN search to reduce the search cost and to provide more information for users. To this aim, we devised a new NN search algorithm: distinctiveness-sensitive nearest-neighbor search. This algorithm determines the distinctiveness of the nearest neighbor based on the intrinsic dimensionality during search operation. If the nearest neighbor is determined to be indistinctive, the search quits and returns a partial result. This enables us not only to cut down search cost but also to notify users whether the nearest neighbor is distinctive or not. These advantages are especially beneficial to the interactive retrieval system which requires quick response as well as informative results.

The contribution of this paper is threefold:

(1) The notion of distinctiveness is introduced to nearest neighbors.

Through theoretical and experimental evaluation, we demonstrate how important it is to distinguish indistinctive nearest neighbors from distinctive ones.

(2) A way of detecting the distinctiveness of the nearest neighbor is presented.

We devised a probabilistic method which determines the distinctiveness of the nearest neighbor.

(3) A new type of NN search method, i.e., distinctiveness-sensitive nearest-neighbor search, is presented.

We devised a new NN search method which detects the distinctiveness of nearest neighbors during search operation. The benefit of the proposed method is demonstrated through experimental evaluation.

This paper is organized as follows. Section 2 shows how indistinctive nearest neighbors affect the similarity retrieval of multimedia information. Section 3 describes how to estimate the distinctiveness of nearest neighbors. Section 4 presents the distinctiveness-sensitive NN search algorithm. The results of experiments are shown in Section 5. Section 6 contains conclusions.

2. Distinctiveness of NNs in feature space

2.1. Distinctiveness of NNs in uniform distribution

When we use NN search, we expect that found neighbors are much closer than the others. However, this intuition is sometimes incorrect in high dimensional space. For example, when points are uniformly distributed in a unit hypercube, the distance between two points is almost the same for any combination of two points. Figure 1 shows the minimum, the average, and the maximum of the distances among 100,000 points generated at random in a unit hypercube. As shown in the figure, the minimum of the distances grows drastically as the dimensionality increases and the ratio of the minimum to the maximum increases up to 24% in 16 dimensions, 40% in 32 dimensions, and 53% in 64 dimensions. Thus, the distance to the nearest neighbor is 53% or more of the distance to the farthest point in 64 dimensional space (Figure 2). In this case, we can consider the nearest neighbor to be *indistinctive*, because the difference between the nearest neighbor and the others is negligible, i.e., the other points are as close to the query point as the nearest neighbor is. From the perspective of the similarity retrieval, when the nearest neighbor is indistinctive, the nearest neighbor has almost the same similarity with the others and does not have distinctive similarity to the query.

As Figure 1 shows, indistinctive nearest neighbors are more likely to occur as dimensionality increases. This characteristics can be verified by estimating the distance to k-th nearest neighbor. When N points are distributed uniformly within the hypersphere whose center is the query point, the expected distance to k-th nearest neighbor d_{kNN} is obtained as follows[4]:

$$E\{d_{kNN}\} \approx \frac{\Gamma(k+1/n)}{\Gamma(k)} \frac{\Gamma(N+1)}{\Gamma(N+1+1/n)} r, \quad (1)$$

where n is the dimensionality of the space and r is the radius of the hypersphere. Then, the ratio of the $(k+1)$-NN

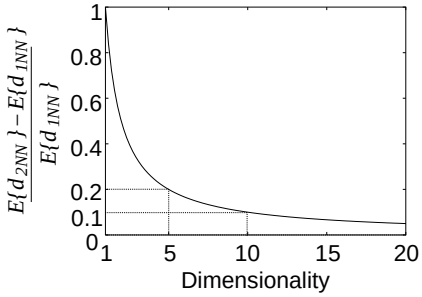


Figure 3. Relative difference between the first and the second nearest neighbor.

distance to the k-NN distance is obtained as follows[4]:

$$\frac{E\{d_{(k+1)NN}\}}{E\{d_{kNN}\}} \approx 1 + \frac{1}{kn}. \quad (2)$$

Thus, when points are distributed uniformly around the query point, it is expected that the difference between the k-th and $(k+1)$-th nearest neighbors decreases as the dimensionality increases. This implies that indistinctive nearest neighbors are more likely to occur in high dimensional space than in low dimensional space.

Equation (2) also indicates that the ratio of $d_{(k+1)NN}$ to d_{kNN} decreases monotonically as k increases. Therefore, the maximum of the ratio between two nearest neighbors is obtained for the first and the second nearest neighbor. From Equation (2), the expected ratio of d_{2NN} to d_{1NN} is approximated as follows:

$$\max_k \frac{E\{d_{(k+1)NN}\}}{E\{d_{kNN}\}} = \frac{E\{d_{2NN}\}}{E\{d_{1NN}\}} \approx 1 + \frac{1}{n}. \quad (3)$$

By subtracting one from both sides, we can estimate the relative difference between the first and the second nearest neighbor when the dimensionality is n:

$$\frac{E\{d_{2NN}\} - E\{d_{1NN}\}}{E\{d_{1NN}\}} \approx \frac{1}{n}. \quad (4)$$

This equation shows that the relative difference between the first and the second nearest neighbor decreases monotonically as the dimensionality increases (Figure 3). Only 20% difference is expected at 5 dimensions, and only 10% at 10 dimensions. This equation clearly shows that we could not expect strong distinctiveness of nearest neighbors for high dimensional uniform distribution.

2.2. Intrinsic dimensionality of feature space

As shown above, indistinctive nearest neighbors are more likely to appear in high dimensional space, and the expected relative difference between the first and the second nearest neighbor is inversely proportional to the dimensionality of the distribution. However, this does not mean that high dimensional feature space is useless. We should note that the discussion above is based on the uniform distribution. If the data distribution over the entire feature space is uniform, we can say that the feature space is useless, but in real applications, the data distribution is not uniform at all.

Instead of assuming the uniform distribution to the entire space, we should employ the intrinsic dimensionality (or effective dimensionality) which is determined by a local characteristics of the data distribution[4]. For example, when the data distribution is governed by a number of dominant dimensions, the intrinsic dimensionality is given by the number of such dominant dimensions. In addition, the intrinsic dimensionality may not be consistent over the data set but vary from one local region to another .

In real applications, we can expect that the data distribution is so skewed that the intrinsic dimensionality could be much smaller than the dimensionality of the feature space. Therefore, we might have indistinctive nearest neighbors in one region but could have distinctive ones in another. In a region with low intrinsic dimensionality, we can expect distinctive nearest neighbors, while we cannot expect distinctive ones in a region with high intrinsic dimensionality. Thus, the intrinsic dimensionality is the important clue for estimating the distinctiveness of the nearest neighbor. In Section 3, we present a probabilistic method which determines the distinctiveness of the nearest neighbor based on intrinsic dimensionality.

2.3. Harmful effect of indistinctive NNs

Indistinctive nearest neighbors have harmful effect on the similarity retrieval with the following respects:

- NN search performance is degraded.

  When the nearest neighbor is indistinctive, there exist many points that have almost the same similarity with the nearest neighbor. Since these points are very strong candidates for the nearest neighbor, NN search operation is forced to examine many points before determining the true nearest neighbor. This degrades the performance of NN search operation.

- Less informative result is returned.

  When the nearest neighbor is indistinctive, NN search operation returns the closest point among many strong candidates that have almost the same similarity with the nearest neighbor. This means that all of the candidates have slight difference with each other. It is not informative for users to choose the nearest neighbor from plenty of similar candidates.

These effects are extremely harmful to the retrieval systems with human-computer interaction. When the nearest neighbor is indistinctive, the system forces users to wait until less informative result is answered with slow response. Thus, it is necessary to handle indistinctive nearest neighbors ap-

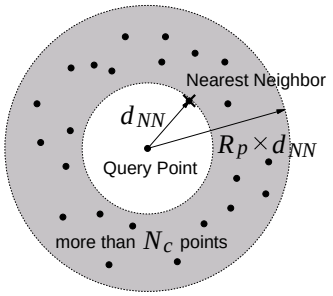


Figure 4. Definition of the indistinctive NN.

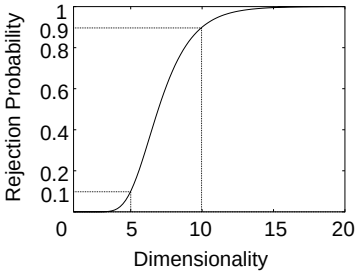


Figure 5. Rejection probability ($R_p = 1.84471$ and $N_c = 48$).

propriately in order to achieve efficient similarity retrieval of multimedia information.

3. Estimating the distinctiveness of NNs based on intrinsic dimensionality

3.1. Determination of indistinctive NNs

As mentioned above, the intrinsic dimensionality plays an important role in determining the distinctiveness of the nearest neighbor. Therefore, we devised a probabilistic method which determines the distinctiveness of the nearest neighbor based on the intrinsic dimensionality.

In the first place, we coined a definition of indistinctive nearest neighbors as follows:

Definition 1 *Let d_{NN} be the distance from the query point to a nearest neighbor. Then, the nearest neighbor is indistinctive if more than or equal to N_c points exist within the range of d_{NN} to $R_p \times d_{NN}$ from the query point.*

Here, $R_p(> 1)$ and $N_c(> 1)$ are controllable parameters (Figure 4). R_p determines the proximity of the query point and N_c determines the congestion of the proximity. For example, we set 1.84 to R_p and 48 to N_c at the experiment described in Section 5.

As dimensionality increases, we are more likely to have plenty of points in the similar distance with the nearest neighbor. This causes congestion in the proximity of the query point. The above definition determines indistinctive nearest neighbors by detecting the congestion in the proximity. This characteristic can be clearly stated by estimating the probability of congestion. We call this probability the *rejection probability*. When points are distributed uniformly in a local region with the intrinsic dimensionality n, the probability that N_c points exist in the proximity specified by R_p is obtained as follows:

$$Pr\{N_c \text{ or more points in } R_p\} = (1 - (1/R_p)^n)^{N_c}. \quad (5)$$

According to the definition above, the nearest neighbor is indistinctive when N_c points exist in the proximity specified by R_p. Therefore, Equation (5) corresponds to the probability that the nearest neighbor is regarded as being indistinctive when the intrinsic dimensionality is n. This probability increases monotonically as the intrinsic dimensionality increases. Equation (5) clearly shows that the congestion is more likely to occur when the intrinsic dimensionality is high. Figure 5 shows the rejection probability when R_p is 1.84471 and N_c is 48.

3.2. Finding appropriate setting of the parameters R_p and N_c

The parameters R_p and N_c play an essential role in the definition of the indistinctive nearest neighbor. We present here a way to find the appropriate setting of R_p and N_c. As Equation (5) shows, the rejection probability is the function of the intrinsic dimensionality n, and the form of the function is controlled by the parameters R_p and N_c. The parameter setting presented below consists of two steps. In the first step, we choose what intrinsic dimensionality to be regarded as having indistinctive nearest neighbors. In the second step, we locate two control points in order to control the form of the function, i.e., the cut-off point (ν_c, ρ_c) and the rejection point (ν_r, ρ_r). Then, the parameters R_p and N_c are determined from these four parameters.

In the first step, we choose what intrinsic dimensionality to be regarded as having indistinctive nearest neighbors. We call this dimensionality the *rejection dimensionality*. We can choose it basing on the expected relative difference described in Section 2.1. As shown in Equation (4) and Figure 3, the expected relative difference between the first and the second nearest neighbor is inversely proportional to the intrinsic dimensionality. Only 20% difference is expected at 5 dimensions, and only 10% at 10 dimensions. Basing on this expected relative difference, we can choose the rejection dimensionality. For example, if we don't care 20% difference, we choose 5 for the rejection dimensionality; if we don't care 10% difference, we choose 10 for the rejection dimensionality.

In the second step, we locate two control points in order to control the form of the function. Since we have chosen the rejection dimensionality in the first step, the ideal form of the function is the step function whose value is zero for the dimensionality less than the rejection dimensionality and one for the dimensionality greater than or equal to the rejection dimensionality. However, the composition of

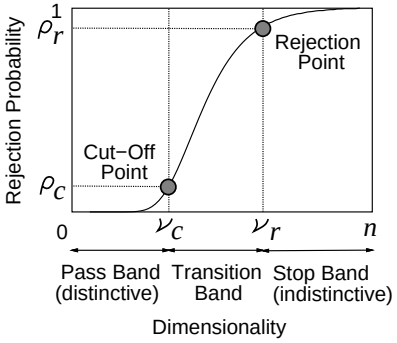
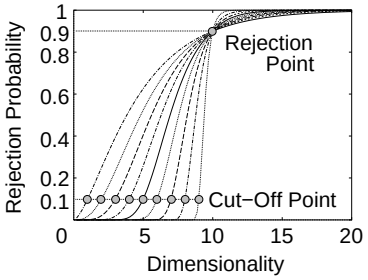


Figure 6. Controlling the rejection probability.

ν_c	R_p	N_c
1	1.31861	1.62113
2	1.41441	3.32326
3	1.51957	6.86386
4	1.65332	16.0256
5	1.84471	48.0277
6	2.15959	232.432
7	2.79551	3070.99
8	4.67486	525245
9	21.8437	2.60715×10^{12}

(a) Parameters R_p and N_c



(b) Rejection probability

Figure 7. The parameters and the rejection probability for different ν_c ($\nu_r = 10$, $\rho_c = 0.1$, and $\rho_r = 0.9$).

this form is infeasible since it is a discontinuous function. Therefore, we must choose a practical form. We locate two control points by analogy with the low pass filter: the cut-off point (ν_c, ρ_c) and the rejection point (ν_r, ρ_r) (Figure 6). Here, ν_r is the rejection dimensionality chosen in the first step and ρ_r is the rejection probability at ν_r, while ν_c is the cut-off dimensionality and ρ_c is the rejection probability at ν_c ($0 < \rho_c < \rho_r < 1$ and $0 < \nu_c < \nu_r$). The range of $n < \nu_c$ corresponds to the pass band where the intrinsic dimensionality n is regarded as having distinctive nearest neighbors. The range of $n > \nu_r$ corresponds to the stop band where the intrinsic dimensionality n is regarded as having indistinctive nearest neighbors. The range of $\nu_c < n < \nu_r$ corresponds to the transition band. Once we locate two control points, we can determine the parameters R_p and N_c by solving the following simultaneous equations:

$$(1 - (1/R_p)^{\nu_c})^{N_c} = \rho_c \quad (6)$$

$$(1 - (1/R_p)^{\nu_r})^{N_c} = \rho_r \, . \quad (7)$$

By the elimination of N_c the following equation is obtained:

$$\frac{\log(1 - (1/R_p)^{\nu_c})}{\log(1 - (1/R_p)^{\nu_r})} = \frac{\log \rho_c}{\log \rho_r} \, . \quad (8)$$

This equation cannot be arithmetically solved. However, since the left side of the equation increases monotonically as R_p increases, it can be easily solved with numerical methods, e.g., Newton's method. Once R_p is determined, N_c is obtained from R_p as follows:

$$N_c = \frac{\log \rho_c}{\log(1 - (1/R_p)^{\nu_c})} . \quad (9)$$

Now, by Equation (8) and (9), we can determine R_p and N_c from the two control points (ν_c, ρ_c) and (ν_r, ρ_r).

As shown above, we can determine the parameters R_p and N_c by locating two control points. The remaining question is where the control points should be located. The answer is that it is a trade-off between the sensitivity and the locality. By putting the cut-off dimensionality ν_c near to the rejection dimensionality ν_r, we can have steep slope in the transition band (i.e., $\nu_c < n < \nu_r$). However, this causes the increase in R_p and widens the region to be examined for determining the distinctiveness of the nearest neighbor. As shown in the following example, R_p increases significantly as ν_c approaches ν_r. When R_p is too large, we cannot exploit a local characteristics of the data distribution. Ideally, we should have both high sensitivity and high locality. But we need to find a compromise between the sensitivity and the locality as shown in the following example.

3.3. Sample setting of the parameters R_p and N_c

Here, we show an example of the parameter setting according to the procedure described above. In the first step, we choose the rejection dimensionality based on the expected relative difference between the first and the second nearest neighbor. We choose 10 for the rejection dimensionality since only 10% difference is expected. In the second step, we locate two control points with making a compromise between the sensitivity and the locality. Firstly, we need to choose the rejection probability at the cut-off and the rejection dimensionality, i.e., ρ_c and ρ_r respectively. Here, we use 0.1 for ρ_c and 0.9 for ρ_r. Secondly, we need to choose the cut-off dimensionality ν_c. Figure 7 shows the parameters and the rejection probability obtained for different ν_c. As shown in Figure 7 (b), we have high sensitivity when ν_c is close to ν_r. However, as shown in Figure 7 (a), the parameters R_p and N_c increase significantly as ν_c approaches ν_r. Therefore, we need to find a compromise between the sensitivity and the locality. When ν_c is 7, R_p is 2.79551 and N_c is 3070.99. This means that the congestion is detected only when 3071 or more points exist in the proximity of the query point. Apparently, 3071 is too large; the locality of the proximity specified by R_p is very low. When ν_c is 5, the selectivity is still high and N_c and R_p are quite small. Therefore, we choose 5 for ν_c. Finally, the two control points are located at (5, 0.1) and (10, 0.9) and R_p and N_c are obtained as 1.84471 and 48. Figure 5 shows the rejection probability with the setting described here. In

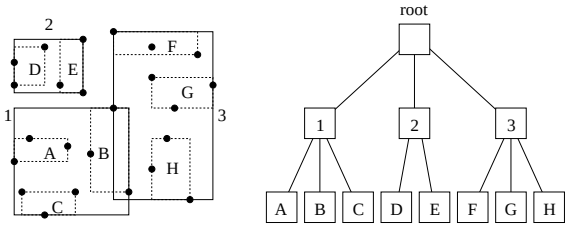


Figure 8. Structure of the R-tree.

Section 5, this parameter setting is used for the experimental evaluation.

4. Distinctiveness-sensitive NN search for multidimensional index structures

In order to circumvent the harmful effect of indistinctive nearest neighbors, we developed a new NN search algorithm for multidimensional index structures with applying the distinctiveness estimation method described in the previous section. The algorithm tests the distinctiveness of the nearest neighbor in the course of search operation. Then, when it finds the nearest neighbor to be indistinctive, it quits the search and returns a partial result. We call this NN search as the *Distinctiveness-Sensitive Nearest-Neighbor Search* since it is sensitive to the distinctiveness of the nearest neighbor. This algorithm not only enables us to determine the distinctiveness of the nearest neighbor but also enables us to cut down the search cost as shown in the following section.

4.1. NN search algorithms for multidimensional index structures

The distinctiveness-sensitive NN search algorithm is designed for multidimensional index structures that splits the data space into the nested hierarchy of regions, e.g., the SS-tree[5], the VAMSplit R-tree[6], the X-tree[7], the SR-tree[2], the LSD^h-tree[8], the Hybrid tree[9], the IQ-tree[10], etc. Figure 8 illustrates the hierarchical structure of the R-tree. Each node of the tree structure corresponds to a region of the data space. Non-internal nodes store data points, while internal nodes store the information on child nodes, i.e., region specifications (e.g., location, size, shape, etc.) and pointers to child nodes. The hierarchical structure permits NN search operation to be completed with examining only some part of the feature space. This reduces CPU time and disk I/O of NN search operation. Therefore, these index structures are widely used for the acceleration of the similarity retrieval of multimedia information.

We designed the new algorithm with extending the NN search algorithm presented by Hjaltason et al.[11] (This algorithm will be called the *basic* NN search algorithm hereafter in order to be distinguished from the distinctiveness-sensitive NN search algorithm). The basic NN search algorithm finds the nearest neighbor of a given query point as follows. The search starts from the root node of the index structure. At every internal node, the distances from the query point to the child nodes are computed and the child nodes are enqueued into a priority queue in ascending order of the distance. Then, one node is dequeued from the priority queue and the search proceeds to the dequeued node. Thus, nodes are visited in ascending order of the distance from the query point. For example, in Figure 9, regions will be searched in the order of Region 1, Region 2, and Region 3. On the other hand, every time a non-internal node is visited, the distances from the query point to the data points in the node are computed. If it is the first visit of a non-internal node, the closest point in the node is chosen for the candidate of the nearest neighbor; otherwise, the closest point in the node is compared with the latest candidate and the closer one is chosen for the new candidate. The distance to the candidate gives the upper bound of the distance to the nearest neighbor, because the nearest neighbor cannot be farther from the query point than the candidate is. On the other hand, the distance to the first node in the priority queue gives the lower bound of the distance to forthcoming candidates, because the first node of the priority queue is the closest among such nodes that are not visited so far. For example, Figure 9 illustrates the case where Region 1 is visited and the closest point in Region 1 is chosen for the candidate. Region 2 and Region 3 are still in the priority queue. The candidate gives the upper bound of the distance to the nearest neighbor, while the distance to the first node of the priority queue, i.e., the distance to Region 2, gives the lower bound of the distance to forthcoming candidates. The search continues until the lower bound of the distance to forthcoming candidates exceeds the upper bound of the distance to the nearest neighbor, i.e., until no node in the priority queue is closer to the query point than the candidate is. Finally, the nearest neighbor is obtained as the final candidate. Although the description above is the search finding the first nearest neighbor, it can be easily extended to the

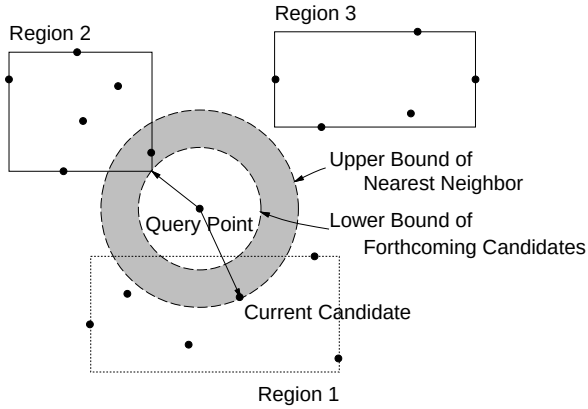


Figure 9. NN search with multidimensional index structure.

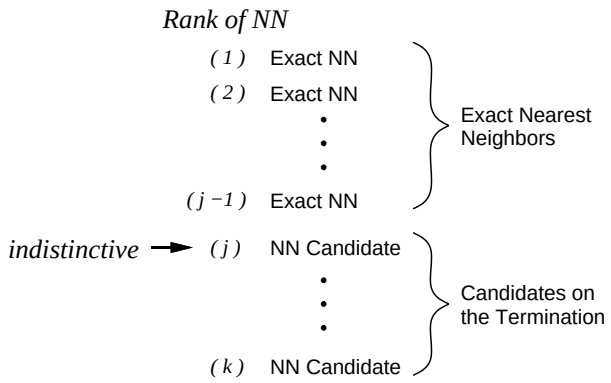


Figure 10. Search result of the distinctiveness-sensitive NN search.

k-nearest-neighbor search, i.e., finding k nearest neighbors.

4.2. Distinctiveness-sensitive NN search algorithm

The distinctiveness-sensitive NN search algorithm is an extension of the basic NN search algorithm described above. The main idea of the extension is counting the number of points that are located within the proximity of the query point in the course of search operation. The proximity is specified by the parameter R_p as in Definition 1. If the congestion of the proximity is detected, the search is terminated and a partial result is returned. The congestion is determined by the parameter N_c as in Definition 1, i.e., the case where the number of points in the proximity exceeds N_c is regarded as the congestion. In order to count points in the proximity, the algorithm takes advantage of the lower bound and the upper bound of the distance to the nearest neighbor d_{NN}. As stated in the description of the basic NN search, the distance to the first node in the priority queue and the distance to the candidate of the nearest neighbor gives the lower and the upper bound of d_{NN} (Figure 9). These bounds also give the bounds of the proximity. According to the definition of the proximity in Definition 1, the range of the proximity is $R_p \times d_{NN}$. When the lower and the upper bound of d_{NN} is $LB\{d_{NN}\}$ and $UB\{d_{NN}\}$ respectively, the lower bound and the upper bound of the range of the proximity can be obtained as $R_p \times LB\{d_{NN}\}$ and $R_p \times UB\{d_{NN}\}$. Since d_{NN} is obtained only at the end of the search, the algorithm uses $UB\{d_{NN}\}$ and $R_p \times LB\{d_{NN}\}$ to count points in the proximity. Because $d_{NN} \leq UB\{d_{NN}\}$ and $R_p \times LB\{d_{NN}\} \leq R_p \times d_{NN}$, the number of points in the proximity is more than or equal to the number of points that are located in the range of $UB\{d_{NN}\}$ to $R_p \times LB\{d_{NN}\}$. Therefore, the congestion can be detected during the search operation by testing whether the number of points located in the range of $UB\{d_{NN}\}$ to $R_p \times LB\{d_{NN}\}$ is greater than or equal to N_c.

With employing this congestion detection method, the distinctiveness-sensitive NN search algorithm tests the distinctiveness of the nearest neighbor in the course of search operation. Then, when it finds the nearest neighbor to be indistinctive, it quits the search and returns the partial result as shown in Figure 10. When j-th nearest neighbor is found to be indistinctive during k-nearest neighbor search, the search is terminated and candidates on the termination are returned for j-th to k-th nearest neighbors. Thus, the mixture of the exact nearest neighbors and the nearest neighbor candidates is returned when the indistinctive nearest neighbor is found during the search operation. This algorithm not only enables us to determine the distinctiveness of nearest neighbors but also enables us to cut down the search cost since this algorithm avoids pursuing exact answers for indistinctive nearest neighbors.

It should be noted that the distinctiveness-sensitive NN search is different from the approximate NN-search algorithms [12, 13]. The approximate NN search [12] terminates when the ratio of the upper bound to the lower bound of d_{NN} is reduced to less than or equal to $(1 + \epsilon)$. Here ϵ is the controllable parameter of an error bound. At this point, the candidate is an approximate answer whose error in the distance from the query point is less than or equal to ϵ. Thus ϵ determines the precision of an approximate answer. The PAC NN search (probably approximately correct NN search)[13] also finds an approximate answer but estimates the lower bound of d_{NN} from the distance distribution of the dataset under the specified probability. Thus, the PAC NN search returns an approximate answer under the given precision and the given probability. While these algorithms focus on the precision of an answer, the distinctiveness-sensitive NN search algorithm focuses on the distinctiveness of the nearest neighbor. As shown in Figure 10, the proposed algorithm returns an inexact answer only when the nearest neighbor is determined to be indistinctive. As long as the nearest neighbor is distinctive, the algorithm returns the exact answer. As its name implies, the salient feature of the proposed algorithm is the sensitivity to the distinctiveness of the nearest neighbor.

5. Experimental evaluation

5.1. Evaluation with synthetic datasets

In order to evaluate the characteristics of the distinctiveness-sensitive NN search, we synthesized datasets having various intrinsic dimensionality. A dataset having the intrinsic dimensionality ν is synthesized by generating n-dimensional points $(x_1, \ldots, x_n)$ in accordance with the following rule:

$$x_i = \begin{cases} U(0,1) & (1 \leq i < \nu) \\ \dfrac{1}{\sqrt{n-\nu+1}} U(0,1) & (i = \nu) \\ x_\nu & (\nu < i \leq n), \end{cases} \tag{10}$$

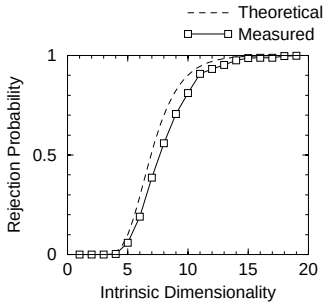


Figure 11. Rejection probability.

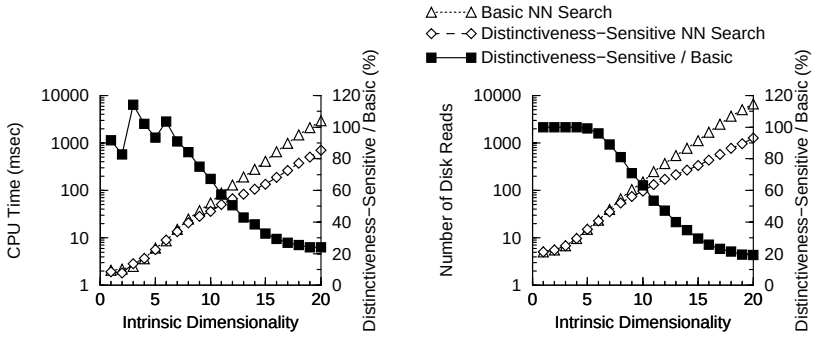


Figure 12. Performance with the synthetic datasets.

where $U(0,1)$ is the uniform distribution in the range of 0 to 1. We synthesized datasets having the intrinsic dimensionality $1 \sim 20$. The dimensionality of the data space, i.e., n in Equation (10), is 20. Each dataset contains 1,000,000 points.

Experiments are conducted on Sun Microsystems Ultra 60 (CPU: UltraSPARC-II 360MHz, main memory: 512 Mbytes, OS: Solaris 2.6). Programs are implemented in C++. The VAMSplit R-tree[6] is employed as the index structure, since it has an efficient static construction algorithm suitable for the NN search in high dimensional space. We measured the average performance of 1,000 random trials of finding the first nearest neighbor. Query points are selected at random from the data set. The parameters R_p and N_c are set to 1.84471 and 48 respectively according to the setting described in Section 3.3.

Figure 11 and 12 shows the result of the experiment. In both figures, the horizontal axis indicates the intrinsic dimensionality of the datasets. Figure 11 shows the rejection probability, i.e., the ratio of such a case that the nearest neighbor is determined to be indistinctive. The measured probability is very close to the theoretical one. This proves the validity of the parameter setting presented in Section 3.3. Figure 12 shows the CPU time and the number of disk reads. When the intrinsic dimensionality is low, the difference between the basic NN search and the distinctiveness-sensitive NN search is negligible. However, when the intrinsic dimensionality is high, both the CPU time and the number of disk reads are reduced remarkably. When the intrinsic dimensionality is 20, the CPU time is reduced by 76% and the number of disk reads is reduced by 81% compared with the basic NN search. This result demonstrates that the NN search performance can be improved by the distinctiveness-sensitive NN search.

5.2. Evaluation with real dataset

We evaluated the performance of the distinctiveness-sensitive NN search with applying it to the similarity retrieval of images. The data set is 60,195 images of Corel Photo Collection contained in the product called Corel Gallery 1,000,000. The similarity of images is measured in terms of the color histogram. Munsell color system is used for the color space. It is divided into nine subspaces: black, gray, white, and six colors. For each image, histograms of four sub-regions, i.e., upper-left, upper-right, lower-left and lower-right, are calculated in order to take account of the composition of the image. Four histograms are concatenated to compose a 36-dimensional feature vector. Similarity of images is measured by Euclidean distance among these 36-dimensional vectors. We measured the performance of finding 100 nearest neighbors with employing every image as a query. Therefore, one of the found nearest neighbors is the query image itself. The platform is the same with the experiment of synthetic datasets and the VAMSplit R-tree[6] is employed as the index structure.

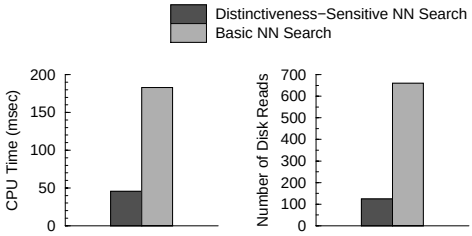


Figure 13. Performance with the real dataset.

We compared the cost of the distinctiveness-sensitive NN search with that of the basic NN search. As shown in Figure 13, both the CPU time and the number of disk reads of the distinctiveness-sensitive NN search are remarkably reduced compared with those of the basic NN search. The CPU time is reduced by 75% and the number of disk reads is reduced by 72%. This result demonstrates that the distinctiveness-sensitivity NN search enables us to cut down the search cost. Although the reduction rate depends on the data set, this cost reduction capability of the distinctiveness-sensitive NN search should be advantageous to the interactive similarity retrieval systems that need quick response to users.

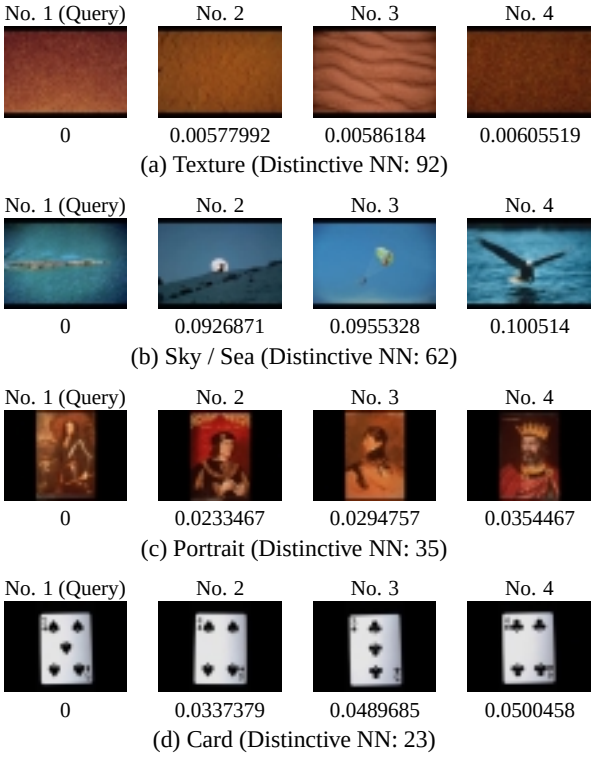
Table 1 shows the number of distinctive nearest neighbors found by the distinctiveness-sensitive NN search. Because we employed every image as a query, the total of the

Table 1. Number of distinctive NNs.

# of Distinctive NNs	# of Occurrence
100	0
80 ∼ 99	46
60 ∼ 79	1
40 ∼ 59	66
20 ∼ 39	124
10 ∼ 19	269
2 ∼ 9	13649
1	46040
Total	60195

Table 2. Categories of search results.

Category (Observed by Hand)	# of Distinctive NNs (at most)
Texture	92
Sky / Sea	62
Portrait	35
Card	23
Firework	23
Sunset	19
Kung-Fu	18
Steamship	17
Desert	16



(a) Texture (Distinctive NN: 92)

(b) Sky / Sea (Distinctive NN: 62)

(c) Portrait (Distinctive NN: 35)

(d) Card (Distinctive NN: 23)

Figure 14. Examples of search results.

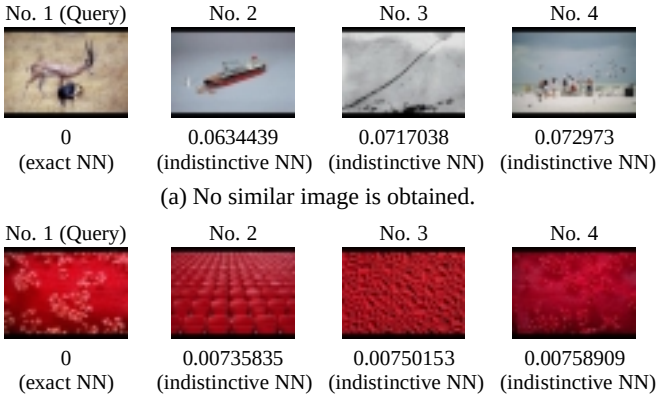
occurrence is equal to the number of images in the data set. Table 2 shows what kind of images are retrieved as distinctive nearest neighbors. We examined such search results that have relatively many distinctive nearest neighbors and then determined by hand what kind of images are retrieved as distinctive nearest neighbors. Figure 14 shows examples of search results when the number of distinctive nearest neighbors is relatively large. Due to the limitation of space, top 4 images are shown. The numerical value under each image is the distance from the query to the image. We can see that similar images are successfully retrieved by the distinctiveness-sensitive NN search.

The amazing result of Table 1 is that we obtained only one distinctive nearest neighbor for 46,040 images. Since each query image is chosen from the images in the data set, the obtained distinctive nearest neighbor is the query itself. Therefore, we obtained no distinctive nearest neighbor except for the query image. In this case, the query image is surrounded by plenty of neighbors that have almost the same similarity to the query. Since Corel Photo Collection collects wide variety of photos, it is not strange that an image has no similar one in the collection. In addition, the collection contains some texture photos. In this case, we have plenty of images with small difference. Figure 15 shows examples of search results when we obtained no distinctive nearest neighbors except for the query image. Due to the space limitation, top 4 images are shown. Figure 15 (a) is the case that the query image is surrounded by plenty of dissimilar images, while Figure 15 (b) is the case that the query image is surrounded by plenty of images with small difference. These examples illustrate that the distinctiveness-sensitive NN search allows us to see how retrieved images are significantly close to the query image. This capability should be advantageous to the interactive information retrieval systems.

In order to validate the results of the distinctiveness-sensitive nearest neighbor search, we plotted cumulative distribution of points around the query points used in the above examples. Figure 16 shows the cumulative distribution of points around such query points that are used in Figure 14 (a), 14 (b), and 15 (a). The number of distinctive nearest neighbors obtained by the search are 92, 62, and 1, respectively. The horizontal axis indicates the distance from the query point normalized by the distance to the 1st nearest neighbor of each query point, while the vertical axis indicates the cumulative frequency of points. As the figure shows, the distribution of nearest neighbors differs distinctively from one query point to another. The broader the distribution is, the more distinctive nearest neighbor is obtained. This tendency meets the aim of the distinctiveness sensitive NN search, i.e., detecting the congestion of nearest neighbors.

6. Conclusion

This paper shows that the similarity retrieval of multimedia information can be improved with the employment



(a) No similar image is obtained.

(b) Images with small difference are obtained.

Figure 15. Examples of results when no distinctive NN is found except for the query.

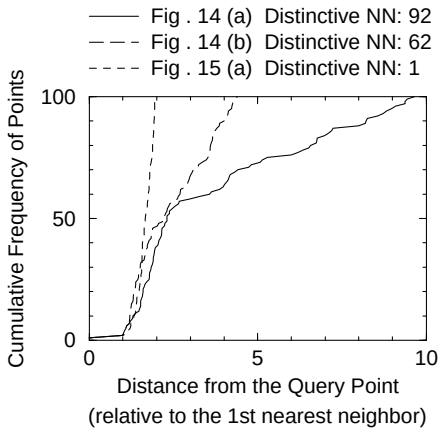


Figure 16. Cumulative distribution of points around query points.

of a new NN search algorithm, the distinctiveness-sensitive nearest-neighbor search. Indistinctive nearest neighbors are likely to occur in high dimensional space. They affect NN search performance because the search operation must examine many strong candidates to determine the true nearest neighbor. In addition, from the perspective of the similarity retrieval, it is less informative to choose the true nearest neighbor from plenty of strong candidates having almost the same similarity.

The distinctiveness-sensitive NN search circumvents this problem by detecting indistinctive nearest neighbors during the search operation. When an indistinctive nearest neighbor is detected, the operation is terminated and a partial result is returned. This reduces both CPU time and disk I/O, and besides enables us to distinguish the distinctive nearest neighbors from the indistinctive ones. These capabilities are especially advantageous to the interactive information retrieval systems since users not only enjoy quick response but also understand how distinctive the nearest neighbor is.

Acknowledgment

This research is partly supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Creative Basic Research (09NP1401) and Encouragement of Young Scientists (12780251).

References

[1] S. Berchtold, C. Bohm, B. Braunmuller, D. Keim, and H.-P. Kriegel, “Fast Parallel Similarity Search in Multimedia Databases,” Proc. of the 1997 ACM SIGMOD, Tucson, USA (May 1997) pp. 1–12.

[2] N. Katayama and S. Satoh, “The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries,” Proc. of the 1997 ACM SIGMOD, Tucson, USA (May 1997) pp. 369–380.

[3] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When Is “Nearest Neighbor” Meaningful?,” Proc. of the 7th Int. Conf. on Database Theory, Jerusalem, Israel, pp.217–235, Jan. 1999.

[4] K. Fukunaga, “Introduction to Statistical Pattern Recognition (2nd ed.),” Academic Press, 1990.

[5] D. A. White and R. Jain, “Similarity Indexing with the SS-tree,” Proc. of the 12th Int. Conf. on Data Engineering, New Orleans, USA, pp.516–523, Feb. 1996.

[6] D. A. White and R. Jain, “Similarity Indexing: Algorithms and Performance,” Proc. SPIE Vol.2670, San Diego, USA, pp.62–73, Jan. 1996.

[7] S. Berchtold, D. A. Keim, and H.-P. Kriegel, “The X-tree: An Index Structure for High-Dimensional Data,” Proc. of the 22nd VLDB Conf., Bombay, India, pp.28–39, Sep. 1996.

[8] A. Henrich, “The LSD^h-tree: An Access Structure for Feature Vectors,” Proc. of the 14th Int. Conf. on Data Engineering, Orlando, USA, pp.362–369, Feb. 1998.

[9] K. Chakrabarti and S. Mehrotra, “The Hybrid Tree: An Index Structure for High Dimensional Feature Spaces,” Proc. of the 15th Int. Conf. on Data Engineering, Sydney, Australia, pp.440–447, Mar. 1999.

[10] S. Berchtold, C. Bohm, H. V. Jagadish, H.-P. Kriegel, J. Sander, “Independent Quantization: An Index Compression Technique for High-Dimensional Data Spaces,” Proc. of the 16th Int. Conf. on Data Engineering, San Diego, USA, pp.577–588, Feb. 2000.

[11] G. Hjaltason and H. Samet, “Ranking in Spatial Databases,” 4th Int. Symp. on Spatial Databases, SSD’95, Portland, USA, pp.83–95, Aug. 1995.

[12] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An Optimal Algorithm for Approximate Nearest Neighbor Searching,” Proc. of the 5th Ann. ACM-SIAM Symposium on Discrete Algorithms, pp.573–582, 1994.

[13] P. Ciaccia and M. Patella, “PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces,” Proc. of the 16th Int. Conf. on Data Engineering, San Diego, USA, pp.244–255, Feb. 2000.