# 距離索引VP-treeにおける解絞り込みの一改良手法

宮本　裕也† 　獅々堀正幹† 　北 　研二††

† 徳島大学 工学部 知能情報工学科
†† 徳島大学 高度情報化基盤センター
E-mail: †{miyamoto,bori,kita}@is.tokushima-u.ac.jp

**あらまし**　マルチメディア・データベースでは検索効率を高めるために，多次元空間に基づく索引化法が使用される．しかし，この手法は距離尺度としてユークリッド距離を用いることが前提であるため，汎用性に欠ける．一方，距離公理の成立のみを前提とする距離空間に基づく索引化法は，ユークリッド距離以外の距離尺度も利用できるため，汎用性が高い．本稿では，距離空間索引化法の一つである VP-tree の改良法を提案する．VP-tree は検索時にルートノードから検索範囲に適合するノードを辿り，最終的に辿り着いたリーフノードにリンクされているオブジェクトとの距離を算出し，検索範囲に適合するかを調べる．しかし，リーフノードにおける距離計算が増大すると検索速度が遅くなってしまう．そこで，リーフノードにおける三角不等式を用いた絞り込み法に着目し，その改良法として問い合わせオブジェクトに対する最近傍点を三角不等式の基準点として用いる手法を提案する．この改良法により，検索範囲をより小さくし，距離計算回数を削減することが可能になる．実際に 10,000 件の画像データを用いて評価実験を行った結果，既存の手法に比べ検索時間を 5%〜12% 削減することができた．

**キーワード**　多次元 DB，マルチメディア DB，主記憶 DB

# An improved method to select candidates on metric index VP-tree

Yuya MIYAMOTO†, Masami SHISHIBORI†, and Kenji KITA††

† Department of Information Science and Intelligent System, the University of Tokushima
†† Center for Advanced Infomation Technology, the University of Tokushima
E-mail: †{miyamoto,bori,kita}@is.tokushima-u.ac.jp

**Abstract**　On multimedia databases, in order to realize the fast access method, indexing methods for the multi--dimension data space are used. However, since it is a premise to use the Euclid distance as the distance measure, this method lacks in flexibility. On the other hand, there are metric indexing methods which require only to satisfy distance axiom. Since metric indexing methods can also apply for distance measures other than the Euclid distance, these methods have high flexibility. This paper proposes an improved method of VP-tree which is one of the metric indexing methods. VP-tree follows the node which suits the search range from a route node at searching. And distances between a query and all objects linked from the leaf node which finally arrived are computed, and it investigates whether each object is contained in the search range. However, search speed will become slow if the number of distance calculations in a leaf node increases. Therefore, we paid attention to the candidates selection method using the triangular inequality in a leaf node. As the improved methods, we propose a method to use the nearest neighbor object point for the query as the datum point of the triangular inequality. It becomes possible to make the search range smaller and to cut down the number of times of distance calculation by these improved methods. From evaluation experiments using 10,000 image data, it was found that our proposed method could cut 5%〜12% of search time of the traditional method.

**Key words**　Multi-dimensional DB, Multimedia DB, Main storage DB

## 1. はじめに

近年，一次／二次記憶装置の低価格化と大容量化により，文書，画像，音楽，映像といったマルチメディアデータを個人向けの計算機にも大量に保存することが可能となった．それに伴い，大量に保存されたマルチメディアデータからユーザが所望するデータのみを素早く，かつ正確に検索する技術が必要となった．検索効率を向上させるためには，事前に格納データから特徴量を抽出し，その特徴量から索引 (インデックス) を構成する必要がある [1]．検索時には，そのインデックスにのみアクセスすることで適合データを取得する．そのため，インデックスの構成方法が検索効率を大きく左右する．

マルチメディアデータから抽出する特徴量は，一般にベクトル表現され，各特徴ベクトル間の距離の近さを類似性とみなし検索を行う [2] [3]．このような特徴ベクトルの索引化法，すなわち，多次元データのインデックスとしては，R-tree [4], R*-tree [5]，SS-tree [6]，SR-tree [7]，X-tree [8]，VA-FILE [9] などが提案されている．しかし，これらの手法は，距離尺度としてユークリッド距離を用いることが前提となっているため，その他の距離尺度に適用することができない．例えば，ユークリッド距離以外の距離尺度としては，多次元データの各次元間の相関を考慮した quadratic form 距離 [10]，文字列間の類似性を計る Edit 距離，画像間の構図の類似性を計る Earth Mover's Distance [11] などがある．

この問題を解決するために，距離のみに基づいてインデキシングする距離空間インデックスについての研究がなされている．多次元インデックスでは多次元空間上での特徴量の座標値をもとにインデックスを作成するのに対し，距離空間インデックスでは距離公理の成立のみを前提とし，特徴量間の距離情報のみを用いてインデックスを作成する．従って，ユークリッド距離以外の距離尺度であっても適用できる．距離空間インデックスは一般的に階層的インデックス木であり，空間 (データ集合) を距離情報に基づき再帰的に分割することにより検索の際の探索空間の縮小を図る．この空間の分割法の違いによって M-tree [12],VP-tree [13] [14]，MVP-tree [15]，MI-tree [16] などが提案されている．M-tree は空間分割の際にボトムアップ的にインデックス木を構成するため，分割した空間の間に共通領域が多くなり，検索効率が低下することが欠点とされている．これに対し，VP-tree は vantage point と呼ばれる基準点を用いて，超球により空間をトップダウン的に分割するため，分割空間に共通領域を生じさせない．検索時にはルートノードから検索範囲に適合するノードを辿り，最終的に辿り着いたリーフノードにリンクされているリーフオブジェクトに逐一アクセスし，距離を算出し検索範囲に適合するかを調べる．しかし，検索時に辿ったこれらのリーフノードにおける距離計算が全体の距離計算回数を増大させ，検索速度が遅くなる原因となっている．

本稿では，VP-tree のリーフノードにおける検索アルゴリズムの改良法を提案し，距離計算回数の削減を試みた．VP-tree のリーフノードにおける三角不等式を用いた解絞り込み法において，従来手法では vantage point を三角不等式の基準点に用いているのに対して，本手法では三角不等式の基準点と問い合わせオブジェクトとの距離が近ければ近いほど，解の絞り込み範囲が狭くなる点に着目した．そこで本手法では，三角不等式の基準点に問い合わせオブジェクトに対する最近傍点を用いることによって検索範囲を小さくし，距離計算回数を削減する．

本改良手法を実装する上で，通常は最近傍点を事前に特定することはできない．そこで本手法では，検索結果リスト内で問い合わせオブジェクトに最も近いオブジェクトを仮の最近傍点と見なすことで最近傍点による絞り込みを実現している．更に，仮の最近傍点を基準点とした三角不等式を利用するためには，仮の最近傍点とリーフノード内のすべてのオブジェクト間の距離が既知でなければならない．ここで，仮の最近傍点も事前に特定することはできないため，実質上すべてのオブジェクト間の距離が必要になる．そこで本手法では，インデキシングの際にオブジェクト間の距離を計算した距離リストファイルを構築する．ただし，大規模な距離リストファイルをオブジェクト毎に分割して構築することで，ファイルの読み込みサイズを軽減している．

本手法と同様に事前に構築した距離リストファイルを用いて解の絞り込みを行う手法として AESA(Approximating and Eliminating Search Algorithm) [19] が存在する．AESA と本手法との違いは，AESA が全オブジェクトを対象にして距離リストによる絞り込みを行うのに対し，本手法はリーフノード内のオブジェクトだけを対象にする点である．全オブジェクトを対象にする AESA では，必然的にファイルの読み込み回数が激増する問題がある．一方，vp-tree を用いると数件のリーフノード内のオブジェクトのみが対象となり，かつ，仮の最近傍点（基準点）が更新された場合にのみ読み込みを必要とするため，ファイルのアクセス回数を抑制することが可能になる．

以下，2 章では VP-tree の構築アルゴリズムと検索アルゴリズムを説明した後，リーフノードにおける絞り込み法について説明する．そして，第 3 章ではリーフノードにおける検索アルゴリズムの改良法を紹介する．第 4 章ではこの改良法を用いた実験と評価について述べる．最後に第 5 章では本稿のまとめを述べ，今後の課題，展望について述べる．

## 2. VP-tree

### 2.1 構築アルゴリズム

VP-tree の構築アルゴリズムを説明する．$N$ 個のデータから成るデータセット $S$ にインデキシングを行うとする．木の各ノードでは，以下に示すようにランダムなアルゴリズムによって vantage point(以後 $vp$ と記す) を選択する．

（1） データセットからランダムに仮の $vp$ を選択
（2） 仮の $vp$ から残りの $N-1$ 個のオブジェクトまでの距離を計算
（3） これらの距離の中間値と分散を計算
（4） 1〜3 を何回か繰り返し，分散が最大となる点を $vp$ とする

ルートノードに選ばれたその $vp$ から $S$ 中のすべてのデータに対する距離の中間値を $\mu$ とする．$d(p,q)$ を点 $p,q$ 間の距離とした場合，データセット $S$ は以下のように $S_1$ と $S_2$ に分割される．

$$S_1 = \{s \in S \mid d(s,vp) < \mu\}$$
$$S_2 = \{s \in S \mid d(s,vp) \geqq \mu\}$$

同様にして，この分割の操作を $S_1$ 及び $S_2$ に再帰的に適用することによりインデックスを生成する．$S_1$ と $S_2$ のようなすべての部分集合は VP-tree の 1 つのノードに相当している．また，リーフノードではいくつかのオブジェクトを格納する．

### 2.2 検索アルゴリズム

VP-tree における範囲指定検索 (range search) 及び件数指定検索 ($k$-nearest neighbor search) のアルゴリズムについて説明する．範囲検索とは問い合わせオブジェクトおよび検索範囲である半径を指定し，円の中心から半径の距離までの範囲にあるオブジェクトの集合を求める検索法である．また，件数指定検索とは問い合わせオブジェクト及び検索件数 $k$ を指定して，距離が近い順に上位 $k$ 件のオブジェクトの集合を求める検索法である．本稿では件数指定検索を用いて実験を行っているが，件数指定検索は範囲指定検索のアルゴリズムに基づいているため，これら 2 つの検索手法について述べる．

まず，範囲検索はルートノードから検索範囲に適合するノードを辿り，リーフにリンクされているリーフオブジェクトと問い合わせオブジェクトとの距離計算を行い，検索範囲に存在するオブジェクトを取得する．

一方，件数指定検索は検索初期値では検索半径は無限大とし，ルートから辿りオブジェクトを検索結果リストに加えていく．検索結果リストの検索数が指定された検索数を越えたら，距離が最大の検索オブジェクトを検索結果リストから削除し，検索結果リストの検索数が指定された検索数を越えないようにする．さらに検索結果リストの最大の距離を検索半径とする．これを繰り返して行うことによって検索半径が絞られ最終的に指定件数分の検索結果が得られる．

### 2.3 リーフノードにおける絞り込み法

2.2 で述べたように，従来の VP-tree では検索中に辿ったリーフノード内に存在する全てのオブジェクトにアクセスし，問い合わせオブジェクトとの距離計算を行っていた．その改善策として，リーフノード内の各オブジェクトの検索時に三角不等式を利用することで，次のようにして解候補を絞り込む手法 [17] が提案された．

リーフノードではリーフノードの $vp$ オブジェクトと各リーフオブジェクト間との距離を距離リストとして保持する．この $vp$ オブジェクトと各リーフオブジェクト間の距離により三角不等式を利用して距離計算回数の削減を行うことができる．問い合わせオブジェクトを $q$，検索範囲である半径を $r$，リーフノードの $vp$ オブジェクトを $v$，リーフノードにリンクするリーフオブジェクトを $o$ とすると，以下の定理が成り立つ．

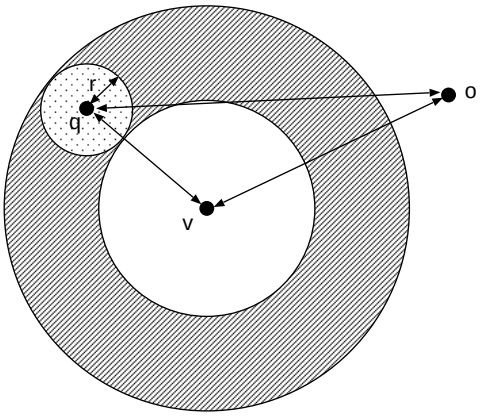


図 1　$vp$ を基準点とする絞り込み

```
Input : q , r , L
Output : L
SearchLeaf (q , r , L)
{
    foreach o (全リーフオブジェクト) {
        if ( |d(v , o) − d(v , q)| ≦ r ) {
            if ( d(o , q) ≦ r ) {
                Lにoを加え, 距離の最大値をrとする ;
            }
        }
    }
}
```

q : 問い合わせオブジェクト
r : 検索の半径
o : リーフオブジェクト
v : カレントリーフのvpオブジェクト
L : 検索結果リスト

図 2　リーフノードにおける検索アルゴリズム

**定理 1**

$|d(v,o) - d(v,q)| > r$ が成り立つならば，
リーフオブジェクト $o$ は検索範囲に存在しない

証明：　三角不等式 $d(v,q) + d(q,o) \geqq d(v,o)$ より

$$d(v,o) - d(v,q) > r \text{ は}$$
$$d(q,o) > r$$

となり，$o$ は検索範囲に存在しないことが分かる．

$$-d(v,o) + d(v,q) > r \text{ も同様にして，}$$
$$d(q,o) > r$$

となる．従って，定理 1 は成立する．

定理 1 の $d(v,o)$ 及び $r$ はリーフノードの検索時いずれも既知であり，$d(v,q)$ は各リーフノードに対して一回計算すればよく，各リーフオブジェクトとの距離を逐一計算せずに，リーフオブジェクトが検索範囲に存在しないことが判断できる．従って，距離計算回数やリーフオブジェクトへのアクセス回数を削減できる．このリーフノードにおける解候補の絞り込みの様子を図 1 に，件数指定検索アルゴリズムを図 2 に示す．図 1 の斜線以外の部分は定理 1 の式が成立する部分であり，ここに存在するオブジェクトに対しては距離計算を省くことができる．一

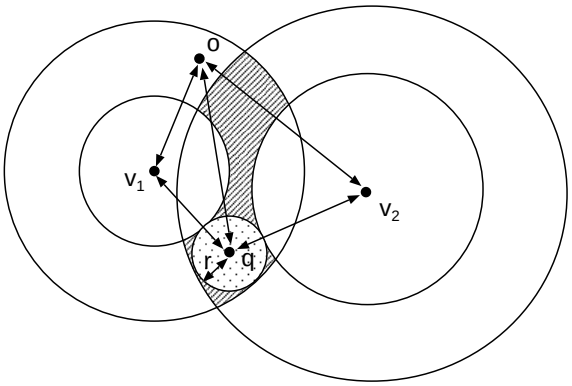


図 3　複数個の $vp$ を基準点とする絞り込み

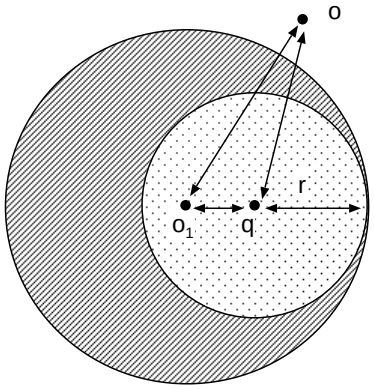


図 4　最近傍点を基準点とする絞り込み

方，斜線部分は定理 1 が成立しない部分であり，ここに存在するオブジェクトは距離計算が必要になる．

また，リーフノードの $vp$ オブジェクトだけではなく，ルートノードからリーフノードまでに至るパス上に存在するすべての $vp$ オブジェクトを用いた絞り込みも可能である．この場合，リーフノードにリンクする全リーフオブジェクトと，ルートノードからそのリーフオブジェクトまでのパス上に存在する全ての $vp$ オブジェクトまでの距離をリーフノードに予め格納しておく必要がある．この複数の $vp$ オブジェクトと各リーフオブジェクト間の距離により三角不等式を利用して距離計算回数の削減を行うことができる．問い合わせオブジェクトを $q$，検索範囲である半径を $r$，ルートノードからリーフノードまでの $k$ 個の $vp$ オブジェクトを $v_i(i = 1, 2, \ldots, k)$，リーフノードにリンクするリーフオブジェクトを $o$ とすると，図 3 のように解候補の絞り込みができる．このアルゴリズムは複数個の $vp$ オブジェクトを用いた比較を行うため解候補を絞り込める可能性が高くなる．

## 3. 最近傍点を用いた絞り込み法

前節までに説明した三角不等式の基準点に $vp$ を用いた場合，定理 1 が成立しない部分 (図 1 の斜線部分) が少ないほど絞り込み効果が向上する．この部分の外周円の半径は，$vp$ から問い合わせオブジェクト $q$ までの距離に検索範囲の半径 $r$ を加えた値となる．ここで，$r$ は検索要求に応じて固定であるため，$vp$ と $q$ との距離が小さいほど絞り込みが有利となる．一方，$q$ に最も近いオブジェクトは最近傍点である．すなわち，$vp$ の代わりに，$q$ の最近傍となるオブジェクトを三角不等式の基準点に用いると定理 1 が成立しない領域を小さくできる．よって，本稿では最近傍点を基準点とする三角不等式を用いた絞り込み法を提案する．

問い合わせオブジェクトを $q$，検索範囲である半径を $r$，検索リストの中で問い合わせオブジェクトに一番近い最近傍点を $o_1$，リーフノードにリンクするリーフオブジェクトを $o$ とすると，以下の定理が成り立つ．

**定理 2**

$d(o_1, o) - d(o_1, q) > r$ が成り立つならば，
リーフオブジェクト $o$ は検索範囲に存在しない

証明：　三角不等式 $d(o_1, q) + d(q, o) \geqq d(o_1, o)$ より

$$d(o_1, o) - d(o_1, q) > r \text{ は}$$
$$d(q, o) > r$$

となり，$o$ は検索範囲に存在しないことが分かる．

従って，定理 2 は成立する．

ここで，もし $d(o_1, o)$ と $d(o_1, q)$ が既知であれば，各リーフオブジェクトとの距離を逐一計算せずに，オブジェクトが検索範囲に存在しないことが判断できる．この様子を図 4 に示す．図のように斜線部分にリーフオブジェクトが存在しなければ問い合わせオブジェクトとの距離計算を省くことができる．実際のリーフノードの検索アルゴリズムを図 5 に示す．

定理 2 の $d(o_1, o)$ は，最近傍点とリーフノード内のオブジェクトとの距離リストがあれば値が与えられる．ただし，どのオブジェクトが $q$ の最近傍点になるかを事前に知ることはできないので，実質的な $o_1$ としては，全オブジェクトを想定しなければならない．そこで，インデキシングの際，リーフノードにリンクするリーフオブジェクトと，その他の全オブジェクトとの距離を計算したファイルを作成する必要がある．しかし，このような巨大なファイルをメモリ上で構成するのは困難となるため，本手法では赤間らの手法 [18] と同様に，各オブジェクト ID をファイル名とするファイル群として構築した．ただし，OS のディレクトリ内ファイル数の制限を受けないよう，ID を下から 3 桁ずつ区切り，最下位のものがファイル名，上位のものをディレクトリ名とした．つまり，1 ディレクトリ内の最大ファイル数を 1000 以下にした．

また，問い合わせオブジェクト $q$ と最近傍点 $o_1$ との距離となる $d(o_1, q)$ についても，最近傍点自体を事前に特定することはできない．そこで，本手法では図 5 に示すように，検索結果リスト $L$ の中で問い合わせオブジェクト $q$ に最も距離が近いオブジェクトを最近傍点 $o_1$ とみなしている．そして，新しく検索範囲に存在するオブジェクトを見つける度に，最近傍点 $o_1$ を更新するという手法を用いている．

```
Input : q , r , L
Output : L
SearchLeaf (q , r , L)
{
    foreach o (全リーフオブジェクト) {
        if ( d(o1 , q) + r < d(o1 , o) ) {
            if ( d(o , q) ≦ r ) {
                Lにoを加え, 距離の最大値をrとする ;
            }
        }
    }
}
```

q : 問い合わせオブジェクト
r : 検索の半径
o : リーフオブジェクト
$o_1$ : 検索リストの中で q に一番距離が近いオブジェクト
L : 検索結果リスト

図 5　リーフノードにおける検索アルゴリズム

## 4. 評　　価

### 4.1 実 験 方 法

本改良手法を VP-tree に実装し，それを用いて類似画像検索の実験を行った．実験マシンとして OS は Linux，CPU は PentiumD の 3.2GHz，メモリは 2G を用いている．

まず，登録画像として 10,000 件のフォト画像を用意し，画像特徴量に HSI ヒストグラムを用いて特徴量を抽出した．HSI ヒストグラムとは色相 (Hue)，彩度 (Saturation)，輝度 (Intensity) から成る色のヒストグラムである．ヒストグラムの次元数には 12($4 \times 3$) 次元及び，24($8 \times 3$) 次元，48($16 \times 3$) 次元，96($32 \times 3$) 次元の 4 種類の特徴量を用いた．次に，特徴量抽出を終えた画像オブジェクト 10,000 件に対して VP-tree でインデキシングを行った．インデキシングの際の $vp$ は毎回最大 100 件のランダムなデータを元に求めた．インデキシングに用いていない 1,000 通りの入力画像で件数指定検索を行い，検索時に要した距離計算回数及び cpu-time の画像 1 件あたりの平均を求めた．

また，画像オブジェクト間の距離尺度として quadratic form 距離を利用した．ヒストグラム $H$ とヒストグラム $K$ との間の quadratic form 距離は，

$$D_q(H,K) = \sqrt{(\mathbf{h}-\mathbf{k})^T\mathbf{A}(\mathbf{h}-\mathbf{k})} \quad (1)$$
$$= \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}(h_i-k_i)(h_j-k_j)}$$

と表される．ここで行列 $\mathbf{A} = [a_{ij}]$ はヒストグラムの $i$ 番目のビンと $j$ 番目のビンの類似度を表す行列である．本論文では以下のような行列式 [17] を用いた．

$$a_{ij} = 1 - d(i,j)/d_{max} \quad (2)$$

$d(i,j)$ は $i$ 番目と $j$ 番目のビンの色空間上での距離 (色差)，$d_{max}$ は $d(i,j)$ の最大値である．

更に，本稿では VP-tree における最近傍点を用いた絞り込み手法を提案しているが，最近傍点を用いた絞り込みを行うアルゴリズムに AESA (Approximating and Eliminating Search Algorithm) [19] が存在する．AESA は，VP-tree のようにインデックス木を構築せず，各オブジェクト間の距離を計算したファイルを事前に作成し，それを用いて絞り込みを行う．また，絞り込みにおいては提案手法と同様，最近傍点を基準点とする三角不等式での絞り込みを用いる．提案手法と AESA は非常に類似したアルゴリズムであるため，本稿では VP-tree における最近傍点を用いた絞り込み改良法の比較対象としてこの AESA を用い，絞り込み効果の優劣を評価する．そこで，AESA の具体的なアルゴリズムを次節で紹介する．

### 4.2 AESA について

AESA では，最近傍点 $s$ を以下の式によって予測する．

$$s = \underset{\forall o \in P-E}{argmin} \sum_{\forall u \in U} |\, d(o,u) - d(q,u) \,| \quad (3)$$

P は全オブジェクトの集合，E は検索対象外として除去されたオブジェクトの集合，U は過去に最近傍点 $s$ として選択されたオブジェクトの集合である．また $q$ は問い合わせオブジェクトである．この式の内容を図 6 を用いて説明する．

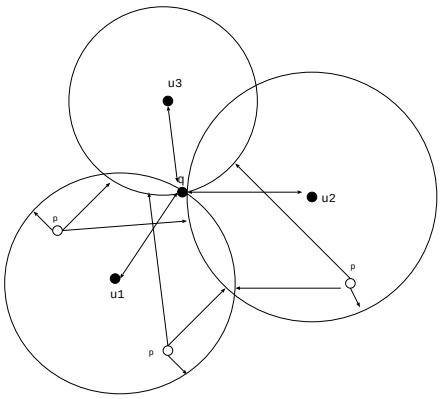


図 6　基準点 $s$ の選択方法

各 $u$ を中心とし，$d(q,u)$ を半径とする円 $C_1$〜$C_3$ を考えると，$\sum |\, d(o,u) - d(q,u) \,|$ は，各円と $o$ との距離の合計ということになる．この距離の合計が最も小さくなる $o$ が $s$ として選択される．すなわち，各 $u$ からみて，最も $q$ に近いと予想される $o$ を割り出しているということである．これにより，問い合わせオブジェクトと全オブジェクトとの距離を直接計算することなしに，その時点で $q$ に近いと思われる点を計算することができる．

続いて，選択された $s$ と問い合わせオブジェクト $q$ との間の距離を計算し，この距離が検索結果リストに入るようであれば，検索結果リストに挿入する．また，過去の最近傍点 (検索結果リスト 1 位) と $q$ との距離よりも小さければ，この $s$ を最近傍点として更新されることになる．

最後に，三角不等式による絞り込みを行う．定理 1，定理 2 と同様，以下の定理が成り立つ．

**定理 3**

$d(s,o) - d(s,q) > r$ が成り立つならば，
オブジェクト $p$ は検索範囲に存在しない

ここで，件数指定検索における検索半径 $r$ は，問い合わせオブジェクト $q$ と検索結果リスト最下位のオブジェクトとの距離となる．また，$d(s,o)$ は事前に作成された各オブジェクト間の距離リストファイルから読み出すことにより使用できる．$d(s,q)$ は直前の最近傍点更新処理において計算されている．従って，定理 3 に用いられる距離は三角不等式適用時にはいずれも既知であり，各オブジェクトと $q$ との距離を逐一計算せずに，オブジェクトが検索範囲に存在しないことが判断できる．よって，不要なオブジェクトとの距離計算を省くことができ，距離計算回数を削減できる．

以上の処理を，すべてのオブジェクトが除去されるまで繰り返すことにより，最終的に指定件数分の検索結果が得られる．

### 4.3 実験結果

個々の改良手法を用いて件数指定検索の実験を行った．グラフ中の各々の凡例は以下の手法を用いたプログラムである．特に従来手法の VP-tree に関しては，1 個の $vp$ を用いた絞り込み手法よりも複数の $vp$ を用いた手法の方が有効であることが報告されているため，本実験では後者の VP-tree を従来法として採用した．

- vp_all：複数の $vp$ を用いた絞り込み法
- vp_nn：最近傍点を用いた絞り込み法
- vp_all_nn：vp_all と vp_nn を組み合わせた絞り込み法
- AESA：AESA による絞り込み法

まず，各次元における距離計算回数の実験結果を図 7〜図 10 に示す．図 7〜図 10 はそれぞれ 12，24，48，96 次元のデータに対する実験結果に対応しており，横軸 k が検索件数，縦軸 calc_num が距離計算回数を示している．図より，VP-tree に関しては vp_all，vp_nn，vp_all_nn の順に距離計算回数が減少していることが分かる．一方 AESA は，VP-tree に比べ少ない距離計算回数で検索できていることが分かる．

次に，各次元における検索実行時間の実験結果を図 11〜図 14 に示す．図 11〜図 14 はそれぞれ 12，24，48，96 次元のデータに対する実験結果に対応している．また図 15 は 12〜96 次元のデータに対し，AESA を用いた検索実行時間の実験結果である．いずれの図も横軸 k が検索件数，縦軸が cpu-time を秒単位で示している．図 11〜図 14 より，VP-tree に関しては vp_all，vp_nn，vp_all_nn の順に cpu-time が減少している．特に，vp_all と vp_nn とを比較すると，さほど差はみられないが，vp_all_nn になると 10%程度の向上率が得られている．これは，最近傍点による絞りこみが従来の $vp$ とは異なった範囲で行われ，双方を併用することで効果的に絞り込み範囲が狭くなっていると考えられる．

次元数の変化に伴う実行時間の向上率に関しては，12 次元の 100 件検索での向上率は 5%程度であるのに対して，96 次元の 100 件検索では 12%程度の向上率が得られている．このように，本手法では次元数が増加しても充分な効果が得られることが分かる．

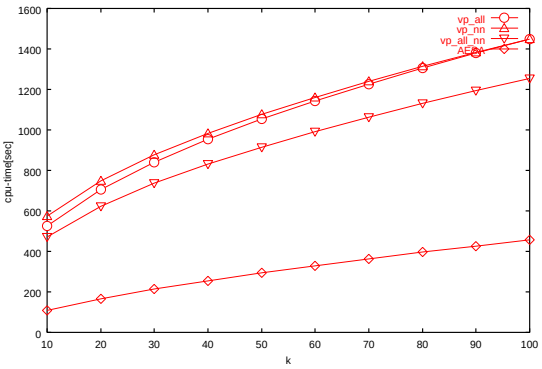


図 7　12 次元データに対する距離計算回数

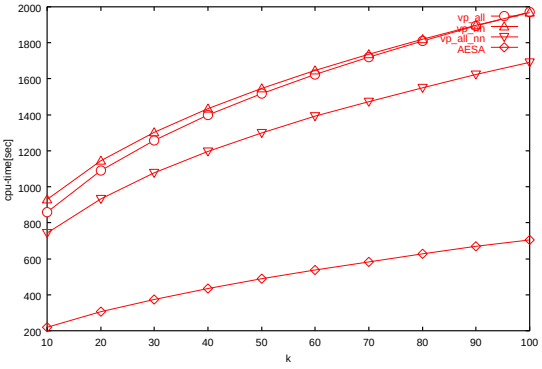


図 8　24 次元データに対する距離計算回数

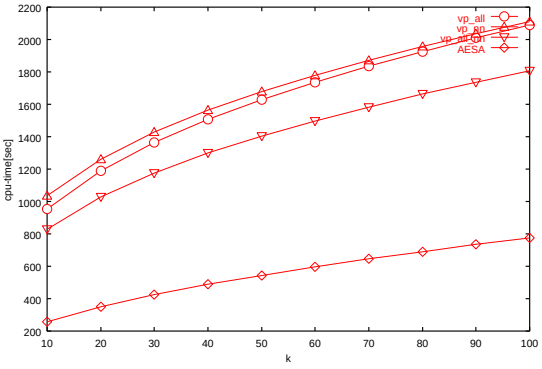


図 9　48 次元データに対する距離計算回数

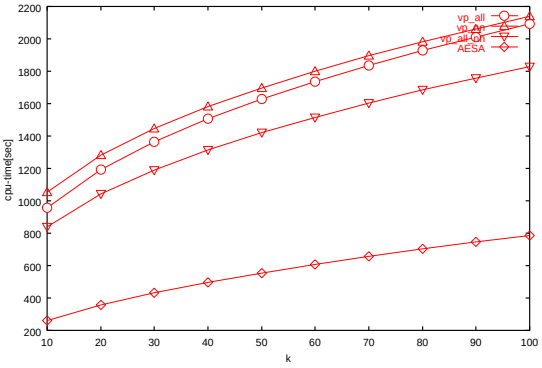


図 10　96 次元データに対する距離計算回数

構築したインデックスの詳細を表 1 に示す．dim は次元数を，node はノード数，leaf_object はリーフオブジェクト数，index_size はインデックスのデータサイズを表している．リー

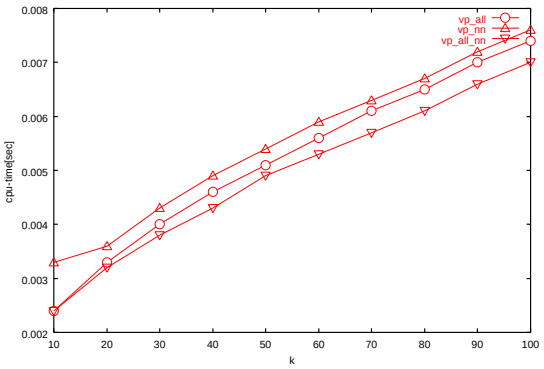


図 11　12 次元データに対する実行時間

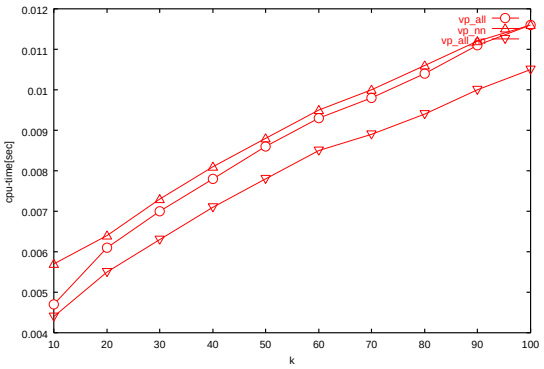


図 12　24 次元データに対する実行時間

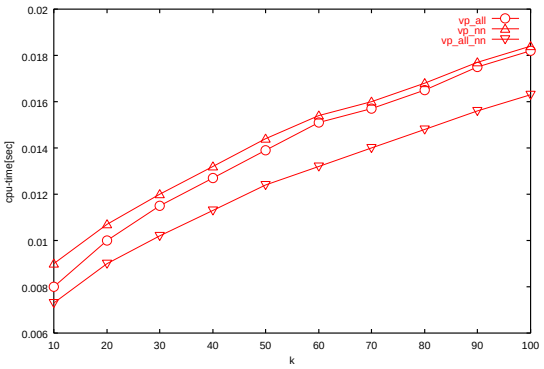


図 13　48 次元データに対する実行時間

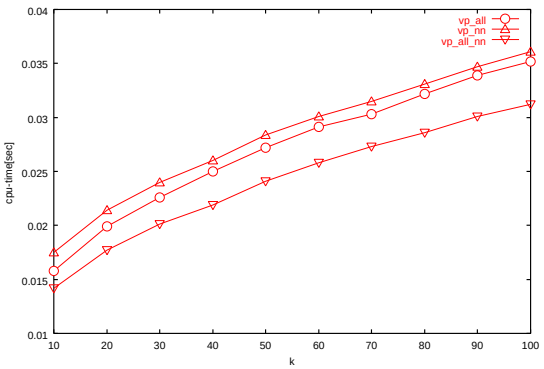


図 14　96 次元データに対する実行時間

フノードにおける最大分岐数は 10 と設定した．また，最近傍点を用いた絞り込み法に必要な距離リストを格納したファイルのサイズは，どの次元においても 313Mbyte となった．

表 1　インデックスの詳細

| dim | node | leaf_object | index_size(byte) |
|---|---|---|---|
| 12 | 2357 | 7643 | 6,000,640 |
| 24 | 2255 | 7745 | 5,742,592 |
| 48 | 2265 | 7735 | 5,767,168 |
| 96 | 2295 | 7705 | 5,844,992 |

表 2　距離リストファイル読み込み回数比較

| 次元数 | AESA | VP-tree |
|---|---|---|
| 12 | 110 | 6 |
| 24 | 219 | 7 |
| 48 | 257 | 7 |
| 96 | 262 | 7 |

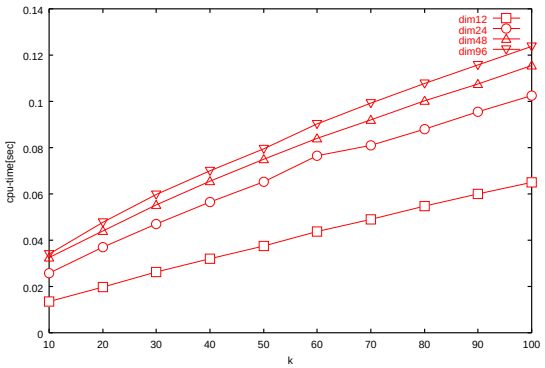


図 15　AESA を用いた実行時間

次に AESA との比較に関して，図 15 より AESA は距離計算回数において VP-tree よりも優れているにも関らず，検索時間が遅くなっている．この理由として，距離リストファイル読み込み回数の差が考えられる．VP-tree と AESA の距離リストファイルの読み込み回数を表 2 に示す．VP-tree における距離リストファイルは，前節で説明した，リーフノードにリンクするリーフオブジェクトと他の全オブジェクトとの距離を計算したファイルである．AESA における距離リストファイルは，全オブジェクト間の距離を計算したファイルである．

AESA では，処理の繰り返し毎に距離リストファイルを読み込む必要がある．つまり，距離計算回数と同じだけ距離リストファイルの読み込みを行っているということになり，これが検索時間に大きな影響を与えていると考えられる．一方 VP-tree では，リーフノードにおけるリーフオブジェクトの絞り込みにおいてのみ距離リストファイルを読み込めば良いため，ファイル読み込み回数を極少数に抑えることが可能である．ゆえに，今回の実験データにおいては AESA よりも VP-tree のほうが検索効率向上に有用であると言える．

## 5. ま と め

本稿では，VP-tree のリーフノードにおける検索アルゴリズムに改良を加え，リーフノードにおける距離計算回数を削減し，検索速度向上を試みた．そして，その改良手法を用いて類似画像検索の実験を行った．その結果，類似画像検索の検索時間を 5%〜12%削減することができた．また，VP-tree が AESA よ

りも検索時間の削減に有効であることを示した．今後の課題として，より少ないインデックスサイズでより多くの距離計算を削減できるような検索アルゴリズムを考案する．

## 謝　　辞

本研究の一部は，科学研究費補助金基盤研究 (B)(17300036)，科学研究費補助金基盤研究 (C)(17500644) を受けて行われた．

## 文　　献

[1] 北 研二, 津田 和彦, 獅々堀 正幹：「情報検索アルゴリズム」，共立出版, 2002.

[2] 吉川 正俊, 植村 俊亮：「マルチメディアデータのための索引技術」, 情報処理, Vol.42, No.10, pp.953-957, 2001.

[3] 片山 紀生, 佐藤 真一：「類似検索のための索引技術」, 情報処理, Vol.42, No.10, pp.958-963, 2001.

[4] Guttman, A.： ”A Dynamic Index structure for Spatial Searching”, Proc.ACM SIGMOD ’84, pp.47-57, 1984.

[5] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B.： ”The R*-tree: An efficient and robust access method for points and rectangles”, Proc. ACM SIGMOD, pp.322-331, 1990.

[6] White, D.A. and Jain, R.： ”Similarity Indexing with SS-tree”, Proc. 12th Int. Conf. on Data engineering, pp.516-523, 1996.

[7] 片山 紀生, 佐藤 真一：「SR-tree:高次元点データに対する最近接検索のためのインデックス構造の提案」, 電子情報通信学会論文誌, Vol.J80-D-I, No.8, pp.703-717, 1997.

[8] Berchtold, S., Keim, D.A., and Kriegel, H.-P.： ”The X-tree An Index Structure for High-Dimensional Data”, Proc. 22nd VLDB, pp.28-39, 1996.

[9] Weber, R., Schek, H.-J., and Blott, S.： ”A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces”, Proc. 24th VLDB, pp.194-205, 1998.

[10] Ioka, M.：”A Method of Defining the Similarity of Images on the Basis of Color Information”, Technical Report RT-0030, IBM Tokyo Research Lab, 1989.

[11] Rubner,Y., Tomasi, C., and Guibas, L.J.： ”The Earch Mover’s Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval”, In Proc. of the ARPA Image Understanding Workshop, pp.661-668, May 1999.

[12] Ciaccia, P., Patella, M., and Zezula, P.： ”M-tree: An Efficient Access Method for Similarity Search in Metric Spaces”, Proc.ACM SIGMOD Int. Conf. on the Management of Data, pp.71-79, 1995.

[13] Yianilos, P.N.：”Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces”, ACM-SIAM SODA’93, pp.311-321, 1993.

[14] Fu, A.W.-C., Chan, P.M.S., Cheung, Y.-L., and Moon, Y.S.： ”Dynamic VP-Tree Indexing for N-Nearest Neighbor Search Given Pair-Wise Distances”, VLDB Journal, pp.2-8, 2000.

[15] Bozkaya, T. and Ozsoyoglu, M.： ”Distance-based Indexing for High-dimensional Metric Spaces”, ACM SIGMOD, pp.357-368, Tucson, AZ, 1997.

[16] 石川 雅弘, 能登谷 淳一, 陳 漢雄, 大保 信夫:「距離索引 $MI\text{-}tree$」, 情報処理学会論文誌, Vol.40, No.SIG6(TOD3), pp.104-114, 1999.

[17] 岩崎 雅二郎：「類似画像検索を実現する距離空間インデックスの実装及び評価」, 情報処理学会論文誌, Vol.40, No.SIG3(TOD1), pp.24-33, 1999.

[18] 赤間 浩樹, 小西 史和, 吉田 忠城, 谷口 展郎, 山室 雅司, 串間 和彦：「近傍検索向け転置ファイル法における外部キー検索と動的データ追加の実装と評価, 情報処理学会論文誌, Vol.40, No.SIG8(TOD4), pp.51-62, 1999.

[19] VIDAL RUIZ：An algorithm for finding nearest neighbours in (approximately) constant average time, Pattern Recognition Letters pp.145-157 1986.