社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2003－FI－ 72　(1)
2003－NL－157　(1)
2003／9／29

距離索引VP-treeにおける解絞り込みの一改良法

中川 嘉之[1]　獅々堀 正幹[1]　北 研二[2]

[1] 徳島大学工学部
[2] 徳島大学高度情報化基盤センター

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，三角不等式に用いる基準点を複数個用いる手法，また，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手法を提案する．これらの改良法により，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58%〜72%削減することができた．

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

Yoshiyuki Nakagawa[1]　Masami Shishibori[1]　Kenji Kita[2]

[1] Faculty of Engineering, Tokushima University
[2] Center for Advanced Information Technology, Tokushima University

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 the method to utilize not only one vantage point but also two or more vantage points as the datum point of the triangular inequality, and the method to use the nearest neighbor object point for the query as the datum point. 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 58%〜72% of search time of the traditional method.

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．

そこで本稿では，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．本手法は，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，三角不等式に用いる基準点を複数個用いること，またはその基準点に，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した．

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 2 つ紹介する．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 μ 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．

$$\begin{aligned} S_1 &= \{s \in S \mid d(s,vp) < \mu\} \\ S_2 &= \{s \in S \mid d(s,vp) \geq \mu\} \end{aligned}$$

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．

- 範囲検索：

  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．

- 件数指定検索：

  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 節で述べたように，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

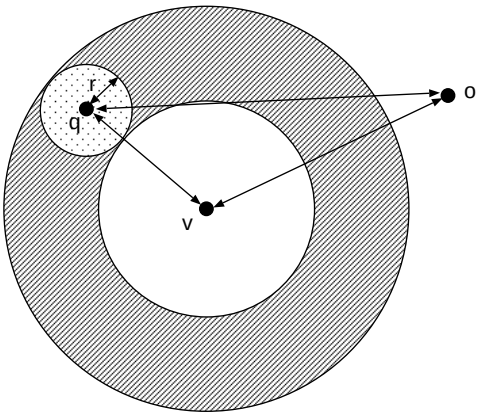


図 1: vp を基準点とする絞り込み

```
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

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．

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 $d(v,q) + d(q,o) \geq d(v,o)$ 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．

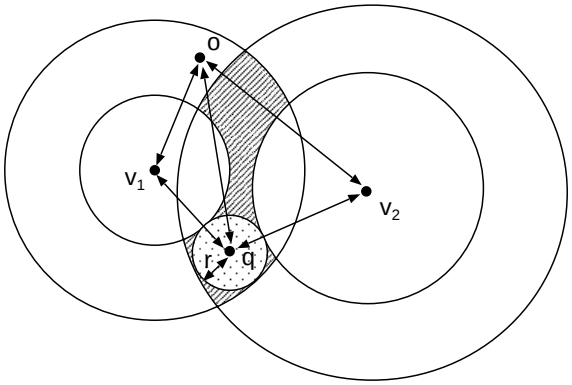


図 3: 複数個の vp を基準点とする絞り込み

$-d(v,o)+d(v,q) > r$ 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．

3 VP-tree の改良

3.1 複数個の vp を用いた絞り込み法

2.3 節ではリーフノードの vp オブジェクトのみを三角不等式に用いていた．本節では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 オブジェクトと，リーフオブジェクトとの距離リストを用いた絞り込み法を提案する．

まずインデキシングの際，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．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 2.3 節のアルゴリズムに比べ，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．

3.2 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 $d(o_1,q) + d(q,o) \geq d(o_1,o)$ 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

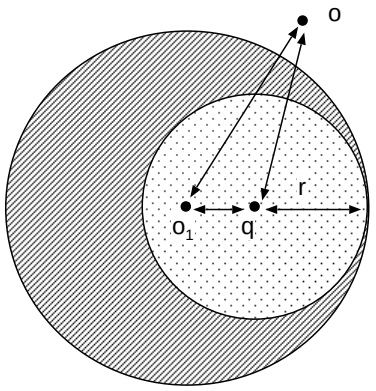


図 4: 最近傍点を基準点とする絞り込み

```
Input : q , r , L
Output : L
SearchLeaf (q , r , L)
{
    foreach o (全リーフオブジェクト) {
        if ( d(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

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無理なので，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，本稿では赤間らの手法 [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．

4 評価

4.1 実験方法

前章で述べた 2 つの改良手法を 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 Pentium-4 の 2.2GHz，メモリは 256MB 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 H のみを用いた 16 次元及び，HSI 全てを用いた 48(16×3) 次元，192(64×3) 次元，384(128×3) 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，

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

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．

また，構築したインデックスの詳細を以下に示す．表 1 は改良前を，表 2 は改良後の詳細を表している．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．リーフノードにおける最大分岐数は 100 と設定した．

表 1: インデックスの詳細 (改良前)

dim	node	leaf_object	index_size(byte)
16	263	9,737	446,052
48	285	9,715	473,188
192	279	9,721	493,540
384	291	9,709	483,364

表 2: インデックスの詳細 (改良後)

dim	node	leaf_object	index_size(byte)
16	263	9,737	13,176,304
48	285	9,715	14,278,504
192	279	9,721	13,979,904
384	291	9,709	14,579,104

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 391Mbyte となった．一方，インデックスの作成と距離リストファイルの作成に要した時間は以下のようになった．表 3 は改良前を，表 4 は改良後の作成時間を表している．index はインデックスの作成に要した時間を，dist_file は距離リストファイルの作成に要した時間を示している．

表 3: インデックスとファイル作成時間 (改良前)

dim	index(sec)	dist_file(sec)
16	11	0
48	21	0
192	113	0
384	360	0

表 4: インデックスとファイル作成時間 (改良後)

dim	index(sec)	dist_file(sec)
16	15	492
48	29	876
192	155	4,713
384	480	14,962

4.2 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．

- base：1 個の vp を用いた絞り込み法 (既存の VP-tree)
- vpall：複数の vp を用いた絞り込み法
- nn：最近傍点を用いた絞り込み法
- vpall_nn：vpall と nn を組み合わせた絞り込み法

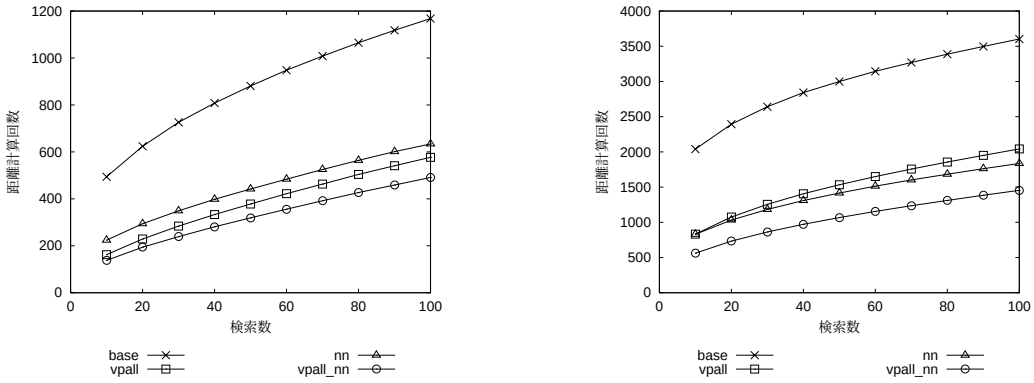
各次元における距離計算回数の実験結果を図 6 に示す．graph1〜graph4 はそれぞれ 16，48，192，384 次元のデータに対する実験結果に対応している．図より 16 次元以外は base，vpall，nn，vpall_nn の順に距離計算回数が減少していることが分かる．特に最も結果の良かった vpall_nn は，base に比べ 58%〜72%の距離計算を削減することができている．

各次元における実行時間の実験結果を図 7 に示す．graph5〜graph8 はそれぞれ 16，48，192，384 次元のデータに対する実験結果に対応している．図より距離計算回数の実験結果と同様に 16 次元以外は base，vpall，nn，vpall_nn の順に cpu-time が減少していることが分かる．特に vpall_nn は，base に比べ 38%〜72%の cpu-time を削減することができている．膨大な距離リストを格納したファイルが必要となる nn は，ファイルを ID ごとに分けることによって，検索時間にそれほど影響を与えずに絞り込みができている．また，高次元になるほど検索速度向上の割合が高くなっている．これはたとえ距離計算回数が同じでも，高次元になるほど距離計算コストが高くなっているためである．

5 まとめ

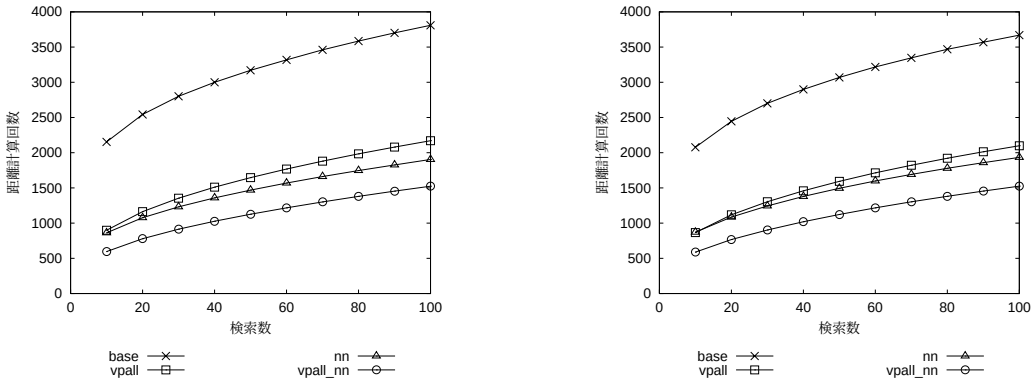
本稿では，VP-tree のリーフノードにおける検索アルゴリズムに改良を加え，リーフノードにおける距離計算回数を削減し，検索速度向上を試みた．そして，その改良手法を用いて類似画像検索の実験を行った．その結果，類似画像検索の検索時間を 58%〜72%削減することができた．

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．



graph1 : 16次元データに対する実験結果

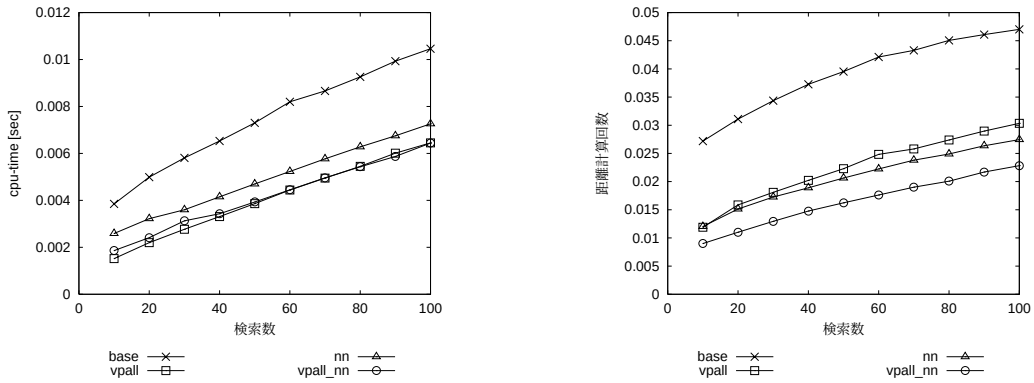
graph2 : 48次元データに対する実験結果



graph3 : 192次元データに対する実験結果

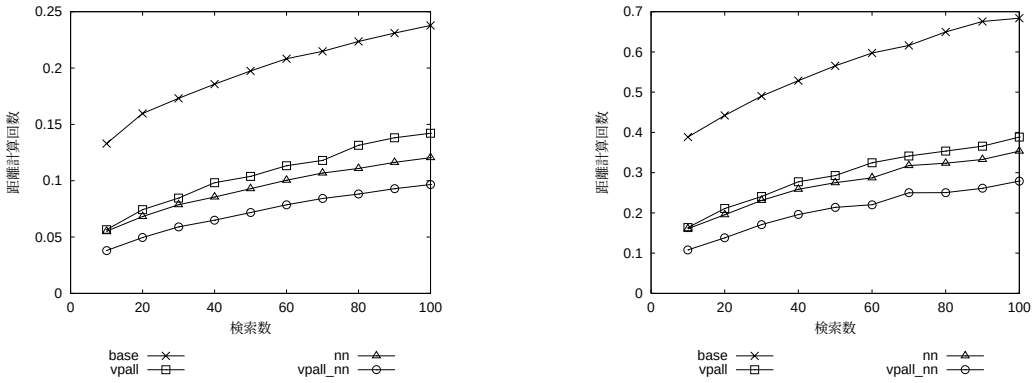
graph4 : 384次元データに対する実験結果

図 6: 距離計算回数の比較



graph5 : 16次元データに対する実験結果

graph6 : 48次元データに対する実験結果



graph7 : 192次元データに対する実験結果

graph8 : 384次元データに対する実験結果

図 7: 実行時間の比較

参考文献

[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.