マルチメディア情報の大規模処理に向けた多次元インデクシング手法の応用

片山 紀生　　佐藤 真一
学術情報センター 研究開発部

概要:　マルチメディアアプリケーションの大規模化を実現するためには, 大量のマルチメディアデータに対する処理コストの低減が不可欠であり, データベースシステム分野の成果の導入が有効である. 本論文では, 多次元インデックス手法の有効性に着目し, インデックスの静的構築法ならびに多次元ベクトル集合を単位とした最近接検索法について報告する. まず, 最近接検索の高速化に適しているとされる Sphere/Rectangle-tree (SR-tree) の静的構築法を提案し, その妥当性を評価するための実験として, 類似画像検索に適用した例を報告する. 次に, ベクトル集合を単位とした最近接検索を多次元インデックス構造を用いて高速実装する方法について報告する. 具体例として, 動画中から抽出された顔画像を照合するシステムを取り上げ, ベクトル集合を用いることによって精度が向上すること, ならびに, 多次元インデックス構造によって高速かつスケーラブルな実装が可能であることを示す.

Application of Multidimensional Indexing Methods toward Massive Processing of Multimedia Information

Norio Katayama and Shin'ichi Satoh
Research and Development Department
NACSIS (National Center for Science Information Systems)

Abstract:　This paper applies multidimensional indexing methods to multimedia applications. Multimedia information is so large that the reduction of processing cost is essential to the scalable implementation of multimedia applications. This paper applies a multidimensional index structure, namely a Sphere/Rectangle-tree (SR-tree), for the cost reduction of the similarity search. In the first place, the static construction method of the SR-tree is presented. Then its effectiveness is demonstrated with the SR-tree being applied to the similarity retrieval of still images. Moreover, the SR-tree is adapted to colored nearest neighbor search, which improves the precision and the recall of the similarity retrieval of video scenes. The effectiveness and the scalability of the SR-tree implementation are verified through performance tests.

1　はじめに

マルチメディアアプリケーションを実現するための情報処理技術について広く関心が集まっており, 例えば, 画像や音声に対する検索システムを実現するために特徴量の抽出法や評価法について盛んに研究が進められている. マルチメディア情報は一般に情報量が大きく, しかも, コストの大きい処理を必要とする. そのため, マルチメディアアプリケーションの大規模化を実現するには, マルチメディアデータに対する処理コストを低減することが不可欠であると考えられる. 大量のデータを効率よく処理するためには, データベースシステム分野の研究成果の導入が有効である. 本論文では, 特にマルチメディアアプリケーションにおける多次元インデックス手法の有効性に着目し, インデックスの静的構築法ならびに多次元ベクトル集合を単位とした最近接検索法について報告する.

まず, 静的構築法とは与えられたデータの集合に対して最適なインデックスを構築する手法のことであり, データが逐次追加される動的構築法と対になる構築法である. 本論文では, 最近接検索の高速化に適しているとされる Sphere/Rectangle-tree (SR-tree)[1, 2] という多次元インデックス構造の静的構築法を提案する. SR-tree はこれまで動的に構築されるインデックス構造として提案されており, 静的構築法については検討

E-mail: {katayama,satoh}@rd.nacsis.ac.jp

が行われていなかった. また, 動的構築法についても, 直観的な考察に基づいて設計されており, その理論的根拠が明らかではなかった. そこで, 本論文では, まず, 動的構築法の理論的根拠を明らかにし, 次に, その結果を応用することで静的構築法を実現できることを示す. そして, その妥当性を評価するための実験として, 類似画像検索に適用した例について報告する.

次に, 本論文では, ベクトル集合を単位とした最近接検索を, 多次元インデックス構造を用いて高速実装する方法について報告する. ベクトル集合を単位とする最近接検索は, 動画像のシーンを類似検索する場合など, ひとつのマルチメディア情報から複数の特徴ベクトルが得られる場合に有効な方法である. しかし, この方法は, ベクトル集合とベクトル集合の比較演算が必要であり, 単一ベクトルを単位とする最近接検索よりも遥かに演算コストが大きい. 本論文では, この問題を解決する方法として, 多次元インデックス構造を応用する方法を提案する. そして, 具体的な応用例として, 動画中から抽出された顔画像を照合するシステムを取り上げ, 提案手法により精度が向上すること, ならびに, 多次元インデックス構造によって高速かつスケーラブルな実装が可能であることを示す.

本論文は以下の構成になっている. まず, 2 節では, 多次元インデックスの基本構造と探索アルゴリズムについて概要を説明する. 3 節では, SR-tree の静的構築法について説明する. 4 節では, ベクトル集合を単位とする最近接検索の高速実装法について説明する. 最後に 5 節で結論を述べる.

2 多次元インデクシング手法の概要

2.1 マルチメディアアプリケーションにおける多次元インデクシング手法の有効性

画像, 音声, 映像といったマルチメディアデータに対し, これらの意味内容にまで立ち入った高次処理の実現が望まれている. しかしながら, その実現のためには, 必然的にこれらマルチメディアデータの認識・理解が必要となってきてしまうが, 残念ながら現在の技術では満足の行く結果は得がたい. そこで現在のマルチメディアアプリケーションでは, もとのマルチメディアデータに対してある種の処理を施して特徴量を取り出し, これをもとに目的のアプリケーションを実現している例が多く見られる. これは, 互いに意味内容的に似たマルチメディアデータからは, 互いに近い特徴量が得られるという仮定によっている. こうしたアプローチは, コンテントベースド×× と呼ばれ, 例えばコンテントベースド画像検索としては QBIC [3] が良く知られている. その現実的な有効性は既に多くのシステムなどで実証されているが, マルチメディアデータが本質的に多義性を持つこと, 意味内容に完全に対応した特徴量の決定が不可能であること等から, 一般にきわめて高次元の特徴量を扱わなければならない (例えば, QBIC では画像の色合いの表現に 64 ないしは 256 次元を利用している). また必ずしも特徴量はマルチメディアデータの意味内容を完全に表してはいないため, ある特徴量に「似た」データを拾ってくるという, 近接探索が本質的な処理となっている. そのため, データベースシステム分野の研究で最近特に研究のさかんな多次元インデクシング手法, 及びそれを用いた探索手法の利用は, 大規模なデータを実用的な時間で扱うことの求められる今後のマルチメディアアプリケーションではきわめて有効である.

2.2 多次元インデックスの基本構造

データベースシステム分野では, これまでに多次元インデクシング手法として様々なインデックス構造が提案されている. R-tree [4] ならびに R-tree を改良した R*-tree [5] は元来, 地理データや CAD データを応用とする空間データのためのインデックス構造として提案されたが, 画像検索など特徴空間のインデックス構造としても使われている [3]. また, 近年, 最近接検索を特に高速化するインデックスとして, SS-tree[6] や SR-tree [1, 2] が提案されている.

これらのインデックスは特徴空間を階層的な部分領域に分割することで, 探索範囲を限定し高速化する (図 1). 階層的なインデックスとしては, R-tree 以外に k-d tree や quadtree が挙げられるが, 部分領域の構成方法に大きな違いがある. まず, k-d tree と quadtree は部分領域を重なりのない独立な領域に分割する. そして, k-d tree はデータの分布に応じて適応的に領域を分割し, quadtree は予め定められた比率で規則的に分割する. これに対して R-tree は部分領域を包囲長方形 (bounding rectangle) によって管理し, それらの間に重なりがあることを許容する. これらの中で特に動的な特性に優れているのが R-tree であり, 重なりを許容することで動的な環境においても木の平衡を維持することが可能になっている. そのため, データベースシステムでの利用に適していると考えられている.

このような階層的なインデックスを構築する際には, 領域の分割法が性能を大きく左右する. SS-tree と SR-tree は, R-tree から派生したインデックス構造であり R-tree と共通した構造を持っているが, 木の構築アルゴ

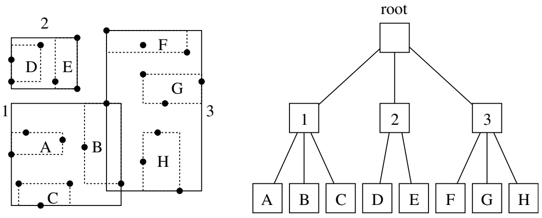


図 1: R-tree の階層構造

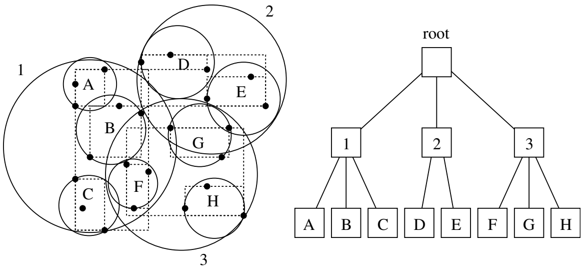


図 2: SR-tree の階層構造

リズムを変更することによって最近接検索の高速化を実現している. まず, 木の構築アルゴリズムは SS-tree, SR-tree とも同じで, もともと SS-tree のために考案されたものを用いている. これは, ベクトルの座標値の分散に基づいて領域を分割する方法で, 評価実験の結果, R*-tree よりも径の小さい領域を生成できることが明らかになっている [1, 2].

次に, 領域の形状は, SS-tree が包囲球 (bounding sphere) を用いているのに対して, SR-tree が包囲球と包囲長方形との交わりを用いている (図 2). SR-tree が包囲長方形を導入しているのは, 領域をより小さくするためである. 包囲長方形と包囲球は, 高次元空間において相補的な関係にあり, 包囲長方形は領域の体積を, 包囲球は領域の径を小さくできるという利点がある. そこで, SR-tree ではこれらの形状を組み合わせることによって領域の体積と径を同時に小さくすることを可能にし, SS-tree を上回る性能を実現している. そこで, 本論文では多次元インデックス構造として, SR-tree を用いている.

2.3 多次元インデックスの探索アルゴリズム

最近接検索は, 与えられた質問点に最も近い点を探す検索であり, 画像などから得られた特徴ベクトルを類似検索する場合など, マルチメディアアプリケーションにおいて広く使われている. 多次元インデックスを利用しながら最近接検索を行う方法には二種類あり, 深さ優先の探索法 [7] と幅優先の探索法 [8] が提案されている. 基本的な考え方はどちらも同じであり, 質問点に近い領域から順に検索結果の候補を集めながら探索し, 集めた候補よりも近い未探索領域が無くなった時点で探索を終える. そして, 探索を終えた時点での候補集合が検索結果になる.

このような探索を行う場合, 中間ノードや葉ノードを質問点から近い順に整列した優先順位付き待ち行列 (priority queue) が必要になる. 根ノードから出発し, 中間ノードを訪れる度に子ノードを待ち行列に加え, 待ち行列の先頭から順に探索していくのである. 深さ優先の探索法と幅優先の探索法の相違点は, この待ち行列の作り方であり, その結果, 木の辿り方が深さ優先と幅優先になっている.

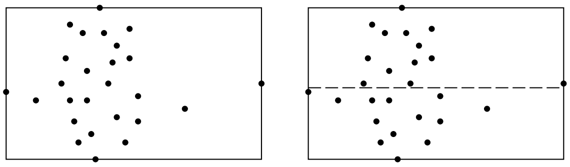
まず, 深さ優先の探索の場合, 訪れた中間ノードごとに待ち行列を作り, 待ち行列の先頭から順に再帰的に探索を行う. つまり, 中間ノードごとに独立した待ち行列が作られ, 木の構造に沿って再帰的に探索が行われるのである. この場合, 待ち行列で二番目のノードが探索されるのは, 一番目のノードの探索が終了した後になるので, 深さ優先の探索になる. これに対して, 幅優先の探索では, 木全体でひとつの待ち行列を用いる. 中間ノードを訪れる度に子ノードを同一の待ち行列に加えるのである. そのため, 深さ優先の探索のように再帰的な探索にはならず, 待ち行列に加えた全ての子ノードの中で最も近いものが順に探索されることになる.

これらの探索法は一長一短であり, 必ずしも一方が優れているとは言えない. 訪れるノードの数については, 幅優先の探索法が最適であり, 必ず最少限のノードしか訪れないことが明らかになっている [9]. ところが, 幅優先の探索法には待ち行列が長くなるという問題がある. 待ち行列は, 質問点からの距離を基準とする優先順位付き待ち行列なので, 待ち行列が長くなるにつれてノードを追加する際の演算コストが大きくなってしまう. 一方, 深さ優先の探索法では, 中間ノードごとに待ち行列を作るので, 待ち行列の長さは高々, 中間ノードあたりの子ノードの数にしかならない. そのため, 演算コストの面では, むしろ幅優先の探索の方が有利なのである. そこで, 本論文では深さ優先の探索法を用いている.

3 SR-Tree の静的構築法

3.1 動的構築法の理論的根拠の解析

SR-tree は, SS-tree[6] のアルゴリズムを用いており, 以下の方法で動的に木構造を構築する. しかし, この



領域内に分布する点の例　　分散が大きい軸による分割

図 3: 領域分割の例

アルゴリズムは, 実験的にその有効性が示されているものの, その理論的根拠が明らかではなかった. ここでは, このアルゴリズムが, 領域内の二点間の二乗距離を最小化するアルゴリズムとして説明できることを示す.

(i) 葉ノードの選択

新たに点を追加するとき, まず, 投入すべき葉ノードを決定する. 根ノードから出発し, 点から部分木の重心までの距離が最短のノードを選びながら, 葉ノードまで達する. そして, そこに空きがある場合には点を追加し, ない場合には (ii), (iii) に従って点の再投入ならびにノードの分割を行う.

(ii) 点の再投入

重心からの距離が遠い一定割合の点を根から再度投入する. すでに再投入が行われたノードが再度あふれた場合には, (iii) に従ってノードを分割する.

(iii) ノードの分割

分割処理 (図 3) は, 分割軸の決定と分割位置の決定の二段階に分かれる. まず, 各空間軸について座標値の分散を求め, 最大の分散を持つ軸を分割軸とする. そして, 選んだ軸と直交する平面を分割面とし, 分割後の二つの領域の分散の和が最小となる位置を分割位置とする. ただし, 分割位置の選択は, 二つの領域の点の数が一方に偏り過ぎない範囲でのみ行う. もし, 葉ノードの分割によって上位のノードがあふれる場合には, 葉ノードと同じ方法で再投入ならびに分割を行う.

解析の第一歩として, まず, 領域内の二点間の二乗距離と座標値の分散とが直接関係付けられることを示す. 今, N 個の点 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ が与えられているとき, それらの間の二乗距離の平均は次式で表される.

$$\begin{aligned} &\operatorname*{avg}_{1\le i\le N, 1\le j\le N} |\boldsymbol{x}_i - \boldsymbol{x}_j|^2 \\ &\qquad = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} |\boldsymbol{x}_i - \boldsymbol{x}_j|^2 \\ &\qquad = \frac{2N}{(N-1)} V, \end{aligned} \tag{1}$$

ここで, V は点の座標値の分散であり, 次式のように定義される.

$$V \equiv \frac{1}{N} \sum_{i=1}^{N} |\boldsymbol{x}_i - \bar{\boldsymbol{x}}|^2, \tag{2}$$

$\bar{\boldsymbol{x}}$ は N 個の点の重心 $(\sum_{i=1}^{N} \boldsymbol{x}_i / N)$ である.

式 1 から分かる通り, 座標値の分散が小さい領域ほど, 二点間の二乗距離の平均も小さくなる. 最近接検索を高速化するには, 互いの距離が近い点を同一の領域に含む方がよい. したがって, 座標値の分散の小さい領域を生成することで最近接検索を高速化できることになる.

このことから, アルゴリズムの (i) と (ii) を説明することができる. 座標値の分散は, その定義より, 重心から各点までの二乗距離の平均と等しい (式 2). したがって, アルゴリズム (i) で重心からの距離が最小のノードを選択している処理, ならびに, アルゴリズム (ii) で重心からの距離が遠い点を再投入している処理は, 領域内の二点間の二乗距離を小さくする処理に他ならないのである.

次に, アルゴリズム (iii) の処理について考えてみる. 後半の分割位置を決定する処理は, 分散を小さくする処理そのものなので, 前半の分割軸の決定方法について検討する. 今, N 個の点 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ が与えられており, これらを二つの領域 L と R に分割することを考える. L と R に属する点の数は, それぞれ $N_\mathrm{L}, N_\mathrm{R}$ とする. このとき, 分割後の領域の分散をそれぞれ V_L, V_R とすると, N 個の点全体の分散 V との間に次の関係が成り立つ.

$$\begin{aligned} &N_\mathrm{L} V_\mathrm{L} + N_\mathrm{R} V_\mathrm{R} \\ &\qquad = NV - \frac{N_\mathrm{L} N_\mathrm{R}}{N} |\bar{\boldsymbol{x}}_\mathrm{L} - \bar{\boldsymbol{x}}_\mathrm{R}|^2 \\ &\qquad = NV - \frac{N_\mathrm{L} N_\mathrm{R}}{N} \sum_{k=1}^{d} \left(\bar{\boldsymbol{x}}_{\mathrm{L}\langle k\rangle} - \bar{\boldsymbol{x}}_{\mathrm{R}\langle k\rangle} \right)^2, \end{aligned} \tag{3}$$

ここで, $\bar{\boldsymbol{x}}_\mathrm{L}$, $\bar{\boldsymbol{x}}_\mathrm{R}$ はそれぞれ領域 L と R に属する点の重心であり, $\bar{\boldsymbol{x}}_{\langle k\rangle}$ は, 点 $\bar{\boldsymbol{x}}$ の k 番目の次元の座標値を表す. d は空間の次元数である. この式から, 分割後の分散 $V_\mathrm{L}, V_\mathrm{R}$ を小さくするためには, 右辺の第二項を小さくすればよいことがわかる.

そこで, 分割軸の選択法と式 3 の第二項との関係について考える. 点の分布が次元によって独立であると仮定すると, 分割軸以外の軸について次の近似が成り立つ.

$$\bar{\boldsymbol{x}}_{\mathrm{L}\langle k\rangle} - \bar{\boldsymbol{x}}_{\mathrm{R}\langle k\rangle} \approx 0 \qquad \text{for } 1 \le k \le d,\ k \ne s, \tag{4}$$

ここで s は分割軸を表す番号である. そして, 分割軸については, 点の分布が一様分布であるという仮定を

行うことで次の近似が成り立つ．

$$|\bar{\boldsymbol{x}}_{\mathrm{L}\langle s\rangle} - \bar{\boldsymbol{x}}_{\mathrm{R}\langle s\rangle}|^2 \approx 3V_{\langle s\rangle}. \qquad (5)$$

以上, 式 3, 4, 5 から, 分割後の分散 $V_{\mathrm{L}}, V_{\mathrm{R}}$ と分割前の分散 V との関係が次式のように近似できることがわかる．

$$N_{\mathrm{L}}V_{\mathrm{L}} + N_{\mathrm{R}}V_{\mathrm{R}} \approx NV - \frac{N_{\mathrm{L}}N_{\mathrm{R}}}{N}3V_{\langle s\rangle}. \qquad (6)$$

この式から, 分割後の領域の分散 V_{L} と V_{R} を小さくするには, 分割軸の分散 $V_{\langle s\rangle}$ を大きくすればよいことがわかる．すなわち, 分散が最大の軸を分割軸にすればよいのである．

以上の考察から, SS-tree ならびに SR-tree の構築アルゴリズムは, 領域内の二点間の二乗距離を最小化するアルゴリズムと説明できることがわかる．

3.2 領域分割による SR-Tree の静的構築

動的構築法の理論的解析によって, SR-tree が領域内の分散を最小化するという規範のもとに構築されていることが明らかになった．この事実は, SR-tree の静的構築法が, 動的構築法で使われている領域分割を応用することによって容易に実現できることを示している．すなわち, 領域分割を再帰的に実行することによって, 階層的な木構造を構築できるのである．

この方法は, すでに R-tree の静的構築法として提案されており, VAMSplit R-tree [10] と呼ばれている．VAMSplit R-tree では, 分割位置を適切に選ぶことによって, 生成されるノードの数を最小化できることも同時に示されている．VAMSplit R-tree のこのような構築法は, そのまま SR-tree に応用することが可能であり, 本論文では, この方法で SR-tree の静的構築を実現する．

一般に, 多次元インデックス構造の性能は, 動的に構築されたものよりも静的に構築されたものの方がよい．これは, 動的に構築される場合, データについての情報が逐次的に与えられるのに対して, 静的に構築する場合, 全てのデータに関する情報を用いて木構造を構築できるからである．また, 使用するノードの数を減らすことが可能なので, 探索時に訪問するノードの数が減り, 演算コスト, 入出力コストが小さくなるという利点もある．

3.3 類似画像検索による性能評価

静的構築法の有効性を検証するために, 画像の類似検索を例として評価実験を行った．検索手法は広く使われているカラーヒストグラムに基づく方法であり, 画像データは NASA (アメリカ航空宇宙局) で公開されているスペースシャトルの静止画像と動画像を使用した．動画像についてはカラーヒストグラムによってシーンチェンジを検出し, 各シーンごとに代表的なフレームを抜き出して使用した．

SR−Tree
CPU 時間 0.244 秒
ディスク I/O 232 キロバイト

線形探索(総当たり探索)
CPU 時間 2.044 秒
ディスク I/O 11,737 キロバイト

図 4: 類似画像検索の結果の例

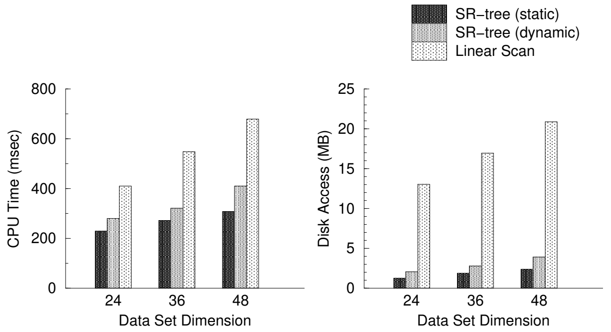


図 5: 静的構築による類似画像検索の性能向上

色空間としてはマンセルの表色系 (Hue, Saturation, Intensity) を使い, ６～１２個の部分空間に分割してヒストグラムを求めた．部分空間としては, 黒, 灰, 白の３色と, 色相 (Hue) で分割した３～９色を用いた．また, 構図についての類似性を加味するために, ヒストグラムは４つの部分領域に分けて計算した．部分領域は, 右上, 左上, 右下, 左下の４つで, それぞれ全体の４分の１の大きさを持つ．４つのヒストグラムを連結して２４～４８次元のベクトルを作り, 特徴ベクトルとして使用した．特徴ベクトル間の類似度はユークリッド距離によって計算した．

検索結果の例を図 4に, 実験結果を図 5 に示す．図 5 が示すとおり, SR-tree を用いることによって, CPU 時間, ディスクアクセスとも削減することに成功している．特に, ディスクアクセスの減少が著しく, これは

SR-tree の領域分割により, 探索範囲が小さくなったことを意味する. また, 静的構築法と動的構築法を比較すると, 静的に構築された SR-tree のコストは動的に構築された SR-tree のおよそ 70 ~ 80%であり, 静的構築法を用いることによって, さらに処理コストが小さくなっている.

4　ベクトル集合を単位とする最近接検索の高速実装

4.1　ベクトル集合を単位とする最近接検索

静止画の類似検索では, ひとつの画像からひとつの特徴ベクトルを求め, ベクトル単位の最近接検索によって画像を照合する. ところが, ひとつのマルチメディア情報から複数の特徴ベクトルが得られる場合には, それらを集合として処理し, 集合単位に最も距離の近いものを見つけることで精度を高めることが可能になる.

例えば, 図 6 では, ひとつの画像シーケンスから複数の特徴ベクトルが得られ, それらが集合を構成している. これらの特徴ベクトルを集合単位で照合する場合, 互いに最も近いベクトルの組 (closest pair) を見つけることによって, ベクトル集合間の距離を測ることが可能になる. 例えば, 映像のシーンごとに特徴ベクトルが生成されているとすると, 最も近いベクトルの組を見つけることは, 最も類似したシーンを共有する映像の組を見つけることになる. このようなベクトル集合を単位とした最近接検索を行う例としては, 後述の動画像から切り出した顔シーケンスの照合や Nayar らによる appearance matching [11] が挙げられる.

ベクトル集合を単位とする最近接検索は, 単一ベクトルを単位とする最近接検索を拡張することで実現できる. 基本的な考え方は, 単一ベクトルを単位としたアルゴリズムと同じで, 質問点の集合からの最小距離でノードのエントリをソートし, 最小距離が短いものから順に探索する. ここで, ノードの領域や点までの距離は, 個々の質問点からの距離の中で最小のものによって決まる. 探索は二段階に分かれており, まず, 指定された数に達するまで点を集め, 検索結果の候補集合を作る. この時, もし同一の集合に属する点が見つかった場合には, 近い方を候補に入れ, 遠い方は候補から外す. 次に, 候補集合に含まれている点の存在範囲を探索範囲として, より近い点がないかどうか調べる. より近い点が見つかった場合には, その都度, 候補集合を作り直し, 探索範囲を絞りながら探索を続ける. ここでも, もし同一の集合に属する点が見つかった場合には, 近い方を候補に入れ, 遠い方は候補から外す. そして, 探索範囲に含まれるリーフを全て探索し終った時点で, 検索が終了する.

4.2　顔シーケンスの照合アルゴリズム

多次元ベクトル集合に対する最近接検索の応用として, 映像から顔を抽出し, 顔の向きや表情や照明条件などが変化しているそれぞれの顔シーケンスから多次元ベクトル集合を構成し, その最近接検索により顔シーケンスの照合の実験を行った.

与えられた映像からの顔シーケンスの抽出は, [12] と同等の方法を利用した. すなわち, まず一定の間隔 (実験では 10 フレーム) の各フレームにニューラルネットワークベースの顔検出 [13] を適用する. 検出された各顔領域から RGB 空間中のガウス分布モデルに基づく肌色モデルの推定を行い, このモデルをもとに前後に続くフレーム中で肌色領域を追跡することで顔シーケンスの抽出を実現している. シーンチェンジが検出されるか, 対応する肌色領域が見つからない場合, 顔シーケンスが終了したと判断する. 抽出された各顔シーケンス S_iは, その中で顔検出システムにより検出された顔 $F_{i,k}$により構成されていると見なす. 加えて, [12] と同等の方法により, 各 S_iごとに $\{F_{i,k}\}$ から最も正面向きの顔 F_i^{best}を選ぶ. すなわち, $F_i^{best} \in \{F_{i,k}\}$.

顔 $F_{i,k}$は, 一定の大きさに正規化され (実験では 64×64), [13] と同等の手法で照明条件の補正とヒストグラム等化が施されている. これらの顔から多次元ベクトル特徴量を得るため, Eigenface の手法 [14] を用いた. 実験では, $\{F_i^{best}\}$ をトレーニング集合として Eigenface を生成し, これをもとに各顔 $F_{i,k}$を 16 次元の Eigenface 空間中のベクトル $f_{i,k}$へと変換している. 二つの顔シーケンス S_iと S_jの距離 $d(S_i, S_j)$ を, 以下のように定義している.

$$d(S_i, S_j) = \min_{k,l}(|f_{i,k} - f_{j,l}|).$$

この定義を用い, 与えられた顔シーケンスに対して, 最も近い顔シーケンスを検索する処理を実現した.

実験は, 5 時間分の CNN Headline News の映像を利用して行った. ここから, 556 の顔シーケンス, 及び 8134 の顔を抽出し, 556 の正面向きの顔をトレーニングセットとした 8134 の Eigenface 空間中の多次元ベクトルを算出した. またこれらの顔シーケンスに対し, 人手により識別できる顔には名前を付与しており, 556 の顔シーケンス中 288 の顔に名前をつけている.

これらのデータを得て, まず顔シーケンス間の距離 $d(S_i, S_j)$ の有効性を確認する実験を行った. 人手で付与した名前をもとに, 顔シーケンス S_iと S_jが同一人物

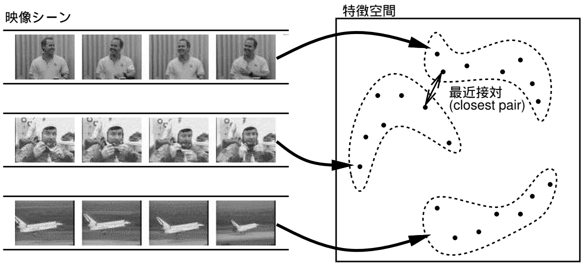


図 6: ベクトル集合を単位とした最近接検索

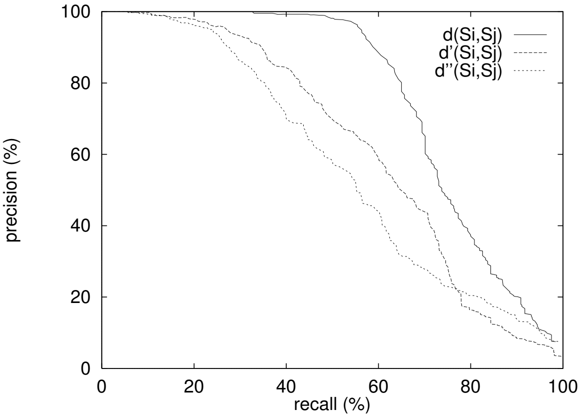


図 7: 顔シーケンス間の距離の評価
横軸は呼出率, 縦軸は適合率を表す.

である事象を $P(S_i, S_j)$ とする. また, 顔シーケンス S_i と S_jについて, これらが同一人物であるという推定として, $d(S_i, S_j) < \theta$である事象を$\hat{P}(S_i, S_j|\theta, d)$ とする. 与えられた d とθを用いた上記の同一人物の推定に対し, 適合率は $Pr[P|\hat{P}]$, 呼出率は $Pr[\hat{P}|P]$ と考えられるので, θを変えながらこれらを平面上にプロットすることにより, 距離 d の性質を見ることができる. ここでは比較のため, 顔シーケンス間の距離として, [12] で述べられている, 両シーケンス中で最も正面向きの顔を用いた距離

$$d'(S_i, S_j) = |f_i^{best} - f_j^{best}|$$

および両シーケンス中の最初の顔を用いた距離

$$d''(S_i, S_j) = |f_{i,1} - f_{j,1}|$$

と提案手法を比較することとした. 結果を図 7に示す. 図に示されている通り, 提案手法の方が呼出率と適合率を同時により大きくすることに成功しており, 顔シーケンス間の距離の評価法として優れていることが分かる.

4.3 多次元インデックスによる高速実装

多次元インデックスを用いて上述の顔シーケンスの照合アルゴリズムを実装し, 多次元インデックスの有効性を評価する実験を行った. 実験データには, 既に述べた CNN Headline News の映像を使用した. 映像から抽出された 556 の顔シーケンスの一部を無作為に選び, 大きさの異なる 6 個のデータセットを作成して使用した. 具体的には, シーケンスが 100, 200, 300, 400, 500, 566 のものを作成した. これらのシーケンスに含まれる顔画像の数は, それぞれ 1414, 2599, 4174, 5504, 7257, 8134 である.

実験では, ある顔シーケンスと最も近いシーケンスを見つけるという照合処理を実行し, 照合に要する CPU 時間とディスクアクセス量 (データ読み込み量) を測定した. それぞれのデータセットに含まれるすべてのシーケンスについて照合を行い, 平均値を測定結果とした. プログラミング言語には C++ を使用し, Sun Microsystems 社製の Ultra-60 (CPU: UltraSPARC-II 360 MHz, メモリ: 512 Mbytes, OS: Solaris 2.6) の上で測定した. 実験結果を図 8に示す. 比較のために, 線形探索を用いて実装した場合の結果も併せて示している. 横軸は, データセットに含まれる顔画像の数である. この値がデータベースの大きさに相当する. 縦軸は, 左が CPU 時間, 右がディスクアクセス量である. SR-tree の効果はディスクアクセス量で顕著であり, 顔画像の数が 8134 の時には線形探索の 17% しかアクセスしていない. これは, 階層的な領域分割を行うことによって探索範囲が小さくなっていることを示しており, 多次元インデックスの利点が発揮されていることがわかる. また, CPU 時間についても線形探索の 24% になっている. さらに, 線形探索に対する SR-tree の比 (図中の “SR-tree / Linear Scan”) を見てみると, データベースが大きくなるほど小さくなっていることがわかる. このことは, 提案手法のスケーラビリティを裏付けるものであり, 大規模なデータベースでの有効性を示唆する結果である.

5 おわりに

本論文では, マルチメディアアプリケーションの大規模化を実現するための一手法として, マルチメディア情報処理に多次元インデックスを導入する方法につい

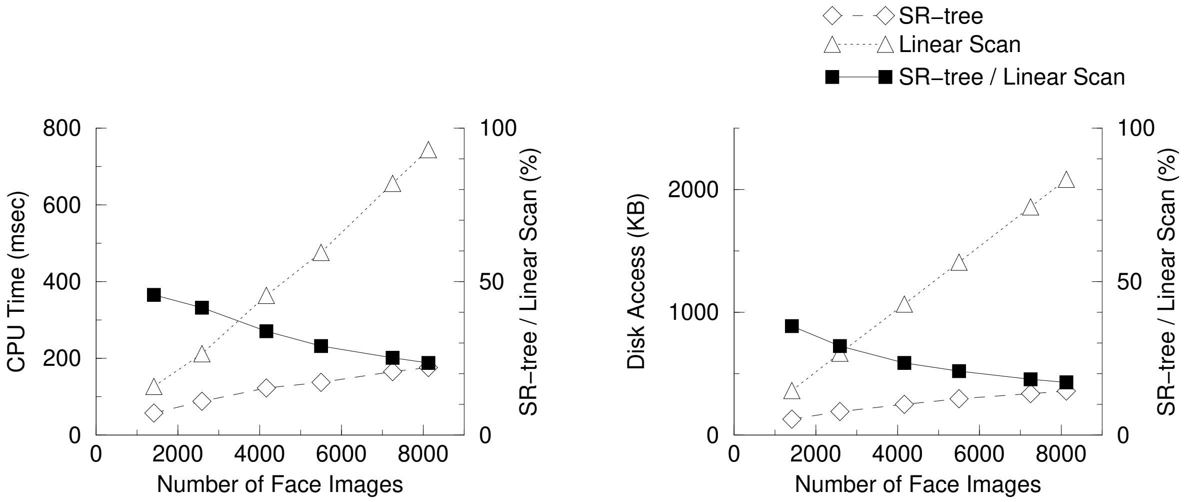


図 8: ベクトル集合単位の最近接検索の性能

て説明し, その有効性について検証した. まず, SR-tree の静的構築法を示し, 類似画像検索を例としてその効果を実証した. 次に, マルチメディア情報の照合方法として多次元ベクトル集合を単位とした最近接検索の有効性を示し, 多次元インデックスを応用することで処理を高速化できることを示した.

マルチメディアアプリケーションに高速化手法を導入することは, 単に処理時間を短縮するだけでなく, 高精度な処理についての実用性や大規模データへの適用可能性を広げるという役割も持っている. そのため, 今後, マルチメディアアプリケーションが広がりを見せるにつれて, 多次元インデクシング手法などの高速化技術の役割も益々大きくなっていくものと予想される.

参考文献

[1] 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, pp.369–380, May 1997.

[2] 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 D-I, vol. J80-D-I, no. 8 (Aug. 1997) pp. 703–717.

[3] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by Image and Video Content: the QBIC System," IEEE Computer, Vol.28, No.9, pp.23–32, Sep. 1995.

[4] A. Guttman, "R-trees: a Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD, Boston, USA, pp.47–57, Jun. 1984.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an Efficient and Robust Access Method for Points and Rectangles," Proc. ACM SIGMOD, Atlantic City, USA, pp. 322–331, May 1990.

[6] 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.

[7] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbor Queries," Proc. ACM SIGMOD, San Jose, USA, pp.71–79, May 1995.

[8] G. Hjaltason and H. Samet, "Ranking in Spatial Databases," 4th International Symposium, SSD'95, Portland, USA, pp. 83–95, August 1995.

[9] S. Berchtold, C. Boehm, D. A. Keim, and H.-P. Kriegel, "A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space," Proc. ACM PODS, Tucson, USA, pp.78–86, May 1997.

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

[11] S. A. Nene and S. K. Nayar, "A Simple Algorithm for Nearest Neighbor Search in High Dimensions," IEEE Trans. PAMI, vol. 19, no. 9, pp. 989–1003, Sep 1997.

[12] S. Satoh, Y. Nakamura, and T. Kanade, "Name-It: Naming and detecting faces in video by the integration of image and natural language processing," in *Proc. of International Joint Conference on Artificial Intelligence*, pp. 1488–1493, 1997.

[13] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. on PAMI*, vol. 20, no. 1, pp. 23–38, 1998.

[14] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.