Hilbert R-treeを用いた一括検索処理の高速化に関する研究

岸川 直樹[†1]　村上 義明[†1]　児玉　晋[†1,a)]　安藤　悠佳[†1]　篠原 武[†1]

概要：
本論文では高速な近似検索を実現するためにHilbertR-tree を用いた一括検索処理を提案する．実験より，一括検索処理が実際に検索処理を高速化することが確認できた．しかし，素朴な一括検索処理を用いると，個別に検索する場合に比べて距離計算回数は増加してしまうことが分かった．そこで，本論文では，一括検索処理の距離計算回数を削減する二つの手法を提案する．実験により，これら二つの手法を適応した一括検索は，個別検索と比較して約 60 %の距離計算を減らすことができることを確認した．

1. はじめに

多次元データを高速に近似検索する手法として，特徴空間を複数の部分空間に分割し，それらを索引付けする階層的空間索引がある．代表的な階層的空間索引としてはR-tree [1] が知られており，その亜種となる空間索引も過去に様々なものが提案されている．それら亜種の中でも，Hilbert R-tree [2] は最も検索効率が良いことが確認されている．近似検索には質問方法として範囲質問や k 近傍質問などがあるが，本論文では，その中でも最近傍質問と呼ばれる質問方法を用いる．階層的空間索引における最近傍質問では，初めに検索範囲を無限大に初期化し，検索範囲と交差している部分空間を持つノードを訪問していく．訪問したノードがオブジェクトを保持している葉ノードは，質問点とオブジェクトとの距離を計算しその距離が検索範囲より小さい時，検索範囲を収縮させそのオブジェクトを暫定解として登録する．このような処理を繰り返していき検索範囲と交差している全ての部分空間を訪問し終わると検索は終了する．最近傍質問においては検索範囲の収縮が重要であり，検索範囲を早く収縮させることができれば検索の高速化につながると考えられる．Hilbert R-tree における近似検索高速化手法として，履歴参照質問 [3] が二宮によって提案された．履歴参照質問では，動画から連続して切り出される画像データが質問点として連続して与えられる状況を想定している．そのような質問点の隣り合うもの同士の非近似度（距離）は動画の性質上非常に小さく，履歴参照質問はこの特性を利用し，過去の検索結果を現在の検索に反映させることで，検索の高速化を図っている．

本論文では，履歴参照質問と同様の状況を想定し，連続して与えられる質問点をまとめて処理を行うことで近似検索の高速化を図る一括検索処理を提案する．一括検索処理は，質問点をまとめて処理することで，空間索引が書き込まれたファイルからのノード読み込みにかかるI/O コストを削減する手法である．質問点を個別に検索する場合（個別検索）は，個々の質問点に対する処理は独立している．そのため，同じノードを訪問する質問点が複数あったとしても空間索引のノードは何度も訪問されファイルから読み込まれる．それに対し，一括検索処理では複数の質問が同じノードを訪問するとき，そのノードへの訪問は一度だけでよくファイルからの読み込みも一度だけで済む．そのため，一括検索処理は個別検索処理に比べ訪問ノード数を削減し，I/O コストを削減できると期待できる．

実験より，一括検索処理は個別検索に比べ，検索時の訪問ノード数を大幅に削減し，検索時間も削減できることを確認した．しかし，個別検索に比べると一括検索に必要な距離計算の回数は大きく増加してしまった．これは個別検索ではノード訪問順を考慮するのに対し，一括検索処理では考慮しないため，個々の質問点の検索範囲の収縮が遅くなるからだと考えられる．そこで，一括検索処理の距離計算回数を削減し，さらなる高速化を図るために，一括検索処理におけるノード訪問順の制御を提案する．また，隣接する質問点間の距離が近いことを利用した，三角不等式による距離計算削減手法を提案する．

動画から切り出された画像データと楽曲から切り出され

[†1] 現在，九州工業大学院情報工学府情報科学専攻
Presently with Kyushu Institute of Technology Department of Artificial Intelligence
[a)] m673012s@iizuka.isc.kyutech.ac.jp

た音データの二つのデータを用いて提案手法の検証を行った結果，画像データでは，ノード訪問順を制御した一括検索処理は，訪問ノード数を約 25 分の 1 に削減することができた．距離計算回数は約 1 割増加したものの，検索時間を約 3 割削減することができた．また，ノード訪問順を制御した一括検索処理に，隣接質問による距離計算削減法を適用すると，適用しない場合に比べ，距離計算回数を約 6 割削減し，検索時間も約 3 割削減することができた．音データでは，ノード訪問順を制御した一括検索処理は画像データの傾向と同じであり，個別検索と比べて検索時間は約 3 割削減できた．また，隣接質問による距離計算削減法を適用すると，適用しない場合に比べ，距離計算回数を約 2 割削減できたが，検索時間は約 2 割悪化してしまった．

上記の結果から，ノード訪問順を制御した一括検索処理は画像データと音データの両方で効果的な手法であるとわかった．また，隣接質問による距離計算削減法は，画像データに関しては，検索時間を大きく削減できたため，効果的な手法であるといえる．しかし，音データではむしろ，隣接質問による距離計算削減法を適用することで検索時間が悪化してしまうことがわかった．

本論文の構成は以下の通りである．2 章では近似検索，3 章では R-tree を説明し，4 章で Hilbert R-tree に関して説明する．そして 5 章で一括検索処理について，従来手法と提案手法について説明する．6 章で実験について説明し，7 章で実験結果と考察を述べる．そして 8 章でまとめと今後の課題を述べる．

2. 近似検索

2.1 距離空間

データ間に非近似度 (距離) を定義することで，質問点からの距離の順番でオブジェクトを取り出すことにより，近似検索を実現することができる．近似検索とは，質問点と近似するデータをデータベースから取り出すことである．

近似検索のデータベースが対象とする特徴空間全体を $\mathcal{U} = \mathbb{R}^n$ とする．ここで，n は特徴データの次元数である．任意の 2 点間のオブジェクト間の非近似度の指標を示す距離関数を $d : \mathcal{U} \times \mathcal{U} \to \mathbb{R}^+$ とし，$\mathcal{D} = (\mathcal{S}, d)$ を距離空間と呼ぶことにする．近似検索では，距離関数 d は距離の公理を満たすものとする．最も重要な条件は三角不等式と呼ばれる以下の条件である．

$$d(X, Y) \leq d(X, Z) + d(Z, Y)$$

ここで，$X, Y, Z \in \mathcal{U}$ である．

本実験で用いる距離計測について説明する．特徴空間内の任意のオブジェクトを x とする．x の特徴は n 組の実数 $(x_{(1)}, x_{(2)}, \ldots, x_{(n)})$ で表される．以下の二つの距離計測関数は距離の公理を満たすものである．

$$L_1\text{距離} : D(x, y) = \sum_{i=1}^{n} |x_{(i)} - y_{(i)}| \quad (1)$$

$$L_2\text{距離} : D(x, y) = \sqrt{\sum_{i=1}^{n} (x_{(i)} - y_{(i)})^2} \quad (2)$$

本論文の実験では，距離計算を L_1距離 で計測する．また，質問方法として最近傍質問を用いる．最近傍質問は，質問点から距離が最も小さいオブジェクトを取得する質問方法である．具体的には，最近傍質問では初めに検索範囲 r を無限大に初期化する．そこから検索をはじめて，検索範囲内のオブジェクトを暫定解として登録していき，検索範囲をそのオブジェクトと質問点との距離に収縮していきながら検索を行う．

2.2 距離計算の打ち切り

最近傍質問ではオブジェクトが検索範囲内にあるかどうかが重要であり，検索範囲外のオブジェクトと質問点の距離を正確に求める必要はない．この点に着目した距離計算の打ち切りについて説明する．

L_1距離 は式 1 からわかるように，特徴空間上の二つのオブジェクトの各軸における値の差の絶対値を総和したものである．質問点とオブジェクトの距離を計算する際，ある軸における値の差の絶対値を足した時点で，その軸までの総和が検索範囲を超えると，そのオブジェクトは必ず検索範囲内に存在しないことがわかる．そこで，各軸に関して計算しその値を足した後すぐに検索範囲と比較を行うことで，全ての軸に関して計算することなく距離計算を途中で打ち切ることができる．この距離計算の打ち切りを行うことで，一距離計算あたりのコストを下げることができる．

3. R-tree

Guttman が提案した R-tree [1] は代表的な空間索引の一つであり，オブジェクトの追加・削除などの動的な操作を可能とする多岐平衡木である．R-tree は，対象とする空間を n 次元超直方体である最小包囲矩形（MBR）で分割する．R-tree の内部ノードはその全ての子ノードを包括する MBR と子ノードへのポインタを持ち，葉ノードはデータベース 内のオブジェクトへのポインタを持つ．R-tree の検索は根ノードから始まり，深さ優先探索でノードを探索していく．検索が葉ノードのとき，その葉ノードが保持しているオブジェクトとの距離を計算し，検索範囲よりも小さければ検索範囲を更新する．検索が内部ノードのとき，質問点とその子ノードの MBR との距離を計算し，検索範囲と交差する MBR を持つノードのみ訪問する．この時，訪問する必要のあるノードは Active Branch List（ABL）と呼ばれる優先度付き待ち行列に挿入する．ABL 内のノードは，質問点と MBR との距離の昇順でソートされる．ABL の先頭から順にノードを訪問することで，質問点との距離

が小さい MBR を持つノードから訪問することが可能になる．このようにノードの訪問順を制御することで，質問点の検索範囲の収縮を早めることができる．図 1 には MBR A，B，C と質問点を示している．

例えば図 1 のように，初期検索範囲が全ての MBR A，B，C，と交差している状況を考えると，訪問順を考慮しない場合は親ノードの要素順，この例では A，B，C の順にすべて訪問される．それに対して訪問順を制御する場合は，質問点から最も近い C から訪問し，C の持つオブジェクトの中で質問点との距離が最も近いものとの距離に検索範囲が収縮する．そうすると，訪問順を制御しない場合では，訪問しなければならなかった A や B に訪問する必要がなくなる．また，A や B の持つオブジェクトとの距離計算もせずに済む．このように R-tree はノードの訪問順を制御することで検索範囲の収縮を早め，効率的な検索を実現している．

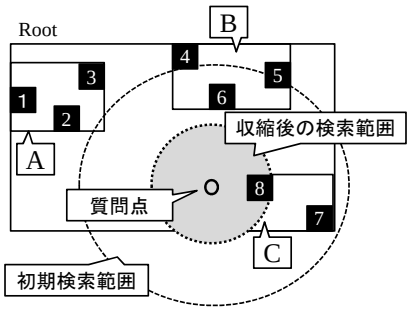

図 1 ノード訪問順と検索半径の収縮
Fig. 1 Node visit order and Shrink of search radius

4. Hilbert R-tree

4.1 Hilbert 曲線順序付け

空間充填曲線の Hilbert 曲線は，多次元データを空間的に近い順に一次元順序付けできることが知られている．過去の研究で，R-tree を Hilbert 曲線を用いて構築した Hilbert R-tree [2] は，R-tree に比べ検索効率が大幅に向上することが確認されている．

Hilbert 曲線順序付け手法としては、Hilbert 値と呼ばれる Hilbert 曲線上の始点からの距離を用いるのが一般的である．しかし，Hilbert 値を取得することは非常にコストがかかるという問題がある．そこで，田中は Hilbert 値を求めることなく高速に Hilbert 曲線順序付けが可能なヒルベルトソート [4] という手法を提案した．ヒルベルトソートを用いることで高速に Hilbert R-tree が構築できることが過去の研究より確認されている．

4.2 ヒルベルトマージ

田島 [5] は Hilbert R-tree に対する効率的な一括挿入手法としてヒルベルトマージ を提案した．ヒルベルトマージは，まずヒルベルトソートを用いて挿入するオブジェクト群を整列させる．そして，空間索引に格納されているデータが Hilbert 曲線順に整列済みであることを利用し，空間索引と挿入オブジェクト群を織り交ぜるようにマージしながら挿入する．ヒルベルトマージは，同一ノードを複数回更新することなく挿入処理を行えるため，逐次的に挿入を繰り返す手法と比べ，高速に挿入処理を行うことができる．

5. 一括検索処理

一括検索処理は，Hilbert 曲線順で近いものは空間的にも近い可能性が高いという点に着目し，各質問点の初期検索範囲を求める前処理を行う．ヒルベルトマージを応用し，それぞれの質問点が Hilbert R-tree のどの葉ノードに挿入されるかを一括して調べる．各質問点は辿り着いた葉ノード内のオブジェクトとの距離を計算し，最も近い距離をそれぞれの質問点の初期検索範囲として登録する．

また，一括検索処理は，根ノードから深さ優先探索でノードを訪問していく．その際，質問オブジェクト群と MBR の交差判定をおこない，一つでも質問オブジェクトがノードを訪問する必要のあるノードのみ訪問する．このように，質問オブジェクトを一括して検索処理を行うことで，個別検索に比べ，訪問ノード数が大幅に削減できる．

ヒルベルトマージを応用した前処理により，一括検索処理は個別検索に比べ距離計算回数の増加を抑えることができることが実験から分かった．しかし，個別検索に比べれば，依然として距離計算回数は多い．そこで一括検索処理の距離計算回数を削減し，さらなる検索高速化を図るため，以下で説明する二つの手法を提案する．

5.1 一括検索処理におけるノード訪問順制御

一括検索処理におけるノードの訪問順を制御するため，R-tree における検索処理と同様に ABL を使用する．一括検索処理で検索が内部ノードのとき，その子ノードの訪問判定を行い，訪問する必要のある子ノードのみ ABL に挿入する．ABL 内のノードは，そのノードを訪問する各質問点と MBR の最小距離を基準に昇順にソートされる．そして，ABL の先頭ノードから順に再びノードを訪問していくことで，質問点群との距離が小さい MBR を持つノードから訪問することができる．このように，ノードの訪問順を制御することで，各質問点の検索範囲の収縮が早まり，訪問順を制御しない一括検索処理では課題となっていた距離計算回数の増加を抑えることができると考えられる．

5.2 隣接質問による距離計算削減法

この手法は，質問点群の隣り合うもの同士の距離が近く，質問点との距離計算を終えたオブジェクトが検索範囲外ならば，その次に続く質問点に対しても検索範囲外である可能性が高いことに着目している．

検索が葉ノードに達したとき，通常ではその葉ノードを訪問した質問点群は，葉ノードが持つすべてのオブジェク

トと距離計算を行う必要がある．しかしこの手法を用いることで，質問点 q_i とオブジェクト o との距離計算を省くことができる．ここで，q_i は葉ノードに到達した質問点群のうち連続した部分の要素であり，i は必ず質問点群の要素数以下である．また q_i の検索範囲を r_{q_i} とし，o との距離計算を終えた質問点を q_0 とする．三角不等式より以下の不等式が成り立つ．

$$d(q_i, o) \geq d(q_0, o) - d(q_0, q_i) \tag{3}$$

このとき，

$$d(q_0, o) - d(q_0, q_i) > r_{q_i} \tag{4}$$

が成り立つならば，必ず

$$d(q_i, o) > r_{q_i} \tag{5}$$

が成り立つ．式 5 より，検索範囲 r_{q_i} 内に o が存在しないことがわかるので，q_i と o の距離計算を省略することができる．距離計算が省略できる場合の例を図 2 に示す．

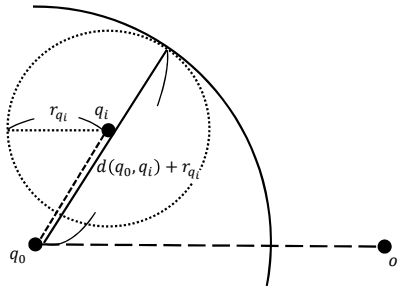

図 2 質問点とオブジェクトの距離を用いた距離計算の省略
Fig. 2 Reduction of distance calculations by using distance between query and object

ここで必要となる q_0 と q_i の距離計算は事前に行わなければならないが，質問点群の各質問点の距離をすべて計算するのはコストが掛かりすぎるという問題がある．そこで，隣接質問による距離計算削減法では，まず検索処理に入る前に各質問点に対してその直後の質問との距離を計算し，その距離を保持させる．そして，三角不等式より以下の条件式が成り立つことを利用して，q_0 と q_i の距離の見積もりを行う．

$$\sum_{n=1}^{i} d(q_{n-1}, q_n) \geq d(q_0, q_i) \tag{6}$$

このとき，

$$d(q_0, o) - \sum_{n=1}^{i} d(q_{n-1}, q_n) > r_{q_i} \tag{7}$$

が成り立つならば，必ず式 5 となるので，q_i と o の距離計算を省略することができる．

葉ノードで，ある質問点がオブジェクトと距離を計算し終えると，その質問点の後に続く質問点に対し上記の条件式 7 を適用していく．この手法により，一括検索処理における距離計算回数を削減することが期待できる．

6. 実験と考察

実験データとして画像データと音データの二つを用いる．画像データは約 2,800 本の動画から切り出した約 700 万件の画像フレームを 64 次元に特徴抽出し登録したデータベースである．質問データとして約 100 本の動画から切り出した 9 万件の画像フレームを特徴抽出したものを用いる．また，音データは約 1500 曲の楽曲から切り出した約 700 万フレームを 96 次元に特徴抽出し登録したデータベースである．質問データとして約 30 曲の楽曲から切り出した 9 万フレームを特徴抽出したものを用いる．特徴抽出には，画像データは 2 次元高速フーリエ変換，音データは 1 次元高速フーリエ変換を用い，各周波数成分のパワースペクトルを特徴量としている．それぞれの質問データは，似たデータが見つかる近質問，やや似たデータが見つかる準近質問，似たデータが見つからない遠質問の各 3 万件で構成されている．質問方法は最近傍質問を行う．

実験に用いる空間索引は，Hilbert R-tree である．Hilbert R-tree に対し画像データと音データを用いて各手法を比較する．一括検索処理に関しては，一括して処理を行う質問データの数を 10 から 1000 まで変化させて実験を行っている．実験に用いた計算機の性能を表 1 に示す．

表 1 実験に用いた PC の性能
Table 1 The spec of PC

CPU	Intel(R) Core(TM) i7 CPU 860 2.80GHz
メモリ	4GBytes

6.1 一括検索処理

本節では，素朴な一括検索処理と個別検索の比較検証を行う．個別検索と素朴な一括検索の画像データの実験結果を表 2，音データの実験結果を表 3 に示す．表 2，表 3 より，素朴な一括検索処理では一括質問数が増えるほどに訪問ノード数が減少していくことがわかる．距離計算回数は一括質問数を変化させても一定であり，一括質問数が 1000 のとき最も高速に検索を行えていることがわかる．また，距離計算の打ち切りを行うと行わない場合に比べ，約 3 割検索時間が削減できることがわかる．

6.1.1 個別検索との比較

素朴な一括検索処理が最も高速に検索を行える一括質問数 1000 のときの結果に関して，個別検索と比較する．表 2，表 3 より，個別検索に比べ素朴な一括検索処理は，画像データで約 170 分の 1，音データで約 140 分の 1 と大幅に訪問ノード数を削減できていることがわかる．しかし，距離計算回数は画像データで約 2 倍，音データで約 1.7 倍と大きく増加してしまった．検索時間は，距離計算の打ち切りを行わない場合は画像データで約 4 割，音データで約 2 割悪化し，距離計算の打ち切り行う場合は画像データで約 3 割，音データで約 1 割悪化してしまった．

6.1.2 考察

一括検索処理は同じノードを訪問する質問点をまとめて処理することで，ノード読み込みに掛かるI/Oコストを削減する手法である．そのため，一度のノード訪問で処理される質問点の数が増えるほど，訪問ノード数が減りI/Oコストも減るため，検索時間は減少すると推測される．6.1.1節より，素朴な一括検索処理では一括質問数が増えるほどに訪問ノード数が大きく減少し，検索時間も減少していった．一括して処理する質問点の数が増えるほど，一度のノード訪問で処理される質問点の数は増えるので，この結果は上記の推測を裏付けるものであるといえる．

個別検索と比べると訪問ノード数は大きく削減することができたが，距離計算回数は大きく増加し，検索時間は悪化してしまった．距離計算回数が増加したのは，素朴な一括検索処理は個別検索のようにノード訪問順を制御していないため，各質問点の検索範囲の収縮が遅くなってしまうためだと考えられる．

6.2 ノード訪問順制御

本節では，一括検索処理におけるノード訪問順制御の効果を検証する．表4にノード訪問順を制御した一括検索処理と他手法の画像データにおける実験結果，表5に音データの実験結果を示す．表4，表5より，ノード訪問順を制御した一括検索処理では，制御しない一括検索処理と同様に，一括質問数が増えるほどに訪問ノード数が大きく減ることがわかる．しかしノード訪問順を制御しない一括検索処理と異なり，距離計算回数は一括質問数が増えるほどにわずかに増加している．検索時間は一括質問数が100のとき最も高速であることがわかる．

6.2.1 距離計算の打ち切り

表4，表5より，距離計算の打ち切りを行うと，距離計算の打ち切りを行わない場合に比べ，個別検索，素朴な一括検索処理，ヒルベルトマージを応用した前処理，ノード訪問順を制御する一括検索処理の四つの手法で，大幅に検索時間を削減できることがわかる．この傾向は，データの違いや手法の違いなどに関係なく表れており，距離計算の打ち切りを行うことでおよそ2割から3割前後検索時間を削減できることがわかる．次節の比較では検索時間に関しては，距離計算の打ち切りを行う場合の結果を用いる．

6.2.2 他手法との比較

ノード訪問順を制御した一括検索処理と制御しない一括検索処理で，互いに最も高速に検索を行える場合を比較すると，距離計算回数は画像データで約3割，音データで約2割と大きく削減できていることがわかる．また，検索時間は画像データで約2割，音データで約1割削減できている．

同様にノード訪問順を制御した一括検索処理が最も高速に検索を行える結果と個別検索の結果を比べると，訪問ノード数はノード訪問順を制御しない一括検索処理同様に劇的に削減できているが，距離計算回数は画像データと音データで約1割ほど増加している．検索時間は画像データと音データで約3割ほど削減できていることがわかる．

6.2.3 考察

上記の比較結果より，提案手法である一括検索処理におけるノード訪問順制御は，各質問点の検索範囲の収縮を早め距離計算回数を削減できたと考えられる．ただ，ノード訪問順を制御しない一括検索処理の傾向とは異なり，ノード訪問順を制御した一括検索処理では一括質問数が増えるほどに距離計算回数が増加していった．また，個別検索に比べると若干距離計算回数が多いが，これらは，一括検索処理におけるノード訪問順制御では，質問点群に最も近いノードから訪問するので，各質問点に関してみると必ずしも最適な訪問順でノード訪問できるわけではないからだと考えられる．つまり，一括質問数が増えるほどに最適な訪問順でノードを訪問できない質問点も増えてしまうので，距離計算回数が増加してしまうと考えられる．

6.3 隣接質問による距離計算削減法

本節では，隣接質問による距離計算削減法の効果を検証するため，ノード訪問順を制御した一括検索処理に隣接質問による距離計算削減法を適用した場合と適用しない場合で比較を行う．表6に画像データにおける隣接質問による距離計算削減法の実験結果を，表7に音データの実験結果を示す．表6，表7より，隣接質問による距離計算削減法を適用すると，画像データと音データの両方で一括質問数が100のとき最も高速に検索が行えることがわかる．以下では隣接質問による距離計算削減法を適用する場合と，適用しない場合の両方で最も高速な一括質問数100における結果について比較検証する．

6.3.1 ノード訪問順を制御した一括検索処理との比較

表4と表5，表6，表7より，距離計算の打ち切りを行わないもの同士で比較すると，隣接質問による距離計算削減法を適用することで，距離計算回数を画像データで約6割，音データで約2割削減できることがわかる．また，検索時間は画像データで約4割，音データで約1割削減できることがわかる．距離計算の打ち切りを行うもの同士を比較すると，隣接質問による距離計算削減法を適用することで，どちらのデータでも距離計算回数をほとんど削減できず，むしろ検索時間が悪化してしまうことがわかる．

6.3.2 考察

上記の比較から，距離計算の打ち切りを行わない場合では，隣接質問による距離計算削減法を用いることで距離計算回数を大幅に削減できることが確認できた．しかし，隣接質問による距離計算削減法では．距離計算の打ち切りを行ってしまうと効果がほとんどなくなってしまうということも確認した．これは，隣接質問による距離計算削減法では質問点とオブジェクトとの距離を利用してその隣接質問

点の距離計算を省略するので，検索範囲外であることがわかったオブジェクトとの距離も，正確に測る方が効果が出やすいためだと考えられる．

画像データにおいては隣接質問による距離計算削減法を適用することで，距離計算の打ち切りを行わなくても検索時間を大幅に削減できることがわかった．それに対し音データでは，距離計算回数を画像データほどは削減できず，むしろ検索時間は悪化してしまった．音データにおいて検索時間が悪化してしまったのは，隣接質問による距離計算削減法であまり距離計算を削減できないため，隣接質問による距離計算削減法の効果よりも，距離計算の打ち切りを行うことの効果が大きいためだと考えられる．また，音データにおいて画像データほど距離計算回数を削減できなかったのは，画像データの方が音データよりも隣接質問間の距離が近いためではないかと考える．

7. まとめと今後の課題

本論文では与えられる質問点をまとめ処理することで近似検索の高速化を図る一括検索処理を提案した．一括検索処理はヒルベルトマージを応用した前処理を適用することで，個別検索よりも高速な検索を実現することができた．しかし，個別検索に比べると距離計算回数に関しては大幅に増加してしまった．そこで，一括検索処理の距離計算回数を削減し，更なる高速化を図るために一括検索処理におけるノード訪問順制御と隣接質問による距離計算削減法という二つの手法を提案した．

一括検索処理におけるノード訪問順制御は，画像データと音データの両方で距離計算回数と検索時間を大きく削減できた．それに対し，隣接質問による距離計算削減法は，画像データにおいては距離計算回数と検索時間を大きく削減し効果的であることがわかった．しかし音データでは，距離計算回数は削減できたが，むしろ検索時間が遅くなってしまった．これは，距離計算の打ち切りを行うと隣接質問による距離計算削減法の効果がほとんどなくなってしまうが，音データにおいては距離計算の打ち切りを行ったほうが検索時間を削減する効果が大きいからだと考えられる．

今後の課題としては，一括検索処理の更なる高速化と，一括検索処理の実際のシステムでの有効性の検証という二つが考えられる．

一括検索処理の高速化に関しては，隣接質問による距離計算削減法と距離計算の打ち切りをうまく併用できる手法の開発を行うことが挙げられる．具体的には，検索範囲外だとわかった時点ですぐに質問点とオブジェクトの距離計算を打ち切るのではなく，隣接質問による距離計算削減法で効果が期待できる程度まで距離を計算することが考えられる．

一括検索処理の実際のシステムでの有効性については，動画同定システム [6] を用いて検証することが考えられる．動画同定システムは動画から連続して切り出される画像フレームを用いて，動画同定を行うシステムである．個別検索は画像フレームを平均 90 ミリ秒で検索することができる．しかし，リアルタイムでの動画同定を実現するためには，動画には 1 秒当たりに画像フレームは 30 枚あるため，1 フレームあたり約 30 ミリ秒で検索を行う必要がある．そのため，個別検索では，範囲限定最近傍質問や，距離計算回数限定 [7] などの手法を用いてリアルタイムでの動画同定を実現していた．一方，一括検索処理を用いると，最も高速に検索が行える場合では，平均 45 ミリ秒で画像フレームを検索できる．つまり，一括検索処理をそのまま用いてもリアルタイムでの動画同定は難しいと予想される．そのため，一括検索処理についても，範囲限定最近傍質問や距離計算回数限定などの手法を適用することでリアルタイムでの動画同定が実現可能か検証する必要がある．

表 2 画像データ結果に対する素朴な一括検索処理
Table 2 Native batch search for image date

				打ち切りなし	打ち切りあり
	一括質問数	訪問ノード数	距離計算回数	検索時間 (分)	検索時間 (分)
個別検索		$\mathbf{1.11 \times 10^9}$	$\mathbf{5.99 \times 10^{10}}$	**161.51**	**138.17**
素朴な一括検索	10	3.50×10^8	1.33×10^{11}	253.96	196.07
	100	5.45×10^7	1.33×10^{11}	245.21	187.60
	1000	$\mathbf{6.39 \times 10^6}$	$\mathbf{1.33 \times 10^{11}}$	**243.95**	**186.32**

表 3 音データに対する素朴な一括検索処理
Table 3 Native batch search for music date

				打ち切りなし	打ち切りあり
	一括質問数	訪問ノード数	距離計算回数	検索時間 (分)	検索時間 (分)
個別検索		$\mathbf{7.98 \times 10^8}$	$\mathbf{4.59 \times 10^{10}}$	**181.01**	**138.76**
素朴な一括検索	10	2.57×10^8	7.79×10^{10}	231.37	166.52
	100	4.23×10^7	7.79×10^{10}	223.05	158.12
	1000	$\mathbf{5.62 \times 10^6}$	$\mathbf{7.79 \times 10^{10}}$	**221.47**	**156.97**

表 4 画像データに対するノード訪問順制御
Table 4 Batch search with controlled node visit order for image date

				打ち切りなし	打ち切りあり
	一括質問数	訪問ノード数	距離計算回数	検索時間 (分)	検索時間 (分)
個別検索		$\mathbf{1.11 \times 10^9}$	$\mathbf{5.99 \times 10^{10}}$	**161.51**	**138.17**
素朴な一括検索	10	3.50×10^8	1.33×10^{11}	253.96	196.07
	100	5.45×10^7	1.33×10^{11}	245.21	187.60
	1000	$\mathbf{6.39 \times 10^6}$	$\mathbf{1.33 \times 10^{11}}$	**243.95**	**186.32**
前処理を適用した一括検索	10	2.93×10^8	9.34×10^{10}	181.05	135.87
	100	5.11×10^7	9.34×10^{10}	173.77	128.29
	1000	$\mathbf{6.32 \times 10^6}$	$\mathbf{9.34 \times 10^{10}}$	**172.45**	**127.02**
ノード訪問順制御	10	2.14×10^8	6.05×10^{10}	129.10	101.03
	100	$\mathbf{4.12 \times 10^7}$	$\mathbf{6.23 \times 10^{10}}$	**124.10**	**95.51**
	1000	6.17×10^6	6.73×10^{10}	134.52	104.42

表 5 音データに対するノード訪問順制御
Table 5 Batch search with controlled node visit order for music date

				打ち切りなし	打ち切りあり
	一括質問数	訪問ノード数	距離計算回数	検索時間 (分)	検索時間 (分)
個別検索		$\mathbf{7.98 \times 10^8}$	$\mathbf{4.59 \times 10^{10}}$	**181.01**	**138.76**
素朴な一括検索	10	2.57×10^8	7.79×10^{10}	231.37	166.52
	100	4.23×10^7	7.79×10^{10}	223.05	158.12
	1000	$\mathbf{5.62 \times 10^6}$	$\mathbf{7.79 \times 10^{10}}$	**221.47**	**156.97**
前処理を適用した一括検索	10	2.08×10^8	5.99×10^{10}	177.72	123.99
	100	3.81×10^7	5.99×10^{10}	170.93	117.24
	1000	$\mathbf{5.42 \times 10^6}$	$\mathbf{5.99 \times 10^{10}}$	**169.64**	**115.94**
ノード訪問順制御	10	1.78×10^8	4.66×10^{10}	146.88	106.68
	100	$\mathbf{3.35 \times 10^7}$	$\mathbf{4.79 \times 10^{10}}$	**141.60**	**100.61**
	1000	5.04×10^6	4.95×10^{10}	144.65	102.69

表 6 画像データに対する隣接質問による距離計算削減法
Table 6 Reduction of distance calculations by neighbor question for image date

	一括質問数	距離計算回数	検索時間 (分)
距離計算打ち切りなし	10	2.84×10^{10}	77.02
	100	$\mathbf{2.72 \times 10^{10}}$	**69.70**
	1000	2.95×10^{10}	81.71
距離計算打ち切りあり	10	5.54×10^{10}	103.98
	100	$\mathbf{5.65 \times 10^{10}}$	**100.23**
	1000	6.10×10^{10}	114.34

表 7 音データに対する隣接質問による距離計算削減法
Table 7 Reduction of distance calculations by neighbor question for music date

	一括質問数	距離計算回数	検索時間 (分)
距離計算打ち切りなし	10	3.79×10^{10}	128.32
	100	$\mathbf{3.85 \times 10^{10}}$	**123.45**
	1000	3.99×10^{10}	128.71
距離計算打ち切りあり	10	4.66×10^{10}	120.99
	100	$\mathbf{4.79 \times 10^{10}}$	**117.29**
	1000	4.95×10^{10}	122.62

参考文献

[1] A. Guttman : R-trees: A Dynamic Index Structure for Spatial Seaching, in ACM SIGMOD, pp.47–57, 1984.

[2] I. Kamel, C. Faloutsos: On Packing R-trees, in Proc. Second Int. Conf. on Information and Knowledge Management (CIKM), pp.490–499, 1993.

[3] 二宮 大輔: 空間索引を用いた大量質問点の類似検索高速化に関する研究，九州工業大学大学院 修士論文，2007.

[4] 田中 晶: 空間索引のための多次元データの順序付けの高速化に関する研究，九州工業大学大学院 修士論文，2001.

[5] 田島 圭，岩崎 瑶平，篠原 武: ヒルベルト曲線順序付けを用いた空間索引構造に対する効率的な挿入法に関する研究，人工知能学会 第 81 回 人工知能基本問題研究会，2011.

[6] 浦郷 祐希, 田島 圭, 青木 隆明, 岩崎 瑶平, 篠原 武 : 空間索引による近似画像の高速検索を用いた動画同定システムの実現, 火の国情報シンポジウム 2009.

[7] 青木 隆明: 空間索引を用いた近傍点検索に対する近似アルゴリズムによる高速化，九州工業大学大学院 修士論文，2009.