学位論文　博士（工学）

高次元データ空間における大規模近傍検索のための近似 k 最近傍グラフに関する研究

2013 年度

慶應義塾大学大学院理工学研究科

岩崎 雅二郎

要旨

スマートフォンの普及に伴いインターネット上にはマルチメディアデータを含む多様な情報が急激に蓄積されるようになり，情報検索技術への要求が急増している．マルチメディアデータを検索する場合にはデータの内容を表す特徴量を抽出し，元データの代わりに，その特徴量のデータを検索する．検索精度を高めるためには高次元データが必要となるので，高次元データを高速に検索することが可能な距離基準空間インデックスが不可欠である．k 最近傍グラフをインデックスとして用いた手法は他の従来手法よりも検索時の計算量を削減できる．k 最近傍グラフは各ノードがエッジによって k 個の近傍ノードへ接続されており，任意のノードから指定された検索点に向かって，より近いノードを順次たどることで，近傍ノードを検索することができる．しかし，この検索手法には 2 つの課題がある．第 1 に，少ない計算量で検索が可能である反面，k 最近傍グラフの生成時に膨大な計算量が必要となる．第 2 に，グラフが分離する場合には候補ノードに到達することができず検索精度（再現率）が抑制される．

本論文では木構造型インデックスを組み合わせた近似 k 最近傍グラフインデックスを提案する．k 最近傍グラフの生成時には各ノードの近傍ノードを特定しなければならず，この処理に多大な計算量が必要となる．提案手法ではノードをグラフに逐次追加し，追加対象のグラフを用いて追加ノードの近傍ノードを検索することで，グラフ生成時の計算量を大幅に減らすことができる．また，逐次ノードをグラフに接続することによりグラフが分離することがなく検索精度の抑制が解消される．さらに，検索時のグラフの探索では任意のノードではなく，木構造型インデックスにより近傍ノードを特定した上で，そのノードからグラフの探索を開始することにより，検索時の計算量の削減が可能となる．一方，提案手法ではエッジが過剰に付与されるノードが生成される傾向があるので，エッジを削減する必要がある．この際，グラフの連結性を維持するために，エッジの削除によりグラフが分離するか否かを判定した上で分離しない場合には過剰なエッジを削除し，分離する場合にはエッジを削除せずに過剰ノードから遠方のノードへ削除対象エッジを移動する手法を提案する．

10 万オブジェクトの 50 次元一様分布データにおいて k 最近傍グラフに対して提案手法では生成時に 96.7% の計算量が削減された．10 万オブジェクトの画像特徴量（1,228 次元）およびエッジ数 k が 16 において検索精度 95% を得る場合に k 最近傍グラフに対して提案手法では約 73% の計算量が削減された．さらに，百万オブジェクトの画像特徴量に対してエッジ削減の手法によりエッジ数が最大 34.2% 削減された．以上より，提案手法によって計算量とエッジ数を削減できることを確認した．最後に，提案手法を実際の商品画像検索に適用する事例を紹介する．

Abstract

As the smartphone has become common recently, huge multimedia data has been accumulated on the Internet, causing the demands of information search technology to grow rapidly. To search for the multimedia data, the features representing their contents are extracted from the data, and then the features are searched for instead of the original data. Since high-dimensional data for the features is necessary to acquire higher search accuracy, the metric space indices are indispensable to search fast for such high-dimensional data. The existing method using a k-nearest neighbor graph can reduce more computational cost than other previous works. On the k-nearest neighbor graph, each node is linked to the k-nearest neighbor nodes with edges, and then the proximity search is enabled by exploring from an arbitrary node to closer ones to a query object along the edges. However, there are two issues on the graph. First, instead of the small computational cost of the search, the huge computational cost is spent to construct the graph. Second, the search accuracy (recall rate) is reduced by disconnected graph nodes which cannot be traced.

In this study, an approximate k-nearest neighbor graph index with a tree-based index is proposed. The construction of the k-nearest neighbor graph spends the huge computational cost during the construction, because k-nearest neighbors should be found to add each node to the graph. Therefore, the proposed algorithm searches k-nearest neighbors using the partially constructed graph as a search index to reduce the computational cost and preserves the graph connectivity by adding nodes incrementally. Furthermore, closer nodes to a query object are searched using the tree-based index, then the closer nodes are used to explore the graph instead of an arbitrary node. On the other hand, some nodes on the graph tend to have a large number of edges which should be eliminated. Therefore, the algorithm checks whether the connectivity would be preserved or not by eliminating the excess edges. If the con-

nectivity is preserved, the edges are eliminated. If the connectivity is not preserved, the edges are replaced to further nodes from the node having excess edges.

The cost of the proposed graph construction was reduced by about 96.7% compared to the k-nearest neighbor graph for 100,000 objects which are 50 dimensional uniform distribution data. The cost of the search was reduced by about 73 % for 95% search accuracy compared to the k-nearest neighbor graph where the number of the edges is 16 for 100,000 objects which are 1,228 dimensional image feature data. This proposed algorithm reduced up to 34.2 % edges on the graph for a million image feature objects. We concluded that the proposed algorithm can reduce the computational costs and the edges. Finally, the application example where the proposed algorithm was applied to a real internet product image search was introduced.

目次

第 1 章　序論 1
1.1　本研究の背景 1
1.2　本研究の目的と問題設定 4
1.3　従来手法の課題と本研究の提案 7
1.4　本研究の寄与 16
1.5　本論文の構成 18

第 2 章　関連研究 19
2.1　本章の目的 19
2.2　近傍検索 19
2.2.1　非距離基準空間インデックス 20
2.2.2　距離基準空間インデックス 23
2.3　近似検索 27
2.4　属性データとオブジェクトの AND 検索 29
2.5　本章のまとめ 29

第 3 章　近傍検索のための更新可能な木構造型インデックス 33
3.1　本章の目的 33
3.2　課題 33
3.3　提案手法 34
3.3.1　vp-tree の改良 35
3.3.2　M-tree の改良 38
3.4　評価実験 42
3.5　本章のまとめ 46

第 4 章　近似近傍検索のための近似 k 最近傍グラフ 47

4.1 本章の目的 47
4.2 課題 47
4.2.1 グラフ構造型インデックスによる検索 48
4.2.2 k 最近傍グラフの課題 50
4.3 提案手法 51
4.4 評価実験 53
4.4.1 一様分布データによる範囲検索と k 最近傍検索の有効性の評価 . . 55
4.4.2 インデックス生成時の計算量の評価 57
4.4.3 一様分布データによる検索時の計算量と検索精度の評価 59
4.4.4 一様分布データによる各種従来手法の検索時の計算量の評価 . . . 61
4.4.5 画像特徴量による検索時の計算量と検索精度の評価 63
4.5 本章のまとめ 64

第 5 章 木構造型インデックスを用いた近似 k 最近傍グラフ 65
5.1 本章の目的 65
5.2 課題 65
5.3 提案手法 66
5.4 評価実験 69
5.4.1 インデックス生成時の計算量の評価 70
5.4.2 一様分布データによる検索時の計算量と検索精度の評価 71
5.4.3 画像特徴量による検索時の計算量と検索精度の評価 73
5.5 本章のまとめ 76

第 6 章 近似 k 最近傍グラフにおける過剰エッジの削減 77
6.1 本章の目的 77
6.2 課題 77
6.3 提案手法 78
6.4 評価実験 84
6.5 本章のまとめ 85

第 7 章 商品画像検索への近似 k 最近傍グラフの適用事例 87
7.1 本章の目的 87
7.2 近似 k 最近傍グラフ適用時の課題 87
7.3 適用手法 88

7.4 評価実験 90
7.5 本章のまとめ 93

第 8 章 結論 95

参考文献 99

公開文献 105

謝辞 107

第 1 章

序論

1.1 本研究の背景

移動体通信における広帯域通信が普及し，さらに，高度な画像処理を可能とする高性能な CPU を有するスマートフォンが一般的になった現在，画像や映像などを撮影しインターネット上で共有することが以前に比べ格段に容易になった．このことからインターネット上で日々膨大なマルチメディアデータが蓄積され続けており，大量のマルチメディアデータから所望のデータを効率良く探索する技術への要求は以前にも増して強くなってきている．なお，本論文では大量のデータから所望のデータを探すことを探索とよび，探索は分類や検索を包含する概念として用いる．

マルチメディアの一つである画像を対象に画像の特徴量を用いた画像検索の様々な研究 [20],[3], [37],[47] が行われてきた．その起点となる研究として QBIC[20] がある.QBIC は画像から色，配色，形状といった特徴量を抽出し検索を行う．QBIC では問合せ画像を与えて検索する手段の他に

- 画像の概略を描く
- 利用者が画像の構成色を指定する

といった手段を提供している．しかし，概略画を描くことは利用者にとって負担が大きい．また，構成色を指定するにしても，色に関する利用者の記憶は曖昧なので利用者が適切に指定できるとはいえない上にやはり負担も大きい．

問合せとしての画像を指定して類似する画像を検索する類似画像検索では，問合せ画像がシステムから提示される場合には利用者は画像を選択するだけでよいので負担が少ない．そこで，検索対象画像集合から任意の画像を提示する方法があるが，単に任意の画像

を提示するだけでは所望の画像に類似する画像が提示されるとは限らず効率が悪い．

また，実際には一回の問合せでは所望の画像を得られることは少なく，検索結果の画像を用いて何度か検索を繰り返すことで画像特徴量の空間を探索し，所望の画像にたどり着ける．しかし，この操作の繰返しによってローカルミニマムに落ち込み，所望の画像にたどり着けない場合がある．さらによくないことには，画像特徴量の空間において対象画像集合の分布を把握する手掛かりが検索結果だけなので分布を的確に把握できない．したがって，利用者はローカルミニマムに陥ったことすら気づかない．

一方，こうした問題点を解決する手段として画像分類がある．その一つとして SOM を用いて分類する方法 [58],[55],[25],[59] がある．また，分類の制約条件をあらかじめ人為的に指定する分類とクラスタリングによる方法を組み合わせた研究 [57] や絵画に特化して自動的に分類する研究 [60]，個々の分類の条件としてテンプレートを利用して自然画像を分類する研究 [32] がある．これらの分類手法では利用者に提示した分類を選択するだけで利用者は所望の画像にアクセスできるだけでなく，対象画像集合の分布が分類として表現され，利用者が画像空間における対象画像集合の分布を把握しやすい．その反面利用者の要求する条件に適合する分類が提示されなければ，利用者は個々の分類を適切に選択することができない．また，分類の精度が悪く所望する画像が誤って他の分類に属してしまっていると，利用者がその画像にアクセスすることが不可能になる．

このように一般的な画像探索手段である検索や分類手法にはそれぞれ問題があり実際のアプリケーションに適用することが困難である．さらに，分類，検索の機能が充実していても，やはり最終的には分類や検索結果として，ある程度の数の画像を実際にブラウズする必要がある．したがって，総合的な画像アクセスの効率化を目指すには，利用者が所望する画像をいかに効率良く提示できるかも重要となる．提示する画像が少数であれば単にリストアップするだけでよいが，100 以上の画像を提示する場合には特徴量空間の視覚化によるビューを提供することが探索の効率化に重要である．情報の視覚化の研究として三次元空間での類似度をバネモデルにより表現し配置する方法 [56],[13] や，Multidimensional Scaling (MDS) を利用する方法 [40] がある．さらに，ボトムアップにクラスタリングすることでビューを階層化する方法 [14] がある．しかし，いずれも計算量が多いので大量の画像をビューへ実時間で配置することが一般に困難である．

このようなことから筆者らは画像の探索機能の融合手法 [62],[61] を提案した．画像検索では適切な問合せ条件の指定が困難である．また，画像分類では利用者の要求にあった分類を生成することが困難であったり，十分な分類精度を得ることができないという問題があった．そこで，画像の分類と検索を融合することにより，分類の代表画像を画像検索での問合せ画像として利用でき，また，分類木をたどって所望の画像に到達できなかった

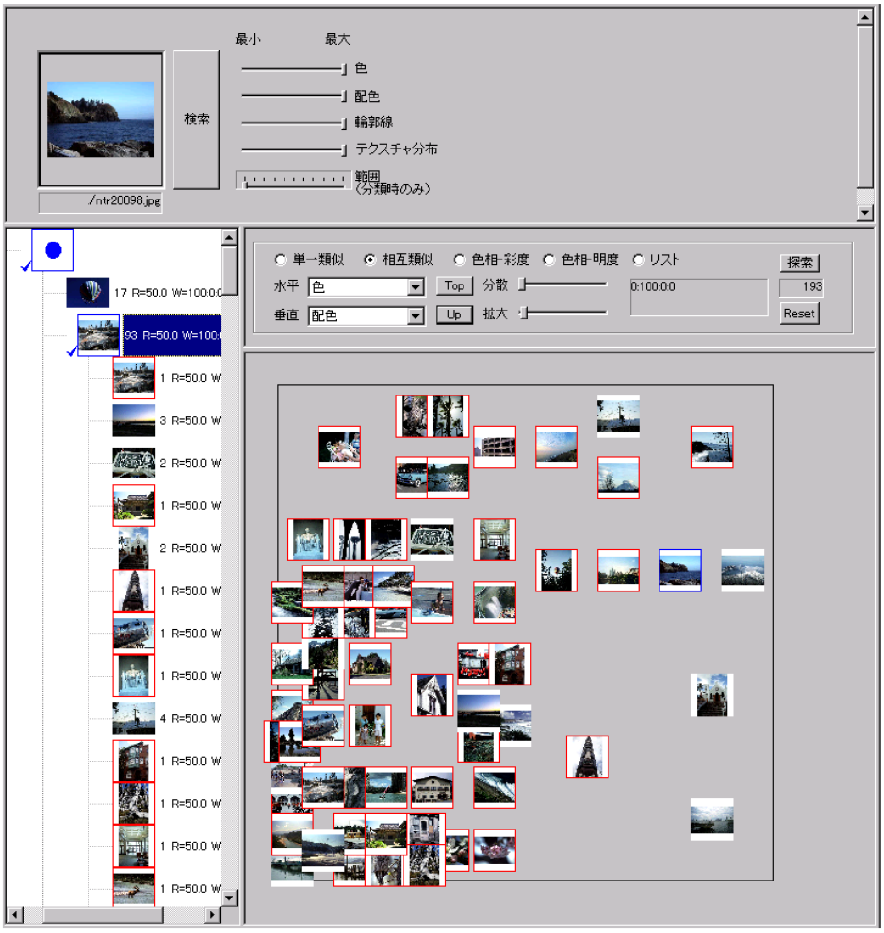


図 1.1　画像探索機能の融合例

場合には適宜検索に移行することで両者の問題点を補完した．さらに，個々の分類および検索結果である画像集合を 2 次元の画像空間へ実時間で視覚化する手法を提案し画像の視認性を高めただけでなく画像特徴量の空間における画像の分布の把握が容易となり，結果としで総合的に画像探索の効率向上を実現した．図 1.1 にその画面を示す．

個人が有する画像数程度，つまり，多くても数千画像程度の画像の場合にはこの手法が有効であるが，数百万件といった画像数の場合には問題が生ずる．画像特徴量を用いてクラスタリングにより画像を階層構造状に分類し，分類の意味を提示するために，その分類を代表とする画像を表示すると図 1.2 のようになる．人が選択しやすいように各分類の画像数の上限を制限する必要があるので，画像数が多くなると階層が深くなるだけでなく，各階層の分類も多くなる一方で代表画像間の差異が少なくなる．その結果，人が分類を選択することが困難になる．したがって，画像特徴量によって生成される分類によって所望

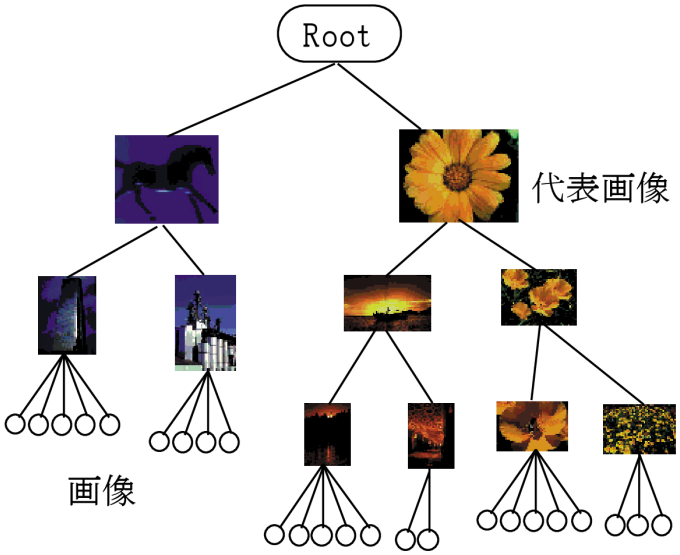


図 1.2　画像特徴量を用いた画像の分類木の例

の画像を得ようとするには，画像数の上限が存在し，数百万件といった画像数には対応できないと予想した．

一方，検索に関しても大量のマルチメディアデータから特徴量を事前に抽出しておいたとしても，特徴量を線形探索する場合には画像数の増加に従い，検索処理の時間も増加する．しかし，特徴量の検索処理時間を低減するインデックスが様々提案されており，数百万といった画像数にも実時間で検索できる可能性があると考え，多次元ベクトルデータの検索インデックスの研究に着手した．

1.2　本研究の目的と問題設定

前節で述べたように近年爆発的に増加しているマルチメディアデータを高速に検索する要望が急増している．長大なマルチメディアデータ自体にアクセスして実用的な処理時間で検索を行うことは不可能なので，通常，その内容を表す特徴量を抽出した上で検索を行う．たとえば，数百万以上の画像から所望の画像を得る方法として指定した画像に対して類似する画像を検索する類似画像検索の技術がある．画像から多次元ベクトルデータとして特徴量のオブジェクトを抽出し，近傍検索（多次元ベクトルデータ空間において指定された画像のオブジェクトに対して近傍のオブジェクトを検索する）を行うことで類似画像検索を実現できる．しかし，事前にオブジェクトを抽出してもオブジェクト自体にアクセスして検索（線形探索）する場合には，検索対象のデータ数が増加するとデータ数に比例

して検索処理の時間を要することになる．したがって，大量のオブジェクトを対象とした検索を可能とするためにはオブジェクトに対するインデックスが必要となる．そこで，本論文では大量のオブジェクトを高速に検索するインデックスを提案することを目的とする．

本論文で提案するインデックスを商品画像データベースなどの実際のシステムに応用することを想定し，以下の問題設定の下に大量のオブジェクトを高速に検索するインデックスの研究を実施した．

- 多次元ベクトルデータ
- 高次元データ
- 近傍検索
- 多様な距離関数への適用
- オブジェクトの追加が可能

次に，各問題設定の詳細について述べる．

多次元ベクトルデータ

本研究の起点はマルチメディアデータから抽出された特徴量の検索の高速化であるが，提案手法自体はマルチメディアの特徴量のみならず様々なデータ種別への応用が可能なので，本論文では対象データを特徴量に限定するものではなく多次元ベクトルデータ（以降多次元データと記す）として論述する．なお，提案手法の評価では主に一様分布データ（各次元要素を乱数で生成したデータ），および，非一様分布データとして画像から抽出した特徴量を利用する．追試が可能なように一様分布データを利用するが，空間上でオブジェクトが均等に分布している一様分布データは現実にはあまり存在しない特異なデータである．そこで，実データの例として画像から抽出した特徴量も評価に利用する．画像特徴量として色ヒストグラム，色やエッジの分布，テクスチャ [67] を用いる．特徴量の詳細に関しては開示できないが，一般的な色ヒストグラムを簡単に説明する．RGB の 3 次元色空間を均等のブロックに分割し，画像を構成する各画素値がどのブロックに属するかを判断することによって，ブロック単位の画素の出現頻度分布が得られる．この出現頻度分布が色ヒストグラムである．

高次元データ

検索精度を高めるためにはマルチメディアの内容をより正確に表現するためにオブジェクトの次元数は必然的に大きくなり数百次元にも達する．高次元データに対し

てはインデックスを生成しても高速検索の効果がなくなることが知られている．そこで，次元を削減する手法がよく利用される．事前にオブジェクトの集合が決定できる場合には，その高次元空間に即した次元削減を行うことができるので効果的である．しかし，実際のシステムに適用することを想定する場合には事前にオブジェクトの集合が限定できない場合も多い．たとえば，商品画像の検索を想定した場合には，商品は常に登録されたり削除されたりするので商品画像から抽出する特徴量としてのオブジェクトの分布が多次元空間上で大きく変化する可能性がある．このような状況で，事前に決定した次元削減を，分布が変化したオブジェクト集合に適用すると，検索精度が低下する可能性がある．したがって，本論文では次元削減を行わず，高次元のオブジェクトのまま扱うこととする．なお，本論文では数百次元以上を高次元，数十次元以下を低次元とし，多次元データは次元数によらないこれらの総称とする．

近傍検索

高次元のオブジェクトの検索では個々の次元要素の範囲を指定して検索することは想定できない．たとえば，類似する画像を検索する場合には画像から抽出した特徴量としてのオブジェクトを指定して類似するオブジェクトを検索対象のオブジェクト集合から検索する．また，色を指定して検索する場合でも指定された色から特徴量の一つである色ヒストグラムのオブジェクトを生成して類似するオブジェクトを検索する．類似するオブジェクトを検索することは，すなわち，オブジェクト空間上で，指定されたオブジェクトからの距離が近いオブジェクトを検索することであり，つまり，近傍検索のことである．したがって，本論文ではこの近傍検索を対象とする．

多様な距離関数への適用

近傍検索の場合には多次元空間上での距離に基づいて検索を行う．マルチメディアの内容を表す特徴量としてのオブジェクト間の距離関数は特徴量の種別により様々である．たとえば色の空間では色差を表す色差関数が利用され，色ヒストグラムの場合には 2 次形式 (quadratic form) 距離が利用されたりする．つまり，ユークリッド距離で代表される L_p 距離とは異なる．したがって，提案するインデックスは L_p 距離以外の多様な距離をも扱えることとする．

オブジェクトの追加が可能

本提案を実際に日常的に利用されるシステムに応用することを想定する．実際のシステムでは対象データが更新される場合が多い．たとえば，その応用例として商品画像の検索を想定すると，一般に商品は日々更新されるので，オブジェクトの追加

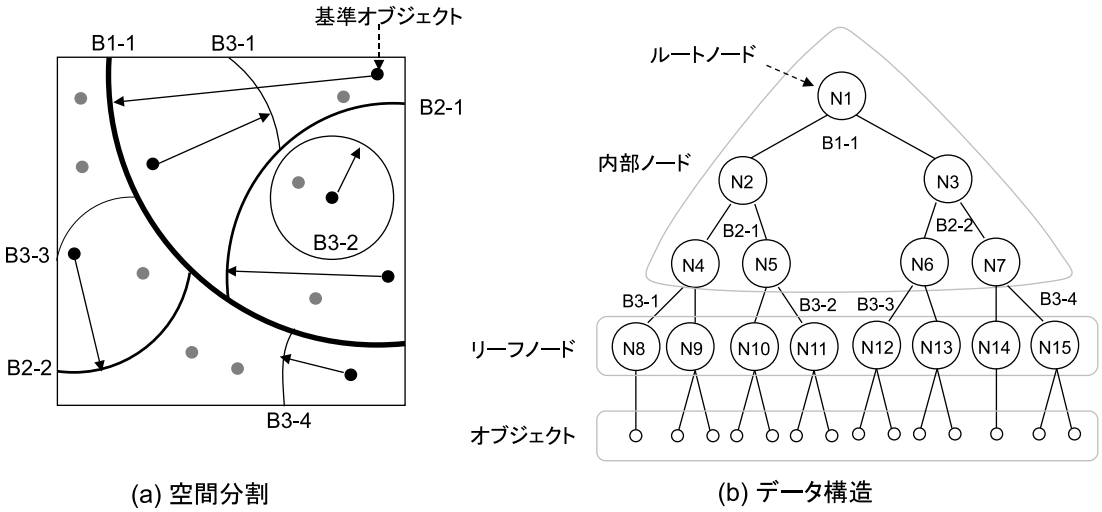


図 1.3　vp-tree の 2 次元空間における空間分割形態とそのデータ構造

更新が可能であることが必須となる．したがって，提案するインデックスではオブジェクトが追加更新できることとする．なお，オブジェクトの追加が実現できる場合には，通常，削除の実現も比較的容易に実現でき，実際に本研究では削除の実装も行っているが，本論文では削除に関しては特に論じない．

1.3　従来手法の課題と本研究の提案

多次元データの検索時の計算量を抑える空間インデックスは様々提案されている．そのようなインデックスの中でもマルチメディアデータから抽出された特徴量をオブジェクトとする場合には多様な特徴量の距離関数に適用できる空間インデックスが必要となる．多様な距離関数に適用可能となるように，距離をインデックスの基準とする空間インデックスも様々提案されている．これを本論文では距離基準空間インデックスとよぶこととする．この距離基準インデックスは一般に検索対象のオブジェクト空間を再帰的に分割することで階層構造を生成する．その多くはインデックスが木構造を構成している．vp-tree[53] を例に，その構造を図 1.3 に示す．図 1.3 (a) は 2 次元空間における分割の形態を示す．多次元データ空間を図示することは困難なので 2 次元データ空間を用いて説明する．図中の正方形はオブジェクトの存在可能範囲を表している．黒および灰色の点はオブジェクトを意味し，黒点は空間を分割する境界面を決定する基準オブジェクトを意味する．vp-tree の場合には基準オブジェクトと半径（図中の矢印）によって決定される円に

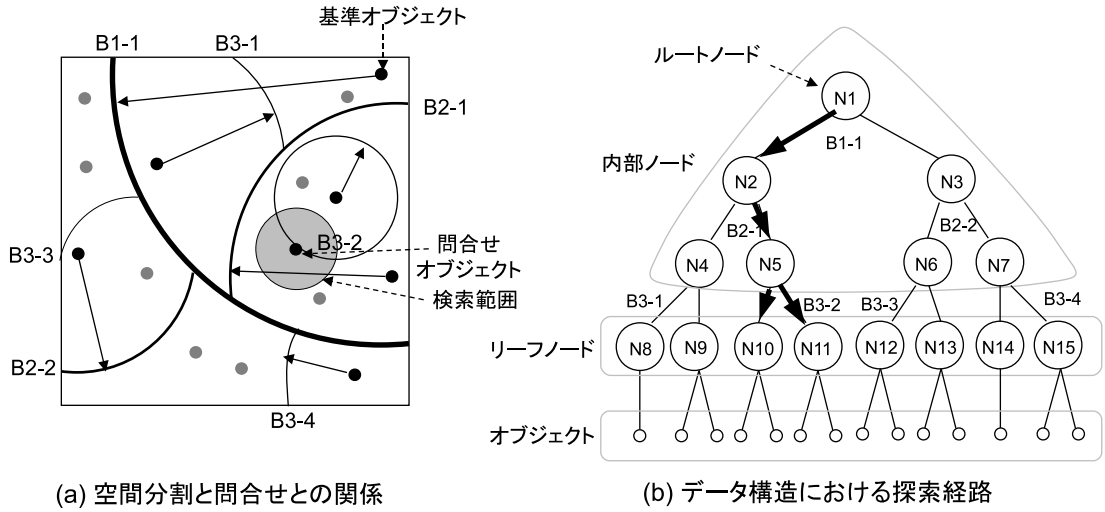


図 1.4 vp-tree の空間分割と問合せの関係，および，その木構造データにおける検索時に探索経路

よって空間を分割する．木構造型のインデックスは一般に空間を再帰的に分割することで木構造を生成する．図では空間全体を B1-1 により 2 つの空間に分割し，2 つの部分空間をそれぞれ B2-1 と B2-2 によって分割している．これを繰り返すことによって空間は細分化される．このように再帰的に分割された空間は図 1.3 (b) のような木構造のデータで表現される．木構造を構成するノードは各空間に関連付けられている．ルートノード N1 は空間全体に対応する．ルートノード（全空間）は B1-1 の円弧によって内部と外部に分割されており，内部の空間がノード N2，外部の空間がノード N3 に対応する．さらに，それぞれのノードが円弧によって分割されるので，図のような木構造が生成される．最下層のノードであるリーフノードには，その部分空間に属するオブジェクトが関連付けられている．次に，範囲検索を例に検索方法を説明する．図 1.4 (a) に空間分割と問合せの関係を示す．図中の灰色の円が検索範囲であり，検索範囲は 2 つの部分空間としか交差しないことがわかる．したがって，検索処理では木構造のルートノードから交差する部分空間に対応するノードを順次たどって行けばよい．図 1.4 (b) において矢印で示されるようにノード N1,N2,N5,N10,N11 のみをたどればよく，その他のノードを探索する必要がないので計算量を削減することが可能となる．ただし，vp-tree も含めほとんどのインデックスはインデックス生成前に検索対象のオブジェクト集合が決定していることが前提となる静的なインデックスである．しかし，実際に応用する上ではインデックスを生成した後にオブジェクトを追加する要求がある．そこで，第 3 章ではデータ構造が比較的簡単で検索量の削減効果の高い vp-tree を，更新可能な動的なインデックスに改良した dvp-tree を提案する．木構造の構成において vp-tree と dvp-tree は同様であるが，dvp-tree では

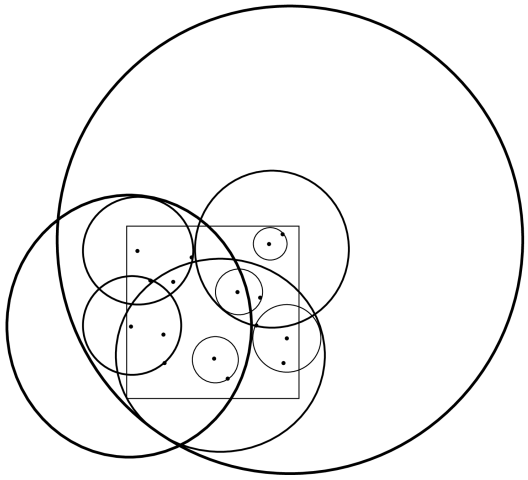


図 1.5　2 次元空間における M-tree の空間分割

オブジェクトを登録する場合に検索と同様の処理により，そのオブジェクトが属するリーフノードの部分空間を特定し，登録オブジェクトをそのリーフノードに追加する．関連付けが可能なオブジェクトの上限数をリーフノードのオブジェクト数が超えた場合には，vp-tree の生成時と同様にリーフノードの部分空間を分割し新たなノードが生成される．したがって，dvp-tree は木構造のリーフが伸長して木構造が成長する．また，インデックス生成後にオブジェクトの追加が可能な動的なインデックスとして M-tree[16] がある．M-tree の空間分割の形態を図 1.5 に示す．図のように M-tree では各部分空間が基準オブジェクトと半径によって決定付けられる超球（図では 2 次元なので円で示される）で分割される．各ノードはすべての子ノードの超球を包含する．オブジェクトを登録するには M-tree においても dvp-tree と同様に登録オブジェクトが属するリーフノードの部分空間を特定し，登録オブジェクトを，そのリーフノードに追加する．リーフノードのオブジェクト数が上限数を超えると dvp-tree と同様にノードが生成されるが，生成されたノードは木構造のルートへ伝播される．これにより，ルートノードが一階層繰り下がって木構造が生長する．つまり，繰り下がる前後において同一階層の各円は一回り大きくなることになる．部分空間が大きくなればなるほど，検索範囲と交差するノードが多くなり，結果的に木構造の探索量が増える．そこで，第 3 章では dvp-tree と同様にリーフノードが成長する登録アルゴリズムを採用した改良 M-tree も dvp-tree に加えて提案する．

これらのインデックスは数十次元の多次元データに対しては計算量の削減効果が高いが数百次元といった高次元データに対しては線形探索と同等の計算量となる傾向がある．データが高次元になるほど，任意の 2 つのオブジェクト間の平均距離が大きくなり，か

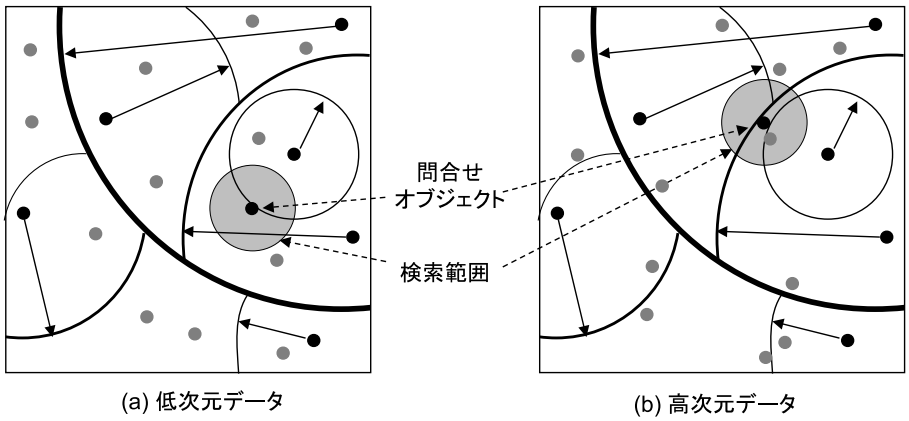


図 1.6　dvp-tree における高次元データの影響

つ，特定の距離に集中する傾向がある．この傾向がインデックスに様々な影響を与える．dvp-tree における顕著な影響の例を図 1.6 に示す．図の (a) は低次元データの例であり，オブジェクトが均等に分散している．図の (b) は高次元データの例である．高次元データを 2 次元平面に表現することはできないので，高次元の傾向を概念的に図示していると捉えて頂きたい．任意の 2 つのオブジェクト間の距離が特定の距離に集中するので，超球の半径もその特定の距離に近くなり，すべてのオブジェクトが超球の近辺に集中しているかのような状態になる．さらには，問合せオブジェクト自体も境界近辺となる可能性が高くなるので，問合せオブジェクトの検索範囲が常に複数の部分空間と交差することになる．したがって，木構造を探索するときに多くのノードを探索することになり，線形探索と同等の計算量となってしまう．一方，高次元データに対して計算量の大幅な削減が可能となる近似検索のインデックスが提案されている．近似検索とは，検索時に本来検索されるべきオブジェクトが検索されない，つまり，検索漏れが生じる検索である．しかし，画像から抽出される特徴量をオブジェクトとして用いた類似画像検索では，その画像特徴量の曖昧性により検索漏れが顕在化しない．また，高次元データの場合には画像特徴量に限らず実用上では問題とならない場合が多いと考え，第 4 章以降はこの近似検索に着目する．

近似検索インデックスの提案はそれほど多くないが，その中で特に検索時の計算量を削減できるインデックスとして k 最近傍グラフを用いる手法 [45], [24] がある．図 1.7 に k 最近傍グラフの例を示す．図中の黒点で示されるグラフのノードがオブジェクトを意味する．k 最近傍グラフは有向エッジ（単方向接続）によって各ノードの k 個の最近傍のノードに関連付けられている．有向エッジによって接続されているグラフを有向グラフという．図の多くのエッジは双方向の矢印となっているが，一部のエッジが単方向の矢印と

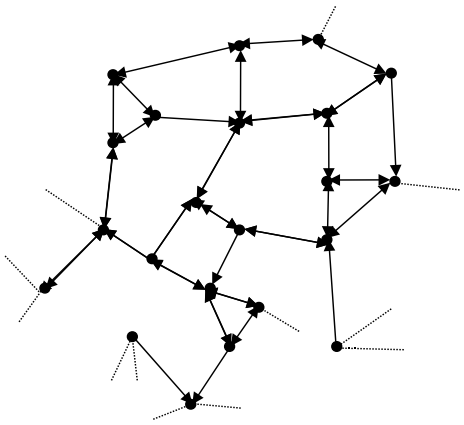


図 1.7 k 最近傍グラフの構造

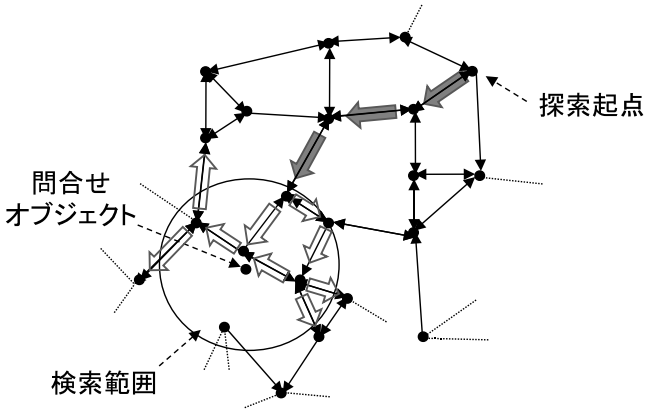


図 1.8 k 最近傍グラフ上での範囲検索時の探索経路

なっていることから有向グラフであることがわかる．図 1.8 に k 最近傍グラフを用いた検索方法を示す．k 最近傍グラフの検索では次の 2 段階の処理を行う．

1. 任意のノードから検索範囲までのノードの探索
2. 検索範囲内での網羅的なノードの探索

第 1 段としてグラフ上の任意のノードを探索起点とし，各ノードにおいて指定された問合せに最も近いノードを k 個の近傍ノードから選択する．これを繰り返すことにより，問合せオブジェクトに順次近付く（図中の灰色矢印）ことができる．検索範囲内に到達したら，第 2 段として，検索範囲内に含まれるノードを網羅的にすべて探索し（図中の白色矢印），検索範囲内のノードを検索結果として獲得する．ただし，検索範囲内に存在するノードでも，検索範囲内で接続されていない場合にはたどることができずに，検索漏れと

なる．したがって k 最近傍グラフを用いる手法では検索漏れが発生する近似検索となる．その反面，k 最近傍検索では高次元データであっても計算量を大幅に抑制できること [45], [24] が知られている．しかし，k 最近傍グラフには以下の 2 つの課題がある．

- k 最近傍グラフ生成の膨大な計算量
- 非連結グラフに起因する検索精度の抑制

そこで，第 4 章ではこれらの問題を軽減する近似 k 最近傍グラフを提案する．

まず，第 1 の課題に関して説明する．総当りで距離を計算し k 最近傍グラフを作成する場合には $O(n^2)$ の計算量が必要となる．計算量を削減する手法も幾つか提案 [12],[15],[38] されているが，劇的に計算量を減らすことはできない．k 最近傍グラフを正確に生成するにはこのように多大な計算量を必要とするが，検索のインデックスとして前述の検索処理を行う上では各ノードが正確に k 個の最近傍ノードに接続している必要性はない．つまり，k 最近傍グラフを近似したグラフを生成することで生成時の検索量を削減できると考えられる．グラフに新規ノードを追加するときに新規ノードに対して k 個の最近傍のノードをグラフ中のノードから検索する処理に対して多大な計算量が発生する．そこで，提案手法である近似 k 最近傍グラフでは，生成途中のグラフを用いて前述の検索を行うことで計算量を大幅に削減が可能である．つまり，生成途中のグラフを用いて新規ノードに対する k 個の最近傍ノードを検索し，検索された k 個の最近傍ノードにその新規ノードをエッジで接続する．これを繰り返すことでインデックスを生成でき，かつ，計算量を大幅に抑制できる．この生成処理はオブジェクトの追加に他ならないので，インデックス生成後にもオブジェクトの追加が可能な動的なインデックスの性質も同時に兼ね備えることになる．

次に，第 2 の課題に関して説明する．連結グラフとはグラフ上の任意の 2 つのノード間の経路が必ず存在するグラフである．k 最近傍グラフは非連結グラフ，つまり，いずれかの 2 つのノード間には経路が存在しない場合があるグラフである．つまり，グラフが 2 つ以上の独立したグラフに分離していることを意味する．前述のように検索処理ではエッジをたどるので，検索範囲内にノードが存在していたとしても経路が存在しない場合には，そのノードに到達することはない．したがって，検索漏れを引き起こし検索精度が低下する．このことから連結グラフを生成することが重要である．インデックス生成において新規ノードをグラフ上の k 個の最近傍ノードへ接続するときに k 最近傍グラフと同様に有向エッジで接続すると新規ノードからグラフへ接続されるが，グラフから新規ノードへは接続されないので，非連結グラフが生成されてしまう．そこで，提案手法である近似 k 最近傍グラフでは，新規ノードをグラフへ接続するときに無向グラフで接続することにより

表 1.1 k 最近傍グラフと近似 k 最近傍グラフの比較

	k 最近近傍グラフ	近似 k 最近傍グラフ
グラフ形態	非連結グラフ	連結グラフ
エッジ種別	有向グラフ	無向グラフ
総有向エッジ数（n:総ノード数）	kn	$2kn$

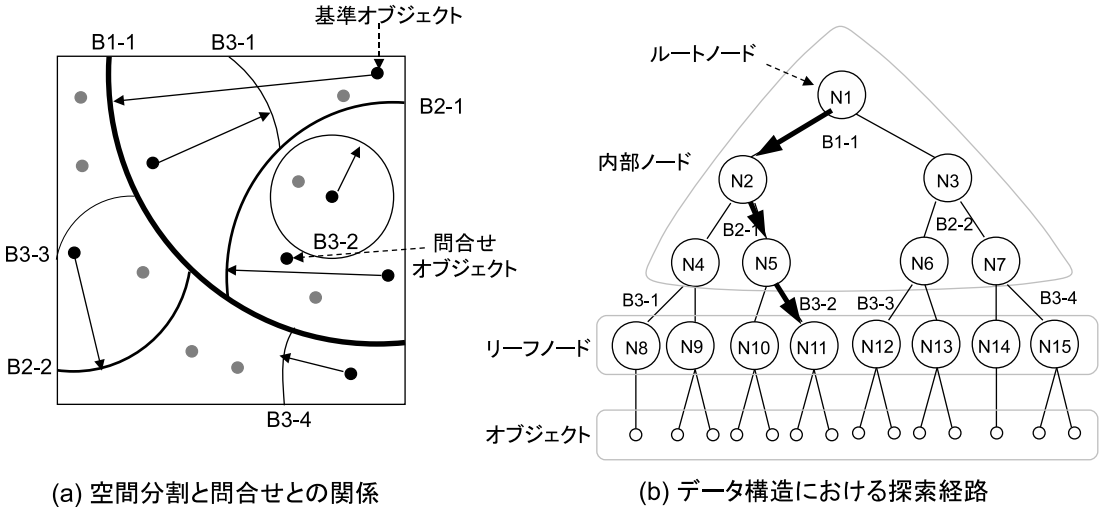


図 1.9 探索起点検索時における vp-tree の空間分割と問合せの関係，および，その木構造データにおける検索時の探索経路

グラフから新規ノードへも接続されて連結グラフが保証される．このようにして生成された近似 k 最近傍グラフと k 最近傍グラフの相違点を表 1.1 に示す．表中の近似 k 最近傍グラフの総有向エッジ数とは 1 本の無向エッジを 2 本の有向エッジとみなした場合のエッジ数を意味する．

提案手法である近似 k 最近傍グラフはグラフ生成時の計算量の低減，検索精度の低下の抑制が可能である．しかし，グラフ中の任意のノードをグラフの探索起点とするので，問合せオブジェクトと探索起点との距離が遠い場合には検索時の第 1 段の処理の計算量が増大する．そこで，第 5 章では検索時の第 1 段の処理をグラフの代わりに第 3 章で提案する dvp-tree を用いる方法を提案する．第 1 段の処理ではグラフ上の任意のノードを探索起点とするので，検索範囲から遠方のノードが選択された場合には各ノードを逐次たどって検索範囲を探索するよりも，dvp-tree を用いて問合せオブジェクトの近傍ノードを検

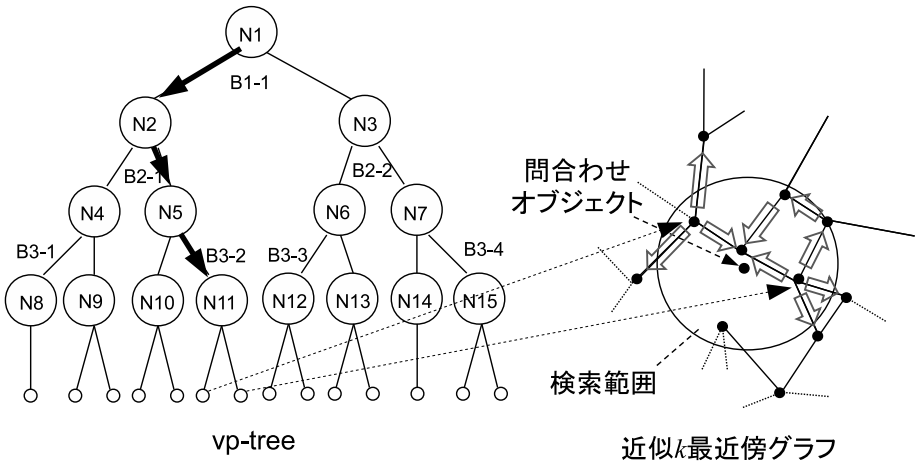


図 1.10 dvp-tree と k 最近傍グラフの関係と検索処理の課程

索する方が計算量が少ない．特にオブジェクトの登録数が多くなるほどその差は広がる．dvp-tree を用いて検索するノードはグラフの探索起点としてのみに利用するので，問合せオブジェクトに対する最近傍ノードである必要はなく，近傍候補ノードであればよい．dvp-tree を高次元データに適用する場合には線形探索と同様の計算量となることをすでに述べたが，問合せオブジェクトの近傍候補ノードを検索する場合には，この限りではない．図 1.9 (a) に近傍候補ノードを検索する場合の問合せオブジェクトと dvp-tree の各空間との関係を示す．近傍候補ノードを検索する場合には検索範囲がなく，単なる点である．したがって複数の部分空間と交差することはない．図 1.9 (b) に，この場合の木構造の探索経路を示す．複数の空間と交差することがないので，図の (b) のようにルートからリーフへ分岐することなくたどることが可能である．つまり，高次元データであっても計算量が増加することはない．しかも，問合せオブジェクトが空間上のどの位置であってもほぼ一定の計算量で近傍候補ノードを取得できる．近傍候補ノードを探索起点としてグラフの探索を開始できるので，任意のノードを探索起点とする場合よりも第 1 段の処理の計算量を削減することが可能である．なお，M-tree では問合せオブジェクトの検索範囲がなくても部分空間自体が交差しているので複数の部分空間をたどる必要があり計算量が増大する．したがって M-tree は本用途には利用できない．図 1.10 に dvp-tree と近似 k 最近傍グラフのデータ構造の関係と検索の課程を示す．dvp-tree のルートノードからリーフノードへたどると複数のオブジェクトを近傍候補として取得できる．dvp-tree のリーフノードに属するオブジェクトは近似 k 最近傍グラフの各ノードに関連付けられているのでリーフノードのオブジェクトを探索起点としてグラフの探索を開始できる．リーフノードの複数のオブジェクトはそれぞれ探索起点となる．

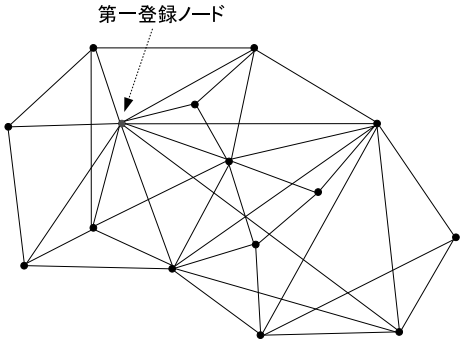
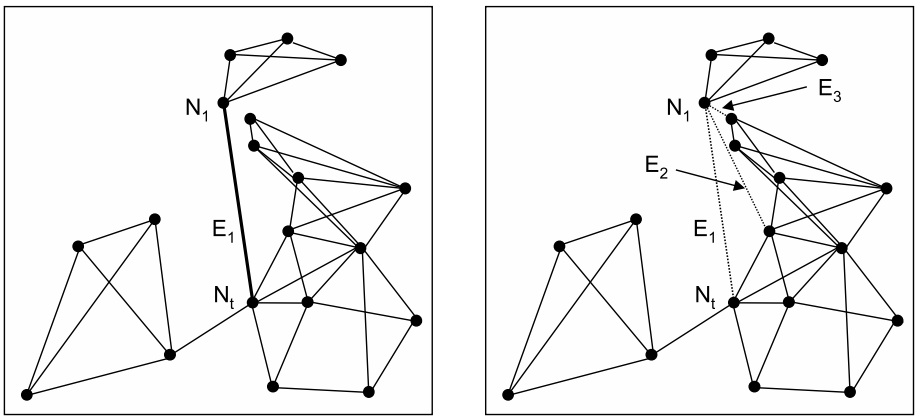


図 1.11 $k = 3$ における過剰なエッジの例

第 5 章で提案する dvp-tree と近似 k 最近傍グラフを用いた手法により大量のオブジェクトを登録しても検索時の第 1 段の処理時の計算量を削減できる．しかし，大量のオブジェクトを登録すると初期ノードにエッジが大量に付与される傾向がある．その例を図 1.11 に示す．エッジ数 $k = 3$ であるが，登録のアルゴリズムの特性上エッジは 3 以上付与されることになる．特に初回に登録されたオブジェクトには 10 本のエッジが付与されている．この例では 15 オブジェクトしか登録していない状況にもかかわらずエッジの偏りが激しい．また，特徴量をオブジェクトとする場合には特徴量の性質によって空間の特定の場所にオブジェクトが集中する場合があり，これにより，特定のノードにエッジが集中することになる．特定のノードの過剰なエッジの多くは近傍のノードに接続されていないことになり，検索処理において無用である上に次のような課題が生じる．

- メモリ領域の浪費
- ノード格納領域の増大による実装機構の複雑化

第 1 に，無用な大量のエッジを格納するためにメモリ領域を大量に確保することとなる．第 2 に，エッジ数が増大しなければ固定長レコードにノードを格納するといった簡便なデータ構造が可能となり，検索時の速度性能の向上が期待できる．しかし，エッジ数が増大することで可変長レコードを構成する必要があり，速度性能が抑制されるといった問題が生じる．そこで，第 6 章では過剰なエッジの削減方法を提案する．しかし，単純に過剰エッジを削減すると連結グラフを維持できなくなる．そこで，削減により連結グラフを維持できなくなるエッジに対してはエッジを別のノードに再配置することで過剰なエッジを有するノードのエッジ数を削減する．図 1.12 (a) に過剰なエッジを有するノードの例を示す．N_t をエッジが過剰なノード，E_1 を削減対象のエッジとすると，E_1 を削除するとグラフが分断されるので他のノードへ再配置することで N_t のエッジを削減する．簡便な



(a) 再配置前　　(b) 再配置後

図 1.12　過剰なエッジの再配置の例

方法として N_t が接続する近傍ノードへ過剰なノードを再配置する方法が考えられる．図 1.12 (b) において E_1 を E_2 に移動する場合である．この場合に，さらにオブジェクトが大量に登録されて N_t のエッジが増えたとする．その結果，同様にエッジの削減処理が繰り返し行われ，今度は N_t の近傍ノードのエッジが増加する．次に，その近傍ノードに対してエッジの削減処理を行うことになり，その際に N_t へエッジが戻ってしまう現象が生じる．これを回避するために，削減対象のノード N_t からより遠く，かつ，過剰エッジが接続している先のノード N_1 に近いノードへ再配置する．つまり，図 1.12 (b) において E_1 を E_3 に配置する．このようにすることで，削除した過剰なエッジが復元することがなくなる．

第 3 章から第 6 章までにおいて，高次元データの検索時の計算量を抑えつつ高精度を実現することが可能であり，かつ，インデックス生成時の計算量を低減できる近似 k 最近傍グラフと dvp-tree を組み合わせた手法を提案する．以上の本論文の提案の流れを図 1.13 に示す．さらに，第 7 章では本手法を実際の商品画像へ適用する事例を示す．

1.4　本研究の寄与

本研究の発端としては画像，映像といったマルチメディアデータから生成された特徴量としての高次元データを少ない計算量で検索することを目的としている．しかし，本提案は特定の距離関数に依存しないことからマルチメディアデータの特徴量に限定されることはなく，様々な多次元データに適用が可能である．たとえば，長年研究されているテキストの類似検索に応用することも可能である．テキストデータにベクトル空間モデルを適用

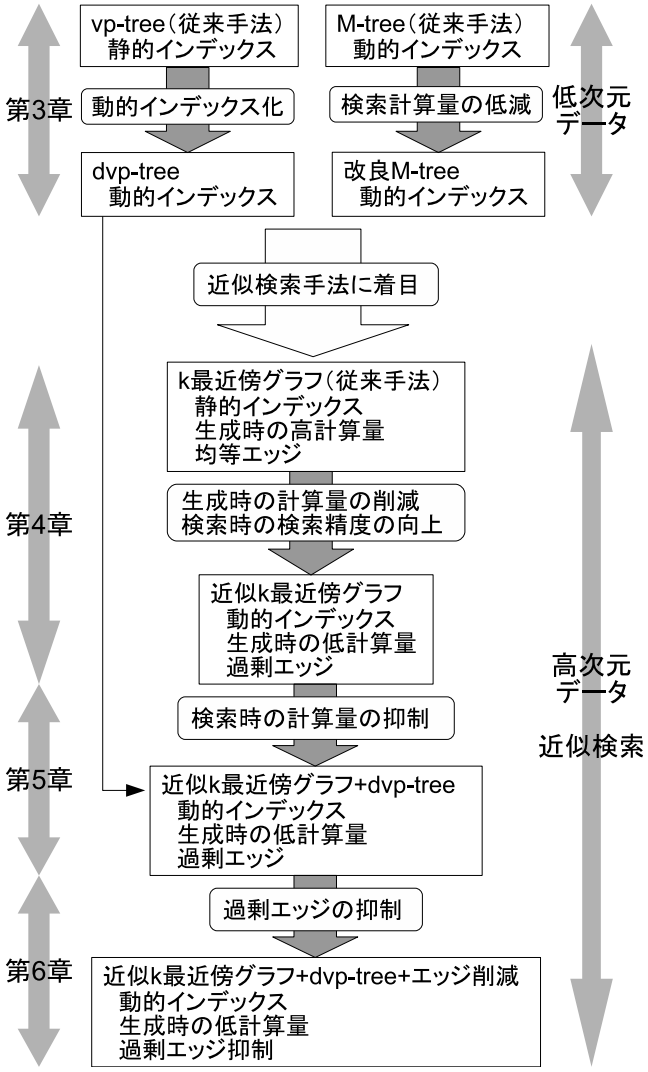


図 1.13　提案の流れ

することで多次元データへ変換することができるので，本提案が応用できる．一方，近年，インターネット上で日々蓄積されるログデータといった極めて大規模なデータ（ビッグデータ）から有益な情報を抽出する技術が注目されている．また，すべての物をインターネットへ接続しようと試みる技術（Internet of Things）も注目されている．これらの技術では，多種多様なデータを大量に解析する技術が重要となる．多種多様なデータは多次元データとして扱うことができるので，本提案が応用できると考える．このように本提案はマルチメディアデータのみならず，幅広く応用できるだけでなく，世の中の情報が増えるにつれ，その重要性は増していく予想している．

1.5　本論文の構成

本節では論文構成について述べる．まず，第 2 章では近傍検索の定義について述べた後，各種インデックスに関する研究，および，提案するインデックスの応用を想定し属性データと特徴量の AND 検索に関する研究について述べる．

第 3 章から第 6 章までは研究の段階を追って提案するインデックスに関して述べる．第 3 章では追加登録ができない既存のインデックスを改良して更新可能な木構造型インデックスを提案する．数十次元程度の低次元データにおいては線形探索と比較して提案する木構造型インデックスは計算量を削減できるが，数百次元といった高次元データではその効果が抑制される．そこで，第 4 章では検索漏れを許容することで検索時の計算量を削減できる近似検索インデックスに着目し，グラフ構造型インデックスを提案する．グラフ構造型インデックスは高次元データにおいてもインデックスの生成時及び検索時の計算量の削減に効果的であるが，グラフの探索を開始するノードを任意に選択するのでオブジェクトの登録数が増加した場合に，検索オブジェクトから遠いノードが偶然にも選択されると検索時の計算量が増加する．第 5 章では第 3 章の木構造型インデックスを用いて検索オブジェクトの近傍ノードを取得し，そのノードから探索を開始する手法を提案する．グラフ構造型インデックスに木構造型インデックスを組み合わせることにより登録数が増加しても検索時の計算量を抑制することが可能となるが，その一方で，初期登録ノードや特定のノードにエッジが大量に付与される傾向がある．第 6 章ではこのような過剰なエッジの削減方法を提案する．第 7 章では，これらの提案手法を実際の商品画像検索に適用する事例を示す．実際の商品画像検索に適用する上では商品分類を考慮した AND 検索が必須となるので，その AND 検索の方法も同時に提案する．最後に，第 8 章にて本研究を総括した後に本論文の寄与および今後の展開を述べる．

なお，各章と公開文献との対応は以下のとおりである．

- 第 3 章: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 [63]
- 第 4 章:近似 k 最近傍グラフによる距離空間の近傍検索 [64]
- 第 5 章:木構造型インデックスを用いた近似 k 最近傍グラフによる近傍検索 [65]
- 第 6 章:商品画像検索へのグラフ構造型インデックスの適用 [66]
- 第 7 章:商品画像検索へのグラフ構造型インデックスの適用 [66]

第 2 章

関連研究

2.1 本章の目的

大量のマルチメディアデータを高速に検索するには，マルチメディアデータから抽出した大量のオブジェクトを高速に近傍検索する必要がある．そこで，本章では近傍検索の定義および低次元データの近傍検索を実現するインデックスの関連研究に関して述べた後，高次元データに有効とされる近似近傍検索のインデックスの関連研究を紹介する．最後に提案するインデックスを実際の検索システムに応用する場合を想定し属性データとオブジェクトとの AND 検索に関する研究を紹介する．

2.2 近傍検索

マルチメディアデータを I，特徴量抽出関数を $e(I)$，特徴量のオブジェクトを $\boldsymbol{x} = \{x_1, x_2, x_3, ..., x_n\}$ とすると $\boldsymbol{x} = e(I)$ であり，マルチメディアデータ I_a, I_b 間の距離（類似度）は $d(e(I_a), e(I_b))$ と表される．たとえば，類似画像検索は指定した問合せ画像に類似する画像を画像集合から検索する．つまり，指定画像のオブジェクトに近いオブジェクトを多次元空間上から検索することであり，これを近傍検索とよぶ．近傍検索の基準となる距離関数としてよく利用されるのはユークリッド距離である．ユークリッド距離を含む L_p 距離の定義を以下に示す．

$$l_p(\boldsymbol{x}, \boldsymbol{y}) = \{\sum_i (x_i - y_i)^p\}^{1/p} \tag{2.1}$$

この距離は $p = 2$ のときにはユークリッド距離となり，$p = 1$ のときには市街地距離となる．しかし，マルチメディアデータから抽出される特徴量の距離関数としては適切ではな

い場合がある．マルチメディアデータから抽出される特徴量として色の出現頻度を表す色ヒストグラムがよく利用される．2 つの色ヒストグラムを比較する場合に L_p 距離を使うと各次元，つまり，各色の類似性を独立して比較することになる．しかし，たとえば，赤とオレンジはある程度類似しているが，この 2 つの色が別々の色ヒストグラムの要素に割り当てられると，つまり，異なる次元要素として扱われると，L_p 距離ではこの類似性が考慮されない．つまり，色ヒストグラムでは，各次元要素間の相関を考慮する必要があり，以下に示す 2 次形式 (quadratic form) 距離が提案 [27] されている．

$$d_q(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{A} (\boldsymbol{x} - \boldsymbol{y}) \tag{2.2}$$

なお，$\boldsymbol{A}$ は相関行列である．このようにマルチメディアデータから抽出した特徴量の場合には多様な距離関数が利用される．この距離関数を用いて近傍検索では以下に示す検索の種類がある．

- 範囲検索 (range query)
  検索範囲を示す円の中心オブジェクト（問合せオブジェクト）$\boldsymbol{o}_q$ および半径 r_q を指定し，$d(\boldsymbol{o}_q, \boldsymbol{x}) \leq r_q$ を満足するオブジェクト $\boldsymbol{x}$ の集合を求める．
- k 最近傍検索 (k-nearest neighbor query)
  検索の中心オブジェクト（問合せオブジェクト）$\boldsymbol{o}_q$ および検索件数 k を指定して $\boldsymbol{o}_q$ との距離 $d(\boldsymbol{o}_q, \boldsymbol{x})$ の昇順で上位 k 件のオブジェクト $\boldsymbol{x}$ の集合を求める．
- 最近傍検索 (nearest neighbor query)
  検索の中心オブジェクト（問合せオブジェクト）$\boldsymbol{o}_q$ を指定して $\boldsymbol{o}_q$ との距離 $d(\boldsymbol{o}_q, \boldsymbol{x})$ が最も小さいオブジェクト $\boldsymbol{x}$ を求める．

マルチメディアデータから抽出した多次元の特徴量のオブジェクトを近傍検索するには，問合せオブジェクトと検索対象となるすべてのオブジェクトとの距離を算出する線形探索を行えばよいが，検索対象のオブジェクト数に比例して検索時間が増加するので，実用的な時間では検索できない．そこで，実用的な時間で検索するには特徴量空間を検索する空間インデックスが必須となる．空間インデックスには大きく分けて非距離基準空間インデックスと距離基準空間インデックスの 2 種類がある．次に，この 2 種類のインデックスに関連する研究について述べる．

2.2.1 非距離基準空間インデックス

本論文では距離に基づかない空間インデックスを非距離基準空間インデックスとよぶこととする．非距離基準空間インデックスでは，データマイニングといった応用を背景に

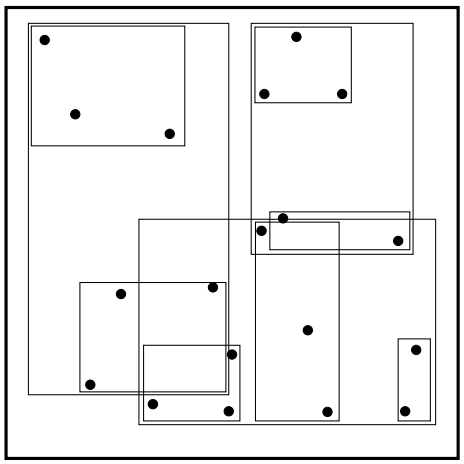


図 2.1 2 次元空間における R-tree の空間分割

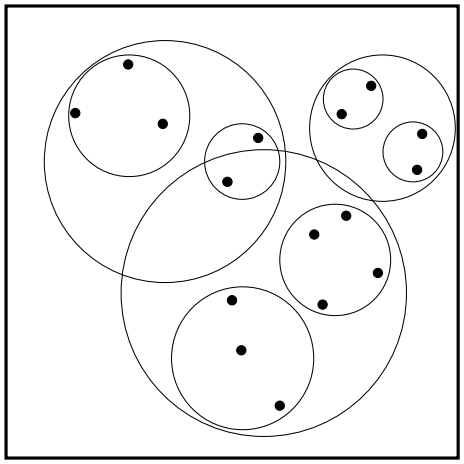


図 2.2 2 次元空間における SS-tree の空間分割

様々なインデックスが研究されている．木構造型のインデックスの場合には多次元空間を再帰的に分割することで木構造を生成しインデックスとする．R-tree[23] では木構造の各ノードの領域が矩形で定義される．図 2.1 は 2 次元データにおける R-tree の空間の分割の形態を表している．外枠の正方形は各次元の取り得る範囲を表し，黒点はオブジェクトを意味する．矩形は木構造の各ノードに割り当てられた領域である．各ノードの領域は子ノードを包含する領域としているので，木構造のルートノードからリーフノードへたどるに従い領域が狭まる．リーフノードには検索対象のオブジェクトが関連付けられている．検索時には検索範囲と重複するノードをルートノードから順次たどり，リーフノードのオブジェクトを検索結果とする．検索範囲と重複しないノードは探索する必要がないの

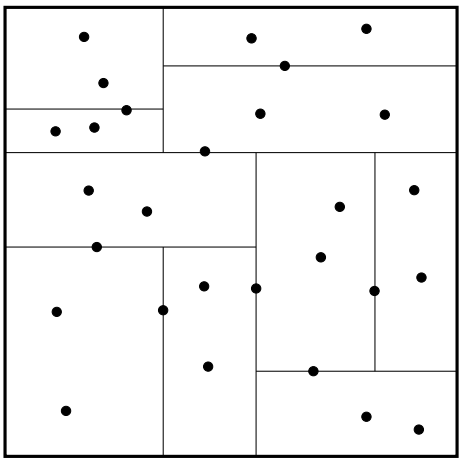


図 2.3　2 次元空間における kd-tree の空間分割

で，検索を高速に実行できる．類似するインデックスとして R-tree の検索や登録のアルゴリズムを改良した R*-tree[6] が提案されている．R-tree や R*-tree では兄弟ノードの領域が重複しやすい傾向があり，それが検索速度を抑制する．そこで，領域の重複を削減することで検索速度を改善する X-tree[8] が提案されている．SS-tree[51]（図 2.2）はノードの領域を矩形ではなく球で定義する．SS-tree の球をより小さくするように改良した SS+-tree[30] が提案されている．さらに，SR-tree[29] では矩形と球を組み合わせて領域を定義している．

木構造型インデックスでは兄弟に位置するノードの領域が重複する傾向があるが，重複がないインデックスがある．kd-tree[7]（図 2.3）では，単一の次元軸を特定の値で二分する．したがって，その分割面はその分割した次元軸と直行する平面となる．kd-tree では領域が重複しないので検索速度の改善に有利である．quadtree[44] は kd-tree に似ているが，各領域を 2 つの領域ではなく 4 つの領域に分割する．木構造ではないインデックスもいくつか提案されており，代表的なものとして VA-File[50] がある．この手法では次元ごとにオブジェクトを圧縮することで線形探索を高速化する．

これらの非距離基準空間インデックスは様々なデータに対して応用されているが，マルチメディア検索に応用する場合には，マルチメディアデータから抽出される特徴量の多様性に起因する問題が生じる．検索精度を重視すると，色ヒストグラムの特徴量の距離には 2 次形式 (quadratic form) 距離，色の特徴量には色差式に基づく距離，そしてテクスチャの特徴量には L_1 距離，といったように様々な距離が利用される．非距離基準空間インデックスは一般に L_p 距離を対象としているので，このような多様な特徴量に対して単純には非距離基準空間インデックスを適応できない．そこで，平均色による非距離基準空

間インデックスによって検索した後に誤検索を除去する方法 [18] や 2 段階の処理により 2 次形式距離に対応したインデックスを生成する方法 [46] などがある．しかし，処理が複雑になることで計算量が増加する問題が生じる．

2.2.2 距離基準空間インデックス

以下の距離定義を満足する距離関数が定義されている多次元データに対して適用できる空間インデックスがある．本論文ではそれを距離基準空間インデックスとよぶこととする．

1. $d(\boldsymbol{x}, \boldsymbol{x}) = 0$
2. $d(\boldsymbol{x}, \boldsymbol{y}) > 0 \ (\boldsymbol{x} \neq \boldsymbol{x})$
3. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ （対称性）
4. $d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$ （三角不等式）

距離基準空間インデックスは距離関数の種別を問わず適応できるという利点により，様々なインデックスが提案されている．距離基準空間インデックスには非距離基準空間インデックスと同じく多様な木構造型のインデックスがある．木構造型の非距離基準空間インデックスと同様に空間を順次分割することで木構造を形成し，どのように空間を分割するかにより各インデックスが特徴付けられている．ただし，距離基準空間インデックスでは空間の分割時に距離のみにより分割する．

gh-tree[48](図 2.4) では generalized hyperplane により空間を分割する．個々のノードは 2 つの基準オブジェクトを持ち，2 つのオブジェクトから等距離にある線を generalized hyperplane とよび，これにより空間を分割する．ただし二分木である gh-tree の場合には木構造が深くなる傾向があり，その結果，検索時の距離計算回数が多くなり検索速度の低下を引き起す．距離基準空間インデックスの検索処理に占める時間は距離計算時間が大きな割合を占める．特に quadratic form 距離のように相関を考慮した距離の場合には増大する．したがって，距離計算の回数（距離計算回数）を減らすことが大きな課題となる．なお，本論文で計算量は距離計算回数を意味する．gh-tree の類似手法である bisector tree[28] や mb-tree[35] では超球も用いて領域をより限定している．mb-tree では親の基準オブジェクトを継承することで超球の拡大を抑制している．

GNAT[10](図 2.5) では，基準オブジェクトを複数設定し，generalized hyperplane により空間を分割する．結果として空間はボロノイ分割となる．各オブジェクトは最も近接する基準オブジェクトに属するように割り振られる．検索時に検索範囲が複数のノード空

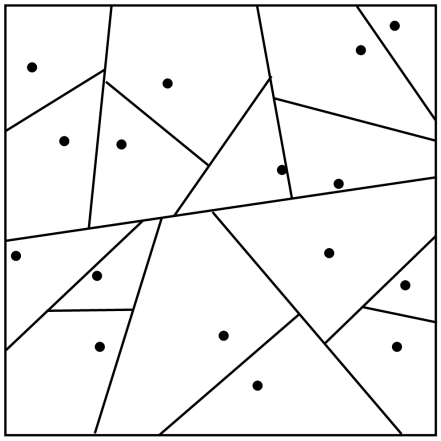


図 2.4　2 次元空間における gh-tree の空間分割

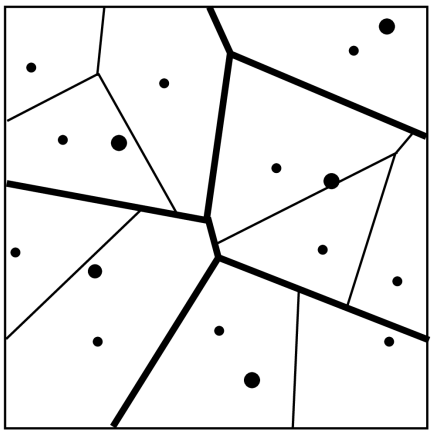


図 2.5　2 次元空間における GNAT の空間分割

間のいずれに属するかを判断する上ですべての基準オブジェクトとの距離を求める必要があり検索速度の低下を招く．そこで GNAT ではすべての 2 つの子ノードの組合せにおいて，子ノードの基準オブジェクトと，もう一方の子ノードに属するオブジェクトの最小最大距離をあらかじめ保持する．その距離から三角不等式に基づき検索範囲が属さない子ノードを距離計算なしに判断することができる．こうして距離計算回数の削減を実現している．ただし，すべての子ノード間の距離を保持するのでノードのデータ量が増加するだけでなく，データ構造が複雑であるという問題点がある．

vp-tree[53](図 2.6) では各ノードの空間は一つの基準オブジェクト（vantage point）と超球（実際には円ではないが説明上図中では円で表わす）によって順次分割される．検索

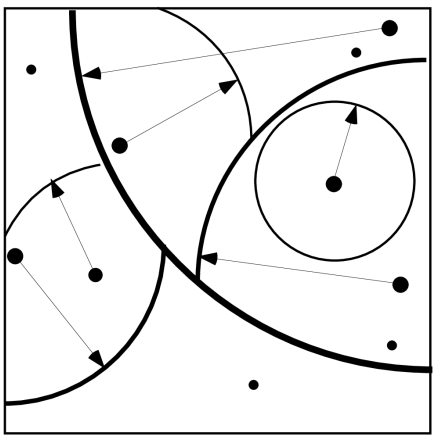


図 2.6 2 次元空間における vp-tree の空間分割

時にはルートノードからたどり，検索範囲が基準オブジェクトと半径によって分割されている領域のいずれに属するかを判断し，たどる子ノードを決定する．これを繰り返して検索範囲に適合するリーフを探す．vp-tree では一つの基準オブジェクトに対し複数の同心円を設定し多分木とすることで，基準オブジェクトからの距離を一回計算するだけで複数のノード領域のいずれに属するかを判断でき，距離計算回数を減らすことができる．しかし，多次元データの場合には空間を 1 つの基準オブジェクトのみで数多く分割しても各領域は極めて細いリング状になり，検索時にすべての分割領域が検索範囲と交差してしまい，分割の効果が薄れてしまう傾向がある．mvp-tree[9] ではデータ構成は複雑となるが基準オブジェクトを複数設定して複数の基準オブジェクトにより空間を分割することで，分割円が細いリング状になることを防ぎ，かつ，多分木を構築することで検索の距離計算回数を削減している．他にも超球で分割する類似手法がいくつか提案されている．bk-tree[11] も vp-tree 同様に複数の同心円で空間を分割するが，vp-tree が要素を同数になるように分割するのに対して bk-tree は同心円で形成されるリングの幅が等しくなるように分割する．fq-tree[4] は bk-tree と類似しているが，木構造上の同一階層のすべての空間は同一のオブジェクトを基準に分割される点が vp-tree とは異なる．vp-forest[54] は超球の近辺に存在するオブジェクトを除いて木構造 (vp-tree) を生成し，除いたオブジェクトを用いて再度木構造を生成し，これを繰り返すことで，複数の木構造を生成する．範囲検索時に複数の木構造をたどる必要があるが，超球の近辺のオブジェクトが存在しないので，木構造を分岐してたどる必要がなく距離計算回数を削減できる．ただし，範囲検索の範囲が極めて狭い場合にしか有効ではない．

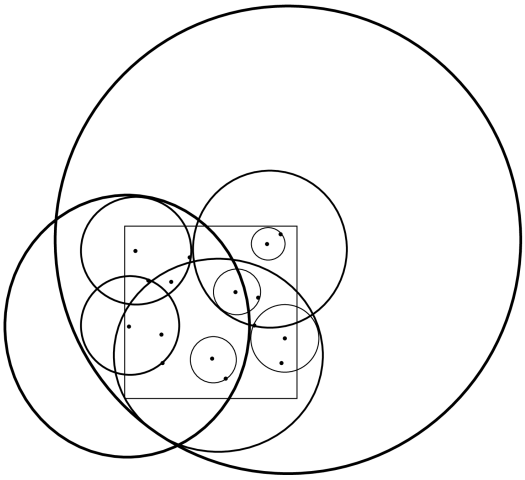


図 2.7　2 次元空間における M-tree の空間分割

さらに，空間を分割することなく，事前に各オブジェクト間の距離を算出し，それをインデックスとして用いる手法が提案されている．AESA[41] ではすべての 2 つのオブジェクト間の距離を算出し，LAESA[33] では特定の複数オブジェクトからその他のすべてのオブジェクトへの距離を算出して保持しておく．検索時には事前に算出した距離を用いてオブジェクトの絞込みを行うので，距離計算回数の削減の効果は高い．しかし，登録データ数に対して指数級数的に距離データであるインデックスが増大するので大量のオブジェクトへの適用には向かない．

以上，述べた距離基準空間インデックスはすべて静的にインデックスを構築するアルゴリズムである．つまり，事前に検索対象のすべてのオブジェクトを用意した上でインデックスを構築し，構築後にはオブジェクトを追加登録したり削除したりといった操作ができない．一方，M-tree[16](図 2.7) は動的にインデックスを生成できる．つまり，インデックスの生成後もオブジェクトの登録が可能である．さらに，M-tree ではオブジェクトを登録してもバランスのとれた木構造を保つことができる点が特徴である．バランスがとれた木構造とはルートノードからリーフノードまでのノード数が均一であることを意味する．バランスがとれていない場合には問合せオブジェクトの位置により検索時の距離計算回数が増減することになる．M-tree の各ノードは基準オブジェクトと半径によって形成され各ノードは複数の子ノードを有する．図 2.7 のようにノードの円は子ノードの円を完全に包含し，ノードの円は必ずいずれかの子ノードの円に接する．M-tree では各ノードが基準オブジェクトと，ノードに属するすべての子ノードの円を包含する円の半径を保持する．さらに各ノードは基準オブジェクトと親ノードの基準オブジェクト間の距離をも

つ．GNAT と同様にこの距離により三角不等式に基づいて検索時に距離計算回数を削減している．しかし，図からもわかるように M-tree の部分空間である円自体が互いに交差する．したがって，部分空間が交差しない他のインデックスに比べ検索範囲が複数の部分空間と交差する可能性が高くなり距離計算回数が増加する傾向がある．

以上の距離基準空間インデックスはどのような距離関数にもインデックスのアルゴリズムを変更せずにそのまま適用できる点で有利である．しかし，高次元データの場合には非距離基準空間インデックス，距離基準空間インデックスともに効果が極端に抑制されて，線形探索と同等になる．そこで，次に高次元データでも計算量を抑制できる近似検索に関する研究について述べる．

2.3 近似検索

非距離基準空間インデックスでは，次元数が多くなるほどインデックスの効果が減少する．オブジェクトの分布の状態にもよるが，100 次元にもなるとインデックスの効果がまったくなくなり線形探索と同等の距離計算回数になる．そこで，検索結果に漏れが生じることを許容して高速に検索する近似検索に関する提案がなされている．非距離基準空間インデックスでは，kd-tree に近似検索を適用した ANN[1] や，ハッシュを用いた LSH[22] といった近似検索の手法が提案されている．

同様に距離基準空間インデックスであってもデータの次元数が多くなるほどインデックスの効果が減少する．距離基準空間インデックスの場合には距離のみに基づきインデックスを生成し，検索するので，次元ごとの値を参照しないアルゴリズムとなっている．もちろん，距離計算時には各次元の値を参照するが，アルゴリズムにとって距離計算はブラックボックスとして利用する．しかし，次元数が多くなればなるほど近接するオブジェクトの個数が減少する，つまり，空間上のオブジェクトが距離的に疎に分布することになる．また，任意の 2 つのオブジェクト間の距離が特定の距離に集中する．このようなことから距離基準空間インデックスも非距離基準空間インデックスと同様に次元数が多くなると検索時の距離計算回数が急増し，線形探索と同等の距離計算回数となる．

近似検索である距離基準空間インデックスとして距離計算回数の削減においてグラフ構造型のインデックスの有効性が確認されている．グラフにおけるノードは登録されたオブジェクトを意味し，各ノードは近傍のノードとエッジで接続されている．グラフ構造型のインデックスの場合には，エッジをたどりながら検索範囲内のノードを探索して検索結果を得る．各ノードに付与されているエッジ数が多い場合にはグラフ生成時の距離計算回数が大きくなり，かつ，探索時の距離計算回数も増えるが，検索精度は多くなる．なお，近

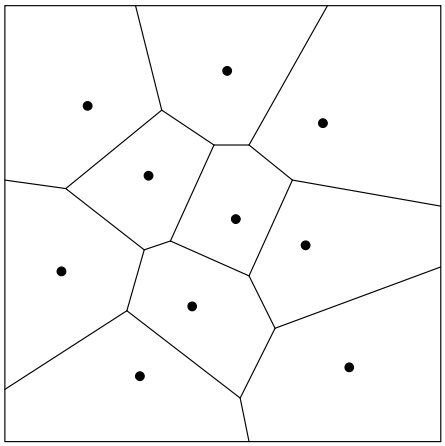


図 2.8　ボロノイ図の例

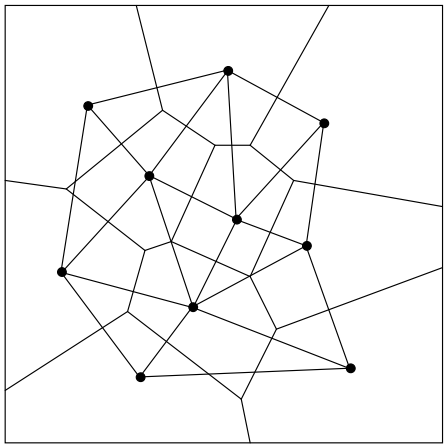


図 2.9　ドロネーグラフの構造

似検索では検索漏れが生じるので距離計算回数だけでなく，検索精度もインデックスの評価尺度となる．検索精度に関しては第 4.4.1 節にて詳細に述べるが本論文ではインデックスの評価における検索精度は再現率を意味する．逆にエッジが少ない場合にはグラフ生成時の距離計算回数が減り，かつ，探索時の距離計算回数も減るが，検索精度は低くなる傾向がある．ボロノイ図において隣接するノードをエッジで結合したグラフであるドロネーグラフ [36] では探索時の距離計算回数を低減でき，かつ，高い検索精度を実現できる．ボロノイ図の例を図 2.8 に，そのドロネーグラフを図 2.9 に示す．

ユークリッド距離の場合にはドロネーグラフの生成が可能なので検索用のインデックスとして利用できるが，多次元データではドロネーグラフの生成時の計算量が多くなる．さらに，ユークリッド距離ではない場合にはドロネーグラフを生成することが困難とされている．しかも，次元数が増えるほどエッジ数は増加するので，ドロネーグラフが生成できたとしても，その膨大な量のエッジにより実装上の問題が生じる．

そこで，ドロネーグラフの一部を構成する木構造型のインデックスである sa-tree[34] や，擬似的なドロネーグラフを生成する最近傍検索手法である NFTG[42] が提案されている．一方，ドロネーグラフではないがグラフの一種である k 最近傍グラフをインデックスとして用いる手法が Sebastian ら [45] や Hajebi ら [24] によって提案されている．k 最近傍グラフの各ノードは検索対象となるオブジェクトであり，各ノードは k 個の最近傍ノードにエッジで接続されている．Sebastian らは k 最近傍グラフによる手法の距離計算回数が vp-tree や AESA より少ないことを示している．また，Hajebi らは同様に LSH や kd-tree よりも少ないことを示しており，k 最近傍グラフの有効性が確認されている．

しかし，k 最近傍グラフは連結グラフ（グラフ内の任意の 2 つのノード間の経路が必ず

存在するグラフ）ではないのでグラフが分断される可能性がある．その結果，検索時に問合せオブジェクトの近傍までグラフをたどることができずに検索精度が抑制される．また，探索を開始する任意のノードが問合せオブジェクトから遠い場合には近傍ノードへの探索時に距離計算回数が増加する．さらには，k 最近傍グラフの生成時の距離計算回数は極めて多い．k 最近傍グラフ生成時の距離計算回数を抑制する方法 [12] [15] が提案されているが，いずれもユークリッド空間を対象とした方法である．距離空間 (metric space) を対象とした方法 [38] もあるが，距離計算回数を削減するのではなく並列処理により処理時間を短縮することに主眼をおいた方法である．

2.4 属性データとオブジェクトの AND 検索

本論文で提案するインデックスを実際のシステムに適用することを想定すると特徴量のみの検索では不十分である．たとえば，商品画像を検索する場合を想定すると商品画像の特徴だけでなく少なくとも商品の分類単位の検索が必要となる．つまり，特徴量以外の属性データとの AND 検索が必要となる．特徴量と属性データの AND 検索に関する研究として Solomon ら [2] は画像特徴量検索と属性データ検索を融合した検索モデルを提案しているが，実際の検索方法に関する提案はなされていない．インデックスが不必要な小規模なデータベースでの AND 検索の研究 [31] があるが，大規模なデータベースの場合には特徴量と属性データを個別に検索した後で融合する方法が一般的である．たとえば Esuli ら [17] は特徴量で検索した後でテキスト検索を行っている．画像特徴量と属性データの AND 検索に関するコンペティションとして ImageCLEF[26] が有名であるが，複数のデータ種別を個別に検索したデータを利用することが前提になっている．

2.5 本章のまとめ

本章では空間インデックスの提案（第 3〜6 章）に関連する従来研究，および，提案手法の適用事例（第 7 章）において必要となる属性データとオブジェクトの AND 検索に関する従来研究について述べた．本節では特に本論文の主題である空間インデックスに関して，従来研究と提案との関係をまとめる．空間インデックスは非距離基準空間インデックスと距離基準空間インデックスの大きく 2 種類に分けられる．非距離基準空間インデックスは各次元要素に基づきインデックスを生成する．つまり，特定の空間に依存するインデックスともいえる．一方，距離基準空間インデックスは距離のみを用いてインデックスを生成するので，様々な距離に利用できる．マルチメディアデータから抽出する特徴量で

表 2.1　距離基準インデックスの特徴

手法	空間分割 形状	インデックス 形状	基準当りの 分岐数	動/静的 インデックス
gh-tree	超平面	木	1	静的
bisector tree	超平面 + 超球	木	1	静的
mb-tree	超平面 + 超球	木	1	静的
GNAT	超平面	木	1	静的
vp-tree	超球	木	2	静的
mvp-tre	超球	木	2	静的
bk-tree	超球	木	1	静的
vp-forest	超球	複数木	2	静的
AESA	非分割	表	該当せず	静的
LAESA	非分割	表	該当せず	静的
M-tree	超球	木	1	動的

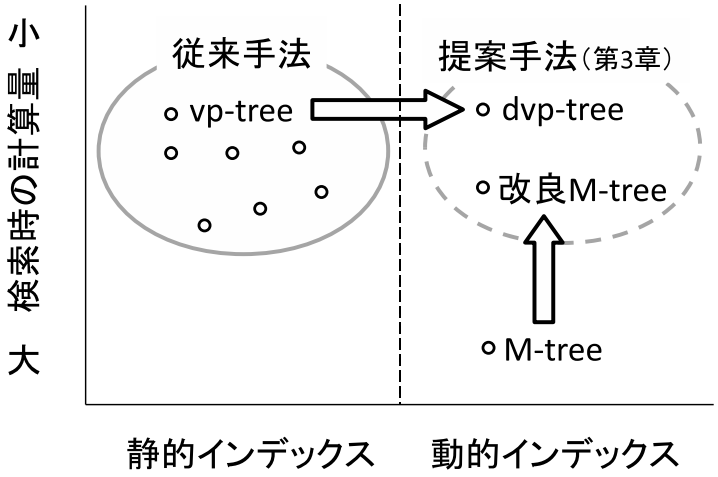


図 2.10　従来手法と提案手法（第 3 章）の関係

は様々な距離が使われるので，距離基準空間インデックスを提案の対象とする．第 3 章では，従来手法の距離基準空間インデックスの改良を行い，動的インデックス，かつ，検索時の距離計算回数の削減，をめざす．そこで，静的なインデックスの中から距離計算回数が少ないインデックスを選択し，動的なインデックスに改良（dvp-tree）する．従来手法の距離基準空間インデックスの特徴を表 2.1 にまとめる．距離計算回数が少ないか否かを判断するには，木構造のノードをたどるときに，各ノードにおいて，問合せオブジェクトと基準オブジェクトの 1 回の距離計算によって，検索範囲が子ノードの各空間と交差す

表 2.2 近似検索空間インデックスの特徴

手法	空間分割 形状	インデックス 形状	生成 基準	動/静的 インデックス
ANN	超平面	木	非距離	静的
LSH	非分割	ハッシュ	非距離	動的
sa-tree	非分割	木	距離	静的
NFTG	非分割	グラフ	距離	静的
k 最近傍グラフ	非分割	グラフ	距離	静的

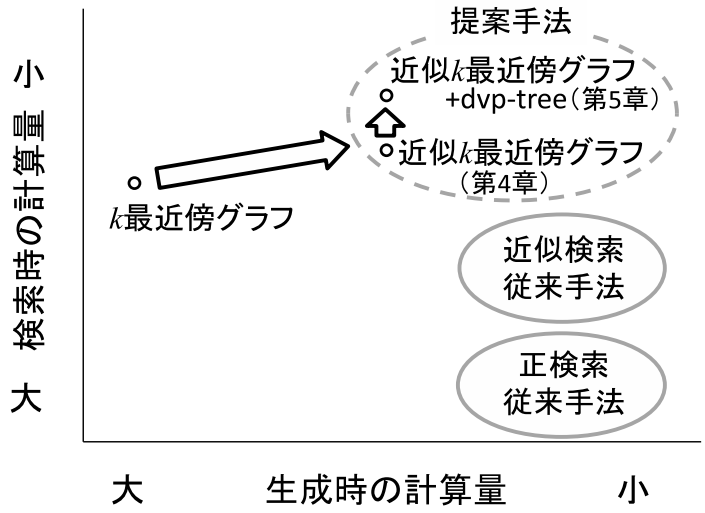


図 2.11 従来手法と提案手法（第 4〜5 章）の関係

るか否かを判断できる数の平均（表中の基準当りの分岐数）が 1 つの目安となる．表より vp-tree 系のインデックスは基準当りの分岐数が多い，つまり，距離計算回数が少ないことがわかる．そこで，vp-tree 系のインデックスの基本アルゴリズムである vp-tree を改良の対象とする．AESA や LAESA は大量の距離計算をインデックス生成時に行う手法であり，動的インデックスに変更しても実用的な時間での登録が困難であると判断した．一方，唯一の動的インデックスである M-tree の距離計算回数削減の改良（改良 M-tree）も同時に行う．図 2.10 に従来手法と提案手法の関係を示す．

第 3 章では低次元データを対象としたが第 4 章以降では高次元データを対象とするために近似検索を採用する．従来手法の近似検索インデックスの特徴を表 2.2 にまとめる．距

離基準インデックスの中で計算量が少ないことが確認されている k 最近傍グラフ [45],[24] に着目し，インデックス生成時の計算量を削減するために，k 最近傍グラフを近似する近似 k 最近傍グラフを第 4 章で提案する．さらに，検索時の計算量を削減するために，dvp-tree を近似 k 最近傍グラフに組み合わせる手法を第 5 章で提案する．図 2.11 に従来手法と提案手法の関係を示す．なお，図中の正検索は漏れのない検索手法を意味する．

第 3 章

近傍検索のための更新可能な木構造型インデックス

3.1 本章の目的

マルチメディアデータから抽出した多次元データで表される特徴量のオブジェクトを高速に検索するためには空間インデックスが必須となる．しかし，その特徴量空間は L_p 空間とならないことが多いことから，非距離基準空間インデックスではなく距離基準空間インデックスを利用する．本章では既存の距離基準空間インデックスである vp-tree および M-tree を改良したインデックスを提案する．vp-tree にはオブジェクトの追加更新が可能なアルゴリズムを導入し，M-tree には検索時の計算量を削減できるオブジェクトの登録アルゴリズムを導入する．

3.2 課題

本論文では実際のシステムへの適用を想定してオブジェクトの追加登録が可能な動的インデックスを提案することを目的とする．しかし，前章で述べたように，ほとんどの距離基準空間インデックスではオブジェクトの登録，削除ができない．一方，唯一の動的インデックスである M-tree ではオブジェクトが登録され木構造が成長するときには B-tree[5] と同様にリーフで生成されたノードが繰り上がってルートノードで木構造全体が繰り下がるように成長する．このことに起因してノード空間が大きくなり，検索速度が低下する傾向がある．

そこで，第 1 に，静的なインデックスに関しては動的に登録できるように改良を行う．

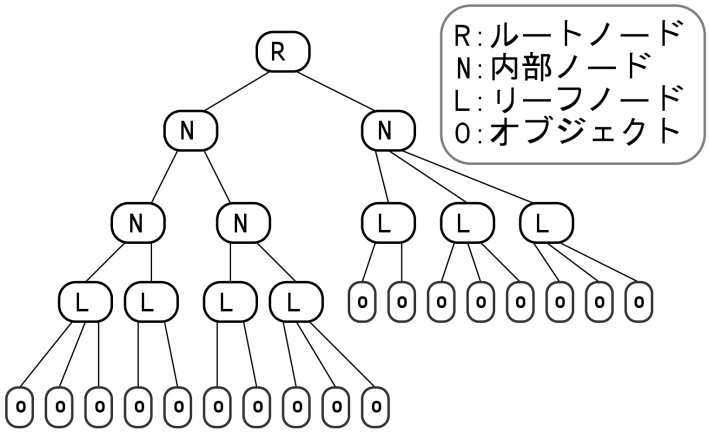


図 3.1　木構造のデータ構成

なお，静的インデックスのうち vp-tree 以外はデータ構造が複雑であり，動的なアルゴリズムに変更することが極めて困難である．さらに，実装できたとしてもデータ構造の複雑さから登録，削除（検索木の縮退）の処理時間の増大が問題になることが容易に予想された．そこで，本章では，比較的簡便なアルゴリズムである vp-tree に改良を加えたインデックスを提案する．第 2 に，動的なインデックスである M-tree に関しては検索速度の向上を目的とする改良を加えたインデックスを提案する．

3.3　提案手法

木構造のデータ構造は基本的に vp-tree と M-tree の両方で共有できる構造とした．図 3.1 に木構造のデータ構造を示す．木構造はルートノード，内部ノード，リーフノード，そしてオブジェクトからなる．ルートノードは木構造の起点となるノードであり，リーフノードは末端のノードである．リーフノードは個々のオブジェクトへのリンクをもつ．本章では検索の高速化のためにノードデータはすべてメモリ上に展開するが，オブジェクトデータは容量が大きいので二次記憶に置くこととした．大容量のメモリを確保できるのであればメモリ上に展開することも可能である．ルートノード，内部ノード，リーフノードは以下のデータで構成される．

- ノード ID
- ノード種別（リーフノード，内部ノードの識別）
- 子ノードのノード ID リスト（リーフノードの場合にはオブジェクトの ID リスト）
- 分割円（vp-tree 用）又はノード円（M-tree 用）の中心オブジェクト

- 分割円の半径（vp-tree 用）
- ノード円の半径（M-tree 用）
- 距離リスト（リーフノードの場合にリーフノードにリンクする全オブジェクトと中心オブジェクト間の距離）

3.3.1 vp-tree の改良

vp-tree に対し動的なインデックス生成への変更を行っただけでなく，検索の高速化のためにリーフノードにおける検索アルゴリズムの改良を行った．なお，以降改良した vp-tree は dvp-tree (dynamic vantage point tree) とよぶこととする．

動的なインデックス生成

vp-tree では登録するオブジェクト集合があらかじめ与えられていることを想定しており，木構造を生成するときに順次このオブジェクト集合から任意のオブジェクトを取得する．したがって，動的にオブジェクトを登録することができない．そこで，静的に生成するアルゴリズムにおいて任意に選択するオブジェクトとして，動的に登録する新規のオブジェクトを扱うことで動的にデータを登録できるように改良した．

登録のアルゴリズムを Algorithm 1 に示す．なお，d は距離関数である．登録オブジェクトはルートノードからオブジェクトを包含するノードを順次たどり，最終的にオブジェクトを包含するリーフノードに到達する．そのリーフノードに空きがあればオブジェクトを追加し，なければリーフノードを分割する．分割するアルゴリズムが Split である．リーフノードは内部ノードとなり分割されたリーフノードの親ノードとなる．Algorithm 1 中のリーフノードを分割する Split のアルゴリズムは様々考えられるが，本論文では次のようなアルゴリズムを用いた．オブジェクト内の任意のオブジェクトを 1 つ選択し，そのオブジェクトから最遠にあるオブジェクトを分割円の中心オブジェクトとする．中心オブジェクトから各オブジェクトとの距離を求め中間点に位置するオブジェクトとの距離を分割円の半径とする．そして，分割円の内部，外部によりオブジェクトを二分する．

検索アルゴリズムの改良

vp-tree で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たどり，最終的にたど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に逐一アクセスし，距離を算出し検索範囲に適合するか否かを調べる．したがって，リーフノードにおけるこの処理が距離計算回数だけでなくオブジェクトへのアクセス回数を増大させている要因となって

Algorithm 1 Insert$(N, \boldsymbol{o}_i)$

```
# In:N, o_i
# N:処理対象ノード，o_i:登録オブジェクト
if N がリーフノードではない then
  # o:N の中心オブジェクト，r:ノード N の分割円の半径
  if d(o, o_i) ≤ r then
    # N_ci:ノード N の分割円内部の子ノード
    Insert(N_ci, o_i)
  else
    # N_co:ノード N の分割円外部の子ノード
    Insert(N_co, o_i)
  end if
else
  # N がリーフノードである
  if N に空きがある then
    N に o_i を追加する
  else
    # N に空きがない
    S ← リーフノードの全オブジェクト ∪ o_i
    # Split:リーフノードの分割
    # In:S, Out:S_i, S_o, o, r
    # S_i:分割円内部のオブジェクト集合，S_o:分割円外部のオブジェクト集合
    Split(S, S_i, S_o, o, r)
    リーフノード N_ci を生成し S_i を設定する
    N_ci の距離リストを設定する
    リーフノード N_co を生成し S_o を設定する
    N_co の距離リストを設定する
    N の子ノードを N_ci, N_co とし o, r を設定する
  end if
end if
```

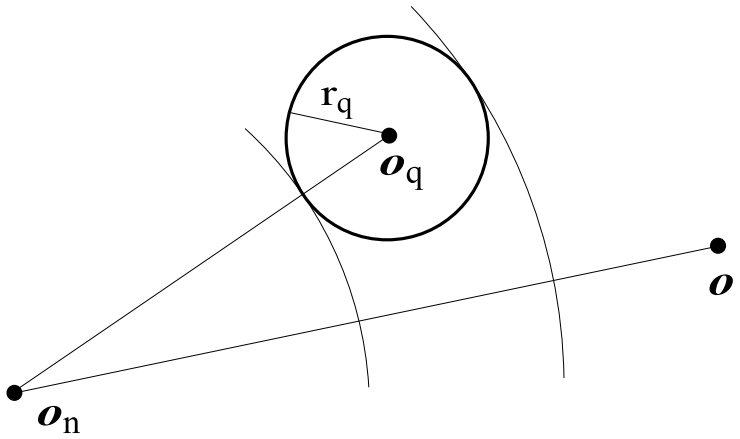


図 3.2 検索範囲とオブジェクトの関係

いる．

そこで，検索速度を向上させるためにリーフノードではリーフノードの中心オブジェクトと各オブジェクト間との距離を距離リストとして保持する．中心オブジェクトはリーフノードにリンクするオブジェクトのうちの任意の単一オブジェクトとする．この中心オブジェクトと各オブジェクト間と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．検索範囲の中心オブジェクトを $\boldsymbol{o}_q$，半径を r_q，リーフノードの中心オブジェクトを $\boldsymbol{o}_n$，リーフノードにリンクするオブジェクトを $\boldsymbol{o}$ とすると，これらの関係は図 3.2 となり，以下の定理が成り立つ．

定理 1

$|d(\boldsymbol{o}_n, \boldsymbol{o}_q) + d(\boldsymbol{o}_n, \boldsymbol{o}_q)| > r_q$ が成り立つならば，オブジェクトは検索範囲に存在しない

証明 1

三角不等式

$$d(\boldsymbol{o}_n, \boldsymbol{o}_q) + d(\boldsymbol{o}_q, \boldsymbol{o}) \geq d(\boldsymbol{o}_n, \boldsymbol{o})$$

より，

$$d(\boldsymbol{o}_n, \boldsymbol{o}) - d(\boldsymbol{o}_n, \boldsymbol{o}_q) > r_q$$

は

$$d(\boldsymbol{o}_q, \boldsymbol{o}) > r_q$$

となり，交差しないことがわかる．

$$-d(\boldsymbol{o}_n, \boldsymbol{o}) + d(\boldsymbol{o}_n, \boldsymbol{o}_q) > r_q$$

も同様にして，

$$d(\boldsymbol{o}_q, \boldsymbol{o}) > r_q$$

となる．したがって，定理 1 は成立する．

定理 1 の $d(\boldsymbol{o}_n, \boldsymbol{o})$ および r_q はいずれも既知であり，$d(\boldsymbol{o}_n, \boldsymbol{o}_q)$ は各リーフノードに対して 1 回計算すればよく，各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．したがって，距離計算回数やオブジェクトへのアクセス回数を大幅に削減できる．ただし，定理 1 の式が成立しないからといって必ず検索範囲に存在するわけではないので，成立しない場合にはオブジェクトとの距離を算出し，検索範囲にあるか否かを調べなければならない．

実際の範囲検索のアルゴリズムを Algorithm 2 に示す．ノードがリーフの場合の 1 つ目の if 文 (*1) での判定はオブジェクトを取得せずに，かつ，個々のオブジェクトとの距離計算なしに判定ができる．2 つ目の if 文 (*2) の判定ではオブジェクトを取得し，かつ，距離計算をしなければ判定できない．したがって，1 つ目の判定により 2 つ目の時間を要する判定をせずに検索半径に包含されないことが判別でき，検索の高速化が可能である．また，k 最近傍検索 (k-nearest neighbor query) は範囲検索のアルゴリズムに基づいており，以下に述べるアルゴリズム [39] を採用した．検索初期値では検索半径は無限大とし，ルートからたど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超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超えないようにする．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を得られる．

3.3.2　M-tree の改良

他の距離基準インデックスとは異なり，M-tree は動的にインデックス生成が可能である反面，空間を分割する円が互いに交差するので距離計算回数が増加する傾向がある．したがって，円の交差を削減するためには円の大きさを抑制すればよい．円の大きさは木構造の成長方式に依存するので木構造の成長のアルゴリズムを改良する．

M-tree の成長アルゴリズム

M-tree は B-tree と同様にボトムアップの木構造の成長が大きな特徴となっている．新たなオブジェクトを挿入するときには検索時と同様にルートから木構造をたどり適切なリーフノードにオブジェクトを加える． 加えた結果，オブジェクト数が最大分岐数を超

Algorithm 2 Search($N, \boldsymbol{o}_q, r_q, L$)

```
# In:N, o_q, r_q,Out:L
# N:処理対象ノード，o_q:検索オブジェクト，r_q:検索半径，L:検索結果リスト
if N がリーフノードではない then
  # o:N の中心オブジェクト，r:N の分割円の半径
  if d(o, o_q) ≤ r + r_q then
    # 分割円の内部と交差する
    # N_ci:ノード N の分割円内部の子ノード
    Search(N_ci, o_q, r_q, L)
  end if
  if d(o, o_q) + r_q > r then
    # 分割円の外部と交差する
    # N_co:ノード N の分割円外部の子ノード
    Search(N_co, o_q, r_q, L)
  end if
else
  # N がリーフノードである
  for all o_c ∈ リーフノードの全オブジェクト do
    # *1
    if |d(o, o_q) − d(o, o_c)| ≤ r_q then
      # *2
      if d(o_q, o_c) ≤ r_q then
        L ← L ∪ o_c
      end if
    end if
  end for
end if
```

えた場合にはリーフノードのオブジェクト集合を 2 つのリーフノードに分割する．一方は元のリーフと置換え，他方は親ノードに加える．その結果，親ノードの子ノード数が最大分岐数を超えた場合には同様に親ノードの子ノード群を分割する．分割した結果，最大分岐数を超えるとさらに親ノードへと遡っていく．ルートノードまで遡った場合にルートノードが最大分岐数を超えるとルートノードを分割し，木構造はルート部分で全体的に繰り下がる形態で成長する．したがって，木構造は常にバランスを保っている．木構造がバランスを保っているので，いかなる検索もほぼ同等の検索速度となる利点があるが，木構造がルートから繰り下がる形態で成長するときに子ノードの円を包含する円を設定するので，成長するごとに円が急激に増大する傾向がある．ノード円が巨大になると各ノード円の重複が増大し，探索枝の絞り込みが悪く検索速度が低下する傾向がある．そこで，このようなボトムアップによる木構造の成長をやめ，R-tree のようなトップダウンの成長方式を取り入れた．トップダウンの成長方式とすることで木構造のバランスが悪くなるという欠点は発生するが，子ノードの円の重複を減少させることでバランスが悪くなることによる検索速度の低下を十分補えると判断した．

成長アルゴリズムの改良

改良した M-tree のオブジェクトの挿入手順を述べる．オブジェクト挿入時にはルートから順に，オブジェクトを包含する子ノードがある場合にはその子ノードをたどり，オブジェクトを包含する子ノードがない場合にはオブジェクトを包含する円を生成するのに最小限の円の拡張で済む子ノードをたどる．こうして，リーフノードまでたどり着き，そのリーフノードに空きがあるならオブジェクトを追加し，空きがなければオブジェクトを加えた上でリーフノードを分割する．具体的なアルゴリズムを Algorithm 3 に示す．

Algorithm 3 中の Split は S を S_1 と S_2 のオブジェクト群に分割する．また，S_1，S_2 の中心オブジェクト $\boldsymbol{o}_1$，$\boldsymbol{o}_2$，半径 r_1，r_2 を同時に求める．2 分するアルゴリズムは様々考えられるが，本章では以下の式を満たすオブジェクト $\boldsymbol{o}_i$，$\boldsymbol{o}_j$ を中心オブジェクトとする 2 つの集合に分割する．

$$\min_{\boldsymbol{o}_i,\boldsymbol{o}_j \in S} \{ \max_{\boldsymbol{o}_n \in S_1} (d(\boldsymbol{o}_i, \boldsymbol{o}_n)) + \max_{\boldsymbol{o}_n \in S_2} (d(\boldsymbol{o}_j, \boldsymbol{o}_n)) \}$$

なお，

$$S_1 = \{\boldsymbol{o} | \boldsymbol{o} \in S \text{ and } d(\boldsymbol{o}_i, \boldsymbol{o}) \leq d(\boldsymbol{o}_j, \boldsymbol{o})\}$$

$$S_2 = \{\boldsymbol{o} | \boldsymbol{o} \in S \text{ and } d(\boldsymbol{o}_i, \boldsymbol{o}) > d(\boldsymbol{o}_j, \boldsymbol{o})\}$$

つまり任意の 2 つのオブジェクトを中心オブジェクトとして選択し他のすべてのオブジェクトを距離の近い方の中心オブジェクトに振り分ける．こうして各中心オブジェクトに属

Algorithm 3 Insert$(N, \boldsymbol{o}_i)$

```
# In:N, o_i
# N:処理対象ノード，o_i:登録オブジェクト
if N がリーフノードではない then
  # o_c:子ノードの中心オブジェクト，r_c:子ノードの半径
  if d(O, o_c) ≤ r_c を満たす子ノードがある then
    d(o_i, o_c) が最も小さい子ノード N_c を求める
    Insert(N_c, o_i)
  else
    d(o_i, o_c) − r_c が最も小さい子ノード N_c を求める
    Insert(N_c, o_i)
  end if
else
  # N がリーフノードである
  if N に空きがある then
    N に o_i を追加する
  else
    # N に空きがない
    S ← リーフノードの全オブジェクト ∪ o_i
    # Split:リーフノードの分割
    # In:S, Out:S_1, S_2, o_1, o_2, r_1, r_2
    # S_n:オブジェクト集合，o_n:S_n の中心オブジェクト，r_n:S_n の半径
    Split(S, S_1, S_2, o_1, o_2, r_1, r_2)
    リーフノード N_1 を生成し S_1, o_1, r_1 を設定する
    リーフノード N_2 を生成し S_2, o_2, r_2 を設定する
    N の子ノードを N_1, N_2 とする
  end if
end if
子ノードまたはリーフノードの全オブジェクトを包含するように r を設定する
```

するオブジェクトを包含する 2 つの円の半径の合計を求める．これをすべての 2 つの組み合せについて行い，半径の合計が最も小さくなる 2 つのオブジェクトを新しいオブジェクトの中心オブジェクトとする．なお，検索アルゴリズムに関しては改良を加えていない．

3.4　評価実験

提案手法である dvp-tree，改良 M-tree，そして比較のために M-tree を実際に色特徴による類似画像検索に実装した上で性能評価を行った．使用マシンは Sun Ultra 60(UltraSPARC-II 296MHz)，Solaris 2.6 で，2 次記憶には Ultra Wide SCSI ハードディスク (7200rpm) を使用した．登録画像としてフォト画像やクリップアート画像などの 25,458 画像を用いた．M-tree および改良 M-tree の内部ノードの最大分岐数およびリーフノードの最大分岐数はともに 10 とした．dvp-tree の内部ノードの分岐数は 2 とし，リーフノードの最大分岐数は 10 とした．

画像特徴量として画像全体の色のヒストグラムを用い，特徴量の距離として 2 次形式（quadratic form）距離を利用した．

$$d_h(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{A} (\boldsymbol{x} - \boldsymbol{y}) \tag{3.1}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} (x_i - y_i)(x_j - y_j) \tag{3.2}$$

なお，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であり，以下のように表される．

$$a_{ij} = 1 - d(c_i, c_j)/d_{max}$$

$d(c_i, c_j)$ は i 番目と j 番目のビンの色空間上での距離 (色差)，d_{max} はその最大値である．ヒストグラムのビン数（次元数）は 54 であり，上記式で示されるように各ビンの相関をすべて計算するのでユークリッド距離に比べて計算量が極めて大きくなる．なお，距離基準空間インデックスは非距離基準空間インデックスと異なり，距離のみに基づきインデックスを形成するので性能は次元数とは直接関係しない．しかし，距離定義により与えられた距離分布の傾向により性能が左右される．登録オブジェクトの距離分布を図 3.3 に示す．登録したオブジェクトから任意に 1,000 組のオブジェクトの組を選択し，そのオブジェクト間の距離を求め，距離を 1 単位に区分し各区分での出現度数をカウントした．

図 3.4 に全オブジェクトの登録時間を示す．登録時間には画像から特徴量としてのオブジェクトの抽出時間は含まれていない．したがって，実際の登録処理全体ではオブジェクトの抽出処理によっては，かなりの時間を要する．図より改良 M-tree は登録速度の面で

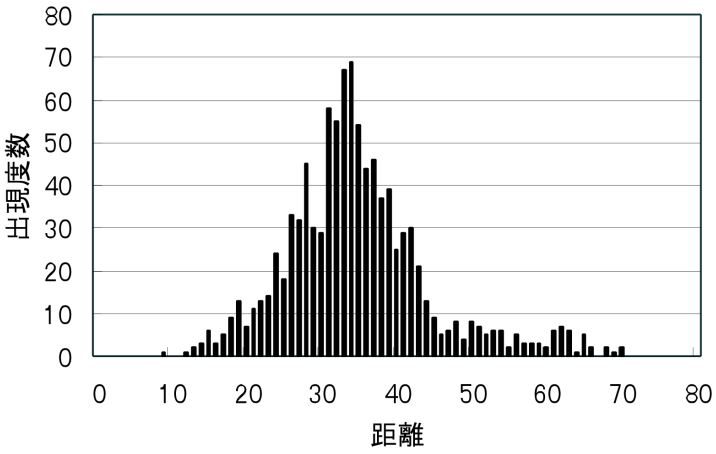


図 3.3 任意の 2 つのオブジェクト間の距離分布

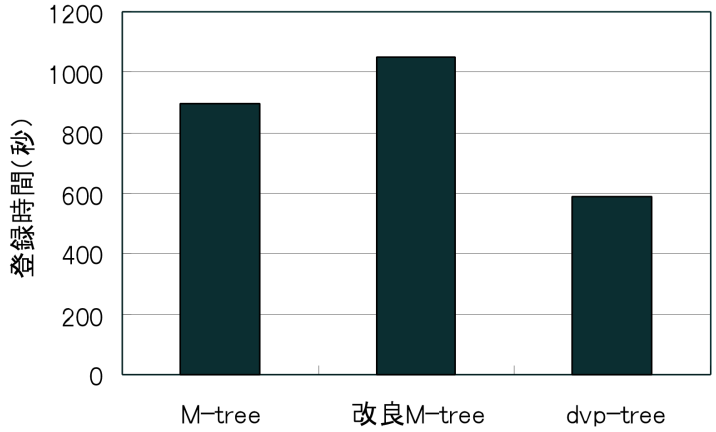


図 3.4 オブジェクトの登録時間

不利である一方 dvp-tree は他者より格段に速く M-tree に対して 34 %削減できている．次に，検索件数を 10,20,50,100 とした場合の k 最近傍検索を行った．この検索数は実用上の類似画像検索での要求仕様を想定して選択した．類似画像検索では範囲検索ではなく k 最近傍検索の要望が多い．また，k 最近傍検索は範囲検索のアルゴリズムに基づいており，k 最近傍検索のアルゴリズム自体はどちらのインデックスでも同様に実装されているので，k 最近傍検索を評価すれば範囲検索も評価することとなる．こういった理由により本評価では k 最近傍検索のみを行った．

検索時間に占める大きな要素の 1 つはオブジェクト間の距離計算である．2 次形式距離のように計算量が大きい場合には検索時間の大きな割合を距離計算が占める．したがっ

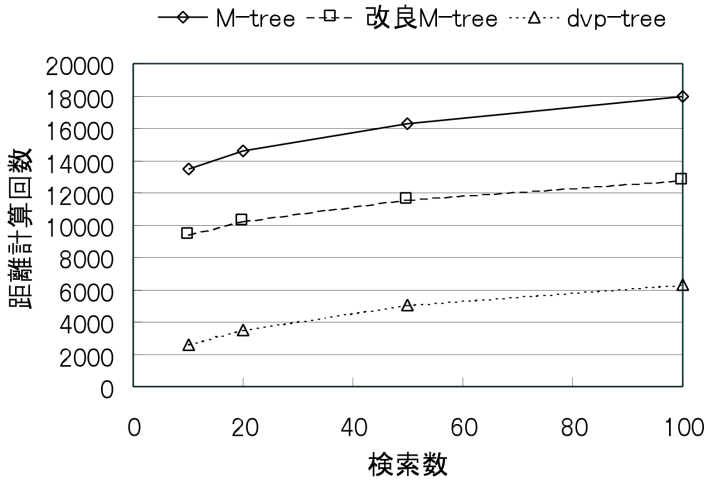


図 3.5　k 最近傍検索の距離計算回数

て，距離計算回数が少ないほど高速な検索が可能となる．検索時間に占めるもう 1 つの大きな要素は 2 次記憶へのアクセス時間である．大容量のメモリが安価に入手できるようになったので，すべてのデータをメモリ上に配置することも近年では可能であるが，本章で提案するインデックスではノードデータはメモリ上に，オブジェクトデータは 2 次記憶に配置している．したがって，オブジェクトデータへのアクセス時間は無視できない．

まず，検索時間に大きな割合を占める 1 つ目の要素である距離計算回数を図 3.5 に示す．図より M-tree, 改良 M-tree，dvp-tree の順で距離計算回数が減少していることがわかる．また，検索数が増加するといずれのインデックスも距離計算回数が増加しているが，検索数に正比例するほどは増加していない．

次に 2 つ目の大きな要素である 2 次記憶へのアクセス時間としてノードのリード回数を図 3.6 に，オブジェクトのリード回数を図 3.7 に示す．いずれも距離計算回数と同様に M-tree，改良 M-tree，dvp-tree の順でリード回数が減少しており，検索件数が増えると増加傾向が見られる．検索件数が増加するとノードのリード回数よりもオブジェクトのリード回数の増加傾向が強い．本実装ではノードデータをメモリ上に配置しているので，ノードのリード回数の増加による検索時間への影響は少ない．しかし，オブジェクトは 2 次記憶に配置しているので，オブジェクトのリード回数が検索時間へ与える影響は大きい．リーフノードには複数（本評価では 10）のオブジェクトが関連付けられるので，総ノード数はオブジェクト数より大幅に少ない．したがって，本評価ではノードデータのみをメモリに配置したが，近年のメモリの価格下落に伴い，オブジェクトデータもメモリに配置することが容易となった．本結果からオブジェクトデータもメモリに配置することで

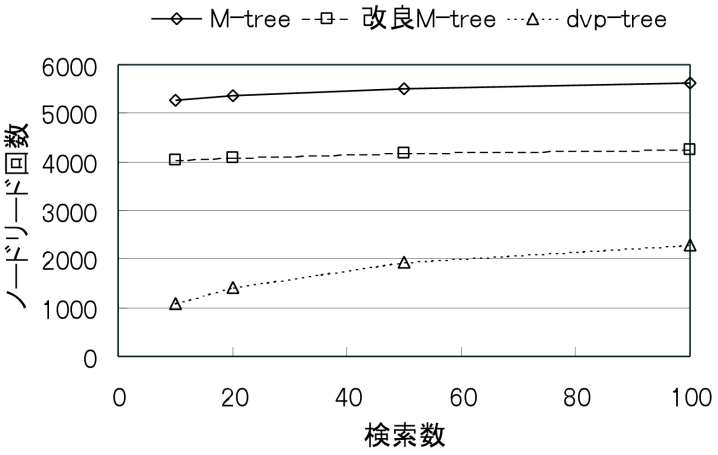


図 3.6 k 最近傍検索のノードリード回数

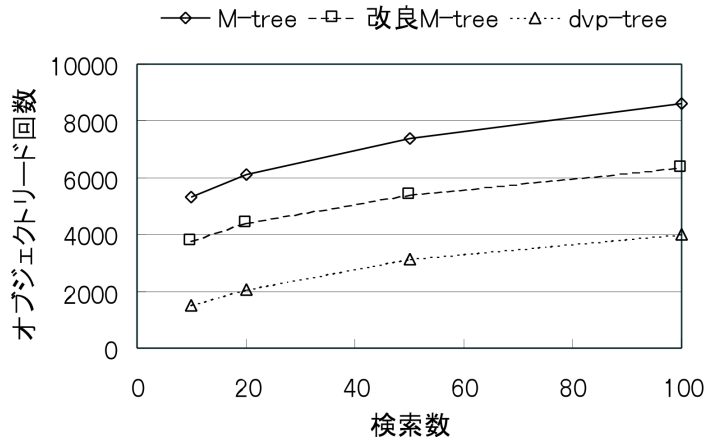


図 3.7 k 最近傍検索のオブジェクトリード回数

検索時間の削減が期待できることを確認した．

次に実際の検索時間を図 3.8 に示す．なお，検索時間には問合せ画像から特徴量としてのオブジェクトを抽出する時間は含まれていない．図中の全件検索とはインデックスを利用せずにすべてのオブジェクトと逐一距離計算を行い検索する線形探索を示す．距離計算回数およびオブジェクトリード回数から予想される通りの傾向を示しており，M-tree，改良 M-tree，dvp-tree の順で検索速度が速くなり，当然全件検索よりも格段に速い．改良 M-tree は M-tree に対して 15〜16%，dvp-tree では 49〜62% 削減できている．

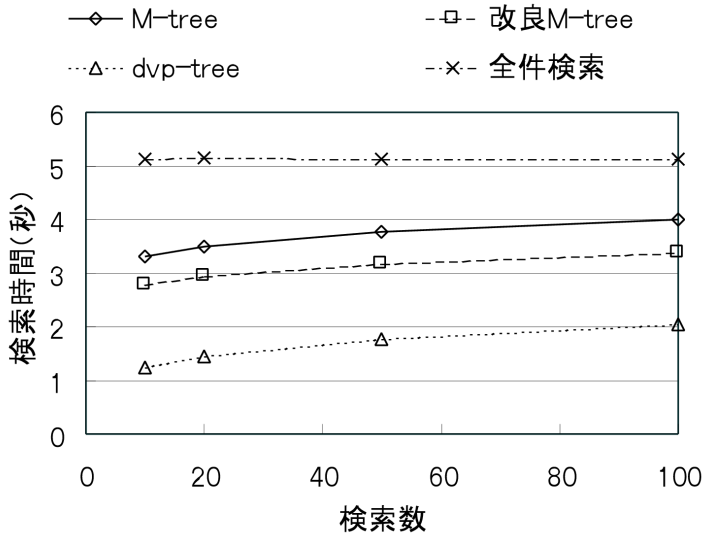


図 3.8　k 最近傍検索の検索時間

3.5　本章のまとめ

オブジェクトの検索インデックスとして L_p 距離に基づいた非距離基準空間インデックスは容易に適用できない．これに対して距離基準空間インデックスは距離定義を満しさえすれば適用可能である．しかし，静的な距離基準空間インデックスである vp-tree は検索時の計算量削減の効果が高い代わりに，オブジェクトを追加登録できないという問題点がある．一方，動的な距離基準空間インデックスである M-tree は追加登録が可能である代わりに，検索時の計算量が比較的多い問題点がある．そこで，静的なインデックスである vp-tree は動的にオブジェクトを追加できるように改良し，検索の高速化のために検索アルゴリズムにも改良を加えた．また，動的な距離基準空間インデックスである M-tree については検索の高速化のためにボトムアップの木構造成長からトップダウンに変更した．さらに，画像特徴量をオブジェクトとして用いて vp-tree を改良した dvp-tree および M-tree を改良した改良 M-tree の評価を行った．その結果，検索時の計算量において dvp-tree が M-tree および改良 M-tree を上回り dvp-tree の有効性を確認した．一方，M-tree はその特徴であるボトムアップの木構造成長よりもトップダウンの木構造成長の方が検索性能が向上することを確認した．しかし，改良 M-tree の検索性能は dvp-tree を優ることはできず，M-tree の空間分割の方式に起因する問題が大きいと判断する．

第 4 章

近似近傍検索のための近似 k 最近傍グラフ

4.1 本章の目的

前章では，オブジェクトの追加更新ができない静的な vp-tree に改良を加えて，追加更新が可能な dvp-tree を提案し，追加更新が可能な M-tree よりも検索時の計算量を抑制できることを確認した．dvp-tree は数十次元程度の比較的低次元のオブジェクトに対しては計算量の削減効果があるが，数百次元といった高次元のオブジェクトに対しては検索時の計算量の削減効果がなくなり，線形探索と同等の計算量となる．そこで，漏れを許容することで検索時の計算量を削減できる近似検索インデックスに着目した．近似検索インデックスの中でも k 最近傍グラフをインデックスとして用いる手法の検索時の計算量が少ないことが知られている．しかし，インデックス生成時の膨大な計算量，および，非連結グラフに起因する検索精度の抑制が課題である．本章では，k 最近傍グラフの構造を近似する近似 k 最近傍グラフを提案する．グラフの生成時には各ノードの近傍ノードを検索する必要があるが，その検索に生成途中のグラフを用いることで大幅な計算量の削減が可能となる．また，グラフに逐一ノードを接続するので連結グラフとなり検索精度が向上する．

4.2 課題

数十次元程度の比較的低次元のオブジェクトの検索時には前章で提案した dvp-tree によって線形探索よりも十分に少ない計算量で検索することが可能となる．しかし数百次元

といった高次元のオブジェクトの場合にはインデックスの効果がなくなり線形探索と同等の計算量になる．距離基準空間インデックスの場合には次元要素を考慮しないが，次元が多くなると任意の 2 つのオブジェクト間の平均距離の分布が変化することでインデックスの効果がなくなる．次元が多くなると任意の 2 つのオブジェクト間の距離が一定の距離に集中する傾向が現れる．dvp-tree を例に距離の集中によりどのような影響を受けるか述べる．dvp-tree の場合には空間を超球により二分するときに，各部分空間に同数のオブジェクトが属するように二分する．高次元データの場合には任意の 2 つのオブジェクト間の距離が特定の距離に集中する．超球の半径も，集中しているその特定の距離とほぼ同一となるので，超球の境界面近辺にオブジェクトが集中することになる．超球の境界面にオブジェクトが集中するのと同様に検索オブジェクトも境界面に存在する可能性が高くなる．したがって，検索範囲は 2 つの領域と重複することになる．その結果，探索空間の絞込みができなくなり計算量の削減ができなくなる．

一方，検索時に検索漏れが発生するが，その代わりに計算量を大幅に削減できる近似検索の手法がある．近似検索の手法の中でも k 最近傍グラフを用いた近似検索手法は計算量が少ないことで知られている．k 最近傍グラフではグラフを構成する各ノードが登録されたオブジェクトであり，各ノードがエッジによって k 個の最近傍ノードへ有向エッジ（単方向接続）で接続されている有向グラフ（単方向接続のエッジで構成されているグラフ）である．次に，k 最近傍グラフを用いた検索アルゴリズムを説明した後に，その課題について述べる．

4.2.1 グラフ構造型インデックスによる検索

グラフが以下の条件 1 を満たすならば，エッジで結合した近傍ノードを探索すること（局所探索）によって最近傍検索が可能である [34] とされている．

条件 1　G をグラフ，p を G のノード，$N(G,p)$ を G における p の近傍ノードの集合，q を問合せオブジェクト，$d(p,q)$ を p，q 間の距離とすると，ノード p が $N(G,p)$ に属する任意のノード x に対して $d(p,q) < d(x,q)$ が成り立つならば，G に属する任意のノード x に対して $d(p,q) < d(x,q)$ が成り立つ．

条件 1 を満足する場合，Algorithm 4 に示す最良優先探索により最近傍検索が可能である．Algorithm 4 の G をグラフ，q を問合せオブジェクト，s を探索起点ノード（グラフの探索を開始するノード）とする．各ノードに全ノードへのエッジが付与されている完全グラフも条件 1 を満たすが，あるノードの近傍ノードは全ノードとなるので，全件に対し

Algorithm 4 NearestNeighborSearch(G, q, s)

$p \leftarrow s$
$P \leftarrow N(G, p)$
$c \leftarrow \underset{x \in P}{\operatorname{argmin}}\, d(x, q)$
if $d(c, q) < d(p, q)$ **then**
  $p \leftarrow$ NearestNeighborSearch(G, q, c)
end if
return p

て線形探索することと同等となる．ドロネーグラフも条件 1 を満足する [34] とされており，かつ，比較的少ないエッジで構成される．k 最近傍グラフの場合には条件 1 を満足しないので，Algorithm 4 では最近傍のノードが見つからない場合が存在する．そこで，異なる探索起点ノードにより何回か最近傍検索を繰り返し，最も近いノードを最近傍のノードとする必要がある．なお，グラフ上の任意のノードを探索起点ノードとして用いる．

範囲検索の場合には，まず最近傍検索 (Algorithm 4) を行い，次にエッジをたどって検索範囲内のすべてのノードを探索する．条件 1 を満足する場合にはこの方法で十分であるが，k 最近傍グラフの場合には，最近傍検索において範囲内の近傍ノードが見つからない場合がある．そこで，異なる探索起点ノードにより範囲内のノードが見つかるまで繰り返し最近傍検索を行う．また，範囲内のノードの探索においても，近傍のノードへの経路が一度範囲外に出た後に再度範囲内に戻る場合を想定しなければならない．そこで，検索範囲より広い範囲を探索する必要があるので，探索する範囲（超球）の探索半径 r_s を式 4.1 のように決定する．

$$r_s = (1 + \epsilon)r \tag{4.1}$$

ただし，r は検索範囲を示す検索半径，ϵ は探索半径係数とする．この探索半径 r_s を用いた範囲検索のアルゴリズムを Algorithm 5 に示す．r を検索半径，R を検索結果集合とし，C は同じノードを何度も探索しないように判定するための集合である．R，C の初期値には空集合を設定する．

本論文では，k 最近傍検索として検索半径の初期値を無限大とし，検索結果の最遠のノードの半径により検索半径を狭めていく一般的な手法 [39] を採用する．したがって，範囲検索 (Algorithm 5) と同様に，まず，最近傍検索を行うが，範囲検索とは異なり検索半径の初期値が無限大であることから，最近傍検索により必ず範囲内（無限大）に入るの

Algorithm 5 RangeSearch(G, q, s, r, R, C)

```
r_s ← (1 + ε)r
for all p ∈ N(G, s) do
  if p ∉ C then
    C ← C ∪ {p}
    if d(p, q) ≤ r then
      R ← R ∪ {p}
    end if
    if d(p, q) ≤ r_s then
      RangeSearch(G, q, p, r, R, C)
    end if
  end if
end for
```

で最近傍検索は 1 回で十分となる．k 最近傍検索のアルゴリズムを Algorithm 6 に示す．k_s には検索結果の最近傍のノードの数を，検索半径 r の初期値には無限大を，R，C の初期値には空集合を設定する．

このようにして最近傍検索，範囲検索，k 最近傍検索が可能であるが，条件 1 を満たさないので，いずれも近似検索となり検索漏れが発生する可能性がある．しかし，エッジ数を増やすことで条件 1 を満たすノードが増加し，その結果，検索精度が向上する．したがって，検索精度をエッジ数により制御が可能であると考える．

4.2.2　k 最近傍グラフの課題

k 最近傍グラフでは次に示す課題が存在する．

- k 最近傍グラフ生成の膨大な計算量
- 非連結グラフに起因する検索精度の抑制

第 1 の課題に関して，k 最近傍グラフのインデックスを生成するには全ノード間の距離計算を必要とすることから距離計算回数は以下の式で求められる．

$$\text{距離計算回数} = \frac{1}{2}N(N-1) \tag{4.2}$$

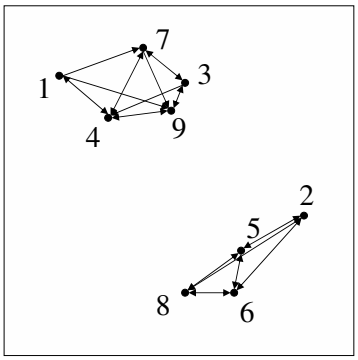


図 4.1 分離した k 最近傍グラフの例

したがって，k 最近傍グラフの計算オーダーは $O(n^2)$ となり，データ量が多くなると現実的な時間ではインデックスが生成できなくなる問題がある．第 2 に，k 最近傍グラフは連結グラフではないためにグラフをたどっても到達できないノードが存在することになり，結果的に検索精度の低下を招くという問題もある．つまり，k 最近傍グラフでは図 4.1 のようにある程度離れたクラスタが存在すると分離したグラフが生成されてしまい，グラフをたどっても到達することができないノードが存在することになる．

4.3 提案手法

k 最近傍グラフの問題点を軽減するために，k 最近傍グラフの以下の点に着目した．k 最近傍グラフは条件 1 を満たしていないので検索漏れが発生するが，それを許容している．漏れが発生することを許容するならば，k 最近傍グラフを正確に構成する必要はなく，k 最近傍グラフが近似されていれば同等の検索精度が実現可能であろうと想定できる．そこで，k 最近傍グラフの 2 つの課題を軽減するために近似 k 最近傍グラフを提案する．次に近似 k 最近傍グラフの生成方法について述べる．

既存のグラフに 1 ノードずつ無向エッジにより逐一連結することで連結グラフを保証する．また，ノードを追加するときには k 個の最近傍のノードを求める必要があるが，全ノードとの距離を計算した上で k 個の最近傍のノードを求める処理は極めて計算量の多い処理である．そこで，生成過程にあるインデックスを利用して前述の k 最近傍検索 (Algorithm 6) を用いて検索することにより少ない計算量で k 個の最近傍のノードを求める．近似 k 最近傍グラフ生成の手順を Algorithm 7 に示す．ただし，P を登録オブジェクトの集合，Random(G) は G から任意の 1 ノードを返す関数，k は各ノードからエッジで結合される近傍のノードの数である．

Algorithm 6 KnnSearch(G, q, s, r, k_s, R, C)

```
for all p ∈ N(G, s) do
  if p ∉ C then
    C ← C ∪ {p}
    if d(p, q) ≤ r then
      R ← R ∪ {p}
      if |R| > k_s then
        R ← R − {argmax_{x∈R} d(x, q)}
      end if
      if |R| = k_s then
        r ← max_{x∈R} d(x, q)
      end if
    end if
    r_s ← (1 + ε)r
    if d(p, q) ≤ r_s then
      KnnSearch(G, q, p, r, k_s, R, C)
    end if
  end if
end for
```

図 4.2 に近似 k 最近傍グラフの例を示す．図中の数字は追加順序を示す．図 4.1 のように k 最近傍グラフでは非連結グラフが生成されるのに対して近似 k 最近傍グラフでは連結グラフが生成される．近似 k 最近傍グラフは無向エッジを逐一追加するので，たとえば 2 つのノードのみが追加された時点では，第 1 のノードには 1 本のエッジしか存在しないが，順次ノードを追加していく上で，第 1 のノードの近傍にノードが追加されると無向エッジが第 1 のノードにも追加されることになり，結果的に各ノードは近傍のノードへのエッジが付与されることになる．実際には 1 つのノードから k 本以上のエッジが生成されることになり，各ノードのエッジ数は k とはならないので，k 本のエッジを有する k 最近傍グラフとはエッジ数では異なるが，k 最近傍グラフと同様に各ノードから近傍のエッジが生成されることになる．なお，初期に追加したノードにはエッジが多く付与されることになるが，全体のエッジ数は kn となる．このように生成された近似 k 最近傍グラフは

Algorithm 7 Create(G, P, k)

```
for all q ∈ P do
  if G = ∅ then
    G ← {q}
  else
    s ← Random(G)
    s ← NearestNeighborSearch(G, q, s)
    r ← ∞
    R ← ∅
    C ← ∅
    KnnSearch(G, q, s, r, k, R, C)
    G ← G ∪ {q}
    N(G, q) ← R
    for all p ∈ R do
      N(G, p) ← N(G, p) ∪ {q}
    end for
  end if
end for
```

k 最近傍グラフとは同一のグラフ構造にはならない．しかし，近似 k 最近傍グラフは少なくとも k 個の最近傍のノードに接続しているので，k 最近傍グラフの性質を有している．したがって，k 最近傍グラフの代替インデックスとして近似 k 最近傍グラフを利用できると考える．一方，k 個の近傍ノードは近似されているので誤った近傍ノードへの接続により検索精度が低下する可能性がある反面，k 最近傍グラフの問題点であるグラフの非連結性が解消されることによる検索精度の向上が期待できる．

4.4 評価実験

本章では，k 最近傍グラフおよび近似 k 最近傍グラフに適用できる範囲検索および k 最近傍検索のアルゴリズムを定義した上で近似 k 最近傍グラフを提案した．そこで，まず，本章で定義した範囲検索および k 最近傍検索を用いた場合の次元，探索半径係数，距離計算回数の関係を示し，範囲検索および k 最近傍検索の有効性を確認する．次に，k 最近傍

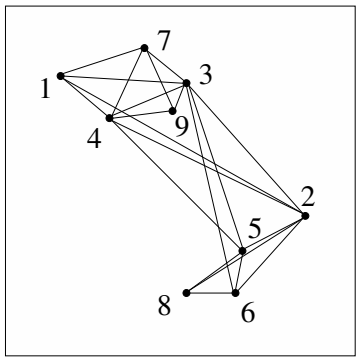


図 4.2　近似 k 最近傍グラフの例

グラフと近似 k 最近傍グラフの生成時の計算量の比較実験を行う．さらに，一様分布データにより両者および従来手法との検索時の計算量の比較実験を行い，最後に，画像特徴量による両者の検索時の計算量の比較実験を行う．

評価に使用した一様分布データと画像特徴量は，いずれもデータ数は 10 万件である．一様分布データは範囲 [0.0,1.0] の浮動小数点の要素からなる次元数 10 から 500 の多次元データであり，L_2 距離を距離関数として用いた．一様分布データでは，すべてのオブジェクトが空間上で均一に分布している．つまり，オブジェクトが最も疎な分布となるので，空間インデックスの効果が最も現れにくい分布となる．したがって，一様分布データの場合には比較的少ない次元数（20,50）を主に利用して評価する．また，画像共有サイトの flickr[21] から取得した画像データから抽出した色ヒストグラム，色やエッジの分布，テクスチャで構成される画像特徴量をオブジェクトして用いた．特徴量の総次元数は 1,228 であり，各次元は 1 バイトの整数型である．式 4.3 のように各特徴量の距離（色差や L_1 距離など）の加重平均を距離関数として用いた．

$$d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{n} \sum_{i}^{n} w_i d_i(\boldsymbol{x}_i, \boldsymbol{y}_i) \tag{4.3}$$

なお，$\boldsymbol{x}_i$，$\boldsymbol{y}_i$ は i 番目の単一特徴量，$\boldsymbol{x}$，$\boldsymbol{y}$ はそれぞれ $\boldsymbol{x}_i$，$\boldsymbol{y}_i$ で構成される複合特徴量，$d_i(\boldsymbol{x}_i, \boldsymbol{y}_i)$ は i 番目の単一特徴量の距離関数，w_i は i 番目の単一特徴量の重付け，n は単一特徴量の総数である

範囲検索および k 最近傍検索の検索精度の測定では，評価ごとに検索を 50 回試行して各指標の平均値を求めて評価した．範囲検索では検索数が 20 件となる検索半径をあらかじめ決定し検索を行った．これは，同一の検索数を保証することによって，次元数を変動させた場合の比較評価を可能とするためである．また，k 最近傍検索の検索結果のノード

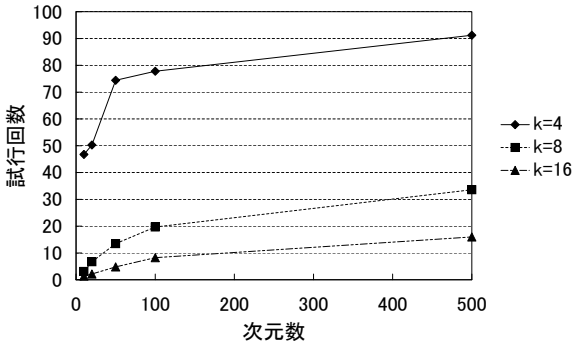


図 4.3 k 最近傍グラフにおけるエッジ数 k に対する次元数と最近傍検索試行回数の関係

数 k_s は 20 とした.

4.4.1 一様分布データによる範囲検索と k 最近傍検索の有効性の評価

k 最近傍グラフを用いて一様分布データに対する範囲検索 (Algorithm 5) と k 最近傍検索 (Algorithm 6) のアルゴリズムの有効性を評価した．前述のように範囲検索では，まず，検索範囲内のノードが見つかるまで任意に選択した探索起点ノードを用いて最近傍検索 (Algorithm 4) を繰り返す．検索範囲内のノードが見つかったなら，そのノードを探索起点ノードとして範囲検索を行う．エッジ数 k ごとに探索半径係数を 0.0 から 0.1 刻みで 0.5 まで変化させて範囲検索したときの最近傍検索の試行回数の平均を図 4.3 に示す．エッジが少ない場合には条件 1 を満たすノードが減り，その結果，範囲内のノードが見つからず，最近傍検索の試行回数が増加している．また，より多くのノードが条件 1 を満たすには次元数が増えるに従いエッジ数を多くしなければならない．同一エッジ数では次元数が増えると検索精度は低下する．したがって，同一エッジ数では次元数が増えるに従い最近傍検索の試行回数が増加することになる．

範囲検索の評価のために，50 次元のデータにおいて探索半径係数を 0.0 から 1.0 へ徐々に増やし各検索における検索精度と距離計算回数を求めて，検索精度が 0.98 を超えた時点で測定を終了した．以降の検索精度の測定は同様の方法で行なった．以下に検索精度の指標である再現率と適合率の算出式を示す．なお，正答数は検索されるべき検索結果の数である．

$$\text{再現率} = \frac{\text{検索結果内の正答数}}{\text{全正答数}} \tag{4.4}$$

$$\text{適合率} = \frac{\text{検索結果内の正答数}}{\text{検索結果数}} \tag{4.5}$$

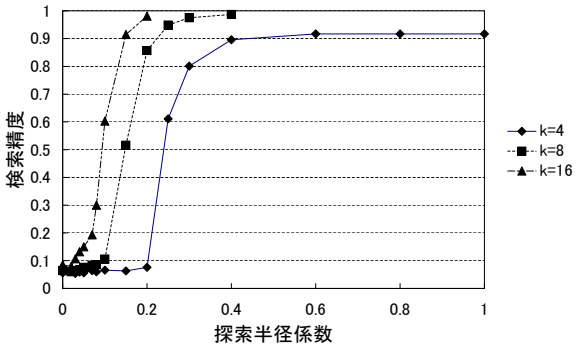


図 4.4　k 最近傍グラフにおけるエッジ数 k に対する探索半径係数と検索精度の関係

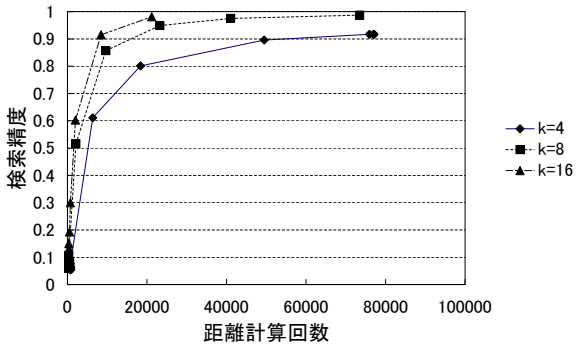


図 4.5　kNNG におけるエッジ数 k に対する距離計算回数と検索精度の関係

範囲検索の場合には，アルゴリズム上，検索結果に誤検索は含まれず，適合率は常に 1.0 となる．また k 最近傍検索の場合には，全正答数と検索結果数はいずれも k となるので，適合率と再現率は常に同率となる．したがって，検索精度の指標として再現率を用いることとし，本論文では検索精度は再現率を示す．

各エッジ数 k に対する探索半径係数と検索精度の関係を図 4.4 に示す．探索範囲が広がれば検索精度が向上することは自明であるが，エッジ数が増加しても検索精度が向上する．つまり，エッジ数が増えることにより，近傍のノードを網羅的に探索することになるので検索精度が向上する．いい換えれば，エッジ数が増加することにより条件 1 を満足するノードが増加するので検索精度が向上する，ともいえる．エッジ数が 4 の場合には探索半径係数が増加しても検索精度が頭打ち状態となっている．これは k 最近傍グラフではエッジの連結性を保証していないので，エッジ数が少ない場合に検索されるべきノードへの経路自体がなくて到達できない状況が発生することに起因すると考えられる．図 4.4 から判断すると検索精度を上げるには単純に探索半径係数を大きくすればよいことになるが，検索時の計算量を考慮する必要がある．高速なメモリ上に配置されたオブジェクトにおける検索時の計算量は，ほぼ距離計算量となる．そこで，検索性能を評価するために，図 4.4 の測定結果の距離計算回数と検索精度の関係を図 4.5 に示す．自明ではあるが距離計算回数が増加するにしたがい，検索精度が向上している．また，エッジ数の多い方が距離計算回数は少なく抑えられている．

エッジ数が 8 における各次元数 d に対する距離計算回数と検索精度の関係を図 4.6 に示す．次元数が増えるに従い距離計算回数が増えている．次に k 最近傍検索でのエッジ数 8 における各次元数 d に対する距離計算回数と検索精度の関係を図 4.7 に示す．k 最近傍検索は範囲検索とほぼ同様の傾向を示している．

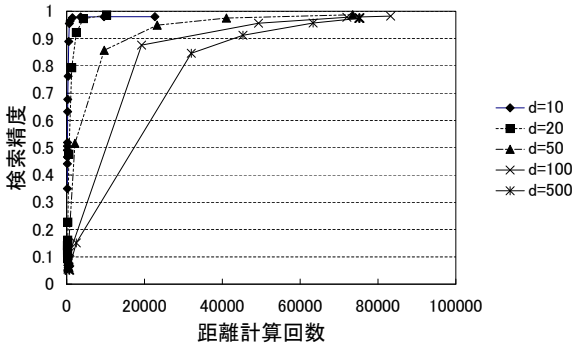


図 4.6 kNNG の範囲検索における次元数 d に対する距離計算回数と検索精度の関係

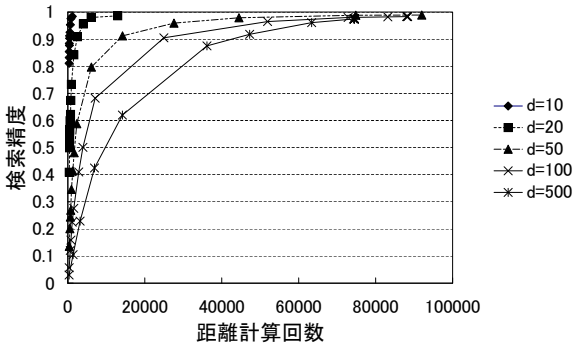


図 4.7 kNNG の k 最近傍検索における次元数 d に対する距離計算回数と検索精度の関係

これらの結果から k 最近傍グラフでは次元数が大きい場合でも本章で述べた範囲検索や k 最近傍検索のアルゴリズムを用いれば距離計算回数を抑えつつ検索可能であると判断できる．また，本評価で用いた次元数より，さらに大きい次元数の場合にもエッジ数を増やすことで距離計算回数を抑えつつ検索が可能であると予想される．

4.4.2 インデックス生成時の計算量の評価

近似 k 最近傍グラフの生成ではインデックスに逐一ノードを追加する．追加時に近傍のノードへのエッジを生成するためにインデックスを用いて k 最近傍検索を行うので，検索時の計算量が少なければ少ないほど，インデックスの生成時の計算量も少なくなるという特徴を近似 k 最近傍グラフは有する．そこで，k 最近傍グラフと近似 k 最近傍グラフのインデックスの生成時の計算量を比較する．

グラフのエッジ数は検索精度と密接に関係するので，インデックスを比較するにはインデックス全体のエッジ数を同数にする必要がある．k 最近傍グラフでは k 個の最近傍のノードへの有向エッジを生成するので，n 個のノードを登録するとインデックス全体では kn 本の有向エッジが生成される．近似 k 最近傍グラフは生成時に検索数 j の k 最近傍検索により j 本の無向エッジが生成される．有向エッジで表現すると $2j$ 本の有向エッジが生成され，インデックス全体では $2jn$ 本の有向エッジが生成されることになる．つまり，k 最近傍グラフのエッジ k 本に対して近似 k 最近傍グラフでは無向エッジ $k/2$ 本を生成することにより，全体では同数の有向エッジで構成されるグラフが生成されることになる．したがって，本評価ではたとえばエッジ数 $k = 4$ とする近似 k 最近傍グラフとは，生

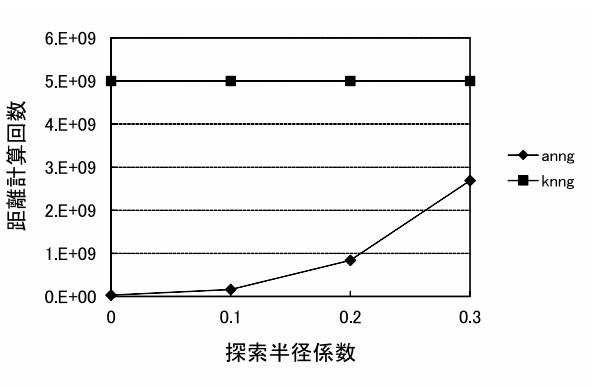


図 4.8　インデックス生成時の探索半径係数と距離計算回数の関係

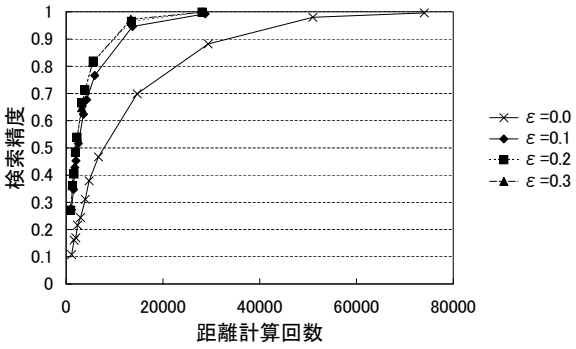


図 4.9　近似 k 最近傍グラフにおけるインデックス生成時の探索半径係数に対する距離計算回数と検索精度の関係

成時に検索数を 2 とする k 最近傍検索を行って生成した近似 k 最近傍グラフを意味する．

k 最近傍グラフと近似 k 最近傍グラフの次元数 50，エッジ数 8 における一様分布データに対するインデックス生成時の探索半径係数と距離計算回数の関係を図 4.8 に示す．図中の knng は k 最近傍グラフ，anng は近似 k 最近傍グラフを示す．以降，図表中ではこの短縮表記を主に用いる．k 最近傍グラフの距離計算回数は式 4.2 により求められるので，探索半径係数の値に関わらず一定値となる．近似 k 最近傍グラフでは探索半径係数が増加すれば距離計算回数も増加しているが，k 最近傍グラフと比較すると極めて少ない距離計算回数でインデックスが生成できる．

近似 k 最近傍グラフでは前述のようにインデックス生成時に生成途中のインデックスを用いて追加ノードを問合せオブジェクトとして k 最近傍検索を行い，得られた近傍ノードに追加ノードをエッジで接続することでグラフに追加する．したがって，インデックス生成時の k 最近傍検索の探索半径係数の違いはインデックスの生成精度に影響し，最終的に生成されたインデックスを用いた検索精度にも影響する．そこで，インデックス生成時の k 最近傍検索の探索半径係数 ϵ が異なるインデックスに対する k 最近傍検索時の距離計算回数と検索精度の関係を図 4.9 に示す．探索半径係数が 0.0 と 0.1 では検索精度に大きな違いが生じているが 0.1 と 0.2 では大差はなく 0.2 と 0.3 ではほぼ同値になっていることから，探索半径係数は 0.1 が適当と判断する．また，図 4.8 において探索半径係数が 0.1 の場合にはインデックス生成時の距離計算回数は k 最近傍グラフに対して近似 k 最近傍グラフは約 3.3 %でしかない．つまり，約 96.7% の削減が可能であり，近似 k 最近傍グラフの生成時の計算量は極めて少ない．

次元数 50，探索半径係数 0.1 においてエッジ数 k に対するインデックス生成途中にお

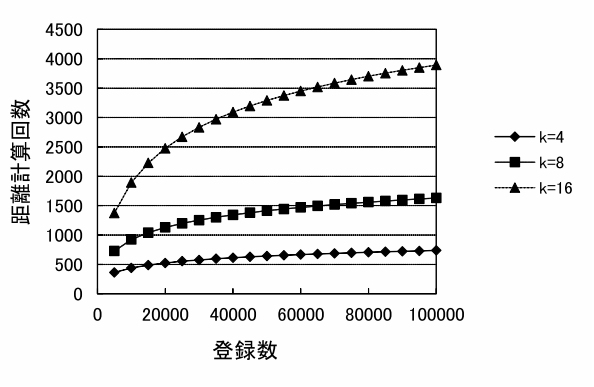


図 4.10 近似 k 最近傍グラフにおけるエッジ数に対する登録数と 1 ノード登録時の距離計算回数の関係

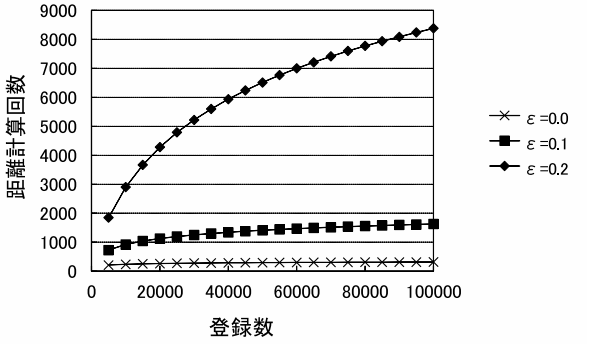


図 4.11 近似 k 最近傍グラフにおける探索半径係数に対する登録数と 1 ノード登録時の距離計算回数の関係

ける 1 ノード登録時の距離計算回数を図 4.10 に示す．また，次元数 50，エッジ数 8 において探索半径係数 ϵ に対する登録途中の 1 ノード登録時の距離計算回数を図 4.11 に示す．これらの図から登録数が多いほど 1 ノードの登録時における距離計算回数はそれほど増えないことが示されている．式 4.2 より k 最近傍グラフの計算オーダーは $O(n^2)$ に対して，登録数が多い場合には近似 k 最近傍グラフは $O(n)$ に近づくと判断できる．したがって，大量のデータに対してインデックスを生成する場合にも距離計算回数を少なく抑えることが可能であると考えられる．

4.4.3 一様分布データによる検索時の計算量と検索精度の評価

次元数 10 と 20 における近傍ノードへのエッジ数 k ごとの k 最近傍グラフと近似 k 最近傍グラフの最近傍検索 (Algorithm 4) の検索精度を表 4.1，表 4.2 に示す．検索精度は最近傍検索を 100 回試行した結果のうちの正答数の割合である．距離計算回数は k 最近傍グラフより近似 k 最近傍グラフの方が多いものの検索精度は近似 k 最近傍グラフの方が高くなっている．k 最近傍グラフは非連結グラフあることから，最近傍のノードまで到達できずに途中で処理を終了する場合が多いので，距離計算回数が少なく，かつ，検索精度が低く抑えられていると考えられる．一方，近似 k 最近傍グラフは連結グラフであることから最近傍のノードへ到達できる場合が多くなり，その分，距離計算回数が増えて検索精度も高くなっていると考えられる．また，表 4.1 のエッジ数 $k = 4$ の場合には k 最近傍グラフの検索精度が近似 k 最近傍グラフより上回っている．これはエッジ数が少ない場合には，近傍のノードにエッジが正確に付与されている k 最近傍グラフの方が効果的に探索

表 4.1　10 次元での距離計算回数と検索精度

	kNNG		ANNG	
k	距離計算回数	検索精度	距離計算回数	検索精度
4	12.9	0.03	20.9	0.00
8	39.1	0.07	61.4	0.08
16	92.4	0.33	151.7	0.53

表 4.2　20 次元での最近傍検索の距離計算回数と検索精度

	kNNG		ANNG	
k	距離計算回数	検索精度	距離計算回数	検索精度
4	12.7	0.00	35.5	0.00
8	33.8	0.01	74.9	0.08
16	71.1	0.05	154.8	0.13

が可能であることに起因すると考えられる．

k 最近傍グラフと近似 k 最近傍グラフの範囲検索において次元数 50 での各エッジ数に対する距離計算回数と検索精度の関係を図 4.12 に示す．図中の k 最近傍グラフおよび anng に続く数値はエッジ数を意味する．k 最近傍グラフは連結性が保証されていないのでエッジ数が少ない場合には検索精度が低くなるが，エッジ数が多い場合には連結性は保証されないが，孤立したグラフが少なくなり，その結果，検索精度の差異がほぼなくなると考えられる．同一の条件で k 最近傍検索時の k 最近傍グラフと近似 k 最近傍グラフの距離計算回数と検索精度の関係を図 4.13 に示す．k 最近傍検索は範囲検索とほぼ同様の傾向を示している．

図 4.13 の距離計算回数が少ない部分のみを図 4.14 に示す．距離計算回数が少ない場合には k 最近傍グラフの方が若干精度が高い．k 最近傍グラフの方が近似 k 最近傍グラフより近傍のノードへのエッジが精度良く生成されているので，距離計算回数が少ない場合，つまりは，探索範囲が狭い場合には精度が高くなる傾向がある．しかし，距離計算回数が多い場合，つまりは，探索範囲が広く網羅的にグラフを探索する場合には連結グラフではないことにより検索精度が伸びないと考えられる．

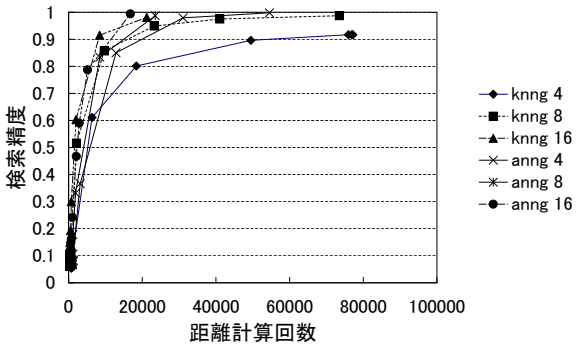


図 4.12 範囲検索時の距離計算回数と検索精度の関係

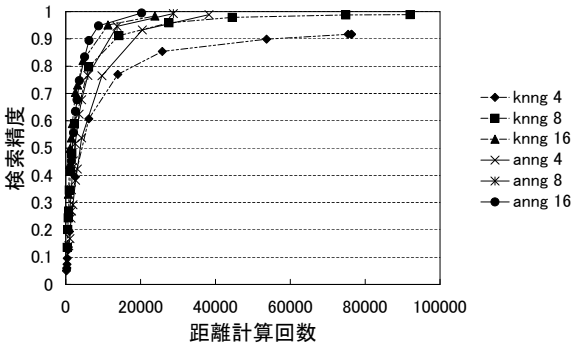


図 4.13 k 最近傍検索時の距離計算回数と検索精度の関係

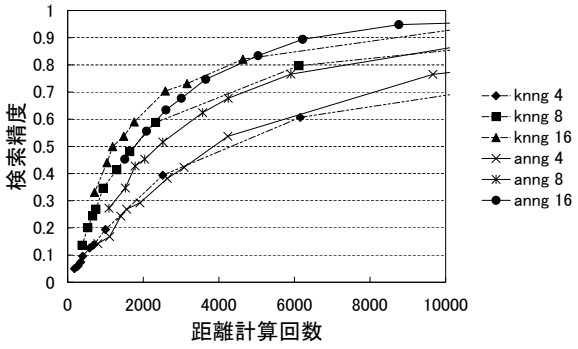


図 4.14 k 最近傍検索時の距離計算回数と検索精度の関係

4.4.4 一様分布データによる各種従来手法の検索時の計算量の評価

20 次元と 50 次元の一様分布データに対して既存の距離基準空間インデックスとの比較評価を行った．比較したインデックスは Metric Spaces Library [*1] で提供されている LAESA，sa-tree，mvp-tree である．20 次元および 50 次元に対する k 最近傍検索時の各インデックスの距離計算回数を図 4.3 および図 4.4 に示す．LAESA に関してはピボット数を 6 として測定した．LAESA では各オブジェクトとピボット間の距離がインデックスとなる．mvp-tree に関しては木構造上のノードの分岐数が 2 と 6 の場合について測定した．木構造上の各ノードのピボット数は 1 である．なお，mvp-tree におけるピボットとは木構造を構成する各ノードの部分空間を分割するための基準となるオブジェクトのこと

[*1] http://www.sisap.org/Metric_Space_Library.html

表 4.3　20 次元での距離計算回数と検索精度

インデックス	LAESA	sa-tree	mvp 2	mvp 6
距離計算回数	89,706.4	91,532.3	100,000.0	100,000.0
検索精度	1.0	0.937	1.0	1.0

表 4.4　50 次元での距離計算回数と検索精度

インデックス	LAESA	sa-tree	mvp 2	mvp 6
距離計算回数	99,977.2	99,999.9	100,000.0	98,922.0
検索精度	1.0	1.0	1.0	1.0

である．いずれのインデックスもほぼ全件の距離計算を行っている．なお，sa-tree のみ検索漏れが生じる可能性のある近似検索なので，20 次元のデータでは検索精度が 1.0 に達していない．

LAESA についてはピボットを増やせば距離計算回数が減るので，ピボット数に対する距離計算回数を測定し，その結果を図 4.15 に示す．なお，LAESA のピボットは距離計算の基準となるノードのことである．20 次元のオブジェクトの場合には，ピボット数が 5,000 ぐらいまではピボット数が増加するにつれ距離計算回数が減少するが，それ以上になると逆に距離計算回数が増加する．ピボット数が 5,000 のときに距離計算回数は最少である約 20,000 となる．50 次元のオブジェクトの場合には，つねに，90,000 を超えている．近似 k 最近傍グラフでは 50 次元の場合には図 4.13 のようにエッジ数 $k = 16$ であれば検索精度 0.995 のときに距離計算回数は約 20,000 となり近似 k 最近傍グラフの距離計算回数が LAESA より大幅に少ないことがわかる．しかも，ピボット数を 5,000 としても LAESA は約 5 億の距離データをインデックスとして保持しなければならないが，それに対して近似 k 最近傍グラフは約 160 万のエッジのデータで十分であり，近似 k 最近傍グラフは LAESA に対してインデックスのサイズをデータ数として比較すると約 0.32 %でしかない．

以上の結果より，近似 k 最近傍グラフは近似検索ではあるものの，既存手法の距離基準空間インデックスよりも検索時の距離計算回数に関して優位であることを確認した．

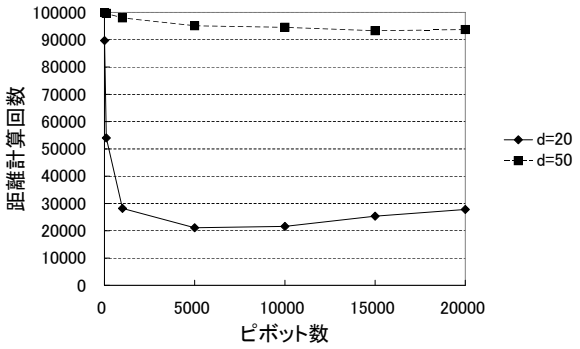


図 4.15 LAESA におけるピボット数と距離計算回数の関係

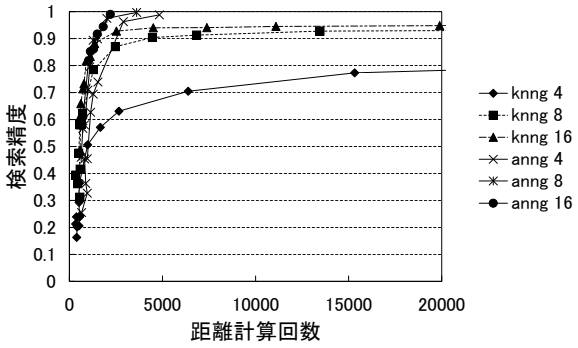


図 4.16 画像特徴量の範囲検索における距離計算回数と検索精度の関係

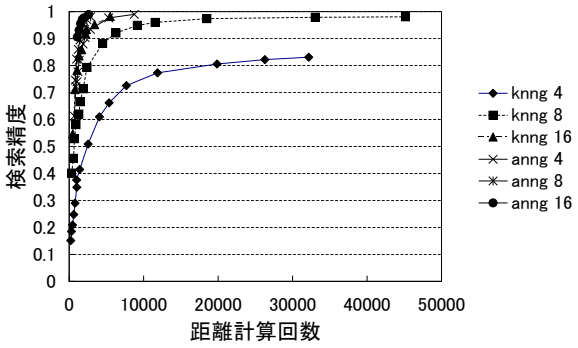


図 4.17 画像特徴量の k 最近傍検索における距離計算回数と検索精度の関係

4.4.5 画像特徴量による検索時の計算量と検索精度の評価

実際の画像特徴量のオブジェクトによる範囲検索の結果を図 4.16 に，k 最近傍検索による結果を図 4.17 に示す．利用した画像特徴量は 1,000 次元を超えるが一様分布のデータと比較すると距離計算回数がかなり抑えられている．画像特徴量では次元数が多くても特徴量空間上で分布に偏りがあり，インデックスの効果が高いと考えられる．評価に用いたデータの各次元が 1 バイトの整数型に量子化されていることも一因と考えられる．これらの結果から，近似 k 最近傍グラフは実データの検索においても k 最近傍グラフに対して優位であることを確認した．

4.5　本章のまとめ

多次元データの近傍検索を実現するグラフ構造型のインデックスとして k 最近傍グラフを用いる手法が提案されている．これは，各ノードに対する k 個の最近傍のノードへのエッジによりグラフが構成されるが，

- インデックス生成時の膨大な計算量
- 非連結グラフであることに起因する検索精度の抑制

といった問題がある．そこで，近似 k 最近傍グラフを提案した．近似 k 最近傍グラフでは生成中の近似 k 最近傍グラフを用いて k 最近傍ノードを検索することでインデックスの生成時の距離計算回数を大幅に削減できる．実際に，10 万オブジェクトの 50 次元の一様分布データにおいて k 最近傍グラフに対して提案手法では約 96.7% の距離計算回数を削減できることを確認した．また，ノードを無向エッジにより逐一グラフに追加することで連結グラフとなり，近似 k 最近傍グラフでは非連結グラフに起因する検索精度の抑制を解消することができ，検索精度が向上した．

第 5 章

木構造型インデックスを用いた近似 k 最近傍グラフ

5.1 本章の目的

高次元のオブジェクトにおいて検索時の計算量の削減には k 最近傍グラフをインデックスとして用いる手法が有効である．しかし，k 最近傍グラフでは生成時の計算量が多く，かつ，非連結グラフにより検索精度が抑制される，という課題があった．そこで，これらの課題を軽減する近似 k 最近傍グラフを提案した．近似 k 最近傍グラフではグラフをたどり検索条件を満たすノードを探索するが，任意に選択したノードからグラフの探索を開始するのでノードが増えるほど検索時の計算量が増大する傾向がある．そこで，本章では探索を開始するノードとして問合せオブジェクトの近傍ノードを効率よく取得できるように木構造型インデックスを利用する方法を提案する．

5.2 課題

近似 k 最近傍グラフでは検索時に次に示す 2 段階の処理を行う．

1. 任意に選択したノードを起点にグラフ上で最良優先探索により問合せオブジェクトに対する単一の近傍ノードの探索を行う．
2. その近傍ノードから網羅的にグラフをたどりつつ，検索条件を満足するノードを探索し検索結果を得る．

1 段目の処理において，登録数が少ない場合には問題は生じないが，登録数に比例して起点から近傍ノードへの経路が長くなり，結果として探索の計算量が増大する．2 段目の探索がグラフ上において局所的な処理なので登録ノード数に依存しないことからも，登録数が増大すると 1 段目の計算量が無視できない状況となる．また，任意に選択したノードが偶然にも問合せオブジェクトから経路として遠方である場合には探索時の計算量が増大するので，検索ごとに計算量が極端に増減する傾向がある．また，近似 k 最近傍グラフでは前述のようにインデックス生成時に検索処理を行う．したがって，検索時の計算量の低減はインデックス生成時の計算量の低減にも寄与するので，検索時の計算量の改善は重要である．

5.3 提案手法

近似 k 最近傍グラフの検索における 1 段目の処理は最近傍検索に他ならないので，何らかのインデックスを用いることにより効率良く検索できると考えられる．ただし，そのインデックスは以下の条件を満足しなければならない．

1. 近似 k 最近傍グラフは距離基準空間インデックスなので，距離基準空間インデックスでなければならない．
2. 近似 k 最近傍グラフが高次元データを対象とするので，高次元データを対象とするインデックスでなければならない．
3. 近似 k 最近傍グラフにデータを登録するときには 1 データごとに近似 k 最近傍グラフに登録しては検索できる必要がある．つまり，追加登録が可能な動的なインデックスでなければならない．

以上の条件のうち，2 つ目の高次元データを対象とするインデックスとは，本論文の最大の目的とするところであり，このようなインデックスはないといってもよい．しかし，近似最近傍が取得できればよいのであれば高次元データの影響を受けにくいインデックスが存在する．

たとえば，木構造型インデックスでは高次元データの場合において探索枝の削減ができなくなる．木構造型インデックスでは各ノードに対応する部分空間が重複するインデックスと重複しないインデックスが存在する．重複するインデックスでは，ルートノード（木構造の起点となるノード）からリーフノード（下位ノードが存在しない末端のノード）へたどる場合には重複するすべてのインターナルノード（下位ノードが存在するノード）を探索しなければならず，高次元データの場合にはこの傾向が顕著になる．しかし，重複しな

いインデックスではルートノードからたどる場合に，つねに各インターナルノードの枝から 1 つの枝をたどればよいので，高次元データであっても影響を受けない．ただし，到達したリーフノードの部分空間は必ず問合せオブジェクトを包含するが，かといって必ずしも最近傍を含むわけではない．しかしながら，検索処理の 1 段目の最近傍探索の目的は 2 段目の起点となるノードを探索するためなので，必ずしも正しい最近傍が得られる必要はない．したがって，第 2 の条件である，高次元データを対象とするインデックス，は，ノード空間の重複がない木構造型インデックス，にいい換えることができる．本章では，これらの条件を満足するインデックスとして第 3 章で提案した dvp-tree を用いることとした．なお，近似 k 最近傍グラフによる提案手法，と，近似 k 最近傍グラフに dvp-tree を加えた提案手法，とを区別するために以降，前者を ANNG (Approximate k-Nearest Neighbor Graph)，後者を ANNGT (Approximate k-Nearest Neighbor Graph with Tree) とよぶこととする．次に dvp-tree を近似 k 最近傍グラフで利用する方法を述べる前に，再度 vp-tree および dvp-tree について簡便に述べる．

vp-tree はあらかじめすべてのオブジェクトが与えられることを前提とし，vantage point とよばれるオブジェクトを中心とした超球により空間を分割することで木構造を生成する．まず，超球の内側と外側の部分空間にすべてのオブジェクトがそれぞれ同数に分配されるように空間を分割する．さらにその各部分空間を新たな超球により同様に分割しオブジェクトを分配する．この処理を再帰的に行うことで木構造の vp-tree を生成する．単一の vantage point に対して 2 つ以上の異なる半径により 3 つ以上の部分空間に分割することも可能である．各部分空間は vp-tree のノードに対応し，リーフノードのみにオブジェクトが属することになる．一方，dvp-tree では初期時には全空間に対応する単一のリーフノードしかない．オブジェクトを追加するときに，まず木構造をたどり対応するリーフノードを求めて，そのリーフノードにオブジェクトを追加する．リーフノードに属するオブジェクトの数が最大数（リーフノードに属することができるオブジェクトの最大個数）を超えた場合には，そのリーフノードの空間を vp-tree と同様に分割してノードを生成することで木構造が成長する．生成方法は異なるが木構造の構成は vp-tree と dvp-tree は同様である．

dvp-tree のインターナルノードの分岐数を 2，リーフノードに関連付けられるグラフのノードの最大数を 3 とした ANNGT のデータ構造を図 5.1 に示す．グラフ構造の近似 k 最近傍グラフの各ノードが dvp-tree のリーフノードに関連付けられている．図 5.1 における dvp-tree によるリーフノードの部分空間とグラフの関係を図 5.2 に示す．円弧で分割されている領域がリーフノードの部分空間を示し，直線がグラフを示している．なお，円弧上のオブジェクトは内側の空間に属するとしている．各リーフノードは部分空間に対

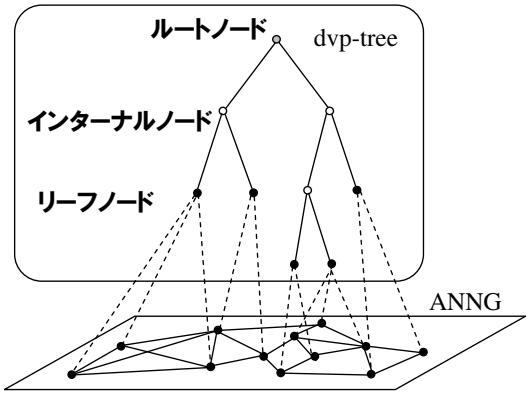


図 5.1　ANNGT のデータ構造

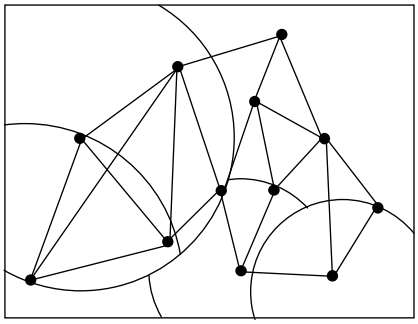


図 5.2　dvp-tree による部分空間

応付けられており，その部分空間に属するノードがリーフノードに関連付けられている．したがって，検索時にはルートノードより問合せオブジェクトを包含するノードを順次たどることでリーフノードに達し，リーフノードに関連付けられているノード集合が問合せオブジェクトに対する近傍ノード集合となる．そのノード集合を用いて 2 段目の処理を行う．ANNG では単一の近傍ノードから 2 段目の探索を行うが，ANNGT では複数のノードを起点として 2 段目の探索行うこととする．範囲検索において探索範囲が小さい場合には，リーフノードにより得られたノードがいずれも範囲内に属さない場合もありえるので，このような場合には，範囲外となったノードから ANNG の 1 段目の処理と同様に最良優先探索により範囲内の近傍ノードを探索する．

ANNGT のインデックスの生成では，前述の検索により登録オブジェクトの近傍ノードを探索し，その近傍ノードへのエッジおよび登録オブジェクトを示すノード（登録ノード）をグラフに登録する．この処理は ANNG の登録処理と同一であり，第 4 章の Algorithm 7 において，KnnSearch に代わって ANNGT の k 最近傍検索を呼び出すことになる．さらに dvp-tree において登録オブジェクトを包含するリーフノードに登録オブジェクトを追加する操作が必要となる．追加すべきリーフノードは登録時における dvp-tree による検索で，すでに特定されているのでそのリーフノードに登録オブジェクトを追加する．リーフノードへの追加の処理は前述の dvp-tree の登録処理と同一である．

5.4 評価実験

一様分布データを用いたインデックスの生成時の計算量および検索精度，そして，実際の画像特徴量を用いた検索精度に関して k 最近傍グラフ，ANNG，ANNGT の比較評価を行った．評価に使用した一様分布データと画像特徴量のオブジェクト数はいずれも 10 万件である．一様分布データは範囲 [0.0,1.0] の浮動小数点の要素からなる 20 次元の多次元データであり，L_2 距離を距離関数として用いた．画像特徴量については第 4.4 節の評価と同一の特徴量および距離関数を用い，特徴量の総次元数は 1,228 である．

dvp-tree のリーフノードに関連付けられるグラフのノードの最大数は 100 とし，インターナルノードの分岐数は 5 とした．つまり，インターナルノードの空間は vantage point を中心とした 4 つの異なる半径の超球により 5 つの部分空間に分割される．この構成の場合には，リーフノードのオブジェクトの数が 100 を超えるとリーフノードを分割して 20 のオブジェクトを有するリーフノードが 5 つ生成される．したがって，リーフノードに平均 60 のオブジェクトが存在すると仮定できるので，10 万のオブジェクトを格納するには約 1,667 のリーフノードが必要となる．木構造を 5 つに分割しているのでルートノードの層を除いて 5 階層あれば 3,125 のリーフノードが生成可能ある．したがって，5 階層ですべてのオブジェクトを格納可能であり，つまりは，平均 5 回の距離計算でルートからリーフノードへ到達できる．ただし，vp-tree とは異なり dvp-tree はルートノードからリーフノードまでの枝の数が必ずしも一定とはならないので，実際の距離計算の平均は 5 回を多少上回ると予想される．一方，20 次元のデータでエッジ数が 8 のときに ANNG の 1 段目の処理である近傍検索時に約 80 回の距離計算を行うことが予備実験からわかっている．したがって，この構成による dvp-tree を用いれば ANNG の近傍検索に比べると ANNGT は極めて少ない距離計算回数となると予想できる．

範囲検索および k 最近傍検索の検索精度の測定では，評価ごとに検索を 50 回試行して各指標の平均値を求めた．k 最近傍検索の検索結果の検索数を 20 とした．また，範囲検索では検索結果の検索数が k 最近傍検索と同一の 20 件となる検索半径をあらかじめ決定し検索を行った．これは，同一の検索数を保証することによって，範囲検索と k 最近傍検索の比較を可能とするためである．なお，前章で述べたように本章でも検索精度として再現率を利用する．

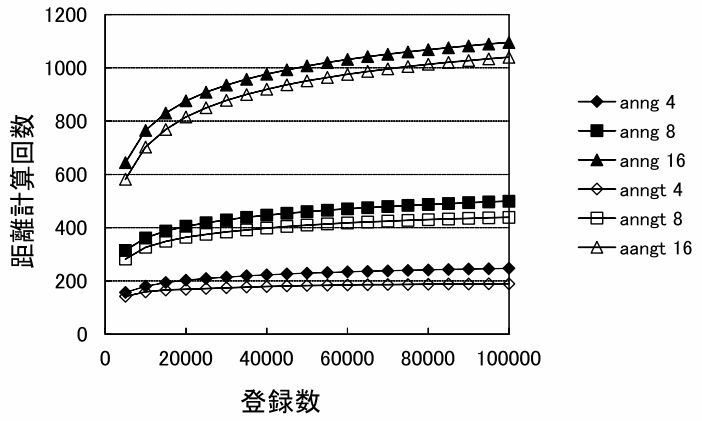


図 5.3　エッジ数に対する登録数と 1 ノード登録時の距離計算回数の関係

5.4.1　インデックス生成時の計算量の評価

グラフのエッジ数は検索精度と密接に関係するので，インデックスを比較するにはインデックス全体のエッジ数を同数にする必要がある．第 4 章で述べたように k 最近傍グラフのエッジ k 本に対して ANNG や ANNGT では無向エッジ $k/2$ 本を生成することにより，全体では同数の有向エッジで構成されるグラフが生成されることになる．したがって，本評価において，たとえばエッジ数 $k = 4$ とする ANNG および ANNGT とは，生成時に検索数を 2 とする k 最近傍検索を行って生成した ANNG および ANNGT を意味する．

動的な追加登録を前提とすると k 最近傍グラフのインデックス生成時にすべてのノード間の距離を求める必要があるので，ノード数に対して距離計算回数が一意に $O(n^2)$ と決まる．一方，ANNG は探索半径係数の値により距離計算回数が変動する．第 4 章で ANNG の探索半径係数 0.1 の場合に ANNG のインデックス生成時の距離計算回数が k 最近傍グラフの約 3.3% しかないことを示した．本評価では ANNG および ANNGT について比較する．次元数 20，探索半径係数 0.1 において各エッジ数のインデックス生成途中における 1 ノード登録時の距離計算回数を図 5.3 に示す．図中の anng および anngt に続く数字はエッジ数を示す．図より両者ともに登録数が多いほど 1 ノードの登録時における距離計算回数の増加傾向は抑えらており，大量のノードを登録できると予想される．また，ANNG に比べ ANNGT の方が距離計算回数が全体にわたって少ないことが示されており，ANNGT がインデックス生成において有効であることを確認した．

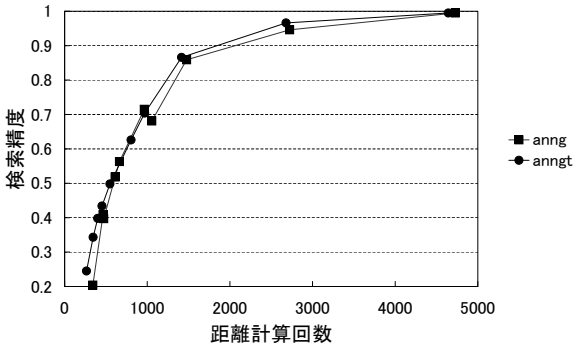


図 5.4 ANNGT のインデックスを用いた範囲検索における距離計算回数と検索精度の関係

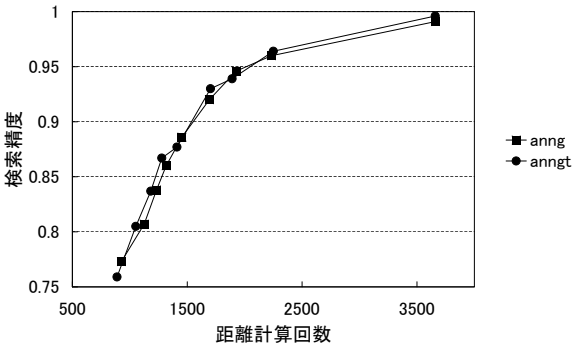
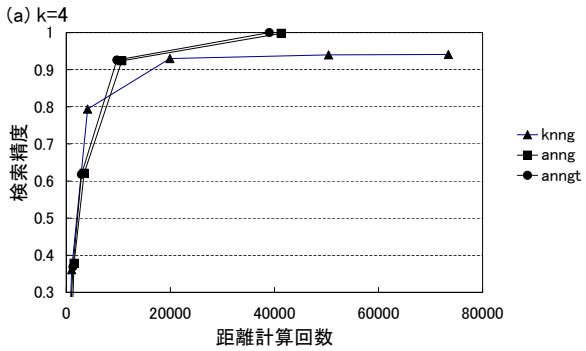


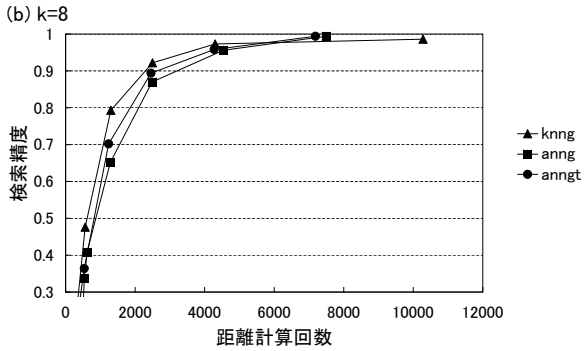
図 5.5 ANNGT のインデックスを用いた k 最近傍検索における距離計算回数と検索精度の関係

5.4.2 一様分布データによる検索時の計算量と検索精度の評価

ANNG および ANNGT のインデックス生成時には k 最近傍検索を行うので，検索の精度が異なるとインデックス生成の精度も異なる．そこで，検索の精度のみを評価するために，ANNGT で生成したインデックスに対して ANNG および ANNGT の検索精度を評価した．比較の方法として，1 段目の処理のみを比較評価する方法も考えられるが，ANNG では，1 段目で単一の最近傍ノードだけを探索し，その単一のノードを 2 段目の探索の起点とする一方，ANNGT では 1 段目で複数の近傍ノードを探索し，その複数のノードを 2 段目の探索の起点とする．したがって，1 段目の処理の結果が 2 段目の処理の距離計算回数にも影響を与えるので，1 段目および 2 段目の全処理を通しての距離計算回数を測定した．探索半径係数を 0.0 から 0.1 刻みで増やし，各探索半径係数における距離計算回数と検索精度を測定した．なお，検索精度が 0.98 を超えたところで測定を終了した．エッジ数が 16 において一様分布データに対する ANNG および ANNGT の範囲検索時の距離計算回数に対する検索精度を図 5.4 に，k 最近傍検索時の距離計算回数に対する検索精度を図 5.5 に示す．図よりほぼ全般にわたってわずかではあるが ANNGT の検索精度が ANNG より上回っており，ANNGT が有効であることを確認した．

つぎに，それぞれの検索方法によりインデックスを生成した場合の評価を行った．エッジ数 $k = 4, 8, 16$ において，一様分布データに対する k 最近傍グラフ (knng)，ANNG および ANNGT の範囲検索時の距離計算回数に対する検索精度を図 5.6 に k 最近傍検索時の距離計算回数に対する検索精度を図 5.7 に示す．範囲検索の検索数を k 最近傍検索の検





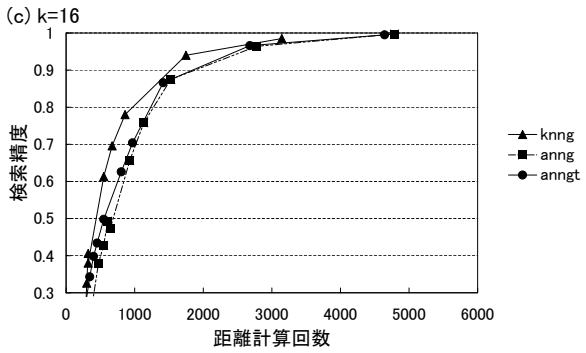
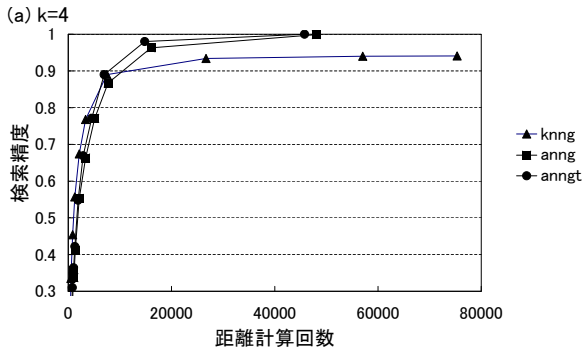
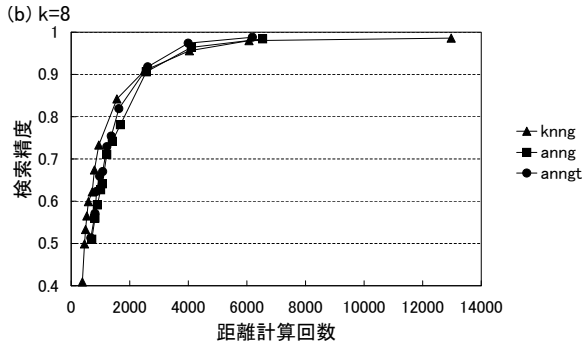


図 5.6　一様分布データの範囲検索における距離計算回数と検索精度の関係





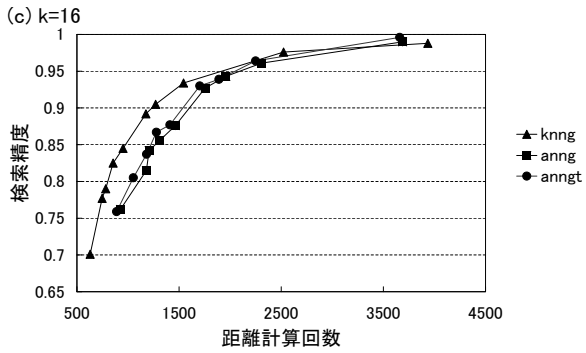


図 5.7　一様分布データの k 最近傍検索における距離計算回数と検索精度の関係

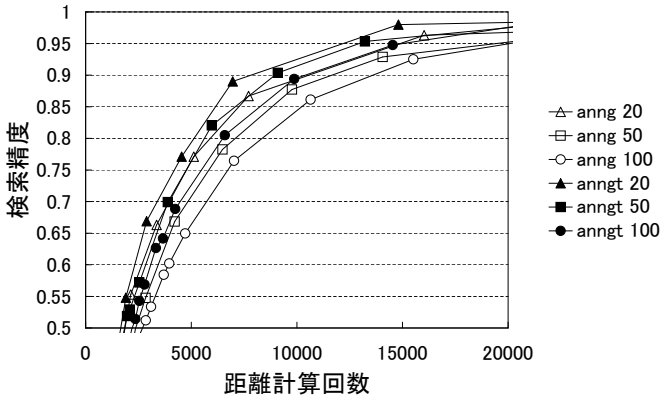


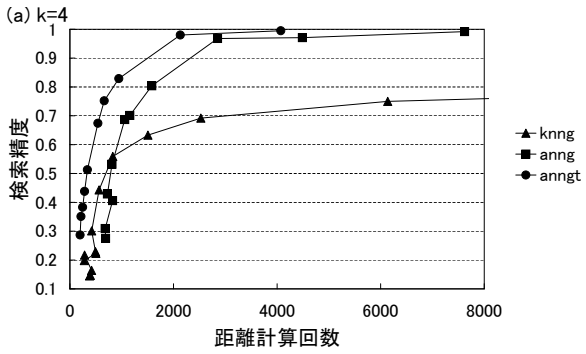
図 5.8 一様分布データの k 最近傍検索における検索数に対する距離計算回数と検索精度の関係

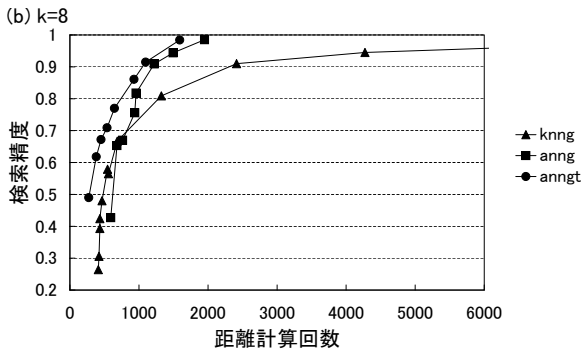
索数と同数としていることもあり，範囲検索と k 近傍検索ではほぼ同様の傾向を示している．k 最近傍グラフはグラフの連結性を保証していないので，エッジ数が少ないほど，たどれないノードが増加し，その結果，距離計算回数が多くても検索精度が上がらない傾向がある．ANNG と ANNGT を比較すると ANNG より ANNGT が全般にわたって距離計算回数が少なく，ANNGT が有効であることを確認した．

さらに，エッジ数が 4 での k 最近傍検索において検索数を変化させた場合の距離計算回数に対する検索精度を図 5.8 に示す．図中の anng および anngt に続く数値が検索数である．検索数が増加すると若干検索精度が低下するが，ANNGT の精度がいずれの検索数でも ANNG を上回っており，検索数に関係なく ANNGT が有効であることを確認した．図示しないがエッジ数が 8 および 16 の場合も同様の傾向を示した．なお，第 4.4 節で従来手法との比較を行い ANNG が従来手法より優位であることを確認しているので本章では言及しない．

5.4.3 画像特徴量による検索時の計算量と検索精度の評価

エッジ数 $k = 4, 8, 16$ において，画像特徴量による範囲検索の結果を図 5.9 に，k 最近傍検索の結果を図 5.10 に示す．利用した画像特徴量は 1,000 次元を超えるが一様分布のデータと比較すると検索精度がかなり向上している．画像特徴量では次元数が多くても特徴量空間上で分布に偏りがあり，インデックスの効果が高いと考えられる．評価に用いたデータの各次元が 1 バイトの整数型に量子化されていることも一因と考えられる．量子化されていることで，任意の 2 つの特徴量の対応する要素を個別に比べた場合に完全一致する要素数が量子化前に比べて極めて多くなる．一致することで距離関数としてはその次元





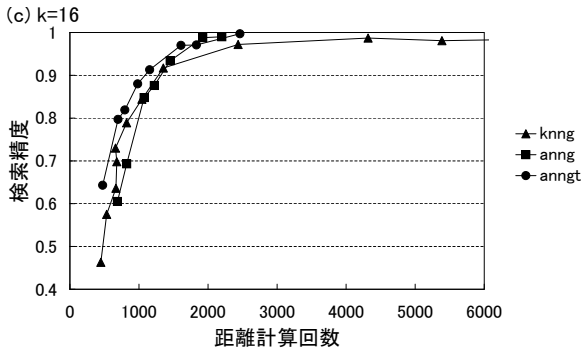
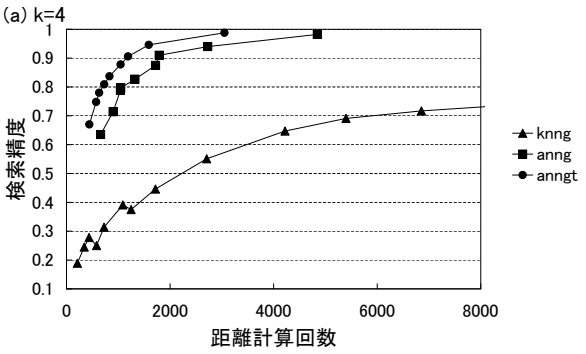
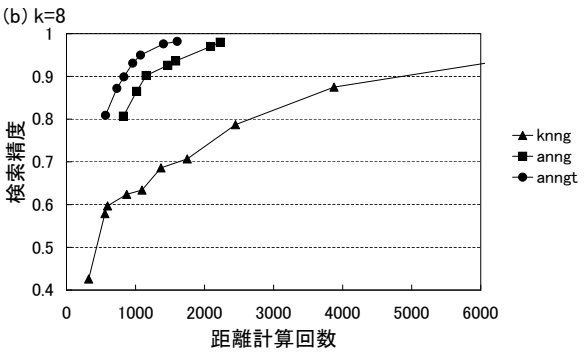


図 5.9　画像特徴量の範囲検索における距離計算回数と検索精度の関係





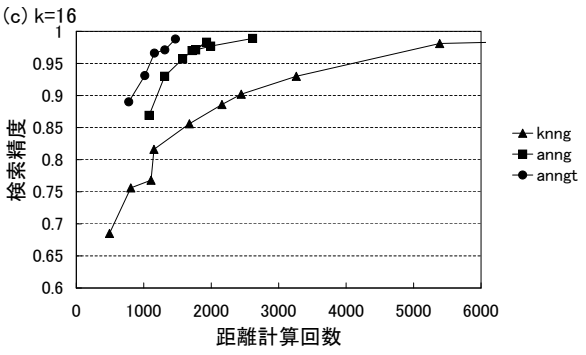


図 5.10　画像特徴量の k 最近傍検索における距離計算回数と検索精度の関係

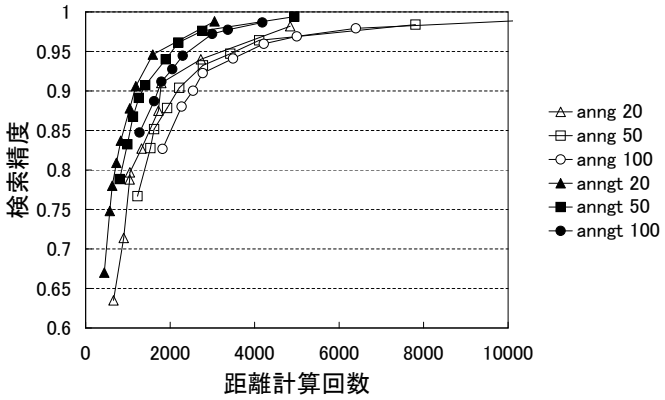


図 5.11 画像特徴量の k 最近傍検索における検索数に対する距離計算回数と検索精度の関係

は無視されるに等しく，一致する要素が多いほど結果的に次元数が削減されるのと同様の効果があると予想する．これらの結果から，ANNGT は実データである画像特徴量の検索においても ANNG よりも有効であることを確認した．

また，範囲検索より k 最近傍検索の方がより ANNGT の効果があることが確認できる．たとえば，(c) の $k = 16$ の場合，範囲検索では 0.95 の検索精度のときに図より距離計算回数が kNNG，ANNG，ANNGT の順に約 2000, 約 1600, 約 1400，k 最近傍検索では約 4100，約 1500，約 1100 である．したがって，kNNG に対して ANNG では範囲検索が約 20%，k 最近傍検索が約 30% の削減だが，ANNGT では範囲検索では約 58%，k 最近傍検索では約 73% 削減できる．k 最近傍検索では，探索範囲の絞込みが速いほど距離計算回数を削減できる．ANNGT では vp-tree により得られるノード数が平均 60 であり，検索数 20 より多いので，2 段目のグラフの探索開始時にすでに探索半径が限定できる．ANNG ではグラフ探索開始時には 1 ノードしか獲得できていないので探索半径が限定できない．一方，範囲検索では探索半径が確定しているので探索範囲の絞り込みの効果が得られない．このことから，範囲検索より k 最近傍検索の方がより ANNGT の効果が現れていると判断する．

エッジ数が 4 での k 最近傍検索において検索数を変化させた場合の距離計算回数に対する検索精度を図 5.11 に示す．図中の anng および anngt に続く数値が検索数である．一様分布データの場合と同様に検索数が増加すると若干検索精度が低下するが，ANNGT の精度がいずれの検索数でも ANNG を上回っており，検索数に関係なく ANNGT が有効であることを確認した．図示しないが一様分布データの場合と同様にエッジ数が 8 および 16 の場合も同様の傾向を示した．

5.5　本章のまとめ

近似 k 最近傍グラフは高次元データ対してグラフ生成時および近傍検索時の計算量が少ない．近似 k 最近傍グラフでは，任意のノードから問合せオブジェクトに対して近傍ノードを探索する処理，その近傍ノードを起点として探索範囲内のノードを網羅的に探索する処理，の 2 段階の処理からなる．しかし，1 段目の処理では任意のノードから近傍ノードを探索するので，ノード数の増加に比例して計算量が増大するという課題がある．そこで，近傍ノードの探索用のインデックスを用いて，1 段目の計算量を抑制できると考えた．ただし，インデックスには以下の条件を満たす必要がある．

- 距離基準空間インデックスである
- 高次元データを対象とするインデックスである
- データの追加が可能な動的なインデックスである

そこで，これらの条件を満たす dvp-tree を利用して 1 段目の探索を行う手法 (ANNGT) を提案し，一様分布データおよび画像特徴量を用い dvp-tree を利用しない手法 (ANNG) よりもインデックス生成時や検索時の計算量が低減できることを確認した．たとえば，10 万の画像特徴量のオブジェクトを用いてエッジ数が 16 の場合に k 最近傍グラフに対して k 最近傍検索の距離計算回数が ANNG では 58% の削減だが，ANNGT では約 73% の削減が可能であることを確認した．

第 6 章

近似 k 最近傍グラフにおける過剰エッジの削減

6.1 本章の目的

近似 k 最近傍グラフの検索では任意のノードからグラフの探索を開始することにより，問合せオブジェクトから遠いノードが選択された場合に計算量が増加する課題がある．そこで，前章では近似 k 最近傍グラフに加え，dvp-tree を利用することにより問合せオブジェクトの近傍ノードを取得し，その近傍ノードからグラフを探索することで計算量を削減する手法 (ANNGT) を提案した．しかし，近似 k 最近傍グラフでは登録オブジェクトが大量になると一部のノードに極めて大量のエッジが付与される傾向があり，これによりメモリの浪費だけでなく，長大なデータレコードによる実装上の問題が生じる．過剰なエッジを単純に削除するとグラフが非連結グラフとなり，検索精度が低下する．そこで，本章では連結グラフを維持するために，連結性の確認を行いながらエッジの削除や再配置を選択的に実行することで特定ノードの過剰なエッジを削減する方法を提案する．

6.2 課題

検索対象であるオブジェクトはグラフのノードとして近似 k 最近傍グラフに登録されるが，近似 k 最近傍グラフの登録処理に起因して初期に登録したノードほどエッジ数が多い傾向になる．また，オブジェクト空間の歪みや登録するオブジェクトの分布の偏りにより特定のノードにエッジが集中する傾向もある．しかも，過剰なエッジは比較的遠方のノードに接続しているので検索精度に貢献しない．このような無用な過剰なエッジによっ

て生じる課題を以下に示す．

- メモリ領域の浪費
- ノード格納領域の増大による実装機構の複雑化

第 1 に，インデックスは高速化のためにメモリ上に展開されるので，大量のデータでは過剰なエッジによるメモリの浪費は無視できない．実装方法にもよるが，インデックス全体に対してグラフ構造部のメモリ量は 9 割にもなるので過剰なエッジを削減することは使用メモリの削減に有効である．第 2 に，ノードごとに近傍ノードを一括して取得できるように実装することが高速性に有利であるので，ノードごとの全近傍ノードの ID を 1 レコードで保持する．しかし，DBMS によってはレコード長の制限が存在し，過剰なエッジによってもたらされる長大なレコードを保存できない場合がある．筆者らが利用した DBMS でもこの問題が発生した．したがって，レコード長を短くするためにも過剰なエッジを削減することが重要である．なお，利用した DBMS は他社の非商用システムであることから開示できないので，ここでは詳細な説明は省略する．

6.3　提案手法

初期に登録したノードに対してエッジが過剰に付与される傾向を 10 次元の一様分布データを用いて示す．ノードを追加するときに検索するノード数（登録するノードに付与されるエッジ数）$k = 2, 4, 8$ の場合においてノードの登録順に対するノードのエッジ数を図 6.1 に示す．図のエッジ数は 5 万ノードごとの平均値である．明らかに登録初期のノードほど過剰にエッジが集中している．また，当然ながら k の値が大きいほどエッジ数が多くなる．また，初期のノードだけでなくオブジェクトの分布の偏りから特定のノードにエッジが集中する場合がある．このようにエッジの集中するノードが生成されるのでエッジが集中したノードのエッジを定期的に削除する必要がある．しかし，過剰なエッジを単に削除するとグラフが分離してしまい，グラフをたどることができなくなるので，検索精度が低下する．したがって，グラフの連結性を維持しつつエッジを削除する必要がある．

グラフの連結性を保ちながら過剰なエッジを削除することは一見簡単なように思える．たとえば，図 6.2 の N_t が過剰なエッジを有するノードであり 3 本のエッジを削除しなければならないとする．このとき，まず，長いものからエッジ 3 本（図 6.2 (b) の点線のエッジ）を削除し，削除されたエッジが連結していたノード N_3，N_5，N_6 を N_t に連結しているノード N_1，N_2，N_4 のうちの最近傍のノードに，それぞれ連結する．その結果が図 6.2 (b) である．このようにすることで過剰なエッジを削減でき，かつ，連結性が保た

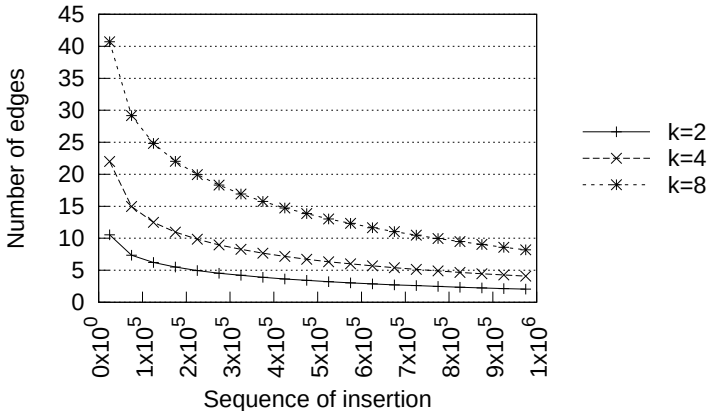


図 6.1 エッジ数 k における登録順とエッジ数の関係.

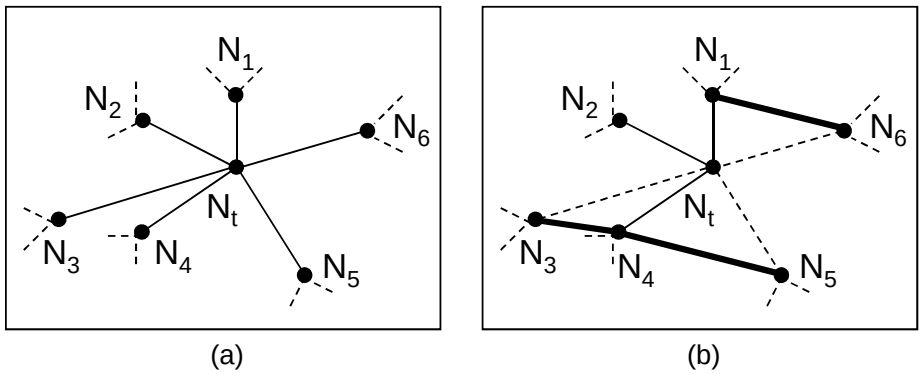


図 6.2 簡便なエッジ削減方法

れる．初期ノードの過剰なエッジについてはこの方法が有効であるが特定のノードにエッジが集中するような場合には問題が生ずる．この状態から，さらにノードを追加すると再度 N_t のエッジが増加するので繰り返し同様の削減処理を行うことになる．すると，今度は N_t の近傍のノードのエッジが増加しそのノードに対してエッジの削減処理を行うことになる．その結果，エッジは再度 N_t に付け直されることになる．このようにしてエッジの削減処理のたびにエッジが近傍ノード間で行ったり来たりするだけでエッジが削減できない状態に陥る．過剰なエッジを近傍ノードへ再配置しても元のエッジが復元される可能性が高いが，より遠方のノードへ再配置することで，元のエッジが復元されることを防ぐことができる．

そこで，図 6.3 に示すようにエッジの削減を行う．図 6.3 (a) の N_t をエッジが過剰なノードとする．図のように N_t は 6 本のエッジを有しており，N_t から 3 本のエッジを削

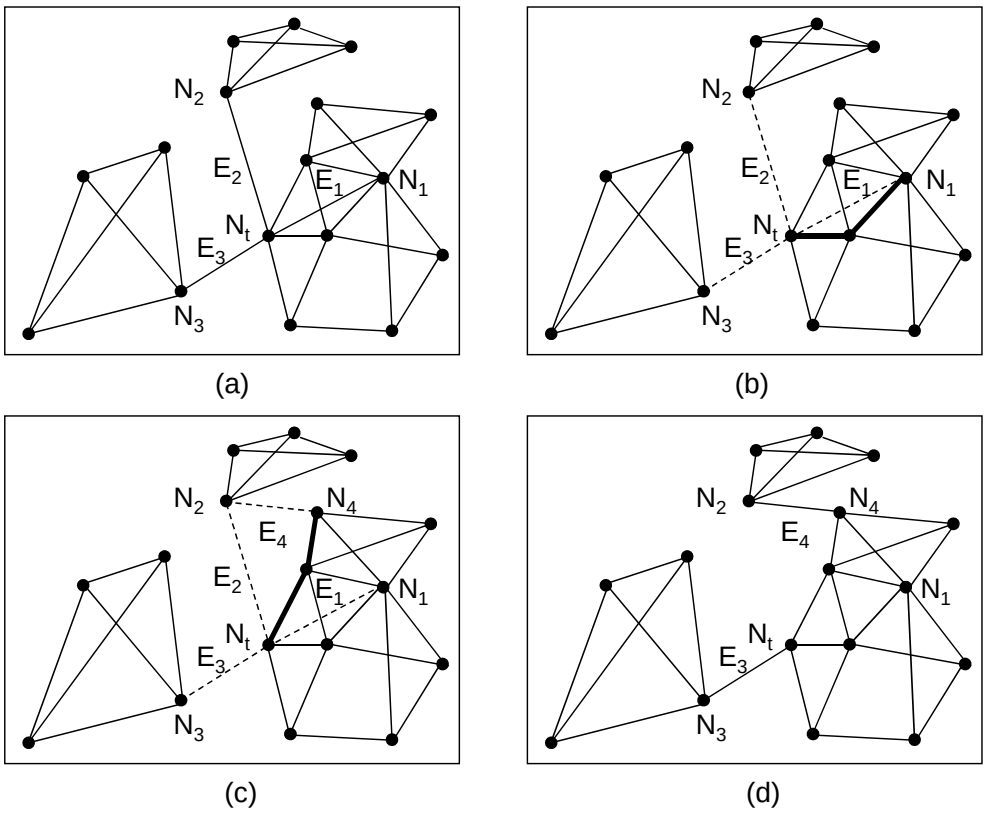


図 6.3　提案するエッジ削減方法

除する必要があるとする．(a) において N_t のエッジの中で，より長いエッジ E_1，E_2，E_3 の 3 本が削除されるべきであると判断される．これらのエッジを単純に削除すると (b) のように 3 つのグラフに分離してしまう．そこで，エッジを削除する方法を 3 通りの場合に分けて説明する．第 1 が (b) の E_1 の場合である．E_1 を削除しても N_t から N_1 への経路 (太線) は残っているので単に E_1 を削除するだけでよい．第 2 が (c) の E_2 の場合である．E_2 を削除すると N_t から N_2 への経路はなくなる．ただし，N_2 から最も近いノードである N_4 への N_t からの経路 (太線) が存在する．したがって，N_t と N_2 の経路を確保するために N_2 と N_4 を接続する E_4 を新たなエッジとして追加する．第 3 が (c) の E_3 の場合である．E_3 を削除すると N_t から N_3 への経路はなくなる．しかも，この場合には N_3 から最も近いノードはやはり N_t であるので，この場合にはエッジ E_3 は削除しない．このようにして，最終的に (a) の N_t のエッジは (d) のように削減される．第 2 の E_2 のようにエッジが再配置される場合には，エッジが過剰であった対象ノードからそのノードが接続されていたノードまでの経路上において対象ノードから最も遠方のノードへエッジを付け直すので，再度エッジの削減を行ったとしてもエッジが元のノードに戻ってしまうよ

Algorithm 8 Optimize(G, k_d, k_s)

for all $p \in G$ **do**
  $N_d \leftarrow \emptyset$
  $T \leftarrow N(G, p)$
  while $|T| > k_d$ **do**
    $s \leftarrow \underset{x \in T}{\operatorname{argmax}}\, d(x, p)$
    if $|N(G, s)| > k_d$ **then**
      $N(G, p) \leftarrow N(G, p) - \{s\}$
      $N(G, s) \leftarrow N(G, s) - \{p\}$
      $N_d \leftarrow N_d \cup \{s\}$
    end if
    $T \leftarrow T - \{s\}$
  end while
  for all $q \in N_d$ **do**
    $r \leftarrow \infty$
    $R \leftarrow \emptyset$
    $C \leftarrow \emptyset$
    KnnSearch(G, q, p, r, k_s, R, C)
    $s \leftarrow \underset{x \in R}{\operatorname{argmin}}\, d(q, x)$
    if $s \neq q$ **then**
      $N(G, q) \leftarrow N(G, q) \cup \{s\}$
      $N(G, s) \leftarrow N(G, s) \cup \{q\}$
    end if
  end for
end for

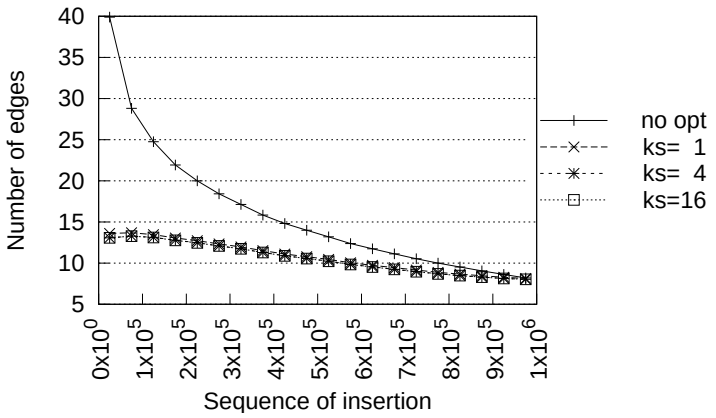


図 6.4　$k = 8$ における登録順に対するエッジ数の関係

表 6.1　エッジの削減率

		ks		
		1	4	16
k	2	16.8%	23.3%	26.7%
	4	25.0%	31.3%	32.1%
	8	32.7%	34.1%	34.2%

うな現象は起きにくくなる．

削減処理中の経路の有無の確認およびエッジの再配置のノードの探索には N_t を探索起点ノード，そして，エッジの接続先ノード (N_1，N_2，N_3) を問合せオブジェクトとしてグラフを検索（経路検索）する．詳細なアルゴリズムを Algorithm 8 Optimize に示す．なお，G はグラフ，k_d は削減対象とするノードの削減後のエッジ数，k_s は経路検索時の検索結果数である．k_d にはインデックス生成時に追加ノードへ付与するエッジ数 k の値を用いる．ANNGT が近似検索でなければ k_s は常に 1 でよいが，近似検索では検索結果に漏れが生じるので k_s を大きくすることで検索精度を高めている．$N(G, p)$ はグラフ G においてノード p に接続されている近傍のノードの集合である．Optimize は Algorithm 6 の KnnSearch を呼び出し k 最近傍検索を行う．なお，q は問合せオブジェクト，s は探索起点ノード，r は検索半径，R は検索結果のノード集合，C は探索済みのノード集合である．

表 6.2 最多エッジの削減率

		ks		
		1	4	16
k	2	54.8% (175)	61.0% (151)	65.6% (133)
	4	65.4% (229)	70.3% (196)	70.7% (194)
	8	73.6% (296)	74.7% (284)	75.0% (281)

注：括弧内はエッジ削減後の最多エッジ数

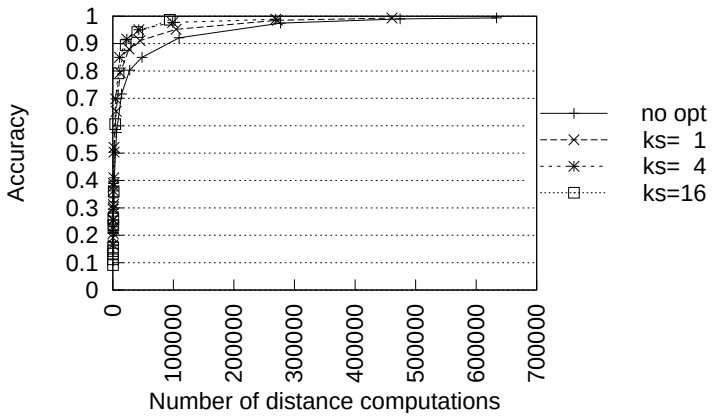


図 6.5 $k = 2$ における距離計算回数と検索精度の関係

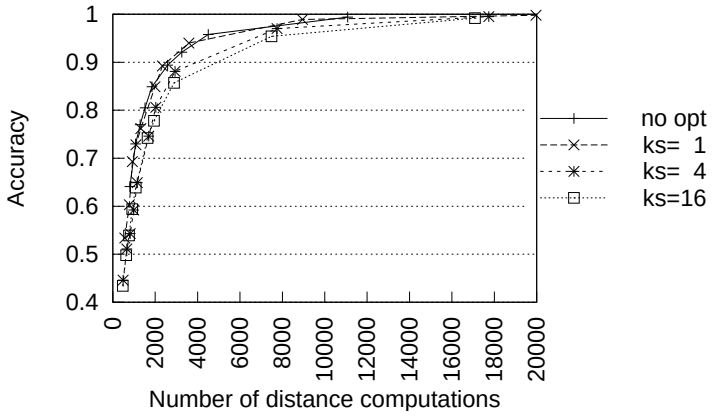


図 6.6 $k = 4$ における距離計算回数と検索精度の関係

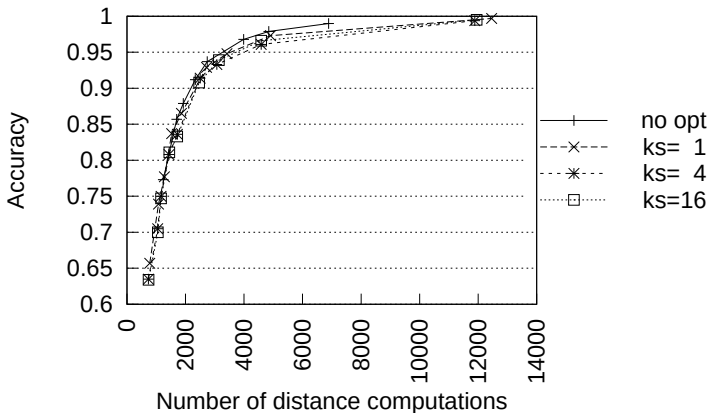


図 6.7　$k = 8$ における距離計算回数と検索精度の関係

6.4　評価実験

オブジェクトとして第 4.4 節の評価と同一の画像特徴量および距離関数を用いて，エッジ削減の効果を示すために 100 万オブジェクトを登録した後にエッジ削減処理を 1 回実行した．図 6.1 と同様に $k = 8$ においてノードの登録順に対するノードのエッジ数の関係を図 6.4 に示す．エッジ削減時のエッジ数 k_d にはこのエッジの付与数 k と同値の 8 を用いている．図中の「no opt」は削減前であり k_s はエッジ削減時の経路検索の検索結果数である．k_s が大きいほどエッジ削減時に経路が発見される確率が高くなる結果，エッジ数が減少する傾向がある．図より全般的にエッジが削減されており，特にエッジが多い登録初期のノードにおいて効果的にエッジが削減されていることを確認できる．エッジの削減率（削減前の全エッジ数に対する削減されたエッジ数の割合）を表 6.1 に示す．表よりエッジ数が最大 34.2% 削減可能であることが確認できる．また，グラフの各ノードごとに接続するエッジ数を調べ，表 6.2 にエッジ数が最多となるノードのエッジ数の削減率 (削減前の最多エッジ数に対する削減後の最多エッジ数の削減エッジ数の割合) を示す．最多となるエッジ数の削減率は最大 75.0% 削減されており，ノードごとのエッジを保持するレコード長の大幅な削減が可能であることを確認した．実装に使用した DBMS ではメモリの拡張方法が特殊であり，実際のメモリ利用量とインデックスのデータ量に大幅な差異があったので，メモリ利用量の測定は行っていない．

次にエッジ削減による検索精度への影響を示す．探索半径係数 ϵ を変化させて検索件数を 20 として k 最近傍検索を行った場合の距離計算回数と検索精度の関係を図 6.5，図 6.6，図 6.7 に示す．検索精度は検索数に対する検索結果中の正解数の割合（再現率）で

あり 50 回の試行の平均値を用いた．各図は登録時のエッジの付与数 $k = 2, 4, 8$ に対応する．$k = 2$ の場合には元々エッジが少ない状況においてエッジ削減により過剰なエッジが再配置される．その結果としてエッジが少なかった部分のエッジが増加する傾向があるので検索精度が上がると考えられる．逆に，$k = 4$ の場合には削減前に高い精度を得るのにほぼ十分なエッジがある状態であったが，エッジ削減によりエッジが減少して精度が下がると考えられる．また，$k = 8$ の場合では，削減前に十分すぎるほど多数のエッジがあったのでエッジを削減しても精度が下がらないと考えられる．$k = 4$ のときには精度が若干下がる可能性があるが，いずれの場合もエッジ削減を行っても概ね精度が維持されることを確認した．なお，$k = 4$ のときのようにエッジ削減により検索精度が若干低下する可能性があるが，十分な総エッジ数となる k を与えることで検索精度の低下を回避できると考えられる．

6.5 本章のまとめ

近似 k 最近傍グラフに加え，dvp-tree を利用する手法では大量に画像を登録するとグラフ中の一部のノードにエッジが過剰に付与される問題がある．過剰なエッジはメモリを浪費するだけでなく，過剰なエッジを保持するレコードが長大になり実装上の問題が生じる．また，エッジを削減するにも検索精度を保つためにグラフの連結性を維持する必要がある．連結性を維持できても局所的にエッジを再配置する場合には削減処理を繰り返すと元のエッジが再現されてしまう．そこで，削減候補のエッジを削除した後にグラフの連結性が維持される場合にはそのエッジを削除し，連結性が損なわれる場合には削除せずに，過剰エッジを有するノードから遠方のノードへそのエッジを再配置する方法を提案した．そして，実際に画像から抽出した特徴量のオブジェクトを用いて提案手法を評価し，検索精度を維持しつつエッジが最大 34.2% 削減されることを確認した．

第 7 章

商品画像検索への近似 k 最近傍グラフの適用事例

7.1 本章の目的

前章までにおいてマルチメディアデータから抽出した高次元のオブジェクトに対して少ない計算量で検索することを可能とする dvp-tree を用いた近似 k 最近傍グラフインデックス (ANNGT) を提案した．さらに，ANNGT はオブジェクトの更新が可能なので応用範囲が広い．本章では実際に ANNGT を商品画像検索に適用する事例を紹介する．また，商品画像検索への適用時に商品分類と画像特徴量の AND 検索が必須となる．そこで，商品分類を分類特徴量として定義した上で，画像特徴量と分類特徴量の空間を融合して検索する方法も同時に提案する．

7.2 近似 k 最近傍グラフ適用時の課題

インターネット上で商品を購買することが一般的になった今日，ショッピングサイトでは多種多様な商品を大量に扱うようになった．ショッピングサイトで提供される商品検索は主に，テキストによる検索，と，商品の各種属性（商品分類，金額，サイズなど）による検索，である．これらの検索方法により，多くの商品は効率良く検索できる．しかし商品の紹介ページのテキストや商品の属性には見た目の情報が十分含まれておらず，ファッションのように見た目が重視される商品では十分に商品を絞り込めない．結果として利用者は多数の商品画像を目で確認して所望の商品を探索することになり，効率良く商品を探索することが困難である．そこで，商品画像から抽出した商品の見た目の特徴を用いて商

品を検索することができれば，利用者は効率良く商品を探索できる．たとえば，Yahoo!ショッピング [52] ではファッション系の商品だけでも 500 万商品以上が存在する．このような大量の商品画像から抽出した特徴量のオブジェクトをインターネット上で対話性を損なわないように高速に検索するには検索用のインデックスが必要である．また，商品は頻繁に追加削除されるので更新可能なインデックスでなければならない．したがって，前章までにおいて提案した dvp-tree を用いた近似 k 最近傍グラフインデックス (ANNGT) の応用として適切だと考えた．

しかし，商品検索では一般に商品が木構造（分類木）として分類されており，画像の特徴量のオブジェクトで検索する場合には同時に分類による絞り込みが必須となる．画像特徴量と分類などの属性との AND 検索は一般に特徴量と分類の検索を独立して行い，両者の結果を統合する．データ数が少ない場合には有効な方法であるが，500 万以上のデータに対して対話性を損なわない時間で全オブジェクトのランキングを生成することは困難である．処理時間を短縮するためにランキングの上位のみを生成すると検索漏れが生じる可能性がある．また，木構造を構成する分類において特定の分類を利用者が指定する場合に，その分類に包含される分類だけでなく兄弟に位置する，つまり，類似する分類も横断して検索したいという要望がある．つまり，分類間の類似性を体現する分類の木構造を考慮した検索が必要である．

7.3　適用手法

分類と画像特徴量の検索を実現するために，分類の木構造に基づく分類の特徴量を定義した上で画像特徴量と統合した単一の特徴量空間を生成することで画像特徴量と分類の AND 検索を実現する方法を提案する．実際のショッピングサイトでは商品分類による検索機能が極めて重要なので，本提案により画像特徴量と任意の分類との AND 検索が可能となり利用者の利便性が格段に向上する．本章では Yahoo!ショッピングの中で見た目が重視されるファッション商品のみを検索の対象とし以下の 2 つの検索を実現する．

- 色指定商品検索：色と分類を個別に指定して色と分類が類似する商品を検索する．
- 類似商品検索：商品を指定してその商品の画像特徴量とその商品の属する分類が類似する商品を検索する．

上記機能を実現するために商品画像を領域分割し，中心に位置する領域を商品領域とし，その商品領域から画像特徴量として第 4.4 節の評価と同一の画像特徴量および距離関数を用いる．

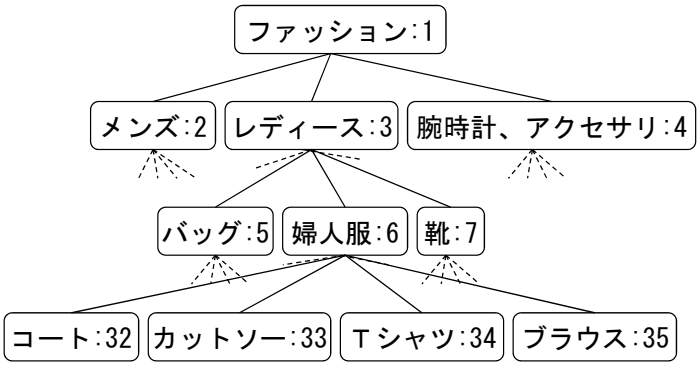


図 7.1 商品の分類木の例

一般に商品は図 7.1 のように分類木により分類されており，商品画像検索では分類と画像特徴量との AND 検索が必須となる．画像特徴量とテキストによる検索を融合することは比較的容易である．テキストデータに対して何らかの距離関数を定義できれば，画像特徴量の距離空間と融合することが可能であり距離空間インデックスによって検索することができる．たとえば形態素解析や N-gram を用いてテキストから単語を抽出した後にベクトル空間モデル [43] により多次元データとして，L_1 距離などを使うことでテキストと画像特徴量を融合した検索が可能となる．この方法を用いた例として筆者らが構築した VisualSeeker [49] がある．

分類においても，個々の分類を単語とみなせばテキスト同様に検索が可能であるが，分類木を考慮しない個々の分類に基づく検索となってしまう．ショッピングサイトの利用者は一般に分類木上でのノード間の関係を考慮しつつ分類間の横断検索を行いたい．たとえば，ブラウスの商品で検索したからといってブラウスだけで検索するのではなく，よく似たものであればワンピースからも検索してほしいという要望がある．かといって，ブラウスで検索しているのに財布といったまったく異なる分類から検索されるのはやはり不適切である．分類木は一般に分類ノード間のエッジ数により分類間の関連性が表現されており，かつ，リーフに近い兄弟ノード間の関連性は高く，ルートに近い兄弟ノード間の関連性は低い．このような場合には前述のテキスト検索の手法を適用できない．

高速に検索するために分類も特徴量として扱い，画像特徴量と分類特徴量を同一の特徴量空間に融合して検索する．分類も特徴量として扱うために分類の経路，つまり，分類木のルートから対象となる分類のノードまでの経路上の分類ノード列を特徴量とする．つまり，分類特徴量はベクトル $\boldsymbol{x}_C = \{x_1, x_2, x_3, ..., x_j\}$ で表わされる．x_i は分類ノード ID，j はルートからリーフまでの最大分類ノード数である．経路上の分類ノード数が最大分類

ノード数に達しない場合には，不足する分類ノードに対して仮想の分類ノード ID として 0 を設定する．たとえば図 7.1 の各分類の数字を分類ノード ID とすると，この分類木では j は 4 となりＴシャツのベクトルは $\{1,3,6,34\}$ となり，靴のベクトルは $\{1,3,7,0\}$ となる．

また，ANNGT で扱うには距離関数を定義する必要があるので，分類特徴量間の距離関数 d_C を以下のように定義する．

$$d_C(\boldsymbol{x}_C,\boldsymbol{x}_C)=\sum_{i=1}^{j}\frac{1}{2^{(i-1)}}f(x_i,y_i) \tag{7.1}$$

$$f(x_i,y_i)=\begin{cases}0 & (x_i=y_i)\\ 1 & (otherwise)\end{cases}$$

なお，$\boldsymbol{x}_C$，$\boldsymbol{y}_C$ を分類特徴量とする．一般的な分類ではルートからリーフへ枝をたどるほど兄弟間の分類の関連性が高くなる．式 7.1 ではルートからリーフへ枝の深さ i が大きくなるほど関数 f の係数が小さくなることでこれを表現している．なお，式 7.1 の関数 f は距離関数であることが自明であり，線形結合の係数が正値であれば距離関数を線形結合した関数もまた距離関数である．したがって，式 7.1 は距離関数の公理を満たす．画像特徴量の距離は以下のように各特徴量の加重平均とする．

$$d_I(\boldsymbol{x}_I,\boldsymbol{y}_I)=\frac{1}{n}\sum_{i=1}^{n}w_i d_i(\boldsymbol{x}_i,\boldsymbol{y}_i)$$

なお，$\boldsymbol{x}_I$,$\boldsymbol{y}_I$ は画像特徴量，n は特徴量種別数，$\boldsymbol{x}_i$,$\boldsymbol{y}_i$ は i 番目の画像特徴量，w_i は i 番目の画像特徴量の重みである．さらに，以下のように画像特徴量の距離と分類特徴量の距離を加重平均したものを AND 検索時の最終的な距離とする．

$$d(\boldsymbol{x},\boldsymbol{y})=(1-w_C)d_I(\boldsymbol{x}_I,\boldsymbol{y}_I)+w_C d_C(\boldsymbol{x}_C,\boldsymbol{y}_C) \tag{7.2}$$

なお，$\boldsymbol{x}$,$\boldsymbol{y}$ は画像特徴量と分類特徴量を結合した特徴量，w_C は分類特徴量の重みである．

7.4　評価実験

分類特徴量による検索の評価に際し Yahoo!ショッピングの「ファッション」の分類に属するすべての商品 (6,822,445 件) を検索対象とした．商品画像から抽出した画像特徴量，および，商品が属する分類から分類特徴量を抽出して ANNGT に登録した．そして，分類木のルートからの分類経路が「ファッション」,「女性もの」,「服」,「Ｔシャツ」とな

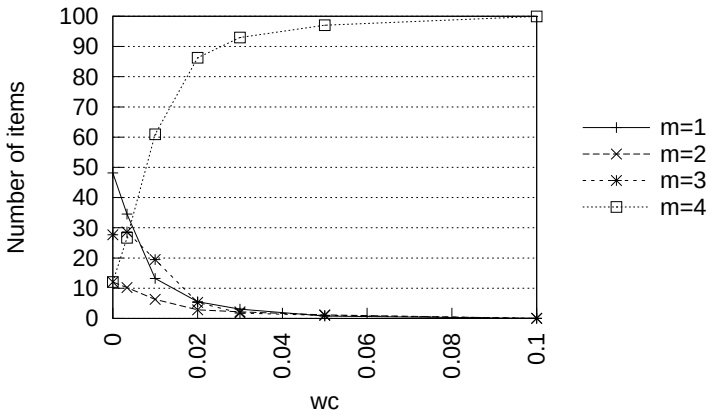


図 7.2 分類の一致数ごとの分類の重み w_C に対する検索された商品数

る分類に属する商品（132,793 件）から任意の 20 商品を選択し，その商品を基に k 最近傍検索を行い各 100 件の検索結果を取得した．式 7.2 の分類特徴量の重み w_C に対して分類の一致数 m ごとに検索された商品数を図 7.2 に示す．$m = 4$ はすべての分類経路が一致している，つまり「T シャツ」の分類に属する商品である．また，$m = 1$ は分類木のルートである「ファッション」のみが一致していることを意味している．「ファッション」以下の商品を対象としたので，すべての商品には必ず「ファッション」の分類が付与されている．したがって m は常に 1 以上である．図で w_C が 0 のときは分類の特徴量を使わずに画像特徴量のみで検索することを意味し，図より「T シャツ」の商品で検索しても，約 10 件つまりは約 10% しか「T シャツ」$(m = 4)$ の商品が検索されていない．しかし，w_C を大きくする，つまり，分類特徴量の重みを重くすることで「T シャツ」の商品数は急激に増加している．逆に，m が 3 以下の場合，つまり，「T シャツ」以外の分類の商品の場合は w_C を大きくすることで検索される商品数が急激に減少している．更に，m が 1 の場合，つまり，「ファッション」しか一致していない場合には，m が 2，3 の場合と比較して急激に商品数が減少しており，分類の木構造上で，より上位の分類経路しか一致しないものほど検索結果から，より多く除去されている．

分類特徴量の重み w_C の変化による k 最近傍検索の結果の変化の例を図 7.3 ，図 7.4 および図 7.5 に示す．いずれにおいても左上の 1 番目の画像が k 最近傍検索時に問合せオブジェクトとして指定した商品であり，この商品は「ファッション」,「女性もの」,「服」,「T シャツ」に属してる．2 から 12 番目の商品画像は類似度順の検索結果である．図 7.3 は分類特徴量の重み w_C を 0 とし分類特徴量を利用せず画像特徴量のみによる検索結果であり，11 件の結果中 2，8 番の 2 商品のみが「T シャツ」である．「T シャツ」の親の分

図 7.3　$w_C = 0$ の場合の類似検索結果

図 7.4　$w_C = 0.01$ の場合の類似検索結果

図 7.5　$w_C = 0.1$ の場合の類似検索結果

類は「服」であるが「服」以外の分類の商品も 1 件含まれている．図 7.4 は分類特徴量の重み w_C を 0.01 として分類特徴量と画像特徴量の両者を使って検索した結果である．11 件中 2，3，5，6，8，10 番目の 6 商品が「T シャツ」に属しており，かつ，すべての商品は親の分類である「服」に属している．分類特徴量の重み w_C を 0.1 とした図 7.5 では，すべての商品が「T シャツ」に属している．これらの結果より，距離関数の重みを変えることにより分類木の検索範囲を変更することが可能であることを確認した．

ANNGT を用いた商品画像検索の UI を図 7.6 に示す．図の左側が検索条件指定部であり，右側が検索結果部である．検索条件指定部は上から色検索用カラーパレット，分類指定，価格指定の領域となる．検索結果部で商品を指定すると類似商品検索が実行される．分類指定の領域で分類を指定することで画像特徴量との AND 検索が可能となる．ANNGT を用いた本商品画像検索は実際に FashionNavi[19] として運用している．

図 7.6　商品画像検索の UI

7.5　本章のまとめ

インターネット上のショッピングサイトでは，大量の商品画像が存在し，かつ，頻繁に商品が更新される．本章では，ANNGT をこのようなショッピングサイトの商品検索に適用する事例を紹介した．なお，商品検索で必須となる商品画像の特徴量と商品分類の AND 検索時において利用者の検索要求を満足する検索結果が得られない問題がある．つまり，分類木における分類間の関連性を考慮した分類横断検索が必要である．そこで分類木の経路を分類特徴量とした上で，分類木の性質に則してルートに近い兄弟間の距離は遠く，リーフに近い兄弟間の距離は近くなるような分類間の距離を定義した．そして，画像特徴量と分類特徴量を融合した単一の空間として ANNGT 上で商品の画像特徴量と分類特徴量の AND 検索を実現する方法を提案した．実際に本システムは FashionNavi として実運用しており，本論文執筆時点ではデータ数は 800 万を超え，日々更新処理を行いながら稼動し続けており ANNGT が実用的なインデックスであると結論付けられる．

第 8 章

結論

昨今のスマートフォンの急増によりマルチメディアデータを手軽に生成できるようになっただけでなく，SNS の発達により容易にそのデータを共有できるようにもなった．このような状況の変化により，今まで以上にマルチメディアデータの検索への要求は強くなっている．多次元データの検索に関する研究は昔から継続して行われてきた．しかしながら，マルチメディアデータの特徴量のように L_p 距離に基づかない多次元データを数百万オブジェクトの単位で検索でき，かつ，実用性を鑑み更新が可能なインデックスは皆無に等しい．

本論文では, 第 3 章にて既存の更新ができない距離基準空間インデックスである vp-tree に改良を加え検索時の計算量が少なく，かつ，更新が可能なインデックス dvp-tree を提案した．dvp-tree は数十次元程度の比較的低次元のオブジェクトに対しては計算量の削減効果があるが，数百次元といった高次元のオブジェクトに対しては検索時の計算量の削減効果がなくなる．そこで，漏れを許容することで検索時の計算量を削減できる近似検索インデックスである k 最近傍グラフをインデックスとして用いる手法に着目した．k 最近傍グラフには，生成時の膨大な計算量，および，非連結グラフに起因する検索精度の抑制といった課題がある．この課題に対して，第 4 章にて，k 最近傍グラフの構造を近似する近似 k 最近傍グラフを提案した．グラフの生成時には各ノードの近傍ノードを検索する必要があるが，その検索に生成途中のグラフを用いることで大幅な計算量の削減が可能となった．実際に，10 万オブジェクトの 50 次元一様分布データにおいて k 最近傍グラフに対して提案手法では約 96.7% の距離計算回数を削減できることを確認した．また，グラフに逐一ノードを接続するので連結グラフが保証され検索精度が向上した．ただし，近似 k 最近傍グラフではグラフをたどり検索条件を満たすノードを探索するが，任意に選択したノードからグラフの探索を開始するのでノードが増えるほど検索時の計算量が増

大する傾向がある．第 5 章にて，近似 k 最近傍グラフに加えて vp-tree を利用する方法 (ANNGT) を提案した．探索を開始するノードとして問合せオブジェクトの近傍ノードを vp-tree を用いて検索することにより計算量が削減された．たとえば，10 万の画像特徴量のオブジェクトを用いてエッジ数が 16 の場合に k 最近傍グラフに対して k 最近傍検索の距離計算回数が ANNGT では約 73% の削減が可能であることを確認した．その一方で，近似 k 最近傍グラフの一部のノードに極めて大量のエッジが付与される傾向があり，これによりメモリの浪費だけでなく，長大なデータレコードによる実装上の問題が生じる．また，エッジを削減するにもグラフの連結性を維持する必要がある．第 6 章にて，グラフの連結性の確認を行いながらエッジの削除や再配置を選択的に実行することで特定ノードの過剰なエッジを削減する方法を提案し，画像特徴量を用いて検索精度を維持しつつエッジが最大 34.2% 削減されることを確認した．最後に第 7 章にて，提案した ANNGT を商品画像検索に適用する事例を紹介した．なお，適用上の課題として商品分類と画像特徴量との AND 検索を実現するために，商品分類を分類特徴量として定義した上で，画像特徴量と分類特徴量の空間を融合して検索する方法も同時に提案した．

本提案によりグラフ構造型インデックス用いた近似検索により高次元データに対する近傍検索の計算量を抑制することが可能である．グラフ構造型インデックスの検索は，グラフの探索を開始するノードから検索範囲までの探索処理，と，検索範囲内の網羅的な探索処理，に分けられる．後者はグラフ構造の大きさに依存しない局所的な処理なので，登録オブジェクト数に影響されない．本提案により木構造インデックスを利用することで登録オブジェクト数が前者へ与える影響を抑えることができる．したがって，本提案は大量のオブジェクトの検索が可能である．本論文での評価実験では最大 10 万件のオブジェクトによる評価であったが，実際には 1 千万強の画像特徴量（1,228 次元）のオブジェクトデータを並列処理せずに約 0.3 秒で検索できることを確認 [49] している．しかも，距離の公理を満たす距離関数さえ定義されていれば任意の距離に適用できるので，広範なデータ種別に適用可能である．たとえば，テキストから抽出される Bag of Words は単なる多次元データとして表現できることから本提案を適用できる．ただし，一般にテキストに対する類似検索の場合，コサイン類似度を利用することが多いが，これは距離の公理を満たさないので，他の距離関数を利用する必要がある．また，Bag of Words の概念を画像に適用した Bag of Features も同様に本提案を適用できる．近年，ビッグデータや Internet of Things といった技術が注目されるようになったことから多様で，かつ，大規模なデータを処理する技術が重要になってきている．多様なデータが混在していても多次元データとして扱うことができると考えられるので，本提案を適用可能である．このように，広範なデータ種別に適用できるが注意も必要である．Bag of Words のようなデータの場合に

は空間上でオブジェクトが疎な状態となりやすい．本提案手法では疎であればあるほどインデックスの効果が減少する．つまり，検索時の計算量が増大したり，検索漏れが増加するといった傾向が顕著になる．何らかの目的のために設計された特徴量を対象オブジェクトとする場合には，その空間が疎であることは特徴量の設計に問題がある可能性があり，特徴量を設計し直せば対処が可能である．しかし，ビッグデータのような既存のデータに本提案を適用し同じような問題が発生したとしても，データを設計し直すわけにはいかない．したがって，既存の多次元データへの本提案の適用には注意が必要である．

近似 k 最近傍グラフは，検索時の計算量削減に効果的なインデックスであるが，グラフ構造に関しては改良の余地があると考えている．各ノードが複数の近傍ノードに接続しており，接続するノード数が多い方が検索時の計算量が削減できる反面，インデックス生成時には計算量が増加する．接続するノード数を減らすとインデックス生成時の計算量は減少するが，問合せオブジェクトの近傍ノードへの経路が減り，より多くの検索時の探索を行わなければならない．つまり，接続するノード数が多い場合と同等の検索精度を得るためには計算量が増える．しかし，エッジを近傍ノードだけではなく，経路をバイパスするように遠方のノードへ故意に接続することで計算量を削減させることができる可能性があると考えている．また，このような経路をバイパスするような長いエッジは前述の疎な空間に対する問題にも効果がある可能性がある．このように，近似 k 最近傍グラフインデックスは，グラフ構造を変更することで特性を大きく変更できるので，様々な性能改善を行う余地が十分にあると考えている．

今後は，さらに多種多様なデータが身の回りで増えることが予想され本研究の成果が高次元データ検索の技術の進歩の一役を担えることを願って，本論文を結ぶ．

参考文献

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, Vol. 45, No. 6, pp. 891–923, 1998.

[2] S. Atnafu, R. Chbeir, and L. Brunie. Efficient content-based and metadata retrieval in image database. *Journal of Universal Computer Science*, Vol. 8, No. 6, pp. 613–622, 2002.

[3] J. R Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C. Shu. Virage image search engine: an open framework for image management. In *Electronic Imaging: Science & Technology*, pp. 76–87. International Society for Optics and Photonics, 1996.

[4] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *Combinatorial Pattern Matching*, pp. 198–212. Springer, 1994.

[5] R. Bayer and E. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, Vol. 1, pp. 173–189.

[6] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Vol. 19, pp. 322–331. ACM New York, NY, USA, 1990.

[7] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, Vol. 18, pp. 509–517, 1975.

[8] S. Berchtold, D. A. Keim, and H. P. Kriegel. The X-tree: An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, pp. 28–39, 1996.

[9] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional

metric spaces. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pp. 357–368. ACM New York, NY, USA, 1997.

[10] S. Brin. Near neighbor search in large metric spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pp. 574–584. IEEE, 1995.

[11] W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, Vol. 16, No. 4, pp. 230–236, 1973.

[12] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, Vol. 42, No. 1, pp. 67–90, 1995.

[13] M. Chalmers and P. Chitson. Bead: Explorations in information visualization. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 330–337. ACM, 1992.

[14] J. Y. Chen, C. A. Bouman, and J. Dalton. Similarity pyramids for browsing and organization of large image databases. In *SPIE Human Vision and Electronic Imaging III*, Vol. 3299, 1998.

[15] J. Chen, H. Fang, and Y. Saad. Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection. *Journal of Machine Learning Research*, Vol. 10, pp. 1989–2012, 2009.

[16] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pp. 426–435, 1997.

[17] A. Esuli. Mipai: using the pp-index to build an efficient and scalable similarity search system. In *Proceedings of the 2nd International Workshop on Similarity Search and Applications*, pp. 146–148. IEEE, 2009.

[18] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, Vol. 3, No. 3, pp. 231–262, 1994.

[19] FashionNavi. http://visseeker.yahoo-labs.jp/fn/.

[20] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, et al. Query by image and video content: The qbic system. *Computer*, Vol. 28, No. 9, pp. 23–32, 1995.

[21] flickr. http://www.flickr.com/.

[22] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via

hashing. In *Proceedings 25th International Conference on Very Large Data Bases*, pp. 518–528, 1999.

[23] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Vol. 14, pp. 47–57. ACM New York, NY, USA, 1984.

[24] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 1312–1317, 2011.

[25] K. Hatano, Q. Qian, and K. Tanaka. A som-based information organizer for text and video data. In *Proceedings of the 5th International Conference on Database Systems for Advanced Applications*, Vol. 6, p. 204, 1997.

[26] ImageCLEF. http://www.imageclef.org/.

[27] M. Ioka. A method of defining the similarity of images on the basis of color information. *IBM Tokyo Research Lab, Technical Report. RT-0030*, 1989.

[28] I. Kalantari and G. McDonald. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, No. 5, pp. 631–634, 1983.

[29] N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Vol. 26, pp. 369–380. ACM, 1997.

[30] R. Kurniawati, J. S. Jin, and J. A. Shepherd. The SS+-tree: An improved index structure for similarity searches in a high-dimensional feature space. In *Proceedings of SPIE/IS&T Conference on Storage and Retrieval for Image and Video Databases V*, Vol. 3022, pp. 110–120, 1997.

[31] P. H. Lewis, K. Martinez, F. S. Abas, M. F. A. Fauzi, S. C. Y. Chan, M. J. Addis, M. J. Boniface, P. Grimwood, A. Stevenson, C. Lahanier, et al. An integrated content and metadata based retrieval system for art. *IEEE Transactions on Image Processing*, Vol. 13, No. 3, pp. 302–313, 2004.

[32] P. Lipson, E. Grimson, and P. Sinha. Configuration based scene classification and image indexing. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1007–1013. IEEE, 1997.

[33] M. L. Micó, J. Oncina, and E. Vidal. A new version of the nearest-neighbour ap-

proximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, Vol. 15, No. 1, pp. 9–17, 1994.

[34] G. Navarro. Searching in metric spaces by spatial approximation. *The VLDB Journal*, Vol. 11, No. 1, pp. 28–46, 2002.

[35] H. Noltemeier, K. Verbarg, and C. Zirkelbach. A data structure for representing and efficient querying large scenes of geometric objects: Mb* trees. In *Geometric Modelling*, pp. 211–226. Springer, 1993.

[36] A. Okabe, B. Boots, and K. Sugihara. *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, Inc. New York, NY, USA, 1992.

[37] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, Vol. 18, No. 3, pp. 233–254, 1996.

[38] E. Plaku and L. E. Kavraki. Distributed computation of the knn graph for large high-dimensional point sets. *Journal of Parallel and Distributed Computing*, Vol. 67, No. 3, pp. 346–359, 2007.

[39] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp. 71–79. ACM New York, NY, USA, 1995.

[40] Y. Rubner, L. Guibas, and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, pp. 661–668, 1997.

[41] E. V. Ruiz. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, Vol. 4, No. 3, pp. 145–157, 1986.

[42] J. Sakagaito and T. Wada. Nearest first traversing graph for simultaneous object tracking and recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pp. 1–7, 2007.

[43] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, Vol. 18, No. 11, pp. 613–620, 1975.

[44] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys CSUR*, Vol. 16, No. 2, pp. 187–260, 1984.

[45] T. B. Sebastian and B. B. Kimia. Metric-based shape retrieval in large databases. In *Proceedings of 16th International Conference on Pattern Recognition*, Vol. 3, pp. 291–296, 2002.

[46] T. Seidl and H. P. Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proceedings of the International Conference on Very Large Data Bases*, pp. 506–515, 1997.

[47] J. Smith and S. Chang. Tools and techniques for color image retrieval. In *Electronic Imaging: Science & Technology*, pp. 426–437. International Society for Optics and Photonics, 1996.

[48] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, Vol. 40, No. 4, pp. 175–179, 1991.

[49] VisualSeeker. http://visseeker2.yahoo-labs.jp/vs/.

[50] R. Weber, H. J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pp. 194–205, 1998.

[51] D. A. White and R. Jain. Similarity indexing with the SS-tree. In *Proceedings of 12th International Conference on Data Engineering*, pp. 516–523, 1996.

[52] Yahoo!ショッピング. http://shopping.yahoo.co.jp/.

[53] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 311–321. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 1993.

[54] P. N. Yianilos. Excluded middle vantage point forests for nearest neighbor search. In *Proceedings of DIMACS Implementation Challenge*, 1999.

[55] H. Zhang and D. Zhong. Scheme for visual feature-based image indexing. In *SPIE Storage and Retrieval for Image and Video Databases III*, pp. 36–46, 1995.

[56] 舘村純一. インタラクティブ視覚化による文献集合からの情報獲得支援. 日本ソフトウェア科学会第 13 回大会, 1996.

[57] 串間和彦, 佐藤路恵, 赤間浩樹, 山室雅司. 大量画像の閲覧を目的とする階層的分類支援機能: 画像目録の実装と評価. 情報処理学会論文誌. データベース, Vol. 41, pp. 54–63, 2000.

[58] 呉君錫, 金子邦彦, 牧之内顕文, 上野敦子. 自己組織化特徴マップに基づいた類似画像検索システムの設計・実装と性能評価. 電子情報通信学会技術研究報告, Vol. 100,

No. 31, pp. 9–16, 2000.

[59] 保木大典, 片山幸治, 仲川亜希, 小西修. 協調マルチメディア情報収集法. 情報処理学会研究報告データベースシステム, Vol. 133-56, pp. 335–340.

[60] 齋藤英文, 村上伸一. 絵画の自動分類法に関する一検討. 電子情報通信学会技術研究報告. IE, 画像工学, Vol. 97, No. 409, pp. 39–44, 1997.

公開文献

[61] M. Iwasaki, N. Takahashi, and K. Morozumi. Access method for image database. In *SPIE Storage and Rtrieval for Media Databases*, pp. 21–29. International Society for Optics and Photonics, 2000.

[62] 岩崎雅二郎. 大量画像データベースへの効率的アクセスを可能とする統合画像アクセスインタフェース. 情報処理学会論文誌. データベース, Vol. 42, No. SIG1(TOD8), pp. 32–42, 2001.

[63] 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. データベース, Vol. 40, No. SIG3(TOD1), pp. 24–33, 1999.

[64] 岩崎雅二郎. 近似 k 最近傍グラフによる距離空間の近傍検索. 情報処理学会論文誌: データベース, Vol. 3, No. 1(TOD45), pp. 18–28, 2010.

[65] 岩崎雅二郎. 木構造型インデックスを用いた近似 k 最近傍グラフによる近傍検索. 情報処理学会論文誌, Vol. 52, No. 2, pp. 817–828, 2011.

[66] 岩崎雅二郎. 商品画像検索へのグラフ構造型インデックスの適用. 画像電子学会誌, Vol. 42, No. 5, pp. 633–641, 2013.

[67] N. Takahashi, M. Iwasaki, T. Kunieda, Y. Wakita, and N. Day. Image retrieval using spatial intensity features. *Signal Processing: Image Communication*, Vol. 16, No. 1, pp. 45–57, 2000.

謝辞

本論文における研究は著者が（株）リコー ソフトウェア研究所，および，ヤフー（株）Yahoo! JAPAN 研究所にて行った研究成果をまとめたものです．本研究の実施，また，本論文の執筆にあたり，多くの方々からご指導，ご支援を賜りまして，この場を借りてお世話になった方々へ御礼申し上げます．

まず，課程外にもかかわらず本論文の主査を快く引き受けていただいた上，論文執筆にあたり丁寧にご指導いただいた慶應義塾大学大学院理工学研究科 遠山元道 准教授には深く感謝いたします．

本論文の副査を引き受けていただき，ご多忙の折，丁寧なご助言を賜りました慶應義塾大学大学院理工学研究科 藤代一成 教授，斎藤英雄 教授，松谷宏紀 専任講師，NTT コミュニケーション科学基礎研究所 小杉尚子 様に感謝いたします．

また，前勤務先である（株）リコー ソフトウェア研究所および現勤務先であるヤフー（株）Yahoo! JAPAN 研究所の皆様には研究をご支援して頂き厚く御礼申し上げます．商品画像検索のデータ提供及びシステムに関するご助言を頂きました Yahoo!ショッピングの関係者の皆様，本論文の研究成果を用いた各サービスを日頃運用管理していただいているヤフー（株）マルチメディア処理の皆様には御礼申し上げます．

最後に，博士論文執筆にあたり日頃温かく見守ってくれた妻と息子に感謝致します．