社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

概算距離の精度向上による近似最近傍探索の高速化

佐藤 智一† 岩村 雅一† 黄瀬 浩一†

† 大阪府立大学大学院工学研究科 〒 599–8531 堺市中区学園町 1–1

E-mail: sato@m.cs.osakafu-u.ac.jp, {masa,kise}@cs.osakafu-u.ac.jp

あらまし 登録されたデータからクエリに最も近いものを探し出す最近傍探索問題では，探索誤りを許容することで計算時間を大幅に削減することができ，これを近似最近傍探索問題と呼ぶ．近似最近傍探索は一般に，最近傍点となる確率の高い点を選択し，それらとクエリとの距離を計算するという 2 段階の処理で実現され，前者が手法の良し悪しを決定する．本稿では，この処理で用いる「概算距離」を計算量を増やすことなく，より高精度に推定することにより，高精度かつ高速な近似最近傍探索，を実現する手法を提案する．実験の結果，50% の精度で比較すると従来手法 [1] と比べて，64 次元のデータで約 4 倍，256 次元のデータで約 2.5 倍の処理速度を得ることが確認できた．

キーワード 近似最近傍探索，多次元ハッシュ，点対バケットの概算距離，ハッシュテーブルの分割

1. まえがき

本稿では**最近傍探索問題**を扱う．これは，データ空間内でクエリ (探索質問点) に最も距離の近いデータ (**最近傍点**) を探索するものであり，この問題を高速に解くための様々な手法が提案されている [2]．これらの処理には一般に予めデータの分布解析を行い，インデクシングを施す必要がある．探索時には，求めたインデックスから最近傍点になり得ないものを除外し，計算コストの削減を図る．しかし，データが高次元である場合，次元の呪いによって高速化の効果を得ることができず，大規模なデータベースを扱う場合には実用が難しくなる．

そこで，距離計算対象をクエリの最近傍点である可能性の高い点（**最近傍候補**）に大きく絞込み，探索誤りを許容することで更なる処理の高速化を図る．これを**近似最近傍探索**といい，探索を高精度かつ高速に行うには，クエリに近い点のみを最近傍候補として効率的に選別することが求められる．

近似最近傍探索においてインデクシングを行うためのデータ構造は大きく分けて２つあり，１つは木構造を用いるもの，もう１つはハッシュ法を用いるものである．木構造を用いる手法には ANN [3]，ball-tree [4]，PCA-tree [5]，vp-tree [6] などがあり，ハッシュ法を用いる手法には Locality Sensitive Hashing(LSH) [7], [8]，Multi-Probe LSH [9]，Spectral Hashing(SH) [10]，バケット距離に基づく近似最近傍探索 [1] などがある．

本稿ではハッシュ法を用いた近似最近傍探索に着目し，概算距離を用いることで高速に解を得ることができる手法を提案する．ハッシュ法を用いる場合には一般に複数のハッシュ関数を用いてデータにインデクシングを施し，クエリのインデックスに一致する点，またはそれに近いインデックスを持つ点を最近傍候補とする．従って，ハッシュ関数によって得られるインデックス間の距離が実際の空間における距離の大小関係を保持すると同時に，インデックス間の距離計算が高速に行えることが重要となる．以後，ハッシュによるインデックス間の距離を概算距離と呼ぶ．

従来手法における概算距離は実際の真の距離の大小関係を十分に保持できているとは言えず，探索の精度を上げるためには多くの最近傍候補を確保する必要があり，高速に解を得ることができない．そこで我々は，概算距離が真の距離の大小関係をよく反映している従来手法 [1] に着目し，２点の改良を行った．１つ目は概算距離の精度向上であり，距離の概算法を領域対領域から点対領域に変更することで，データ構造を変えること無く精度を向上させる．２つ目は距離の概算に必要なハッシュサイズの低減であり，ハッシュテーブルを分割してインデクシングを施すことにより，ハッシュサイズを探索に適した大きさにする．これら２点の改良により，高精度かつ高速に概算距離を求めることができ，実験の結果，50% の精度で比較すると従来手法 [1] と比べて，64 次元のデータで約 4 倍，256 次元のデータで約 2.5 倍の処理速度を得ることが確認できた．

2. 関連手法

本節では，代表的な近似最近傍探索手法の概要について説明する．木構造を用いる手法として ANN，ハッシュ法を用いる手法として LSH，SH，従来手法 [1] を取り上げる．

2.1 ANN

木構造を用いる手法の中で最も代表的なものの一つが ANN [3] である．ANN は２分木をベースとした手法であり，木の構築ではデータ空間を階層的に２等分していき，葉に入る点が１つになるまで分割を繰り返す．探索時には近似度 ϵ を与え，$\epsilon = 0$ であれば厳密な最近傍探索となる．クエリが与えられると，木を辿り，到達した葉に登録されているデータと距離計算を行う．その距離が r であるとすると，図１のように分割された各領域の最も近いところがクエリから半径 $\frac{r}{1+\epsilon}$ に入る領域を探索領域とする．図の★はクエリ，●はデータ点，着色部分が探索領域を表す．$\epsilon = 0$ であれば，r より近い点が存在する可能性のある

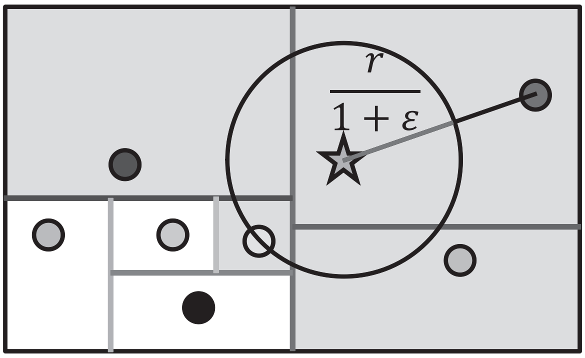
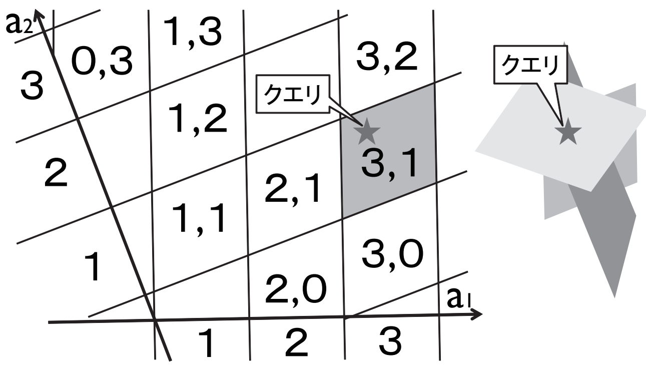


図 1　ANN



（a）インデクシングの様子　（b）探索領域

図 2　Locality Sensitive Hashing

領域を全て探索するので，必ず真の最近傍点を得ることができる．また，ϵ を大きくすると探索半径は小さくなり精度は下がるが，処理速度は大きくなる．

2. 2　Locality Sensitive Hashing

Locality Sensitive Hashing(LSH) [7], [8] はハッシュ法を利用した近似最近傍探索手法の中で最も代表的な手法の一つである．ここでは LSH の中でも本研究に関連する，ベクトル空間での LSH [8] について述べる．

図 2(a) に示すように，LSH はデータ空間をランダムに生成された基底方向に等間隔に分割することで，空間をバケットと呼ばれる領域に分割してインデクシングを施す．図 2(a) の軸 a_1, a_2 はランダムに生成された基底であり，セル状の１つ１つの領域がバケットである．そして，探索時にはクエリと同じバケットに属する点を最近傍候補とする．しかし，これだけでは真の最近傍点を候補から漏らす可能性が高いため，この処理を数回繰り返すことで候補を増やして精度を上げる工夫をしている．図 2(b) は３回の射影によって得られた探索領域の様子を表す．文献 [8] の LSH では次式のようなハッシュ関数を用いる．

$$H_j(\boldsymbol{x}) = \{h_{j1}(\boldsymbol{x}), h_{j2}(\boldsymbol{x}), \ldots, h_{jk}(\boldsymbol{x})\} \tag{1}$$

$$h_{ji}(\boldsymbol{x}) = \left\lfloor \frac{\boldsymbol{a}_{ji} \cdot \boldsymbol{x} + b_{ji}}{w} \right\rfloor \tag{2}$$

ただし，$j = 1 \ldots L$，$\boldsymbol{x}$ は任意の点，$\boldsymbol{a}_{ji}$ は各次元の要素の値がガウス分布から独立に選ばれたベクトル，w はハッシュ幅であり，b_i は区間 $[0, w]$ から一様に選ばれた実数である．そして，最近傍候補はクエリ $\boldsymbol{q}$ に対して $\exists(H_j(\boldsymbol{q}) = H_j(\boldsymbol{p}))$, $j = 1, \ldots L$ となる点である．

LSH はデータの分布に依らないランダムな基底に射影を行うため，最近傍候補に入るかどうかが真の距離の大小関係をあまり反映できないことが多く，効率的な最近傍候補の絞り込みが難しい．

2. 3　Spectral Hashing

Spectral Hashing(SH) [10] はデータ空間の主成分を分散の大きい方からいくつか選択し，これらを元にデータをバイナリ符号化してインデックスとする．そして，クエリに与えられた符号とのハミング距離が閾値以下のものを最近傍候補とする．SH のハッシュ関数は次のようなものを用いる．

$$H(\boldsymbol{x}) = \{h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_l(\boldsymbol{x})\} \tag{3}$$

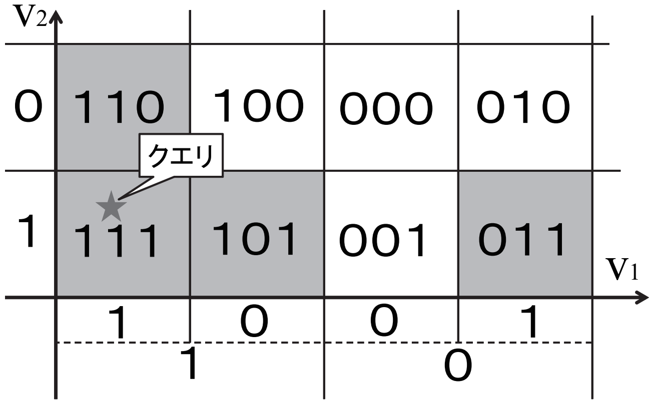


図 3　Spectral Hashing

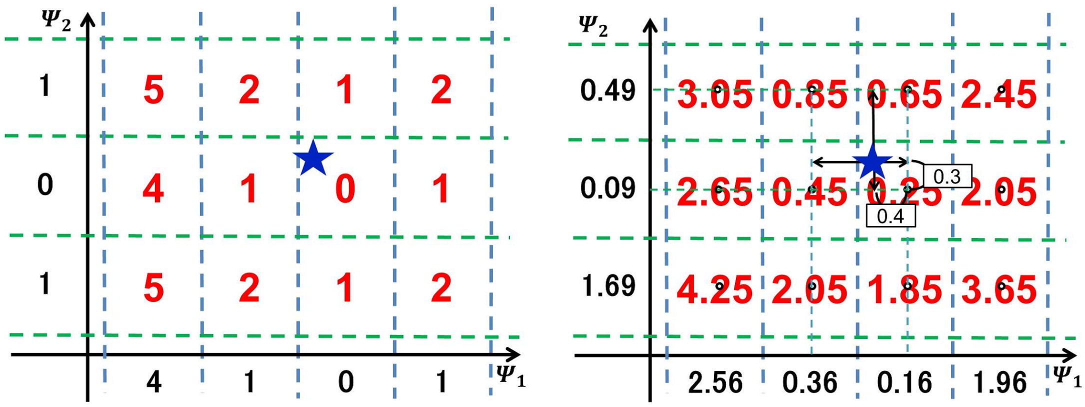
$$h_i(\boldsymbol{x}) = \begin{cases} 0 & (\Phi_i(\boldsymbol{x}) < 0) \\ 1 & (\Phi_i(\boldsymbol{x}) \geqq 0) \end{cases} \tag{4}$$

$$\Phi_i(\boldsymbol{x}) = \sin\left(\frac{\pi}{2} + \frac{k\pi}{\max_i - \min_i}\boldsymbol{x} \cdot \boldsymbol{v}_i\right) \tag{5}$$

ただし，k は空間分割の周波数，$\boldsymbol{v}_i$ は主成分ベクトル，$\max_i$，$\min_i$ は主成分方向の最大値と最小値を表す．図 3 はインデクシング（符号化）の様子を示したものであり，着色領域はハミング距離の上限を１とした場合の探索領域を表したものである．SH は主成分に射影を行うため，射影後も元の距離が保持されやすいと言えるが，概算距離がバイナリ符号のハミング距離で表されるため，距離尺度の違いから真の距離との誤差が生じ，図 3 に示すように，クエリから遠い 011 の領域が最近傍候補となるといった問題がある．

2. 4　バケット距離に基づく近似最近榜探索

従来手法 [1] はデータ空間を任意の正規直交基底に対して共通の分割幅で等分し，これを多次元ハッシュによって表現する（図 4(a)）．この処理は，データをスカラー量子化することに等しく，真の距離をよく反映したデータ構造となっている．探索時には，クエリと各点が属するバケットのインデックスから概算距離を求めることで，クエリを中心とする近似的な超球領域から高速に最近傍候補を抽出することができる．$\boldsymbol{x}$ を任意の点，Ψ_i を正規直交基底，w を分割幅とすると，v 次元ハッシュ関数 H は次のようになる．



(a) 従来手法 [1]　　　(b) 提案手法

図 4　従来手法 [1] と提案手法の概算距離の比較

$$H(\boldsymbol{x}) = \{h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_v(\boldsymbol{x})\} \tag{6}$$

$$h_i(\boldsymbol{x}) = \left\lfloor \frac{\Psi_i \cdot \boldsymbol{x}}{w} \right\rfloor \tag{7}$$

また，概算距離は各バケットのインデックスの距離（**バケット距離**）を用いる．ここで，$B(\boldsymbol{p})$ を任意の点 $\boldsymbol{p}$ が属するバケットの重心であるとすると，２点 p_1, p_2 の概算距離 (L_p ノルム) は次のように表される．

$$D(B(\boldsymbol{p}_1), B(\boldsymbol{p}_2)) = \left| \sum_{i=1}^{v} \{h_i(\boldsymbol{p}_1) - h_i(\boldsymbol{p}_2)\} \right|^p \tag{8}$$

図 4(a) の数字はクエリが属するバケットからの各バケット距離を表す．探索時には探索半径 R を与え，$D(B(q), B(p)) \leqq R$ を満たす点 p を最近傍候補とする．

この手法は高次元データに対して，高速な探索を行うことができない．これは，ハッシュの次元数とハッシュサイズの関係から生じる問題である．データの次元数が大きくなると，概算距離の精度を維持するためにはハッシュの次元数 v もそれに合わせて大きくする必要がある．また，ハッシュの次元数 v を大きくするとハッシュサイズが膨大なものとなる．例えば，１つの基底の分割数を s とするとハッシュサイズのオーダーは $O(s^v)$ となる．例え s を最小の 2 に抑えても 30 次元のハッシュを構成するには約 10 億のハッシュサイズが必要となる．ハッシュサイズはデータによって適切な大きさがあり，一般にハッシュサイズがデータ数を大きく上回ると，一つ一つのバケットが疎になり，最近傍探索の精度を維持するために多くのバケットを参照する必要が生じることから，高速に最近傍候補を抽出することができなくなる．図 5(a)，5(b) に v を変化させた時の精度と処理時間の関係を示す．$v = 24$ の時，ハッシュサイズは約 16000 万となっており，v を大きくしていくと徐々に処理時間が短くなっていっているが，$v = 24$ を超えると処理時間が大幅に増加している．

従って，メモリだけでなく処理速度の観点からもハッシュサイズには上限が存在し，高次元データに対してもハッシュの次元数を大きくすることができず，高速な探索を行うことができない．

3. 提案手法

近似最近傍探索においては，最近傍候補がクエリを中心とする超球領域から選ばれることが理想であり，これを実現するには概算距離が元の空間での距離をうまく保持していることが重要となる．しかし，従来手法の多くは概算距離が真の距離の大小関係を十分に保持することができず，探索の精度を上げるためには多くの最近傍候補を確保する必要があり，高速に解を得ることができない．

そこで，概算距離が真の距離の大小関係をよく反映している従来手法 [1] に着目し，２点の改良を施した高精度かつ高速な概算距離により，処理全体の高速化を図る．

3.1 点対バケットの概算距離

１つ目の改良は概算距離の精度向上である．従来手法 [1] ではクエリとデータのそれぞれが属するバケット間の距離，すなわちバケット対バケットの距離で概算距離を算出した．本稿では厳密なクエリの位置から各データが属するバケットへの距離，すなわち点対バケットの距離で概算距離を求めることにより，同じデータ構造まま概算距離の精度を向上させる手法を提案する．

ハッシュ関数として従来手法 [1] と同じ式 (7) を用いる．点 $\boldsymbol{p}_1$ と，点 $\boldsymbol{p}_2$ が属するバケット $B(\boldsymbol{p}_2)$ の間の距離（L_p ノルム）は次のように表される．

$$D(\boldsymbol{p}_1, B(\boldsymbol{p}_2)) = \sum_{i=1}^{v} \left| \frac{\boldsymbol{p}_1 \cdot \Psi_i}{w} - \left(h_i(\boldsymbol{p}_2) + \frac{1}{2} \right) \right|^2 \tag{9}$$

$h_i(\boldsymbol{p}_2) + \frac{1}{2}$ はバケット $B(\boldsymbol{p}_2)$ の Psi_i 方向の座標を表しており，式 (9) はクエリとバケット重心の距離に等しい．クエリの位置が特定されている分，(8) 式よりも精度の高い概算距離となっている．図 4(b) に従来手法 [1] の図 4(a) と同じ位置にクエリ入った時の概算距離を例示する．図の 0.3,0.4 は Psi_1, Psi_2 方向の，クエリとクエリが属するバケットの重心までの距離である

3.2 ハッシュテーブル分割による距離の概算

[1] の手法ではハッシュサイズの制約から，高次元データに対してもハッシュの次元数を大きくすることができなかった．そ

こで，高次元のハッシュテーブルを分割し，低次元のハッシュテーブルから得られる概算距離を統合することによって高次元ハッシュの概算距離を求める手法を提案する．v 次元ハッシュを M 個のハッシュテーブルによって行う場合，v 個のハッシュ関数を M 個の組に分けて，ハッシュテーブルを構成する．つまり，ハッシュ関数は次のようになる．

$$H_j(\boldsymbol{x}) = \{h_{j1}(\boldsymbol{x}), h_{j2}(\boldsymbol{x}), \ldots, h_{jt_j}(\boldsymbol{x})\} \tag{10}$$

$$h_{ji}(\boldsymbol{x}) = \left\lfloor \frac{\Psi_{ji} \cdot \boldsymbol{x}}{w} \right\rfloor \tag{11}$$

ただし，$j = 1, \ldots, M$，$\sum t_j = v$ である．そして，クエリ q から任意の点 $\boldsymbol{p}$ への概算距離を

$$D(q, B(p)) = \sum_{j=1}^{M} D_j(q, B_j(p)) \tag{12}$$

で表すことができ，これは v 次元ハッシュによって求められる概算距離に等しい．ハッシュテーブルを分割する利点は，同じ次元数のハッシュを表現する場合でも１つのハッシュテーブルを用いる場合に比べて飛躍的にハッシュサイズが小さくなることにある．一つの基底方向にそれぞれ s 分割されている場合を考えると，１つのハッシュテーブルによって v 次元ハッシュを表現する場合，ハッシュサイズは $O(s^v)$ であるのに対し，M 個のハッシュテーブルに分割して v 次元ハッシュを表現する場合，１つのテーブルのハッシュサイズは $O(s^{v/M})$ となり，M に対して指数関数的に減少することが分かる．従って，ハッシュテーブルを分割して多次元ハッシュを表現することにより，高次元データに対しても最近傍候補の抽出速度を落とすことなく，概算距離の精度を向上させることができる．

4. 予備実験

従来手法 [1] の多次元ハッシュの次元数 v を変化させた時の，精度と処理時間の関係を示す．ここで用いたデータは 64 次元または 128 次元，1000 万点の正規分布に基づく人工データ，クエリは同じ条件で生成した 2000 点である．用いた計算機は，CPU が Opteron(tm)6174(2.2GHz)，メモリは 256[GB] であり，実験はシングル・コアで行った．結果を図 5(a)，5(b) に示す．人工データにおいては従来手法 [1] と提案手法共に $v = 24$ が最も精度と処理時間の関係が良くなったので，以降，人工データを用いた実験においては $v = 24$ とする．

5. 実　　験

本節では提案手法の性能を評価するため，前節で紹介した従来手法と提案手法の比較実験を行う．計算機は予備実験と同じものを用いた．従来手法 [1] 及び提案手法で用いる基底は，人工データでは元の基底を分散の大きいものから v 個選び，実データでは主成分分析で得られた主成分を分散の大きい方から v 個を選んだ．

5.1 実　験　1

Spectral Hashing，従来手法 [1]，提案手法 (ハッシュテーブル数：１) において，ハッシュサイズを変化させたときの概算距離と実際のユークリッド距離の相関係数を示す．データには 32 次元の正規分布に基づく人工データを１万点，クエリには同じ条件で生成した 100 点を用いた．各クエリから１万点のデータへのユークリッド距離と概算距離の相関係数を示す．Spectral Hashing の符号長は 4〜32bit の間で 4bit ずつ増加させた．この時のハッシュサイズは 2^4〜2^{32} となる．従来手法 [1]，提案手法 (ハッシュテーブル数：１) は同じデータ構造であり，共に分割幅 $w = \{\max(\Psi_v \cdot p) - \min(\Psi_v \cdot p)\}/2$（$v$ 番目の基底が２分割される幅）とし，ハッシュの次元数は v を 4〜28 の間で 4 ずつ増やしたものと 30（ハッシュサイズが 2^{32} を超えない最大の基底数）を用いた．結果を図 6(a) に示す．横軸がハッシュサイズ，縦軸が相関係数である．

この結果から，ハッシュテーブルの数が１つであっても，SH や従来手法 [1] に比べて，提案手法の概算距離が実際のユークリッド距離をよく表してることが分かる．従って，提案手法で導入した点対バケットの概算距離は最近傍候補の抽出に有効であることが分かった．

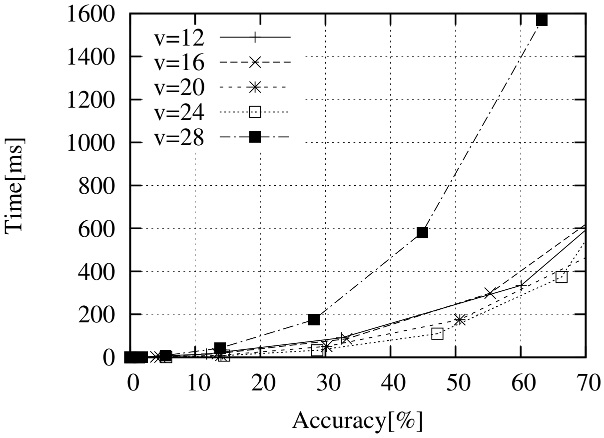
参考に，概算距離と実際のユークリッド距離の関係を図示する．データは上と同じものを用い，クエリとしてデータ全体の平均ベクトルを与えた．Spectral Hashing の符号長は 32bit(ハッシュサイズは 2^{32} = 約 42 億)，従来手法 [1] と提案手法は基底数 $v = 30$ とした．この時，提案手法と従来手法 [1] のハッシュサイズは約 24 億であった．結果を図 6(b)〜図 6(d) に示す．横軸が概算距離，縦軸が実際のユークリッド距離である．

5.2 実　験　2

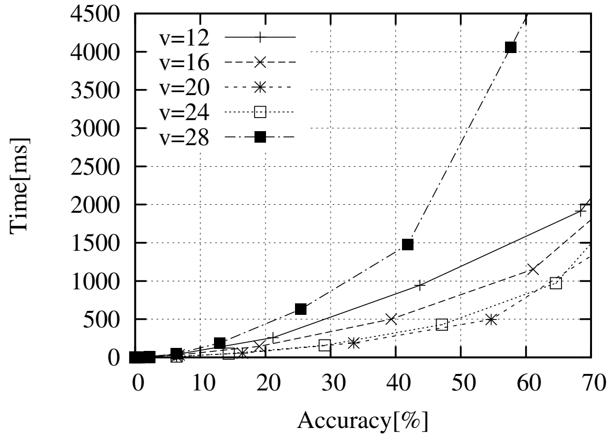
ANN，SH，従来手法 [1]，提案手法で最適なパラメータにおける精度 (真の最近傍点が得られた割合) と処理時間 (クエリを与えてから解を得るまでの時間の平均) の関係と，その時のメモリ使用量を示す．ここでの最適とは同一精度で比較したときに処理時間が最も短くなる状態を指す．予備実験の結果，パラメータとして SH はビット長が $\log_2 n$ である時，従来手法 [1]，提案手法では次元数 $v = \log_2 n \times M$，分割幅 $w = \{\max(\Psi_v \cdot p) - \min(\Psi_v \cdot p)\}/2$ である時が最適であることがわかっている．これらのパラメータはハッシュサイズがデータ数 n と同程度になる値である．

データは 64 次元，128 次元，256 次元の正規分布に基づく人工データ（各基底で分散は 100〜400 で一様に選ばれる）と，TRECVID2010 の Instance Search タスクで配布された動画の各フレーム画像から抽出した SIFT 特徴量 (128 次元) [11](128 次元) の４種類をそれぞれ 1000 万点用意した．クエリはデータベースと同じ条件でつくられた 2000 点を用い，その平均を結果とする．精度と処理時間の関係を図 7(a)〜図 7(c) に，この時のメモリ使用量を表 1 に示す．なお，図は横軸を精度，縦軸を処理時間としており，Single Table は提案手法においてハッシュテーブルを分割しなかった場合，Multi Table はハッシュテーブルを分割した場合の結果である．

実験の結果，全てのデータにおいて同一精度で比較したときに提案手法が最も高速であった．人工データにおいて，Single Table と Mmulti Table を比べると低次元のデータに対しては Single Table の方がわずかに良い結果が得られているが，次元

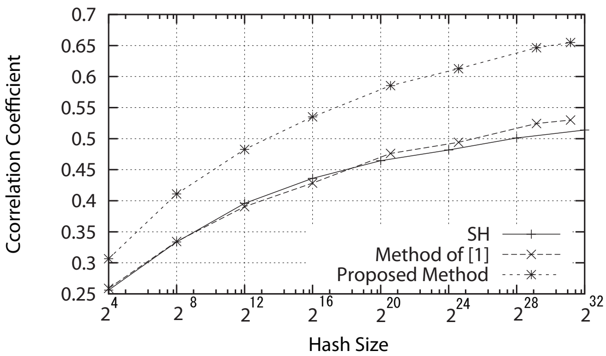


(a) 64 次元

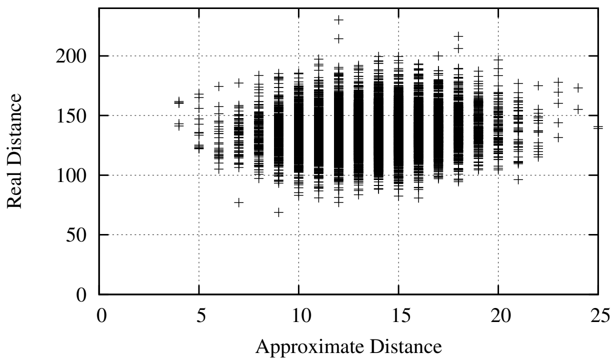


(b) 128 次元

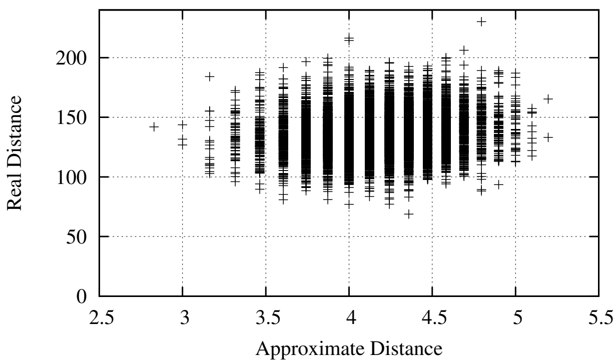
図 5　従来手法において v を変化させた時の精度と処理時間の関係



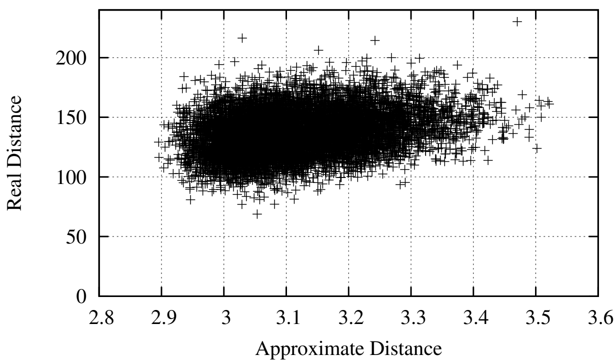
(a) 相関係数



(b) Spectral Hashing



(c) 従来手法 [1]



(d) 提案手法（単一ハッシュテーブル）

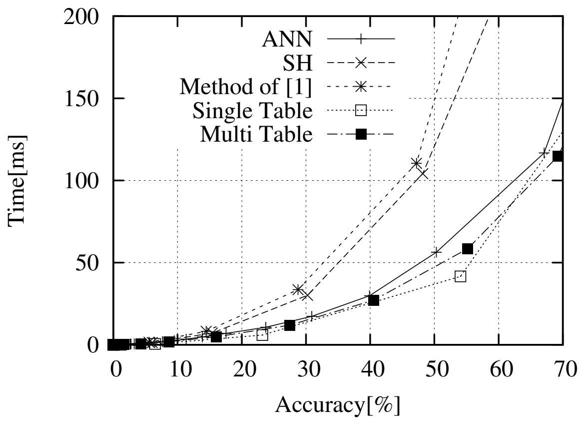
図 6　概算距離と実際の距離の関係

数が大きくなると，Multi Table の有効性が現れる．これは，ハッシュテーブル分割により探索の考慮に入る基底の数が増え，これによって概算距離の精度の低下を抑えることができたからである．故に，高次元データに対してハッシュテーブル分割が有効であると言える．

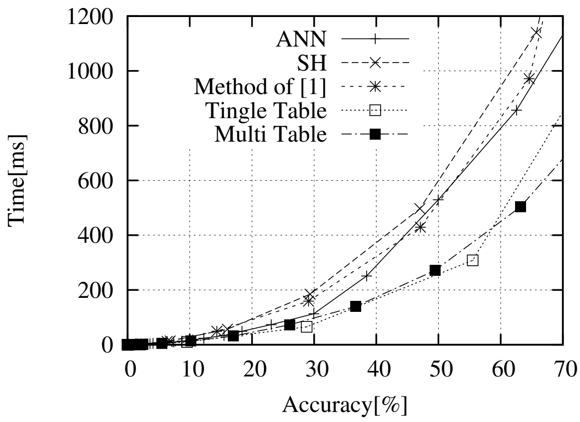
SIFT 特徴量 (128 次元) を見ると，Single Table が優勢である．処理時間を見ると，同制度で比べたときに 64 次元の人工データよりも高速に解を得られていることが分かる．つまり，SIFT 特徴量は見かけは 128 次元であるが，実質的な次元数は半分以下であり，それ故に Single Table が優勢になったと考えられる．

6. ま と め

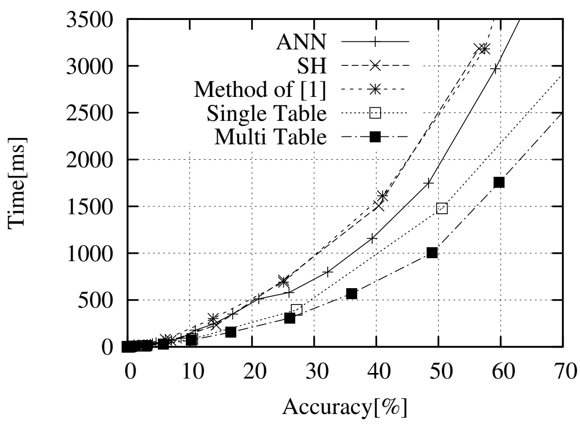
従来手法 [1] 2 点の改良を加えることにより，高精度化つ高速な近似最近傍探索の手法を提案した．１つ目の改良は概算距離の精度向上，２つ目の改良は距離の概算によるすハッシュサイズの低減であった．以上２点の改良により，高次元データに対しても効率的に最近傍候補を抽出することが可能となった..実験１ではハッシュサイズを変化させた時の，概算距離と実際の距離の相関について調べ，同じデータ構造で従来手法 [1] よ



(a) 64 次元 1000 万点



(b) 128 次元 1000 万点



(c) 256 次元 1000 万点

図 7　精度と処理時間の関係（人工データ）

表 1　メモリ使用量

	64 次元	128 次元	256 次元
ANN	3.4GB	5.8GB	11GB
SH	3.0GB	5.3GB	10GB
従来手法 [1]	3.0GB	5.3GB	10GB
提案手法	3.0GB	5.3GB	10GB

りも距離の相関が増加していることが分かった．実験２では従来手法と提案手法の精度と処理時間のトレード・オフの関係を調べ，全てのデータ対して同一精度で比較したときに従来手法よりも高速に解を得ることができた．

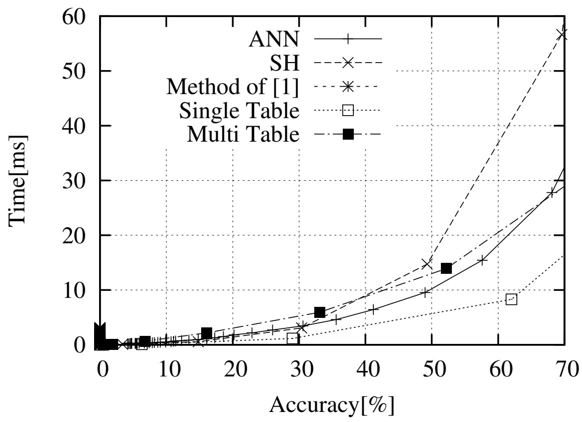


図 8　精度と処理時間の関係（実データ）

謝辞　本研究の一部は科学技術戦略推進費「安全・安心な社会のための犯罪・テロ対策技術等を実用化するプログラム」の一環で実施され，科研費補助金基盤研究 (B)(22300062) ならびに科学技術振興機構 CREST の補助を受けた．ここに記して感謝する．

文　　献

[1] 佐藤智一，武藤大志，岩村雅一，黄瀬浩一，“パケット距離に基づく近似最近傍探索,” データ工学と情報マネジメントに関するフォーラム論文集 E2-6，E2-6，Feb. 2011.

[2] 和田俊和，“最近傍探索の理論とアルゴリズム,” コンピュータビジョン 最先端ガイド 3，第５章，pp.119–136，アドコム・メディア，Dec. 2010.

[3] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, “An optimal algorithm for approximate nearest neighbor searching in fixed dimensions,” Journal of the ACM, vol.45, no.6, pp.891–923, nov 1998.

[4] S.M Omohundro, “Five balltree construction algorithms,” Technical Report, pp.89–063, 1989.

[5] R.F. Sproull, “Refinements to nearest-neighbor searching in k-dimensional trees,” Algorithmica, pp.579–589, 1991.

[6] P.N Yianilos, “Data structures and algorithms for nearest neighbor seach in general metric spaces,” Symposiun on Discrete algorithms, pp.311–321, 1993.

[7] P. Indyk and R. Motwani, “Approximate nearest neighbor: Towards removing the curse of dimensionality,” Proc. 30th Symposium on Theory of Computing, pp.604–613, 1998.

[8] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” Proc. 20th annual symposium on Computational geometry, pp.253–262, 2004.

[9] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probe LSH: efficient indexing for high-dimensional similarity search,” Proceedings of the 33rd international conference on Very large data bases, pp.950–961, VLDB ’07, VLDB Endowment, 2007. http://portal.acm.org/citation.cfm?id=1325851.1325958

[10] Y. Weiss, A. Torralba, and R. Fergus, “Spectral Hashing,” Advances in Neural Information Processing Systems, pp.1753–1760, 2008.

[11] D. Lowe, “Distinctive image features from scale-invariant,” IJCV, vol.60, no.2, pp.91–110, 2004.