2008－CVIM－162 (46)
2008／3／11

一般分布に対する Principal Component Hashing

松下 裕輔† 和田 俊和†

†和歌山大学システム工学部 〒640-8510 和歌山市栄谷 930 番地
E-mail: †{ymatsushita,twada}@vrl.sys.wakaya-u.ac.jp

あらまし 最近傍探索は，事例ベース画像処理・解析などに用いられる重要な処理である．しかし，高次元空間での最近傍探索の高速化は困難であるため，近似最近傍を求めることによって探索の高速化を図るという方法が提案されている．Approximate Nearest Neighbor(ANN)や Locality Sensitive Hashing(LSH)などがその典型例である．特に LSH では誤差比の期待値と計算量の関係が明確であるため，近年特に脚光を浴びている．しかし，LSH は，プロトタイプの分布を参照せずにハッシュ値を計算するため，探索が失敗する，あるいは探索に時間がかかる，という場合がある．我々は，プロトタイプの分布を利用するハッシュ型近似最近傍探索の手法 Principal Component Hashing (PCH)を提案し，これらの問題点を解決した．本報告では，PCH で導入した「プロトタイプ分布は正規分布で近似できる」という仮定を取り除いた手法 A-PCH とその性質について述べる．実験結果から A-PCH は，LSH や ANN，さらに従来の PCH と比べると，同じ精度でより高速であることを確認した．

キーワード 高次元空間，近似最近傍探索，ANN，LSH，PCH

Principal Component Hashing for General Distributions

Yusuke MATSUSHITA†, Toshikazu WADA†

† Faculty of System Engineering, Wakayama University 930 Sakaedani, Wakayama, 640-8510, Japan
E-mail: †{ymatsushita,twada}@vrl.sys.wakayama-u.ac.jp

Abstract Nearest Neighbor (NN) search is an important technique for example based image analysis. However, acceleration of NN search in high dimensional space is a difficult problem. Hence, approximate NN search algorithms, such, as Approximate Nearest Neighbor (ANN) and Locality Sensitive Hashing (LSH) are applied to such problems. Especially, LSH is getting highlighted recently, because it has a clear relationship between relative error ratio and the computational complexity. However, LSH computes hash values without referring the prototype distribution, and hence, sometimes the search fails or the search consumes considerably long time. For solving this problem, we already proposed Principal Component Hashing (PCH), which exploits given prototype distribution. In this report, we extend PCH by removing the assumption, "prototype distribution can be well approximated by Gaussian", which is introduced in PCH. Through the experiments, we confirmed our extended algorithm A-PCH is faster than ANN, LSH, and standard PCH at the same accuracy.

Keyword high dimensional space, approximate NN search, ANN, LSH, PCH

1. はじめに

最近傍探索技術は，最近傍識別器[1]，および，Hallucination[2]や Irregularity detection[3]などの事例ベース画像処理・解析に用いられる重要かつ基本的な処理である．しかし，画像の探索などで必要となる高次元空間での最近傍探索の高速化は困難な問題であるため，完全な最近傍ではなく近似最近傍を求めることによって探索の高速化を図るという方法が検討されている．Approximate Nearest Neighbor(ANN)[4][5]や Locality Sensitive Hashing(LSH)[6][7]などがその典型例である．ANN は k-d tree 同様に，空間を張る軸の分割を繰り返して作られる木構造を用いた探索法であるが，木探索の後に行われる探索(priority search)の際に，探索範囲を狭めることによって近似探索を行っている．具体的には，許容誤差 ε と暫定距離 r (クエリと暫定的な最近傍との距離)で定義される半径 $r/(1+\varepsilon)$ の超球と交わる分割領域のみを探索する．$\varepsilon = 0$ の時は正確な最近傍探索となり，ε が大きくなれば，探索範囲が狭くなるので，探索時間は短くなるが，最近傍を見落とす確率が増加する．これに対し，ハッシュ型の近似最近傍探索法である LSH は誤差比(クエリと最近傍の距離に対する誤差の倍率) の期待値と計算量の関係が明確であるため，近年特に脚光を浴びている．

しかし，LSH では与えられたプロトタイプの分布を参照せずに空間を同一のハッシュ値を持つ複数の「バケット」に分割するため，与えるクエリによっては，1）探索自体が失敗するケースがある，2）探索時間が長くなりすぎるケースがある，などの問題点がある．

例えば，プロトタイプの密度が低い部分にクエリが与えられた時は，対応するバケットにプロトタイプが登録されておらず，探索が行えなくなる．プロトタイプの密度が高い部分では，バケットに登録されたプロトタイプが多いため，最近傍候補の絞込みの効率が悪くなってしまう．

この問題を解決するため，我々はプロトタイプの分布を利用するハッシュ型近似最近傍探索の手法である Principal Component Hashing(PCH)を提案した[13]．この手法では，各バケット内のプロトタイプの個数の期待値を一定にするハッシュと，ハッシュにより求められた最近傍候補集合から効率的に近似最近傍を絞り込む，という利点がある．しかし，登録されているプロトタイプの分布が正規分布に従うと仮定しているため，その分布に従わない場合は，バケット内のプロトタイプの個数の期待値を一定にすることができなくなる．

本報告では，上記の PCH の問題点を解決し，一般の分布に対してもバケット内のプロトタイプの個数の期待値を一定にすることができる PCH の拡張法を提案する．実験では，PCH の性能を ANN，LSH を用いた探索精度と速度に関する比較実験を通じて確認する．また，PCH の拡張の効果を従来の PCH との比較実験から示す．

2. 近似最近傍探索

現在までに提案されている高速最近傍探索アルゴリズムは，以下の２つを利用したものが殆どである．

- 三角不等式を利用して，距離計算対象を絞込む．[9][10][11][12]
- 近似最近傍を求め，その距離よりも距離が小さくなり得ないプロトタイプの距離計算を打ち切る．[4]

しかし，これらの手法は高次元空間において効果を失い，ほぼ全探索と同程度の計算効率になってしまうことが知られている．

2.1 $R(c)$ 最近傍問題

距離空間 $\boldsymbol{X}$ 中の要素 x_1, x_2 間の距離を $D(x_1, x_2)$，プロトタイプの集合を $\boldsymbol{S} \subset \boldsymbol{X}$ で表す．このとき，クエリ q に対する最近傍を $\boldsymbol{NN}(q) \in \boldsymbol{S}$ としたとき，

$$D(q, \boldsymbol{NN}'(q)) \leq cD(q, \boldsymbol{NN}(q)) \qquad (1)$$

を満足する近似最近傍 $\boldsymbol{NN}'(q)$ を求める問題を「$R(c)$ 最近傍問題」，c を「誤差比」と呼ぶ．

この問題に対して，以下の事柄が示されている．

［定義１］

$\boldsymbol{X}$ からハッシュ値の集合 $\boldsymbol{U}$ への写像を行なうハッシュ関数 $h(x): \boldsymbol{X} \rightarrow \boldsymbol{U}$ のうち，以下に示す２条件が成り立つものを Locality Sensitive であると言う．

- $D(v,q) \leq r_1$ であるとき，$\Pr[h(q) = h(v)] \geq p_1$ が成り立つ．
- $D(v,q) > r_2$ であるとき，$\Pr[h(q) = h(v)] < p_2$ が成り立つ．

但し，$p_2 \leq p_1$ 且つ $r_2 = cr_1$ である．

［定理１］

Locality Sensitive なハッシュ関数の系列 $h_1, h_2, \cdots$ が与えられたとき，n 個のプロトタイプから $D(q, \boldsymbol{NN}'(q)) \leq cD(q, \boldsymbol{NN}(q))$ を 1/2 以上の一定確率で満足する $\boldsymbol{NN}'(q)$ が $L = n^{\rho(c)}$ 回のバケット検索で求められる．但し，$\rho(c) = \ln p_1 / \ln p_2$ である．

この定理に基づく近似最近傍探索法が LSH である．LSH において距離計算の効率を規定するのは，$\rho(c)$ である．この関数は，c に対する単調減少関数であり，より急速に減少する関数を求めた方が効率および精度共によい．

問題は，どのようなハッシュ関数が性質の良い $\rho(c)$ をもたらすのかということであるが，これに関しては現在様々な研究が行われている．

2.2 p-stable LSH

p-stable LSH[7]では，一般の距離空間ではなく，ベクトル空間内でのユークリッド距離に基づく探索問題に絞ったハッシュ関数が用いられている．これは入力クエリベクトルを $\boldsymbol{q}$ としたとき，あるベクトル $\boldsymbol{a}$ と定数 b, ω によって，次式で定義される．

$$h_{\mathbf{a},b}(\boldsymbol{q}) = \left\lfloor \frac{\mathbf{a} \cdot \boldsymbol{q} + b}{\omega} \right\rfloor \qquad (2)$$

但し，$\lfloor \cdot \rfloor$ は Gauss 記号である．

式(2)中の ω は空間を何個のバケットに分割するかを規定するパラメータであり，これは $\boldsymbol{a}$ と各プロトタイプとの内積値の分布幅に基づいて決められる．b はバイアスを調整するためのものであり，$[0, \omega]$ の区間内で発生させた一様乱数から選ばれる．$\boldsymbol{a}$ は各次元が独立な正規乱数に従って生成されるベクトルである．

このハッシュ関数を用いることによって，$\rho(c)$ も，[6]で提案されていた $1/c$ より低く押さえられることが示されている．最近では，$\rho(c) = 1/c^2$ とすることができるといった研究[8]も発表されているが，プロトタイプの Voronoi 分割を利用するという高次元空間では実行不可能な計算を含んでおり，高性能 LSH の実現の

ためには更なる研究が必要である．

3. Principal Component Hashing

我々は，プロトタイプの分布を利用するハッシュ型近似最近傍探索の手法である Principal Component Hashing(PCH)を提案した．PCH は，以下に示す特徴を持っている．

- 各バケット内に入るプロトタイプの個数の期待値が一定であり，かつ空間を隙間なく埋めるようにバケットの配置を行う．このため，プロトタイプの分布から密度が低い部分にクエリが与えられても，ハッシュ値に対応するバケットを見つけることができる．また，プロトタイプの密度が高い部分にクエリが与えられても，バケット内のプロトタイプの数が一定になるため，探索の効率が良くなる．
- ハッシュ値の計算では，主成分分析で求めた寄与率の高い軸を用いている．これを利用し，寄与率の高い射影軸上の成分から距離計算を行う．この距離計算の途中で，暫定距離(現時点での最近傍であるプロトタイプと，クエリとの距離)を超えたら，距離計算を打ち切れる．また，重複度(クエリと同じバケットになった回数)の大きいプロトタイプから距離計算を行う．このため，絞込みの早い段階で最近傍に近い暫定距離が求まるため，効率のいい絞込み探索ができる．

以下に，PCH のハッシュ関数と探索アルゴリズムについて述べる．

3.1 ハッシュ関数

PCH では，全体の分布が正規分布に従うと仮定する．このため，主軸上の周辺分布も正規分布に従い，その射影軸上のプロトタイプの確率密度分布 $p(x)$ を正規分布で近似することができる．この確率密度分布から累積確率 $P(x)$ を求め，$P(x)$ を等間隔に区切ったものをバケットとする(図 1).

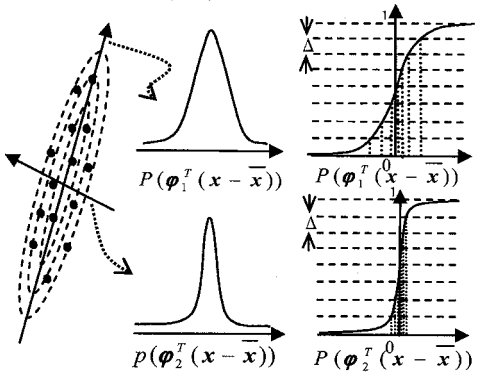

図 1. PCH におけるハッシュ関数とバケット分割

以下に示すのは，確率密度分布から累積確率を求める式である．

$$P(x)=\int_{-\infty}^{x} p(\xi)d\xi \qquad (3)$$

ここで，PCH では，正規分布の積分を次に示す式で近似している．

$$P_s(x)=\frac{1}{1+e^{-x/a^2}} \qquad (4)$$

この近似によって，i 番目の主軸 $\boldsymbol{\varphi}_i$ を用いた際のハッシュ関数は，次式のように表される．

$$h_i(\boldsymbol{x})=\left\lfloor P_s(\varphi_i^T(\boldsymbol{x}-\bar{\boldsymbol{x}}))/\Delta \right\rfloor \qquad (5)$$

但し，$\boldsymbol{\varphi}_i$ は正規直交基底であり，$\Delta\in(0,1)$ はバケット間隔と呼ばれる．

3.2 探索アルゴリズム

PCH の探索手順を以下に示す．但し，**Step1,2** は，探索の前に行っておく前処理である．

Step1. 主成分分析を行い，主軸を求める．

Step2. 探索に用いる主軸を決定し，選ばれた主軸毎に式(5)のハッシュ関数を求める．求めたハッシュ関数を用いて，プロトタイプをバケットに登録する．

Step3. 前処理で求めたハッシュ関数を用いて，主軸毎にクエリの属するバケットを検索する．

Step4. 検索された各バケット内にあるプロトタイプの和集合を最近傍候補 $\boldsymbol{C}(\boldsymbol{q})$ とする．また，探索精度を向上させるために両脇 δ 個の隣接バケットまで見る．δ をマージンと呼ぶ．さらに，高速化のため，重複度が大きい順にソーティングを行い，上位 $b\%$ のみを最近傍候補とする．b をカットオフ比と呼ぶ．

Step5. $\boldsymbol{C}(\boldsymbol{q})$ に対して距離計算の打ち切りによって，絞込みを行い，探索する．プロトタイプ $\boldsymbol{x}$ の重複度を $w(\boldsymbol{x})$ と表す．このとき，$w(\boldsymbol{x})$ が最大の $\boldsymbol{x}$ とクエリ $\boldsymbol{q}$ との距離を計算し，その結果を暫定距離 z とする．

$$z=D(\boldsymbol{q},\arg\max_{\boldsymbol{x}\in\boldsymbol{C}(\boldsymbol{q})} w(\boldsymbol{x})) \qquad (6)$$

この暫定距離を用いて距離計算の打ち切りを行う．以降の議論では，L_p 距離として $D_p(\boldsymbol{x}_1,\boldsymbol{x}_2)$ 距離を用いることにする．M 次元ベクトルの場合，距離は次式のように表される．

$$D_p(\boldsymbol{x}_1,\boldsymbol{x}_2)=\sqrt[p]{\sum_{i=1}^{M}(\boldsymbol{x}_{1i}-\boldsymbol{x}_{2i})^p} \qquad (7)$$

Parseval の等式より，任意の正規直交基底への射影成分から L_p 距離は計算できる．したがって，

$\boldsymbol{C}_0(\boldsymbol{q})$ 内の候補 $\boldsymbol{x}$，$m\le M$ について

$$\sum_{i=1}^{m}\left|\varphi_i^T\boldsymbol{q}-\varphi_i^T\boldsymbol{x}\right|^p > z^p \qquad (8)$$

が成立すると，このプロトタイプは最近傍候補になり得ないため，距離計算を打ち切る．$m = M$ について，式(8)が成立しなければ，式(7)より $D_p(\boldsymbol{q},\boldsymbol{x})$ を求める．このとき，$z > D_p(\boldsymbol{q},\boldsymbol{x})$ ならば，$z = D_p(\boldsymbol{q},\boldsymbol{x})$ とし暫定距離の更新を行う．$\boldsymbol{C}(\boldsymbol{q})$ のすべてのプロトタイプに対して，絞込みを行った後で，暫定距離であるプロトタイプを近似最近傍として，処理を終了する．

4. 一般の分布への拡張

PCH のハッシュ関数では，プロトタイプが正規分布に従うという仮定を導入することで，その分布に応じたバケットの分割を行っていた．このため，分布のモデルに従わないデータが与えられた場合，バケット内のプロトタイプの個数を一定にできず，効率的な探索が行えない．

この問題を解決し，一般の分布に対応するためには，あらかじめプロトタイプの分布のモデルを仮定しておくのではなく，登録されたプロトタイプから分布のモデルを推定する必要がある．

以下に，新しいバケットの分割と検索の方法について述べる．以降，従来の PCH を **PCH** とし，一般の分布への拡張を行った PCH を **Adaptive PCH(A-PCH)** と呼ぶことにする．

4.1 新しいバケットの分割

一般の分布に対応するためには，主軸毎にプロトタイプの分布を調べる必要がある．このため，射影軸上のプロトタイプの累積度数を等間隔に区切るようにバケットの分割を行う．これは，以下に示す理由によるものである．

- 累積度数をプロトタイプの個数で割ったものは，累積確率と同じである．
- バケットの検索には，主軸の分割位置に対して二分探索を行うことで，複雑な計算をせずにバケットを見つけることができる．

このように，主軸毎にプロトタイプの累積度数を調べることで，プロトタイプの分布のモデルをあらかじめ用意する必要がない．また，確率密度関数では，表現が困難な分布に対しても，バケット内のプロトタイプの個数を一定にするようなバケットの分割を行うことができる．

ここからは，具体的なバケットの分割方法について述べる．プロトタイプを射影軸上の座標をもとに，最も座標が小さいものから順に $1,2,\cdots,n$ と番号を割り振る．この番号は，対応する座標以下にあるプロトタイプの総数と同じなので，射影軸上のプロトタイプの累積度数を表している。この累積度数が等間隔になるように区切ったものをバケットとする(図 2).

ここで，i 番目の主軸上の左から j 番目の分割位置 d_{ij} は，次式のように示される．

$$d_{ij} = c_i(\lfloor \Delta \cdot n \rfloor \cdot j) \qquad (9)$$

但し，$c_i(x)$ は，i 番目の主軸上の x 番目のプロトタイプの座標である．

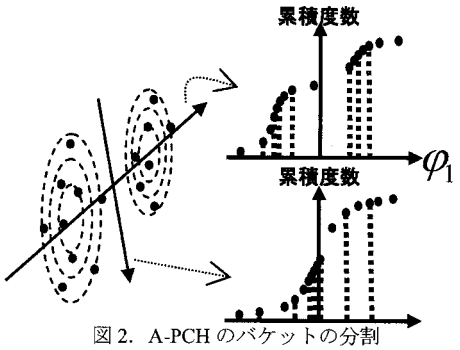

図 2. A-PCH のバケットの分割

4.2 新しいバケットの検索

プロトタイプの登録や，最近傍候補の生成を行うためのバケットの検索には，主軸の分割位置に対して二分探索を行う．PCH と A-PCH のバケットの検索の違いについて以下に示す．

- 前処理の時に，A-PCH では主軸の分割位置を求めておく．これは，射影軸上の座標をもとにプロトタイプをソーティングすることで求められる．それに対して，PCH では式(4)の累積確率の近似の精度によって，前処理に掛かる時間やバケットの配置が変化してしまう．
- A-PCH のバケット検索は，射影軸上の座標の大小比較によって，バケットの検索を行うことができる．これに対して，PCH では式(5)の計算を行い，そのハッシュ値に対応するバケットを検索結果とする．このため，射影軸上の座標を累積確率に変換する必要があり，複雑な計算が必要となる．

5. 実験

PCH の性質と今回の拡張の効果を示すために，以下に示す実験を行う．

- シミュレーションデータと実画像データを対象とした A-PCH，LSH，ANN の性能比較
- A-PCH の拡張の効果を評価するため，PCH との性能比較

各実験について説明する．従来手法との比較実験では，プロトタイプの分布を利用することによる探索精度，探索速度の向上の効果を確認する．この実験では，シミュレーションデータとして等方性分布に従うデータと，実画像データとして顔画像データベース CAS-PEAL の画像データを用いた．次に，PCH と A-PCH との比較実験では，今回の拡張によって，正規分布に従わないデータに対しても効率的な探索ができることを確認する．この実験では，前述の実験で用いた 2 種類のデータと，今回の拡張の効果を顕著にするため 2 つの正規分布が混在するデータを用いた．

LSH，ANN，PCH を対等に比較するために，探索時間と誤差比のグラフを用いる．この探索時間と誤差比は，クエリを 1000 個与えた時の平均から求めている．但し，LSH はクエリに対して最近傍が求まらないケースがあるので，誤差比は，探索失敗が起きてない場合にのみ，その平均値を求める．実験に用いた計算機は，CPU :Intel Xeon 3.7GH z ×2，Memory:32GB を用いた．

5.1 A-PCH，LSH，ANN の性能比較

この実験では，以下に示すデータを用いた．

1. 等方性分布のデータ(各軸 $N(0,\sigma)$ に従う 3000 次元，10000 個のデータ)に対して，各軸 $(-3\sigma,3\sigma)$ の一様分布に従うクエリを与えたもの．
2. 画像データベース CAS-PEAL に含まれる画像データ(画像サイズ 4096(64*64)，10000 個のデータ)に対して，同じ画像を含まないクエリを与えたもの．

等方性分布のデータを用いた実験では，プロトタイプの分布と異なる入力が与えられた場合の各手法の比較を行う．また，CAS-PEAL の画像データを用いて，実画像データに対しての有効性の評価を行う．

実験に用いた各手法のパラメータの設定値は以下の通りである．

- ANN：許容誤差 ε を 1 から 100 まで変化．
- LSH：1 つのバケットを求めるのに必要なハッシュ関数の数 k を 2 から 40 まで変化，探索時に求めるバケットの数 L を 3 から 171 まで変化させる．
- A-PCH：主軸の本数 A を 5 から 100 まで変化，主軸の分割数 $1/\Delta$ を 5 から 100 まで変化，マージン δ を 0 から 2 まで変化，カットオフ比 b を 100(カットオフを用いない)，80，60，40，20%に変化させる．

図 3 には等方性分布のデータ，図 4 には CAS-PEAL のデータを用いた場合の探索時間と誤差比の関係を示す．横軸を探索時間:time[s]，縦軸を誤差比:error ratio とする．誤差比は,値が 1 に近いほど良く，値が 1 ならば最近傍を探索したことを示す．左下にプロットされているものほど探索時間，探索精度が優れているものを示す．

図 3 より，LSH では，プロトタイプの分布から離れた位置にクエリが与えられた時に，ハッシュ値が一致するプロトタイプがなくなる．これに対して，A-PCH では各バケット内のプロトタイプ数の偏りをなくすことで，効率的な探索ができたことが分かる．このデータの場合，同一の探索時間での誤差比，同一誤差比での探索時間は PCH が最も良く，次に ANN，LSH の順になった．次に，図 4 では，プロトタイプとクエリの分布が一致しているため，LSH は ANN よりも探索が良くなっている．また，等方性のデータに比べ，実画像データは，プロトタイプの分布に空間的な偏りがあるため，距離計算の打ち切りによる絞込みの効果が表れている．このため，図 3 の結果と比べても A-PCH は，ANN や LSH より高速かつ精度の良い探索ができていることが分かる．

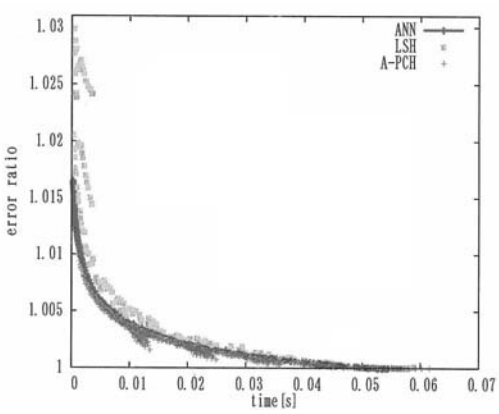

図 3　等方性分布のデータの場合

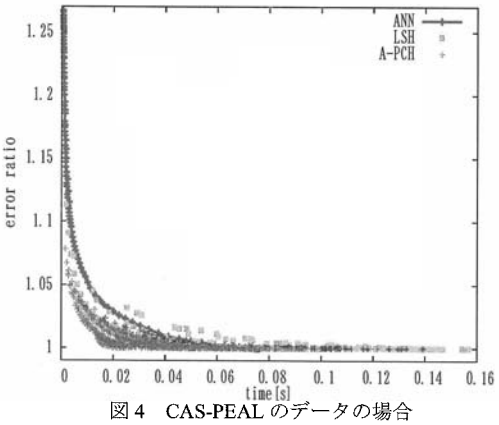

図 4　CAS-PEAL のデータの場合

5.2 従来の PCH との性能比較

4 章で述べた一般の分布に対する拡張の効果を確かめるために，PCH と一般の分布に対する拡張を行った PCH である A-PCH との比較実験を行った．比較を行うため，**5.1** の実験に用いた 2 つのデータに加え，以下に示すデータを用いた．

3. 2 つの正規分布が混在するデータ(各軸 $N(3\sigma,\sigma)$ に従うか，各軸 $N(-3\sigma,\sigma)$ に従うかをランダムに決定した 3000 次元，10000 個のデータ)に対して，同じ分布に従うクエリを与えたもの．

このデータでは，PCH はバケット内のプロトタイプの個数を一定にすることができないため，効率的な探索ができない．

この実験では，両手法のパラメータの設定値は，**5.1** の実験で用いた A-PCH と同様の設定とする．また，両手法の比較には，探索時間と誤差比の関係を用いる．

図 5 には等方性分布のデータ，図 6 には CAS-PEAL のデータ，図 7 には 2 つの分布が存在するデータを用いた場合の結果を示す．図 5，図 6 では，PCH と A-PCH にほとんど差が見られなかった．これは，プロトタイプの分布がほぼ正規分布に従っているため，両者のバ

ケットの分割にほとんど差がないことを表している．図7では，A-PCHと比較してPCHは，探索時間や誤差比の変化の幅が狭い．これは，第一主軸の分布では分布の中心から離れた狭い範囲にプロトタイプが集中するため，主軸の分割数を増やしても，空のバケットのみが増えるためである．

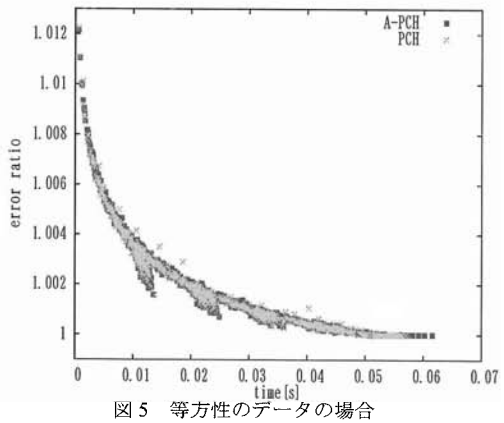

図5　等方性のデータの場合

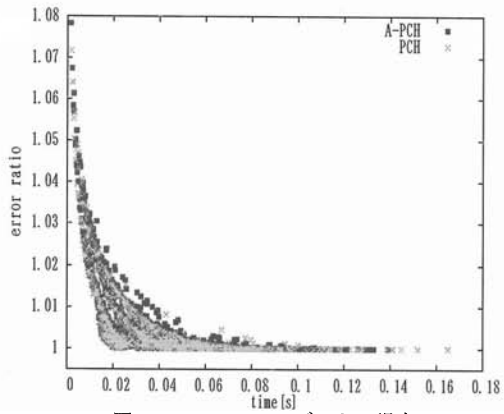

図6　CAS-PEALのデータの場合

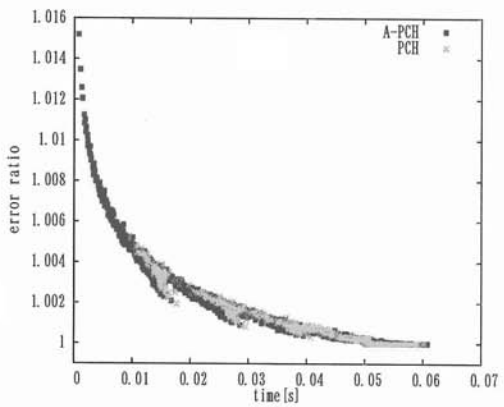

図7　2つの正規分布が混在する場合

次に，PCHとA-PCHの差を定量的に示すため，同じパラメータを用いたときのPCHに対するA-PCHの探索時間，誤差の割合を図8に示す．この結果は，実験に用いたすべてのパラメータ毎に割合を求め，その平均を示している．図8より，2つの分布が存在するデータに対しては，誤差をほとんど変化させずに，探索時間を74%にできることが分かる．

データの種類	探索時間の割合	誤差の割合
等方性分布	0.996762	1.000021
CAS-PEAL	0.998023	1.000538
2つの分布が存在するデータ	0.740392	1.001116

図8　PCHに対するA-PCHの探索時間，誤差の割合

6. まとめ

本研究では，一般の分布に対するPCHの拡張について提案した．この手法は，あらかじめプロトタイプの分布を仮定せずに，登録されたプロトタイプのみから分布に応じたバケットの配置を行うので，効率的な探索を行うことができる．

文献

[1] T. M. Cover, and P. E. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, Vol. IT-13, No.1, pp.21-27, 1967

[2] S. Baker and T. Kanade, "Limits on Super-Resolution and How to Break Them", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 9, September 2002, pp. 1167-1183.

[3] O. Boiman, M. Irani, "Detecting Irregularities in Images and in Video," In Proc. of ICCV 2005, pp. 462-469, 2005, Beijing

[4] S. Arya, , D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching," Journal of the ACM, Vol.45, pp. 891-923, 1998

[5] ANN: Library for Approximate Nearest Neighbor Searching (http://www.cs.umd.edu/~mount/ANN/)

[6] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," In Proceedings of the 30th ACM Symposium on Theory of Computing (STOC'98), pp.604-613, May 1998.

[7] M. Datar, P. Indyk, N. Immorlica, and V. Mirrokni, "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions," In Proceedings of the 20th Annual Symposium on Computational Geometry(SCG2004), June 2004.

[8] A. Andoni, P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions", In Proc. of FOCS'06, pp. 459-468, 2006

[9] Vidal, R., "An algorithm for finding nearest neighbor in (approximately) constant average time," Pattern Recognition Letters, No. 4, pp. 145-158, 1986

[10] Mico, L., Oncina, J., and Vidal, E., "A new version of the nearest-neighbor approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements," Pattern Recognition Letters, No. 15, pp. 9-17, 1994

[11] Brin, S., "Near neighbor search in large metric spaces," in Proc. of 21st Conf. on very large database (VLDB), Zurich, Switzerland, pp. 574-584, 1995

[12] Yianilos, P. Y., "Data structures and algorithms for nearest neighbor search in general metric spaces," in Proc. of the Fourth Annual ACM-SIAM Symp. on Discrete Algorithms, Austin, TX, pp. 311-321, 1993

[13] 松下　裕輔，和田　俊和，"Principal Component Hashing"，画像の認識・理解シンポジウム (MIRU2007)，pp.127-134，JULY　2007.