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.

近さの多段階表現に基づく近似最近傍探索

多田　匡志† 　武藤　大志† 　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: †{tada,mutoh}@m.cs.osakafu-u.ac.jp, †{masa,kise}@cs.osakafu-u.ac.jp

あらまし　近似最近傍探索は，クエリと最も距離が近い点を探索する最近傍探索問題において，探索精度を犠牲にすることで計算時間，メモリ使用量を大幅に削減する手法である．本稿では，検索精度が同一であれば，ハッシュを用いた近似最近傍探索手法である LSH や PCH より少ない計算時間とメモリ使用量で達成する手法を 2 種類提案する．1 つ目の提案手法は，近さの情報を多値化して処理の効率化を図り，計算時間を LSH の約 50%に削減した．2 つ目の提案手法は，既存の手法では不可欠だった距離計算の処理を省き，LSH に対してメモリ使用量を約 50%，計算時間を約 50%削減した．

キーワード　近似最近傍探索, Locality Sensitive Hashing, Principal Component Hashing, 計算時間，メモリ使用量, 探索精度

Approximate Nearest Neighbor Search Based on a Multi-Valued Expression of Closeness

Masashi TADA†, Tomoyuki MUTO†, Masakazu IWAMURA†, and Koichi KISE†

† Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuencho, Naka, Sakai, 599-8531 Japan
E-mail: †{tada,mutoh}@m.cs.osakafu-u.ac.jp, †{masa,kise}@cs.osakafu-u.ac.jp

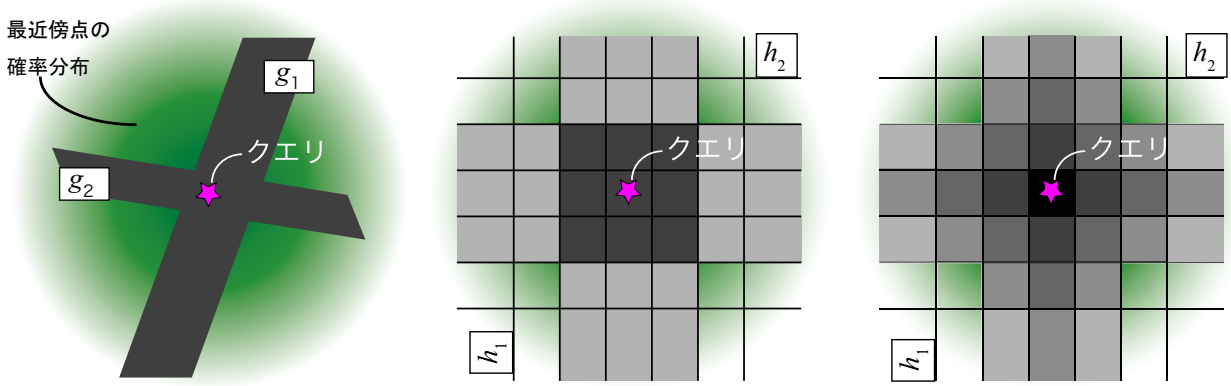
Abstract　Approximate nearest neighbor search is a technique which greatly reduces processing time and required amount of memory for nearest neighbor search. In this paper, we propose two methods which achieve the same accuracy, with less processing time and less required amount of memory compared to existing methods such as LSH and PCH. The first one achieved about 50% of processing time as compared to LSH by using multi-valued information for improving processing efficiency. The second one achieved about 50% of required amount of memory and about 50% of processing time as compared to LSH by eliminating distance calculation process which is requisite for existing methods.

Key words　Approximate nearest neighbor search, Locality sensitive hashing, Principal component hashing, Processing time, Required mount of memory, Retrieval accuracy

1. 前　書　き

近年の計算機性能の向上と画像照合技術の発展によって，大規模データベースを用いた物体認識研究が盛んに行われている．例えば，Hervé ら [1] や野口ら [2] は，約 100 万枚の画像の中からクエリ画像と一致する画像を高速に検索する手法を提案している．これらを実現するためには，いずれの場合も画像から SIFT [3] や PCA-SIFT [4] などの特徴量ベクトルを抽出し，データベースに登録されている特徴量ベクトルと類似のものを探索する必要がある．この探索において，検索精度が重視されるのか，それとも高速性や省メモリが重視されるのかはアプリケーションに依るが，検索精度が同一であれば，計算時間とメモリ使用量が小さいほうが良いのは言うに及ばない．

本稿では，上記のような探索問題のうち，最近傍探索と呼ばれるものを扱う．改めて問題を定義しておくと，最近傍探索とは，ベクトルで表現されるデータ（点）の中から，クエリと最も類似している（距離が小さい）データ（以後，最近傍点と呼ぶ）を探し出す問題である．大規模データに対する最近傍探索問題において，最近傍点を正確かつ高速に求めるためには膨大な計算時間やメモリ使用量を必要とするため，最近傍探索の誤りを許容することで，計算時間とメモリ使用量を大幅に削減できる近似最近傍探索が近年注目されている．



(a) LSH　(b) PCH　(c) 提案手法

図 1　LSH，PCH，MVH（提案手法）の距離計算対象の選定の様子．各図の背景は最近傍点の確率分布であり，クエリに近い程最近傍点が存在する確率は高くなることを表している．

本研究では，近似最近傍探索手法のうち，特にハッシュを使用する方法に注目し，探索精度を維持しつつ，計算時間とメモリ使用量を削減する方法を検討する．基礎的な検討を行うために，データが一様分布に従う場合を考える．検探索精度としては，探索によって求めた近似最近傍点が真の最近傍点と一致する確率を用いる．

ハッシュを用いる近似最近傍探索の代表的な手法として，Locality Sensitive Hashing(LSH) [5]〜[7] がある (注1)．LSH は，複数のハッシュ関数を利用して，クエリの近傍にあると考えられる特徴点を選出し，それらの特徴点に対してのみ距離計算を行う．LSH の計算時間は，距離計算に要する時間が支配的であるため，距離計算の候補を絞れば絞る程計算時間を削減することができる．しかし，絞り過ぎると真の最近傍点が距離計算の対象に含まれないため，正しく探索できない可能性が高くなる．つまり，最近傍点である可能性の高い点をうまく絞り込むことが求められる．

距離計算の対象になる点を効率良く選定する手法として，Principal Component Hashing(PCH) [8] や Adaptive PCH(A-PCH) [9] が提案されている (注2)．LSH では，複数のハッシュを調べて，1 回でもクエリと同じビンに入った点を全て距離計算の候補としたが，PCH と A-PCH では，クエリと同じビンやその近傍のビンに入った回数を数え，回数が多い点のみを距離計算の候補とする．これにより，LSH より効率の良い探索が実現できる．

本稿では，距離計算の対象をさらに効率良く選定するために，クエリとの近さに応じた重みを導入した 2 種類の手法を提案する．1 つ目は，各ハッシュにおいて，クエリに近い点に大きな重みを，遠い点に小さな重みを与え，重みの総和が大きい点のみを距離計算の候補とする手法である．この重みの総和はクエリの近傍では L_1 距離を概ね反映しているため，距離計算の候補を PCH よりも正確に求めることができる．しかも，重みを導入しても PCH より計算量が増加しないという優れた性質を持つ．2 つ目は，1 つ目の手法の距離計算を省く手法である．すなわち，重みの総和が大きい点を最近傍点と決定する手法である．このようにすれば，距離計算のために保持しておく必要があった特徴ベクトルを一切使用しなくても良いので，メモリ使用量の大幅な軽減が実現できる．

(注1)：正確には，LSH は近似最近傍探索手法ではなく，クエリの近傍点を効率良く探索する近似近傍探索手法である．本稿では，本来の意味の LSH で求めた近似近傍点に距離計算を適用することで近似最近傍探索を実現しており，この手法を LSH と呼ぶ．

(注2)：本稿ではデータが一様に分布していると仮定した場合の特殊な PCH を PCH と呼ぶ．本来，PCH や A-PCH はデータが非一様に分布している場合に力を発揮するように設計されているため，これは PCH にとって最も不利な条件であるが，提案手法の有効性を検証するために条件を揃えた．

2. ハッシュを用いる近似最近傍探索手法

本研究に関連するハッシュを用いた既存の近似最近傍探索を概説する．

2.1 Locality Sensitive Hashing(LSH)

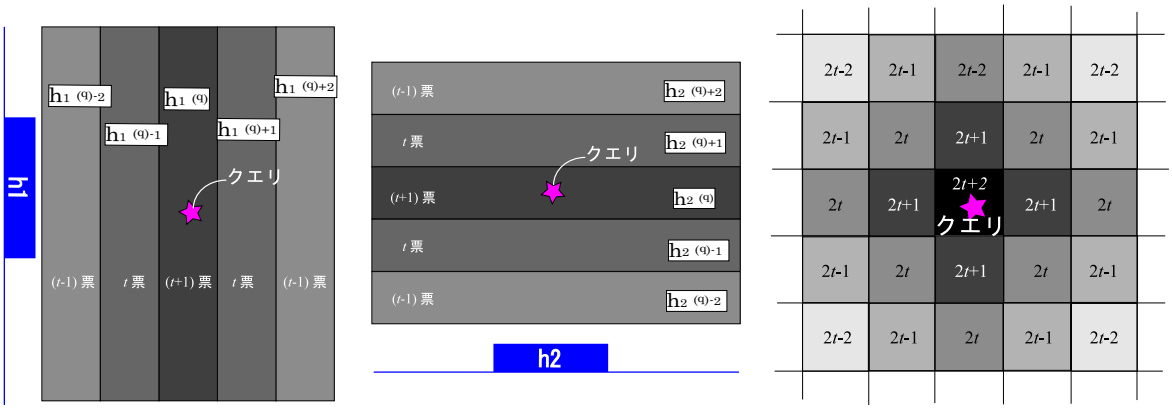
Locality Sensitive Hashing(LSH) [5]〜[7] はハッシュを用いた近似最近傍探索の代表的手法である．ここでは，LSH の中でもベクトル空間で用いることが出来る文献 [6] の LSH について，本研究と関連のある部分のみを述べる．LSH は，複数のハッシュを用いて距離を計算する対象になる点を求め，それらの点とクエリとの距離計算を行うことにより近似最近傍点を求める．

LSH が近似最近傍点を求めることができるのは局所性に鋭敏な (Locality Sensitive) ハッシュ関数を用いているためである．局所性に鋭敏なハッシュ関数とは，距離が近い点同士は同じハッシュ値を取る確率が高く，距離が遠い点同士は同じハッシュ値を取る確率が小さいハッシュ関数である．文献 [6] の LSH では次式を用いて局所性に鋭敏なハッシュ関数を実現している．

$$h_{ji}(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{a}_{ji} \cdot \boldsymbol{p} + b_{ji}}{w} \right\rfloor \tag{1}$$

ただし，$\boldsymbol{a}_{ji}$ は各次元の要素の値をガウス分布から独立にとったベクトル，$\boldsymbol{p}$ はデータ点，b_{ji} は区間 $[0, w]$ からランダムに選ばれた実数であり，w はハッシュ幅である．

LSH では，ハッシュ関数 h_{ji} を k 個組み合わせてハッシュ関



(a) $h_1(q)$ の投票処理　(b) $h_2(q)$ の投票処理　(c) $h_1(q)$ と $h_2(q)$ の投票情報の和

図 2　提案手法の投票処理

数群 g_j を作る．図 1(a) は LSH を用いて濃着色の距離計算の対象領域にある点を求める様子である．図 1(a) の g_1 は $k=2$ のとき，h_{j1} と h_{j2} の 2 つのハッシュ関数で求めた距離計算の対象領域の共通部分である．このようなハッシュ関数群 g_j を L 個作り，L 個の領域を組み合わせた領域を距離計算の対象領域とする．図 1(a) は $L=2$ の場合であり，図 1(a) の着色部分は，g_1 と g_2 の 2 つのハッシュ関数群を用いて得られる距離計算の対象領域を図示したものである．LSH は，上記の手順で距離計算の対象を削減して処理を高速化する．

2.2　Principal Component Hashing

Principal Component Hashing(PCH) [8] は，LSH と同様に距離計算を必要とするが，距離計算の候補を効率的に求めることで性能の向上を図っている．PCH は本来，データの主成分分析を行い，データの分布を考慮した処理を行うのだが，本稿ではデータが一様分布に従う場合のみを考慮するため，データが一様に分布している場合の特殊な PCH を以下で述べる．

PCH で用いられるハッシュ関数は

$$h_i(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{\varphi}_i \cdot \boldsymbol{p}}{w} \right\rfloor \tag{2}$$

である．$\boldsymbol{\varphi}_i$ は正規直交基底，$\boldsymbol{p}$ はデータ点，w はハッシュ幅である．PCH では，各ハッシュ関数 $h_i(p)$ について，各点がクエリと同じビンに入った回数を数え上げ，これを重複度と呼ぶ．そして，重複度が高い点のみを距離計算する．また，LSH ではクエリ点とハッシュ値が同じビンだけを探索していたのを，距離計算の候補に最近傍点が含まれる確率を高めるため，PCH ではその両脇 δ 個までの隣接するビンを参照する．

図 1(b) は，$\delta=1$ の時，ハッシュ関数 h_1，h_2 を用いて得られる距離計算の対象領域を図示したものである．h_1 と h_2 の両方に参照される領域，つまり重複度が高い領域は濃い着色部分で示した．その領域にある点に対してのみ距離計算を行い，近似最近傍点を得る．

3. 提 案 手 法

本節ではハッシュを用いた近似最近傍探索を改良した提案手法（Multi-Valued Hashing; MVH）を 2 種類示す．

3.1　ハッシュを用いた既存の近似最近傍探索の問題点

ハッシュを用いた近似最近傍探索の処理は大きく分けると 2 段階ある．1 段階目の処理において，ハッシュを用いて距離計算の対象となる点を選ぶ．2 段階目の処理では，1 段階目の処理によって得られた点とクエリの距離を計算して，近似最近傍点を得る．

1 つ目の提案手法（MVH1 と呼ぶ）では，1 段目の距離計算対象の選定能力を改良する．MVH1 は，LSH や PCH のように近さの情報を「近い」と「近くない」の 2 値情報で表現するのではなく，多値情報で表現することで，図 1(c) のようにクエリの近傍では L_1 距離の概算値を反映した情報を得ることができる．

2 つ目の提案手法（MVH2 と呼ぶ）では，MVH1 から距離計算の処理を省いた手法を提案する．このようにするのは，MVH1 で L_1 距離の概算値を反映した情報を得ることができるならば，2 段目の処理は不要と考えられるからである．

ここで，各手法の比較を表 1 にまとめておく．

3.2　提案手法１：MVH1

既存のハッシュを用いる近似最近傍探索に，近さを多値化する処理を導入して性能の向上を図る手法を提案する．図 1(a) と図 1(b) に示すように，LSH や PCH では「近い」か「近くない」かの 2 値情報を用いる．しかし，2 値情報では距離計算の候補選定能力が不十分と考えられるため，提案手法では選定に多値情報を用いる．

手法の詳細を以下で述べる．PCH 同様，提案手法でも，式 (2) で示されるハッシュ関数を用いる．まずハッシュ関数 h_i に関して投票を行う．図 2(a) は h_1 の投票処理に関するものである．任意の投票パラメータ t に対して，クエリと同じビンに入った点には $t+1$ 票の票数情報を付加する．そして，その両隣のビン（$h_1(p)=h_1(q)\pm1$ を満たすビン）に入った点には t 票の票数情報を付加する．つまり，s 個隣のビンについて考えると，クエリの s 個隣のビン（$h_1(p)=h_1(q)\pm s$ を満たすビン）には $(t-s+1)$ 票の票数情報を付加する．これを，票数情報が 1 票となる，t 個隣のビン（$h_1(p)=h_1(q)\pm t$ を満たすビン）まで行う．図 2(b) は h_2 の投票処理に関するものであり，こちらも

表 1　各手法の比較

	距離計算	射影する軸	ハッシュで用いる情報	隣接するハッシュのビンの参照
LSH	あり	斜交基底	2 値情報	しない
PCH	あり	正規直交基底	2 値情報	する
MVH1	あり	正規直交基底	多値情報	する
MVH2	なし	正規直交基底	多値情報	する

h_1 と同様に投票を行う．このような処理を全ての h_i に対して行う．

次に h_i で付加された票数情報を全て加算する．図 2(c) は，h_1, h_2 で付加された票数情報を全て加算した結果を示したものである．例えば，h_1 で t 票，h_2 で t 票の情報を付加された点はそれぞれの票数を足して，$2t$ 票の情報を保持する．これを全ての点に対して行う．その結果を基にして，投票ヒストグラムを作る．4. の実験でも示すが，この投票ヒストグラムはクエリの近傍では概ね L_1 距離を反映する．故に，この投票ヒストグラムの中で，任意の閾値 v よりも大きな点のみを距離計算の対象とすることで，距離計算のコストの削減が期待できる．このような処理により，PCH の重複度よりも距離を反映した情報を利用することができ，1 段階目の処理では同じ計算量で高い精度が得られる．

3.3　提案手法２：MVH2

上述の多値化処理を施した提案手法では近さの多値情報を付与して距離計算の対象を効率的に絞った．しかし，票数が距離を反映しているのであれば，そもそも距離計算を行う必要がない．故に，投票ヒストグラムの中で最も投票数が大きい点を近似最近傍点とすれば十分のはずである．そこで，２つ目の提案手法として，距離計算をせずに近似最近傍点を得る手法を提案する．このような処理では，精度が低下することが予想されるが，距離計算が不要になると距離計算を行うための元のデータを保持する必要がなくなる利点も存在する．そのため，精度をある程度犠牲にする代わりに，計算時間とメモリ使用量の大幅な削減が期待できる．

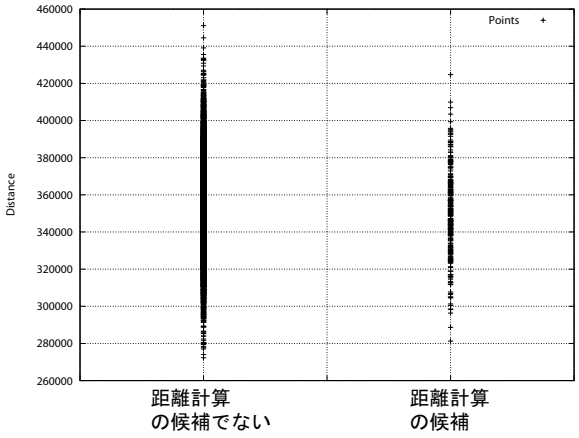
4.　実験および考察

提案手法の有効性を確認するための実験を行った．実験で用いた計算機は，CPU が Opteron8378(2.4GHz) でメモリは 128GB である．

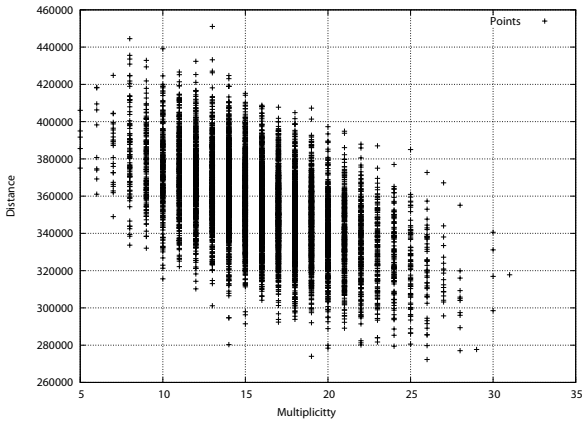
4.1　実験１：距離計算対象の選定能力

LSH と PCH と提案手法の，距離計算対象の選定結果と実際の距離の関係を確認する実験を行った．実験は 100 次元空間に一様に分布する人工データを作成して行った．各次元の要素の値が [0,10000] であるような 100 次元のデータ 1 万点を生成した．

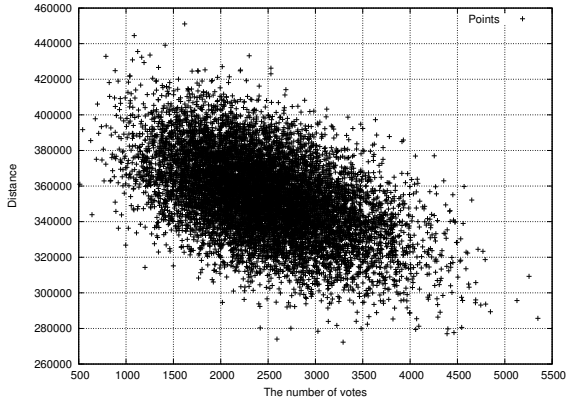
図 3 は LSH，PCH，提案手法にあるクエリを与えた時の距離計算対象の選定結果と実際の距離（L_1 距離）の関係を示している．LSH は，ハッシュの個数 $k = 95$，ハッシュ関数群数 $L = 1$，ハッシュ幅 $w = 3$ とした．PCH と提案手法は，$k = 95$，$w = 3$ とし，隣のビンの参照数（PCH では δ，提案手法では t）は 300 とした．ハッシュ関数で使用する正規直交基底 $\boldsymbol{\varphi}_i$ は



(a) LSH—距離計算候補と距離 (k=95，L=1，w=3)



(b) PCH—重複度と距離 (k=95，w=3，δ=300)



(c) 提案手法—投票数と距離 (k=95，w=3，t=300)

図 3　距離計算対象の選定能力（100 次元，1 万点のデータ）

100 個の中から 95 個をランダムに選んだ．

図 3(a) は，LSH の場合である．横軸は，距離計算の候補か距離計算の候補でないかを表し，縦軸は L_1 距離である．図 3(b),

(c) はそれぞれ PCH，提案手法にクエリを与えた時の重複度または投票数と実際の距離の関係を示している．横軸は各点に対する投票数，縦軸は各点の L_1 距離である．図 3(a) では，距離計算の候補を効率的に選定できていないことがわかる．図 3(b) と (c) より重複度や投票数がどちらも距離と負の相関を持つため，重複度や投票数を用いた距離計算対象の選定は有効と考えられる．

4.2 実験２：MVH1 の性能評価

LSH と PCH(注3) と MVH1 の性能比較の実験を行った．用いたデータは 3 種類で，

（1） 100 次元，10 万点のデータ
（2） 100 次元，100 万点のデータ
（3） 1000 次元，10 万点のデータ

である．用いるデータは 4.1 の実験と同じである．

実験で用いたパラメータは下記の範囲から選び，その組み合わせをいくつか試した．LSH は w を 10000〜500000，k を 1〜10，L を 1〜10 とした．PCH は w を 10〜10000，k を 1〜100，隣のビンの参照数 δ を 0〜5，投票数の閾値 v を 0〜1 とした．MVH1 は w を 1〜10000，隣のビンの参照数 s を 0〜1000，投票数の閾値 v を 0〜1 とし，k は 100 次元のデータの時は 1〜100，1000 次元のデータの時は 10〜1000 とした．

図 4(a)，(b)，(c) はそれぞれデータ (1)，(2)，(3) に対するもので，いずれも精度が 98%から 99%の間の時の計算時間とメモリ使用量の関係を表している．横軸は計算時間 (ms)，縦軸はメモリ使用量 (MB) とした．計算時間は 1 クエリ当たりの平均時間とした．

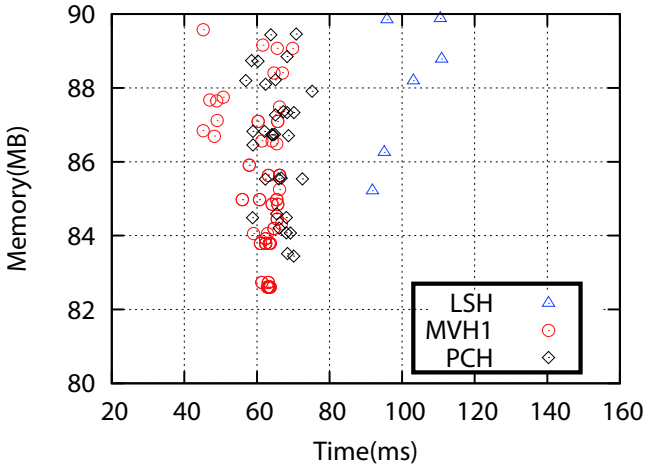
図 4 より，同一の精度で同一のメモリ使用量の場合，MVH1，PCH，LSH の順で計算時間が少ない．図 4(a) では，MVH1 のメモリ使用量は LSH や PCH に対して同程度で，計算時間は PCH に比べて約 15%，LSH に対しては約 50%少なかった．MVH1 が PCH よりも計算時間が短い理由は，両手法の違いである近さの情報の多値化よるものと考えられる．次元数や点数を大きくした場合は，MVH1 は LSH や PCH よりも概ね優れた性能を示した．また，MVH1 の計算時間は次元数や点数に対して比例関係を示していることが分かる．これらのことより，次元数や点数の変化に対しても概ね有効であると言える．

4.3 実験３：MVH2 の性能評価

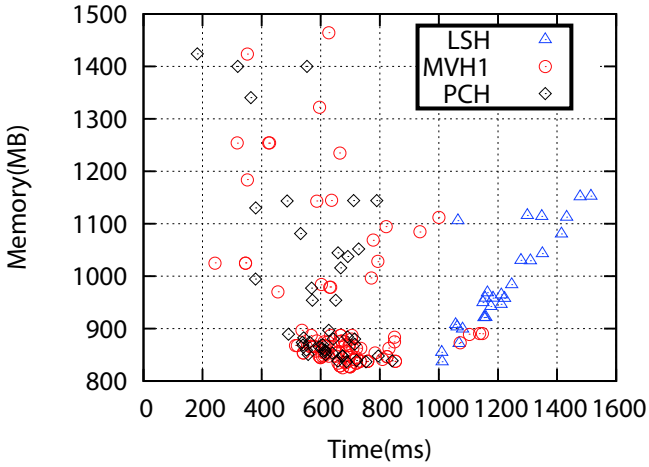
LSH と MVH1 と MVH2 と PCH の距離計算を省く手法の性能を比較する実験を行った．用いるデータ，与えるパラメータは，4.2 と同じである．

図 5(a)，(b)，(c) はそれぞれデータ (1)，(2)，(3) に対するものでいずれも精度が 90%から 99%の間の時の計算時間とメモリ使用量の関係を表している．横軸は計算時間 (ms)，縦軸はメモリ使用量 (MB) とした．

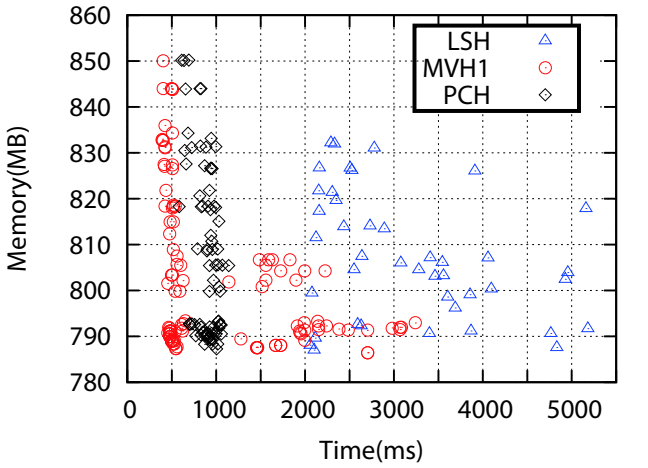
まず，MVH2 と同様に PCH でも距離計算の処理を省略する



(a) 100 次元 10 万点



(b) 100 次元 100 万点



(c) 1000 次元 10 万点

図 4 MVH1 の性能評価（精度が 98%–99%の時の計算時間とメモリ使用量の関係）

手法は最大でも 50%程度の精度しか達成できなかったため，図には掲載していない．図 5(a) では，LSH に比べて MVH2 のメモリ使用量は約 50%少なく，計算時間は約 50%少なかった．同様に，MVH1 に比べて MVH2 のメモリ使用量は約 50%少なく，計算時間は同程度もしくは若干増加であった．これは，距

(注3)：文献 [8] の PCH では重複度が上位 b%である点を距離計算の候補と定め，重複度の降順に距離計算を行う．我々が実験した限りでは，このソートを必要とする方法より，以下のソートを必要としない方法のほうが高速であった．そこで，本稿の PCH では，重複度がその最大値の v%以上の点を距離計算の候補と定め，順序にこだわらずに距離計算を行うことにした．

離計算が不要なことによる計算時間の削減，および元のデータが不要なことによってメモリ使用量が抑制されたことが主な要因と考えられる．図 5 より MVH2 は計算時間とメモリ使用量を削減できることが分かる．MVH1 より MVH2 の計算時間が増加する原因としては，距離計算の処理を省略する代わりに多くのハッシュを参照する必要があるためと考えられる．また，図 5(c) では，次元数が増加した場合，MVH2 の計算時間が他の手法に比べて短い．これは，次元数が増加することで，距離計算のコストが増加したからと考えられる．

5. 結　　び

本稿では，近似最近傍探索問題において，同一の探索精度を従来手法より小さい計算時間とメモリ使用量で実現する方法を 2 通り検討した．1 つ目の提案手法では，近さの情報を多値化することでクエリの近傍での L_1 距離の概算値を求め，距離計算の対象を効率的に選定できるようにした．2 つ目の提案手法では，クエリの近傍での L_1 距離の概算値を利用して距離計算の処理を省いた．実験より，1 つ目の提案手法では計算時間の削減ができ，2 つ目の提案手法では計算時間とメモリ使用量を削減できた．今後の課題は，以下の通りである．

- 一様分布に基づくデータ以外のデータに対して実験を行う．
- 用いるハッシュ関数のハッシュ幅をデータに応じて可変にし，性能の向上を図る．
- 別の探索精度を用いて実験を行う．
- 木構造の手法である ANN [10] など，ハッシュを用いる手法以外のものと性能比較実験を行う．

文　　献

[1] H. Jégou, M. Douze, and C. Schmid, “Packing bag-of-features,” Proc. ICCV2009, pp.2357–2364, Sept. 2009.

[2] K. Kise, K. Noguchi, and M. Iwamura, “Robust and efficient recognition of low-quality images by cascaded recognizers with massive local features,” Proc. 1st Int Workshop on Emergent Issues in Large Amount of Visual Data (WS-LAVD), pp.2125–2132, Oct. 2009.

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

[4] Y. Ke and R. Sukthankar, “PCA-SIFT: A more distinctive representation for local image descriptors,” Proc. CVPR2004, pp.506–513, 2004.

[5] 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.

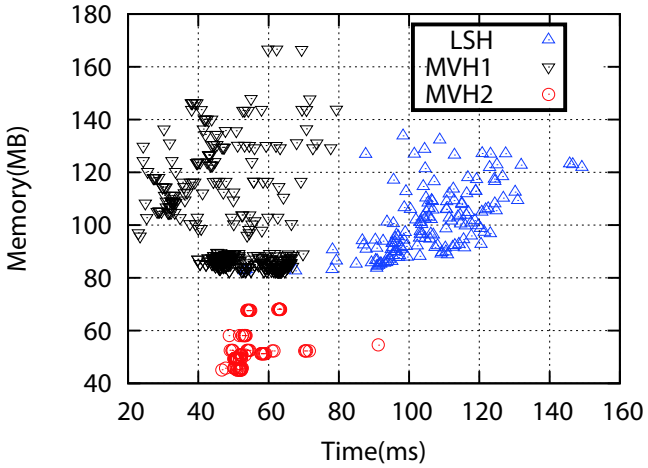
[6] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” Proc. 20th annual symposium on Computational geometry, pp.253–262, 2004.

[7] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06), pp.459–468, Oct. 2006.

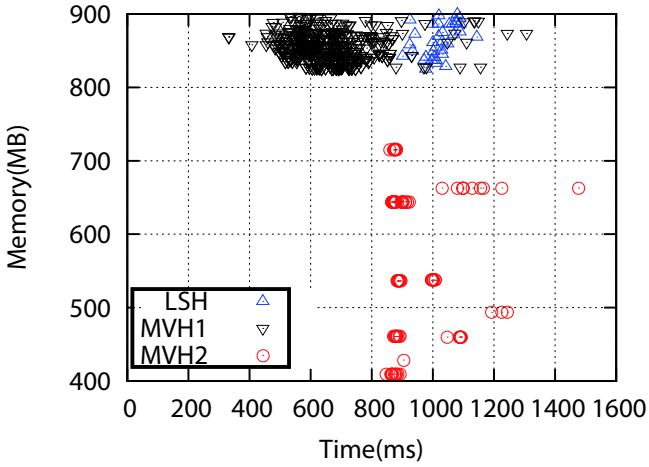
[8] 松下裕輔，和田俊和，“Principal Component Hashing —等確率バケット分割による近似最近傍探索法—,” 画像の認識・理解シンポジウム (MIRU2007) 論文集，pp.127–134，July 2007.

[9] 松下裕輔，和田俊和，“一般分布に対する Principal Component Hashing,” 信学技報，PRMU2007-290，March 2008.

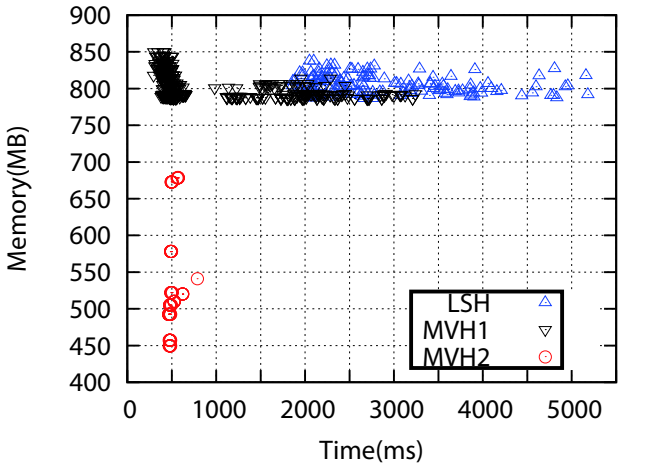
[10] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, “An optimal algorithm for approximate nearest neighbor searching in fixed dimensions,” Journal of the ACM, vol.45, no.6, pp.891–923, Nov. 1998.



(a) 100 次元 10 万点



(b) 100 次元 100 万点



(c) 1000 次元 10 万点

図 5　MVH2 の性能評価（精度が 90%-99%の時の計算時間とメモリ使用量の関係）