DEIM Forum 2013 E10-3

BoF の類似度に適合する改良版 LSH を用いた 高速類似画像検索

清水　大輝†　　寺沢　憲吾††

† 公立はこだて未来大学大学院システム情報科学研究科　〒 041–8655 北海道函館市亀田中野町 116–2
†† 公立はこだて未来大学システム情報科学部　〒 041–8655 北海道函館市亀田中野町 116–2
E-mail: †{g2111016,kterasaw}@fun.ac.jp

あらまし　本研究では大規模データベースからの類似画像検索の高速化を目指す．今回我々は画像工学の分野で広く使われている Bag of Features 特徴量（BoF）と intersection 類似度によるマッチングを Locality Sensitive Hashing（LSH）によって高速化することを考えた．LSH は特徴空間内の近傍点をハッシュ値の衝突により絞り込むことで，クエリ点との距離計算の対象となるデータセットの特徴点の数を削減する．現在 intersection に対応する LSH が存在しないことから，これに対応するハッシュ関数群を新たに考案した．考案した関数群は類似度が高いほどハッシュ値の衝突率が高いという LSH の必要条件を満たしており，これを用いることで設定精度を満たす検索対象の絞り込みを実現した．

キーワード　類似画像検索，Locality Sensitive Hashing，Bag of Features，intersection

Hiroki SHIMIZU† and Kengo TERASAWA††

† Future University Hakodate, Graduate School of Systems Information Science
116–2 Hakodate, Kamedanakano, Hokkaido, 041–8655 Japan
†† Future University Hakodate, School of Systems Information Science
116–2 Hakodate, Kamedanakano, Hokkaido, 041–8655 Japan
E-mail: †{g2111016,kterasaw}@fun.ac.jp

Key words　Content Based Image Retrieval，Locality Sensitive Hashing，Bag of Features，intersection.

1. 序　　論

本稿では類似画像検索に，より適した LSH の可能性について論じる．LSH アルゴリズムは近似最近傍探索問題を解くアルゴリズムである [1]〜[4]．これまで LSH アルゴリズムはランダムに作成されたデータ点の集合に対してその有効性は確認されており，さらに我々の研究 [5] によってそれが実データに対しても有効であることが分かった．ここでランダムに作成されたデータ集合と実データの集合には，特徴空間内での分布が均一であるか，疎密があるかという違いがある．ランダムに作成されたデータに対しては，Terasawa ら [6] の研究によって LSH アルゴリズムの有効性が示されていた．またこの論文はデータの特徴に着目し，そのデータの特性に合わせてハッシュ関数を設計することで，LSH の性能が，より向上することを示している．

近年，HDD などの記録媒体の大容量低価格化や，ディジタルカメラの普及等によって大量の画像データが溢れている．画像の検索を行う場合，Google [7] などのようにキーワードによる検索が一般的であるが，近年は画像をクエリとして類似画像を検索するサービスも台頭してきている [7]〜[9]．これらの画像をクエリとして類似の画像を検索するサービスは，キーワードによる画像検索よりもユーザの望んだ結果を出す場合がある．しかしながら，これらのサービスはキーワードによる検索に比べ検索に時間がかかるというデメリットがある．

先ほど挙げた [7], [8] といったサービスなどに用いられている，画像自体の特徴を用いた検索手法を Content-Based Image Retrieval (CBIR) という．CBIR では一般的にキーワードによる検索よりも大きな計算コストがかかるため，多くの研究者が様々な手法でその計算コストを減らす研究をしている．例えば Csurka ら [10] の研究では，CBIR で使われる SIFT 特徴量をもとにした BoF と呼ばれる特徴量を提案している．SIFT 特徴量 [11], [12] とは，画像中から複数点の特徴点を検出し，回転やリサイズにロバストな 128 次元の特徴量ベクトルを各特徴点ごとに算出するものである．これにより，画像間の類似度を，クエリ画像の各特徴点と対象の画像中に近傍である点の組がいくつあるかで表現する事ができる．SIFT 特徴量は類似画像の

検索に有効ではあるが，一枚の画像中に特徴点が数百から数千点表れるため，計算コストが高い．そこで Csurka らは大量の SIFT 特徴量をクラスタリングし，任意数の visual words を算出しヒストグラムビンとした後，各画像内のすべての特徴量を投票してヒストグラムを作成した．これをベクトルとみなすことにより，各画像を任意次元のベクトル一つで表現することで計算コストを減らした．

本研究ではすでに述べた BoF 特徴量による画像マッチングを LSH によって低コストで行うことを主眼としている．

2. 要素技術

2.1 BoF

BoF とは Csurka らの提案した特徴量計算アルゴリズムであり，Lowe らによって提案された SIFT 特徴量を基にしている．SIFT 特徴量とは，各画像の中から複数の特徴点を抽出し，その各点ごとに特徴量を算出するものである．この特徴量は向きと大きさを持つため画像の回転や大きさの変化に強い．加えて画像の類似度を，特徴点の特徴量ベクトル同士の距離計算で求めるため，オクルージョンなどにも頑健である．

SIFT は 128 次元の特徴量を持つ特徴点が画像ごとに数百から数千点出ることもある．SIFT 特徴量によるマッチングでは図 1 のように，すべての特徴点がデータセット内のどの画像のどの特徴点と最も距離が近いかという計算を行い，その得票数を類似度とする．そのため計算コストが大きくなりがちであり，加えて画像ごとに特徴点の数が不定であるため計算コストの予測が立てにくい．

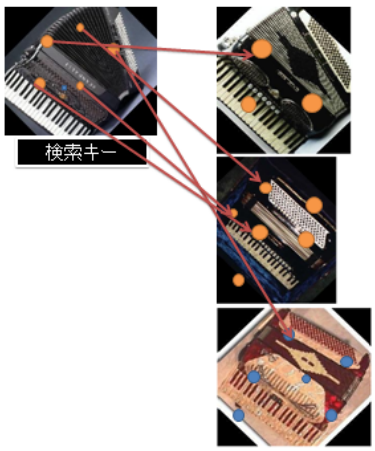

図 1　SIFT による類似画像検索

そこで，BoF では画像ごとの SIFT 特徴量をまとめ，画像ごとに一意の特徴量ベクトルを一つ作成することでこの問題を解決した．手順としてはまず，大量の SIFT 特徴量を c-means 法によってクラスタリングすることで，c 個の中心ベクトルを求める．この中心ベクトルを visual word と呼ぶ．visual words をヒストグラムビンとして，最も距離の近い visual word のビンに画像中の各特徴量を投票し，ヒストグラムを作成する．このヒストグラムを特徴量とみなす．これによって得られる BoF 特徴量と SIFT 特徴量との違いは，まずひとつに画像の特徴量の数である．画像中に数百から数千の特徴量を含む SIFT と違い，BoF では特徴量を各画像に一つに減らすことができる．また，特徴量の次元数も 128 次元の SIFT に比べ，BoF では c-means の c の値が特徴量の次元数になるため，自ら決定できるという違いがある．BoF と SIFT のマッチングによる精度については [13] で述べられているように，BoF の次元数自体が精度に関わってくる．近年では [14], [15] のように，ヒストグラムを投票数で正規化し，intersection によるマッチングや分類を行う研究も多い．

2.2 LSH

最近傍探索問題を解く手法で最も単純なアルゴリズムは，クエリ点と対象のすべての点との距離を計算する brute-force 法である．しかし，この手法はデータセット内の点の数や特徴量の次元数が増えるほど線形的にコストが増加してしまう問題がある．

この問題を解決する一つの手法として LSH がある．LSH は $(R, 1–\delta)$ 近傍探索問題を解く乱択アルゴリズムである．$(R, 1–\delta)$ 近傍探索問題とは，設定された許容誤り率 δ に対して，おおむね $1–\delta$ 以上の確率でクエリ点から距離 R 以内の点を返すアルゴリズムである．LSH の計算コストは brute-force 法に比べ少なくすむ．しかし，LSH におけるコストの少なさというのは精度とトレードオフの関係である．δ の値が大きいほど高速になるが近傍点を正しく出力する確率は低くなり，小さいほど多くの正しい点を返すが計算コストは brute force 法と変わらなくなっていく．

LSH の基本的なアイデアは，あらかじめすべての点をハッシングしハッシュバケットに格納するということである．こうすることでクエリとの近傍点を探す際には，クエリとすべての点との距離を計算せず，クエリをハッシングし同一のバケットに含まれる各点との距離を計算することで近傍点を見つけることができる．

ここで LSH におけるハッシングについて説明する．ハッシュ関数 h は局所鋭敏ハッシュ関数群と呼ばれる $\mathcal{H}$ からランダムに選択される．また，LSH を効果的にするためには，ハッシュ関数が以下を満たす必要がある．

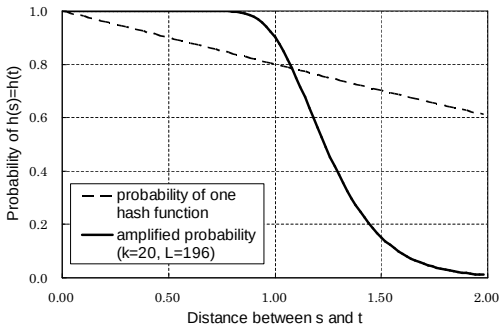

図 2　$h(s)=h(t)$ となる率と距離 $\mathrm{dist}(s,t)$ の関係

同一特徴空間に含まれる二点 $s, t \in P$ について $\mathrm{dist}(s,t) \geqq R$ の時．すなわち点 s と t が距離 R 以上離れている時，二つ

の点に対するハッシュ値 $h(s)$，$h(t)$ について，図 2 の破線のように $\Pr_{\mathcal{H}}[h(s)=h(t)]$ が小さくなる．この性質を満たすハッシュ関数群から複数回ハッシュ関数を選ぶことで，実際に距離とハッシュ値が衝突する確率の関係が図 2 の実線のようになるようにパラメータを設定する．

まず，各特徴点を K 個のハッシュ関数でハッシングすることで，各特徴量毎に K 次元のハッシュ特徴量を作る．この K 次元のハッシュ特徴量が一致した特徴点を検索対象とする．この時，K の値が大きくなるほどハッシュ特徴量が一致する確率は下がっていく．そこで，この動作を L 回繰り返し，一度でもハッシュ特徴量が一致したものを検索対象として追加していくことで，近傍の点が取りこぼれることのないようにする．これにより，近傍点が検索対象となる確率は，少なくとも

$$1-(1-P_1^K)^L \tag{1}$$

になる．この二つのパラメータ K, L について，K の値は任意で決めてよいが，L の値は K の値と閾値 R でのハッシュ値の衝突率 P_1 から以下の式で求める．すなわち，

$$L \geqq \frac{\log \delta}{\log\left(1-P_1^K\right)} \tag{2}$$

を満たす最小の整数を L とする．

これらのことから各特徴点ごとに KL 個のハッシュ値を計算する必要があることがわかる．よって，

$$(\text{ハッシュのコスト} = \text{クエリハッシングのコスト}) \propto \mathrm{KL}$$

となる．このコストはパラメータの値が大きくなるほど大きくなる．そのかわり，検索対象の数は減っていくので，

$$\text{全体のコスト} = \text{クエリハッシングのコスト} + \text{検索コスト}$$

が最も小さくなるようなパラメータを選出する必要がある．

画像特徴量などは，ランダムに作成されたデータ集合に比べ特徴空間内に疎密が存在する．我々の研究 [5] によって画像特徴量に対しても LSH アルゴリズムは有効であることが示されている．

LSH で利用されるハッシュ関数群は以下の二つの性質を満たす必要がある．

- ランダム性
- 局所鋭敏性

LSH は先に述べたように近傍点を検索対象として追加しつつ，距離 R 以内の点のうち，$1-\delta$ 以上の確率で点を見つけることが保証されなければならない．そのために，ハッシュ関数のパラメータをランダムに変化させながら複数回ハッシュする必要がある．十分なランダム性を確保するため，ハッシュ関数群には無限個または実用上無限個とみなせる程度の数のハッシュ関数が含まれている必要がある．

局所鋭敏性とは，近傍の点同士と，ある程度離れた点同士ではハッシュ値の衝突率に差があることである．近傍の点に対しては高い衝突率を示し，距離が離れる程に衝突率は下がっていかなければならない．

例えば，パラメータとして格子の基底ベクトルを持ち，同一の格子内の点に同一のハッシュ値を与えるハッシュ関数群を考える．図 3(a) のように格子を設定した場合，図中星印で示すクエリ点との最近傍点が同じ格子に入らないこともある．そこで，別の基底ベクトルを持つハッシュ関数に変更し，図 3(b) のように追加でハッシュし，クエリと同一の格子に含まれた点を計算対象として追加していく．この動作を繰り返すことで近傍点を極力取り逃がすことなく，非近傍点の多くを検索対象から除外する事が可能である．このハッシュ関数群は基底ベクトルがパラメータであるので，無限個のハッシュ関数を持つ．また格子で空間を分割するため，近傍点ほど同一のバケットに入りやすい．

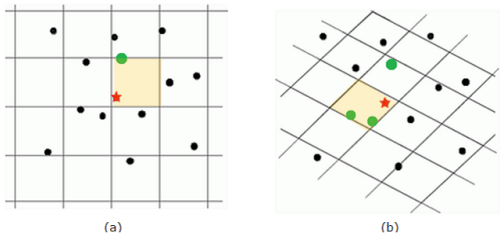

図 3　LSH によるハッシング例

2.3　LSH の種類について

LSH には特徴量をハッシュするために使われるハッシュ関数の種類によって様々なものがある．Gionis ら [1] は d-次元のランダムな単位ベクトルをハッシュ関数として利用し，ハミング距離で利用可能な LSH を考案した．

Datar ら [2] は E2LSH と呼ばれる，ユークリッド空間に拡張したハッシュを考案した．これは空間を格子で分割し，同じ格子に含まれた点に同じハッシュ値を返す．

Andoni ら [3] は R^d 空間にランダムに単位球を配置することで 空間を分割する、ball-partitioning ハッシュ関数を考案した。

また，Terasawa ら [6] は Spherical LSH (SLSH) を提案した．これは単位超球に内接する多胞体をランダムに回転させ，単位超球表面のベクトルをその最近傍の多胞体の頂点に帰属させることで，単位超球の表面を分割する．

SLSH は大きさが 1 に正規化された特徴量にしか使えないが，計算コストの点で一般的なパッケージである E2LSH よりも低コストである．

3.　提案する LSH

LSH にはハッシュの方法によって様々な種類があることは先に述べた．また，Terasawa らのようにデータの特徴に着目し，そのデータの特性に合わせてハッシュ関数を設計することで，既存のものよりもより限定された条件下で有効な LSH を設計することが可能であることもわかった．

今回我々は intersection に対して使える LSH を考案することにした．なぜなら，既存の LSH パッケージは L_2 ノルムやハミング距離を対象としたもので intersection に使えるものはないからである．しかしながら類似画像検索において，多くの

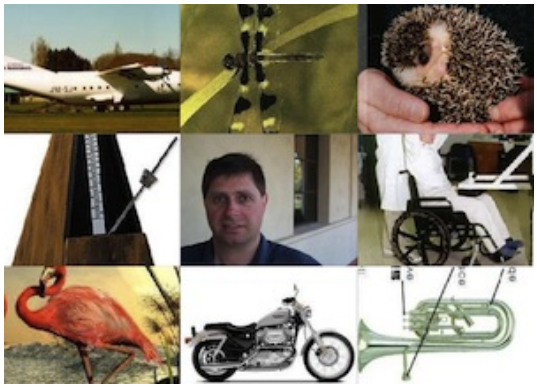

図 5　caltech101（一部）

研究が intersection によって類似度を調べており，加えて, [16] のように大量の画像を扱う研究が増えてきている．intersection による類似度検索を高速に行えるようにすることに意義があると考えた．

そこで我々は，c-means 法を利用したハッシュを考案した．図 4(a) のように空間上に特徴量ベクトルが分布している時を考える．現在，この空間には図のように点が存在している．そこから図 4(b) 中で四角で示す点のように，ランダムに複数の点を抽出する．その特徴量に対して c-means 法を行い，典型ベクトル（図 4(c) の画像星型で示す点）を c 個作成する．この典型ベクトルに $1, 2, \ldots\ldots, c$ と番号をつけ，各画像特徴量に最も近い典型ベクトルの番号をその画像特徴量のハッシュ値とする．図 4(d) では，例として c を 5 として空間を分割した．

本手法によりデータセット内の類似画像を intersection 類似度を用いてハッシングすることが可能になる．また，これにより検索対象の点の数を全検索よりも絞込みながら類似画像をとりこぼさないということが大きな狙いである．

4. 実　　験

本研究ではこのハッシュ関数の性能を確認するために，まず 2 つの実験を行った．一つはこのハッシュ関数によってハッシングされた点同士の距離と衝突率の確認のための実験である．考案したハッシュ関数が実際に近傍点同士では衝突率が高く，距離が離れるほど衝突率が低いのかを，データベース内の全画像対のハッシュ衝突率 P を求める実験によって確認した．

二つ目は提案したハッシュ関数が実際に有用かどうかを評価する前段階として，BoF を intersection 類似度によってクラス判定する際に必要とされる閾値を求める実験である．データベース内の全画像対の類似度とハッシュ値から，ハッシュ値が衝突する類似度の分布と，衝突しない類似度の分布を求める．その結果から，LSH によって検索対象として拾うべき画像の閾値を設定する．

4.1　実験 1：衝突率の確認

実験 1 では実際に実データに対してハッシュすることで，提案するハッシュ関数の衝突率を調べる．これにより，提案するハッシュ関数が局所鋭敏性を持っているかを確認する．

表 1　実験データ

実験画像セット	caltech101
特徴量	BoF
特徴量の次元数	10〜500
ハッシュ数（c)	10〜50

4.1.1　データチューニング

本実験で利用したデータを 表 1 に示す．画像セットとして caltech101（図 5） に含まれる 101 クラス 8677 枚の画像を用意した．まずすべての画像の SIFT 特徴量を計算する．その後各クラスから 5 枚ずつ選択し，SIFT 特徴量のクラスタリングを行い visual words を 10〜500 点算出する．各画像中の SIFT 特徴量を visual words に基づいて量子化し，各画像毎に visual words 次元の BoF 特徴量を計算する．こうすることで 8677 点の BoF 特徴量を用意した．

4.1.2　ハッシング

前項で用意した特徴量から任意の数だけランダムに点をサンプル点として選択する．これらの点を c-means 法でクラスタリングし，重心ベクトルを c 点算出し，ナンバリングする．各画像の特徴量に対して最も近いハッシュ点と同じ番号を各画像にハッシュ値として割り当てる．その後すべての画像の組について，その画像同士の intersection による類似度を計算し，ハッシュ値が一致したか不一致だったかを記録する．そうすることで，合計およそ 3700 万対の類似度と，ハッシュ値の組み合わせが得られる．各類似度ごとに衝突した対の数と，非衝突であった対の数を数え上げることで，各類似度での衝突率が算出できる．

特徴量の次元数，重心ベクトルの数を変えながら同じ実験を複数回行った．結果を図 6 に示す．図中，横軸が画像同士の類似度で，縦軸がハッシュの衝突率である．図から読み取れるように提案したハッシュは類似度が高いほど衝突率が高く，LSH に要求される「画像間の類似度が高いほど衝突率が高い」という性質を満たしていることが確認できる．

4.2　実験 2：閾値の設定

実験 2 では本研究での検索対象として拾うべき画像群の類似度を設定する．ここで述べる拾うべき画像群の類似度とは LSH における R のことである．つまり，その類似度以上の画像の多くが類似画像として拾うべき画像であり，その類似度以下の画像の多くが非類似画像であるような類似度の閾値を調べる．

4.2.1　データチューニング

本実験以降扱う BoF の次元数は 500 とした．理由は [13] 中などの文献に，次元数が高いほうが一般的にクラスマッチングの精度が高いという報告があったため，今回用意した異なった次元数の BoF の中で，最も次元数が高いものを利用することにした．

今回の実験では，データベース内の全画像をクエリとし検索対象もデータベース内の全画像とした．全画像対でこの実験を行ったので対数はおよそ 3700 万である．各画像の組について，類似度とハッシュ値を調べた．類似度を 0.1 刻みに階級化し，各階級についてハッシュ値が衝突した数と，ハッシュ値が非衝

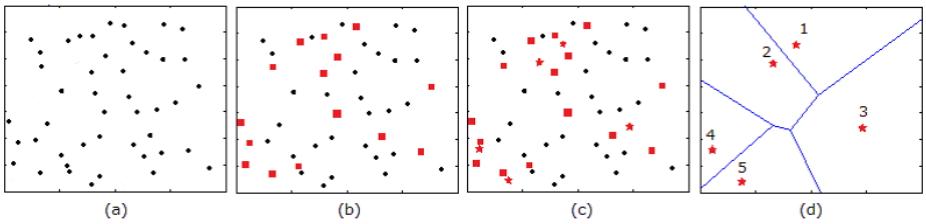

図 4　提案するハッシュ関数による特徴空間ハッシング手順

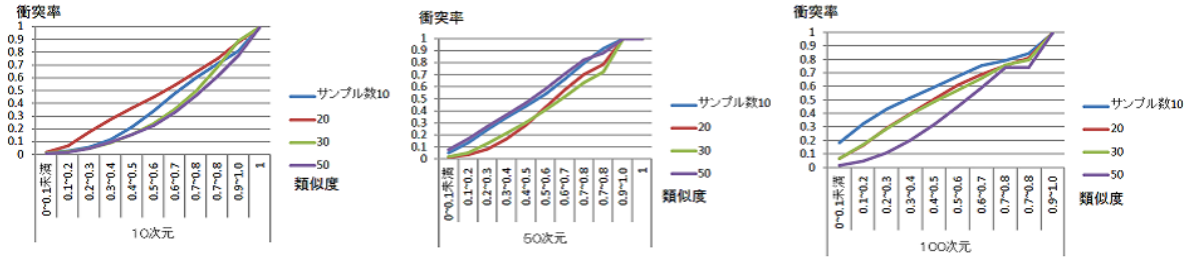

図 6　提案ハッシュによる画像間の類似度と衝突率の関係

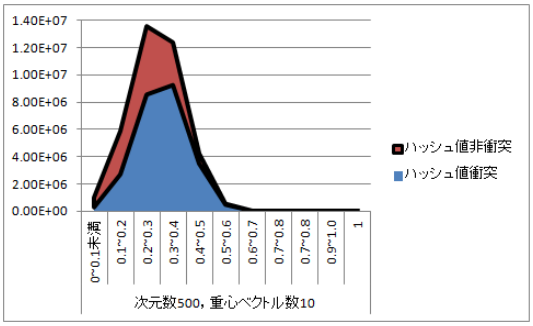

図 7　BoF 次元数 500，重心ベクトル 10 個のハッシュ衝突数

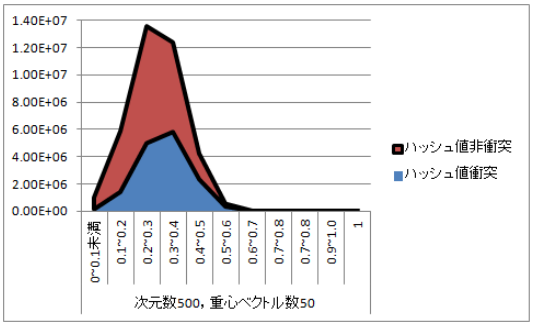

図 8　BoF 次元数 500，重心ベクトル 50 個のハッシュ衝突数

突だった数を記録した結果は図 7, 8 となっている．また，衝突率は図 9 となっている．これらからわかることは，画像間の intersection による類似度が高いほど提案ハッシュの衝突率は高く，類似度が低い時ほど衝突率も低いということである．加えて，重心ベクトルの数が増えると全体にハッシュの衝突率が下がるということである．本稿では図を省略するが，他の次元数の BoF 特徴量においても同様の傾向が見られた．

今回使用する 500 次元の特徴量は図 7, 8 のように類似度が 0.3〜0.4 の間にハッシュ衝突のピークがあり，ハッシュ非衝突のピークは類似度 0.2〜0.3 に来ることがわかった．これらより，R_1 を 0.4，R_2 を 0.3 と設定した．これは，$R_1 = 0.4$ 以上

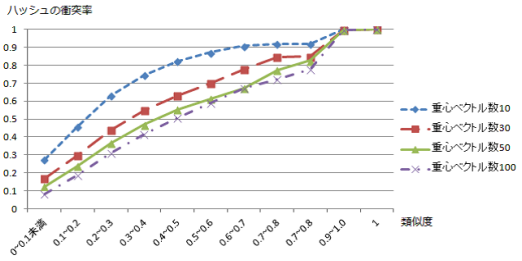

図 9　BoF 次元数 500，各重心ベクトル数でのハッシュ衝突率

の類似度の画像を取りこぼさなければ，クエリに対して多くの類似画像を計算対象として残せることを意味し，$R_2 = 0.3$ 以下の類似度の画像を計算対象から除外することが出来れば，多くの非類似画像を計算対象から除外できることを意味する．

よって，本研究では

- ハッシュ関数：データセットからランダムに選ばれた点から c-means 法によって求められた複数の重心ベクトルのうち，最も近いベクトルのラベルを返す関数
- BoF 特徴量：caltech101 から各画像ごとに作成された 500 次元の特徴量
- 対象とする類似度：intersection
- 閾値 R_1：クエリ画像との類似度が 0.4
- 閾値 R_2：クエリ画像との類似度が 0.3

である LSH を提案し，その性能を評価した．

5. 評 価 実 験

LSH の性能は，5.1 節で述べる指標 ρ で記述することができる．本研究では Leech Latice [17] と提案手法の ρ を比較する．その後，実際に提案手法による LSH を実装することにより，パラメータと計算コストを実測することでその性能を評価する．

本来この ρ の評価は，対象とするデータを絞り込むための

表 2　500 次元 BoF における重心ベクトル数と ρ の関係

重心ベクトルの数	ρ
10	$0.60948 \leqq \rho \leqq 0.64394$
30	$0.70486 \leqq \rho \leqq 0.72802$
50	$0.74676 \leqq \rho \leqq 0.76603$
100	$0.74045 \leqq \rho \leqq 0.75789$
Leech Latice	0.656454

ハッシュ関数群の距離尺度が同一である必要がある．よって，今回提案したハッシュは，本来であれば同じように intersection によるハッシュを行う LSH と性能を比較するべきであるが，本研究以前に intersection によるハッシュを行う LSH は存在しないため，今回はあくまで目安の一つとして，他の LSH の比較対象として多く用いられる Leech Latice との比較を行った．

5.1　ρ の 評 価

LSH における評価関数 ρ は P_1 を閾値 R_1 でのハッシュ衝突率，P_2 を閾値 R_2 でのハッシュ衝突率とした時，以下の数式で表される．

$$\rho = \frac{\log \frac{1}{P_1}}{\log \frac{1}{P_2}} \tag{3}$$

LSH の計算量は $O(n^\rho)$ で規定されるため、ρ の値は小さいほど性能が優れていることを示す。

提案手法と Leech Latice での ρ の値は表 2 のようになった．なお，今回我々は $R_1 = 0.4$, $R_2 = 0.3$ としたので，[17] より，Leech Latice について $R_1 = 0.9$, $R_2 = 1.2$ として R_1と R_2の比 が同一になるようにし算出した．この数値は，各重心ベクトル数でデータセットの画像を実際にハッシングし，その距離での画像の組でのハッシュ衝突率を算出するという試行を 400 通りのハッシュで行い，その平均をハッシュの衝突率 P_1 と P_2 として算出した．

本来この ρ の評価は，対象とするデータを絞り込むためのハッシュ関数群の距離尺度が同一である必要がある．しかし，提案手法は intersection で類似度を計るハッシュ関数群であり，Leech Latice は L_2 ノルムでのハッシュ関数群である．よって，本実験によって重心ベクトルの数が 10 個であるときの提案手法が最も低い ρ を示しているが，これだけから単純に提案手法が Leech Latice よりも優れていると断定できるわけではない．ではあるが、ある程度の目安として、提案手法はおおむね Leech Latice と比較できる水準の性能を示しているとは言うことができる．

5.2　計算コストの評価

画像間の距離計算には，CPU の性能などに応じて一定のコストが掛かる．そのため，検索対象の数に比例してコストが増えていく．

クエリ一つあたりの計算コストは，brute-force 法においては単純に（データセットに含まれる特徴点の数）×（特徴点対 1 つあたりの類似度計算コスト）で記述できる．一方 LSH においては距離計算の前にハッシュ関数の計算が必要なため，総コストは（クエリハッシングの計算コスト）＋（絞り込まれた特徴点の数）×（特徴点対 1 つあたりの類似度計算コスト）となる．さらに今回提案したハッシュ関数におけるクエリハッシングのコストは（重心ベクトル数× K × L）×（特徴点対 1 つあたりの類似度計算コスト）であるため、結局今回提案する LSH の総コストは（重心ベクトル数× K × L ＋絞り込まれた特徴点の数）×（特徴点対 1 つあたりの類似度計算コスト）となる．

図 10　クエリの一部

表 3　パラメータ

重心ベクトル数	K	L
10	2	2
	4	4
	6	6
30	2	5
	4	15
	6	42
50	2	9
	4	41
	6	177
100	2	13
	4	75
	6	434

この計算コストを調べるため，まずパラメータ K 毎の L を算出し，実際に提案ハッシュによる LSH を実装しハッシュ衝突した特徴点の数を算出することで，計算コストを brute-force 法と比較した．

実験はこれまで同様に画像セットとして caltech101 に含まれる 101 クラス 8677 枚の画像を用意し BoF 特徴量を算出．8677 点の BoF 特徴量を用意し，クエリ（図 10）として accordion, airplanes, anchor, ant, barrel, bass, beaver, binocular, bonsai, blain クラスから 1 枚ずつ計 10 点を選出した．

重心ベクトルの数は $10, 30, 50, 100$ とし，各重心ベクトルごとのパラメータを表 3 に記す．

便宜上以下では，クエリ accordion を q1，クエリ airplanes を q2，$\cdots$ クエリ blain を q10 と呼ぶ．また，許容誤り率は $\delta = 0.1$ とした．各クエリとパラメータ K，検索対象数の関係を各重心ベクトル数毎に表 4, 5, 6, 7 に記す．

表 4　重心ベクトル数 10 の時の検索対象数の推移

	検索対象数 (K = 2)	検索対象数 (K = 4)	検索対象数 (K = 6)
q1	6649	5757	4985
q2	6649	5797	6492
q3	457	105	60
q4	7669	6539	4716
q5	7669	5859	6145
q6	4239	4098	5660
q7	3655	6342	6076
q8	6649	6461	4158
q9	6521	4535	5965
q10	6649	4326	4937

表 5　重心ベクトル数 30 の時の検索対象数の推移

	検索対象数 (K = 2)	検索対象数 (K = 4)	検索対象数 (K = 6)
q1	3371	3396	1996
q2	7569	7439	6374
q3	1006	2133	437
q4	7442	5752	4233
q5	7097	6325	6398
q6	6893	5411	5582
q7	7382	7078	5958
q8	7446	6922	5117
q9	7544	6443	6039
q10	7488	5881	5636

表 6　重心ベクトル数 50 の時の検索対象数の推移

	検索対象数 (K = 2)	検索対象数 (K = 4)	検索対象数 (K = 6)
q1	4498	4016	3187
q2	6847	7066	5774
q3	3990	1431	315
q4	4292	3950	4076
q5	6840	6509	6138
q6	6161	5459	3586
q7	6319	6508	5208
q8	4901	4415	3476
q9	5840	6977	5009
q10	5058	5227	4118

また，クエリとの類似度が閾値 R 以上の点の数を正解数，クエリとの類似度が閾値 R 以上の点で検索対象となった点の数を正答数とする．各重心ベクトル数とパラメータの関係，$K = 6$ の時の正答率（小数点第二位四捨五入）を表 8, 9, 10, 11 に記す.

上記より，提案ハッシュは概ねパラメータの増加にしたがって検索対象数が減少傾向にあり，かつ表 8, 9, 10, 11 の正答率のカラム（最右）より，閾値以上の類似画像を $1 - \delta$ 以上で検索対象として見つけている．本稿で行った intersection と LSH の関係を調べるいくつかの実験によって，既存の LSH では対応しきれなかった intersection に対して有効なハッシュを構築できる可能性を示した．多くの研究で用いられる intersection

表 7　重心ベクトル数 100 の時の検索対象数の推移

	検索対象数 (K = 2)	検索対象数 (K = 4)	検索対象数 (K = 6)
q1	5035	4176	3100
q2	7175	6758	6170
q3	3550	399	216
q4	7021	4877	4782
q5	6868	5943	5705
q6	5144	3978	3928
q7	6292	5932	5402
q8	5975	5937	4702
q9	5989	5664	5472
q10	4847	4963	5038

表 8　重心ベクトル数 10 での正答数

	正解数	正答数：K = 2	K = 4	K = 6	正答率%（K=6）
q1	52	41	48	46	88.4
q2	1098	1077	1014	1084	98.7
q3	3	2	1	1	33.3
q4	405	389	389	323	79.7
q5	571	557	540	565	98.9
q6	1616	969	1359	1473	87.5
q7	3720	2192	3256	3508	94.3
q8	0	0	0	0	/
q9	1266	1189	889	1064	84.0
q10	2885	2663	2077	2360	81.8

表 9　重心ベクトル数 30 での正答数

	正解数	正答数：K = 2	K = 4	K = 6	正答率%（K=6）
q1	52	39	45	41	78.8
q2	1098	1082	1096	1088	99.0
q3	3	3	3	2	66.7
q4	405	389	371	371	91.6
q5	571	565	567	569	99.6
q6	1616	1576	1562	1569	97.1
q7	3720	3529	3641	3467	93.2
q8	0	0	0	0	/
q9	1266	1254	1212	1195	94.3
q10	2885	2838	2530	2575	89.3

による検索の高速化の可能性を示せた点で本研究の意義がある．

一方で、提案手法の総コストを規定する（重心ベクトル数 $\times K \times L$ ＋ハッシュ衝突した特徴点の数）という観点で見ると、パラメータの選び方によっては brute-force よりもコストが大きくなるケースも発生してしまった。ハッシュ関数をよりうまく設計すれば、K を増加させることによりハッシュ衝突する特徴点の数をより絞り込めるはずであるという点も含め、ハッシュ関数の設計に関してはまだ改良の余地が大きいと考えている。

6. 結　　言

本研究では類似画像検索の高速化を目指し，BoF と LSH を組み合わせることでより高速な検索を行うことを目的とした．BoF 特徴量については intersection によって類似画像や同一カ

表 10　重心ベクトル数 50 での正答数

	正解数	正答数：K = 2	K = 4	K = 6	正答率%（K=6）
q1	52	42	50	46	88.5
q2	1098	1070	1079	1059	96.4
q3	3	3	3	2	66.7
q4	405	407	346	384	94.8
q5	571	570	567	564	98.8
q6	1616	1592	1546	1415	87.6
q7	3720	3546	3588	3286	88.3
q8	0	0	0	0	/
q9	1266	1181	1236	963	76.1
q10	2885	1937	2442	2003	69.4

表 11　重心ベクトル数 100 での正答数

	正解数	正答数：K = 2	K = 4	K = 6	正答率%（K=6）
q1	52	49	48	44	84.6
q2	1098	1050	1083	1069	97.4
q3	3	3	3	3	100
q4	405	395	375	375	92.6
q5	571	569	561	567	99.3
q6	1616	1487	1427	1410	87.3
q7	3720	3293	3464	3302	88.8
q8	0	0	0	0	/
q9	1266	1015	1136	1054	83.3
q10	2885	2105	2311	2452	85.0

テゴリの画像を探し出すという研究が多い．また BoF に限らずとも，CBIR による類似画像検索では intersection による類似度を計算することがしばしばある．そこで我々は intersection による類似画像の検索を LSH アルゴリズムによって高速化することを目指し，intersection に対応するハッシュを考案した．具体的には，まずデータセット内の画像からランダムに複数画像分の BoF 特徴量を選出し，c-menas 法によって重心ベクトルを c 個求め，それらに番号を割り振る．その後，データセット内のすべての点についてもっとも近傍な重心ベクトルの番号をハッシュ値とするハッシュを提案した．実験によってこのハッシュの性能を評価した．今回我々が提案したハッシュは結果として，LSH に求められる局所鋭敏性を持ち，近傍点を取りこぼすことなく距離計算対象の点を絞り込む事ができた．また，intersection 類似度を対象に LSH を適用可能にした点も，本研究の意義の 1 つである．

今後の課題としては，さらに計算コストを減らすハッシュ関数を考案する必要があると考える．今回我々の提案したハッシュには計算コスト削減の余地がまだある．例えば，パラメータを大きくした際に非近傍の画像を検索対象からより多く減らせるハッシュ関数などを考案することで，より計算対象を絞込み，更に高速に類似画像検索ができるようになると考えている．

文　献

[1] A. Gionis, P. Indyk, and R. Motwani, “Similarity Search in High Dimensions via Hashing”, Proc. 25th International Conference on Very Large Data Bases, VLDB1999, pp.518–529, 1999.

[2] M. Datar, P. Indyk, N. Immorlica, and V. Mirrokni, “Locality-Sensitive Hashing Scheme Based on p-Stable Distributions”, Proc. Symposium on Computational Geometry 2004, pp.253–262, 2004.

[3] 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, 2006.

[4] A. Andoni and P. Indyk, “Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions”,
Commun. ACM, vol.51, no.1, pp.117–122, 2008.

[5] Hiroki Shimizu and Kengo Terasawa, “Effectiveness of image searching with LSH algorithms”, ,International Workshop on Advanced Image Technology Janu. 2012.

[6] Kengo Terasawa and Yuzuru Tanaka, “Approximate Nearest Neighbor Search for a Dataset of Normalized Vectors”, IEICE Transactions on Information and Systems, vol.E92-D, No.9, pp. 1609–1619, Sept. 2009.

[7] Google Image Searching,
`http://www.google.co.jp/imghp?hl=ja\&tab=wi`

[8] TinEye, `http://www.tineye.com/`

[9] Flickr, `http://www.flickr.com/`

[10] G. Csurka, C. R. Dance, L. Fan, J. Willamowski and C. Bray, “Visual Categorization with Bags of Keypoints”, In ECCV International Workshop on Statistical Learning in Computer Vision, 2004.

[11] D. G. Lowe, “Object recognition from local scale-invariant features”, Proc. 7th International Conference on Computer Vision, ICCV’99, vol. 2, pp. 1150–1157, 1999.

[12] D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[13] 秋山瑞樹, 柳井啓司, “特定物体認識手法による大量画像を用いた一般物体認識”, 画像の認識・理解シンポジウム (MIRU), 2010.

[14] Svetlana Lazebnik, Cordelia Schmid, JeanPonce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories”, Computer Vision and Pattern Recognition, 2006.

[15] Vincent Delaitre, Ivan Laptev, Josef Sivic “Recognizing human actions in still images:a study of bag-of-features and part-based representations”, Proceedings of the 21st British Machine Vision Conference, September, 2010,

[16] A. Torralba, R. Fergus and W. T. Freeman, “80 million tiny images: a large dataset for non-parametric object and scene recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.30(11), pp. 1958-1970, Nov. 2008.

[17] 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’99, pp. 459–468, 2006.