社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－CVIM－155 (9)
2006／9／8

特徴ベクトルの近傍探索と物体認識の効率に関する実験的検討

野口 和人† 中居 友弘†† 黄瀬 浩一†† 岩村 雅一††

† 大阪府立大学工学部 〒599–8531 大阪府堺市中区学園町 1–1
†† 大阪府立大学大学院工学研究科 〒599–8531 大阪府堺市中区学園町 1–1
E-mail: {noguchi,nakai}@m.cs.osakafu-u.ac.jp, {kise,masa}@cs.osakafu-u.ac.jp

あらまし 画像の局所記述子として SIFT や PCA-SIFT を用いる物体認識では，一画像あたりの特徴ベクトルの数が膨大となるため，特徴ベクトルの照合速度が全体の効率を左右する．特徴ベクトルの照合は，最近傍探索の枠組みで捉えられるため，最近傍探索の高速化が中心的課題となる．本稿では，「物体認識のためには，正しく照合される特徴ベクトルの数が，誤って他の物体に照合される数を上回ればよい」という観点から，近似最近傍探索における距離計算の回数を大幅に削減する 2 つの手法を提案する．一つは，多くの距離計算が避けられない特徴ベクトルを削除する方法である．もう一つは，全く距離計算を行わない手法である．1 万画像のデータベース，2,000 画像の検索質問を用いた実験から，ANN を用いる手法と比べて同じ認識率を実現するために必要な処理時間を 1/2 から 1/3 に削減でき，例えば認識率 98%，処理時間 8.3ms/query を達成できることが分かった．

キーワード 物体認識，近似近傍探索，PCA-SIFT，ANN，LSH，ハッシュ

Experimental Investigation of Relation Between Near Neighbor Search Methods for Feature Vectors and Efficiency of Object Recognition

Kazuto NOGUCHI†, Tomohiro NAKAI††, Koichi KISE††, and Masakazu IWAMURA††

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

Abstract Efficiency of object recognition methods using local descriptors such as SIFT and PCA-SIFT depends largely on the speed of matching between feature vectors since images are described by a large number of feature vectors. Because the matching is considered to be "nearest neighbor (NN) search" of feature vectors, the problem is paraphrased by "how to make the NN search efficient". For the object recognition, it is required that the number of incorrect matching does not exceed that of correct matching. In other words, a certain number of incorrect matching is acceptable. This observation allows us to make NN search more efficient using *approximate* NN search with reduced distance calculation. For this purpose, we propose two methods: one is to eliminate feature vectors that require a number of distance calculations. The other is to use *no* distance calculation. From experimental results with 10,000 database images and 2,000 query images, it is shown that the proposed method is two to three times efficient as compared to a method using ANN and can achieve, recognition rate of 98% with 8.3 ms/query.

Key words Object recognition, Approxiate nearest neighbor search, PCA-SIFT, ANN, LSH, Hash

1. まえがき

デジタルカメラやカメラ付き携帯電話の普及に伴って，単にスナップ写真を撮るだけではなく，カメラを情報入力機器としても利用したいという要望が高まっている．一つの可能性として，カメラで捉えた物体を認識し，それに応じた情報処理を行うことが考えられる．

何も制限を設けずに物体を認識することは未だに困難と言わざるを得ないが，近年の技術的な発展により，対象に制約を加えることができれば物体認識は現実味を帯びてきている．例え

ば，対象が3次元物体ではなく平面上のパターン(平面物体)であること，物体のクラス（例えば，写真の物体が車というカテゴリに属するかどうか）を認識するのではなく，インスタンス（車のあるモデルをある角度から撮影した写真かどうか）を認識することなどが仮定できれば，すでにサービスが可能なレベルにある(注1)．このような平面物体の認識が可能になれば，ポスターや商品の写真を撮影することによる誘導だけではなく，既存の画像やビデオの自動索引付けへの道も開けてくる．

さて，物体認識のためには，画像から特徴を抽出する必要がある．本稿では，平面物体を対象とした局所記述子 (local descriptor) を用いる認識に着目する．局所記述子とは，画像の局所的な特徴を捉えて多次元の特徴ベクトルを抽出し，画像を記述するものである．値が局所的に決定されるので，隠れや画像の変動に対して比較的ロバストであるという性質がある．

局所記述子を用いた物体認識法では，2つの画像から得た特徴ベクトル同士の距離を測り，最近傍のものに対応付けることが基本演算となる．そして，カメラで得た画像と，データベース中の多数の画像の間で特徴ベクトルを対応付け，データベース中の画像に対して投票する．最後に，得票数の最も多い画像のラベルを「認識結果」として出力する．ただし，特徴ベクトルの次元数が数十から数百，数が，画像あたり数百から数千というオーダであることを考えると，単純に全ての組み合わせの距離を計算することは実用的ではないことが分かる．

ところが，近年の最近傍探索技術の発展 [1], [2] により，膨大な数の特徴ベクトルを短時間で探索することが可能となってきた．特に ANN(Approximate Nearest Neighbor) [3], LSH (Locality Sensitive Hashing) [4] は，各々，木構造，ハッシュ表を用いて，近似的な最近傍探索を行うことにより，高速な探索を実現している．国内では，例えば，正確な最近傍探索に対する SR-Tree [5] に加え，近似最近傍探索の手法として小林らの分散コーディング [6] がある．

さらに，物体認識という観点から，和田らは最近傍識別器 [7] という概念とそれを具体化した KDDT [8] という手法を提案している．各物体が一つの特徴ベクトルに対応しており，その物体のカテゴリを認識する問題を考えるとき，認識対象の物体から得た特徴ベクトルがどのカテゴリの特徴ベクトルに近いのかが分かればよく，「最近傍」の特徴ベクトルを求める必要はない．これにより，正確な最近傍探索を用いる場合に比べて，数倍から数百倍の高速化が可能であることが示されている．

本稿では，局所記述子のように，各物体を多数の特徴ベクトルで表現する場合に対して，高速な物体認識法を検討する．投票によって認識結果を決定する場合，最近傍識別器と同様，個々の特徴ベクトルに対しては，最近傍の特徴ベクトルを求める必要はなく，対応する画像がどれであるのかが分かればよい．さらに，別の物体の特徴ベクトルに誤って照合しても，最終的に正解と不正解の得票数が逆転しなければよい．従って，特徴ベクトルの探索の正確さを犠牲にして，大幅な近似最近傍探索を実施することにより，処理時間を稼ぐことが可能である．

このような観点から，本稿では，ハッシュ表を用いて2通りの高速化手法を提案する．高速化の一つは，特徴ベクトルの距離計算の回数を減らす方法である．具体的には，近傍に多数の特徴ベクトルがあって，多くの距離計算が避けられないような場合，そのような特徴ベクトルを破棄することによって高速化を図る．もう一つは，距離計算を一切行わない手法である．処理としてはハッシュ表を引いて投票することだけを行う．データベースとして1万画像，検索質問として2,000画像を用いた実験の結果，近似最近傍探索を行う ANN や LSH を用いる場合と比較して，同程度の認識率を得るために必要な処理時間が，1/2 から 1/3 程度で済むことが分かった．

2. 特徴ベクトル

まず，本手法で利用する特徴ベクトルについて述べる．

2.1 SIFT

SIFT (Scale Invariant Feature Transform) [9] とは，Lowe によって提案された特徴点とそれに付随する特徴ベクトルの抽出法である．その名が示す通り，画像の拡大縮小，回転や視点のずれに対して，ロバストであるという特徴を持つ．従来は処理時間が問題視されてきたが，GPU(Graphical Processing Unit) の利用によって，高速な処理が可能となりつつある．

本手法では，Lowe によって提供されているソフトウェア(注2)を用いて特徴点を抽出する．特徴ベクトルは，128次元の整数値 (0–255) のベクトルである．

2.2 PCA-SIFT

Ke らは，SIFT の特徴ベクトルに対して，主成分分析 (PCA) を適用することにより，SIFT の安定性や識別性を向上させる PCA-SIFT を提案している [10]．本手法では，この PCA-SIFT を画像の局所記述子として利用する．PCA-SIFT によって得られる特徴ベクトルは，36次元の実数値ベクトルである(注3)．

5. の実験に使った画像を用いて PCA-SIFT を計算すると，各次元は図1に示すような値の分布を持つことが分かった．1次元目は双峰性の分布であり，2次元目以降は単峰性の分布を示す．また，次元が大きくなるにつれて分散が小さくなる．平均値はいずれも0の付近である．

3. 物体認識と近似最近傍探索

3.1 投票による物体認識

画像データベースに多数の画像が納められており，各々の画像は1つの物体を表すものとする．認識対象の画像（以下，検索質問と呼ぶ）が与えられたとき，物体認識のタスクを，検索質問に最もマッチする画像をデータベースから検索することと定義する．

(注1)：大日本印刷によるサービス (http://www.dnp.co.jp/jis/news/2005/051101.html ; クレメンテックの技術 (US. Patent 20040208372) を利用)，オリンパスのサービス (http://www.olympus.co.jp/jp/news/2006a/nr060629syncj.cfm)，NEC のサービス (Evolution robotics の技術 http://www.evolution.com/core/vipr.masn を利用) などがある．

(注2)：http://www.cs.ubc.ca/~lowe/keypoints/

(注3)：SIFT から得た特徴ベクトルに対して，http://www.cs.cmu.edu/~yke/pcasift/ で提供されているソフトウェアを用いることにより，36次元のベクトルに変換される．

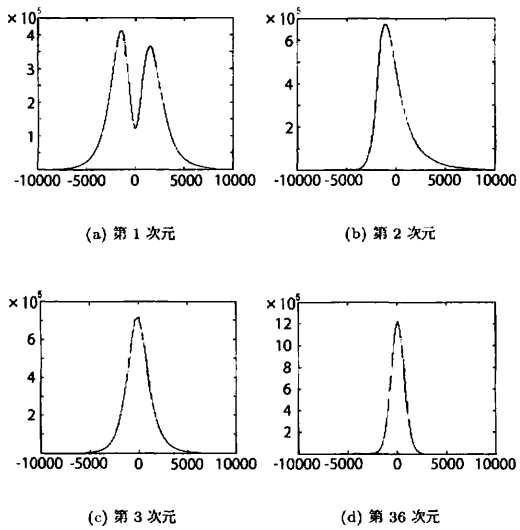


図 1　特徴ベクトルの値分布．横軸は各次元の値，縦軸は頻度．

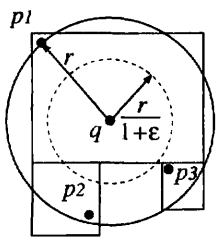


図 2　ANN による近似最近傍探索

この目的のため，本手法では投票方式を用いる．いま検索質問の画像を Q, データベース中の画像を I_p と表す．また，Q, I_p から得た d 次元特徴ベクトルを，$\boldsymbol{q}_i$, $\boldsymbol{p}_j$ と表す．近似最近傍探索の結果，$\boldsymbol{q}_i$ に対応する特徴ベクトルとして，$\boldsymbol{p}_j$ が得られたとすると，画像 p に 1 票を投じる．このような投票を Q から得られた全ての特徴ベクトルに対して実行し，最終的に得票数が最大となった画像を認識結果として提示する．

このように，検索質問から得た個々の特徴ベクトルに対して，データベース中の全ての画像から得た特徴ベクトルとの間で近似最近傍探索を行うため，近似最近傍探索をどのように高速化するかがポイントとなる。以下では，代表的な手法として ANN と LSH について述べる．

3.2 ANN

ANN(Approximate Nearest Neighbor) [3] は、木構造を用いて近似最近傍探索を高速に行う手法である。木のノードは、特徴空間を分割した hyperrectangle(以後，セルと呼ぶ) に対応しており、葉ノードには特徴ベクトルも対応つけられている．

ANN による近似最近傍探索の概念を図 2 に示す．ただし，簡単のため，説明に関与しないセルは描いていない．いま，$\boldsymbol{q}$ を検索質問の特徴ベクトル，$\boldsymbol{p}_1$,$\boldsymbol{p}_2$,$\boldsymbol{p}_3$ をデータベース中の画像の特徴ベクトルとし，現在，$\boldsymbol{p}_1$ が近傍のベクトルとして発見されているとする．最近傍探索を実行する場合，実線で示される超球と重なるセルには，$\boldsymbol{p}_1$ より近傍の特徴ベクトルが存在する可能性があるため，探索の対象となる．一方，近似最近傍探索を行う場合，$\boldsymbol{p}_1$ までの距離 r に対して，許容誤差 ε を用いて定義される半径 $r/(1+\varepsilon)$ の超球を考え，それと交わるセルのみを探索の対象とする．これにより，最近傍の特徴ベクトル (図 2 の場合は $\boldsymbol{p}_3$) を発見できない可能性は出てくるが，対象となるセルの数が減少するため，探索時間を削減できる．

3.3 LSH

LSH(Locality Sensitive Hashing) は，ハッシュ表を用いた近似最近傍探索の手法である [4]．ここでは，実験で用いる E^2LSH (Exact Euclidean LSH; 以後単に LSH と呼ぶ) について述べる．

d 次元ベクトル $\boldsymbol{p}=(p_1,...,p_d)$ を考える．LSH では，一つの特徴ベクトルを L 通りの k 次元ベクトルに変換し，各々に対応する L 個のハッシュ表に登録する．検索時には，検索質問の特徴ベクトル $\boldsymbol{q}$ を用いて，全てのハッシュ表を検索し，得られた特徴ベクトル $\boldsymbol{p}_1,...,\boldsymbol{p}_s$ の中から $\boldsymbol{q}$ とのユークリッド距離が最小のものを結果とする．このように複数のハッシュを用いることによって，良い近似最近傍が安定的に求められる．

もう少し具体的に見てみよう．処理は検索質問の特徴ベクトル，データベース中の特徴ベクトルに共通するので，一般に特徴ベクトルを $\boldsymbol{v}$ で表す．$\boldsymbol{v}$ は，次の手順で生成された L 個の関数 $g_1(\boldsymbol{v}),...,g_L(\boldsymbol{v})$ を用いて，対応する L 個のハッシュ表に格納される．個々の $g_j(\boldsymbol{v})$ は，$\boldsymbol{v}$ を $g_j(\boldsymbol{v})=(h_1(\boldsymbol{v}),...,h_k(\boldsymbol{v}))$ のように k 次元ベクトルに変換するものである．$h_i(\boldsymbol{v})$ は，$\boldsymbol{v}$ を自然数に変換する関数であり，次のような形を持つ．

$$h_{\boldsymbol{a},t}(\boldsymbol{v})=\lfloor\frac{\boldsymbol{a}\cdot\boldsymbol{v}+t}{w}\rfloor \tag{1}$$

ここで，$\boldsymbol{a}$ は，各次元が独立に正規乱数に従って生成された d 次元ベクトルであり，t は $[0,w]$ の一様乱数によって定められるスカラである．このような値を用いることによって，$\boldsymbol{v}_1$ と $\boldsymbol{v}_2$ のユークリッド距離が小さければ，それだけ $h_i(\boldsymbol{v}_1)=h_i(\boldsymbol{v}_2)$ となる可能性が高いという効果を実現できる．

LSH では，k 個の異なる $\boldsymbol{a}$,t を用いて k 次元ベクトルとすることにより，ユークリッド距離の離れた $\boldsymbol{v}$ が同じベクトルとならないようにしている．一方で，L 個の g_j を用いることにより，ユークリッド距離の近い $\boldsymbol{v}$ が対象から漏れてしまうことを防いでいる．

4. 衝突の削減による高速近似近傍探索

4.1 考え方

物体の局所的な特徴を捉えた特徴ベクトルを用いて，投票処理によって物体を認識する場合，検索質問の特徴ベクトルに対して，必ずしも最近傍の特徴ベクトルをデータベースから発見する必要はなく，特徴ベクトルに付与された画像のラベルが正解のものであればよい．さらに，認識結果が投票によって決定されるため，正解の得票数が逆転しなければ，誤った票が他の画像に入っても問題は生じない．このような特性を活かして，

本手法では，大幅な近似を施すことにより，ANN や LSH を用いる場合と比べて高速な処理を実現する．

ANN や LSH を用いる場合，最も計算時間が必要な部分は，$\boldsymbol{q}$ と $\boldsymbol{p}_j$ の距離計算である．従って，これをいかに削減するかがポイントとなる．ただし，検索の精度が著しく低下したり，必要なメモリ量が大幅に増大すると問題となる．

本手法では，データの特性を活かしたハッシュ関数を用いることによって，高速化の問題を解決する．手法としては次の 2 通りを考える．一つは，距離計算を行うが，その対象となる特徴ベクトルの数を削減する方法である．具体的には，多数の衝突が生じている場合，すなわち，同じハッシュ値を持つ特徴ベクトルが多数登録されているとき，それらを予めハッシュ表から消去する．これにより，検索質問の特徴ベクトルあたりの距離計算回数を一定値以下に削減することが可能となる．もう一つは，全く距離計算を行わない方法である．衝突回数に応じた消去を行うと，ハッシュ表には画像を識別する上で効果的な特徴ベクトルが残ることになる．そこで，これらの特徴ベクトルを用いれば，投票のみでも正しい結果が得られると期待できる．

4.2 データ登録

まず，2 手法に共通のデータ登録について述べる．

同様にハッシュ表を用いる LSH では，ハッシュ表の数が多くなると大量のメモリを消費する．そこで本手法では，メモリ量を削減するため，ハッシュ表を 1 つだけ使うこととする．

特徴ベクトルをハッシュ表に登録する方法は次のとおりである．PCA-SIFT によって得られた 36 次元の実数値ベクトル $\boldsymbol{p}$ の第 1 次元から第 d 次元までをとり，$\hat{\boldsymbol{p}} = (p_1, p_2, ..., p_d)$ とする．次に，

$$u_j = \begin{cases} 1 & \text{if } p_j \geqq 0, \\ 0 & \text{otherwise,} \end{cases}$$

によって各次元を 2 値化し，ビットベクトル $\boldsymbol{u} = (u_1, ..., u_d)$ を作成する．そして，

$$H_{\text{index}} = \left(\sum_{i=0}^{d} u_i 2^i \right) \mod H_{\text{size}} \qquad (2)$$

によってハッシュのインデックスを求め，ハッシュ表に登録する．ここで H_{size} は，ハッシュ表のサイズである．ハッシュ表に登録するデータは，距離を用いるか否かによって異なる．距離を用いる場合には，特徴ベクトル $\boldsymbol{p}$ に対する画像 ID のほか，$\boldsymbol{p}$ そのものを登録し，検索時の距離計算に用いる．一方，距離を用いない場合には，$\boldsymbol{p}$ の登録は不要である．

登録時に衝突が生じた場合は，図 3 のように，チェイン法により複数の特徴ベクトルをリストとして登録する．このとき，リストが長くなりすぎると，距離計算のコストがかかりすぎるという問題が生じる．そこで本手法では，リスト長 n に対する閾値 c を設け，$n > c$ を満たすとリスト全体をハッシュ表から削除する (注4)．同じハッシュ値を持つ特徴ベクトルが多いということは，その特徴ベクトルが画像の識別にあまり寄与しないことを意味する．従って，削除をしても影響は比較的少ないと考えられる．

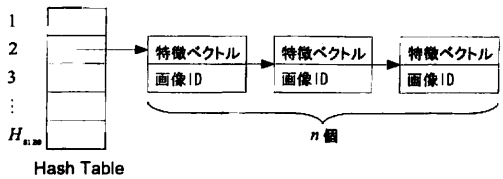


図 3 ハッシュ表

以上の処理を，データベースに登録する全ての特徴ベクトルに対して施すことにより，データの登録は完了する．

4.3 距離計算を用いる方法

次に距離計算を用いる検索について述べる．本手法では，検索質問 Q から得た各特徴ベクトル $\boldsymbol{q}$ に対して，上記のハッシュ表から特徴ベクトルを検索する．得られた特徴ベクトルの集合を P とすると，次に P の中から $\boldsymbol{q}$ の最近傍となる特徴ベクトル $\boldsymbol{p}_*$ を求める．そして，$\boldsymbol{p}_*$ に対応する画像 ID に投票する．

この処理において，最も重要なステップは，いかに $\boldsymbol{q}$ に対する特徴ベクトルを検索するかにある．最も単純な手法は，登録時と同様に $\boldsymbol{q}$ に対してもビットベクトルを求め，ハッシュ関数によって同じハッシュ値を持つ特徴ベクトルを求めることである．ところが，このような処理では，距離の計算回数は十分削減できるものの，次の理由によって十分な認識率を得ることができない．特徴ベクトルの各次元の値は撮影条件によって変動することがある．もし，閾値を超えるような変動があると，ビットベクトルが異なるものとなり，もはや対応する特徴ベクトルを得ることができなくなる．

LSH では同様の問題に対処するため，式 (1) において，一様乱数 t を値に加えることにより，閾値付近の値をランダムに移動させている．また，小林らの手法 [6] では，特徴ベクトルに回転行列をかけることで，閾値の相対的な位置を変化させている．

本手法では，値の変動幅 e をパラメータとして，変動への対処を施す．具体的には，$\boldsymbol{q} = (q_1, ..., q_d)$ とするとき，$|q_j| \leq e$ を満たす次元 j に対しては，u_j だけではなく $u'_j = (u_j + 1) \bmod 2$(0 ならば 1, 1 ならば 0) も用いて，特徴ベクトルを検索する．ただし，このような「両方試す」という処理を制限なく導入すると，膨大な計算時間が必要となってしまう．この処理では，処理の対象となる次元数を b とすると，2^b 通りのビットベクトルを用いてハッシュ表にアクセスすることになる．そこで本手法では，b をあまり大きくない値に留めることとする．$|q_j| \leq e$ を満たす次元の数が b を上回るときには，次元のインデックスが小さいものから b 個を採用する (注5)．

なお，このような変動への対処は，検索時ではなく登録時に行うことも可能である．具体的には，登録の際に同様にビットベクトルを 2^b 個作成し，ハッシュ表に登録する．こうすると，

(注4)：予備実験として，情報検索で用いられる各種の重み付けも試したところ，精度的にあまり大きな差はなかった．削除は精度だけではなく，速度にも有利であるため，本手法では重み付けではなく削除を採用している．

(注5)：対象となる次元を，確率的に決めることも考えられる．実際に試したところ，精度にはほとんど差が出ず，計算時間が余分に必要であった．

(a) A の画像例

(b) B の画像例

(c) C の画像例

図 4　登録画像の例

検索時に複数のビットベクトルを用いてハッシュ表にアクセスする必要がなくなるため，処理時間の短縮が期待できる．しかしながら，多数の特徴ベクトルを登録するため，メモリへの負担は大きくなる．予備実験の結果，処理時間には大きな差がなく，メモリへの負担が目立ったため，本手法では，検索時に変動に対処することとした．

4.4　距離計算を用いない方法

距離を用いない方法では，検索質問の特徴ベクトル q に対して上記のような距離計算を施して近似最近傍を求めるのではなく，ハッシュ表から得た特徴ベクトルの集合 P に属する全ての特徴ベクトル $p \in P$ に対して投票処理を施す．処理のパラメータは，距離を用いない方法と同様，特徴量の変動幅 e，変動に対処する次元の数 b の 2 つである．

5.　実　　験

提案手法の有効性を検証するため，比較実験を行った．

5.1　実験条件

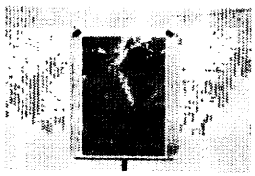
5.1.1　画像データベース

最初に，実験に用いる画像について説明する．まず，収集方法の異なる A，B，C の 3 種類のデータセットを準備した．A は，Google のイメージ検索を用いて収集した 3,100 枚の画像である．検索キーワードとしては，ポスター，雑誌，表紙などを用いた．図 4(a) に例を示す．B は PCA-SIFT のサイト (注6) で公開されている画像であり，画像数は 18,500 枚である．このデータは主に自然写真や人物の写真などで構成されている．図 4(b) に例を示す．C は，写真共有サイトの flickr において animal，birthday，food，japan などのタグにより収集した 78,400 枚の画像からなる．主に図 4(c) に示すような物体や自然の写真，人物の写真などを含む．なお，収集の際には，600×600 pixel 以下のサイズの画像は除外し，画像の長辺が 640 pixel 以下になるように縮小した．また，特徴ベクトルが 100 個以下の画像も除外した．画像の一辺の長さの平均は A，B，C それぞれ 498，612，554 pixel であった．

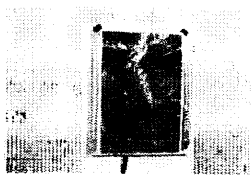
(注6)：http://www.cs.cmu.edu/~yke/pcasift/

表 1　データベースに含まれる画像数

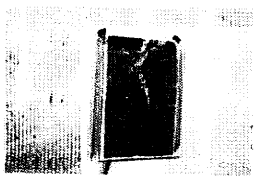
データセット	DB1	DB2	DB3	DB4	DB5
A	334	1,667	3,100	3,100	3,100
B	333	1,667	3,450	18,500	18,500
C	333	1,666	3,450	28,400	78,400
合計	1,000	5,000	10,000	50,000	100,000



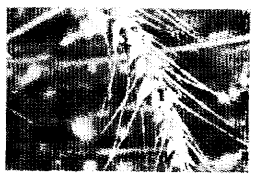
(a) 撮影角度 90°



(b) 撮影角度 75°



(c) 撮影角度 60°



(d) 撮影範囲一部

図 5　検索質問の例

次に，A，B，C の画像を用いて，表 1 に示した画像数からなるデータベース，DB1，...，DB5 を作成し，実験に用いた．ここで，大きいデータベースは，小さいデータベースをその一部として含む．なお，DB3 からは，一画像あたり平均 2,069 個の特徴ベクトルが抽出された．

5.1.2　検索質問画像

検索質問として，次の手順で作成した画像を 2,000 枚用いた．まず，DB1 に含まれる画像の中で A，B，C から，それぞれ 100，200，200 枚を無作為に選択し，A4 の紙面に印刷した．次に，カメラを用いて印刷した紙面を撮影した．撮影した画像の例を図 5 に示す．図に示す通り，紙面全体が写る配置で，紙面に対するカメラの光軸の角度 θ を 90°，75°，60° に変化させた．また，角度を 90° として紙面の一部分を撮影した．その結果，1 枚の紙面に対して，合計 4 通りの画像を得た．さらに，撮影した画像を 512 × 341pixel に縮小し，PCA-SIFT により特徴ベクトルを求めた．その結果，画像一枚あたり平均 605 個の特徴ベクトルが得られた．なお，印刷には OKI C5200n(カラーレーザプリンタ)，撮影には CANON EOS Kiss Digital (630

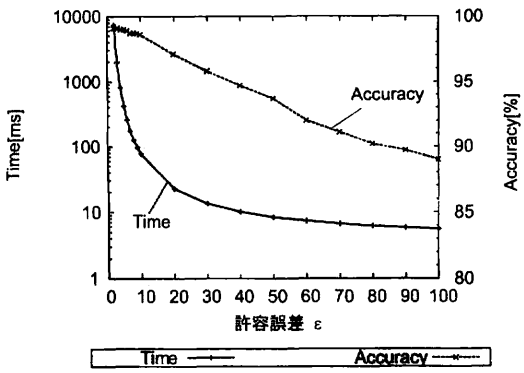


図 6 ANN: 許容誤差 ε と認識率，処理時間の関係

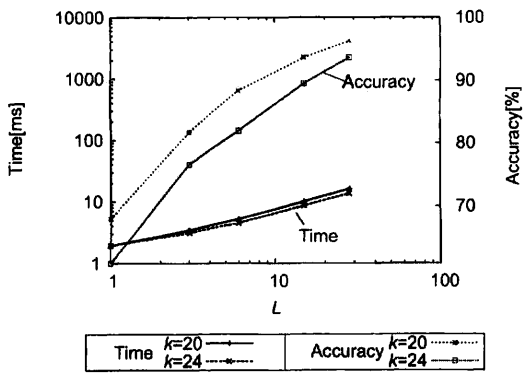


図 7 LSH: パラメータ L と認識率，処理時間の関係

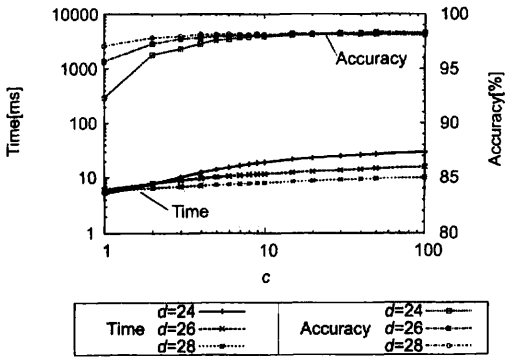


図 8 距離計算あり: c と認識率，処理時間の関係

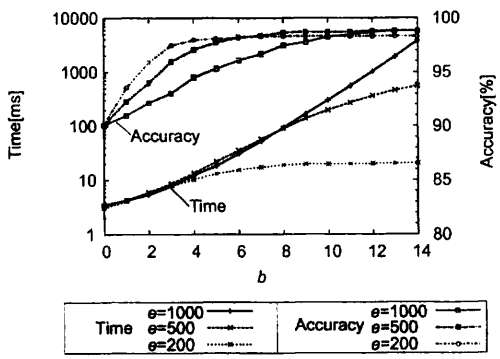


図 9 距離計算あり: b, e と認識率，処理時間の関係

万画素）と付属のレンズ EF-S 18-55mm USM を用いた．

5.1.3 評　　価

実験では，近似最近傍探索の比較手法として ANN と LSH を用い[(注7)]，提案手法と比較した．評価基準としては，認識率と処理時間を用いた．認識率は，検索質問の画像が正しく認識できた割合を表す．また，処理時間は，検索質問の画像 1 枚あたりの検索に要した時間を表す．ただし，特徴ベクトルの抽出に必要な時間は含めていない．なお，実験に用いた計算機は，CPU AMD Opteron 2.8GHz，メモリ 16GB のものである．

5.2 DB3 を用いた比較実験

まず，DB3 を用いて各手法のパラメータと認識率，処理速度の関係について述べる．

5.2.1 ANN

ANN を用いて，許容誤差 ε を 2 から 100 まで変化させた結果を図 6 に示す．ε の増加に伴って，認識率，処理時間が減少していることが分かる．ε が 2 から 10 程度までは，処理時間の減少に比べ，認識率の減少は緩やかである．

5.2.2 LSH

図 7 に，LSH を用いて変換後のベクトルの次元数 k とハッシュ関数の数 L を変化させた結果を示す．まず，L の増加に伴って，認識率，処理時間が増加していることが分かる．L を更に増加させると，認識率を向上させることができると考えられるが，メモリ不足により実行できなかった．また，図示されているもの以外にも種々の k について試したところ，k を減少させると，認識率は改善するものの，処理時間が増大することが分かった．この理由は，k が小さいと，距離計算の対象となる特徴ベクトルの数が増加するためであると考えられる．

5.2.3 提案手法 (距離計算あり)

距離計算ありの提案手法を用いて，衝突の閾値 c と認識率，処理時間の関係について調べた．このとき，ハッシュ表のサイズとしては $H_{\text{size}} = 2^d$ とした．$e = 200, b = 7, d = 24, 26, 28$ とし，c を変化させた結果を図 8 に示す．c が減少するにつれ，処理時間が減少していることが分かる．ただし，c を小さくしすぎると，認識率が低下した．これは，認識に寄与していたものも削除してしまったためと考えられる．一方，c を増加させた場合に，計算時間は増加するものの，認識率が減少することはほとんどなかった．これは，最近傍にはなり得ない特徴ベクトルを検索したとしても，距離計算によって排除可能なためと考えられる．

また，b と認識率，処理時間の関係について調べた．ハッシュのインデックスを求めるために使用する次元を $d = 26$ とした

(注7)：ANN としては http://www.cs.umd.edu/~mount/ANN/, LSH としては http://www.mit.edu/~andoni/で提供されているソースコードを用いた．

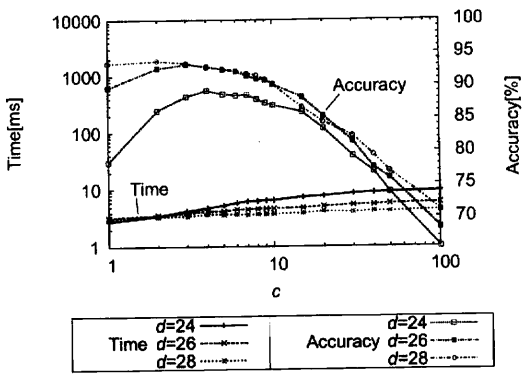


図 10 距離計算なし: c と認識率，処理時間の関係

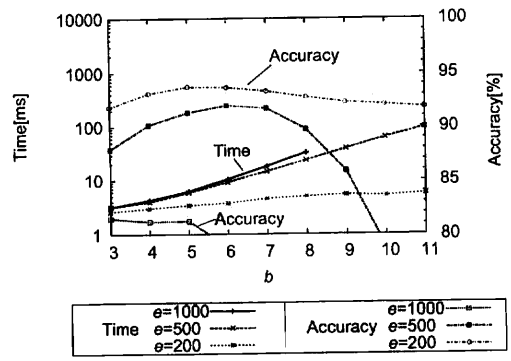


図 11 距離計算なし: b, e と認識率，処理時間の関係

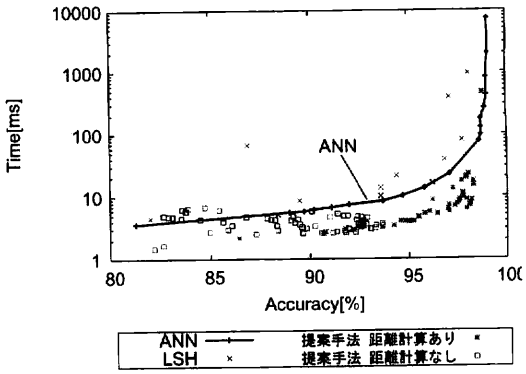


図 12 各手法の比較

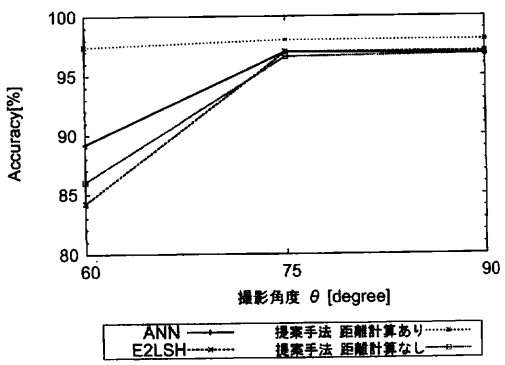


図 13 撮影角度と認識率の関係

上で，$e = 200, 500, 1000$，$c = \infty$ とし，b を変化させた結果を図 9 に示す．b を増加させると処理時間は増加するものの，認識率が向上することが分かる．b が比較的小さい場合は，$e = 200$ の場合に認識率が高い．

5.2.4 提案手法 (距離計算なし)

次に，距離計算なしの提案手法を用いて，c と認識率，処理時間の関係について調べた．$d = 24, 26, 28$，$e = 200$，$b = 5$ とし，c を変化させた結果を図 10 に示す．$d = 24, 26, 28$ の値について，それぞれ $c = 2, 3, 4$ という小さい値のときに認識率が最大となった．これは，距離計算を用いない手法では，c が大きくなるにつれて，最近傍にはならない特徴ベクトルが多数投票に関与するためと思われる．図 8 に示した距離計算を用いる場合と好対照であることが分かる．

また，b と認識率，処理時間の関係についても調べた．$d = 28$，$e = 200$，$c = 2$ とし，b を変化させた結果を図 11 に示す．$b = 5$ までは，b の増加に伴って認識率が向上しているが，それ以上 b が増加すると，認識率は低下している．これは，b の増加によって，最近傍とはなり得ない不適切な特徴ベクトルを介した投票が増大したためと考えられる．図 9 の距離を計算するものでは，b を増加させた場合に，認識率が減少することはなかった点を考えると，同様に好対照であると言える．

5.2.5 各手法の比較

各手法の特徴を比較するため，パラメータをさまざまに変え，横軸に認識率，縦軸に処理時間を描いたグラフを図 12 に示す．ANN でパラメータを変化させたものを線で描き，評価の基準とした．右にプロットされているものほど認識率が高く，下にプロットされているものほど処理時間が短い．そのため，右下にプロットされているものほど優れていると言える．LSH は，ほぼ ANN の線を越えることは無かった．提案手法で距離の計算を行うものは，認識率が 98%以下の場合は，ANN よりも優れていた．提案手法で距離の計算を行わないものは，ほとんどの場合で ANN より優れていた．

次に，撮影角度と認識率の関係を調べた．処理時間がおよそ 10ms で認識率の最も良いものを図 13 に示す．パラメータは，ANN $e = 40$，LSH $k = 20, L = 15$，距離計算ありの手法 $e = 200, b = 4, c = 8, d = 24$，距離計算なしの手法 $e = 200, b = 5, c = 2, d = 28$ である．ただし，距離計算なしの手法による処理時間は 3.4ms のものを示している．距離計算ありの手法は，同じ処理時間で，ANN，LSH と比べ高い認識率が得られていることが分かる．距離計算なしの手法では，$\theta = 60°$ の場合を除くと，1/3 の処理時間で ANN と同程度の認識率を得られることが分かる．

各種パラメータの代表的な値を用いた認識率と処理時間を

表 2 各手法の認識率 [%] と処理時間 [ms]

手法	パラメータ	60 度		75 度		90 度		一部		平均	
		精度	時間	精度	時間	精度	時間	精度	時間	精度	時間
ANN	$e=3$	98.6	1261.2	99.0	1304.9	99.2	1294.9	99.6	4020.0	99.1	1970.2
	$e=10$	98.2	48.4	98.6	53.0	98.4	52.8	99.0	151.0	98.6	76.3
	$e=20$	94.6	13.9	98.2	16.0	98.0	16.0	97.4	43.7	97.1	22.4
LSH	$k=20, L=28$	91.4	9.5	98.0	11.7	97.6	12.0	97.8	30.8	96.2	16.0
	$k=20, L=15$	84.2	6.0	97.0	7.2	97.0	7.4	96.2	19.5	93.6	10.0
提案手法	$b=9, e=200, d=26$	97.8	12.1	98.6	15.0	98.4	14.8	98.4	36.7	98.3	19.6
距離あり	$b=4, e=200, c=50, d=26$	97.2	5.7	98.4	6.5	98.4	6.6	98.0	14.2	98.0	8.3
提案手法	$b=5, e=200, c=2, d=28$	86.0	2.5	96.6	2.9	96.8	2.7	95.4	5.4	93.7	3.4
距離なし	$b=0, c=11, d=24$	57.6	1.4	89.6	1.5	91.8	1.5	91.6	2.2	82.7	1.6

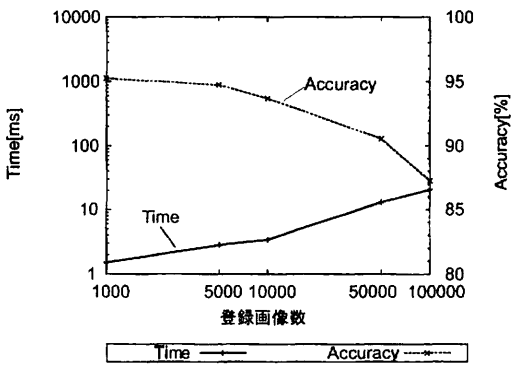


図 14 登録画像数と認識率，処理時間

表 2 に示す．距離計算ありの手法は，ANN に比べ同程度の認識率を，1/3 程度の処理時間で実現していることが分かる．一方，距離計算なしの手法では，平均の精度は ANN に及ばない．ただし，その原因は $\theta = 60°$ の場合に精度が低いことにある．$\theta \geq 75°$ に限定できる状況では，96%程度の認識率を 4ms 以下という短い処理時間で実現可能であることが分かる．

5.3 DB1–DB5 を用いた実験

距離計算なしの手法を除く全ての手法では，検索のために元の特徴ベクトルのデータを保持しなければならないため，DB4, DB5 のデータについては，メモリ不足で検索を実行できなかった．一方，距離計算を用いない手法は，ハッシュ表に画像 ID のみを登録すればよいため，メモリへの負担が少なく，10 万画像までの実験を行うことができた．そこで，$e = 200, d = 28$ とし，b と c を変化させ，登録画像数と認識率，処理時間の関係について調べた．最も認識率の良いものを図 14 に示す．そのときの b は，DB1 から順に 5, 6, 5, 6, 5 で，c は 1, 1, 2, 4, 5 であった．登録画像数を 10 万件に増加させた場合でも，認識率 87.3%，処理時間 20.6ms を得た．$\theta = 60°$ の場合を除外すると認識率は 91.4%となる．このように，距離計算を用いない手法は，認識率という点では他に及ばないものの，ある程度の認識率で満足できる場合には，スケーラビリティという点で優れた手法と言える．また，処理がハッシュ表へのアクセスと投票という単純なものであるため，この面での利点もあると考えられる．

6. ま と め

本稿では，平面物体を対象とし，局所記述子として PCA-SIFT を用いた投票方式による物体認識法として，距離計算を用いる手法，用いない手法の 2 つを提案した．本手法の特徴は，ANN や LSH という従来の近似最近傍探索法を用いる場合と比べて，同じ認識率を達成するために必要な計算時間が 1/2 から 1/3 でよい点にある．また，距離計算を用いない手法では，メモリの使用量が少ないため，スケーラビリティという点でも優れていることが分かった．

今後の課題には，より実際的な環境で有効性を検証することに加え，平面物体以外にも適用することが挙げられる．

文 献

[1] P. Indyk, Nearest neighbors in high-dimensional spaces, Handbook of discrete and computational geometry (Eds. by J. E. Goodman and J. O'Rourke), Chapman & Hall/CRC, pp.877–892, 2004.

[2] G. Shakhnarovich, T. Darrell and P. Indyk Eds., Nearest-neighbor methods in learning and vision, The MIT Press, 2005.

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

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

[5] 片山紀生，佐藤真一，"類似検索のための索引技術，" 情報処理，vol.42, no.10, pp.958–964, Oct., 2001.

[6] 小林卓夫，中川正樹，"分散コーディングによる高次元の最近傍探索，" 信学技報 PRMU2006–41, June, 2006.

[7] 和田俊和，"空間分割を用いた識別と非線形写像の学習 (1) 空間分割による最近傍識別の高速化，" 情報処理，vol.46, no.8, pp.912–918, Aug., 2005.

[8] 柴田智行，加藤丈和，和田俊和，"K-d decision tree とその応用 — 最近傍識別器の高速化と省メモリ化，" 信学論 (D-II), vol.J88-D-II, no.8, pp.1367–1377, Aug., 2005.

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

[10] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors, CVPR2004, Vol. 2, pp.506–513, 2004.