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.

特定物体認識における Bloomier Filter を用いたメモリ削減

井上　勝文†　　黄瀬　浩一†

† 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:　†inoue@m.cs.osakafu-u.ac.jp, †kise@cs.osakafu-u.ac.jp

あらまし　特徴ベクトルの最近傍探索を用いた特定物体認識では，特徴ベクトルの情報を記録するために莫大なメモリ量が必要となる．そこで，特徴ベクトル間の距離を計算しないことで，特定物体認識に必要なメモリの使用量を削減する枠組みを提案する．本手法の特徴は，ハッシュ表などなどと比べて空間効率の良い Bloomier Filter を用いる点にある．また，平面および 3 次元特定物体の認識実験を通して，ハッシュ表を用いる手法と提案手法を比較し，有効性について検討する．

キーワード　Bloomier filter，Bloom filter，メモリ削減，特定物体認識，局所特徴量

Memory Reduction with Bloomier Filters for Specific Object Recognition

Katsufumi INOUE† and Koichi KISE†

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

Abstract　Specific object recognition based on nearest neighbor search of feature vectors required a huge amount of memory to store all feature vectors for distance calculation. To solve this problem, we propose a memory reduction method for specific object recognition with a strategy of skipping the distance calculation of feature vectors. The proposed method is characterized by the use of Bloomier filters, which are far memory efficient than hash tables, for the storage of feature vectors. The proposed method is evaluated based on experiments of planar and 3D specific object recognition in comparison to a method with a hash table.

Key words　Bloomier filter, Bloom filter, Memory reduction, Specific object recognition, Local features

1. はじめに

特定物体認識とは，物体のインスタンスを認識するタスクである．本稿では，SIFT [1] などの局所特徴量を用いた特定物体認識について述べる．

我々の周囲には物体のインスタンスが無数にあるため，特定物体認識を真に実用的なものとするには，識別できる物体数の大規模化が必須である．局所特徴量は数十から数百の次元を持つ実数値ベクトルであり，高い識別性を持つため，多数の物体を識別する目的に適している．認識処理の基本は，未知の物体から得た局所特徴量と既知の物体から得た局所特徴量を照合する処理である．特徴ベクトルの照合は最近傍探索によって容易に実現できるものの，その大規模化には，まだいくつかの障害を乗り越える必要がある．最も重要な障害は，処理速度とメモリ量に関するものである．

幸い，処理速度については，これまでに有効な手法が提案されている．その中には，k-d 木などの木構造に基づくもの [1]，ハッシュに基づくもの [2] などがある．これらの手法では，近似最近傍探索を用いることによって，処理速度を大幅に改善している．

一方，メモリ量の問題については十分な解決策は得られていない．局所特徴量は画像あたり数百から数千個にもなるため，物体数が大規模になるとその記録には膨大なメモリが必要となる．これまでに，特徴ベクトルを Visual Word [3] と呼ばれる代表ベクトルに置換することでメモリ使用量を削減する手法が提案されている．しかし，大規模な特定物体認識を行うためには，Visual Word の数を多くしなければならず，高い認識率を得るためには，一つの Visual Word に 2〜3 個の特徴ベクトルを対応づけるのが限界である [4]．この他のメモリ削減手法として，スカラー量子化 [5] や特徴ベクトルの取捨選択 [4], [6] などがある．しかし，上記の近似最近傍探索を用いた照合では，特徴ベクトル間の距離を計算する必要があるため，いずれにせよ個々の特徴ベクトルを記録しなければならない．したがって，記録する特徴ベクトル数に応じたメモリ量が必要となり，削減

には限界が生じる．また，

この問題を解決する一つの考え方は，照合に際して距離計算を放棄することである．これにより特徴ベクトルの記録も不要となるため，大幅なメモリ削減が可能となる．実際，特徴ベクトルをハッシュ表に登録しておき，それを検索することによって認識する手法が提案されている [2]．この手法では，特徴ベクトルの照合を，距離あるいは類似度という量的な概念に基づく類似検索ではなく，ハッシュ関数の値（ハッシュ値）が同じかどうかという一致検索によって実現している．ハッシュ表には特徴ベクトルの有無が記録されるだけなので，大幅なメモリ削減が可能となる．

ところが，この手法にもまだ問題点がある．正しい認識のためには，類似した特徴ベクトルが同じハッシュ値を持つようにしなければならない．ところが，これによって，ハッシュ表に記録される特徴ベクトルに偏りが生じる．その結果，ハッシュ表の大部分には何も記録されず，空間効率が悪化する．

本稿では，この問題を解決するため，Bloomier Filter [7] と呼ばれる空間的効率の良い確率的データ構造を用いた新しい手法を提案する．上記のハッシュ表を用いる手法と同様に，Bloomier Filter でも特徴ベクトルが記録されているかどうかを検索できる．大きな違いは，記録されていない特徴ベクトルを誤って検出するという偽陽性 (False Positive) が生じるかどうかにある．ハッシュ表を用いる手法では偽陽性は生じ得ないが，Bloomier Filter を用いる手法では，一定の偽陽性を容認することの引き替えに，高い空間効率を得ることが可能となる．

また本稿では，平面および 3 次元特定物体の認識実験を通して，ハッシュ表を用いる手法と提案手法を比較し，有効性について検討する．

2. ハッシュ表に基づく従来手法

本節では，ハッシュ表に基づく手法である野口らの手法 [2] について説明する．これは，提案手法の基礎となる手法であり，以後，従来手法と呼ぶ．以下に，具体的な処理について説明する．

2.1 データベース作成

従来手法は，PCA-SIFT [8] で求めた 36 次元の特徴ベクトルを用いる特定物体認識手法である．まず，特徴ベクトル $\boldsymbol{p}$ の第 1 次元から第 d 次元 ($d \leqq 36$) までをとり，$\boldsymbol{p}' = (p_1, p_2, \ldots, p_d)$ を作成する．次にこのベクトル $\boldsymbol{p}'$ を用い，

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

によって各次元を 2 値化したビットベクトル $\boldsymbol{u} = (u_1, u_2, \ldots, u_d)$ を作成する．そして，

$$H_{\text{index}} = \left(\sum_{j=1}^{d} u_j 2^{(j-1)} \right) \tag{2}$$

というハッシュ関数より，ハッシュ表に登録するためのインデックスを求める．求まったインデックスに，特徴ベクトル $\boldsymbol{p}$ に対する物体 ID を登録する．

従来手法では，特徴ベクトルの登録時に衝突が生じた場合，リストとして特徴ベクトルを登録していく．但し，計算コストの削減のために，リスト長に閾値 c を設け，リスト長が c よりも長くなる場合，リスト全体をハッシュ表から削除し，以後の登録を禁止する．これは，同じハッシュ値を持つ特徴ベクトルは互いに類似しているため，物体認識にあまり寄与しないという考えに基づく．以上の処理を全ての特徴ベクトルに対して行い，データベースを作成する．

2.2 認識処理

従来手法では，質問画像から PCA-SIFT で求めた特徴ベクトルを用い，データベース作成処理と同様に，ハッシュ表のインデックスを求める．そして，求めたインデックスに登録されている物体 ID に対して投票を行う．この処理を，質問画像から得られる特徴ベクトル全てに対して行い，最終的に最も得票数が多い物体を認識結果とする．以下に具体的な処理について説明する．

まず，質問画像から得られる特徴ベクトルから，ハッシュ表のインデックスを求める．このとき，特徴ベクトルの各次元の値が，撮影条件などの違いにより変化するため，変動を考慮してインデックスを求める．具体的には，各次元の値に許容変動幅 e を設け対処する．検索質問となる特徴ベクトル $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)$ において，$|q_j| \leqq e$ を満たす次元 j に対して，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．しかし，この処理を全ての次元に対して行うと，処理時間が膨大になってしまうため，処理の対象とする次元数に閾値 b を設ける．これにより，最も多い場合で，検索に用いるビットベクトル $\boldsymbol{u}'$ の数を 2^b 個に抑えている．$|q_j| \leqq e$ となる次元数が b 個より多くなる場合，次元のインデックスの大きいものから b 個採用する．

同じハッシュインデックスに，同じ物体 ID が複数登録されている場合，同一物体には一度しか投票を行わないこととする．また投票には，多くの特徴ベクトルが得られる物体が投票されやすいという状態を避けるため，次式の重みを付ける．

$$\frac{1}{\sqrt{r_i}}, (i = 1, \ldots, R) \tag{3}$$

ここで，r_i は物体 ID i の物体から得られる特徴ベクトル数，R は総物体数である．以上の処理を行うことで，ハッシュ表に登録されている物体 ID に投票を行い，最も得票数の多い物体を認識結果とする．

また，処理を効率化するために，複数の識別器を直列に並べた構成とする．第 s 段では，$b = s - 1$ とした識別器を用いる．この識別器の多段階化による認識手法の特徴は，識別器の段が移ったとき，前の段で検索に用いられたビットベクトル以外のビットベクトルを用いて認識を行うという，差分探索性があることである．このため，最終段の識別器まで移ったとしても，最初から 2^b 個のビットベクトルを用いて認識するのとほぼ同じ処理時間で認識できる．

2.3 従来手法の問題点

従来手法では，認識率を確保するために，ビットベクトルの

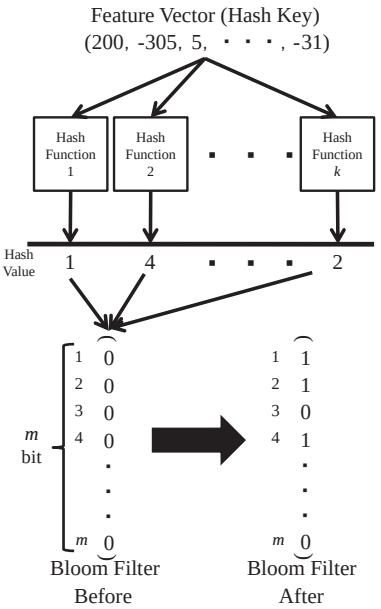

図 1　Bloom Filter の処理の流れ

次元 d の値を大きくする必要がある．しかし，d の値を大きくすればハッシュ表の大きさも指数的に増すという問題点がある．また，予備実験により約一千万個の PCA-SIFT 特徴ベクトルを従来手法でハッシュ表にマッピングしてみたところ，$d = 24$ のとき 65%以上，$d = 28$ のとき 96%以上のインデックスが，一つの特徴ベクトルにも対応しない，またはリストが削除された状態となっていることが分かった．以上から，従来手法には，ハッシュ表の空間効率が悪いという問題点があると言える．この問題に対処するために，本稿では，ハッシュ表と比べて空間効率の良いデータ構造である Bloomier Filter を用いる．

3. Bloom Filter と Bloomier Filter

本節では，提案手法に用いる Bloomier Filter [7] と，その基礎となる Bloom Filter [9] について説明する．

3.1 Bloom Filter

Bloom Filter は，あるデータ集合と要素が与えられたとき，この要素がデータ集合のメンバであるか否かを調べるために用いられる．この手法には，ある要素がデータ集合のメンバではないにもかかわらず，集合のメンバであると判断される偽陽性 (False Positive) が生じる可能性があるという問題点や，データ集合から元の要素を取り出すことができないという問題点がある．一方，同じメモリ量で保存できる要素数は，平衡 2 分探索木やハッシュ表と比べると圧倒的に多いという利点がある．本研究では，この空間効率の良さを利用し，Bloom Filter を特徴ベクトルの圧縮表現に用いる．本稿では，以後，データ集合の要素を特徴ベクトルとして話を進める．以下に，Bloom Filter への特徴ベクトル登録方法と特徴ベクトルの認識方法の具体的な処理について説明する．

図 1 に特徴ベクトルの登録処理の流れを示す．まず，空の Bloom Filter として m bit の配列を用意し，全ての bit を 0 で初期化する．以下では，m を “TableSize”と呼ぶこととする．次に，k 個のハッシュ関数を用意し，それぞれのハッシュ関数に対して，特徴ベクトルをキーとしてハッシュ値を計算する．ここで，得られるハッシュ値は 1〜m の整数値とする．そして，得られた k 個のハッシュ値 $x_1, x_2, \ldots, x_k$ を基に，Bloom Filter の $x_i (i = 1, 2, \ldots, k)$ 番目の bit を 1 にすることで特徴ベクトルを登録する．特徴ベクトルの認識処理も登録処理と同様に，特徴ベクトルをキーとしてそれぞれのハッシュ関数からハッシュ値を得る．そして，得られたハッシュ値に対応する bit が全て 1 になっていれば，データ集合にその特徴ベクトルが含まれているとする．

3.2 Bloomier Filter

Bloomier Filter は，複数の Bloom Filter を用いることで，登録した特徴ベクトルと関連する値を連想させるデータ構造である．本研究では，この性質を利用し，Bloomier Filter を特定物体認識に用いる．以下に，Bloomier Filter の動作について具体的に説明する．

Bloomier Filter で連想させる値が 0 と 1 の二種類のみの例について説明する．まず 2 つの Bloom Filter X と Y を用意する．次に，特徴ベクトルに連想させたい値が 0 である場合，X に特徴ベクトルを登録しておき，連想させたい値が 1 である場合，Y に特徴ベクトルを登録しておく．これによりある特徴ベクトルを認識させた時，X に存在すれば，その特徴ベクトルから連想される値は 0 である可能性が高く，また，Y に存在すれば，連想される値は 1 である可能性が高いとすることができる．

Bloomier Filter を用いてより多くの数字を連想させる場合は，連想させたい値を 2bit 表現し，各 bit ごとに二つの Bloom Filter を用意するれば良い．本研究では，この Bloomier Filter の動作を利用し，特定物体を認識するために用いる物体 ID を連想させることを考える．これにより，R 個の物体を識別するために，必要な Bloom Filter は，$\log_2 R$ 個で済む．

4. 提 案 手 法

本節では，Bloomier Filter を用いた特定物体認識手法について述べる．提案手法でも，従来手法と同様に画像から PCA-SIFT によって求めた特徴ベクトルを用いる．以下に具体的な処理について説明する．

4.1 データベース作成

本節では，Bloomier Filter へ特徴ベクトルを登録するデータベース作成処理について説明する．今物体 ID を n bit で表現することを考える．このために $2n$ 個の Bloom Filter を用意する．ここで 0 を連想させる Bloom Filter を $X_1, X_2, \ldots, X_n$，1 を連想させる Bloom Filter を $Y_1, Y_2, \ldots, Y_n$ とする．それぞれの Bloom Filter の TableSize $M_f^g (f \in \{0, 1, \ldots, n\}, g \in \{0, 1\})$ は，

$$M_f^g = a \times N_f^g \quad [\text{bit}] \tag{4}$$

とする．ここで，N_f^g は，物体 ID の f 番目の bit が g である全ての物体から得られる特徴ベクトルの総数であり，a は定数である．予備実験より，a が小さいほどメモリ使用量を削減できるが，認識率が低下するということが分かっている．

次に，物体 ID を連想させるために，n 個の Bloom Filter に特徴ベクトルを登録していく．例えば，2bit で物体 ID を表現するとき，物体 ID の 3 を “10”で表現したとすると，bit 列の 1 bit 目が “1”，2 bit 目が “0”であるため，Y_1 と X_2 の Bloom Filter に物体 ID 3 の物体から得られる特徴ベクトルを保存する．以下に具体的な処理を説明する．提案手法では従来手法と同様に，特徴ベクトルの d 次元を用いて，ビットベクトル $\boldsymbol{u}$ を作成する．このとき，同じビットベクトルに変換される特徴ベクトルの数が閾値 c 個以上あれば，その特徴ベクトルは，認識にはあまり有効ではないと考え，データベースに登録しない．それ以外の場合は，求めたビットベクトル $\boldsymbol{u}$ をキーとして，k 個のハッシュ関数を用いて，Bloom Filter のどのビットを 1 にするかを決める．提案手法では，$k = 8$ とし，ハッシュ関数に [10] で紹介されている 8 個のハッシュ関数を用いる．以上の処理を全ての特徴ベクトルに対して行い，データベースを作成する．

提案手法では，上記の処理で作成したデータベースに加え，投票時に投票誤りを検出するための Bloomier Filter も作成する．これは，ある検索質問画像から得られた特徴ベクトルを用いて認識処理で求まった ID に，本当に投票を行って良いかを決めるための簡易的な誤り検出器である．この誤り検出 Bloomier Filter を以下のように作成する．まず，二つの Bloom Filter P_0, P_1 を用意する．そして，物体 ID を 2bit 表現したとき，1 の数が偶数個の物体 ID から得られる特徴ベクトルを P_0 に登録し，1 の数が奇数個の物体 ID から得られる特徴ベクトルを P_1 に登録する．それぞれの Bloom Filter の TableSize は，式 (3) と同様に与えられる．

4.2 物体認識

本節では，Bloomier Filter を用いた物体認識手法について説明する．まず，提案する物体認識手法の処理の流れを説明する．この手法では，検索質問画像から得られる特徴ベクトル $\boldsymbol{q}$ を用い，物体 ID の $i(i = 1, 2, \ldots, n)$ 番目の bit が 0 か 1 かを求めるために，X_i と Y_i の Bloom Filter に q に対応する特徴ベクトルが登録されているかを調べる．このとき，X_i に登録されていれば，物体 ID の i 番目の bit を 0 とし，Y_i に登録されていれば，物体 ID の i 番目の bit を 1 とする．そして，全ての bit に対して値を求め，最終的に求まった物体 ID の物体に投票を行う．但し投票は，誤り検出 Bloomier Filter で正しいと判断された物体 ID の物体にのみ行われる．この処理を検索質問画像から得られる特徴ベクトル全てに対して行い，最大得票数の物体を認識結果とする．以下に具体的な処理について説明する．

質問画像から得られる特徴ベクトルは，一般に各次元の値がデータベース作成に用いる特徴ベクトルのものと異なったものとなる．このため提案手法でも，従来手法と同様に，各次元の値の許容変動幅 e を設け変動に対処する．また，この処理を行う次元数を閾値 b 個以下に制限し，次元数が b を上回る時には，次元のインデックスの大きいものから b 個を採用する．そしてこれらのビットベクトルを用いて，物体 ID の $i(i = 1, 2, \ldots, n)$ 番目の bit が 0 か 1 かを求める．このとき，Bloom Filter X_i に存在していれば，物体 ID の i 番目の bit を 0 とし，Bloom Filter Y_i に存在していれば，物体 ID の i 番目の bit を 1 とする．X_i と Y_i の両方に含まれていない場合は，このビットベクトルをキーとした特徴ベクトルはデータベースに登録されていないとし，i 番目以降の処理を打ち切る．この処理では，X_i と Y_i の両方に含まれていた場合に問題が起こる．これは，どちらかの Bloom Filter が偽陽性を引き起こし，実際には登録されていないにもかかわらず，登録されていると判断されるためである．提案手法では，このような場合両方の可能性を試すことにより対処する．具体的には，i 番目の bit が 0 となる物体 ID と 1 となる物体 ID の両方を投票の対象とする．但し，全ての bit に対して上記の処理を行うと誤投票が増加してしまうため，提案手法では，偽陽性に対処する物体 ID の次元数に閾値 t を設け，投票する物体 ID 数を 2^t 個に制限する．

図 2　3 次元物体の例

上記の処理を繰り返し，求まった物体 ID の物体に投票を行うかどうかを，誤り検出 Bloomier Filter を用いて決める．このとき，誤り検出 Bloomier Filter に登録されていなかった場合や，P_0, P_1 の両方に登録されていた場合，求まった物体 ID は誤った ID と判断し，投票を行わない．また，提案手法でも従来手法と同様に，投票には重みを付けて行う．提案手法で用いる重みも，従来手法と同様のものを用いる．この処理を検索質問画像から得られる全ての特徴ベクトルに対して行い，最終的に最も得票数の多かった物体を認識結果とする．

また提案手法では，認識処理の効率化を図るために，従来手法と同様に，第 s 段が $b = s - 1$ とした識別器を用いて認識を行う．これによって，十分に他の物体との得票数差が得られれば，途中の段で処理を打ち切ることができるため，認識処理を効率的に行うことができる．

5. 実　　験

本節では，提案手法の有効性を確かめるために，55 個の 3 次元物体を用いたデータセットと，10,000 個の 2 次元物体を用いたデータセットに対して行った実験の結果について述べる．

5.1 実験条件

まず，55 個の 3 次元物体のデータセットについて説明する．図 2 に本実験で用いた 3 次元物体の例を示す．本実験では，物体に対して，真正面，真正面から上へ 15°，30° の角度から，ウェブカメラで，5° ごとに物体を回転させて撮影した画像を用いた．この内，データベース作成用画像として，回転角度が 0°, 10°, . . . , 350° の画像 36 枚 ×3 方向の，1 物体当たり 108 枚を用い，残りの画像を検索質問画像とした．データベース作成用

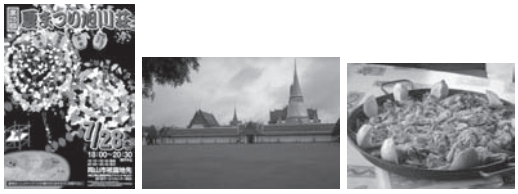

図 3　2 次元物体の例

(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 4　検索質問の例

画像から得られる特徴ベクトルの数は，約 120 万個である．

次に，10,000 個の 2 次元物体のデータセットについて説明する．本実験では，Google イメージ検索や，写真共有サイト Flickr から収集した画像 10,000 枚をデータベース作成用画像とした．図 3 に例を示す．データベース作成用画像から得られる特徴ベクトルの数は，約 2 千万個である．検索質問画像としては，次のように用意した．まず，データベースに含まれる画像のから合計 500 枚を無作為に選択した．次に，プリンタを用いて印刷したものをカメラで撮影した．撮影した画像の例を図 4 に示す．図 4 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．検索質問の数は，$500 \times 4 = 2,000$ 枚である．

本実験では，提案手法と従来手法で用いられているパラメータ b, c を変化させて物体の認識率を調べた．各パラメータの変動幅は，$b = 0, 1, \ldots, 10$，$c = 1, 2, \ldots, 10$ である．また両手法において，$e = 200$ とした．TableSize 作成に用いる定数 a は，$a = 8$ とした．これは，予備実験より認識率が極端に低下しない最低数である．実験結果で示す処理時間とは，一枚の画像の認識にかかった平均の処理時間であり，特徴ベクトルの抽出時間等は含まれていない．使用した計算機は，CPU が AMD Opteron8378 2.4GHz，メモリが 128GB のものである．

5.2　実験 1：55 個の 3 次元物体を用いた実験

提案手法において，前述のパラメータ以外に，パラメータ t も変化させて物体の認識率を調べた．t の変動幅は，$t = 0, 1, \ldots, 5$ である．また, [2] で用いられている $d = 28$ で両手法を比較した．認識率と処理時間との関係を図 5 に示す．横軸に認識率，縦軸に平均処理時間を示す．結果より，提案手法は従来法と比べて，同じ認識率を得るのに長い処理時間が必要となることが

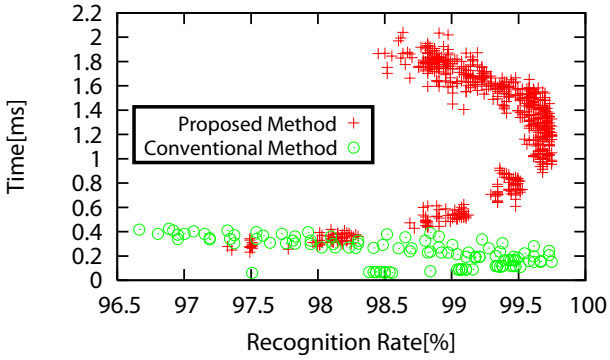

図 5　3 次元物体認識の認識率と処理時間

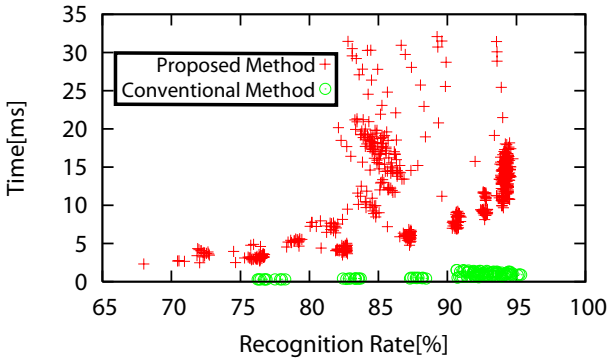

図 6　2 次元物体の認識率と処理時間

分かった．処理時間がかかる原因として，従来手法では，一つの検索質問ベクトル $\boldsymbol{q}$ に対してハッシュ表を一度検索するだけで投票を行えるが，提案手法では，$2n$ 個の Bloom Filter を検索しなければ投票を行えないことが挙げられる．

次に，メモリ使用量に関与するパラメータを変化させた時の，両手法のメモリ使用量，認識率，処理時間を調べた．提案手法において，メモリ使用量に関与するパラメータはリスト長の閾値 c，従来手法では，c 及びビットベクトルの次元数 d である．表 1 に両手法で最も認識率の高かった結果を示す．結果より，提案手法は従来手法と比べ，認識率を同程度にした時，処理時間は長いものの，メモリ使用量を削減できることが分かった．これは，従来手法では認識率を確保するために，大部分が何も記録されないにもかかわらず，特徴ベクトルの保存に大きなハッシュ表を用意しなければならないが，提案手法では，Bloomier Filter を用いることで，小容量のメモリ量で特徴ベクトルを保存できるからである．このことから，提案手法の空間効率の良さが明らかとなった．

5.3　実験 2：10,000 個の 2 次元物体を用いた実験

実験 1 と同様にパラメータを変化させて物体の認識率を調べた．パラメータの変動幅は実験 1 と同じである．実験 2 でも $d = 28$ で，両手法を比較した．認識率と処理時間の関係を図 6 に示す．図の見方は，図 5 と同様である．結果は，実験 1 と同様のものとなった．また，実験 1 の結果と比較すると，特徴ベクトル数が約 20 倍となったために，処理時間も約 20 倍になっていることが分かる．このため，特徴ベクトル数が増加しても，処理時間が増加しない枠組みを導入する必要がある．

表 1　3 次元物体認識の認識率とメモリ使用量と処理時間

手法	c	d	その他パラメータ	認識率 [%]	メモリ使用量 [MB]	処理時間 [ms]
提案手法	5	28	$b=3, e=200, t=1$	99.75	73	0.95
従来手法	3	24	$b=3, e=200$	99.83	231	0.22
	2	28	$b=3, e=200$	99.75	2199	0.32

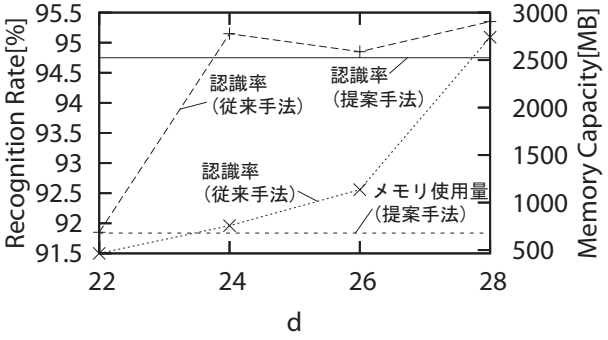

図 7　2 次元物体認識の認識率とメモリ使用量

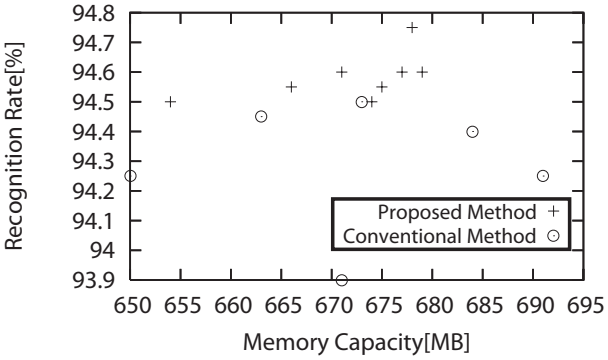

図 8　2 次元物体認識のメモリ使用量と認識率

次に従来手法において，メモリ使用量に関与するパラメータの内，d を変化させた時の，両手法のメモリ使用量，認識率を調べた．提案手法は，$d=28$ の時，最も認識率の高かったパラメータ $b=9, c=9, t=1$ を用いて実験を行った．また従来手法は，d 毎に最も認識率の高かったパラメータを用いて実験を行った．図 7 に実験結果を示す．結果より，$d=24$ の時，両手法の性能が同程度のものとなることが分かった．予備実験より，両手法の傾向として d の値を大きくすれば認識率が向上するため，さらに d の値を大きくすれば，認識率の向上が見込まれる．このとき，従来手法では d を大きくすると指数関数的にメモリ使用量が増加するが，提案手法ではメモリ使用量が増加しないため，提案手法の方が空間効率が良いと考えられる．d を大きくしたときの認識率とメモリ使用量の関係は，今後の課題である．

最後に，同程度のメモリが使える状態で，提案手法と従来手法とで認識率を比較した．従来手法のパラメータ c, d を変化させることで，提案手法と同程度のメモリ使用量とする．メモリ量と認識率の関係を図 8 に示す．図には，そのメモリ使用量において最も認識率の高かったものを示している．結果より，メモリ使用量を同程度にしたとき，提案手法の方が，従来手法と比べて，認識率が高いことが分かった．したがって，提案手法の方が従来手法と比べ，効率よく特徴ベクトル表現を圧縮できていると言える．

6. おわりに

本稿では，空間効率の良い Bloomier Filter を用いることで，特徴ベクトルを圧縮表現し，特定物体認識に必要なメモリ使用量を削減する手法を提案した．二種類のデータセットを用いた実験の結果，従来の手法と比べ，認識率が同程度の時，メモリ使用量を削減できることを示した．このことから，同じ認識率を得るとき，処理時間が長くてもメモリ使用量を削減したい場合は，提案手法の方が有効であると言える．

今後の課題として，更なる大規模なデータセットで実験，ビットベクトルの次元数 d を増加させての実験，Similarity Hashing の導入，提案手法の理論的解析といったことが挙げられる．

謝辞

本研究の一部は科学研究費補助金基盤研究 (B)(19300062) の補助による．

文　献

[1] D. Lowe: “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision, Vol. 60, No. 2, pp. 91–110 (2004).

[2] 野口和人, 黄瀬浩一, 岩村雅一：“近似最近傍探索の多段階化による物体の高速認識”, 画像の認識・理解シンポジウム (MIRU2007) 論文集，OS-B2-02, pp. 111–118 (2007).

[3] D. Nistér and H. Stewénius: “Scalable Recognition with a Vocabulary Tree”, Proc. CVPR2006, pp. 775–781 (2006).

[4] 本道貴行, 黄瀬浩一：“特定物体認識のためのデータベース容量削減法の検討〜局所特徴量の量子化と取捨選択〜”, 電子情報通信学会技術研究報告，Vol. 108，No. 484，PRMU2008-265, pp. 171–176 (2009).

[5] K. Kise, K. Noguchi and M. Iwamura: “Memory Efficient Recognition of Specific Objects with Local Features”, Proc. of the 19th International Conference of Pattern Recognition (ICPR2008) WeAT3.1 (2008).

[6] 井上勝文, 三宅弘志, 黄瀬浩一：“局所記述子に基づく 3 次元物体認識のためのメモリ削減—局所記述子の取捨選択によるアプローチ—”, 画像の認識・理解シンポジウム (MIRU2008) 論文集，OS15-3, pp. 363–370 (2008).

[7] B. Chazelle, J. Kilian, R. Rubinfeld and A. Tal: “The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Table”, Proc. 15th Annual ACM-SIAM SODA, pp. 30–39 (2004).

[8] Y. Ke and R. Sukthankar: “PCA-SIFT:A more distinctive representation for local image descriptors”, Proc. of CVPR2004, Vol. 2, pp. 506–513 (2004).

[9] B. H. Bloom: “Space/Time Trade-offs in Hash Coding with Allowable Errors”, Commun. ACM, Vol. 13, No. 7, pp. 422–426 (1970).

[10] General Purpose Hash Function Algorithms: http://www.partow.net/programming/hashfunctions/index.html#RSHashFunction.