4P303-15

大規模特定物体認識における Bloomier Filter を用いたメモリ削減法の有効性の検証

◎井上 勝文 †,††　黄瀬 浩一 †

† 大阪府立大学大学院 工学研究科　†† 日本学術振興会

inoue@m.cs.osakafu-u.ac.jp kise@cs.osakafu-u.ac.jp

1 はじめに

近年，Android 携帯や iPhone 等に代表される高性能携帯電話の普及により，これらの機器を対象とした様々なサービスが提供されている．その一つに，これらに搭載されているカメラを用いた情報検索サービスがある．例えば “Google Goggles” [1] では，ユーザがある物体に関して十分な知識を有していなくても，ユーザがその物体をカメラで撮影することにより，インターネットから有益な情報を得ることができる．このようなサービスを実現している技術として，特定物体認識がある．本研究では，その中でも特に，物体認識の分野で盛んに研究され，高精度な物体認識を実現してる局所特徴量を用いた特定物体認識に着目する．

PCA-SIFT [2] に代表される局所特徴量を用いた従来の特定物体認識手法には，多様な条件で撮影された物体の画像から多数の局所特徴量を抽出し，それらを全てデータベースに登録するものがある．この手法では，質問画像から得られる局所特徴量と，データベースに登録されている局所特徴量の照合によって物体を認識する．一般的に，局所特徴量は多次元実数ベクトルで表現され，照合には，最近傍探索が用いられる．この単純な手法の利点は，データベース作成が容易で，高精度な認識を行うことができることである．しかし，認識に関与する特徴ベクトルの数が膨大となるため，必要なメモリ容量が莫大となる問題点がある．

この問題を解決する一つのアプローチは，ハッシュ表などを用い，特徴ベクトルの有無のみを記録することである [3]．このアプローチでは，特徴ベクトルを距離のような量的な概念に基づく類似検索ではなく，ハッシュ値が同じかどうかという一致検索によって照合する．これにより，距離計算に必要な特徴ベクトルの情報を記録する必要がなくなり，大幅なメモリ削減が可能となる．しかし，この手法にもまだ問題がある．特徴ベクトルの識別性を維持するには，広大なハッシュ表が必要となる．ところが，広大なハッシュ表を用いると，特徴ベクトルの識別能力は維持できる一方で，ハッシュ表の大半に何も登録されなくなる．このことから，ハッシュ表には，空間効率が悪いという問題点がある [4]．

この問題を解決するために，我々はこれまでに，ハッシュ表の代わりに Bloomier Filter [5] と呼ばれる空間効率の良い確率的データ構造を用いる手法 [4] を提案している．Bloomier Filter とは，登録されている要素に関連する値を連想させることのできる連想配列である．提案手法では，この Bloomier Filter を用い，特徴ベクトルから物体 ID を連想させることで物体を認識する．これまでに，55 個の 3 次元物体という小規模なデータベースを対象とした実験により，ハッシュ表を用いる手法と比較してメモリ使用量を大幅に削減できることを示している [4]．しかし，[4] では，提案手法のスケーラビリティについて十分な議論がなされておらず，データベースを拡張した場合に，認識率・処理時間・メモリ使用量に生じる影響を検証する必要がある．そこで本研究では，400 個の 3 次元物体データベースを用いて，提案手法のスケーラビリティについて検証する．

2 Bloom Filter と Bloomier Filter

本節では，提案手法に用いる Bloomier Filter [5] と，その基礎となる Bloom Filter [6] について述べる．

2.1 Bloom Filter

Bloom Filter は，m bit のビット列で構成され，ハッシュ表よりも空間効率の良い確率的データ構造である．ここで，m を Bloom Filter の “TableSize” と呼ぶ．Bloom Filter は，ある要素がデータ集合のメンバであるかどうかを記録するのに用いられる．但し，Bloom Filter には，集合のメンバでない要素が，集合のメンバであると判断される偽陽性が生じる可能性があるという問題点がある．本稿では，以後，データ集合の要素を特徴ベクトル，データ集合を物体データベースとして話を進める．

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，全ての bit を 0 に初期化した m bit のビット列を用意する．次に，特徴ベクトルをハッ

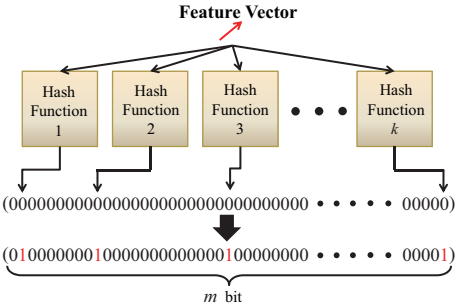

図 1: Bloom Filter への特徴ベクトル登録処理の流れ

シュキーとし，k 個のハッシュ関数からそれぞれハッシュ値を得る．そして，Bloom Filter の $h_i(i = 1, 2, \ldots, k)$ 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，その特徴ベクトルはデータベースに含まれていると判断する．

2.2 Bloomier Filter

Bloomier Filter は，複数の Bloom Filter を組み合わせた構成となっており，データベースに登録されている特徴ベクトルに関連する値を連想することができる．以下に，Bloomier Filter の動作について説明する．

説明の簡単化のため，Bloomier Filter で連想させる値が 0 と 1 の二種類の場合について説明する．まず，二つの Bloom Filter $B^{(0)}$ と $B^{(1)}$ を用意する．次に，特徴ベクトルに連想させる値が 0 (1) の場合，$B^{(0)}(B^{(1)})$ に特徴ベクトルを登録する．推定処理では，ある未知の特徴ベクトルが，$B^{(0)}(B^{(1)})$ に登録されていると判断された場合，その特徴ベクトルに関連する値は，0 (1) の可能性が高いと言える．

ある値を二進数表現し，桁ごとに二つの Bloom Filter を用意すれば，Bloomier Filter に任意の値を連想させることができる．本研究では，特徴ベクトルから物体 ID を連想させるために Bloomier Filter を用いる．Bloomier Filter を上記のように構成すると，N 物体の識別に必要な Bloom Filter の数は，$2\log_2 N$ 個で済む．

3 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．提案手法では，局所特徴量として PCA-SIFT [2] で求めた 36 次元の特徴ベクトルを用いる．

3.1 データベース作成処理

本節では，データベース作成処理について詳しく述べる．まず，PCA-SIFT で求めた特徴ベクトル $\boldsymbol{p}$ の第 1 次元から d 次元 $(d \leq 36)$ を用いて

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

によって 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．ここで，θ_j は，認識対象となる画像全ての p_j の中央値である．そして，物体 ID を ν 桁の二進数表現に変換し，2ν 個の Bloom Filter を用意する．ここで，このように二進数表現された物体 ID を，以後 “*bit ID*” と呼ぶ．また，0 を連想させる Bloom Filter を $B_1^{(0)}, B_2^{(0)}, \ldots, B_\nu^{(0)}$，1 を連想させる Bloom Filter を $B_1^{(1)}, B_2^{(1)}, \ldots, B_\nu^{(1)}$ とする．それぞれの Bloom Filter の TableSize は以下の式で与えられる．

$$m_\kappa^\tau = a \times n_\kappa^\tau + \alpha_\kappa^\tau \tag{2}$$

ここで，$m_\kappa^\tau, n_\kappa^\tau (\kappa \in \{1, 2, \ldots, \nu\}, \tau \in \{0, 1\})$ は，それぞれ *bit ID* の κ 桁が τ の Bloom Filter の TableSize と，そのような *bit ID* を持つ物体から得られる特徴ベクトルの総数を意味する．また α_κ^τ は，TableSize が $a \times n_\kappa^\tau$ より大きな最小の素数となるための定数であり，a は，TableSize の大きさを左右するパラメータである．

次に，ν 個の Bloom Filter にビットベクトルを登録する．例えば、2 桁の二進数表現で物体 ID を表現する場合を考える．このとき，物体 ID 2 を “10” と表現したとすると，1 桁目 (2 桁目) が “0” (“1”) であるため，$B_1^{(0)}(B_2^{(1)})$ にそれぞれ物体 ID 2 の物体から得られるビットベクトルを登録する．提案手法では，Bloom Filter への登録に用いるハッシュ関数に MD5 を用い，得られる 128bit のビット列を 4 等分 $(k = 4)$ したものをハッシュ値とする．

提案手法では，Bloom Filter で偽陽性が発生し，正しい物体 ID が得られない場合に対処するために，パリティビットを連想させる Bloomier Filter も作成する．まず，二つの Bloom Filter $P^{(0)}, P^{(1)}$ を用意する．そして，*bit ID* において，1 の数が偶数個 (奇数個) の物体 ID から得られる特徴ベクトルを $P^{(0)}(P^{(1)})$ に登録する．それぞれの TableSize は，式 (2) と同様に得られる．

3.2 認識処理

提案手法の認識処理の流れは，以下の通りである．まず，質問画像から得られる特徴ベクトル $\boldsymbol{q}$ を用い，どの Bloom Filter に $\boldsymbol{q}$ が登録されているかを調べ，物体 ID

図 2: 3 次元物体の例

を求める．そして求まった物体 ID の物体に対してスコアを増加させる．但し，パリティ用 Bloomier Filter で正しいと判断された物体 ID のみ認識処理に用いる．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，最大スコア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．

まず，質問画像から得られる特徴ベクトル $\boldsymbol{q}$ より，ビットベクトルを求める．そして，得られたビットベクトルを用いて，*bit ID* の $i(i = 1, 2, \ldots, \nu)$ 番目の桁が 0 か 1 かを求める．このとき，Bloom Filter $B_i^{(0)}(B_i^{(1)})$ に登録されていた場合，*bit ID* の i 番目の桁を 0(1) とする．$B_i^{(0)}$ と $B_i^{(1)}$ のいずれにも登録されていない場合，このビット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，偽陽性の影響で，$B_i^{(0)}$ と $B_i^{(1)}$ の両方に登録されていると判断される場合がある．提案手法では，両方の可能性を試すことにより，この問題に対処する．つまり，i 番目の桁が 0 となる *bit ID* と 1 となる *bit ID* の両方を認識処理に採用する．そして，求まった *bit ID* のうち，パリティ用の Bloomier Filter で正しいと判断された *bit ID* を持つ物体に対して，$s_i \leftarrow s_i + 1/\sqrt{\omega_i}$ のようにスコアを更新する．ここで，s_i は物体 ID i の物体のスコア，ω_i は物体 ID i の物体から得られる特徴ベクトルの数である．この処理を繰り返し，最終的に最もスコアが高い物体を認識結果とする．

4 実験

本研究では，提案手法の有効性を示すために，400 個の 3 次元物体を用いて実験を行った．

4.1 実験条件

本節では，400 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 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．このうち，データベース作成用画像として，回転角度が $0°, 10°, \ldots, 350°$ の画像 36 枚 ×3 方向の，1 物体あたり 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画像から得られる特徴ベクトルの総数は，約 2200 万個である．

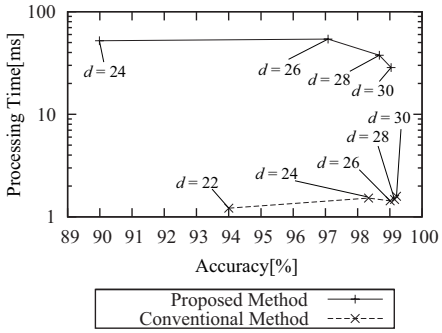

図 3: ビットベクトルの次元数 d を変化させたときの認識率と処理時間の関係

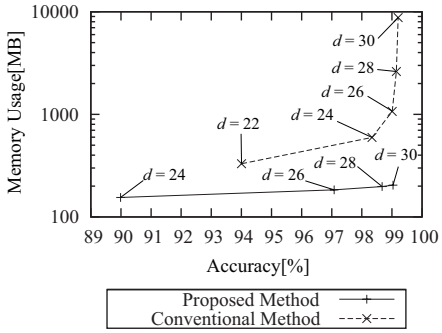

図 4: ビットベクトルの次元数 d を変化させたときの認識率とメモリ使用量の関係

本実験では，ハッシュ表に基づく手法 [3] を比較手法として用いた．この手法は，データベース作成に Bloomier Filter ではなくハッシュ表を用いる点で，提案手法と異なるが，他は同じである．実験に用いた計算機は，CPU AMD Opteron8378 2.4GHz，128GB RAM のものである．提案手法と従来手法で共通するパラメータ d は，$d = 22, 24, 26, 28, 30$ を採用した．また，従来手法で用いるハッシュ表のサイズは，[3] で採用している $H_{\text{size}} = 2^d$ とした．また提案手法では，上記のパラメータの他に，TableSize を左右するパラメータ a を用いており，$a = 8$ とした．また，以下の実験結果で示す処理時間は，1 枚の画像の認識にかかった平均の処理時間で，特徴ベクトルの抽出時間等は含まれていない．

4.2 実験結果

まず，従来手法と認識率，処理時間，メモリ使用量について比較した結果について述べる．ビットベクトルの

表 1: 物体データベースの規模の違いによる認識率，処理時間，メモリ使用量の関係

手法 (データベースの規模)	パラメータ	認識率 [%]	処理時間 [ms]	メモリ使用量 [MB]
提案手法 (55 物体)	$d = 26$	99.85	3.18	20
提案手法 (400 物体)	$d = 30$	99.63	10.88	202
従来手法 (55 物体)	$d = 24$	99.88	0.32	174
従来手法 (400 物体)	$d = 26$	99.65	0.52	1053

次元数 d を変化させた場合の最も認識率が高い結果とそのときの処理時間との関係を図 3 に，認識率とメモリ使用量の関係を図 4 にそれぞれ示す．

図 3 より，提案手法は従来手法と比較して処理時間がかかることがわかった．これは，1 つのビットベクトルから物体 ID を求めるために，従来手法ではハッシュ値を一度求めれば良いのに対し，提案手法では，2ν 個の Bloom Filter それぞれに対して，ハッシュ値を k 個求めなければならないためである．

次に，認識率とメモリ使用量の関係について着目すると，図 4 より，提案手法は従来手法と同程度の認識率を得るのに必要なメモリ使用量が少なくてすむことがわかった．特に，従来手法の結果において，提案手法で最も認識率の高かった 99.04%の結果と同程度の認識率となる結果に着目すると，提案手法は約 1/5 のメモリ使用量で認識できることがわかった．

また，提案手法のスケーラビリティについて議論するため，物体データベースの規模を変化させ，認識率，処理時間，メモリ使用量にどのような影響が生じるかを，従来手法と比較しつつ調べた．この実験では，質問画像として，55 個の 3 次元物体を前節で述べたような位置から撮影した画像のうち，回転角度が $5°, 15°, \ldots, 355°$ の画像 36 枚 $\times 3$ 方向の，1 物体あたり 108 枚を用いた．

表 1 に，物体データベースの規模を変化させた場合の結果のうち，最も認識率の高かった結果を示す．また，従来手法の結果のうち，提案手法と同程度の認識率となる結果も表 1 に示す．結果より，まず処理時間に着目すると，物体データベースの規模を大きくすると，より長い処理時間が必要であることがわかった．次に，メモリ使用量に着目すると，物体データベースの規模を大きくした場合，提案手法では認識に必要なメモリ使用量が 10 倍になった．また，従来手法の結果と比較し，提案手法の有効性が低くなった．これは，提案手法で用いる Bloom Filter の TableSize が，特徴ベクトル数に依存しているためである．このため，今後さらに物体データベースを大規模化するために，特徴ベクトル数に依存しない手法を考案する必要があろう．最後に認識率に着目すると，提案手法では，物体データベースの規模を大きくしても，ほとんど認識率を落とすことなく認識できることがわかった．このことから，処理時間やメモリ使用量が許容できるのであれば，認識率の点において提案手法はスケーラビリティがあると言える．

5 おわりに

本稿では，Bloomier Filter を用いた特徴ベクトルの一致検索に基づく特定物体認識手法を提案した．400 個の 3 次元物体を用いた実験より，ハッシュ表に基づく手法と比較し，処理時間では劣るものの，同程度の認識率を得るのに必要なメモリ使用量を約 1/5 にすることができた．また，物体データベースの規模を変化させた実験より，提案手法のスケーラビリティについて確認した．

今後の課題として，さらなる物体データベースの大規模化や，処理の高速化，Bloomier Filter と認識率・処理時間・メモリ使用量の関係を理論的に解析するといったことが挙げられる．

謝辞 本研究の一部は，日本学術振興会科学研究費補助金 基盤研究 (B) (22300062)，特別研究員奨励費 (22・8970) の補助による．

参考文献

[1] “http://www.google.com/mobile/goggles/”.

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

[3] 野口和人, 黄瀬浩一, 岩村雅一：“大規模特定物体認識における認識率，処理時間，メモリ量のバランスに関する実験的検討”, 電子情報通信学会論文誌 D, **J92-D**, 8, pp. 1135–1143 (2009).

[4] 井上勝文, 黄瀬浩一：“特定物体認識における Bloomier Filter を用いたメモリ削減法とその実験的評価”, 電子情報通信学会論文誌 D, **J93-D**, 8, pp. 1407–1416 (2010).

[5] 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).

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