Vol.2013-CG-153 No.19
Vol.2013-CVIM-189 No.19
2013/11/29

対応点探索のための二値特徴量

安倍　満[1,a)]

概要：局所特徴量を二値ベクトルで表現する手法について紹介する．これは，キーポイントの特徴量記述を実数のベクトルで表現せず，数十〜数百個程度の 0 と 1 の列（すなわち二値のベクトル）で表現するというものである．この方法により，計算速度・メモリ消費量の問題が飛躍的に改善されたことから，コンピュータビジョンの研究者の幅広い関心を集めるようになった．今では，SIFT や SURF に代わる新たな局所特徴量として急速に普及しつつある．本稿では，キーポイント周辺からコンパクトな二値特徴量を抽出する手法について体系的にまとめる．また，研究目的に利用可能なソフトウェアライブラリについても紹介する．

キーワード：局所特徴量，二値特徴量，バイナリコード

1. はじめに

局所特徴量は，画像間のマッチングを可能とする強力なツールであり，物体認識，三次元復元，画像検索など，様々なアプリケーションの飛躍的な進歩に貢献してきた．局所特徴量は，キーポイント検出と特徴量記述という二段階の処理からなる．キーポイント検出と特徴量記述の両方を，幾何学的変化（回転，スケール，アフィン変形）や照明変化に対して頑健になるように設計する点がポイントであり，Scale-Invariant Feature Transform (SIFT)[16] をはじめとして，数多くの局所特徴量が提案されてきた．例えば，特徴量の記述性能の改善を試みた手法 [14], [17] や，特徴量の抽出を高速化する手法 [6] などが提案されている．

特に近年では，特徴量の記述に必要なデータサイズをできるだけ小さくすることで，局所特徴量の消費メモリ量を減らし，かつ局所特徴量間のマッチング速度を高速化しようと試みる動きがある．その 1 つのアプローチが，本稿のテーマとして掲げる「二値特徴量[*1]」である．二値特徴量は処理負荷が軽く，モバイル端末などでもリアルタイム処理が可能であることから，SIFT に代わり得る手法として有望視されている．

1.1 実数特徴量の欠点

SIFT[16]，SURF[6] など，従来の手法では，キーポイント周辺の領域を高次元の実数ベクトルで表現していた．そのため，メモリ消費量が多く，また特徴量同士の距離の計算が遅いという 2 つのデメリットがあった．

(1) メモリ消費量

例えば SIFT の場合，局所特徴量は 128 次元の実数ベクトルで表現される．これを適切にスカラー量子化し，最も小さなデータ型である unsigned char で格納したとしても，1 つのキーポイントあたり 128byte を消費する．これがキーポイントの数だけ必要であるから，仮に画像から 1,000 点のキーポイントが検出されたとすると，$1,000 \times 128\text{byte} \approx 125\text{Kbyte}$ のメモリを消費することになる．扱う画像枚数が増えれば増えるほど，メモリ消費量の問題は深刻となる．

(2) 距離計算の速度

2 つの特徴量 $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D$ の距離は，次の (1)(2) 式に示すユークリッド距離 d_{E} やベクトル間角度 d_θ で表せる．

$$d_{\mathrm{E}}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^\top (\mathbf{x}_1 - \mathbf{x}_2)} \tag{1}$$

$$d_\theta(\mathbf{x}_1, \mathbf{x}_2) = \arccos\left(\frac{\mathbf{x}_1^\top \mathbf{x}_2}{|\mathbf{x}_1||\mathbf{x}_2|}\right) \tag{2}$$

例えばユークリッド距離の場合，D 回の引き算，D 回の掛け算，$D-1$ 回の足し算，および 1 回の平方根の演算が必要である．SIFT をはじめとする多くの局所特徴量は次元数 D が大きいため，距離計算の負荷が高くなりやすい．

[1] 株式会社デンソーアイティーラボラトリ
Shibuya CROSS TOWER 28th Floor 2-15-1 Shibuya Shibuya-ku Tokyo, 150-0002 Japan
[a)] manbai@d-itlab.co.jp

[*1] 英語では, Binary Features, Binary Descriptors, Binary Codes などの表記がこれに対応する．

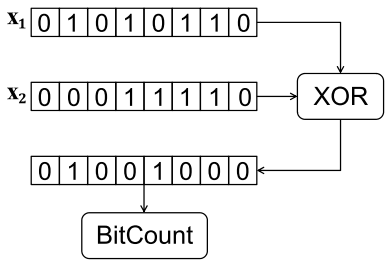


図 1 ハミング距離の計算

近年では，大規模な画像検索などの応用において数百〜数千万個の局所特徴量をメモリ上に格納して類似度計算を行うことも珍しく無い．そのため，メモリ消費量と距離計算の速度の 2 点における改善が求められている．

1.2 二値特徴量の利点

これに対し，キーポイントの特徴記述を実数ベクトルで表現せず，二値のベクトルで表現するという手法が注目されている．これが本稿で紹介する二値特徴量である．B 次元の二値特徴量 $\mathbf{y}$ は，

$$\mathbf{y} \in \{-1, +1\}^B \tag{3}$$

もしくは

$$\mathbf{y} \in \{0, +1\}^B \tag{4}$$

として定義される．つまり，$\mathbf{y}$ の各要素は，二通りの整数のうち，いずれか一方の値のみを持つ．(3) 式を用いるか，(4) 式を用いるかは各提案手法によって異なるが，単に数学的な定式化において，都合の良い方が選ばれているだけであり，これらの間に本質的な違いは無い（例えば，後述する Binary Hashing による手法の場合は，(3) 式で定義されることが多い）．ポイントは，(3)(4) 式のいずれの場合においても，$\mathbf{y}$ を B ビットの 0 と 1 の列でメモリ上に格納できるという点である[*2]．そのため，メモリの消費量を大幅に減らせるというメリットがある．

また，もう一つのメリットは，二値特徴量間の距離をハミング距離で測れるという点である．ハミング距離とは，二つの同じ長さのビット列が与えられたとき，対応する位置において一致していないビットの個数に相当する．これは，2 つのビット列の XOR を計算し，1 が立っているビットの数を数えるだけで得られるため，極めて高速に計算可能である．図 1 に例を示しておく．現在，一般に利用されている CPU では，ビットカウント用の専用命令を実装しているものも多く，これを用いることでさらなる高速化も可能である．

1.3 研究動向

特徴量を二値のベクトルで表現できれば，メモリ消費量の削減と距離計算の高速化という二つの利点が得られる．そこで近年では，局所特徴量の記述性能を十分に保ったまま，如何にして二値で記述をするか，という点について議論が深まっている．図 2 は，その研究の動向を示したものである．対応点探索のための二値特徴量を抽出する手法は，次の 2 つに大別できる．

(1) 直接的に二値特徴量を生成する方法
　実数で記述された特徴量を介さず，二値の特徴ベクトルを直接求める．
(2) 間接的に二値特徴量を生成する方法
　従来の手法で実数の局所特徴量を算出し，これを二値特徴量に変換する．

直接／間接いずれのアプローチでも，初めは機械学習に基づかない単純な方法が検討され，その後に教師なし学習によって記述性能が改善できることが示された．特に近年では，教師あり学習によりさらなる記述能力の向上が見込めることが報告されており，（実験条件にもよるが）64bits の二値特徴量で SIFT に匹敵する性能が達成された事例も報告されている [25].

1.4 本論文の構成

以下，2 節では，直接的に二値特徴量を生成する方法について，3 節では，間接的に二値特徴量を生成する方法について，それぞれ解説する．4 節では，2，3 節で解説した手法について，体系的にまとめる．5 節では，前節までに紹介した手法を利用するためのソフトウェアライブラリ，実装方法，研究を進めるうえで有用な評価ツールやデータセットについて紹介する．

なお，数学的な記法の整合性をとるため，原著論文で用いられている記法を部分的に変更した．ただし，あまり変え過ぎると原著論文を参考する際に混乱するため，変更は必要最小限にとどめた．

2. 直接的に二値特徴量を生成する方法

2010 年に，機械学習によらない単純な手法として，BRIEF[9] が提案された．これはピクセル間の輝度差の情報から二値特徴量を生成するというものであり，アルゴリズムが非常にシンプルでありながらもそれなりの性能を発揮したため，注目を集めた．この考え方は後続の BRISK[15] や ORB[20] に受け継がれ，スケール・回転不変性に関する検討や，特徴量記述能力の改善が試みられた．ORB は OpenCV を開発している Willow Garage の研究であり，いち早く OpenCV に実装されたことから，発表直後から広く使われるようになった．ORB や FREAK[3] で教師なし学習の概念が取り入れられ，機械学習により特徴量記述性能が改善できることが示された．その後に提案された D-BRIEF[26] や BinBoost[25] では，教師あり学習に関する検討が行われた．これは，パッチの positive ペアと

[*2] (3) 式の定義に従う場合でも，-1 を 0 とみなしてメモリ上に配置すればよい．

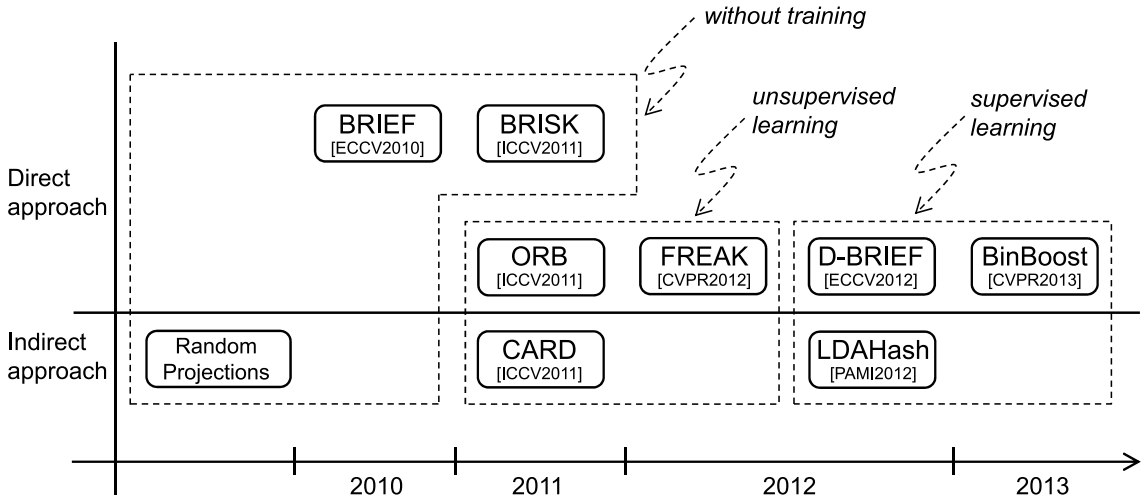


図 2 対応点探索のための二値特徴量の研究動向

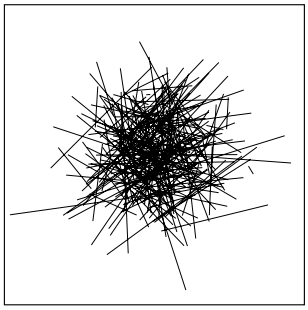


図 3 BRIEF のサンプリングパターン

negative ペアを収集し，positive ペア同士のハミング距離が小さく，negative ペア同士のハミング距離が大きくなるように学習をすることで，より記述能力の高い特徴抽出を狙うというものである．本節では，以上に紹介した直接的に二値特徴量を生成する方法を，提案された順に沿って紹介する．

2.1 BRIEF: Binary Robust Independent Elementary Features

Calonder ら [9] によって提案された BRIEF は，キーポイント周辺のパッチ内においてランダムに選択された 2 点の輝度差の符号から二値特徴量を生成するというシンプルなアイディアに基づいた特徴記述手法である．パッチ内からランダムに選ばれた 2 点を $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^2$ と記述する．これを B 組用意しておく．すなわち $i = 1, \cdots, B$ である．それぞれの点における輝度を $I(\mathbf{u}_i), I(\mathbf{v}_i)$ と記述する．このとき，輝度差 $I(\mathbf{u}_i) - I(\mathbf{v}_i)$ が正であれば 1, 負であれば 0 を割り当てることで，B bits の二値特徴量を生成する．$\mathbf{u}_i, \mathbf{v}_i$ のペアの選び方には様々な方針が考えられるが，著者らは単純にキーポイントの位置を中心としたガウス分布に基づいてランダムに選択するのが良いという実験結果を示している．これを図 3 に示した．なお，ノイズに対する耐性を高めるため，パッチにはあらかじめガウシアンフィルタを適用してスムージングをかけておく．このように，2 点間の輝度差を所望のビット数分計算するだけで済むため，高速に特徴抽出を行える．

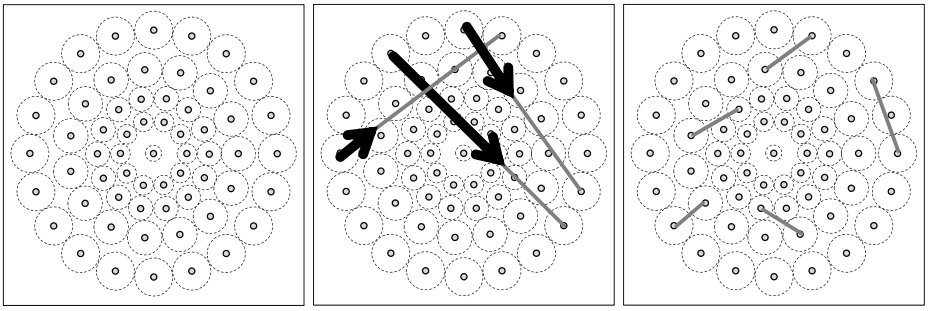
ただし，BRIEF のアルゴリズム自体にはスケール・回転不変性が無いため，これらの不変性が必要である場合は，SIFT や SURF によるキーポイント検出と組み合わせるなどといった工夫が必要である．このような不完全さがありながらも，2 点の輝度差で二値特徴量を生成し，ハミング距離でマッチングを行えば対応点探索が可能という実験事実は興味深いものである．この考え方は，後続の BRISK や ORB でも受け継がれている．

2.2 BRISK: Binary Robust Invariant Scalable Keypoints

Leutenegger ら [15] によって提案された BRISK は，離れた 2 点の輝度差に着目するという BRIEF の考え方にスケール不変性と回転不変性を導入したものである．

まずキーポイント検出について概略を述べる．スケール不変性を獲得するために，入力画像を多段階に縮小したピラミッド画像を作成する．それぞれのレイヤに対して FAST[19] を適用し，全ての画素に関して特徴点らしさのレスポンス値を計算しておく．この値がしきい値以上であり，かつ方向およびスケール方向の 3 次元空間において局所最大となる点をキーポイントとして抽出する．キーポイントの座標およびスケールは，レスポンス値に対して 2 次の多項式フィッティングを適用することで高精度に求めている．

次に，特徴量記述について述べる．BRISK では，パッチに配置された 4 つの同心円上において，等間隔にサンプリングされた 60 箇所の輝度値を用いる（図 4(a)）．BRIEF はランダムサンプリングに基づいているため，ビット数 × 2 回分の画素値のアクセスが必要だが，BRISK では規則的に並んだ 60 点の画素値のみが分かれば良い．各サン



(a) サンプリング位置　(b) 長距離ペア $\mathcal{L}$　(c) 短距離ペア $\mathcal{S}$

図 4 BRISK のサンプリングパターン

プリング点における輝度は，中心からの距離に比例する分散を持つガウスフィルタによりスムージングをかけることでノイズ耐性を獲得している．図 4(a) では，この分散の値を点線の丸で示した．以下，60 個のサンプリング点を $\mathbf{u}_i \in \mathbb{R}^2 \ \ (i = 1, \cdots, 60)$ と記述する．

BRISK では，サンプリング点間の距離が $\delta_{\min}$ 以上であるペアの集合 $\mathcal{L}$（長距離ペア）を用いてオリエンテーションの推定を行う．また，サンプリング点間の距離が $\delta_{\max}$ 以下であるペアの集合 $\mathcal{S}$（短距離ペア）を用いて特徴量の記述を行う．図 4(b)(c) に $\mathcal{L}, \mathcal{S}$ として選択されたペアの一部を示した．オリエンテーション推定に長距離 $\mathcal{L}$ を用いる理由は，パッチの大局的な勾配方向を捉えることを意図しているからである．一方，特徴量記述に短距離ペア $\mathcal{S}$ を用いるのは，パッチ内の局所的な画像特徴を捉えることを狙っているからである．

パッチのオリエンテーションは，長距離ペア $\mathcal{L}$ を用いて次のように算出される．まず，一組のペア $(\mathbf{u}_i, \mathbf{u}_j) \in \mathcal{L}$ に関して，勾配に相当する量を (5) 式で定義する．

$$\mathbf{g}(\mathbf{u}_i, \mathbf{u}_j) = (\mathbf{u}_j - \mathbf{u}_i) \cdot \frac{I(\mathbf{u}_j, \sigma_j) - I(\mathbf{u}_i, \sigma_i)}{||\mathbf{u}_j - \mathbf{u}_i||^2} \qquad (5)$$

$I(\mathbf{u}_i, \sigma_i), I(\mathbf{u}_j, \sigma_j)$ は分散 σ_i, σ_j のガウスフィルタでスムージングされた後の輝度値である．勾配 $\mathbf{g}(\mathbf{u}_i, \mathbf{u}_j)$ はペア $\mathbf{u}_j, \mathbf{u}_i$ を結ぶ直線と同じ方向を向いており，その長さはペア間の輝度差で与えられる．図 4(b) では，これを太い矢印で示した．オリエンテーション α は，集合 $\mathcal{L}$ に所属するペアの平均勾配ベクトル $\mathbf{g}$ の角度として定義される．

$$\mathbf{g} = (g_x, g_y)^\top = \frac{1}{|\mathcal{L}|} \sum_{(\mathbf{u}_i, \mathbf{u}_j) \in \mathcal{L}} \mathbf{g}(\mathbf{u}_i, \mathbf{u}_j) \qquad (6)$$

$$\alpha = \mathrm{atan2}(g_y, g_x) \qquad (7)$$

推定した α に従って $\mathbf{u}_i, \mathbf{u}_j$ を回転させた座標位置を $\mathbf{u}_i^\alpha, \mathbf{u}_j^\alpha$ とする．これを用いて，(8) 式により二値特徴量の各ビットを算出する．

$$y_{ij} = \begin{cases} 1 & I(\mathbf{u}_j^\alpha, \sigma_j) > I(\mathbf{u}_i^\alpha, \sigma_i) \\ 0 & (\text{otherwise}) \end{cases}, \forall (\mathbf{u}_i, \mathbf{u}_j) \in \mathcal{S} \qquad (8)$$

このように，特徴量記述には短距離ペア $\mathcal{S}$ を用いることで局所的な画像特徴を捉えている．$\mathcal{S}$ の要素数が 512 個になるように $\delta_{\max}$ が設定されているため，最終的に出力される二値特徴量は 512 ビットになる．

2.3 ORB: Oriented FAST and Rotated BRIEF

BRISK と同様に，ORB も BRIEF にスケールと回転不変性を導入した手法であり，2 点の輝度差を用いて二値特徴量を生成するという考え方に基づいている．キーポイント検出に FAST を用いるという点は BRISK と同じだが，オリエンテーションの推定方法と，輝度差を求める 2 点のペアの選択方法が異なっている．ここではキーポイント検出の説明は省略し，オリエンテーション推定と二値特徴量の生成方法について解説する．

ORB では，オリエンテーション α を推定するときに勾配を用いない．パッチの輝度値に関する 0,1 次モーメントから重心 (intensity centroid, C) を計算し，キーポイント中心 O と重心 C を結ぶ直線の傾きを α として用いる．キーポイント中心 O を原点とすると，モーメント m_{pq} と重心 C は次のように計算できる．

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \qquad (9)$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \qquad (10)$$

従って，オリエンテーション α は次式で与えられる．

$$\theta = \mathrm{atan2}(m_{01}, m_{10}) \qquad (11)$$

二値特徴量を求める際には，輝度差を求める 2 点のペアの座標位置を α だけ回転させることで，回転不変性を獲得している．

ORB では，統計的に「良い」性質を持つサンプリングペアを教師なし学習により選択する．さて，一体どのような二値特徴量が「良い」特徴量であろうか？情報量の観点から考えると，各ビット y_i が 0 もしくは 1 になる確率はそれぞれ 50% となることが望ましい．また，異なる二つのビット y_i, y_j は確率的に独立である方が良い．そのよう

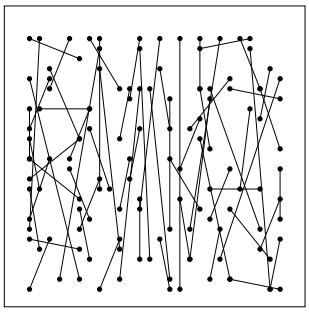


図 5 ORB のサンプリングパターン

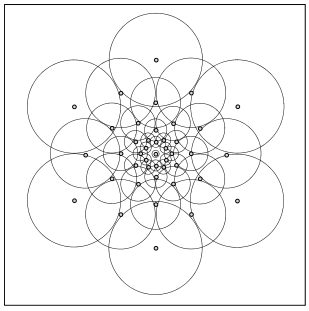


図 6 FREAK のサンプリングパターン

な二値特徴量はエントロピーが高く無駄がないといえるからである．これについては，ORB に限らず，様々な文献で同様の指摘がされている [27], [29]．これは，異なる二つのビット y_i, y_j の相関が小さく，y_i の分散が大きい方が良い二値特徴量である，と言い換えても良い．例えば，y_i が 100%の確率で 0 を取ってしまうような極端なケースの場合，y_i の分散はゼロになる．この場合，明らかに y_i は二値特徴量としては不適切である．逆に，0 と 1 が等確率でバランスよく発生するのであれば，y_i は最大の分散を持つ．これは良い二値特徴量であるといってよい．

そこで ORB では，パッチ内で有り得る 205,590 種類のペアを全て候補として列挙し，ビットの分散が大きく，かつペア同士の相関が低くなるようなペアを Greedy アルゴリズムで選択することにより，256 組のペアに絞り込むことで，記述能力の高い二値特徴量を生成している．以下にアルゴリズムの概要を示す．

(1) 全てのペアの候補について，ビットの分散が大きいもの順にソート．
(2) 最大の分散を持つペアを採用．
(3) 次に大きな分散を持つペアに着目し，このペアが採用済みのどのペアとも相関が低いのであれば採用．相関が高ければ棄却．
(4) 3 番目のステップを繰り返し，256 組のペアが見つかった時点で終了．

図 5 に ORB によって選択された上位 64 個のペアを示した．選択されたペアには縦方向の偏りが生じていることが分かる．これは，オリエンテーションの補正が入ることでパッチ内の輝度の分布に偏りが生じるからである．

2.4 FREAK: Fast Retina Keypoint

Alahi ら [3] によって提案された FREAK もまた，ORB や BRISK と同様に輝度差の符号によって二値特徴量を生成する手法であり，生物の網膜の構造を模している点が特徴的である．

FREAK では，眼球内において，網膜の中心に近づくほど神経節細胞の密度が高くなるという生物学的事実を参考とし，キーポイント中心に近づくほど密度が高くなるようなサンプリングパターンを用いることを提案している．図 6 は FREAK で用いられているサンプリングのパターンである．それぞれのサンプリング点を中心とする円は，平滑化のためのガウスフィルタの範囲を表す．この円の大きさは外側に向かうほど指数関数的に大きくなるように設計されている．外側のサンプリング点における平滑化の範囲は，互いに重なり合うほど大きくなるから，サンプリング点における値は隣り合う点と独立にはならない．しかしながら，このような冗長性は生物の視覚システムの中にも存在しており，これが識別性能の向上に寄与すると Alahi らは主張している．

FREAK で採用されているサンプリング点の数は ORB や BRISK と比べても少なく，たった 43 箇所である．しかしながら，これらの全ての組み合わせを用いて二値特徴量を抽出すると，そのビット長は数千ビットにも及ぶため，ここでもやはりペアの選択が必要である．そこで，ORB と同じ Greedy アルゴリズムを適用することで，二値特徴量の情報量が大きくなるような 512 種類のペアを選択している．

ここで Alahi らは，選ばれた 512 種類のペアを観察すると，そこには生物の視覚システムと似た挙動が現れていると主張している．生物の視覚システムは，周辺視野で大まかに対象を把握したのちに，中心視野で対象物をより細かく観察するように機能する．一方で，FREAK において用いられている 512 種類のペアを，Greedy アルゴリズムで選ばれた順に追っていくと，初めは外側のサンプリングパターンが選択されていることが多く，後半になると中心に近いサンプリングパターンが選択されていることが多いことがわかる．すなわち，生物の視覚システムと同様に coarse-to-fine な特徴選択が行われていることがわかる．

この点に着目し，FREAK では 512 ビットの二値特徴量の距離計算を行う際に，128 ビットごとに 4 分割し，4 段のカスケード構造を取ることで，マッチング処理の高速化を図っている．すなわち，各段では 128 ビット分のみの距離計算を行い，距離の値が所定のしきい値を超える場合は計算を打ち切ることで，マッチング処理を高速化している．実際には，初段で 90%以上の候補が棄却されるため，実質 128 ビット分の距離計算のみでマッチングが行える．

なお，FREAK におけるキーポイント位置検出，スケー

ルおよびオリエンテーション推定は BRISK とほぼ全く同じである．

2.5 D-BRIEF: Discriminative BRIEF

BRIEF, BRISK, ORB, FREAK では，事前に定めたサンプリング点の周辺に平滑化などのフィルタを適用した後に，2 つのサンプリング点間の輝度差の符号を用いて二値特徴量を定義した．キーポイント周辺のパッチを $\mathbf{x}$ とベクトルで表記[*3]すれば，フィルタの演算と輝度差の演算は，$\mathbf{x}$ に関して線形であるから，結局のところ i 番目のビット y_i は (12) 式で表せる[*4]．

$$y_i = \mathrm{sgn}(\mathbf{w}_i^\top \mathbf{x} + \tau_i) \tag{12}$$

例えば，フィルタ演算を用いず，単純にサンプリング点間の輝度差の符号で二値特徴量を生成するのであれば，$\tau_i = 0$ とし，$\mathbf{w}_i$ のサンプリング点のペアに対応する位置にそれぞれ+1 と-1，それ以外に 0 を代入すればよい．結局のところ，BRIEF, BRISK, ORB, FREAK の違いは，$\mathbf{w}_i$ と τ_i の選び方の違いに他ならないといえる．

Trzcinski ら [26] によって提案された D-BRIEF は，教師あり学習を適用することで，より記述能力の高い二値特徴量を生成できる $\mathbf{w}_i$ と τ_i を求めようとするものである．実際には $\mathbf{w}_i$ はパッチのピクセル数分の次元を持つため，$\mathbf{w}_i$ と $\mathbf{x}$ の内積を直接計算すると大きな計算コストがかかる．そこで D-BRIEF ではさらなる工夫として，高速に演算可能な線形フィルタの候補を多数用意しておき，$\mathbf{w}_i$ をこれらのフィルタの中から選ばれた少数のフィルタの線形和で表現することで，$\mathbf{w}_i^\top \mathbf{x}$ の計算を高速化している．

$$\mathbf{w}_i = \mathbf{D}\mathbf{s}_i \tag{13}$$

$\mathbf{D}$ はフィルタの辞書であり，$\mathbf{s}_i$ は係数である．$\mathbf{s}_i$ がスパースであれば，$\mathbf{w}_i$ は少ない数のフィルタで再構成されることになる．フィルタの候補としては，矩形やガウスフィルタを用いる．例えば矩形のフィルタであれば，積分画像を用いて高速に演算できる．

D-BRIEF における学習処理は，(14) 式のコスト関数を最小化することで達成できる．

$$\begin{aligned}\min_{(\mathbf{s}_i,\tau_i)} \quad & \sum_{i\in 1,\cdots,B} \\ & \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}} \mathrm{sgn}((\mathbf{D}\mathbf{s}_i)^\top\mathbf{x}+\tau_i)\mathrm{sgn}((\mathbf{D}\mathbf{s}_i)^\top\mathbf{x}'+\tau_i) \\ & - \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} \mathrm{sgn}((\mathbf{D}\mathbf{s}_i)^\top\mathbf{x}+\tau_i)\mathrm{sgn}((\mathbf{D}\mathbf{s}_i)^\top\mathbf{x}'+\tau_i) \\ & + \lambda|\mathbf{s}_i|_1 \\ & \text{subject to } (\mathbf{D}\mathbf{s}_i)^\top(\mathbf{D}\mathbf{s}_j) = \delta_{ij}\end{aligned} \tag{14}$$

ここで，$(\mathbf{x},\mathbf{x}') \in \mathcal{P}$ はハミング距離を小さくしたいパッチのペアの集合，$(\mathbf{x},\mathbf{x}') \in \mathcal{N}$ はハミング距離を大きくしたいペアの集合であり，これらは学習サンプルとして事前に収集しておく．λ が大きいほど，$\mathbf{s}_i$ が疎になり，$\mathbf{w}_i$ はより少ない数のフィルタで再構成されることになる．また，δ_{ij} は $i = j$ ときに 1，それ以外は 0 を取る関数である．制約条件 $(\mathbf{D}\mathbf{s}_i)^\top(\mathbf{D}\mathbf{s}_j) = \delta_{ij}$ は，異なるビット同士を独立にして二値特徴量が持つ情報量を高めることを狙っており，これは ORB の学習における考え方に類似している．

(14) 式は原著論文に記載の表記そのものであるが，やや複雑で理解しにくいと思われるため，少し整理しておく．$\mathbf{x},\mathbf{x}'$ を二値特徴量に変換したものをそれぞれ $\mathbf{y},\mathbf{y}'$ と表記すると，(14) 式は以下のように簡単に整理できる．

$$\min_{(\mathbf{s}_i,\tau_i)} \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}} \mathbf{y}^\top\mathbf{y}' - \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} \mathbf{y}^\top\mathbf{y}' + \lambda|\mathbf{s}_i|_1 \tag{15}$$

つまり，$\mathcal{N}$ に所属するペアに関しては，二値特徴量同士の内積を小さくし，$\mathcal{P}$ に所属するペアに関しては，二値特徴量同士の内積を大きくするように $\mathbf{s}_i,\tau_i$ を学習していることが分かる．D-BRIEF においては，$\mathbf{y},\mathbf{y}' \in \{-1,+1\}^B$ と定義されているから、二値特徴量のハミング距離 $d_H(\mathbf{y},\mathbf{y}')$ と内積 $\mathbf{y}^\top\mathbf{y}'$ の間には (16) 式の関係がある．

$$\mathbf{y}^\top\mathbf{y}' = B - 2d_H(\mathbf{y},\mathbf{y}') \tag{16}$$

つまり，ふたつの二値特徴量 $\mathbf{y},\mathbf{y}'$ の内積を大きくするという作用は，$\mathbf{y}$ と $\mathbf{y}'$ のハミング距離を小さくするという作用に相当する．したがって，(14) 式を最小化することで，$\mathcal{P}$ に所属するペア間のハミング距離が小さくなり，$\mathcal{N}$ に所属するペア間のハミング距離が大きくなるように $\mathbf{w}_i,\tau_i$ が最適化されることが分かる．

以上，D-BRIEF における問題は (14) 式の最小化問題に帰着できることを示したが，実際には (14) 式を $\mathbf{s}_i,\tau_i$ に関して直接解くことは困難であるため，最適化のための工夫が必要である．そこで，sgn 関数と L1 正則化の項を取り除き，$\mathbf{s}_i$ ではなく，$\mathbf{w}_i$ について解くことを考えると，(17) 式のように簡略化できる．

$$\{\mathbf{w}_i^0\} = \arg\min_{\{\mathbf{w}_i\}} \sum_i \frac{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}}(\mathbf{w}_i^\top(\mathbf{x}-\mathbf{x}'))^2}{\sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{N}}(\mathbf{w}_i^\top(\mathbf{x}-\mathbf{x}'))^2} \tag{17}$$

$\mathbf{w}_i$ に直交性を仮定すると，これは Linear Discriminant

[*3] 例えば，$N \times N$ ピクセルのサイズのパッチから二値特徴量を抽出することを考えるのであれば，$\mathbf{x}$ はパッチ内の輝度を並べた N^2 次元のベクトルで表せる．
[*4] これは，3 節で後述する Binary Hashing と本質的には同じ考え方である．

Embedding (LDE)[8] の問題設定そのものであり，固有値計算で解を得ることができる．sgn 関数を取り除くことでコスト関数から τ_i が消えてしまっているが，LDE によって求めた $\mathbf{w}_i$ を固定し，τ_i に関する一次元の探索問題として (14) 式を解きなおすことで τ_i の解を得ることができる[*5]．

$\mathbf{w}_i^0$ を少数のフィルタの線形和で近似するためには，$\mathbf{w}_i^0$ と $\mathbf{D}\mathbf{s}_i$ の誤差が小さく，かつ $\mathbf{s}_i$ ができるだけ疎になるような解を求めればよい．すなわち，求めたそれぞれの $\mathbf{w}_i$ について，次のコスト関数を最小化することで，$\mathbf{s}_i$ を得る．

$$\{\mathbf{s}_i^0\} = \arg\min_{\mathbf{s}_i} ||\mathbf{w}_i^0 - \mathbf{D}\mathbf{s}_i||_2^2 + \lambda|\mathbf{s}_i|_1 \tag{18}$$

これは凸関数であるから，近接勾配法などで解くことができる．

2.6 BinBoost

Trzcinski ら [25] によって提案された BinBoost もまた教師あり学習に基づく手法であり，K 個の弱識別器の線形和の符号で 1 ビット分の二値特徴量を生成する．

$$y_i(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}_i^\top \mathbf{h}_i(\mathbf{x})) \tag{19}$$

$\mathbf{x}$ はパッチの輝度，$y_i \in \{-1, +1\}$ は i 番目のビット，$\mathbf{h}_i \in \mathbb{R}^K$ は K 個の弱識別器の出力，$\mathbf{w}_i \in \mathbb{R}^K$ はそれらに対する重みである．$\mathbf{h}_i$ はパッチの輝度 $\mathbf{x}$ に関して非線形の関数でもよく，任意の特徴抽出手法を適用できる．この点が D-BRIEF と異なる部分であり，弱識別器に適切な特徴抽出器を用いることで，高い記述能力を持った二値特徴量が生成できる．論文中では，パッチ内の矩形領域内における輝度勾配の方向を弱識別器として提案している．

BinBoost では，AdaBoost で採用されている考え方に近い定式化が行われている．まず，N 個のラベル付き訓練サンプルを $\{(\mathbf{x}_n, \mathbf{x}'_n, l_n)\}_{n=1}^N$ で定義する．$\mathbf{x}_n$ と $\mathbf{x}'_n$ が対応点である場合は $l_n = +1$, そうでない場合は $l_n = -1$ とする．このとき，(20) 式のコスト関数を $\mathbf{w}_i$ と $\mathbf{h}_i$ に関して最小化することで学習を行う．

$$\mathcal{L} = \min_{\{\mathbf{w}_i, \mathbf{h}_i\}_{i=1}^B} \sum_{n=1}^N \exp(-\gamma l_n \sum_{i=1}^B c_i(\mathbf{x}_n, \mathbf{x}'_n; \mathbf{w}_i, \mathbf{h}_i)) \tag{20}$$

γ は学習のパラメータであり，関数 c_i は次のように定義されている．

$$\begin{aligned} c_i(\mathbf{x}_n, \mathbf{x}'_n; \mathbf{w}_i, \mathbf{h}_i) &= y_i(\mathbf{x}) y_i(\mathbf{x}') \\ &= \mathrm{sgn}(\mathbf{w}_i^\top \mathbf{h}_i(\mathbf{x})) \mathrm{sgn}(\mathbf{w}_i^\top \mathbf{h}_i(\mathbf{x}')) \end{aligned} \tag{21}$$

このコスト関数の最小化は，$l_n = +1$ のペアに関してはハミング距離を小さくし，$l_n = -1$ のペアに関してはハミング距離を大きくするように作用する．これは (16) 式の関係が成立することから明らかである．

AdaBoost における考え方に倣い，$i = 1$ から順番に弱識別器 $\mathbf{h}_i$ と重み $\mathbf{w}_i$ を最適化することを考える．これは，(20) 式の代わりに (22) 式のコスト関数を最大化することで達成される．

$$\max_{\mathbf{w}_i, \mathbf{h}_i} \sum_{n=1}^N l_n W_i(n) c_i(\mathbf{x}_n, \mathbf{x}'_n; \mathbf{w}_i, \mathbf{h}_i) \tag{22}$$

$W_i(n)$ は訓練サンプルごとに定義される重み係数であり，次のように定義される．

$$W_i(n) = \exp(-\gamma l_n \sum_{i'=1}^{i-1} c_{i'}(\mathbf{x}_n, \mathbf{x}'_n; \mathbf{w}_i, \mathbf{h}_i)) \tag{23}$$

すなわち，$1, \cdots, i-1$ 番目までのビットで正しく識別できていない訓練サンプルは重みが大きくなり，既に識別できている訓練サンプルは重みが小さくなる．これは AdaBoost の考え方と非常に似ている．

関数 c_i の中にある sgn 関数は微分できないため，依然として (22) 式は解くことが難しい．そこで sgn 関数を取り除き，近似したコスト関数を最大化する．これを (24) 式に示す．

$$\max_{\mathbf{w}_i, \mathbf{h}_i} \mathbf{w}_i^\top (\sum_{n=1}^N l_n W_i(n) \mathbf{h}_i(\mathbf{x}) \mathbf{h}_i(\mathbf{x}')^\top) \mathbf{w}_i \tag{24}$$

まず，$\mathbf{h}_i$ に関してこの問題を解くことを考える．弱識別器の候補は無数に存在して良いが，それらの中から大きな重み $W_i(n)$ が与えられている訓練サンプルを適切に識別できる弱識別器の集合を求めなければならない．これは特徴選択の問題であり，BinBoost では，[23] で提案されている特徴選択手法を適用してビット毎に K 個の弱識別器を選択している．つまり，選択される弱識別器はビットごとに異なるが，その数は常に K 個である．合計で，$B \times K$ 個の弱識別器によって B ビットの二値特徴量が構成されることになる．

次に，$\mathbf{b}_i$ に関して (24) 式を解くことを考える．$\mathbf{h}_i$ が決まれば，(24) 式は次のように簡単に整理できる．

$$\max_{\mathbf{w}_i} \mathbf{w}_i^\top \mathbf{M} \mathbf{w}_i \tag{25}$$

なお，

$$\mathbf{M} = \sum_{n=1}^N l_n W_i(n) \mathbf{h}_i(\mathbf{x}) \mathbf{h}_i(\mathbf{x}')^\top \tag{26}$$

と置いた．(19) 式からも明らかである通り，$\mathbf{w}_i$ はスケール倍の不定性があるため $||\mathbf{w}_i||_2 = 1$ としても一般性は失われない．従ってこれを仮定すると，(25) 式の最大化問題は $\mathbf{M}$ の最大固有値に対応する固有ベクトルを求めることで解くことができる．

[*5] 話が前後するが，このしきい値 τ_i を一次元の探索問題として解くという解き方は，後述する LDAHash[24] で行われているやり方と全く同じである．

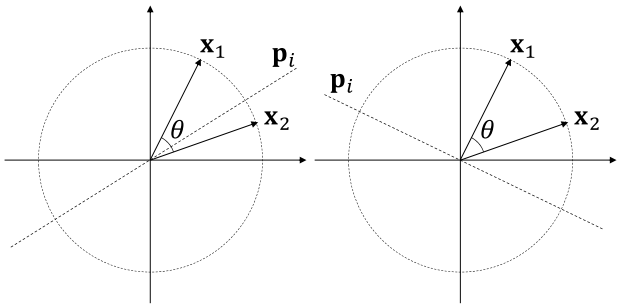


図 7 Random projections におけるベクトル間角度とハミング距離の関係

3. 間接的に二値特徴量を生成する方法

実数ベクトルを二値ベクトルに変換する手法は Binary Hashing と呼ばれ，コンピュータビジョンの分野だけでなく，機械学習や web，大規模検索などといった他の分野でも活発に研究されている．その中でも最も基本的なアプローチは，線形写像と符号関数による手法である．これを (27) 式に示す．

$$\mathbf{y} = \mathrm{sgn}(\mathbf{P}\mathbf{x} + \mathbf{t}) \tag{27}$$

$\mathbf{x}$ は D 次元の実数ベクトル，$\mathbf{y}$ は B 次元の二値ベクトル，$\mathbf{t}$ はしきい値，$\mathbf{P}$ は B 行 D 列の変換行列である．$\mathbf{P}$ および $\mathbf{t}$ をどのように設計するのかがポイントであり，これまでに様々な研究が行われている．

本節では，まずは最も単純な方法である Random projections[5], [10] を紹介する．次に，対応点探索のための局所特徴量を二値に変換することを特に意図して設計された二つの手法 (CARD[4], LDAHash[24]) を紹介する．CARD は教師なし学習，LDAHash は教師あり学習に基づく方法である．

3.1 Random Projections

Random projections [5], [10] は最も簡単な Binary Hashing 手法であり，$\mathbf{P}$ の各要素を乱数で生成する．これは二値ベクトルに変換するやり方としては一見乱暴に思えるが，実は必ずしもそうではない．なぜなら，$\mathbf{x}$ が，原点を中心とした D 次元空間内における超球上に分布している場合，$B \to \infty$ の極限において，元々の実数の特徴空間におけるベクトル間角度と，二値特徴量のハミング距離とが，定数倍の関係を持つからである．

この理由は，実数ベクトルが 2 次元 ($D = 2$) のケースで考えると直感的に理解しやすい．図 7 は，2 次元の空間において，原点を中心とした半径 1 の円上に二つの実数ベクトル $\mathbf{x}_1, \mathbf{x}_2$ が存在している様子を示したものである．今，これらのベクトルの間の角度を θ とおく．

(27) 式は線形の変換を行った後にその符号を得るというものである．$\mathbf{t} = 0$ であれば，これは $\mathbf{P}$ の各行が原点を通る直線を定義しており，これが 2 次元の空間をふたつの領域に分割していると解釈できる．二つに分かたれた空間のうち，実数特徴量がどちらに属しているかによって，アサインされるビットが異なるわけである．

Random projections では $\mathbf{P}$ を乱数で生成するため，2 次元の空間をふたつに分割する直線もまた，ランダムに決定されることになる．θ が大きければ $\mathbf{x}_1$ と $\mathbf{x}_2$ に異なるビットがアサインされる確率が高くなり，θ が小さければ $\mathbf{x}_1$ と $\mathbf{x}_2$ に異なるビットがアサインされる確率が低くなる．明らかに，この直線が $\mathbf{x}_1$ と $\mathbf{x}_2$ の間に入る確率は，θ に比例している．すなわち，(28) 式が成立する．

$$\Pr[\mathrm{sgn}(\mathbf{p}_i^\top \mathbf{x}_1) \neq \mathrm{sgn}(\mathbf{p}_i^\top \mathbf{x}_2)] = \frac{\theta}{\pi} \tag{28}$$

(28) 式の左辺は，ある直線 $\mathbf{p}_i$ で決定されるビットが異なる確率であり，右辺は $\mathbf{x}_1$ および $\mathbf{x}_2$ の間の角度を正規化した値である．左辺は確率であるが，$B \to \infty$ の極限においてこれはハミング距離をビット長 B で正規化した値と一致する．したがって，Random projections が実数の空間における距離関係を保存することがわかる．

実際に SIFT や SURF をこの方法で二値特徴量にするためには，平均を原点に移動 (mean centering) した後に Random projections を計算すればよい．もちろん，Random projections で十分な性能を出すためには，比較的長いビット長が必要であるが，それでも十分実用に足る場合も多い．Random Projections は学習処理も不要であるため，手軽に試せるのは大きな利点である．

3.2 CARD: Compact And Real-time Descriptors

Ambai[4] らによって提案された CARD は，SIFT とほぼ同等の定義に基づく勾配ヒストグラム特徴量を抽出し，これを二値特徴量に変換するという 2 段階の手順を取っている．ポイントは，(1) ルックアップテーブルを用いることで，勾配ヒストグラム特徴量の演算を大幅に高速化したこと，および (2)Binary Hashing の変換行列 $\mathbf{P}$ を疎行列にすることで，実数ベクトルから二値ベクトルへの変換を高速化したことにある．CARD のキーポイント検出は，先に紹介した BRISK，ORB とほぼ同じ考え方（ピラミッド画像に対するコーナー検出）に基づいているためここでは説明を省略する．以下，勾配ヒストグラム特徴量の高速抽出方法と，疎行列による Binary Hashing の部分について解説する．

3.2.1 ルックアップテーブルを用いた勾配ヒストグラム特徴量の高速抽出

SIFT と同様に，CARD ではパッチのオリエンテーションを補正した後に，キーポイント周辺領域を K 個のセルに分割し，それぞれのセルから L 次元の勾配ヒストグラム特徴量を抽出するという考え方に基づいている．CARD では特徴量の記述能力をより高めるために，4×4 のグリッ

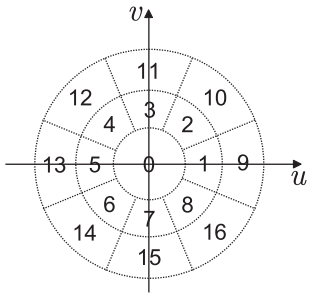
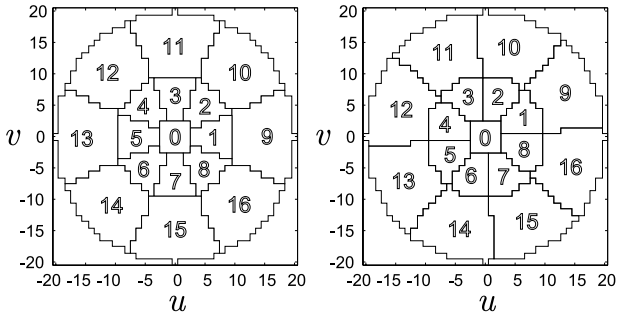


図 8 CARD におけるパッチの分割



(a) $\alpha = 0$ (b) $\alpha = 3$

図 9 パッチの分割パターンのテーブル

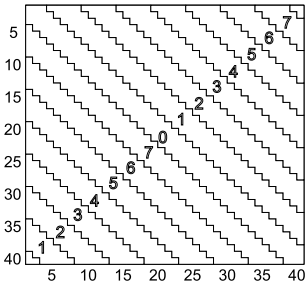


図 10 量子化レベルの変換テーブル

ド分割ではなく，放射状に分割している（図 8）．すなわち $K = 17, L = 8$ であり，特徴量は 136 次元である．

まず，SIFT と同じ手続きに基づいてパッチのオリエンテーション α を算出する．ただし，SIFT では勾配ヒストグラムに 2 次の多項式フィッティングを適用することでオリエンテーションを連続値として求めるが，CARD ではこれを行わない．すなわち，$\alpha = 0, 1, \cdots, M-1$ という量子化された値を取るものとする．オリエンテーションが離散値であることを許すことで，テーブル化のテクニックが適用できるようになる．

CARD では，オリエンテーションに従ってパッチの画素配列を回転させるのではなく，パッチの分割パターンの方を回転させることを考える．α は M 通りの離散的な値しか取りえないため，M 通りの回転させた分割パターンを予め用意しておけば良い．図 9 に，$M = 40$ のとき $\alpha = 0, 3$ にそれぞれ対応する分割パターンを示した．分割パターンを用いることで，パッチ周辺の画素 $(x, y)^\top$ がどのセルに属するかを高速に求められる．

次に各セルから L 次元の勾配ヒストグラムを求めることを考える．ここでのポイントは，オリエンテーションを求めるときに，M 段階に量子化された勾配方向 $\theta(x, y)$ が既に計算済みであるという点である．オリエンテーションを求めるときは比較的細かい量子化間隔を用いるため，$M \gg L$ が成立する．従って，（わずかな誤差を伴うが）M から L への量子化レベルの変換が可能であるから，$\theta(x, y)$ を再利用できる．そこで，量子化レベル数の変換と，オリエンテーションの補正を (29) 式で同時に行う．

$$l = Q_L(Q_M^{-1}(\theta(x, y) - \alpha)) \tag{29}$$

l は小領域内の勾配ヒストグラムの投票位置を示すインデックスであり，$l = 0, \cdots, L-1$ という値を取る．なお，$Q_N(\cdot)$ はラジアンで表現されている角度を 0 から $N-1$ に量子化する関数であり，$Q_N^{-1}(\cdot)$ はその逆関数である．$\theta(x, y) - \alpha$ がオリエンテーションの補正に相当し，$Q_L(Q_M^{-1}(\cdot))$ が量子化レベルの変換に相当する．(29) 式は複雑に見えるが，$\theta(x, y)$ と α がそれぞれ M 通りの値しか取りえないことに着目すると，図 10 に示すように，$M \times M$ のテーブルで表現できることが分かる．従って，図 9，10 に示した 2 種類のテーブルを用いることで，高速に勾配ヒストグラム特徴量を算出できるようになる．

3.2.2 二値特徴量への変換

(30) 式に示す線形の Binary Hashing により実数ベクトルを二値ベクトルに変換する．

$$\mathbf{y} = \mathrm{sgn}(\mathbf{P}\mathbf{x}) \tag{30}$$

ここでは，実数特徴量 $\mathbf{x}$ の平均が原点に位置するようにあらかじめ平行移動されているものとし，しきい値 $\mathbf{t}$ は無視する．

Binary Hashing の課題は，変換速度が遅いという点である．変換行列 $\mathbf{P}$ は B 行 D 列の密な行列であるから，(30) 式の計算には $B \times D$ 回の掛け算と $B \times (D-1)$ 回の足し算が必要である．

そこで，次の 2 つの条件に基づいて $\mathbf{P}$ を最適化により求めている．

(1) 二値特徴量に変換前の距離 (ベクトル間角度) と変換後の距離（ハミング距離）がなるべく一致するように $\mathbf{P}$ を最適化する．

(2) $\mathbf{P}$ は S 個の非ゼロ要素から成る疎行列であり，各要素が $-1, 0, 1$ のうちいずれかの値のみを取るという制約のもとで $\mathbf{W}$ を最適化する．

条件 (1) により，生成された二値特徴量は元々の特徴量の記述能力を維持できるようになる．また，条件 (2) により，(30) 式による二値特徴量への変換は S 回の足し算のみで行えるようになる．

直感的には $\mathbf{P}$ が疎になればなるほど性能が劣化すると考えられるが，興味深いことに $\mathbf{P}$ の要素が 90% 程度ゼロで

あっても，密な行列の場合と性能がほとんど変わらないことが実験により確かめられている．これは高速化にとって非常に都合が良い性質である．

なお，バイナリコードのビット長は目的に応じて任意に決めることができるが，著者らは精度と速度のバランスの観点から 128bit 程度を推奨している．

3.3 LDAHash

Strecha ら [24] によって提案された LDAHash は，教師あり学習に基づいた手法である．物体のある一点を，異なる 2 つの視点から撮影して得られた局所特徴量のペアの集合を $(\mathbf{x}, \mathbf{x}') \in \mathcal{P}$ と定義する．また，物体の異なる 2 点から得られた局所特徴量のペアの集合を $(\mathbf{x}, \mathbf{x}') \in \mathcal{N}$ と定義する．このとき，$\mathcal{P}$ に属するペア間のハミング距離を小さく，かつ $\mathcal{N}$ に属するペア間のハミング距離を大きくするような変換行列 $\mathbf{P}$ およびしきい値 $\mathbf{t}$ を選択することを考えることで，より記述能力の高い二値特徴量を得ようというのが LDAHash における考え方である．これは，(31) 式のコスト関数を最小化することで達成できる．

$$L = \alpha E\{d_H(\mathbf{y}, \mathbf{y}')|\mathcal{P}\} - E\{d_H(\mathbf{y}, \mathbf{y}')|\mathcal{N}\} \tag{31}$$

d_H は $\mathbf{y}$ と $\mathbf{y}'$ のハミング距離を求める関数である．$E\{\cdot|\mathcal{P}\}$, $E\{\cdot|\mathcal{N}\}$ は $\mathcal{P}$,$\mathcal{N}$ に関する条件付き期待値であり，$\mathcal{P}$,$\mathcal{N}$ それぞれの集合に対する平均値と考えればよい．

ここで，(31) 式の最小化問題の性質について整理する．(31) 式は，次に示す (32)(33) 式の最小化と等価である．

$$L = E\{\mathbf{y}^\top \mathbf{y}'|\mathcal{N}\} - \alpha E\{\mathbf{y}^\top \mathbf{y}'|\mathcal{P}\} \tag{32}$$

$$L = \alpha E\{||\mathbf{y} - \mathbf{y}'||^2|\mathcal{P}\} - E\{||\mathbf{y} - \mathbf{y}'||^2|\mathcal{N}\} \tag{33}$$

$\mathbf{y} \in \{-1, +1\}^B$ であるから，(31) 式と (32) 式が等価であることは，ハミング距離 d_H と内積 $\mathbf{y}^\top \mathbf{y}'$ の間に (16) 式の関係が成立することから明らかである．また，(31) 式と (33) 式の関係は，次のように導出できる．まず，ユークリッド距離の二乗の式を次のように展開する．

$$||\mathbf{y} - \mathbf{y}'||^2 = ||\mathbf{y}||^2 - 2\mathbf{y}^\top \mathbf{y}' + ||\mathbf{y}'||^2 \tag{34}$$

(16) 式を (34) 式に代入して整理すると，(35) 式を得る．

$$||\mathbf{y} - \mathbf{y}'||^2 = 4d_H(\mathbf{y}, \mathbf{y}') - 2B + ||\mathbf{y}||^2 + ||\mathbf{y}'||^2 \tag{35}$$

$\mathbf{y} \in \{-1, +1\}^B$ と定義されていることから，特徴量によらず $||\mathbf{y}||^2 = ||\mathbf{y}'||^2 = B$ であることは明らかである．したがって，最適化に影響を与えるのは右辺の第一項目のみであり，やはり (31) 式と (33) 式の最小化問題は等価であることがわかる．

$\mathbf{P}$, $\mathbf{t}$ に関して (33) 式を直接最小化できれば良いが，実際にはこれは困難である．なぜなら $\mathbf{y}, \mathbf{y}'$ の計算には微分不可能な sgn 関数が含まれるからである．そこで，(33) 式から sgn 関数を取り除き，緩和したコスト関数を最小化することを考える．

$$\tilde{L} = \alpha E\{||\mathbf{Px} - \mathbf{Px}'||^2|\mathcal{P}\} - E\{||\mathbf{Px} - \mathbf{Px}'||^2|\mathcal{N}\} \tag{36}$$

(36) 式の最小化は，しきい値 $\mathbf{t}$ とは独立になる．そこで，まずは (33) 式を緩和した (36) 式を $\mathbf{P}$ について最小化し，次に $\mathbf{P}$ を固定しつつ (32) 式を最小化することで $\mathbf{t}$ を得るという二段階のステップで最適化を行う．以下，それぞれの手順について述べる．

3.3.1 変換行列 P の選択

Strecha らは，(36) 式の最小化方法として，Linear Discriminant Analysis (LDA) による方法と Difference of Covariances (DIF) による方法の二種類を提案しており，いずれも固有値問題に帰着することで解を得ている．

学習サンプル集合 $\mathcal{P}, \mathcal{N}$ それぞれについて，ペアの差分ベクトルの共分散行列 $\mathbf{\Sigma}_\mathcal{P}, \mathbf{\Sigma}_\mathcal{N}$ を次のように定義する．

$$\mathbf{\Sigma}_\mathcal{P} = E\{(\mathbf{x} - \mathbf{x}')(\mathbf{x} - \mathbf{x}')^\top|\mathcal{P}\} \tag{37}$$

$$\mathbf{\Sigma}_\mathcal{N} = E\{(\mathbf{x} - \mathbf{x}')(\mathbf{x} - \mathbf{x}')^\top|\mathcal{N}\} \tag{38}$$

このとき, LDA では, 二つの共分散行列の比 $\mathbf{\Sigma}_\mathcal{R} = \mathbf{\Sigma}_\mathcal{P}\mathbf{\Sigma}_\mathcal{N}^{-1}$ について固有値計算を行い，DIF では，二つの共分散行列の差 $\mathbf{\Sigma}_\mathcal{D} = \alpha\mathbf{\Sigma}_\mathcal{P} - \mathbf{\Sigma}_\mathcal{N}$ について固有値計算を行う．以下，これらの導出過程について，それぞれ説明する．

Linear Discriminant Analysis (LDA)

$\mathbf{\Sigma}_\mathcal{P}, \mathbf{\Sigma}_\mathcal{N}$ を用いると，(36) 式は次のように変形できる．

$$\tilde{L} = \alpha\mathrm{tr}(\mathbf{P}\mathbf{\Sigma}_\mathcal{P}\mathbf{P}^\top) - \mathrm{tr}(\mathbf{P}\mathbf{\Sigma}_\mathcal{N}\mathbf{P}^\top) \tag{39}$$

このとき，$\mathbf{x}$ に $\mathbf{\Sigma}_\mathcal{N}^{-1/2}$ を掛けて正規化しておくと，(39) 式の第二項目は定数とみなせる．したがって，(39) 式は次のように整理できる．

$$\tilde{L} \propto \mathrm{tr}(\mathbf{P}\mathbf{\Sigma}_\mathcal{N}^{-1/2}\mathbf{\Sigma}_\mathcal{P}\mathbf{\Sigma}_\mathcal{N}^{-\top/2}\mathbf{P}^\top) \tag{40}$$

$$= \mathrm{tr}(\mathbf{P}\mathbf{\Sigma}_\mathcal{P}\mathbf{\Sigma}_\mathcal{N}^{-1}\mathbf{P}^\top) = \mathrm{tr}(\mathbf{P}\mathbf{\Sigma}_\mathcal{R}\mathbf{P}^\top) \tag{41}$$

$\mathbf{\Sigma}_\mathcal{R} = \mathbf{\Sigma}_\mathcal{P}\mathbf{\Sigma}_\mathcal{N}^{-1}$ であり，これは $\mathbf{\Sigma}_\mathcal{P}$ と $\mathbf{\Sigma}_\mathcal{N}$ の比に相当する．$\mathrm{tr}(\mathbf{P}\mathbf{\Sigma}_\mathcal{R}\mathbf{P}^\top)$ を最小化するためには，$\mathbf{\Sigma}_\mathcal{R}$ の小さい方から B 個の固有値に対応する固有ベクトルで張られる部分空間への射影を考えればよい．従って，著者らは

$$\tilde{\mathbf{S}}_\mathcal{R}^{-1/2}\tilde{\mathbf{U}}_\mathcal{R}^\top\mathbf{\Sigma}_\mathcal{N}^{-1/2} \tag{42}$$

という変換を推奨している．$\tilde{\mathbf{S}}_\mathcal{R}$ は対角要素に小さい方から B 個の固有値を格納した B 行 B 列の行列であり，$\tilde{\mathbf{U}}_\mathcal{R}$ はそれに対応する固有ベクトルを格納した D 行 B 列の行列である．

Difference of Covariances (DIF)

(39) 式は，二つの共分散行列の差 $\mathbf{\Sigma}_\mathcal{D} = \alpha\mathbf{\Sigma}_\mathcal{P} - \mathbf{\Sigma}_\mathcal{N}$ を用いて次のようにも表せる．

$$\tilde{L} = \mathrm{tr}(\mathbf{P}\mathbf{\Sigma}_\mathcal{D}\mathbf{P}^\top) \tag{43}$$

LDA と同様に，$\mathbf{P}$ が正規直交であることを仮定して $\mathbf{\Sigma}_{\mathcal{D}}$ の固有値分解を行えばよい．

対角要素に小さい方から B 個の固有値を格納した B 行 B 列の行列 $\tilde{\mathbf{S}}_{\mathcal{D}}$ と，それに対応する固有ベクトルを格納した D 行 B 列の行列 $\tilde{\mathbf{U}}_{\mathcal{D}}$ を用いると，変換行列 $\mathbf{P}$ は，(44) 式で与えられる．

$$\mathbf{P} = \tilde{\mathbf{S}}_{\mathcal{D}}^{-1/2}\tilde{\mathbf{U}}_{\mathcal{D}}^{\top} \tag{44}$$

LDA とは異なり，DIF は $\mathcal{P}$ と $\mathcal{N}$ の影響のバランスを決定するパラメータ α に選択の余地がある．$\alpha \to \infty$ の極限においては，$\mathbf{\Sigma}_{\mathcal{N}}$ の影響が無くなるため，これは $\mathbf{\Sigma}_{\mathcal{N}} = \mathbf{I}$ を仮定したことと等価になる．

3.3.2 しきい値 t の選択

次に，$\mathbf{P}$ を固定して (32) 式を $\mathbf{t}$ に関して最小化することを考える．この最適化問題において，$\mathbf{t}$ の各要素は独立している．したがって，$\mathbf{t}$ の i 番目の要素 t_i の解は，次の最適化問題を解くことで得られる．

$$\begin{aligned}\min_{t_i}\ & E\ \{\mathrm{sgn}((\mathbf{p}_i^{\top}\mathbf{x}+t_i)^{\top}(\mathbf{p}_i^{\top}\mathbf{x}'+t_i))|\mathcal{N}\} \\ & -\ \alpha E\{\mathrm{sgn}((\mathbf{p}_i^{\top}\mathbf{x}+t_i)^{\top}(\mathbf{p}_i^{\top}\mathbf{x}'+t_i))|\mathcal{P}\}\end{aligned} \tag{45}$$

$\mathbf{p}_i^{\top}$ は，$\mathbf{P}$ の i 番目の行ベクトルである．これは 1 変数の最適化問題であるから，t_i を変化させて最適値を探索すればよい．

4. 各手法の特徴のまとめ

2, 3 節で紹介した手法の特徴を表 1 にまとめた．この表では，該当項目における星の数が多いほど性能が良いとしている．この表は各論文における評価結果を参考にし，また著者らの主張を尊重した上でまとめたものであるが，この結果は実験条件や実装によっても異なるであろうし，これらの手法を同一の条件で比較した実験例が示されていないことからも，あくまで全体を理解するための一つの目安として参考されたい．

現在までの動向を見る限り，LDAHash, D-BRIEF, BinBoost といった教師あり学習による手法が最も高い記述能力を持つ二値特徴量を生成できるようである．しかしながら，これらの手法にも問題が無いわけではない．LDAHash は Binary Hashing における変換行列が密であるため，特徴抽出の速度が遅く，リアルタイムなアプリケーションには向かないと考えられる．一方，D-BRIEF では二値特徴量を少ない数のフィルタの線形和から生成するという高速化のための配慮がなされているが，スケールと回転不変性を導入したい場合，パッチを回転させた上で所望の形状にリサイズするなどの前処理が必要である．BinBoost は特徴抽出自体が遅く，1 つの二値特徴量を抽出するのに 1 msec ほどかかることが論文中で報告されている．これは局所特徴量抽出にかかる処理時間としてはかなり遅い方である．教師あり学習による記述能力の向上，回転・スケール不変性の導入，特徴量の高速抽出をどのように同時に達成するかが，今後の課題と言えるかも知れない．

```
#include "intrin.h"
int _mm_popcnt_u32 ( unsigned int a);
int _mm_popcnt_u64 ( unsigned __int64 a);
```

図 11 SSE4.2 のビットカウント

本稿で紹介してきた手法を今一度振り返ってみると，その全ては (46) 式で一般化できる．

$$\mathbf{y} = \mathrm{sgn}(\mathbf{P}f(\mathbf{x}) + \mathbf{t}) \tag{46}$$

$\mathbf{x} \in \mathbb{R}^{N^2}$ は $N \times N$ ピクセルのサイズを持つパッチの輝度を並べたベクトル，$\mathbf{P} \in \mathbb{R}^{B\times D}$ は変換行列，$\mathbf{t} \in \mathbb{R}^{B}$ はしきい値，そして $f(\cdot)$ は $\mathbf{x}$ から何らかの特徴抽出を行う関数であり，

$$f : \mathbb{R}^{N^2} \to \mathbb{R}^{D} \tag{47}$$

で定義される．各手法の違いは，$\mathbf{P}$, $\mathbf{t}$, $f(\cdot)$ の選び方の違いそのものである．これを表 2 にまとめた．例えば，BRIEF, BRISK, ORB, FREAK, D-BRIEF はパッチの輝度に対する線形の演算と符合の判定のみで構成されることから，パッチの輝度 $\mathbf{x}$ に対して直接的に線型の Binary Hashing を適用していると解釈できる．すなわち，$f(\mathbf{x}) = \mathbf{x}$ である．一方，BinBoost では，$f(\mathbf{x})$ に D 個の非線形の弱識別器を用い，$\mathbf{P}$ には弱識別器の重みが用いられていると解釈できる．間接的な方法と位置付けた Random projections, CARD, LDAHash では，$f(\mathbf{x})$ は SIFT や SURF のアルゴリズムの演算そのものである．

つまり，ここで紹介したすべての手法は，何らかの特徴変換を施した後に線形の Binary Hashing を適用したものであると，統一的に理解できる．本稿ではこれらの手法を直接的／間接的という二つの観点で分類したが，結局のところ，両者の違いは $f(\mathbf{x})$ にどれだけ複雑な処理を担わせるかの違いであるといえよう．唯一，BinBoost では $f(\mathbf{x})$，$\mathbf{P}$ の両方を Boosting の枠組みで同時に設計する方針をとっており，直接法と間接法の中間に位置しているといえる．

5. 実装方法／研究用のリソースについて

5.1 高速なビットカウント

ハミング距離は XOR とビットカウントの組み合わせで計算できる．Intel の Core i7 など，SSE4.2 に対応しているプロセッサであれば，専用の演算命令を用いることでビットカウントを極めて高速に処理できる．C/C++言語であれば，intrin.h を include することで 32/64 ビット用のビットカウント命令を使えるようになる (図 11) [*6]．ただし，64 ビット用のビットカウント命令_mm_popcnt_u64() は，64 ビットアプリケーションでのみ利用可能であること

[*6] GCC ではコンパイル時にオプション (-msse4.2) の指定が必要．

表 1 特徴量の比較

分類	特徴量	学習の有無	記述能力	特徴抽出の速度	特徴量のサイズ	スケール・回転への対応
直接	BRIEF (ECCV 2010)	不要	☆	☆☆☆	☆	サンプリングパターンを回転・拡大縮小することで対応可能
	BRISK (ICCV 2011)	不要	☆☆	☆☆☆	☆	
	ORB (ICCV 2011)	教師なし	☆☆	☆☆☆	☆☆	
	FREAK (CVPR 2012)	教師なし	☆☆	☆☆☆	☆	
	D-BRIEF (ECCV 2012)	教師あり	☆☆☆	☆☆	☆☆☆	事前にパッチを正規化するなど，別途対処が必要
	BinBoost (CVPR 2013)	教師あり	☆☆☆☆	☆	☆☆☆	
間接	Random projections	不要	☆	☆	☆	変換前の特徴量が既にスケール・回転不変性を持っていれば問題ない
	CARD (ICCV 2011)	教師なし	☆☆	☆☆☆	☆☆	
	LDAHash (PAMI 2012)	教師あり	☆☆☆	☆	☆☆☆	

表 2 特徴抽出アルゴリズムの一般化

特徴量	変換行列 $\mathbf{P}$	しきい値 $\mathbf{t}$	特徴抽出 $f(\mathbf{x})$
BRIEF (ECCV2010)	平滑化したサンプリング点の輝度差	$\mathbf{t}=0$	$f(\mathbf{x})=\mathbf{x}$
BRISK (ICCV2011)	平滑化したサンプリング点の輝度差	$\mathbf{t}=0$	$f(\mathbf{x})=\mathbf{x}$
ORB (ICCV2011)	平滑化したサンプリング点の輝度差	$\mathbf{t}=0$	$f(\mathbf{x})=\mathbf{x}$
FREAK (CVPR2012)	平滑化したサンプリング点の輝度差	$\mathbf{t}=0$	$f(\mathbf{x})=\mathbf{x}$
D-BRIEF (ECCV2012)	フィルタの線形和	一次元探索	$f(\mathbf{x})=\mathbf{x}$
BinBoost (CVPR2013)	弱識別器の重み	$\mathbf{t}=0$	非線形の弱識別器
Random projections	ランダム	$\mathbf{t}=0$	従来手法 (SIFT, SURF など) + mean centering
CARD (ICCV2011)	$\{-1,0,+1\}$ のいずれかの要素のみを取る疎行列	$\mathbf{t}=0$	LUT による勾配ヒストグラム特徴量抽出+ mean centering
LDAHash (PAMI2012)	LDA もしくは DIF による変換行列	一次元探索	従来手法 (SIFT, SURF など)

```
inline unsigned int popcnt32(unsigned int x)
{
    x = (x & 0x55555555) + ((x >>  1) & 0x55555555);
    x = (x & 0x33333333) + ((x >>  2) & 0x33333333);
    x = (x & 0x0f0f0f0f) + ((x >>  4) & 0x0f0f0f0f);
    x = (x & 0x00ff00ff) + ((x >>  8) & 0x00ff00ff);
    x = (x & 0x0000ffff) + ((x >> 16) & 0x0000ffff);
    return x;
}
```

図 12 ビットを数えるアルゴリズム：分割統治法

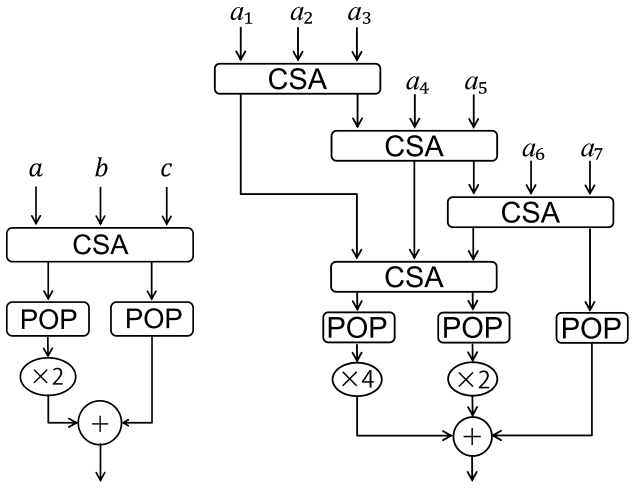
に注意する必要がある．

ビットカウントを計算する専用命令が搭載されていない場合でも，基本的な四則演算と論理演算の組み合わせで高速に計算することが可能である．このテクニックについては文献 [18], [28] が詳しい．誰でも簡単に思いつく方法としては，ビット列を 8 ビットずつに分割し，256 通りのビットカウントの結果をルックアップテーブルに保存しておくというアプローチがある．多くの環境では，この実装方法でも十分高速に動作する．また別の方法として，分割統治法によるアルゴリズムが知られている．これを図 12 に示す．驚くべきことに，このアルゴリズムではループを必要としない．どちらが速いかは一概には言えず，環境や実装方法によるようである．

ビットカウントをしたいビット列が配列に保存されており，全体のビット数が非常に長い場合については，キャリー保存加算器 (carry-save adder, CSA) を用いると高速化できることがある [18]．キャリー保存加算器は独立した全加算器が並んだものであり，3 つの入力 a,b,c と 2 つの出力 h,l を持つ．C/C++言語であれば図 13 のようにマクロで定義できる．本来，a,b,c のビットカウントの総和を

```
#define CSA(h,l,a,b,c) \
  {unsigned u = a ^ b; unsigned v = c; \
    h = (a & b) | (u & v); l = u ^ v;}
```

図 13 キャリー保存加算器 (carry-save adder, CSA)



(a) 3 語のグループ　(b) 7 語のグループ

図 14 キャリー保存加算器によるビットカウント

得るためには，3 回のビットカウントを行わなければならないが，CSA が持つ次の性質を利用すると，1 回の CSA と 2 回のビットカウントに置き換えることができる．

$$\mathrm{pop}(a) + \mathrm{pop}(b) + \mathrm{pop}(c) = 2 \cdot \mathrm{pop}(h) + \mathrm{pop}(\mathrm{l}) \quad (48)$$

pop(·) はビットを数える関数である．すなわち，a, b, c に関してビットカウントを 3 回行う代わりに，a, b, c に関して CSA を 1 回適用して h, l を求め，l のビットカウント値と h のビットカウント値の 2 倍を足すことで，全く同じ結果が得られる．ビットカウントの演算速度よりも，CSA の演算速度が速い場合は，a, b, c を CSA によって 3 つまとめて処理した方が高速に演算できるわけである．この計算過程は，図 14(a) のように回路図として書きあらわすと分かりやすい．同様の考え方で，CSA の出力を別の CSA に入力し，CSA の回路を多段に構成することで，より多くの数をグルーピングして処理できる．図 14(b) は 7 つをグルーピングした場合の例である．このように，4 回の CSA と 3 回のビットカウントで処理できることがわかる．ビットカウントよりも CSA の方が速い場合は，グルーピングする数が多いほど高速に処理できるようになる．また，多段に組んだ CSA の回路構成において，最終的な出力の一部を最初の入力にフィードバックするというテクニックも高速化に寄与することが知られている．このあたりのテクニックは文献 [18] が詳しい．

このテクニックはビットカウントの専用命令を備えていない CPU で，比較的長いビット長のビットカウントを行いたい場合には利用を検討してもよい．しかしながら，CPU がビットカウントの専用演算命令を備えており，その実行速度が CSA よりもはるかに速い場合はこれらの工夫は無意味であり，素直に専用演算命令のみでビットカウントを行うべきである．

5.2 ソフトウェア／データセット

本稿で紹介した二値特徴量のソフトウェアについて，2013 年 10 月時点で公開されているものを表 3 にまとめておく．画像間の対応点探索性能を評価するためのデータセットとしては，Mikolajczyk ら [17] による Affine Covariant Regions Datasets[1] が最も頻繁に利用されている．これは，ほぼ平面で近似可能な対象を複数の視点から撮影した画像を集めたものである．画像間の Homography 行列が与えられているため，これを用いて真の対応点の位置を求めることができる．Brown ら [7] による Multi-view Stereo Correspondence Dataset[2] では，Difference of Gaussian (DOG) や Harris コーナー検出器によって得たキーポイント周辺のパッチを，スケールとオリエンテーションに関して正規化して得た 64×64 ピクセルの画像データを提供している．パッチ同士がマッチしているか否かのラベル情報も与えられているため，教師あり学習の研究用途に向いている．

6. おわりに

本稿では「対応点探索のための」二値特徴量を解説することに焦点を絞ったが，実数特徴量を二値特徴量に変換するという研究は，それはそれで Binary Hashing という一つの分野として幅広く研究されている．そこで，最後にこれらの研究動向についても軽く触れておく．2007 年に SIGIR の Workshop で Hinton らのグループが RBM を用いた Semantic Hashing[21] を発表し，これを受けて Weiss らが 2008 年の NIPS で Spectral Hashing[29] を発表したころから Binary Hashing の研究は存在感を増してきた．コンピュータビジョンの分野でも研究は進み，2011 年の CVPR で発表された Iterative Quantization[12] は理論が明快で実装がしやすく，性能も良かったことから，Binary Hashing の研究におけるベースラインとして頻繁に利用されている．近年では非線形の方法 [13] や，VLAD および Fisher Vector のような高次元の特徴量を二値に変換する手法 [11]，実数特徴量から二値特徴量への変換を高速化した手法 [22] などが検討されている．これらの研究で示されている知見は，対応点探索のための二値特徴量を設計する上でも参考となるところが多い．

本稿では，対応点探索のための二値特徴量について，特に 2010 年から 2013 年に発表された文献を中心に解説した．また，本稿で取り上げた手法が相互にどのような関係にあるかについて総括した．4 節でも指摘したように，今のところはどの手法も特徴抽出と線形の Binary Hashing の組み合わせとみなせるようである．このような包括的な視点が，読者の理解の一助となれば幸いである．もちろん，今後はこの解釈から外れた新たな手法が提案されることもあろう．この分野の今後の発展に期待したい．

参考文献

[1] : Affine Covariant Regions Datasets, (online), available from ⟨http://www.robots.ox.ac.uk/∼vgg/data/data-aff.html⟩ (accessed).
[2] : Multi-view Stereo Correspondence Dataset, (online), available from ⟨http://www.cs.ubc.ca/∼mbrown/patchdata/patchdata.html⟩ (accessed).
[3] Alahi, A., Ortiz, R. and Vandergheynst, P.: FREAK: Fast Retina Keypoint, *CVPR*, pp. 510–517 (2012).
[4] Ambai, M. and Yoshida, Y.: CARD: Compact And Real-time Descriptors, *ICCV*, pp. 97–104 (2011).
[5] Andoni, A. and Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, *Communications of the ACM* (2008).
[6] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L.: Speeded-Up Robust Features (SURF), *Computer Vision and Image Understanding*, Vol. 110, pp. 346–359 (2008).
[7] Brown, M., Hua, G. and Winder, S.: Discriminant Learning of Local Image Descriptors, *PAMI*, Vol. 33, pp. 43–57 (2011).
[8] Brown, M., Hua, G. and Winder, S.: Discriminative

表 3 利用可能なソフトウェア

特徴量	開発環境	入手先
BRIEF (ECCV2010)	C++(OpenCV2.0 以上必須)	http://cvlab.epfl.ch/research/detect/brief
BRISK (ICCV2011)	C++(OpenCV2.2 以上必須), MATLAB	http://www.asl.ethz.ch/people/lestefan/personal/BRISK
	OpenCV	http://opencv.org/
ORB (ICCV2011)	OpenCV	http://opencv.org/
FREAK (CVPR2012)	C++(OpenCV 必須)	http://www.ivpe.com/freak.htm
	OpenCV	http://opencv.org/
	MATLAB	MATLAB R2013a Computer Vision System Toolbox 5.2 以降
D-BRIEF (ECCV2012)	C++(OpenCV2.0 以上必須), MATLAB	http://cvlab.epfl.ch/research/detect/dbrief
BinBoost (CVPR2013)	C++	http://infoscience.epfl.ch/record/186246?ln=en
CARD (ICCV2011)	MATLAB	http://cvlab.jp/ (Denso IT Laboratory, Inc.)
	iOS 用サンプルアプリケーション	アプリ名: CARDesc, AppStore (Apple Inc.)
LDAHash (PAMI2012)	C++(OpenCV 必須)	http://cvlab.epfl.ch/research/detect/ldahash

Learning of Local Image Descriptors, *PAMI*, Vol. 33, No. 1, pp. 43–57 (2011).
[9] Calonder, M., Lepetit, V., Strecha, C. and Fua, P.: BRIEF: Binary Robust Independent Elementary Features, *ECCV*, pp. 778–792 (2010).
[10] Goemans, M. X. and Williamson, D. P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, *Journal of the ACM*, Vol. 42, pp. 1115–1145 (1995).
[11] Gong, Y., Kumar, S., Rowley, H. A. and Lazebnik, S.: Learning Binary Codes for High-Dimensional Data Using Bilinear Projections, *CVPR*, pp. 484–491 (2013).
[12] Gong, Y. and Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes, *CVPR*, pp. 817–824 (2011).
[13] Heo, J.-P., Lee, Y., He, J., Chang, S.-F. and Yoon, S.-E.: Spherical hashing, *CVPR*, pp. 2957–2964 (2012).
[14] Ke, Y. and Sukthankar, R.: PCA-SIFT: A More Distinctive Representation for Local Image Descriptors, *CVPR*, pp. 506–513 (2004).
[15] Leutenegger, S., Chli, M. and Siegwart, R.: BRISK: Binary Robust invariant scalable keypoints, *ICCV*, pp. 2548–2555 (2011).
[16] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints, *IJCV*, Vol. 60, pp. 91–110 (2004).
[17] Mikolajczyk, K. and Schmid, C.: A Performance Evaluation of Local Descriptors, *PAMI*, Vol. 27, pp. 1615–1630 (2005).
[18] Oram, A. and Wilson, G.: *Beautiful Code: Leading Programmers Explain How They Think*, Oreilly & Associates Inc (2007).
[19] Rosten, E. and Drummond, T.: Machine learning for high-speed corner detection, *ECCV*, pp. 430–443 (2006).
[20] Rublee, E., Rabaud, V., Konolige, K. and Bradski, G.: ORB: An efficient alternative to SIFT or SURF, *ICCV*, pp. 2564–2571 (2011).
[21] Salakhutdinov, R. R. and Hinton, G. E.: Semantic hashing, *SIGIR workshop on Information Retrieval and applications of Graphical Models* (2007).
[22] Sato, I., Ambai, M. and Suzuki, K.: Sparse Isotropic Hashing, *IPSJ Transactions on Computer Vision and Applications*, Vol. 5, No. 0, pp. 40–44 (2013).
[23] Shakhnarovich, G.: Learning Task-Specific Similarity, *Ph.D thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science* (2006).
[24] Strecha, C., Bronstein, A., Bronstein, M. and Fua, P.: LDAHash: Improved Matching with Smaller Descriptors, *PAMI*, Vol. 34, No. 1, pp. 66–78 (2012).
[25] Trzcinski, T., Christoudias, M., Lepetit, V. and Fua, P.: Boosting Binary Keypoint Descriptors, *CVPR*, pp. 2874–2881 (2013).
[26] Trzcinski, T. and Lepetit, V.: Efficient Discriminative Projections for Compact Binary Descriptors, *ECCV*, pp. 228–242 (2012).
[27] Wang, J., Kumar, S. and Chang, S.-F.: Sequential Projection Learning for Hashing with Compact Codes, *ICML* (2010).
[28] Warren, H. S.: *Hacker's Delight (2nd Edition)*, Addison-Wesley Professional (2012).
[29] Weiss, Y., Torralba, A. and Fergus, R.: Spectral Hashing, *NIPS*, pp. 1753–1760 (2008).