高速かつメモリ消費量の少ない局所特徴量

CARD: Compact And Real-time Descriptors

安倍 満† 吉田 悠一†

† 株式会社デンソーアイティーラボラトリ
E-mail: †{manbai,yyoshida}@d-itlab.co.jp

あらまし 本論文では，高速に計算可能であり，128 ビット程度のバイナリコードで表現可能な局所特徴量 Compact And Real-time Descriptors (CARD) を提案する．CARD は SIFT や GLOH と同様に，勾配ヒストグラム特徴量の一種である．提案手法は，(1) ルックアップテーブルによる高速な局所特徴量抽出，(2) Supervised Sparse Hashing (SSH) による高速なバイナリコード化の 2 段階から成る．本手法では，2 種類のルックアップテーブルを用いることで，1 つのキーポイントあたり SIFT の約 16 倍の速度で局所特徴量を計算可能とした．また，抽出された特徴量をハッシュ関数によりバイナリコードに変換することで，メモリ消費量を低減した．ここで用いるハッシュ関数は疎行列の掛け算により構成されているため，高速にバイナリコードへ変換が可能である．実験により，CARD は実行速度，メモリ消費量の二つの点において，既存の手法よりも優れていることを示した．

キーワード 局所特徴量，バイナリコード

1. はじめに

画像間の対応付けはコンピュータビジョンにおける最も基本的なタスクであり，近年では局所特徴量が有効であると考えられている．Scale-Invariant Feature Transform (SIFT) [1] はその代表的な手法であり，スケールおよび回転不変な局所特徴量を抽出可能としている．SIFT の応用範囲は広く，3 次元形状復元 [2]，特定物体認識 [3]，一般物体認識 [4] など，様々な目的に利用されている．SIFT を始めとして，多くの局所特徴量が提案されているが，これらは計算速度が遅く，またメモリ消費量が大きいという課題を抱えている．そのため車載向け CPU や携帯端末などといった，PC よりも比較的性能の低い機器で動作させることが困難という課題が残されている．

以下，具体例として SIFT を例に挙げて議論する．図 1 は SIFT の計算速度を示したものであり，サイズが 640×480 の画像から 3,762 個のキーポイントを得たときの結果である．全体の処理に 2 秒近くを要しており，そのうち約 75%がキーポイントの方向推定と局所特徴量の抽出に費やされていることが分かる．また，SIFT は 128 次元という高次元の特徴ベクトルで表現されるため，大きくメモリを消費する．多くの場合，この特徴ベクトルはメモリ消費量を抑えるために符号無し 8bit 整数の配列に格納されるが，それでも 1 つのキーポイントあたり $8 \times 128 = 1024$ ビット必要である．

そこで本論文では，これら 2 つの課題に焦点を当てる．すなわち，高速に計算可能であり，128 ビット程度のバイナリコードで表現可能な局所特徴量 Compact And Real-time Descriptors (CARD) を提案する．

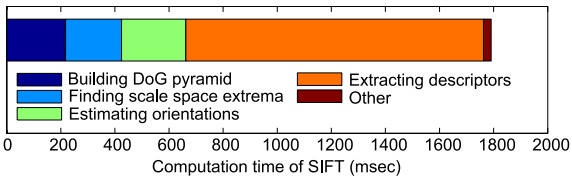


図 1 SIFT の計算時間 (PC: Intel Core 2 Duo 2.66GHz).

1.1 関連研究

局所特徴量の高速化: 局所特徴量の高速化は，産業応用という観点から非常に意義が深く，幅広く研究が行われている．Bay [5] らは，SIFT と同様に回転かつスケール不変な局所特徴量 Speeded-Up Robust Features (SURF) を提案し，SIFT に対して数倍の高速化を達成した．Takacs [6] らは，携帯端末上での高速な特徴追跡とコンテンツ認識を目的として Rotation-Invariant, Fast Feature (RIFF) を提案した．同様に，Wagner [7] らもまた，携帯端末上の動作を目的とした局所特徴量を提案し，拡張現実感における高速なカメラ姿勢推定を実現した．近年では，前記の研究事例よりもさらなる高速化に焦点を当てた研究 [8], [9] が行われており，高速化という課題に対する注目度の高さを窺い知ることができる．しかしながら，これらの研究の多くは，SIFT よりも簡略化された局所特徴量を用いることで高速化を達成しているため，たとえば大規模な特定物体認識などに適用すると十分な認識性能が得られないという課題がある．

局所特徴量のコンパクト表現: 近年，特徴ベクトルを短いバイナリコードに変換する Binary hashing [10], [11], [12], [13], [14], [15] が注目されている．数多くの手法が

提案されているが，変換前の局所特徴量を $\mathbf{d} \in \mathbb{R}^D$，変換後のバイナリコードを $\mathbf{b} \in \{0,1\}^B$ と記述するとき，Binary hashing は $\mathbf{b} = (\mathrm{sgn}(f(\mathbf{W}^\top \mathbf{d})) + 1)/2$ と一般的に定義できることが知られている [13]．ここで，D は局所特徴量の次元数，B はバイナリコードのビット長，$f(\cdot)$ は任意の関数，$\mathbf{W} \in \mathbb{R}^{D \times B}$ は変換行列である．バイナリコード同士の類似度はハミング距離によって計算できる．これは変換前における局所特徴量同士の類似度の計算 (コサイン距離，ベクトル間角度，ユークリッド距離など) よりもはるかに高速に計算可能である．従って，メモリ効率化だけでなく，最近傍探索の速度も向上できるという利点がある．

Binary hashing の性能は，$\mathbf{W}$ と $f(\cdot)$ の定義によって大きく異なる．最も単純な例は Random projections [12], [16] であり，$f(\cdot)$ は恒等関数，$\mathbf{W}$ の各要素は正規分布に基づく乱数で定義される．一方，学習を導入することで，乱数に基づく方法よりもより短いバイナリコードが得られることが知られている [10], [13]．例えば Spectral hashing [10] では，$f(\cdot)$ に非線形関数，$\mathbf{W}$ に学習データの主成分軸を用いている．以上，Binary hashing について述べたが，この他にも情報エントロピーに基づく方法 [17]，次元削減に基づく方法 [18] など，局所特徴量を圧縮するための様々な手法が提案されている．

以上述べた手法は，バイナリコードへの変換にかかる計算時間についての議論が欠けている．リアルタイムなアプリケーションにおいては，この変換も高速に行える必要がある．Sparse random projections [14], [15] はこの課題に焦点を当てている．Achlioptas [14] らは，$\mathbf{W}$ の要素として 3 種類の整数 $\{-1, 0, 1\}$ をそれぞれ $\{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\}$ の確率で選択することで，Binary Hashing における $\mathbf{W}^\top \mathbf{d}$ の計算量を大幅に削減した．これに対し，Li [15] らは 3 種類の整数 $\{-1, 0, 1\}$ をそれぞれ $\{\frac{1}{2\sqrt{D}}, 1 - \frac{1}{\sqrt{D}}, \frac{1}{2\sqrt{D}}\}$ の確率で選択し，$\mathbf{W}$ をより疎にすることでさらなる高速化が可能であると指摘している．なお，$f(\cdot)$ としては恒等関数を用いる．Sparse random projections は学習に基づいていないため，Random projections と同様に，バイナリコード化すると性能が低下するという欠点がある．

1.2 提案手法 (CARD) の特徴

高速な局所特徴量抽出: 2 種類のルックアップテーブルを用いることで，1 つのキーポイントあたり SIFT の約 16 倍の速度で局所特徴量を計算可能とした．CARD は SIFT および Gradient Location and Orientation Histogram (GLOH) と同様に勾配ヒストグラム特徴量であり，十分な識別能力を有することを示した．

Supervised sparse hashing: Sparse random projections に学習の概念を導入した．変換前における距離と，変換後のバイナリコード間のハミング距離が相関するように $\mathbf{W}$ を学習することで，Sparse random projections の性能を改善した．

2. 高速な局所特徴量抽出

2.1 キーポイント検出

スケール不変性はキーポイント検出における重要な性質である．Lowe [1] は Difference of Gaussians (DoG) ピラミッドの極値をスケール不変なキーポイントとして定義した．Dufournaud [19] らは，分散を徐々に大きくした複数のガウシアンフィルタを入力画像に適用することで，スケール不変な Harris コーナー検出器を実現した．これらの欠点は，異なる分散のガウシアンフィルタを幾度も適用しなければならず，計算が重いという点である．

これに対して我々は，入力画像を一定の割合で縮小した多段のピラミッド画像を作成し，各層における画像から独立に特徴点 [20] を抽出する方法を採用した．この方法でも，実用上十分なスケール不変性が得られることが報告されている [6], [7]．0 段目を入力画像とするとき，$n-1$ 段目の画像を $1/\sqrt{2}$ 倍に縮小することで n 段目の画像を生成した．実際の計算においては，n 段目の画像は $n-2$ 段目の画像を 1/2 にダウンサンプリングするだけで得られるため，高速にピラミッド画像を生成できる．

2.2 勾配ヒストグラムに基づく局所特徴量

SIFT や GLOH は勾配ヒストグラムに基づく局所特徴量であり，他に比べて高い識別能力を持つことが知られている [21]．本節では，これを高速に算出するアルゴリズムについて述べる．提案するアルゴリズムは，(a) ピクセルの線形補間処理が不要，(b) ルックアップテーブルによる効率的な計算が可能という 2 つの特徴を持つ．

2.2.1 従来手法 (SIFT, GLOH) の概要

提案アルゴリズムを説明する前に，従来手法 (SIFT, GLOH) による局所特徴量抽出の概要を述べる．まずは記号を整理する．入力画像を $I(x,y)$，キーポイントの座標を $(p_x, p_y)^\top$ と記述する．ピクセルの勾配の強度 $m(x,y)$ と方向 $\theta(x,y)$ は次のように計算できる．

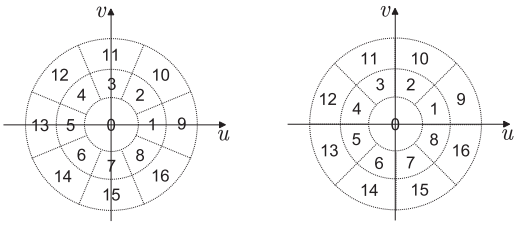
$$m(x,y) = \sqrt{I_x(x,y)^2 + I_y(x,y)^2} \qquad (1)$$

$$\theta(x,y) = \mathrm{angle}(I_x(x,y), I_y(x,y)) \qquad (2)$$

ここで，I_x および I_y は $I(x,y)$ の x,y 方向微分であり，$\mathrm{angle}(u,v)$ は方向をラジアンで返す関数である．角度は u 軸の正方向から v 軸正方向に向かって反時計回りに $[0, 2\pi)$ の範囲の値をとる．

推定された方向とスケールに基づき，キーポイント周辺の領域は変形 (拡大／縮小，回転) される．これにより，局所特徴量はスケールおよび回転に関する不変性を得る．変形された周辺領域は K 個の小領域に分割される．例えば，SIFT では 4×4 のグリッド分割，GLOH では放射状 (log-polar) 分割が行われる．各小領域から L 次元の方向ヒストグラム $\mathbf{h}_k$ を抽出する．

$$\mathbf{h}_k = (h_{k,0}, h_{k,1}, \cdots, h_{k,(L-1)})^\top \in \mathbb{R}^L \qquad (3)$$



$\rho = 0$ $\rho = \pi/8$

図 2　log-polar 分割のパラメータ表現.

ここで，k は小領域のインデックスである．$\mathbf{h}_k$ を全て統合することで，KL 次元の局所特徴量 $\mathbf{d} \in \mathbb{R}^{KL}$ を得る．

$$\mathbf{d} = (\mathbf{h}_0^\top, \mathbf{h}_1^\top, \cdots, \mathbf{h}_{K-1}^\top)^\top \tag{4}$$

記述を明確にするために，次節以降ではキーポイント周辺領域の分割の仕方を「ブロック分割パターン」，各小領域を「ブロック」と記載することにする．

なお，勾配ヒストグラムを求めるときに必要な，ラジアン表現の角度を 0 から $N-1$ に量子化する関数 $Q_N(\theta)$ は次のように定義できる．

$$Q_N(\theta) = \lfloor \frac{N\theta}{2\pi} + \frac{1}{2} \rfloor \bmod N \tag{5}$$

2.2.2　ピクセル補間を用いない局所特徴量抽出

従来手法では，キーポイント周辺の領域を変形 (拡大／縮小，回転) させることにより，スケールおよび回転に関する不変性を獲得しているが，この処理には膨大なピクセル補間処理が伴う．

これに対し，我々はブロック分割パターン自体をキーポイント周辺領域の方向（以下,「主軸」と記述）に応じて回転させることで，計算負荷の高いピクセル補間処理を回避した．主軸の角度（ラジアン）を ρ とするとき，回転済みブロック分割パターンは 3 つのパラメータを持つ関数 $\psi(u, v, \rho)$ で記述できる．ここで $(u, v)^\top$ はキーポイントの座標 $(p_x, p_y)^\top$ を中心としたときの相対座標である．$\psi(u, v, \rho)$ は 0 から $K-1$ のブロック番号を返す関数であり，ブロック分割パターンは uv-平面上に現れ，ρ に応じて回転するように定義する．図 2 に $\psi(u, v, \rho)$ の例を示した．以上に基づくと，キーポイント周辺における座標位置 $(x, y)^\top$ のピクセルが属するブロックは次のように決定できる．

$$k = \psi(x - p_x, y - p_y, \rho) \tag{6}$$

座標位置 $(x, y)^\top$ におけるピクセルの方向は L 段階に量子化され，投票処理を行うことで $\mathbf{h}_k$ を得られる．ただしブロック分割パターンが回転しているため，ρ による回転の影響を打ち消すためにピクセルの勾配方向 $\theta(x, y)$ を補正する必要がある．勾配方向を補正し，量子化した値 l は次式で計算できる．

$$l = Q_L(\theta(x, y) - \rho) \tag{7}$$

結果として，キーポイント $(p_x, p_y)^\top$ から局所特徴量 $\mathbf{d}$ を抽出するアルゴリズムは次のように簡略化できる．

1. $\mathbf{h}_0, \cdots, \mathbf{h}_{K-1}$ をゼロで初期化.
2. キーポイント $(p_x, p_y)^\top$ の周辺全てのピクセル $(x, y)^\top$ について次の 2 つの処理 (a)(b) を適用.
   (a) 式 (6)(7) を計算し，k, l を取得.
   (b) $h_{k,l} \leftarrow h_{k,l} + m(x, y) G_{\sigma_1}(x - p_x, y - p_y)$
3. $\mathbf{h}_0, \cdots, \mathbf{h}_{K-1}$ を統合し，$\mathbf{d}$ を取得.
4. $\mathbf{d}$ の長さを 1 に正規化.

（※ G_σ は分散 σ の正規分布とする.）

ブロック分割パターンとしては，GLOH で採用されているような放射状 (log-polar) 分割が最も良いと報告されている [21]．そこで，本論文でもこれを適用する．このとき，関数 $\psi(u, v, \rho)$ は次のように定義できる．

$$\psi(u, v, \rho) = \tag{8}$$
$$\begin{cases} 0 & , \quad 0 \leqq r < r_1 \\ 1 + Q_8(\mathrm{angle}(u, v) - \rho) & , \quad r_1 \leqq r < r_2 \\ 9 + Q_8(\mathrm{angle}(u, v) - \rho) & , \quad r_2 \leqq r \leqq r_3 \end{cases}$$

ここで，$r = \sqrt{u^2 + v^2}$ であり，r_1, r_2, r_3 は分割のためのしきい値である．図 2 のように，キーポイント周辺領域は 17 個のブロックに分割される．なお，提案手法では，放射状分割だけでなく SIFT で採用されているグリッド分割など，任意のパターンを扱うことができる．

2.2.3　ルックアップテーブルによる高速化

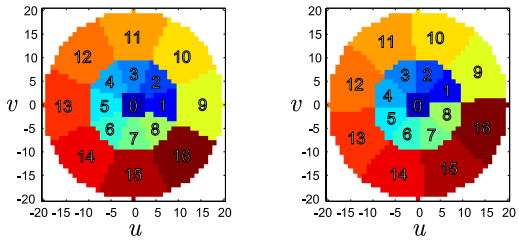
実際には，式 (6)(7) の計算は後述する 2 つのルックアップテーブル（$\bar{\psi}$ および χ）で置き換えられるため，大幅な高速化が可能である．

ルックアップテーブルに基づくアルゴリズムを説明する前に，キーポイントの主軸の求め方について整理しておく．主軸は，キーポイント周辺領域の勾配を M 段階に量子化し，勾配ヒストグラム $\hat{\mathbf{h}}$ の最大値を探すことで得られる．

1. $\theta(x, y)$ を M 段階に量子化 : $i_\theta(x, y) = Q_M(\theta(x, y))$.
2. $\hat{\mathbf{h}} = (\hat{h}_0, \cdots, \hat{h}_{M-1})^\top$ をゼロで初期化.
3. キーポイント $(p_x, p_y)^\top$ の周辺全てのピクセル $(x, y)^\top$ について次の 2 つの処理 (a)(b) を適用.
   (a) $i = i_\theta(x, y)$.
   (b) $\hat{h}_i \leftarrow \hat{h}_i + m(x, y) G_{\sigma_2}(x - p_x, y - p_y)$
4. 分散 σ_s の正規分布で $\hat{\mathbf{h}}$ を平滑化.
5. 勾配ヒストグラム $\hat{\mathbf{h}}$ の最大値を主軸として取得.
   $i_\rho = \arg\max_i(\hat{h}_i)$.

まず，式 (6) をルックアップテーブルで置き換えることを考える．主軸 i_ρ は量子化されているため，$\psi(u, v, \rho)$ の計算はルックアップテーブル化が可能である．そのために，まず $Q_N(\theta)$ の逆関数を導入する．

$$Q_N^{-1}(i) = (2\pi(i \bmod N))/N, \tag{9}$$



$i_\rho = 0 \quad (\rho = 0)$ $\qquad$ $i_\rho = 3 \quad (\rho = \pi/8)$

図 3　ブロック分割パターンテーブル $\bar{\psi}(u, v, i_\rho)$. $(M = 40)$

$Q_N^{-1}(i)$ は量子化済の勾配方向を，当該量子化レベルにおける代表値（ラジアン）に変換する関数である．量子化済の主軸 i_ρ は，$Q_N^{-1}(i)$ を用いてラジアンに戻すことができる．そこで，式 (8) に $\rho = Q_M^{-1}(i_\rho)$ を代入することで，ブロック分割パターンテーブル $\bar{\psi}(u, v, i_\rho)$ が得られる．

$$\bar{\psi}(u, v, i_\rho) = \psi(u, v, Q_M^{-1}(i_\rho)). \tag{10}$$

u, v, i_ρ は有限の範囲内の整数しか取りえないため，$\bar{\psi}(u, v, i_\rho)$ は事前に計算が可能である．図 3 に $\bar{\psi}(u, v, i_\rho)$ を図示した．$\bar{\psi}(u, v, i_\rho)$ を用いると，式 (6) を次のように置き換えることができる．

$$k = \bar{\psi}(x - p_x, y - p_y, i_\rho) \tag{11}$$

次に，式 (7) をルックアップテーブルで置き換えることを考える．主軸を求める処理において，M 段階に量子化された勾配方向 $i_\theta(x, y)$ が得られているが，これは式 (7) を計算するときに再利用が可能である．そして，この再利用は次の近似を導入することで，ルックアップテーブルにより実現できる．

$$\begin{aligned} l &= Q_L(\theta(x, y) - \rho) \\ &\approx Q_L(Q_M^{-1}(i_\theta(x, y) - i_\rho)) = \chi(i_\theta(x, y), i_\rho) \end{aligned} \tag{12}$$

式 (12) では，$i_\theta(x, y)$ から i_ρ を引くことにより回転を補正し，合成関数 $Q_L(Q_M^{-1}(\cdot))$ により量子化レベル数を M から L に変換している．この近似は，$M \gg L$ であれば妥当である．$i_\theta(x, y), i_\rho$ は 0 から $M - 1$ の範囲内の整数しか取りえないため，関数 χ は事前に計算可能であり，$M \times M$ の大きさのテーブルとしてメモリ上に保持できる．従って，式 (6)(7) を式 (11)(12) で置き換えることで，大幅な高速化が達成される．(なお，正規分布 $G_{\sigma_1}, G_{\sigma_2}, G_{\sigma_s}$ もテーブル化が可能である．)

実験ではパラメータ $\{r_1, r_2, r_3, \sigma_1, \sigma_2, \sigma_s, M, L\}$ としてそれぞれ $\{3, 10, 20, 15, 10, 3, 40, 8\}$ を用いた．この設定の場合，$8 \times 17 = 136$ 次元の局所特徴量が得られる．

3. Supervised sparse hashing

3.1 コスト関数

抽出された局所特徴量 $\mathbf{d} \in \mathbb{R}^D$ は，次式に示す Binary hashing 関数によりバイナリコード $\mathbf{b} \in \{0, 1\}^B$ に変換する．

$$\mathbf{b} = (\mathrm{sgn}(\mathbf{W}^\top \mathbf{d}) + 1)/2 \tag{13}$$

ここで D は局所特徴量の次元数，B はビット長，$\mathbf{W} \in \mathbb{R}^{D \times B}$ は変換行列である．バイナリコード間の距離は，ハミング距離をビット長で正規化した値で与えられる．

$$D_\mathrm{h}(\mathbf{b}_u, \mathbf{b}_v) = \frac{1}{B} \parallel \mathbf{b}_u - \mathbf{b}_v \parallel_1 \tag{14}$$

Supervised sparse hashing では，変換前の局所特徴量間の距離と，変換後のバイナリコード間の距離がなるべく一致するように $\mathbf{W}$ を学習することを考える．そこで，次のようにコスト関数を定義する．

$$C(\mathbf{W}) = \sum_{(u,v) \in P} (D_\theta(\mathbf{d}_u, \mathbf{d}_v) - D_\mathrm{h}(\mathbf{b}_u, \mathbf{b}_v))^2 \tag{15}$$

ここで P は学習用ペアの集合である．$D_\theta(\mathbf{d}_u, \mathbf{d}_v)$ はベクトル間の角度を π で正規化したものであり，次のように定義される．

$$D_\theta(\mathbf{d}_u, \mathbf{d}_v) = \arccos\left(\frac{< \mathbf{d}_u, \mathbf{d}_v >}{||\mathbf{d}_u||\ ||\mathbf{d}_v||}\right)/\pi \tag{16}$$

実行速度が重視されるアプリケーションにおいては，式 (13) を高速に計算しなければならない．バイナリコード化のボトルネックは $\mathbf{W}^\top \mathbf{d}$ の計算部分であり，$D \times B$ 回の掛算と $(D-1) \times B$ 回の加算を要する．そこで Sparse random projections [14], [15] の考え方に基づき，$\mathbf{W}$ が疎で整数であるという制約条件を導入する．すなわち，$\mathbf{W}$ の要素は $\{-1, 0, 1\}$ のいずれかの値をとり，非ゼロの要素数が S 個であるという制約の下で，$C(\mathbf{W})$ を最小化する．これにより，$\mathbf{W}^\top \mathbf{d}$ は合計 S 回の加算と引算で計算できるようになる．S を小さく設定することで，変換に要する計算時間を十分小さくできる．次章の実験で示すが，興味深いことに $\mathbf{W}$ の要素のうち 90 %をゼロに設定した場合でも，性能を落とすことなくバイナリコード化ができることが分かった．これはバイナリコード変換に要する負荷を大幅に減らすことに繋がるため，応用上重要な性質であるといえる．

3.2 Greedy アルゴリズムによる最適化

コスト関数 $C(\mathbf{W})$ は非線型の不連続関数であるから，Levenberg-Marquardt 法や Gauss-Newton 法などのような勾配に基づく手法が適用できない．そこで，Greedy アルゴリズムによる最適化アルゴリズムを適用した．我々の戦略では，ランダムに 2 つの要素を $\mathbf{W}$ から選択し，この 2 つの要素を最適な値に置き換えることを考える．

$\mathbf{W}$ は $w_{ij} \in \{-1, 0, 1\}$ および $\sum |w_{ij}| = S$ という 2 つの制約条件を満たす必要があるが，この制約のおかげで，選んだ 2 つの要素が取りえる値は少ない組み合わせに限られる．従って，次に示す Greedy アルゴリズムで最適化が行える．

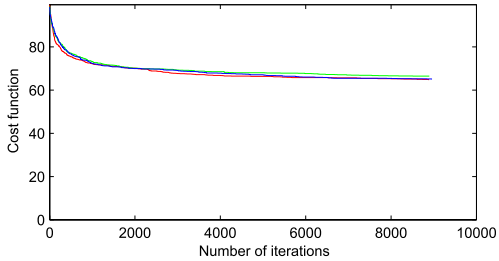
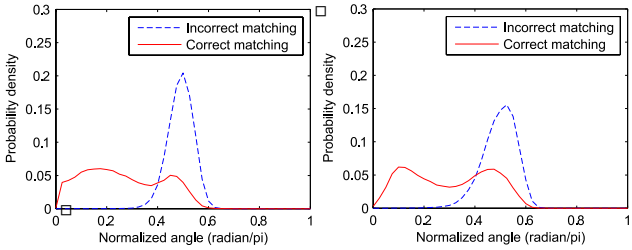


図 4　コスト関数の最小化の様子．25,000 個の学習ペアを用いた．$B = 64$，$\mathbf{W}$ の要素のうち 90%はゼロ．



(a) CARD　　(b) SIFT

図 5　局所特徴量間の角度の確率密度関数．x 軸は角度 (ラジアン) を π で正規化した量．

Problem:　$w_{ij} \in \{-1, 0, 1\}$, $\sum |w_{ij}| = S$ という制約の下で $C(\mathbf{W})$ を最小化する．

Algorithm:

1. 次の 2 つの処理 (a)(b) により，$\mathbf{W}$ を初期化．
   (a) 全ての要素を 1,-1 でランダムに初期化．
   (b) ランダムに $(DB - S)$ 個の要素を選び，0 に設定．
2. ランダムに 2 つの要素 w_{uv}, w_{pq} を選択．
3. w_{uv}, w_{pq} を更新．
   (a) (w_{uv}, w_{pq}) が両方とも非ゼロの場合
   $(1, -1)$, $(-1, 1)$, $(1, 1)$, $(-1, -1)$ をチェックし，
   $C(\mathbf{W})$ を最小化する組み合わせを採用．
   (b) (w_{uv}, w_{pq}) が両方ともゼロの場合
   ステップ 2 に戻って繰り返し．
   (c) (w_{uv}, w_{pq}) のいずれか一方がゼロの場合
   $(0, 1)$, $(0, -1)$, $(1, 0)$, $(-1, 0)$ をチェックし，
   $C(\mathbf{W})$ を最小化する組み合わせを採用．
4. 収束するまでステップ 2 に戻って繰り返し．

図 4 に $C(\mathbf{W})$ が減少する様子を示した．3 つの異なる初期値から始めた結果を図示しているが，どの場合においても $C(\mathbf{W})$ が正しく減少していることが分かる．以上により，局所特徴量はバイナリコードへ変換できる．なお，多くの Binary hashing 手法と同様に，実際に適用するときには $\mathbf{d}$ の平均を求め，事前に重心を原点にずらしておく．

4. 実験結果

本節では，提案する局所特徴量をバイナリコード化した場合としなかった場合の両方について評価を行う．以下，$\mathbf{W}$ におけるゼロ要素が占める割合 $(1 - \frac{S}{DB})$ を「ゼロ要素率」と記述することにし，ゼロ要素率が性能にどのように影響するかを議論する．

4.1 弁別性

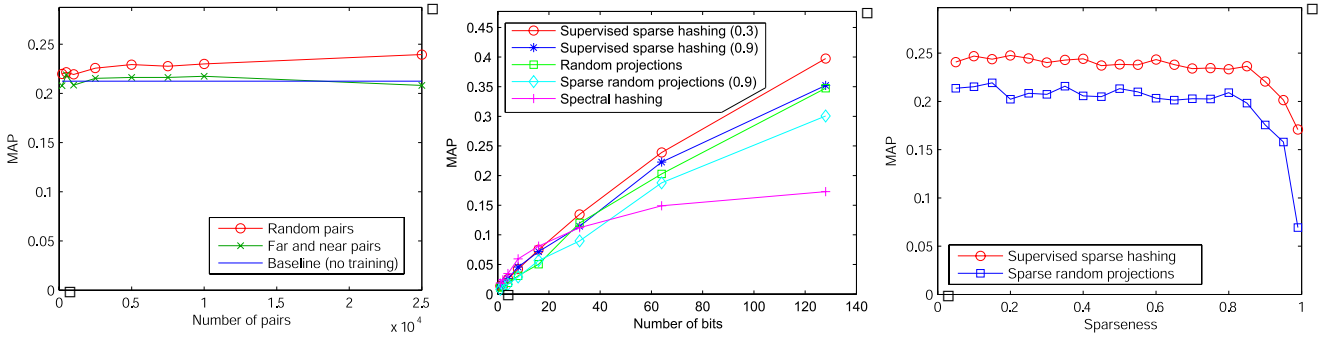
まず，バイナリコード化する前の，局所特徴量の弁別性能を確かめる．そこで図 5 に，局所特徴量間の角度の確率密度関数を示した．左が CARD，右が SIFT の結果であり，SIFT では DoG フィルタをキーポイント検出器として用いた．2 画像間の正確な Homography 行列が既知である Mikolajczyk らによるデータベース [21], [22] を用いてマッチング結果として正しいペアと正しくないペアを別々に収集し，それぞれから得た確率密度関数を独立に示した．

CARD と SIFT から得られた確率密度関数は非常に似ていることが分かる．正しくないマッチングペアの確率密度分布は分散の小さい正規分布とみなせる．一方，正しいマッチングペアの確率密度分布は広い分布を持つようである．これは，Mikolajczyk らによるデータベースにはアフィン歪みや射影歪みなどのような複雑な変形が含まれており，正しいマッチング位置であっても，局所特徴量が十分な弁別性能を発揮できなかったことによると考えられる．これは，SIFT,CARD ともに同様である．

正しくないマッチングペアの確率密度分布は正規分布で近似できるため，適切なしきい値 t よって誤マッチングを棄却できると考えられる．CARD の場合，この正規分布の平均と標準偏差はそれぞれ 0.49 および 0.05 であるから，3 シグマの法則に従うと，$t = 0.49 - 3 \times 0.05 = 0.34$ と設定することで 99.7%の誤マッチングが棄却できると考えられる．

4.2 バイナリコードの性能

局所特徴量の隣接関係は，バイナリコード変換前と変換後で保存されねばならない．たとえば，バイナリコード変換前の空間において最近傍に位置する局所特徴量は，変換後のハミング空間においても最近傍であることが望まれる．そこで，Mean Average Precision (MAP) によって Binary hashing による近傍探索の性能を評価した．MAP は Precision-Recall 曲線によって囲まれる領域の面積を近似したものであり，次の手順によって計算した．まず Caltech101 データセット [23] の各カテゴリから 1 枚ずつ画像を抽出し，136 次元の局所特徴量を計算した．ここから 50,000 個の学習用サンプルと 10,000 個のテスト用サンプルを，重複の無いようにランダムに抽出した．それぞれのテストサンプルにおいて，学習用サンプルとの距離 (式 (16)) を計算し，距離が t 以下の候補を「真の近傍」として定義した．次に，学習用サンプルを用いて $\mathbf{W}$ を学習し，全ての学習用サンプルとテスト用サンプルをバイナリコードへ変換した．それぞれのテストサンプルについて，学習用サンプルをハミング距離 (式 (14)) の近いものから順に並べたランキングリストを



(a) MAP vs. 学習ペアの数　(b) MAP vs. ビット数　(c) MAP vs. ゼロ要素率

図 6　バイナリコードの性能

作成した．このランキングリストの上位に真の近傍が正しく現れるか否かを MAP により評価した．

図 6(a) は縦軸に MAP，横軸に学習ペアの数を示したものである．Supervised sparse hashing で用いる学習ペアとしては，学習用サンプルからランダムに選択する方法と，ペア間の距離が近いものと遠いものを優先的に選択する方法の 2 種類を試した．また，学習を用いない Sparse random projections の結果も比較対象として示した．いずれの場合においても，ゼロ要素率は 0.3，ビット長は 64 である．図 6(a) に示されている通り，学習ペアをランダムに選択したときに限り，学習ペアを増やすと MAP が向上した．すなわち，学習によって Sparse random projections の最近傍探索性能が改善できることが分かった．ただし，学習ペアの選び方としてペア間の距離を活用した方法では MAP の改善は見られなかった．学習ペアの選択方法は Supervised sparse hashing の性能を決める重要な検討項目であるといえる．ここでは単純にランダムに選択することを推奨する．

図 6(b) は Supervised sparse hashing, Spectral hashing [10], Random projections [12], [16], Sparse random projections [14], [15] の比較結果を示したものである．Li [15] らが $\mathbf{W}$ の要素を $1 - 1/\sqrt{D}$ の確率でゼロに設定することを推奨していることに基づき，Sparse random projections ではゼロ要素率を $(1 - 1/\sqrt{136}) \approx 0.9$ とした．また，Supervised sparse hashing では 2 種類のゼロ要素率 0.3，0.9 を用いた．図 6(b) から分かる通り，Supervised sparse hashing は高いゼロ要素率であっても，広い範囲のビット長において従来手法よりも高い MAP を達成できていることが分かる．

Fig. 6(c) は縦軸に MAP，横軸にゼロ要素率を示したものである．興味深いことに，Supervised sparse hashing のゼロ要素率を 0.9 付近まで高めても，MAP が低下しないことが分かった．学習を用いない Sparse random projections の場合，Li [15] らによって同様の知見が既に報告されているが，学習を導入した場合でも同じ効果が認められるようである．従ってゼロ要素率を 0.9 に設定することで，近傍探索性能を損なうことなく，バイナリコード化を高速に処理できるようになる．これは応用上，非常に重要な利点であるといえる．

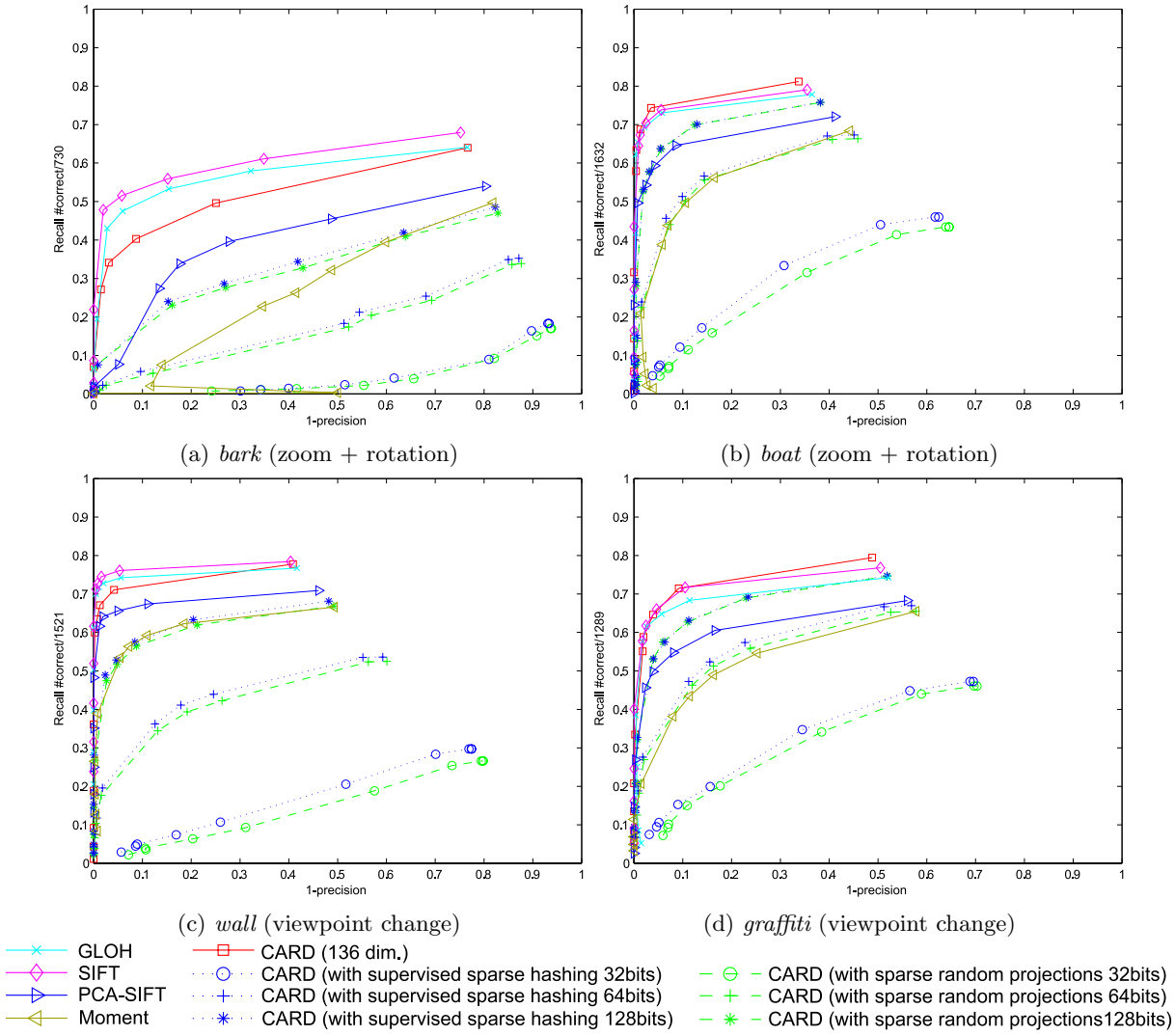
4.3　2 視点のマッチング性能

Mikolajczyk [21], [22] らの方法に従って 2 視点間のマッチング性能を評価した．Mikolajczyk らはマッチング性能評価用のデータベースとして 8 種類のカテゴリを提供しており，それぞれカテゴリは同一の対象を異なる条件で撮影した 6 枚の画像 No.1〜No.6 で構成されている．そして，No.1 と残りの画像 No.2〜No.6 との間の Homography 行列がそれぞれ与えられている．

評価実験では，No.1 と No.2 の画像間の対応点を，最も近い局所特徴量と 2 番目に近い局所特徴量の距離の比に対してしきい値処理を適用することで求め，Homography 行列によってその対応点が正しいか否かを検証した．検証結果に従って 1-precision vs. recall 曲線をプロットすることにより，局所特徴量の性能を評価した．$\mathbf{W}$ を学習するために，8 種類のカテゴリの画像 No.4〜No.6 から局所特徴量を抽出し，ランダムに 50,000 個の学習サンプルを選定した．ここからランダムに 25,000 ペアを抽出し，$\mathbf{W}$ を学習した．ゼロ要素率は 0.9 とし，異なる 3 種類のビット長 $B = 32, 64, 128$ について評価した．

図 7 に 4 つカテゴリ *bark*, *boat*, *wall*, *graffiti* についての結果を示した．カテゴリ *bark*, *boat* は回転およびスケール変化の評価用であり，カテゴリ *wall*, *graffiti* は射影変換を含む視点位置変化の評価用として用意されている．実験では，GLOH, SIFT, PCA-SIFT [18], モーメント不変量 [21], CARD を比較した．提案手法である CARD については，バイナリコード化を適用した場合としなかった場合の両方について評価した．バイナリコード化の手法としては，学習の効果を確かめるために Supervised sparse hashing と Sparse random projections の両方を適用した．いずれの方法においても，ゼロ要素率として 0.9 を設定した．これらのバイナリコード化手法では乱数が用いられているため，同じ実験を 10 回繰り返して得た 1-Precision と Recall の平均をプロットした．本実験では，いずれの場合においても 2.1 節で記載したキーポイント検出器を用いた．

バイナリコード化前の CARD の結果を見ると，全ての



(a) *bark* (zoom + rotation)　(b) *boat* (zoom + rotation)

(c) *wall* (viewpoint change)　(d) *graffiti* (viewpoint change)

図 7　2 視点間のマッチング性能を示す 1-Precision vs. Recall 曲線.

カテゴリにおいて GLOH, SIFT に匹敵していることが分かる．バイナリコード化を適用した後でも，128bit あればモーメント不変量よりも性能が高く，また PCA-SIFT に匹敵する性能が得られていることが分かる．また，同じビット長における Supervised sparse hashing と Sparse random projections の結果を比較すると，$\mathbf{W}$ を学習することによりマッチング性能が改善できていることが読み取れる．

図 8 に SIFT, SURF, CARD のスケールおよび回転不変性の性能評価結果を示した．SIFT では DoG，SURF では Harr-like 特徴量をキーポイント検出器として適用した．Mikolajczyk によるデータベースのカテゴリ *graffiti* の画像 No.1 を変形させ，回転およびスケール変換させた画像セットを生成した．オリジナルの画像と変形後の画像間において，最近傍となる局所特徴量を探索し，RANSAC による Homography 行列推定処理を適用することでアウトライアを除去した．最近傍探索で得られたマッチング数に対するインライア数の割合を求め，これを図 8 にプロットした．図に示されている通り，提案手法では短いビット長においても SIFT, SURF に匹敵する性能が得られていることが分かる．

4.4 計算時間

本節では，Intel Core 2 Duo 2.66GHz を用いて CARD の計算時間を評価した結果について述べる．1 つの局所特徴量あたりの計算時間を図 9 に示した．ここでは，キーポイント検出における計算時間は排除して評価している．x 軸は CARD の計算時間であり，SIFT の場合を 100%として正規化して示してある．なお，SIFT を計算するソフトウェアとして Hess による実装 [24] を用いた．図中上段は，バイナリコード化手法として Random projections を用いた結果であり，下段はゼロ要素率を 0.9 として Supervised sparse hashing を用いた結果である．ここではビット長 $B = 128$ とした．疎行列によって大幅な高速化が達成されていることが分かる．局所特徴量抽出の部分もまた，ルックアップテーブル化によって大幅に高速化されているため，全体で SIFT の約 6%の計算量 (約 16 倍) の高速化を達成できた．

最後に，表 1 にキーポイント検出器を含めた総合的な評価結果をまとめておく．SURF の実装としては OpenCV

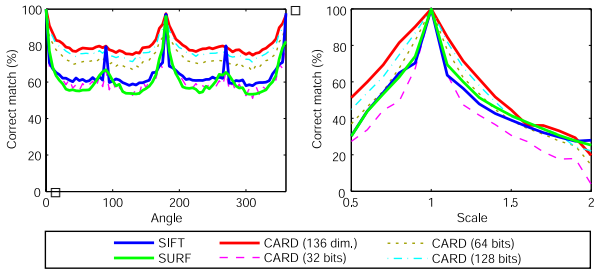


図 8　スケールおよび回転不変性.

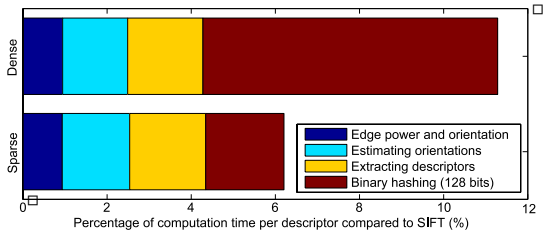


図 9　1 つの局所特徴量あたりの計算時間 (SIFT を 100%として表示).

表 1　SIFT,SURF,CARD の計算時間 (画像サイズ 640 × 480)

	SIFT	SURF	CARD (128 bits)
計算時間 (msec)	1030.5	206.1	129.3
キーポイント数	1553.8	1267.8	1658.0

を用いた．実験では，640 × 480 の大きさの画像を用いた．5 つの異なる画像に対して SIFT, SURF, CARD を適用し，計算時間とキーポイント数の平均値をまとめた．CARD は従来手法よりも高速であり，SIFT の約 8 倍，SURF の約 1.6 倍という結果を得た．

5. おわりに

本論文では，高速に計算可能であり，128 ビット程度のバイナリコードで表現可能な局所特徴量 CARD を提案した．提案手法は少ない計算量・メモリ量で動作できるため，車載向け CPU や携帯端末などといった，低スペック機器に向いていると考えられる．残る課題としては，Supervised sparse hashing の最適化手法の改善や，アフィン変形への対応などが挙げられる．これらの課題を解決することで，応用範囲を拡大できると期待される．

文　献

[1] D. G. Lowe: “Distinctive image features from scale-invariant keypoints”, IJCV, **60**, pp. 91–110 (2004).

[2] N. Snavely, S. M. Seitz and R. Szeliski: “Photo tourism: exploring photo collections in 3D”, SIGGRAPH, pp. 835–846 (2006).

[3] D. Nister and H. Stewenius: “Scalable recognition with a vocabulary tree”, CVPR, pp. 2161–2168 (2006).

[4] S. Lazebnik, C. Schmid and J. Ponce: “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories”, CVPR, pp. 2169–2178 (2006).

[5] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool: “Speeded-up robust features (SURF)”, Computer Vision and Image Understanding, **110**, pp. 346–359 (2008).

[6] G. Takacs, V. Chandrasekhar, S. Tsai, D. Chen, R. Grzeszczuk and B. Girod: “Unified real-time tracking and recognition with rotation-invariant fast features”, CVPR, pp. 934–941 (2010).

[7] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond and D. Schmalstieg: “Pose tracking from natural features on mobile phones”, ISMAR, pp. 125–134 (2008).

[8] M. Calonder, V. Lepetit, P. Fua, K. Konolige, J. Bowman and P. Mihelich: “Compact signatures for high-speed interest point description and matching”, ICCV, pp. 357–364 (2009).

[9] S. Taylor, E. Rosten and T. Drummond: “Robust feature matching in 2.3μs”, CVPR Workshop, pp. 15–22 (2009).

[10] Y. Weiss, A. Torralba and R. Fergus: “Spectral hashing”, NIPS, pp. 1753–1760 (2008).

[11] J. Wang, S. Kumar and S.-F. Chang: “Semi-supervised hashing for scalable image retrieval”, CVPR, pp. 3424–3431 (2010).

[12] K. Min, L. Yang, J. Wright, L. Wu, X.-S. Hua and Y. Ma: “Compact projection: Simple and efficient near neighbor search with practical memory requirements”, CVPR, pp. 3477–3484 (2010).

[13] J. Wang, S. Kumar and S.-F. Chang: “Sequential projection learning for hashing with compact codes”, Proceedings of the international conference on Machine Learning (2010).

[14] D. Achlioptas: “Database-friendly random projections”, Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 274–281 (2001).

[15] P. Li, T. J. Hastie and K. W. Church: “Very sparse random projections”, Proceedings of the international conference on Knowledge Discovery and Data mining, pp. 287–296 (2006).

[16] M. X. Goemans and D. P. Williamson: “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”, Journal of the ACM, **42**, pp. 1115–1145 (1995).

[17] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk and B. Girod: “CHoG: Compressed histogram of gradients a low bit-rate feature descriptor”, CVPR, pp. 2504–2511 (2009).

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

[19] Y. Dufournaud, C. Schmid and R. Horaud: “Matching images with different resolutions”, CVPR, pp. 612–618 (2000).

[20] J. Shi and C. Tomasi: “Good features to track”, CVPR, pp. 593–600 (1994).

[21] K. Mikolajczyk and C. Schmid: “A performance evaluation of local descriptors”, PAMI, **27**, pp. 1615–1630 (2005).

[22] “Affine covariant regions datasets”, http://www.robots.ox.ac.uk/ vgg/data/data-aff.html.

[23] R. F. L. Fei-Fei and P. Perona: “Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories”, CVPR Workshop, p. 178 (2004).

[24] R. Hess: “An open-source SIFTLibrary”, Proceedings of the international conference on Multimedia, pp. 1493–1496 (2010).