Vol.2015-CVIM-195 No.2
2015/1/22

行列分解と早期棄却による多クラス物体検出の高速化

黒川 貴都[1,a)]　山内 悠祠[1,b)]　安倍 満[2,c)]　山下 隆義[1,d)]　藤吉 弘亘[1,e)]

概要：物体検出では，画像をラスタスキャン時に多数の検出ウインドウを検出対象クラスと非検出対象クラスに識別する必要があるため．非常に高い計算コストを要する．この問題を解決するために，二クラスの線形識別器の高速化手法として，SVM の重みベクトルを二値基底行列とスケール係数ベクトルに分解することで，特徴量と重みベクトルの内積を近似的に計算することで高速化する手法が提案されている．画像の見えが似ている複数の検出対象の画像から one-vs.-rest により多クラス識別器を構築し，ベクトル分解法を適用すると各クラスの二値基底行列は類似した行列が得られる．そこで，本研究では多クラス物体検出において共通化した二値基底行列に分解する行列分解法を提案する．共通した二値基底行列を用いることで，内積を近似的に計算する際の処理を大幅に削減することができる．さらに，カスケード構造の識別器に早期棄却を導入することで多クラス識別を高速化する手法を提案する．評価実験より，各クラスの重みベクトルが平均相関値 0.38 のとき，識別演算にかかる計算時間を約 21 倍に高速化できることを確認した．

1. はじめに

顔や歩行者等の物体検出では，検出対象と非検出対象の 2 クラス識別を対象としている．しかしながら，実世界にはさまざまな物体が存在しており，これら多数のクラスを対象とした物体検出の実現が求められている．Image Net[1] や Caltech256[2] 等のベンチマークでは，200 から 1000 クラスの物体検出を対象としている．多クラス物体検出では，画像上を検出ウィンドウの位置と大きさを変えながら網羅的にラスタスキャンし，得られた多数の検出ウィンドウを対象クラスもしくは非対象クラスに識別する必要がある．各検出ウィンドウを識別するには，画像から特徴量の抽出と多クラス識別器により識別処理を要する．そのため，検出対象のクラス増加に伴い，識別に必要な計算時間が線形的に増加するという問題がある．

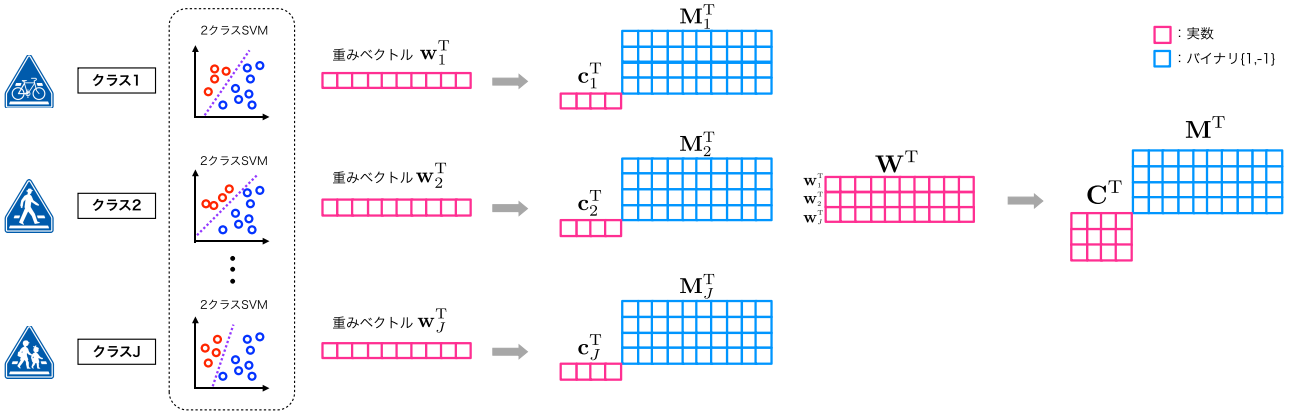
この問題を解決するには，特徴抽出の高速化と識別計算の高速化が重要であり，これまでに多くの高速化に関する研究が取り組まれてきた [3]-[14]. 高速な特徴量の計算手法として，積分画像 [3] や積分ヒストグラム [4] がある．これらは，任意の領域における輝度値や勾配強度の総和を高速に計算することができる．さらに，Dollár らは，積分画像や積分ヒストグラムから抽出できる特徴量のみを用いることにより，高速な人検出を実現した [5].

識別処理の高速化手法は，計算の打ち切りと識別計算の近似による演算量の削減の 2 つの方法が提案されている．計算の打ち切りによる識別の高速化では，カスケード構造の識別器を用いた Viola 等の顔検出法がある [3]. この手法は，Adaboost[6] により学習した強識別器をカスケード状に並べ，各段で得られた計算結果に基づき早期に棄却する．

一方，近似的に計算することで演算量を削減する方法には, ハッシングを用いた方法 [7] と識別器の計算を近似する方法が提案されている [8]，[11]-[14]. ハッシングを用いた方法として，Dean らは抽出された特徴量からハッシングを用いてどのクラスの識別器に入力すべきかを絞り込む手法を提案し，高速化を実現した [7]. 識別器の計算を近似する方法として，Hare らは識別器の重みベクトルを 2 値の基底行列とスケール係数ベクトルに分解し，識別時の内積計算を近似的に高速計算する方法を提案した [8]．ベクトル分解法では，入力特徴が二値ベクトルという条件を満たせば，学習した識別器の重みベクトルを分解するため, 識別器を再学習する必要がない．後藤らは，HOG 特徴量 [9] を 2 値化した B-HOG 特徴量 [10] とベクトル分解法を適用した識別器の早期判定による高速な人検出法 [11] を提案した．Cheng らはベクトル分解法を適用した識別器により，300FPS で動作する Objectness 検出を実現した [12].

また，より少ない基底数で効率的に分解する方法も提案

[1] 中部大学 春日井市
Department of Computer Science, Chubu University, 1200 Matsumoto, Kasugai, Aichi, 487–8501 Japan
[2] 株式会社デンソーアイティーラボラトリ
Denso IT Laboratory, Shibuya CROSS TOWER 28th Floor 2-15-1 Shibuya, Shibuya-ku, Tokyo, 150–0002 Japan
a) kuro@vision.cs.chubu.ac.jp
b) yuu@vision.cs.chubu.ac.jp
c) manbai@d-itlab.co.jp
d) yamashita@cs.chubu.ac.jp
e) hf@cs.chubu.ac.jp



(a) ベクトル分解　　(b) 提案手法 (行列分解)

図 1　多クラス識別器における重みベクトルの分解

されている [13][14]. Yamauchi らは，二値基底行列をしらみつぶし探索により最適化することで，Hare らの分解法よりも分解時の誤差を削減した [13]. Ambai らは二値基底行列を三値基底行列に拡張することで，より小さな基底数で分解誤差を小さくする方法を提案した [14].

これらの二値ベクトル分解法は，one-vs.rest により構築した多クラス識別器にも容易に適用可能である．二値ベクトル分解法を多クラス識別器に適用すると，クラス毎の二値基底行列が算出することになる．従って，多クラス識別では，入力特徴ベクトルとの内積の計算回数がクラス数に比例して増加するという問題がある．例えば，Hare らのベクトル分解法を適用すると，クラスの数だけ二値基底行列とスケール係数ベクトルが得られる．識別時には，入力特徴ベクトルと各クラスの二値基底行列の内積の計算が必要となる．行列分解法では，図 1(b) に示すように，全てのクラスにおいて多クラス識別のための共通の二値基底行列を用いることで，演算回数を削減することが可能である．さらに，逐次的に行列分解することでカスケード構造となる識別器を構築し，識別時に早期棄却のアプローチを導入する．これにより，非検出対象クラスに対する演算を打ち切ることで識別演算の高速化が期待できる．

2. ベクトル分解法

物体検出に利用されている 2 クラス識別器である線形 Support Vector Machine(SVM)[15] は，D 次元の入力特徴ベクトルを $\mathbf{x} \in \mathbb{R}^D$，重みベクトルを $\mathbf{w} \in \mathbb{R}^D$ とすると，式 (1) により出力を計算できる．

$$F(\mathbf{x}) = \mathrm{sign}[\mathbf{w}^{\mathrm{T}}\mathbf{x} + b] \tag{1}$$

この識別関数 F の出力を閾値処理することで検出対象クラスと非検出対象クラスに識別する．

線形 SVM では，入力特徴ベクトルと重みベクトルの内積を計算する必要があるため，識別に多くの時間を要する．このような問題に対して，重みベクトルを二値基底行列とスケール係数ベクトルに分解し，内積を近似的に高速計算する方法 [8][13][14] が提案されている．本章では，実数ベクトルの分解について説明した後，従来の実数ベクトルの分解アルゴリズムについて述べる．そして，ベクトル分解法を多クラス識別に適用する方法と問題点について述べる．

2.1 実数ベクトルの分解と近似内積計算

式 (1) の線形 SVM の識別関数 $F(\mathbf{x})$ の重みベクトル $\mathbf{w}$ を，二値基底行列 $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k\} \in \{-1, 1\}^{D \times k}$ とスケール係数ベクトル $\mathbf{c} = \{c_1, c_2, \cdots, c_k\} \in \mathbb{R}^k$ に分解すると，識別関数 $F(\mathbf{x})$ は式 (3) のように近似することができる．

$$F(\mathbf{x}) \approx \mathrm{sign}[\mathbf{c}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{x} + b] \tag{2}$$

$$= \mathrm{sign}\left[\left(\sum_{i=1}^{k} c_i \mathbf{m}_i^{\mathrm{T}}\mathbf{x}\right) + b\right] \tag{3}$$

ここで，$\mathbf{x}$ は入力特徴ベクトルで $\mathbf{x} \in \{-1, 1\}^D$ の二値ベクトルで表される．k は基底数を表し，基底の数を増やすことで，識別関数 $F(\mathbf{x})$ の出力の値をより近似することができる．二値ベクトル間の内積計算は，ハミング距離の計算に置き換えることができるため，高速な演算が可能である．本章では，重みベクトル $\mathbf{w}$ を二値基底行列 $\mathbf{M}$ とスケール係数ベクトル $\mathbf{c}$ に分解する 2 つの方法について述べる．

2.2 Hare らのベクトル分解法

Hare らの分解法 [8] は，重みベクトル $\mathbf{w}$ を貪欲法により二値基底行列 $\mathbf{M}$ とスケール係数ベクトル $\mathbf{c}$ に分解する．二値基底ベクトル $\mathbf{m}$ とスケール係数ベクトル $\mathbf{c}$ を貪欲法により交互に求めるため，高速な分解が可能という特長がある．

Hare らの分解法では，各基底 $i = (1, 2, \ldots k)$ において残差 $\mathbf{r}$[*1] との誤差が最小となるスケール c_i と二値ベクトル

[*1] 基底 $i = 1$ のみ，$\mathbf{r} = \mathbf{w}$ とする．

$\mathbf{m}_i$ を繰り返し求める．

$$\mathbf{r} \leftarrow \mathbf{r} - c_i\mathbf{m}_i \tag{4}$$

$$\mathbf{m}_i = \mathrm{sign}(\mathbf{r}) \tag{5}$$

$$c_i = \mathbf{r}^{\mathrm{T}}\mathbf{m}_i/||\mathbf{m}_i||^2 \tag{6}$$

得られた二値ベクトル $\mathbf{m}_i$ とスケール係数 c_i から残差を式 (5) により更新し，予め決めた基底数 k に達するまで繰り返す．

2.3 Yamauchi らのベクトル分解法

Yamauchi らの分解法 [13] は，重みベクトル $\mathbf{w}$ をしらみつぶし探索により二値基底行列 $\mathbf{M}$ とスケール係数ベクトル $\mathbf{c}$ に分解する．二値基底ベクトル $\mathbf{m}$ とスケール係数ベクトル $\mathbf{c}$ をしらみつぶし探索により最適化するため，より高精度に近似できるという特長がある．

この分解法では，式 (7) を最小化する二値基底行列 $\mathbf{M}$ とスケール係数ベクトル $\mathbf{c}$ を求める．

$$||\mathbf{w} - \mathbf{Mc}||_2^2 \tag{7}$$

二値基底行列 $\mathbf{M}$ とスケール係数ベクトル $\mathbf{c}$ を同時に最適化することは難しいため，Hare らの方法と同様に交互に最適化する．スケール係数ベクトル $\mathbf{c}$ については，二値基底行列 $\mathbf{M}$ を固定し最小二乗法により求める．一方， 二値基底行列 $\mathbf{M}$ については，スケール係数ベクトル $\mathbf{c}$ を固定し，-1 と 1 の組み合わせの全てを試行し，式 (7) が最小となる二値基底行列 $\mathbf{M}$ を採用する．

2.4 多クラス識別への利用

2 クラス識別器を用いて多クラス識別器を構築する方法として，one-vs.-rest 法が提案されている．one-vs.-rest 法では，まずクラス j とそれ以外のクラスを識別する 2 クラス識別器を J クラス分の学習により，識別器を構築する．識別時には，学習した J クラス数の 2 クラス識別器に特徴ベクトルを入力しスコアを得る．そして，最も高いスコアを出力した識別器に対応したクラスを最終結果として出力する．この際，全ての識別器のスコアの値が小さい場合には，最終結果として非検出対象と出力する．

ベクトル分解法を one-vs.-rest により構築した多クラス識別器に適用した場合，図 1(a) のように二値基底行列 $\mathbf{M}$ がクラス毎に算出される．従って，多クラス識別では特徴ベクトル $\mathbf{x}$ と $\mathbf{Mc}$ の内積の計算回数がクラス数 J 回に増加するという問題が発生する．

3. 提案手法

各クラスの重みベクトル $\mathbf{w}$ 間の相関が高いと，各二値基底行列 $\mathbf{M}$ も類似すると考えられる．そこで，本研究では各クラスの重みベクトルを 1 つの行列 $\mathbf{W}$ として表し，全クラスで共通する二値基底行列 $\mathbf{M}$ とスケール係数行列 $\mathbf{C}$ に分解する行列分解法を提案する．共通化した二値基底行列を用いることにより，入力特徴ベクトル $\mathbf{x}$ と $\mathbf{M}$ の内積の計算が 1 回で済むため，高速な計算が期待できる．さらに，多クラス識別器をカスケード状に並べることにより，早期的に棄却するアプローチを導入することで高速化する．

3.1 行列分解法

まず，各クラスに SVM により学習し，重みベクトル $\mathbf{w}_j$ を求める． 次に，SVM により算出された J クラス分の重みベクトルを図 1(b) のようにスタックして，重み行列 $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_J\}$ を構築する． 行列分解では，構築した重み行列 $\mathbf{W}$ を二値基底行列 $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k\} \in \{-1, 1\}^{D\times k}$ と基底スケール係数行列 $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_J\} \in \mathbb{R}^{k\times J}$ に分解する．このとき，式 (8) を最小化するように二値基底行列 $\mathbf{M}$ と基底スケール係数行列 $\mathbf{C}$ を最適化する．

$$||\mathbf{W} - \mathbf{MC}||_F^2 \tag{8}$$

提案する行列分解法の流れを **Algorithm 1** に示す．**Algorithm 1** は $\mathbf{M}$ と $\mathbf{C}$ を交互に最適化し，式 (8) を最小化する．はじめに，$\mathbf{M}$ を $\{-1, 1\}$ の乱数，$\mathbf{C}$ を実数の乱数で初期化する．次に，$\mathbf{C}$ については $\mathbf{M}$ を固定し最小自乗法により求め， $\mathbf{M}$ については $\mathbf{C}$ を固定し-1 と 1 の組み合わせを試行し式 (8) が最小となる $\mathbf{M}$ を採用する．これを，式 (8) の値が収束するまで繰り返す．また，局所解に影響を抑制するために，初期値を変えて分解を L 回試行し，式 (8) が最小となる $\mathbf{MC}$ を採用する．

Algorithm 1 行列分解のアルゴリズム

Require: $\mathbf{W}$, L, k
  for $l = 1$ to L **do**
    $\mathbf{M}_l$ を $\{-1, 1\}$ の乱数により初期化.
    $\mathbf{C}_l$ を実数の乱数により初期化.
    repeat
      $\mathbf{M}_l$ を固定し, 式 (8) を最小化する $\mathbf{C}_l$ について最小二乗法により求める
      $\mathbf{C}_l$ を固定し, 式 (8) を最小化する $\mathbf{M}_l$ について {-1, 1} をしらみつぶし法により最適化する.
    until 式 (8) の値が収束
  end for
  $\{\mathbf{M}_l\}$ と $\{\mathbf{C}_l\}$ のうち式 (8) を最小とする二値基底行列と基底スケール係数行列を $\hat{\mathbf{M}}, \hat{\mathbf{C}}$ とする.
  return $\hat{\mathbf{M}}$, $\hat{\mathbf{C}}$

3.2 逐次的な行列分解によるカスケード構造の構築

行列分解は二値基底行列 M について −1 と 1 の全ての組み合わせを試行する．そのため，基底が 1 つ増やすと分解に必要な時間が 2 倍となるため，基底数が増加すると現実的に分解できない時間が必要となる．そこで，逐次的に行列分解しカスケード構造を構築することで，分解に必要な

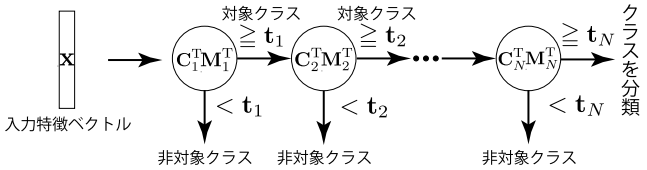


図 2 カスケード構造による早期棄却

時間を削減する．逐次的な行列分解によるカスケード構造の構築法を **Algorithm2** に示す．**Algorithm2** では，1 段目にて重み行列 $\mathbf{W}$ に対して行列分解法を適用し $\mathbf{M}_1, \mathbf{C}_1$ を算出する．2 段目以降では，前の段で得られた残差行列 $\mathbf{R}$ に行列分解法を適用し $\mathbf{M}_n, \mathbf{C}_n$ を算出し， $\mathbf{W}$ と $\mathbf{MC}$ の差を残差行列 $\mathbf{R}$ とする．

Algorithm 2 行列分解法を導入したカスケード型識別器の構築アルゴリズム

Require: $\mathbf{W}$, k, N, L
  $\mathbf{R} \leftarrow \mathbf{W}$
  for $n = 1$ to N **do**
    Algorithm 1 により $\mathbf{R}$ を $\hat{\mathbf{M}}_n$ と $\hat{\mathbf{C}}_n$ に分解する．
    $\mathbf{R} \leftarrow \mathbf{R} - \hat{\mathbf{M}}_n\hat{\mathbf{C}}_n$
  end for
  return $\{\hat{\mathbf{M}}\}$, $\{\hat{\mathbf{C}}\}$

3.3 近似内積計算と早期棄却による識別の高速化

Algorithm2 で構築したカスケード構造を持つ識別器により多クラス識別を行う．カスケード構造を持つ識別器に学習サンプルを例に入力し，内積の近似計算結果をグラフ化したものを図 3 に示す．図 3(a)，(b)，(d) は段数が多いほどより正確に識別できることがわかる．しかし，中には図 3(c) のように識別境界から大きく離れたネガティブサンプルは少ない段数で判定することが可能である．そこで，さらなる高速化のために識別境界から大きく離れた非検出対象クラスに対して近似内積計算を打ち切るため，早期棄却を導入する．識別時は，図 2 のように各段の近似内積計算で算出されたスコアとしきい値 $\mathbf{t} \in \mathbb{R}$ により非対象クラスの棄却する．

n 段目における各識別器の出力 $s_{n,j} \in \mathbb{R}$ は式 (9) の近似内積計算により算出する．

$$\mathbf{s}_{n,j} = \sum_{i=1}^{n} \mathbf{C}_i^{\mathrm{T}}\mathbf{M}_i^{\mathrm{T}}\mathbf{x} \tag{9}$$

$\mathbf{M}_i^{\mathrm{T}}\mathbf{x}$ は 2 値同士の内積となるので，$\mathbf{x}$ の次元数を D としたとき式 (10) で計算することができる．

$$\mathbf{M}_i^{\mathrm{T}}\mathbf{x} = D - 2\mathrm{HammingDistance}(\mathbf{M}_i, \mathbf{x}) \tag{10}$$

式 (10) の HammingDistance は XOR 演算とビットカウントで計算できるため，高速な演算が可能となる．ビットカウントは Streaming SIMD Extensions(SSE)4.2 より実装されている POPCNT 関数を使用することで高速な演算が可能となる．

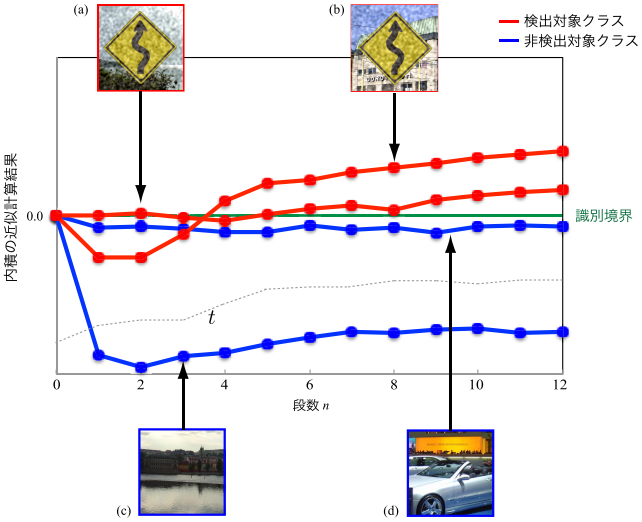


図 3 カスケード構造を持つ識別器による内積の近似計算結果

非対象クラスの棄却は n 段目で近似内積計算により算出されたスコア $\mathbf{s}_n$ をしきい値処理することにより決定する．このとき，各段のしきい値 $\mathbf{t}$ は学習に使用したポジティブサンプルの特徴量を入力し算出されたスコアの中から最小となる値を使用する．入力画像を I，入力画像から得られる検出ウィンドウの総数を L としたときの近似内積計算と早期棄却による物体検出を **Algorithm3** に示す．

Algorithm 3 提案手法による物体検出

Require: $\mathbf{M}$, $\mathbf{C}$, I, N
  for $l = 1$ to L **do** //l：検出ウィンドウの数
    検出ウィンドウ $I(l)$ から特徴ベクトル $\mathbf{x}_l$ を抽出．
    for $n = 1$ to N **do**　//N：カスケード型識別器の段数
      $\mathbf{s}_n = \sum_{i=1}^{k_n} \mathbf{C}_i^{\mathrm{T}}\mathbf{M}_i^{\mathrm{T}}\mathbf{x}_l$
      if $\mathbf{s}_n < \mathbf{t}_n$ **then**
        $y_l = 0$
        break //近似計算の打ち切り
      end if
    end for
    $y_l = \arg\max_{j \in J} \mathbf{s}_{N,j}$
  end for
  return $y_1, y_2, \ldots, y_L$

4. 評価実験

評価実験では，提案手法の有効性を確認するために従来の 2 値分解法であるベクトル分解法 [8][13] と比較する．

4.1 データセット

評価実験では以下の 3 種類の実験により提案手法の有効性を確認する．実験 1 では少ないクラスの識別を対象とし，図 4(a) に示す 4 クラスの分類問題を対象とする．実験 2 では図 4(b) に示すように 27 クラスの分類問題を対象とする．

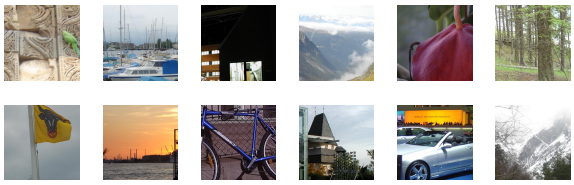
(a)4 クラスの標識画像

(b)27 クラスの標識画像

(c)24 クラスの標識画像

図 4 各実験で対象とする標識画像

(a) ポジティブサンプル



(b) ネガティブサンプル

図 5 ポジティブ画像とネガティブ画像

実験 3 では，図 4(c) に示すように，クラス間で形状が異なる 24 クラスの分類問題を対象とする．

学習サンプルのポジティブサンプルには，これらの標識画像に幾何変化と照明変化，ノイズを付与し，合成したものを用いる．図 5(a) に生成したポジティブサンプルの一例を示す．ネガティブサンプルは，図 5(b) に示すように背景画像から切り出したパッチ画像を使用する．学習には各ポジティブサンプル 500 枚とネガティブサンプル 5,000 枚を使用する．識別精度の評価時にはポジティブサンプル 5,000 枚とネガティブサンプル 200 枚を使用する．

4.2 実験概要

多クラスにおけるベクトル分解法と提案手法の行列分解法による誤差を比較する．実験に用いる識別器は SVM とし，データセットの画像から抽出した 1,158 次元の B-HOG 特徴量 [10] を使用する．また，識別精度と 1 枚あたりの識別時間を比較する．識別時間の比較では，CPU:Intel Xeon CPU X7542@2.67GHz, RAM:256GB の PC を用いる．評価実験では，Hare らの分解法と Yamauchi らの分解法を適用した one-vs.-rest による多クラス識別器と比較する．

4.3 実験結果

提案手法の有効性を示すため，分解による誤差，識別精度，識別時間について従来法と比較する．

4.3.1 4 クラス識別器における近似誤差と識別時間

図 4(a) から生成した 4 クラスの識別問題において，クラスあたりの基底数に対する二乗誤差を図 6(a) に示す．図 6(a) より，1 クラスあたりの基底数が 3 のとき，提案手法はベクトル分解法 [13] より誤差を 64.4%に減少することができた．

早期棄却の有効性を確認するために識別時間を比較する．分解前の SVM により算出した重み行列 $\mathbf{W}$，ベクトル分解法 [8][13]，行列分解のみ，行列分解 + 早期棄却を比較する．識別時間と False Positive Rate(FPR)=0.01 であるときの True Positive Rate(TPR) を表 1 に示す．表 1 より，提案手法は分解前に比べ約 13 倍高速に，さらにカスケード構造による早期棄却を導入することで約 21 倍高速化することができた．

実画像での識別結果と棄却された領域を図 7 に示す．図 7(a) では紫の矩形で囲われた領域が検出された領域を表す．矩形の上辺は検出されたクラスを表しており，それぞれのクラスは図 7 の上部に示す．図 7(b) では早期棄却された領域を水色で示している．図 7 より，背景領域の多くが早期棄却されていることことがわかる．

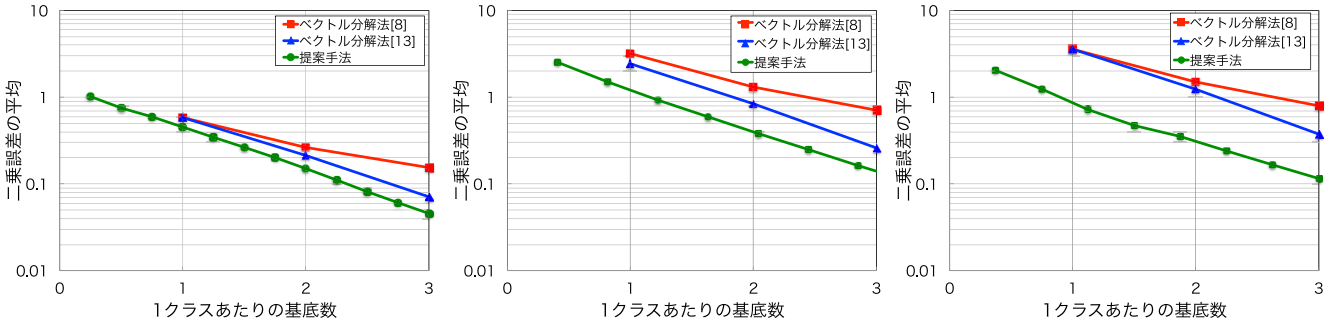
4.3.2 27 クラス識別器における近似誤差と識別時間

図 4(b) から生成した 27 クラスの識別問題において，提案手法の基底数を 11 としたときの，クラスあたりの基底数に対する二乗誤差を図 6(b) に示す．図 6(b) より，1 クラスあたりの基底数が 5 のとき，提案手法はベクトル分解法 [13] より誤差を 44.3%減少させることができた．

表 1 より提案手法は分解前に比べ約 4.2 倍高速化，さらに早期棄却を導入することで約 31 倍高速化することができた．実験 1 の 4 クラスの実験結果と比較すると，クラス数が多いほど提案手法の効果が高いことがわかる．

4.3.3 24 クラスの標識画像における近似誤差

図 4(c) のクラス間で形状が異なる標識画像から生成した 24 クラスの識別問題において，提案手法の基底数を 11 としたときのクラスあたりの基底数に対する二乗誤差を図 6(c) に示す．結果より，形状が異なる標識画像においても，



(a)4 クラス識別器における近似誤差　(b)27 クラス識別器における近似誤差　(c)24 クラス識別器における近似誤差

図 6　重み行列の分解による近似誤差

(a) 識別結果　(b) 棄却結果

図 7　実画像における識別結果と棄却領域

実験 1, 2 と同様に提案手法の誤差が最も小さいことがわかる．また，表 1 より提案手法は分解前に比べて約 4 倍，さらに早期棄却を導入することで約 8.7 倍高速化することができた．しかし，実験 2 の 27 クラスの結果と比べると，クラス数が減少したのにも関わらず，計算時間の短縮率が悪い結果となった．

4.4　考察

4.4.1　分解法に関する考察

分解対象行列の相関を変化させた時の近似誤差を従来法と提案手法で比較した結果を図 8 に示す．このとき，分解対象の行列は次元数 1158 次元でクラス数は 27 とし，クラス間の相関を 0.1 から 0.7 まで変化させる．提案手法の各段の基底数は 10 とし，クラス当たりの基底数を 3 である時の二乗誤差の平均を図 8 に示す．図 8 より相関が 0.34 以上であれば従来法と同等以上の精度で分解できることがわかる．評価実験において分解対象の重み行列 $\mathbf{W}$ の平均相関値は 0.38 であるため，従来法に比べ近似誤差が減少したと考えられる．

4.4.2　早期棄却に関する考察

早期棄却を導入した提案手法において各段で非対象クラスを棄却できた累積割合を図 9 に示す．図 9 より，27 クラスの標識の識別問題において，6 段目までの識別器により 89.1%の非検出対象クラスを棄却した．また，最後の段数の識別器に辿りついた非検出対象サンプルは 1.2%であった．この結果より，ほとんどの非対象クラスのサンプルを初期の段数の識別器により棄却できるため，早期棄却を導入することで約 7.5 倍高速に識別することが可能になった．

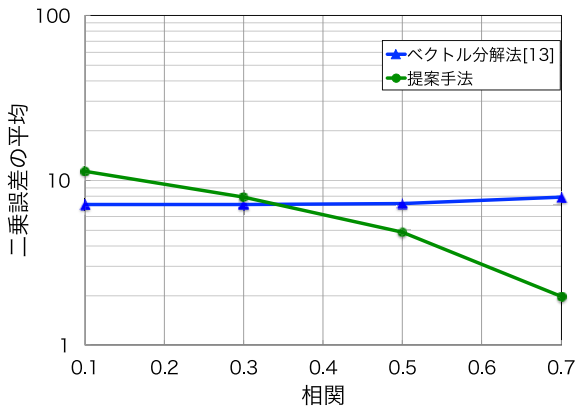


図 8　相関を変化させたときの近似誤差

5.　おわりに

提案手法は，行列分解とカスケード構造による早期棄却を導入することで，高速な物体検出が可能であることを確

表 1 識別時間と識別精度の比較

手法	4 クラスの標識		27 クラスの標識		24 クラスの標識	
	識別時間 [μs]	TPR(FPR=0.01)	識別時間 [μs]	TPR(FPR=0.01)	識別時間 [μs]	TPR(FPR=0.01)
分解前	8.05	0.843	35.41	0.577	31.03	0.886
ベクトル分解法 [8]	1.53	0.802	12.53	0.524	8.67	0.872
ベクトル分解法 [13]	0.75	0.836	8.59	0.561	6.31	0.878
提案手法 (棄却なし)	0.61	0.846	8.45	0.535	6.41	0.882
提案手法 (棄却あり)	0.37	0.840	1.12	0.533	3.56	0.875

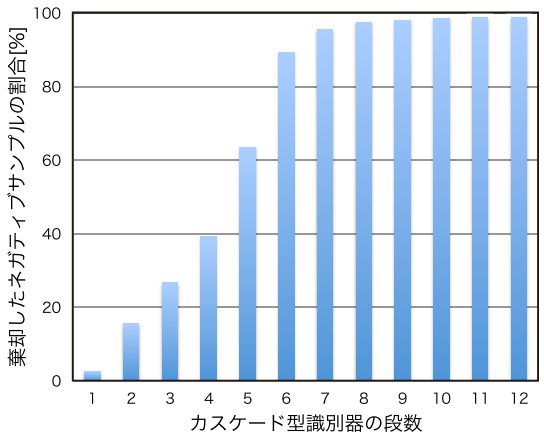


図 9 カスケード型識別器の段数とネガティブサンプルの棄却率の関係

認した．多クラス識別問題において，クラス間の類似性が高いほど提案手法の効果が高いことが判明した．今後は，早期判定の導入により対象物体においても高速に識別することを目的とする．

参考文献

[1] Image Net: `http://www.image-net.org`.
[2] Caltech 256: `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`.
[3] Viola, P. and Jones, M.: Rapid object detection using a boosted cascade of simple features, *Computer Vision and Pattern Recognition*, Vol. 1, pp. 511–518 (2001).
[4] Porikli, F.: Integral histogram: A fast way to extract histograms in cartesian spaces, *Computer Vision and Pattern Recognition*, Vol. 1, pp. 829–836 (2005).
[5] Dollár, P., Tu, Z., Perona, P. and Ramanan, D.: Integral Channel Features, *BMVC*, Vol. 2, No. 3 (2009).
[6] Freund, Y. and Schapire, R. E.: Experiments with a new boosting algorithm, *International Conference on Machine Learning*, Vol. 96, pp. 148–156 (1996).
[7] Dean, T., Ruzon, M. A., Segal, M., Shlens, J., Vijayanarasimhan, S. and Yagnik, J.: Fast, accurate detection of 100,000 object classes on a single machine, *Computer Vision and Pattern Recognition*, pp. 1814–1821 (2013).
[8] Hare, S., Saffari, A. and Torr, P. H.: Efficient online structured output learning for keypoint-based object tracking, *Computer Vision and Pattern Recognition*, pp. 1894–1901 (2012).
[9] Dalal, N. and Triggs, B.: Histograms of oriented gradients for human detection, *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1, pp. 886–893 (2005).
[10] 松島千佳，山内悠嗣，山下隆義，藤吉弘亘：物体検出のための Relational HOG 特徴量とワイルドカードを用いたバイナリーのマスキング，電子情報通信学会論文誌 D, Vol. J94-D, No. 8, pp. 1172–1182 (2011).
[11] 後藤雄飛，土屋成光，山内悠嗣，藤吉弘亘：近似計算を導入した線形識別器の早期判定による高速な識別，電子情報通信学会論文誌 D, Vol. 97, No. 2, pp. 294–302 (2014).
[12] Cheng, M.-M., Zhang, Z., Lin, W.-Y. and Torr, P.: BING: Binarized normed gradients for objectness estimation at 300fps, *IEEE Conference on Computer Vision and Pattern Recognition* (2014).
[13] Yamauchi, Y., Ambai, M., Sato, I., Yoshida, Y. and Fujiyoshi, H.: Distance Computation Between Binary Code and Real Vector for Efficient Keypoint Matching, *IPSJ Transactions on Computer Vision and Applications*, Vol. 5, pp. 124–128 (2013).
[14] Ambai, M. and Sato, I.: SPADE: Scalar Product Accelerator by Integer Decomposition for Object Detection, *Computer Vision–ECCV*, Vol. 8693, Springer, pp. 267–281 (2014).
[15] Cortes, C. and Vladimir, V.: Support-Vector Networks, *Machine Learning*, Vol. 20, No. 3, pp. 273–297 (1995).