社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

カメラ撮影文字の事例に基づく実時間認識 (訂正版)

岩村　雅一†　　辻　　智彦†　　黄瀬　浩一†

† 大阪府立大学大学院工学研究科　〒 599-8531 大阪府堺市中区学園町 1-1
E-mail: {masa,kise}@cs.osakafu-u.ac.jp, tsuji@m.cs.osakafu-u.ac.jp

あらまし　本稿では，カメラで撮像した文字の認識問題において，事例を大量にデータベースに蓄積しておくことで，高速かつ正確な認識を実現する方法を述べる．具体的には，我々が既に提案している実時間カメラベース文字認識手法に，我々が特定物体認識のために提案している高速かつ頑健な近似最近傍探索手法を新たに導入する．これにより，100 種類のフォントを学習しても認識率と処理時間の両方において高い性能を実現することが確認できた．
キーワード　カメラベース文字認識，実時間，事例，近似最近傍探索

Real-Time Case-Based Recognition of Camera-Captured Characters (Revised)

Masakazu IWAMURA†, Tomohiko TSUJI†, and Koichi KISE†

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

Abstract　In this paper, in the recognition problem of camera-captured characters, we present how to realize quick and precise recognition using a lot of case examples stored in a database. To be more precise, we introduce our fast and robust approximate nearest neighbor search method, originally designed for specific object recognition, to our real-time camera-based character recognition method. In the experiments using a database storing 100 fonts, we confirmed that the proposed method achieved high performance in both recognition rate and processing time.
Key words　camera-based character recognition, real time, case example, approximate nearest neighbor search

1. ま え が き

本稿では，クラスラベル付きの文字データ（事例）を大量にデータベースに蓄積しておき，事例を用いて高速かつ正確に文字を認識する方法を述べる．本稿の方法は，字種の識別境界を計算する等，学習データから得られる情報を要約する，いわゆる「学習」を行わない．その代わり，大量に蓄えた事例の中から最も類似するものを発見することで，OCR をあたかも文字の定義を知っているかのように振る舞わせることが本研究の目指すところである．

大量の事例を扱う上でまず問題となるのが，探索に要する時間である．本稿では実時間処理を標榜するため，この問題は不可避である．認識の頑健性も達成すべき重要な目標である．本稿ではカメラで撮像した文字を認識対象とするため，撮像した文字画像に生じる射影歪みや解像度の低下，ぼけにも対処する必要がある．

これらの要求を満足するために，本稿では，我々が既に提案しているカメラベース文字認識手法 [1] を出発点とする．この手法は，(1) 実時間処理が可能，(2) 射影歪みに頑健，(3) 文字の配置に依らない認識が可能，という 3 つの要件を満たす最初の手法である．安価な web カメラと接続したノートパソコン上で実時間で動作し，計算機にサーバーを利用した場合には 1 秒間に 200〜250 文字程度（1 つのフォントを登録した場合）の高速な認識が可能であった．しかし，その一方で，頑健性には課題があった．具体的には，(i) 射影歪みを受けた画像の認識性能が十分ではない，(ii) データベースに多数のフォントを登録すると認識性能が低下する，という問題である．

そこで，この問題を解決し，「事例を用いた高速かつ正確な文字認識」を実現するために，本稿では前述のカメラベース文字認識手法に，文献 [2] で提案されている高速かつ頑健な特定物体認識のための近似最近傍探索手法を新たに導入する．これにより，100 種類のフォントを学習しても認識率と処理時間の両方において高い性能を実現できる．

本稿では，簡単のために，白色の背景に黒色の文字が書かれていることを想定する．文字画像はカメラで撮影するので，射影歪み，ぼけや解像度低下の影響を受けるが，文字の連結成分

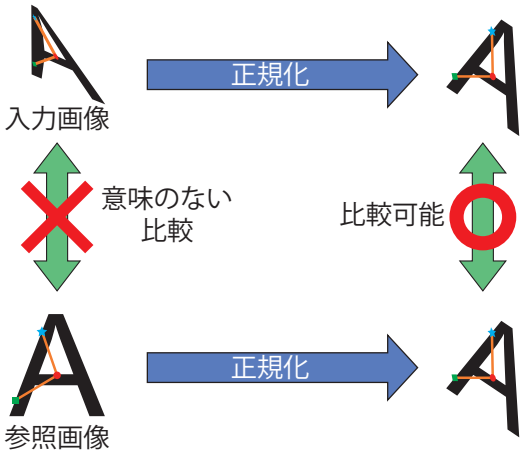


図 1: 従来手法 [1] における画像（連結成分）の照合方法.

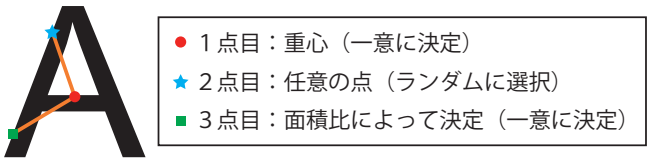


図 2: 従来手法 [1] における対応する 3 点の選択方法.

は簡単な処理で切り出せるとする．また，全ての文字は同一平面上に存在するとする．

2. 文献 [1] の認識システム

本節では文献 [1] の認識手法（以下，従来手法）の概要について述べる．その核となるのは，図 1 に示すアフィン変換を受けた連結成分の照合部分である．画像に歪みがある場合，2 画像を重ね合わせて単純に比較することはできない．しかし，もし画像中の対応する 3 点が既知であれば，点を結ぶ線分が直交し，長さが同じになるように画像を正規化することで比較可能となる．このような対応する特徴点に基づく画像照合手法としては，Geometric Hashing (GH) [3], [4] がよく知られている．GH では多数の特徴点から 3 点の組み合わせをランダムに選択し，図形の照合を試みる．この処理に要する計算量は，特徴点数を P とすると $O(P^4)$ である．従来手法では照合する対象が連結成分であることを利用し，特徴点を図 2 のように定めることで $O(P^2)$ に削減した．

従来手法は図 3 のように分離文字取扱部，登録部，認識部で構成されており，以下では各部の処理について概説する．なお，パラメータに関する記述は本稿の実験で使用した値であり，文献 [1] と同一とは限らない．

2.1 分離文字取扱部

本節では「i」や「j」等，複数の連結成分から成る文字の取り扱いについて述べる．分離文字を扱うために，登録時に文字を構成する連結成分の数を調べる．2 以上あれば，図 4 に示す分離文字テーブルに 2 つの連結成分の相対位置や大きさの関係を記載する．認識時に分離文字テーブルを参照し，この条件を満たす連結成分があれば連結成分を統合し，一文字と認識する．

Arial フォントの場合，「i」の下部，「I（アイ）」，「l（エル）」

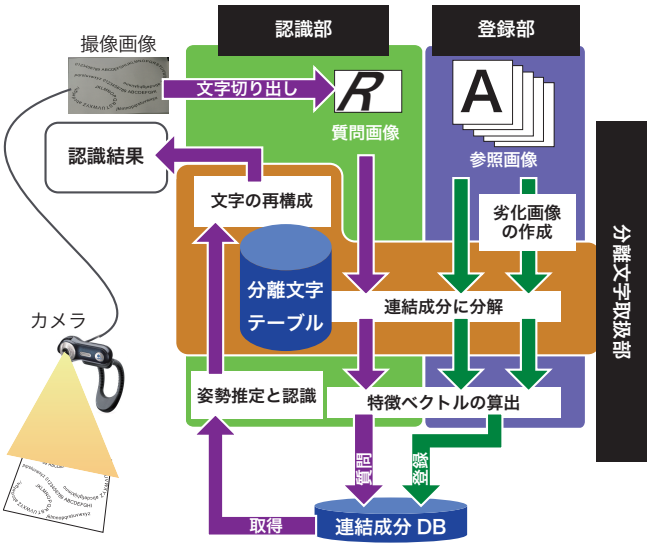
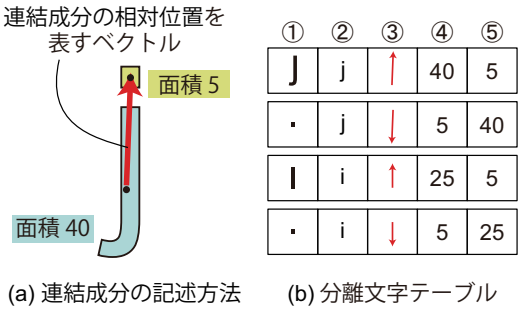


図 3: 文献 [1] の認識システムの概要.



(a) 連結成分の記述方法　(b) 分離文字テーブル

図 4: 分離文字の記述方法．分離文字テーブルの要素は左から，①連結成分の形状 (連結成分番号)，②元の文字，③連結成分の相対位置，④自連結成分の面積，⑤組になるべき連結成分の面積である．

はアフィン歪みを受けると外見が同一になり，識別不能である．「i」を正しく認識するためには，「I」や「l」のように外見が同一である連結成分も「i」の一部であるか調べる必要がある．そのため，アフィン変換によって同一形状になる連結成分を同一クラスとみなすクラス分け処理を行う．これは，参照画像を登録する際に，作成中のデータベースを用いて 1 枚ずつ認識し，既に類似の連結成分が登録されている場合は同じクラスとみなす処理である．クラス分けによって異なる字種の連結成分が同一クラスになってしまう場合があることに注意が必要である．図 5 に示す例では，劣化した「c」が「0」と同じクラス 1 に属しているため，認識時にクラス 1 であると認識された連結成分が「0」か「c」かを区別できない．このような同一クラス内の字種の判別は文献 [5] で述べる後段の単語認識処理に委ねる．

2.2 連結成分登録部

連結成分登録部では，二値化した連結成分の画像をデータベースに登録する．その際，アフィン変換を受けた連結成分の画像をアフィン不変な特徴ベクトルで記述する．

2.2.1 劣化画像の生成

撮影時のピンぼけや解像度の低下に対処するため，生成型学習法 [6] を用いて劣化した参照画像を生成する．本論文では，ガウスぼかしの重畳を 3 段階 (適用なしを含む) と解像度の低下

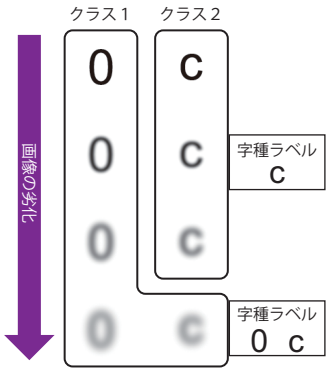


図 5: 連結成分のクラス分け処理の例.

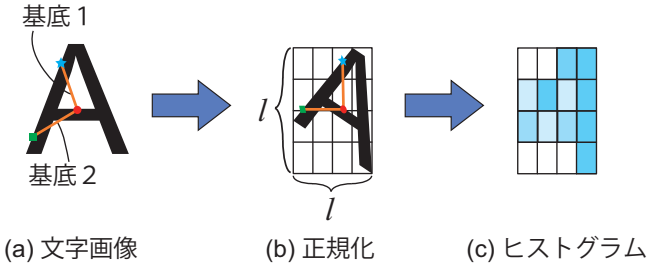


図 6: 特徴ベクトルの計算方法.

を 3 段階 (適用なしを含む) の合計 9 段階で行い，元の参照画像を含めて 9 枚の参照画像を得た．生成された劣化画像は再び二値化し，以後は元の参照画像と同様に扱う．

2.2.2 特徴ベクトルの生成

特徴ベクトルを以下の二段階の処理によって計算する．

一段目では，図 6 に示すように，2 本の基底によって張られる不変座標系を用いて k 次元の特徴ベクトルを作成する．図 6(a) を 2 本の基底を用いて正規化したものが図 6(b) である．そして，$k(=l\times l)$ 個の均一な部分領域を設定し，図 6(c) のような黒画素数のヒストグラムを計算する．最後にヒストグラムの合計が 1 になるように正規化し，k 次元の特徴ベクトルを得る．予備実験の結果から，本稿では $l=4$ とした．

二段目では，$(3k)$ 次元の特徴ベクトルを得る．図 6(a) のように，特徴点が 3 点あれば，そのうち 1 点を原点とすることで 2 本の基底ベクトルが計算できる，原点の取り方は 3 通りあり，基底ベクトルの取り方も 3 通りあるため，3 本の k 次元特徴ベクトルが計算できる．これを単純に結合することで得られる $3k$ 次元の特徴ベクトルを提案手法で使用する．結合の順序は特徴点が選択された順序に基づいて一意に決定する．理論上，同一の 3 点が得られる限り，アフィン歪みの影響を受けずに常に同じベクトルが計算できる．

2.2.3 連結成分データベースへの登録

図 7 に示すハッシュ表に連結成分の情報を登録する．連結成分毎に保持する情報は，連結成分番号，特徴ベクトル，基底の作成に用いた特徴点 3 点の座標である．衝突が起こる場合はリストで連結する．特徴点の座標を登録しておくのは，認識の際の姿勢推定で利用するためである．

ハッシュ値 H_{index} は次式で計算する．

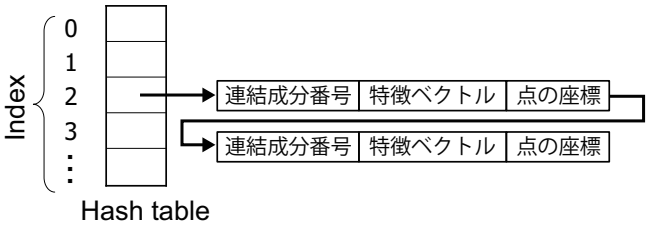


図 7: 連結成分番号を保持するハッシュ表.

$$H_{\mathrm{index}}=\left(\sum_{i=1}^{3k}D^{i-1}r_i\right)\bmod H_{\mathrm{size}}, \tag{1}$$

ここで H_{size} はハッシュ表の大きさであり，$2^{19}-1$ とした．r_i は特徴ベクトルの i 番目の要素を D 段階に量子化した値である．文献 [1] では $D=3$ としたが，本稿では $D=2$ とした．衝突が起こる場合はリストで連結する．

2.3 連結成分認識部

2.3.1 画像の取得と連結成分の切り出し

画像はデジタルカメラや web カメラで静止画ないし動画として取得する．動画として撮像した場合はフレーム毎に分解して，複数の静止画として扱う．得られた各画像を質問画像と呼ぶ．質問画像に適応二値化と輪郭抽出手法を適用し，連結成分を抽出する．

2.3.2 特徴ベクトルの生成

連結成分から特徴ベクトル（以後，クエリ特徴ベクトルと呼ぶ）を生成する．この処理は 2.2.2 の処理と基本的に同じである．唯一異なるのは，速度向上のために，作成可能な全ての特徴ベクトルを計算するのではなく，あらかじめ定める S 個に限定することである．S が小さいと性能は低下するものの処理時間は小さくなる．本稿では $S=10$ とした．

2.3.3 認識と姿勢推定

まず，紙面の姿勢を推定する．本稿では全ての文字は同一平面 (紙面) 上に存在すると仮定している．この場合，アフィン変換行列から計算される 4 つのアフィン変換パラメータのうち，せん断変形と独立変倍のパラメータは全ての連結成分で共通になるはずである．そこで従来手法では図 8(b) のような 2 次元空間での密度推定を利用して，尤もらしいパラメータの組を推定する．推定には，質問画像の各連結成分の特徴ベクトルからハッシュ値を計算して，ハッシュ表を参照して得られる連結成分の情報を使用する．ただし，これらの情報には誤ったものが存在するため，まず質問画像の連結成分と参照画像の連結成分の面積比が 1 から大きく外れる連結成分は信頼できないとして除外する．さらに，図 8(a) に示す字種に対する重み付き投票によって，信頼できる連結成分に絞り込む．ここでは最大得票数の 9 割の得票を得た「推定グループ」に属する連結成分のアフィン変換行列のみを前述の 2 次元空間にプロットする．プロットされた点の中から近傍の密度が最も高い点 (図 8(b) の星マーク) を選択する．

次に，連結成分毎に認識結果を定める．図 8(c) のような連結成分の回転角と連結成分番号が作る 2 次元空間での密度推定

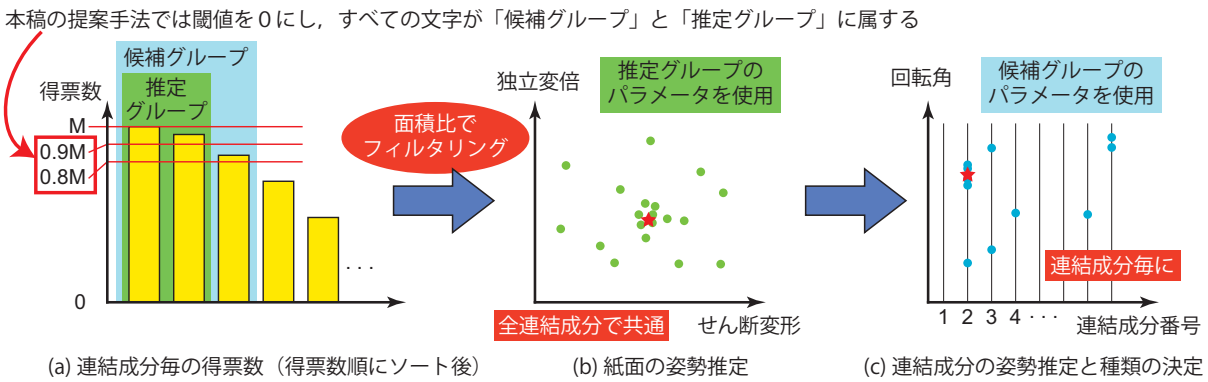


図 8: 連結成分の認識と姿勢推定.

を利用して，尤もらしい回転角のパラメータと連結成分番号の組を推定する．推定には，前述の投票で最大得票数の 8 割の得票を得た「候補グループ」に属する連結成分のアフィン変換行列のみを用いる．図 8(b) の場合と異なるのは，回転角は連続量であるが，連結成分番号は離散であるため，各連結成分番号について 1 次元の密度推定を行うことである．以上の処理により，各連結成分の種類 (連結成分番号) と姿勢 (せん断変形，独立変倍，回転角) を求めることができる．

3. 提 案 手 法

本節では，前節で述べた従来手法に文献 [2] の方策を 3 つ導入することにより，改良手法を提案する．

1 番目の方策は，距離計算の導入である．2.3.3 節で述べたように，ハッシュ表から得られる情報には誤りが含まれているため，その中から正しいものを選択する必要がある．従来手法では図 8(a) に示す字種に対する投票で信頼性の高い情報を絞り込んでいたが，本稿ではその代わりに，クエリ特徴ベクトルと，ハッシュ表から得られた特徴ベクトルのユークリッド距離を計算し，距離が閾値以下のものを選択する (注1)．本稿では距離の閾値を t/r と定めた．ここで r は画像の大きさの正規化（次節の実験で述べる）後の連結成分の黒画素数である．t は定数で，登録部のクラス分類時には $t = 200$，認識部には $t = 80$ を用いた (注2)．この方策は非常に強力で，データベースに 100 フォントを登録した場合，撮像角度に依らず，認識率が約 8%向上し，処理時間は約 10ms 減少することを確認している．

2 番目の方策では，図 9 に示すように，クエリ特徴ベクトルのビット反転によって新たなクエリ特徴ベクトルを作成する．1 番目と 2 番目の方策を同時に用いれば，認識率が撮像角度に依らず約 8%，処理時間が撮像角度に依って約 0.3ms〜4.0ms 減少することを予備実験により確認した．本稿では，48 次元の特徴ベクトルに対して $e = 0.002$ と $b = 8$ を用いた．

3 番目の方策はハッシュ値の衝突に関する工夫である．従来手法ではハッシュ表の一部のビンで大量の衝突が起こることがあった．ハッシュの処理時間は衝突の数に比例するため，衝突が大量に発生すると処理時間が非常に遅くなるという問題があった．そこで本稿では，衝突の数が c より大きくなったビンは c 個に間引きくこととした．すなわち，衝突が大量に発生したビンの要素を c 個を除いて削除する (注3)．この操作により，メモリ使用量と処理時間を大幅に削減できる．予備実験により，この方策を単独で用いた場合は認識率が撮像角度に依らず約 5%，処理時間が撮像角度に依らず 1/3 程度にまで減少することを確認した．本稿では $c = 20$ とした．

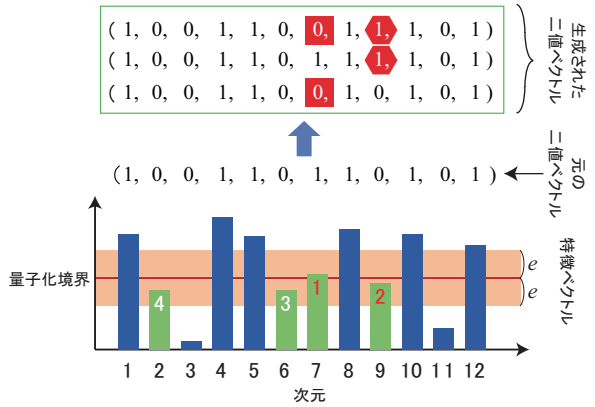
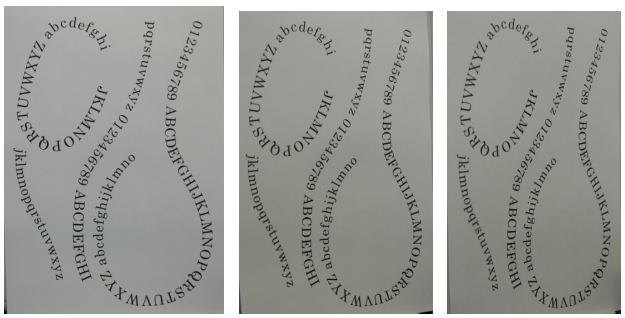


図 9: 文献 [2] のクエリ特徴ベクトルの二値化方法．この例では，特徴ベクトルは 12 次元，$b = 2$ である，実数値ベクトルの二値化の際に，閾値とベクトルの各要素の差を昇順に並べ，差が e より小さい要素を最大 b 個選択し，ビットを反転させる．選択の際には差が小さい要素を優先する．図の例では，7 と 9 の次元が選ばれ，ビット反転され，3 本の新しいベクトルが生成された．

4. 実　　験

提案手法の有効性を確認するために，最大 100 フォントを登録したデータベースを用意し，カメラで撮像した様々なフォントの文字画像を認識した．

(注1)：文献 [2] の手法は距離が最小のもののみを選択し，本稿の手法と若干異なる．予備実験のおいて両者を比べたところ，閾値を用いる方が性能が良かった．
(注2)：登録時と認識時の閾値の違いは次節で述べる連結成分の正規化の大きさの違いが一因である．
(注3)：文献 [2] の手法は閾値以上の衝突が起こると，そのビンに登録されている情報を全て削除するため，本稿の手法と若干異なる．文献 [2] の方法をそのまま試したところ，「O」のような円形の字種だけが選択的に認識できなくなった．この原因として，円形の文字ではほぼ全ての特徴ベクトルが同じビンに登録されるため，大量の衝突のためにそのビンの情報が全て削除されると全く認識できなくなることが考えられる．



(a) 0 度. (b) 30 度. (c) 45 度.

図 10: デジタルカメラで様々な角度から撮像し，手動で切り出した認識対象画像（Century フォントの場合）.

4.1 準　　備

実験に使用した字種は，大文字と小文字のアルファベットと数字の合計 62 字種である．参照画像 1 枚につき 8 種類の劣化画像を生成するため，100 フォントでは合計 55800 枚の参照画像をデータベースに登録した．認識対象として，図 10 に示すような，各文字を 2 回ずつ含み（1 枚当たり 124 文字），文字が曲線上に並ぶようにレイアウトされた画像を用意した．そして，A4 用紙に印刷し，デジタルカメラで 0 度，30 度，45 度から撮像したものを手動で切り出して認識対象とした．図 10 に Century の認識対象画像を示す．画像の大きさはそれぞれ 1577×2209，1397×2185，1265×2201 である．

使用したフォントは，Microsoft Windows 7 にインストールされているフォントから選んだ 100 フォントである．選択の際，ストロークの細いフォントは解像度低下等の影響によって連結成分が 2 つ以上の成分に分解され易いため，除外した．選択したフォントのうち 10 フォントを図 11 に示す．

実験では，登録フォント数を 1〜100 まで徐々に増加させ，認識率と処理時間の変化を見た．登録フォント数は，1〜10 フォントまでは 1 フォントずつ，それ以降は 5 フォントずつ増加させた．認識対象を 75 フォント分しか用意できなかったため，登録フォント数が 1〜75 フォントまでと 80 フォント以降では実験方法が若干異なる．75 フォントまでは，登録フォントと同じフォントの認識対象を認識した．すなわち，1 フォント目は Arial をデータベースに登録し，認識対象として Arial の文字画像を認識対象として用いた．2 フォント目は Arial と Century を登録し，同じく Arial と Century の文字画像を認識した．80 フォント以降は登録フォント数に依らず，75 フォント全ての認識対象を認識した．

2.1 節で述べたように，連結成分は登録処理の際に自動的にクラス分けした．その際，3. 節の提案手法で述べた 2 番目の方策（クエリ特徴ベクトルを生成する方策）は使用しなかった．クラス分けの様子を示す例として，Arial の 62 字種をクラス分けした結果を表 1 に示す．表には生成された 55 クラスのうち，2 字種以上が属するクラスのみを列挙した．

実験では CPU が Opteron 2.8GHz で，メモリが 32GB である計算機を用いた．画像の登録と認識に要する計算量を削減するため，連結成分の幅と高さの大きい方が，参照画像では 100 ピクセル，質問画像では 50 ピクセルになるように正規化した．

Arial 0123ABCDabcd
Century 0123ABCDabcd
Times New Roman 0123ABCDabcd
Verdana 0123ABCDabcd
Meiryo bold 0123ABCDabcd
OCRB 0123ABCDabcd
Book Antiqua Bold 0123ABCDabcd
BellGothicStd-Black 0123ABCDabcd
Franklin Gothic Medium 0123ABCDabcd
TektonPro-Bold 0123ABCDabcd

図 11: 認識に用いたフォントの一部（1〜10 番目のフォント）.

表 1: Arial の 62 字種から自動的に生成された 55 クラスのうち，2 字種以上が属するクラス.

0 O o	6 9	7 L	C c	E m	I l	N Z z
S s	V v	W w	b q	d p	n u	

4.2 実 験 結 果

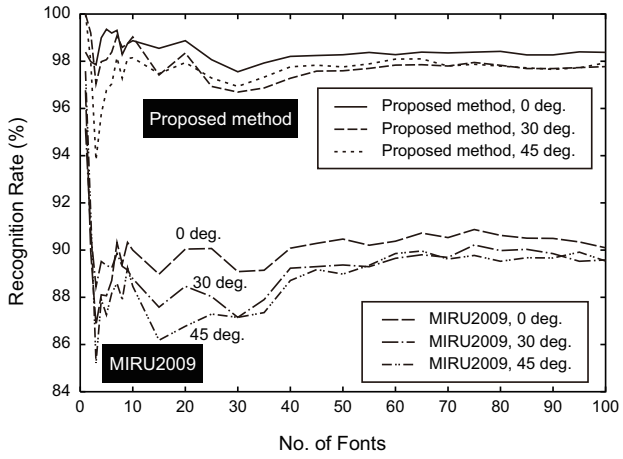
まず，認識率と 1 文字当たりの平均処理時間を図 12(a)，(b) に示す．図中の “MIRU2009” は従来手法を表す．認識率については，従来手法は複数のフォントを登録すると認識率が減少したのに対して，提案手法の登録フォント数に依らず，高い水準でほぼ一定であった．処理時間については，提案手法と従来手法の両方が登録フォント数の増加に伴って処理時間が増加しているが，提案手法の方が傾きが緩やかであった．

提案手法の認識結果を具体的な数字で見てみると，正面から撮像した文字の認識率は 98.4%（従来手法に比べて 7.94%の増加），45 度から撮像した場合でも 97.9%の認識率（同 8.4%の増加）を達成した．処理時間は 7.2ms（従来手法の約 1/3）で，1 秒間に約 140 文字を認識可能である．したがって，本稿で導入した 3 つの方策は非常に有効であることが確認できた．

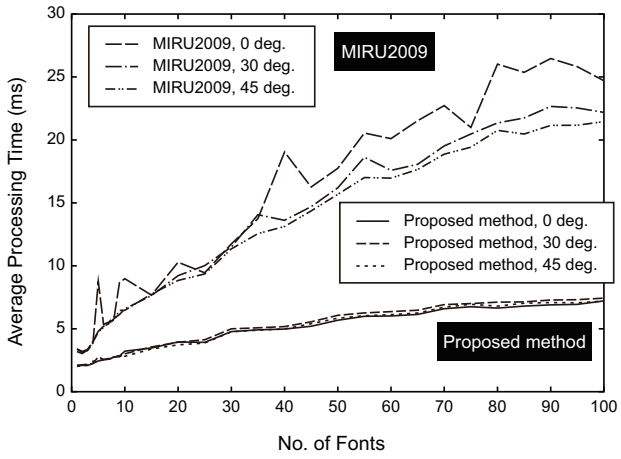
次に提案手法のクラス数とメモリ使用量を図 12(c)，(d) に示す．クラス数は，登録フォント数が 1 で 55 クラス，10 で 397 クラス，100 で 1672 クラスであった．クラス数は登録フォント数の増加に伴って単調に増加したものの，増加率は徐々に減少した．これは新規に登録したフォントの一部は，既に登録された参照画像と同じクラスに属したためと考えられる．一方，メモリ使用量は登録フォント数にほぼ比例して増加した．この原因として，クラス数の増加に拘らず，ハッシュ表に登録される情報はほとんど変わらない事が考えられる．100 フォントの場合のメモリ使用量は約 4GB であったが，実装の工夫によってメモリ使用量は大きく減少可能と考えている．

5. む　す　び

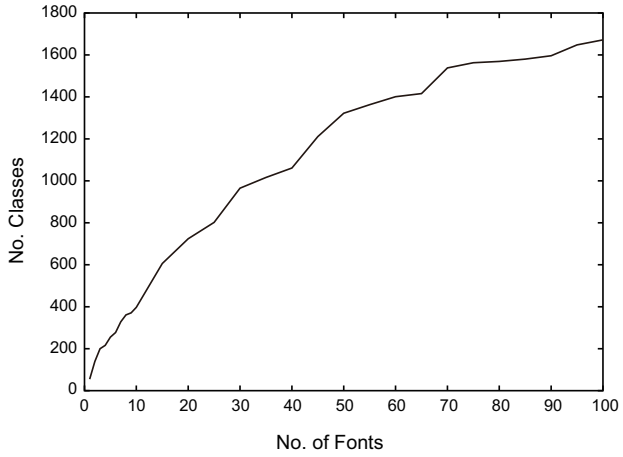
本稿では，カメラで撮像した文字の認識問題において，事例を大量にデータベースに蓄積しておくことで，高速かつ正確な認識を実現する方法を述べた．具体的には，我々が既に提案している実時間カメラベース文字認識手法に，我々が特定物体認識のために提案している高速かつ頑健な近似最近傍探索手法



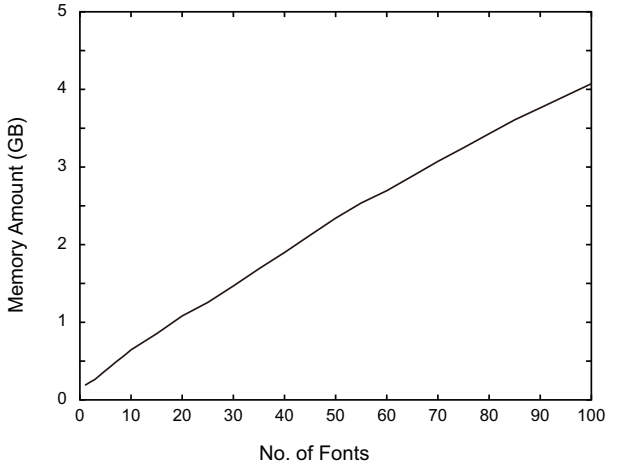
(a) 認識率.



(b) 1 文字当たりの平均処理時間.



(c) クラス数.



(d) メモリ使用量.

図 12: 認識結果.

を新たに導入した．これにより，データベースに 100 フォント（登録画像総数は 55800 枚）を登録し，認識対象の文字画像に劣化（射影歪みや解像度の低下，ぼけ）が生じるという条件の下で 1 秒間に 140 文字程度を認識することが可能になった．したがって,「事例を用いた高速かつ正確な文字認識」を認識率と処理時間の両方において高いレベルで実現できたといえる．

本稿で使用したプログラムは実装時にメモリ使用量を削減する工夫をそれ程行わなかったため，メモリ使用量の削減は今後の課題である．

謝辞 本研究の一部は，科研費補助金若手研究 (B) 21700202 ならびに平成 21 年度シーズ発掘試験（発掘型）（課題番号 11-084）の補助による．

文　献

[1] 岩村雅一，辻　智彦，堀松　晃，黄瀬浩一，“レイアウト非依存な実時間カメラベース文字認識,” 画像の認識・理解シンポジウム (MIRU2009) 論文集，pp.174–181，July 2009.

[2] 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，vol.J92-D，pp.1135–1143，Aug. 2009.

[3] Y. Lamdan and H.J. Wolfson, “Geometric hashing: a general and efficient model-based recognition scheme,” Proc. ICCV1988, pp.238–249, 1988.

[4] H.J. Wolfson and I. Rigoutsos, “Geometric hashing: an overview,” IEEE Computational Science and Engineering, vol.4, no.4, pp.10–21, 1997.

[5] 辻　智彦，岩村雅一，黄瀬浩一，“リアルタイム単語認識技術を利用したカメラベース情報取得システム,” 信学技報，Feb. 2010.

[6] H. Ishida, S. Yanadume, T. Takahashi, I. Ide, Y. Mekada, and H. Murase, “Recognition of low-resolution characters by a generative learning method,” Proc. CBDAR2005, pp.45–51, 2005.