



大阪府立大学
OSAKA PREFECTURE UNIVERSITY



カメラ撮影文字の 事例に基づく実時間認識

岩村雅一 辻智彦 黄瀬浩一

カメラベース文字認識システム

リアルタイムで動作

認識結果

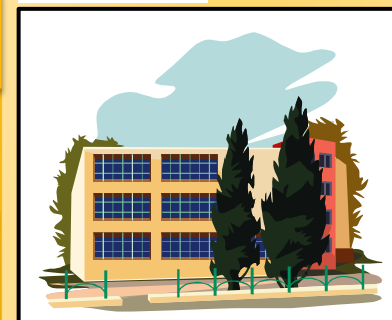
"University"

関連情報

翻訳

・大学

画像



音声



カメラ



ノートPC



文書

応用例

環境中の全ての文字を認識して、
必要な情報のみを提供することができる

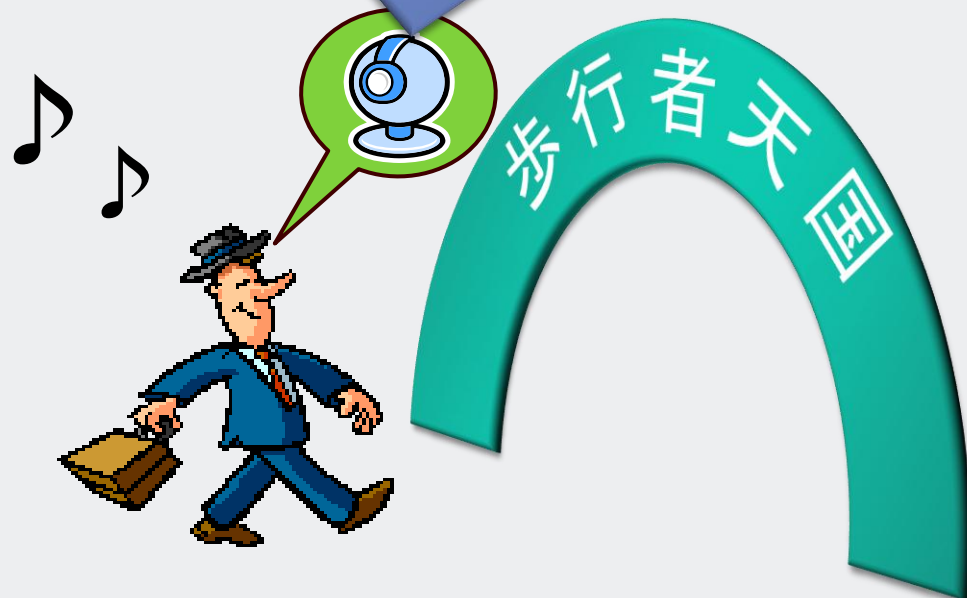
視覚障害者への音声案内

『押ボタン信号
があります』



翻訳システム

Car-free mall



認識の流れ

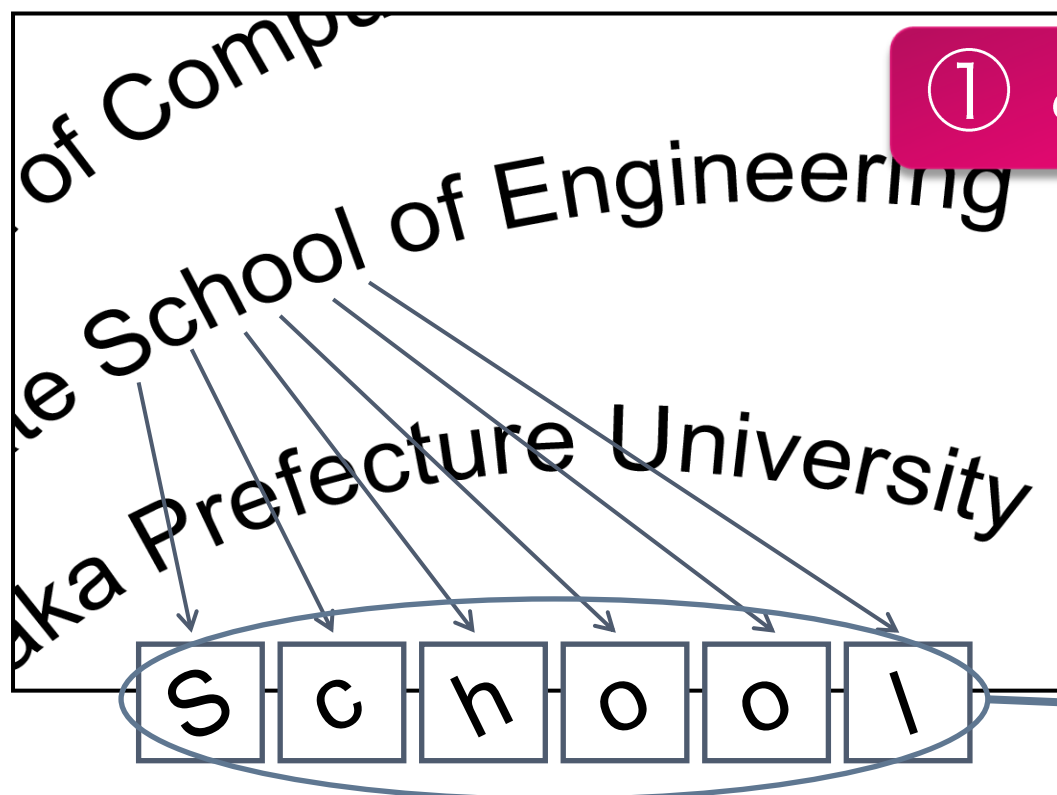
辻 智彦, 岩村 雅一, 黄瀬 浩一:
“リアルタイム単語認識技術を利用した
カメラベース情報取得システム”
(PRMU2009-216)

① 1文字ごとに文字認識

本発表

② 文字を連結して単語を推定

昨日発表



① どこにどんな文字があるか

② どんな単語があるか

"School"

従来手法の長所

(MIRU2009 / CBDAR2009にて発表)

実時間処理

ノートPCで動作可能

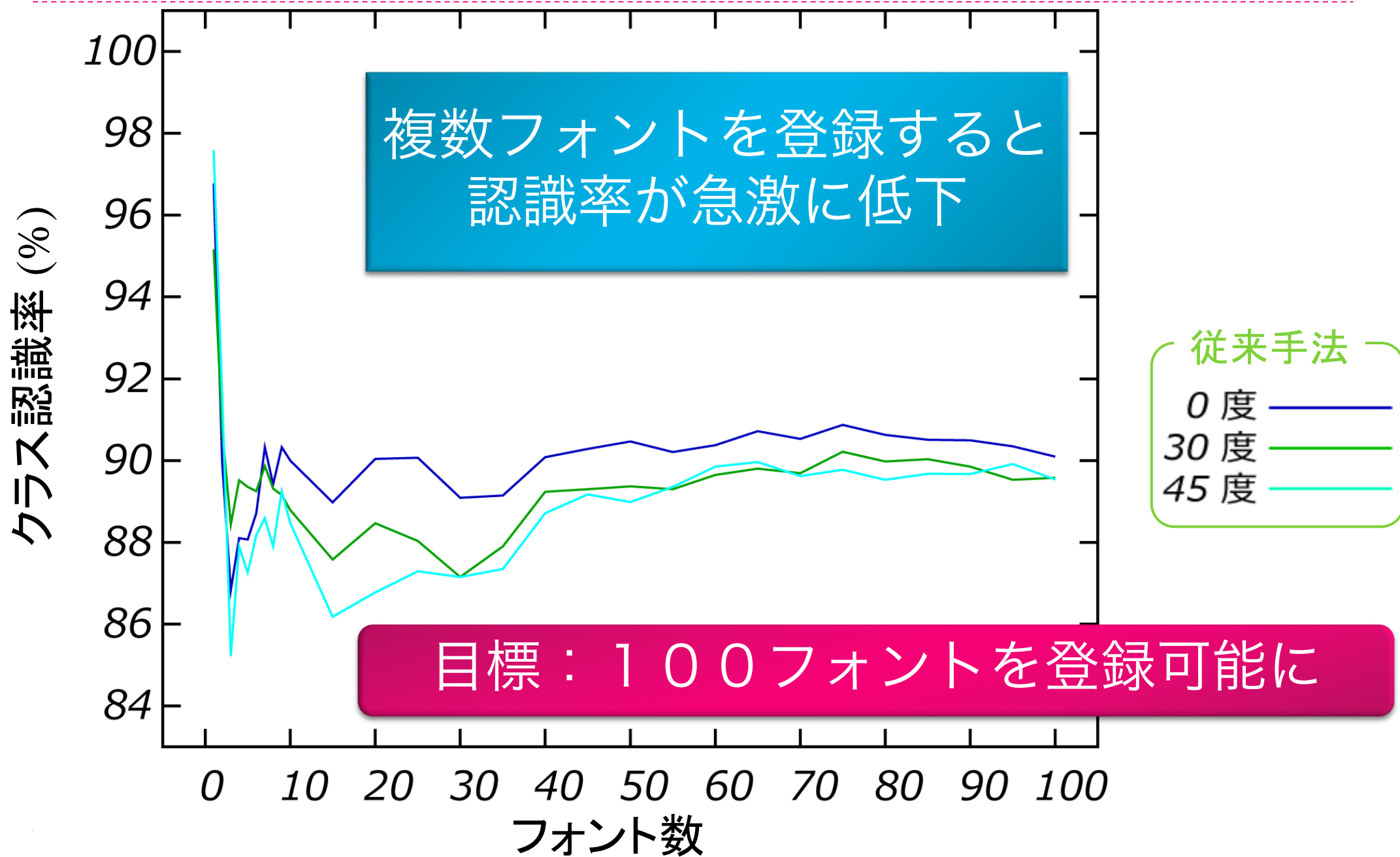
射影歪みに頑健

斜め45度から認識可能

レイアウトフリー

方針：テンプレートマッチングによる
カメラ撮影文字の認識

従来手法の短所： 多種のフォント登録による認識性能の低下



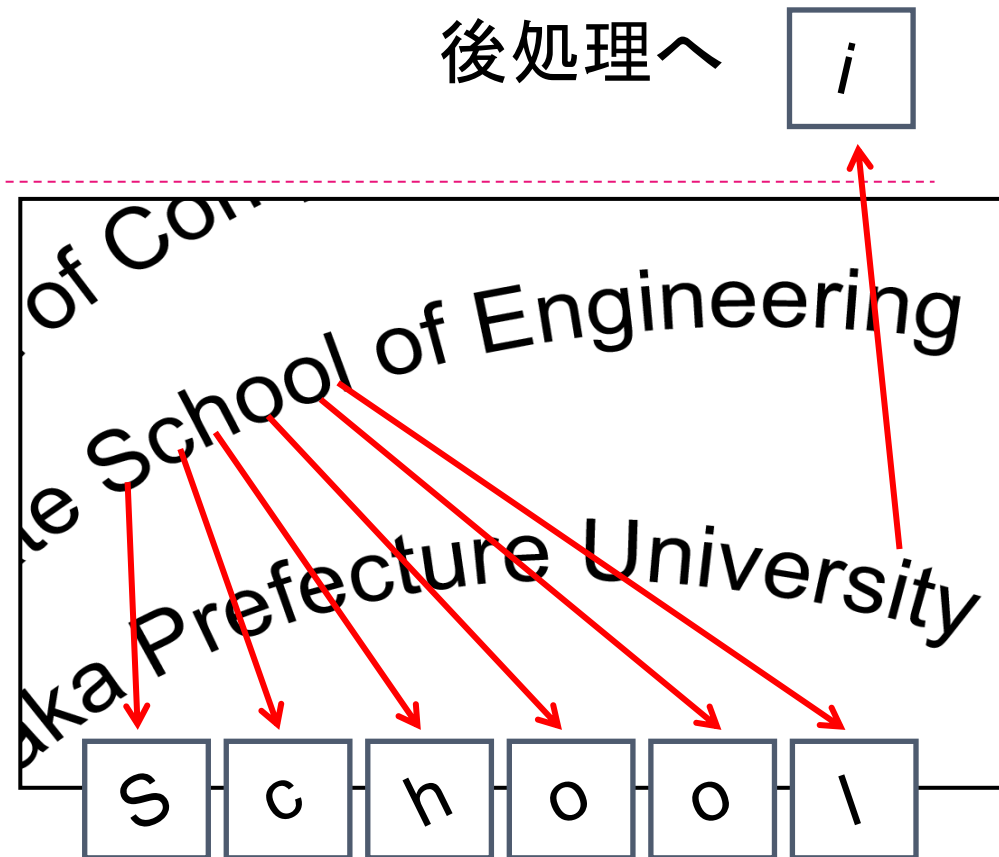
目次

1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ



従来手法 1 : 前提条件 (1)

- ▶ 連結成分単位の認識
- ▶ 問題設定
 - ▶ 文字は同一平面上に存在
 - ▶ 文字は二値化で簡単に抽出可能

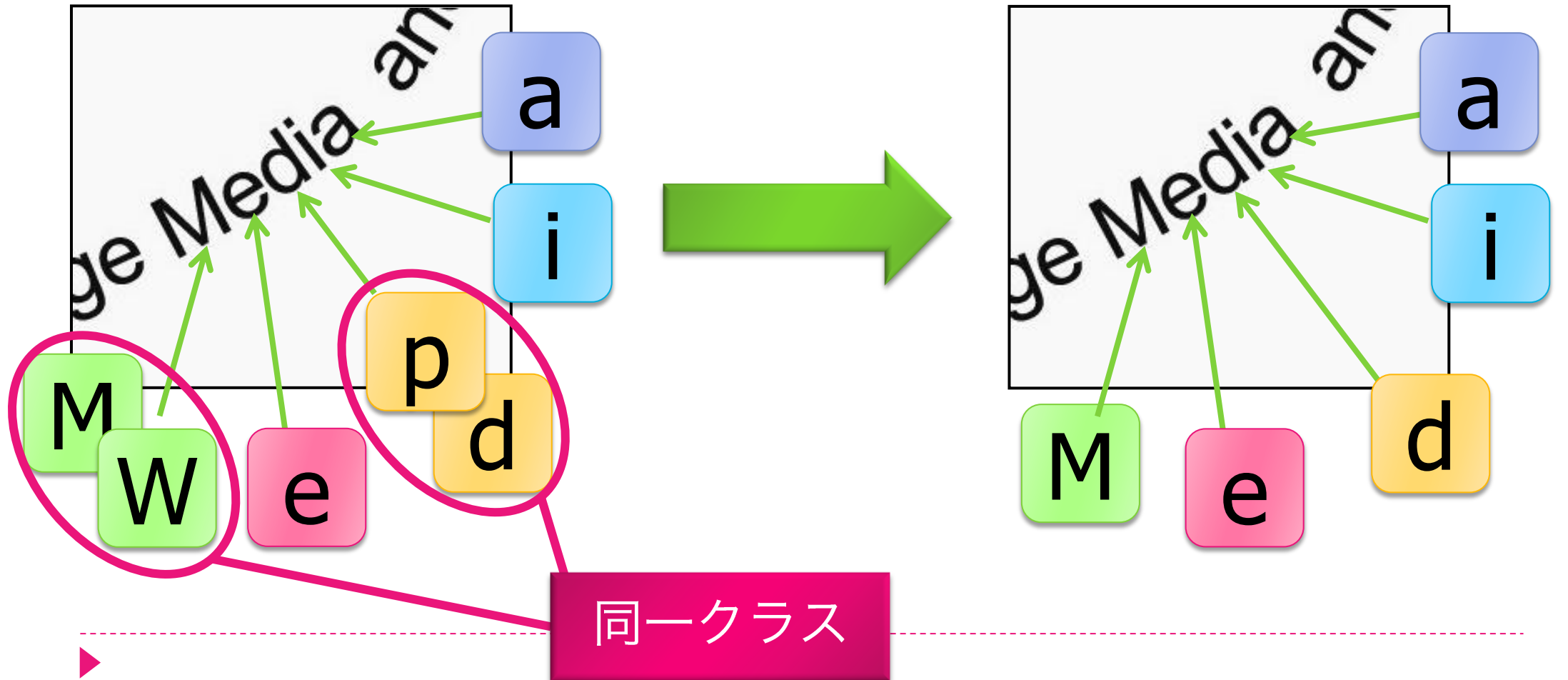


切り出した後の文字の高速処理に特化

従来手法 1 : 前提条件 (2) — クラス単位の認識

文字認識部分

単語認識部分



従来手法 1 :

前提条件 (2) — クラス単位の認識

- ▶ 同一クラスに統合された字種 (自動的に生成)

- ▶ Arialの場合

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



目次

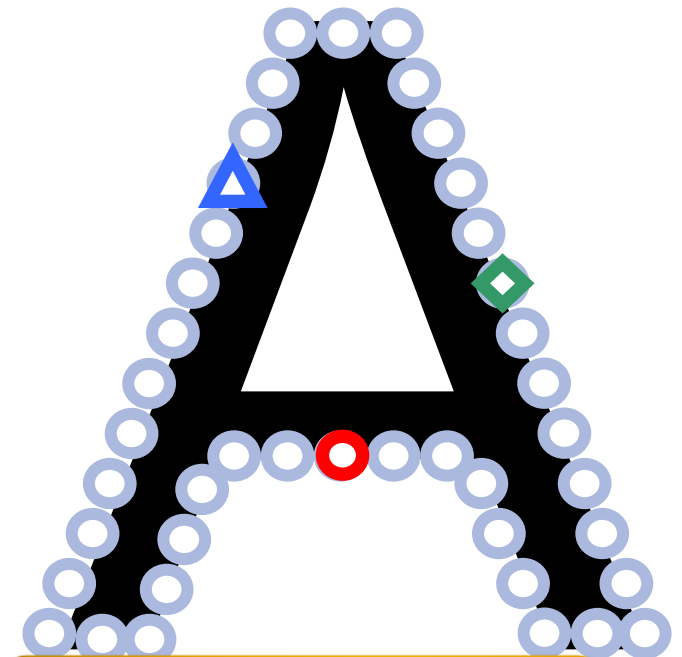
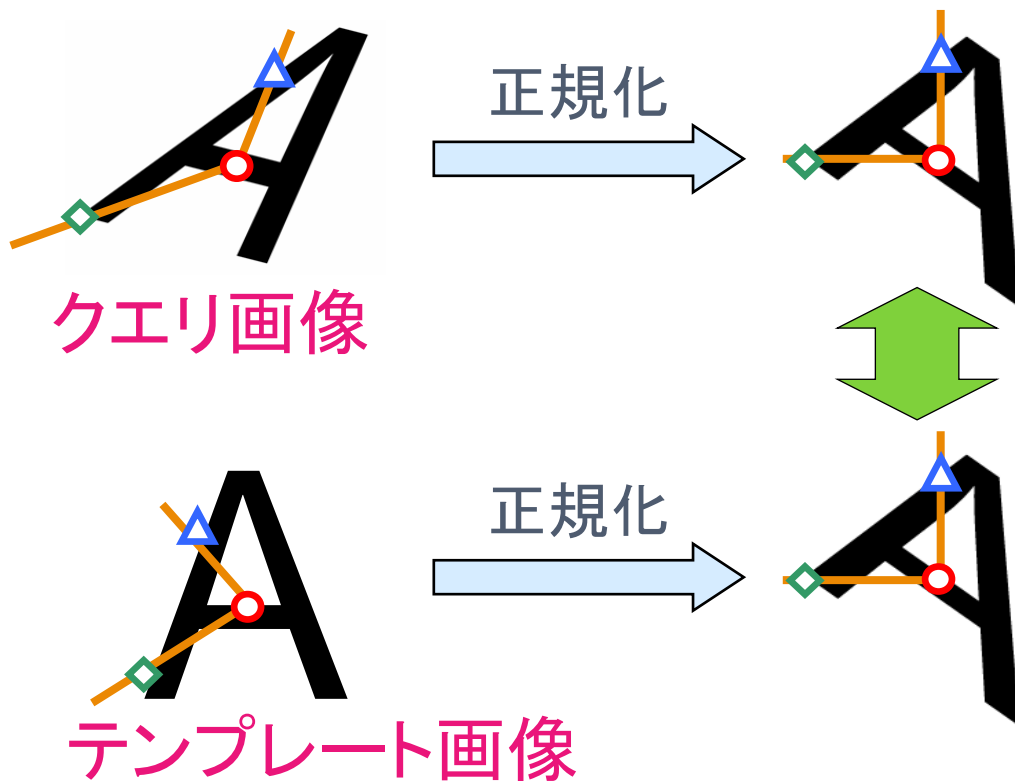
1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ



従来手法 1 - 2 : アフィン不変な認識

- ▶ アフィン不変な認識
 - ▶ 同一の 3 点が選択できれば、照合可能

射影歪みに頑健な
認識の実現

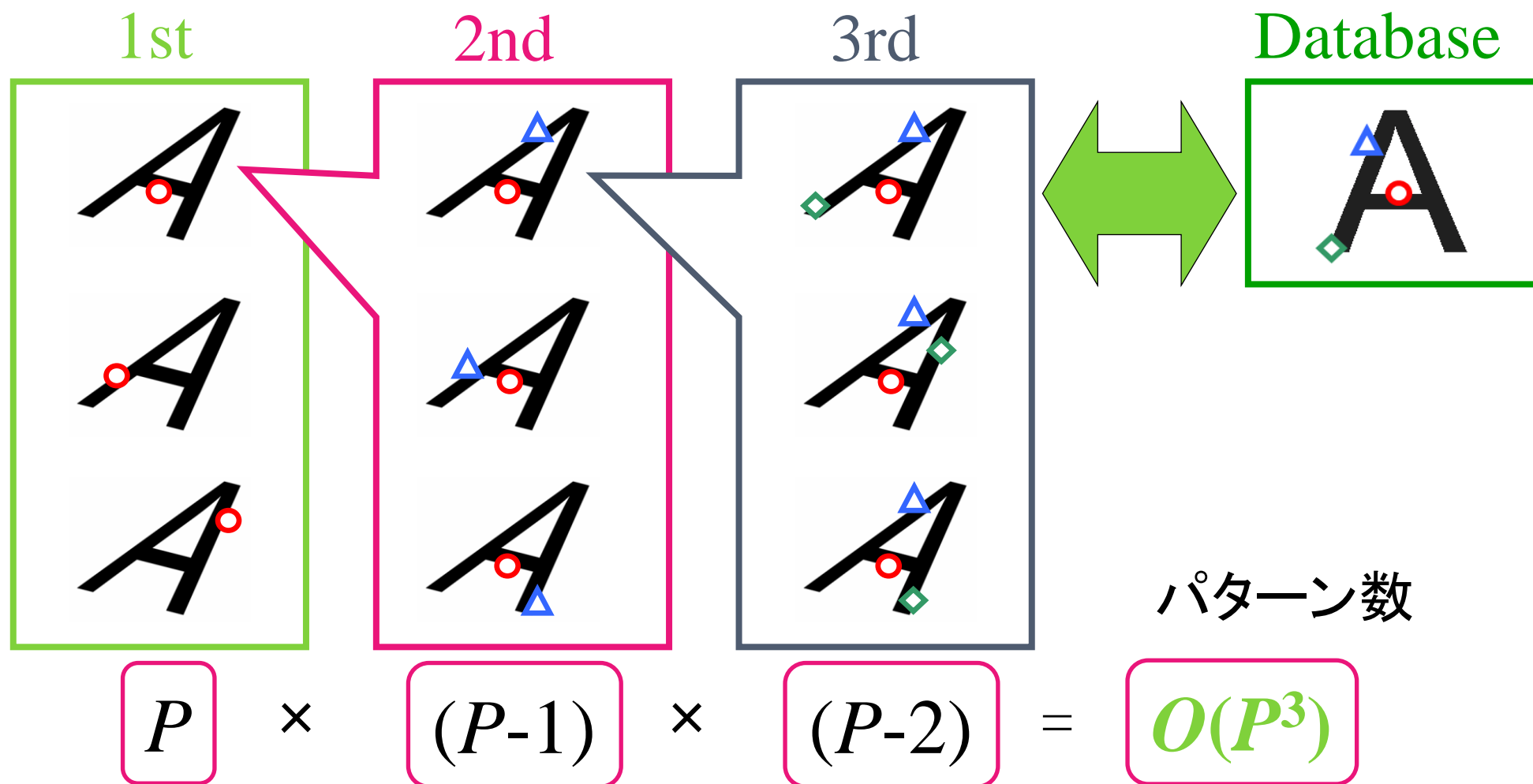


特徴点数 : P

従来手法 1 - 2 :

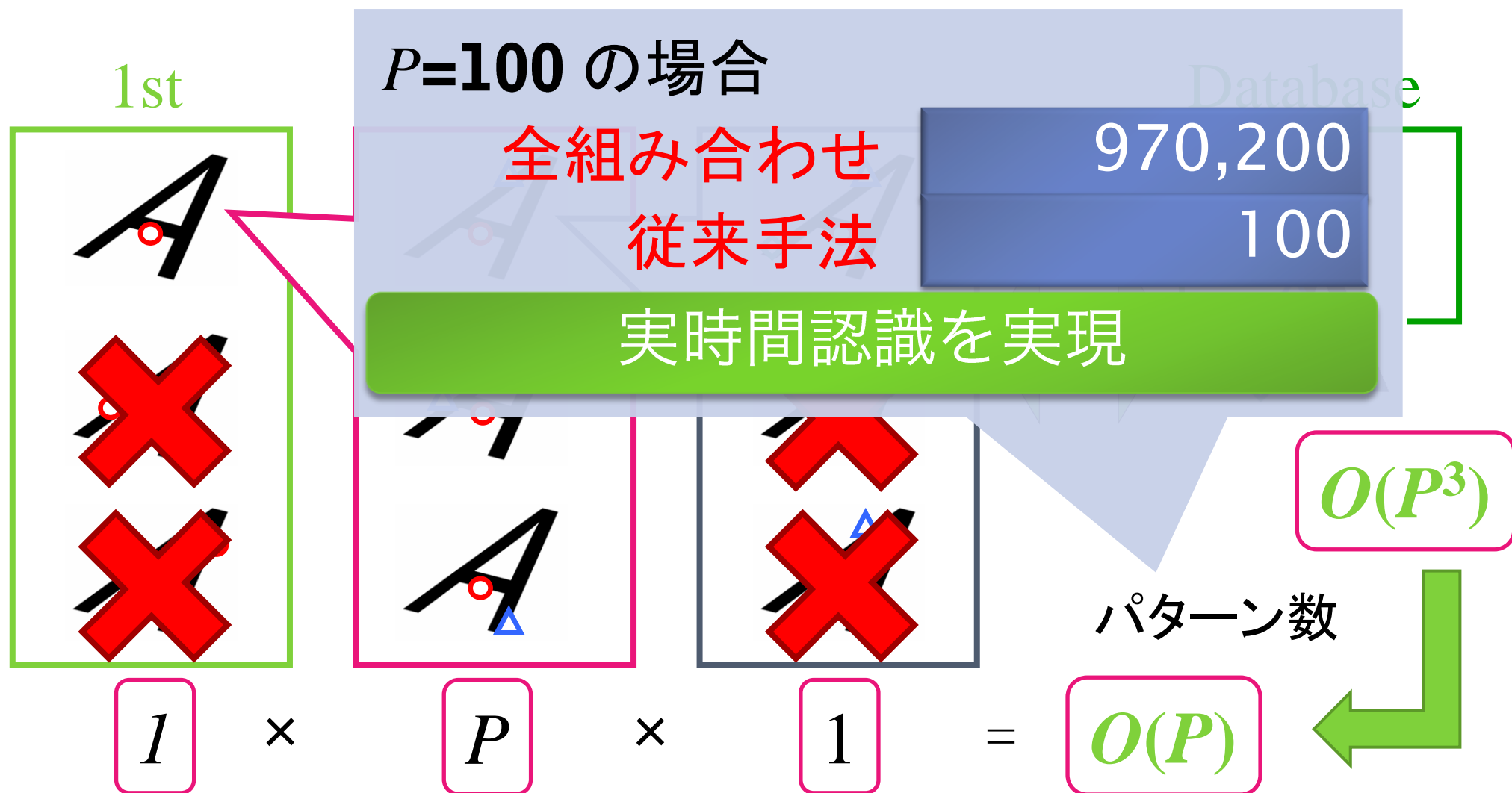
同一の 3 点を選択する方法 (単純な場合)

- ▶ P 点から 3 点を選択する全ての組み合わせを試す



従来手法 1 - 2 : 従来手法が作る 3 点の配置

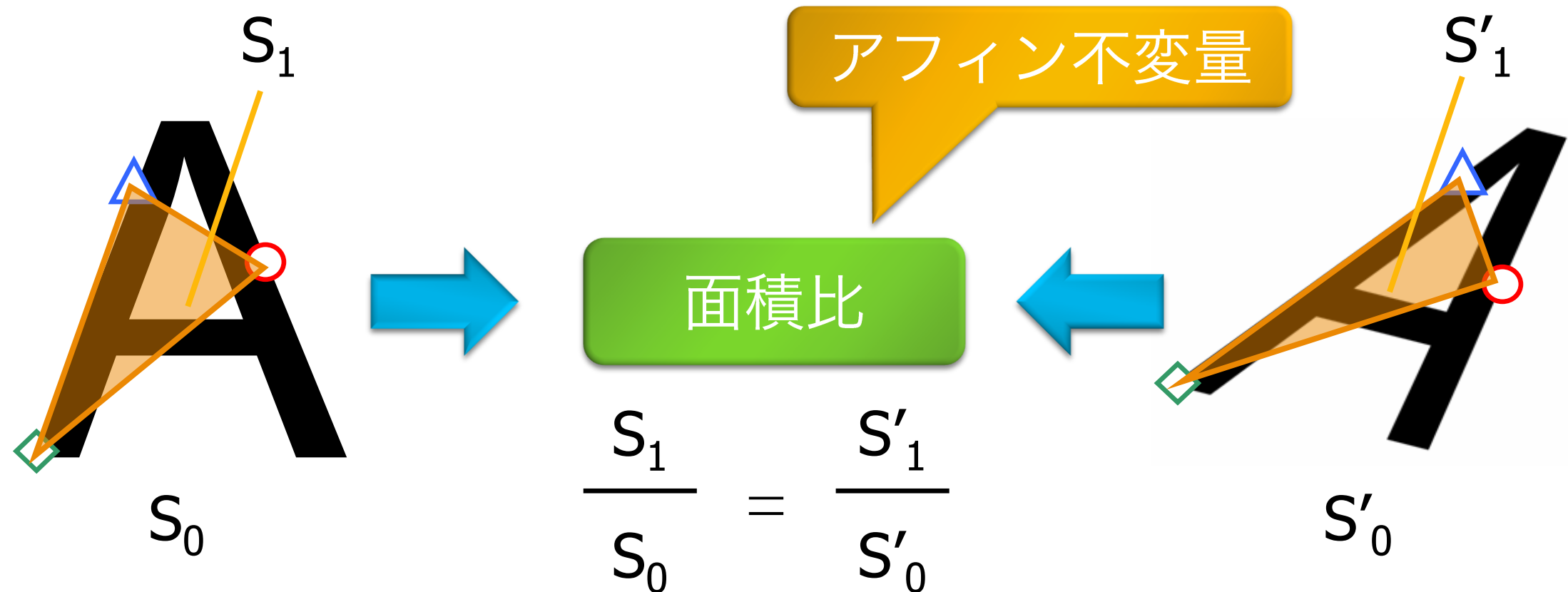
- 登録したテンプレートと
対応しない組み合わせを計算しない



従来手法 1 - 2 : パターン数を削減する原理

通常の方法

- ▶ 面積比
 - ▶ 3点の配置 → 面積比

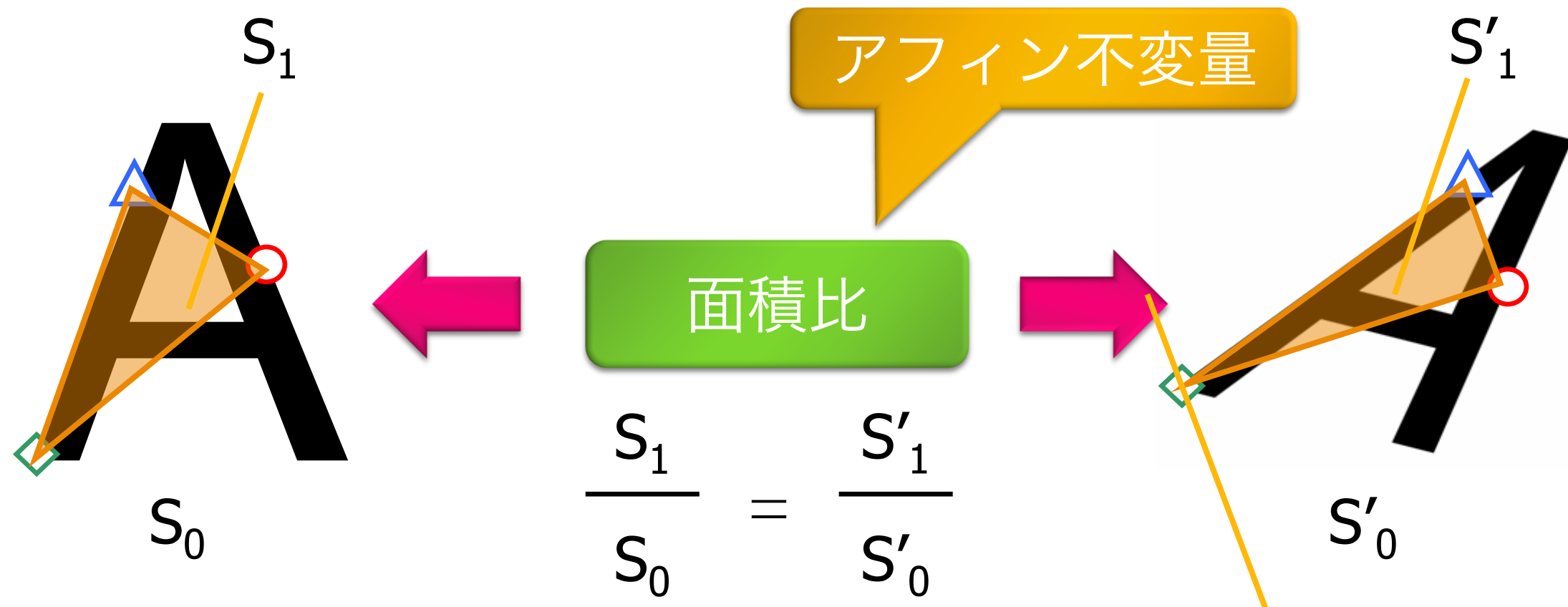


従来手法 1 - 2 : パターン数を削減する原理

通常とは逆の方法

▶ 面積比

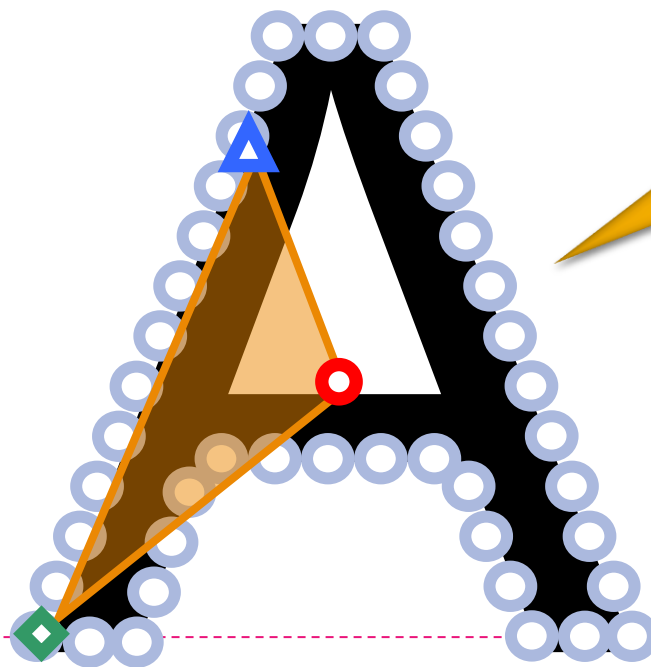
- ▶ 2点の配置 + 面積比 → 3点目の位置



従来手法 1 - 2 :

従来手法のパターンの生成方法

- ▶ 1点目 : 図形の重心 (アフィン歪みに不変) ← 一意
- ▶ 2点目 : 輪郭上の任意の点
- ▶ 3点目 : 面積比によって決定 ← 一意



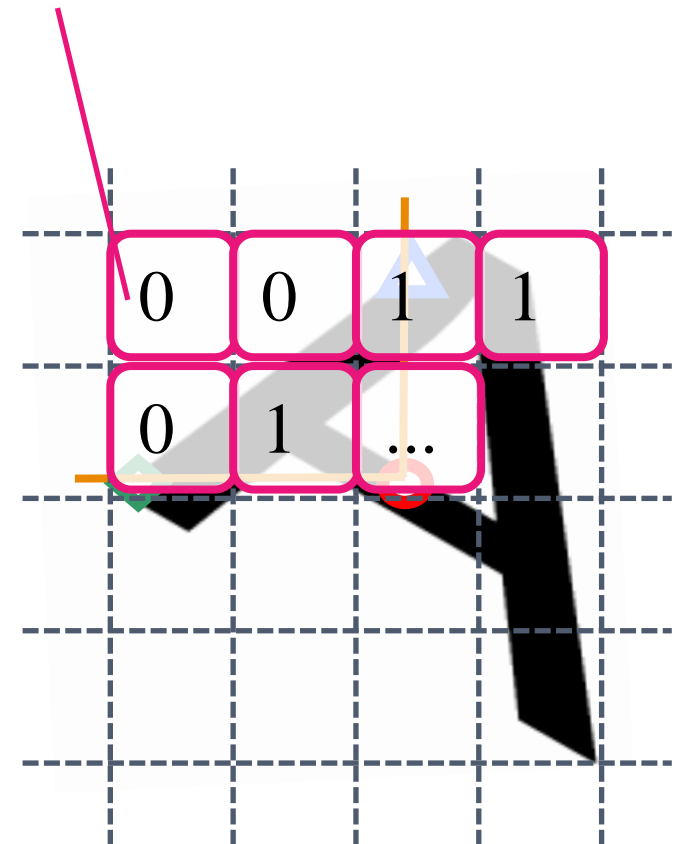
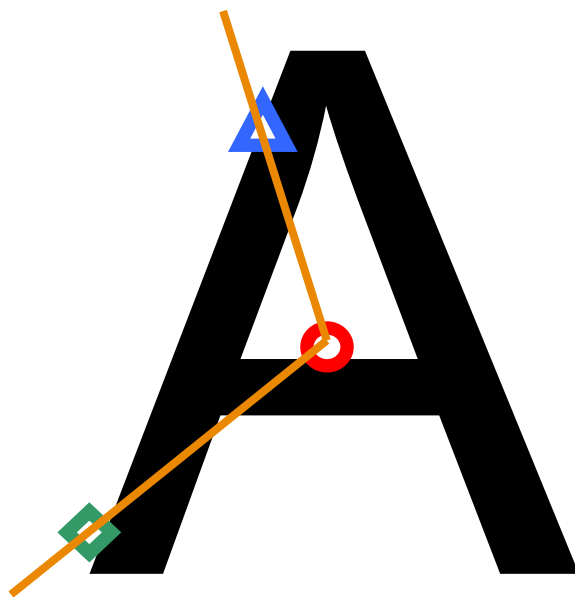
特徴点数 : P

従来手法 1 - 3 : 特徴ベクトルを用いた図形の照合

▶ 特徴ベクトルの計算

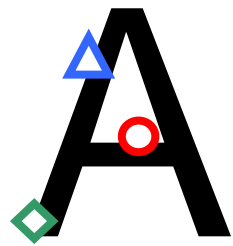
1. 正規化
2. 領域分割
3. 黒画素の割合のヒストグラム作成
4. 量子化

特徴ベクトル

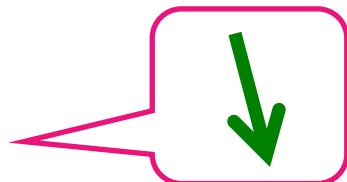
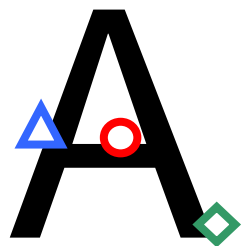


従来手法 1-4 : ハッシュを用いた高速化 — 登録

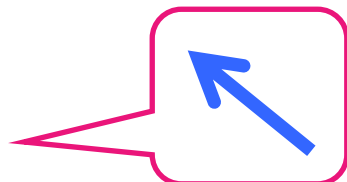
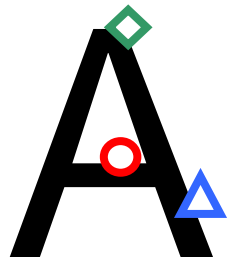
- ▶ 特徴ベクトルをハッシュテーブルに登録



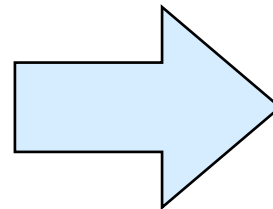
Hash ID : 1



Hash ID : 5



Hash ID : 2



データベース

0

E ↓ M ↓

1

A →

2

A ←

3

B ↑

4

5

A ↓ R ↓

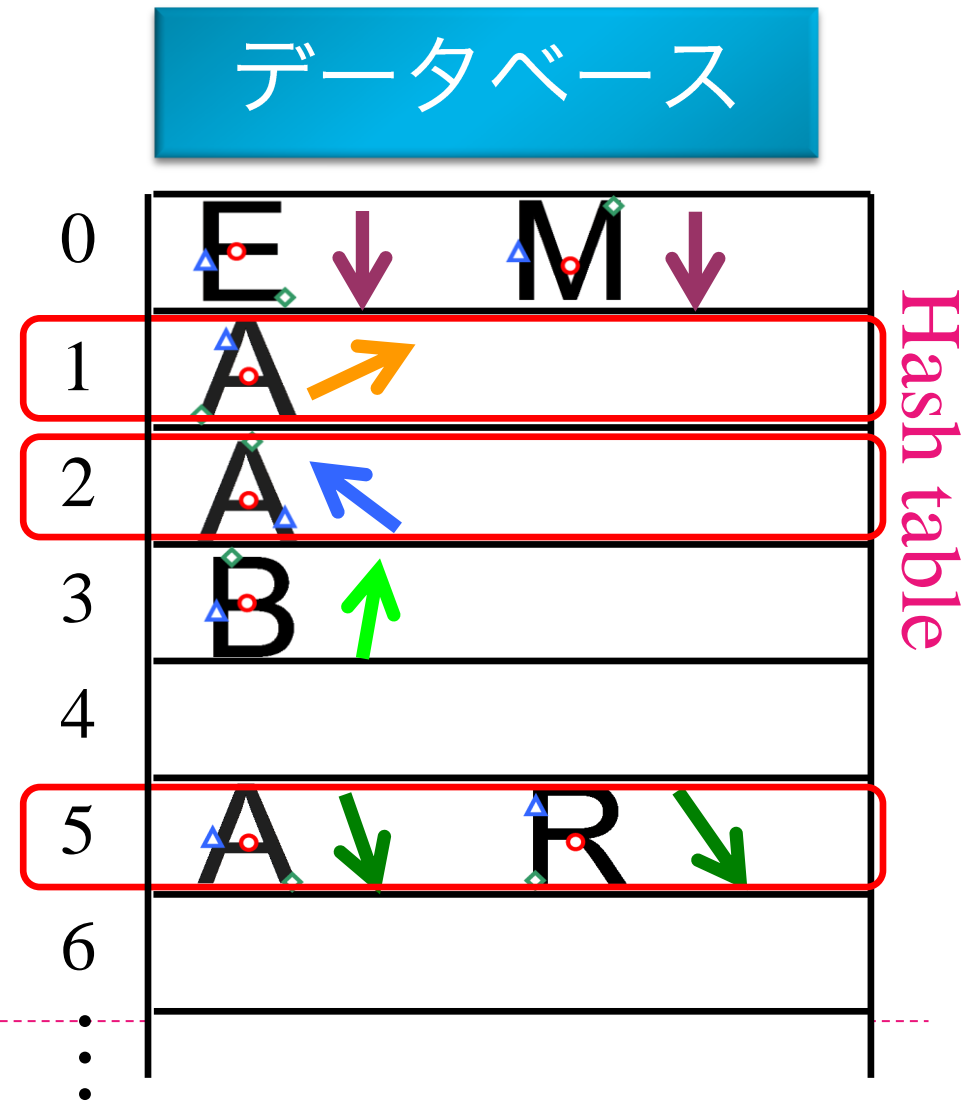
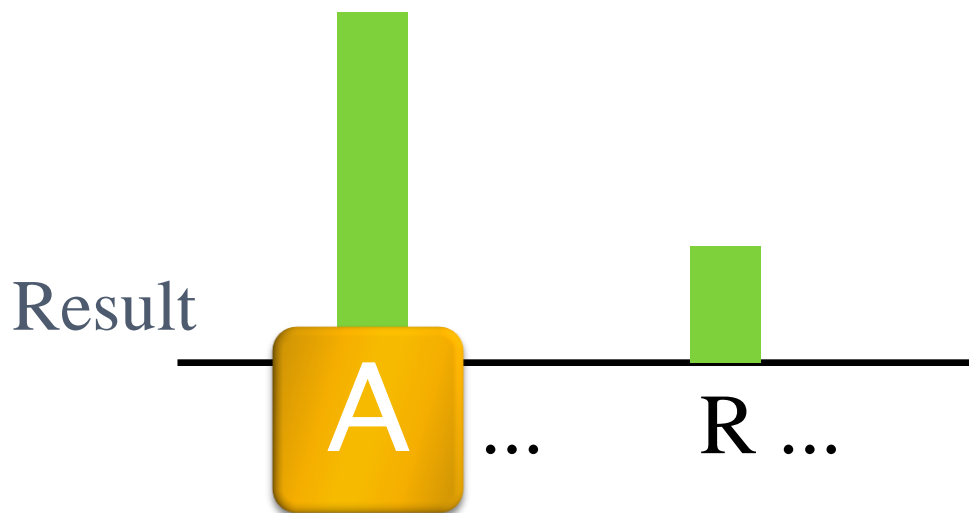
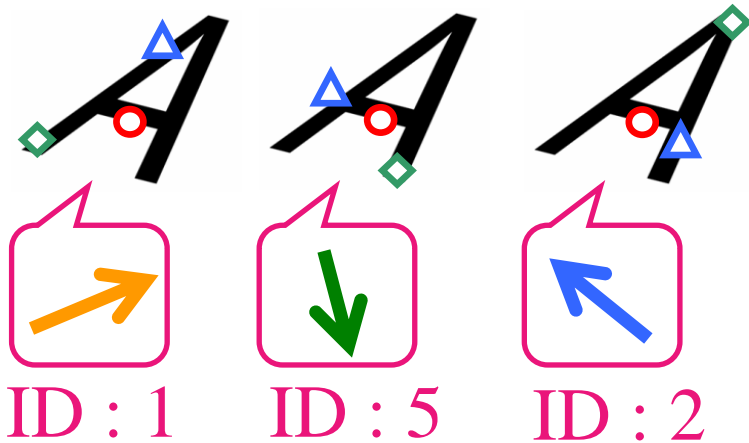
6

⋮

Hash table

従来手法 1-4 : ハッシュを用いた高速化 — 認識 (検索)

1. 特徴ベクトルを作成
2. 字種に投票



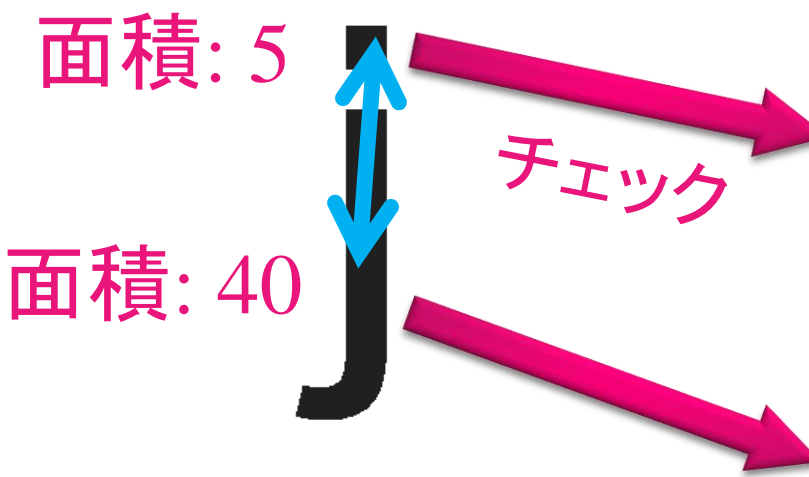
目次

1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ



従来手法2：分離文字の認識

▶ 分離文字テーブルを作成

	連結成分	字種	相対位置	面積	相手の面積
 <p>面積: 5</p> <p>面積: 40</p> <p>チェック</p>	■	i	↓	5	25
	■	j	↓	5	40
	┆	i	↑	25	5
	J	j	↑	40	5

目次

1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ



従来手法 3 : 姿勢推定 (1)

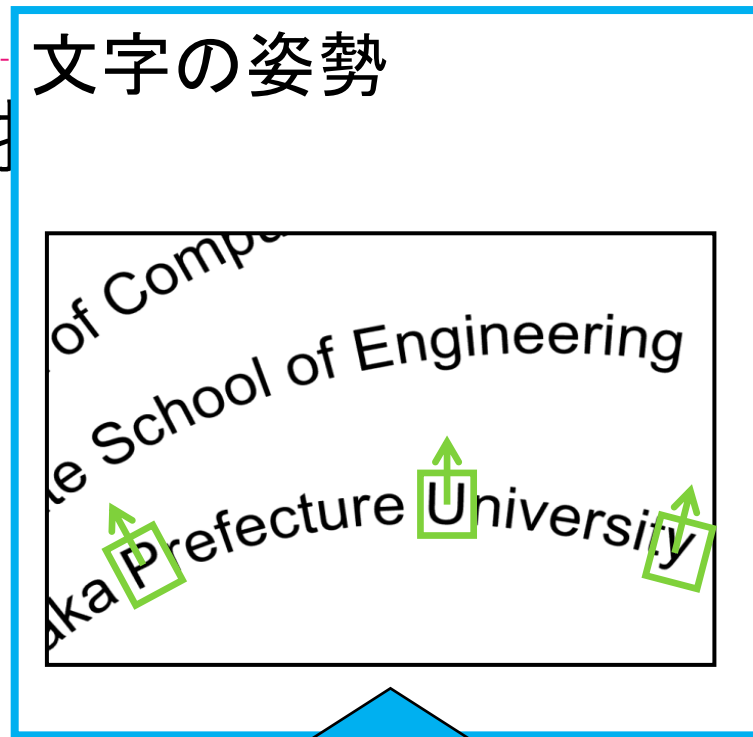
▶ 対

紙面の姿勢



イン変換

文字の姿勢

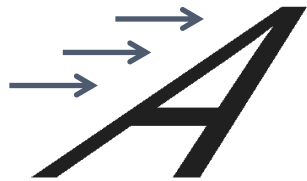


ン変換

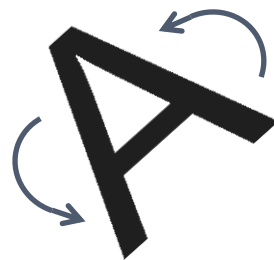
パラメータ



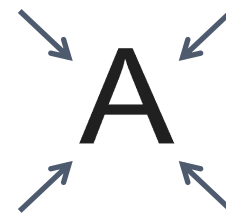
独立変倍



シアー



回転



拡大・縮小

従来手法 3 : 姿勢推定 (2)

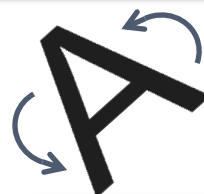
連結成分の対応関係から
パラメータを推定

紙面の姿勢

文字の姿勢



回転角



最も密度の高い点を選択

最も密度の高い点を選択

全連結成分で共通

シアー



連結成分毎に

1 2 3 4 ... 連結成分番号

目次

1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. **提案手法**
 1. 改良1 : 距離計算の導入
 2. 改良2 : 新たなクエリ特徴ベクトルの生成
 3. 改良3 : 登録データの間引き
4. 実験
5. まとめ



提案手法

- ▶ 特定物体認識の高速化に使用した
アイデアを3つ流用

既発表の特定物体認識手法

データベースの大きさ：
100万画像（26億ベクトル）

精度：約90%

計算時間：約60ms

メモリ使用量：33.6GB



目次

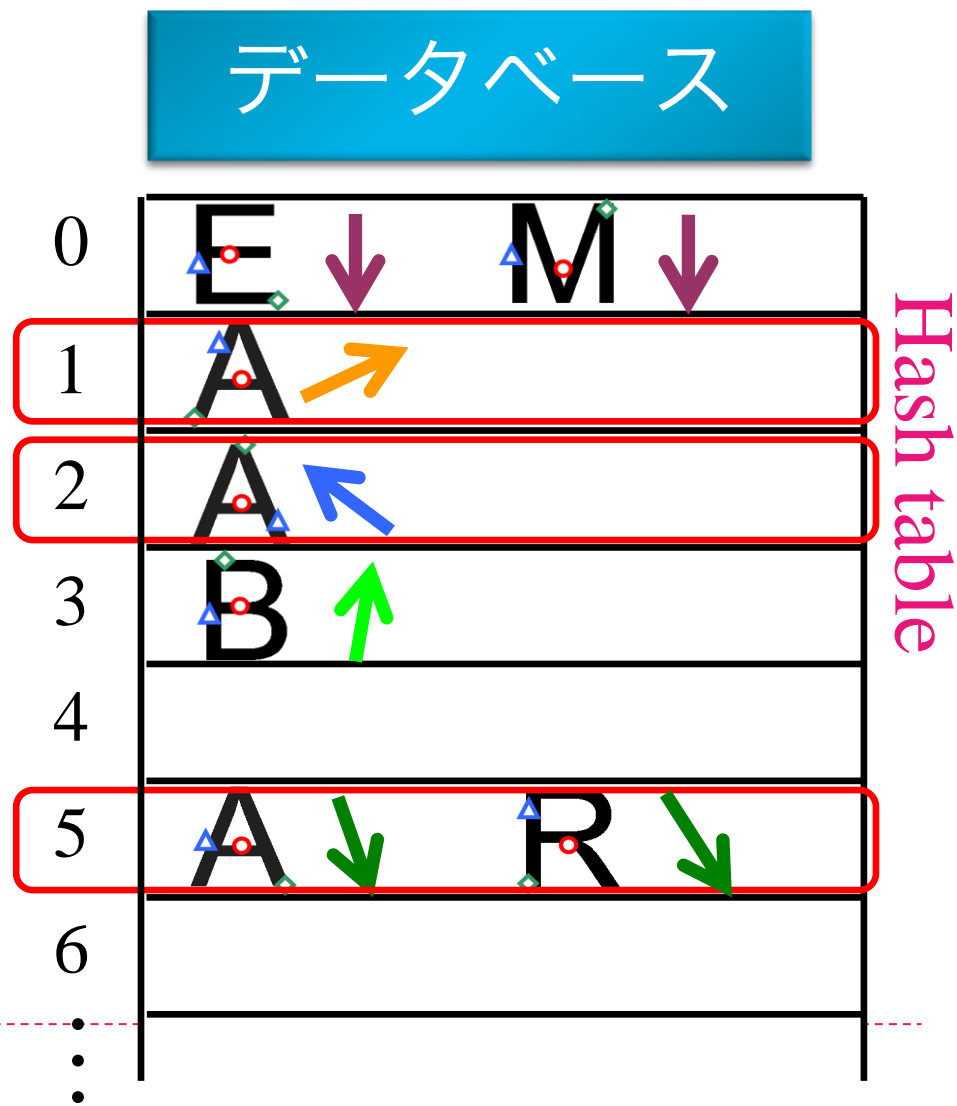
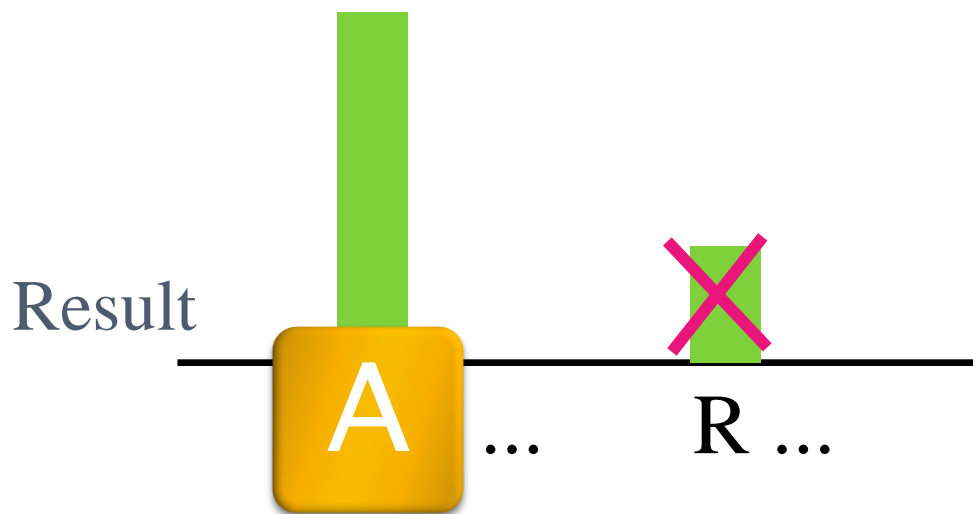
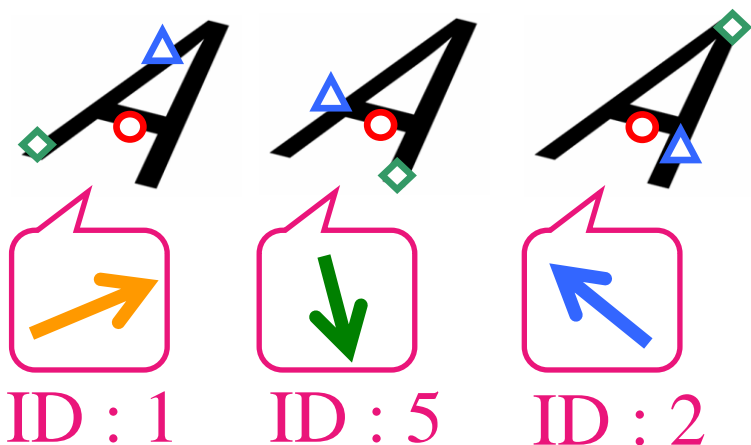
1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1 : 距離計算の導入
 2. 改良2 : 新たなクエリ特徴ベクトルの生成
 3. 改良3 : 登録データの間引き
4. 実験
5. まとめ



提案手法：

改良1：距離計算の導入（1）

1. 特徴ベクトルを作成
2. 字種に投票

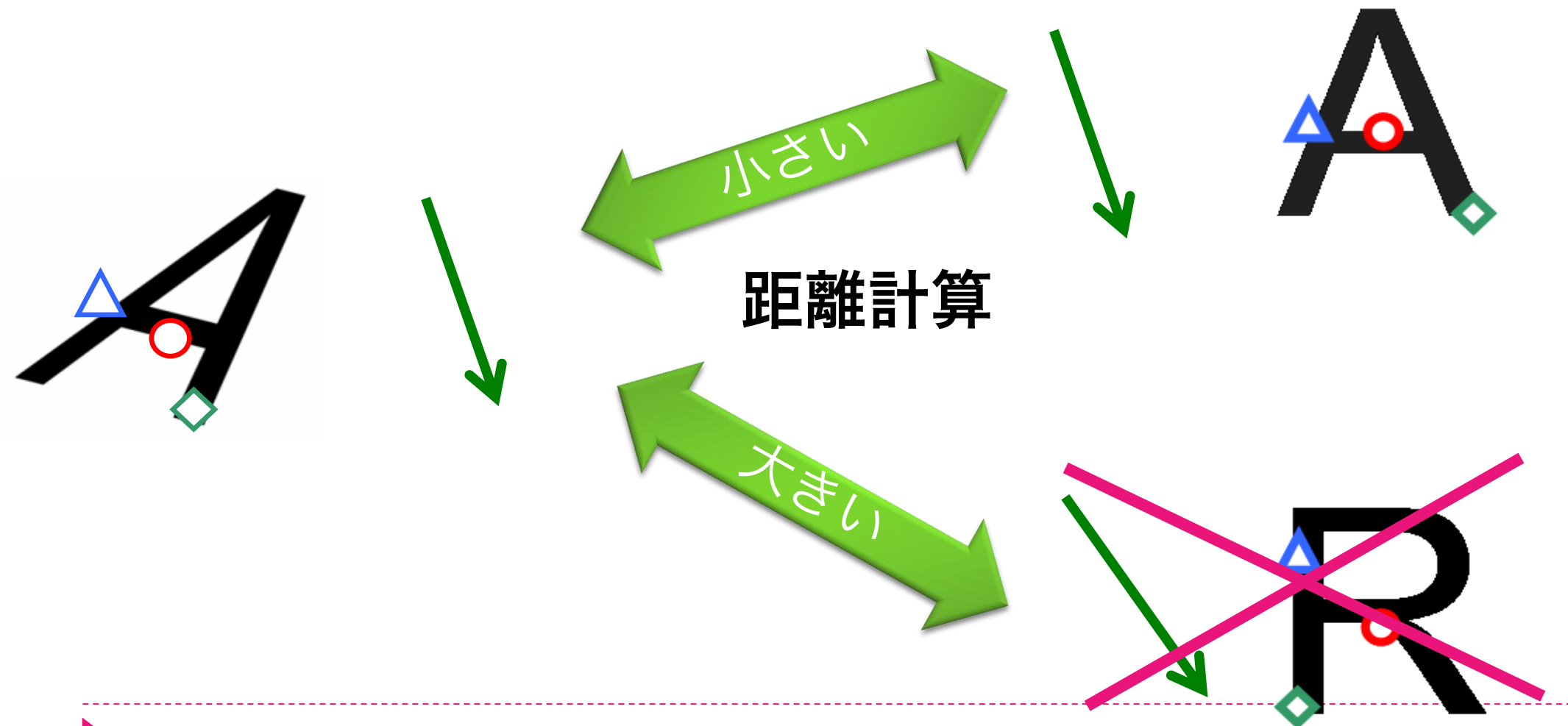


提案手法：

改良1：距離計算の導入（2）

クエリ

データベース



目次

1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ



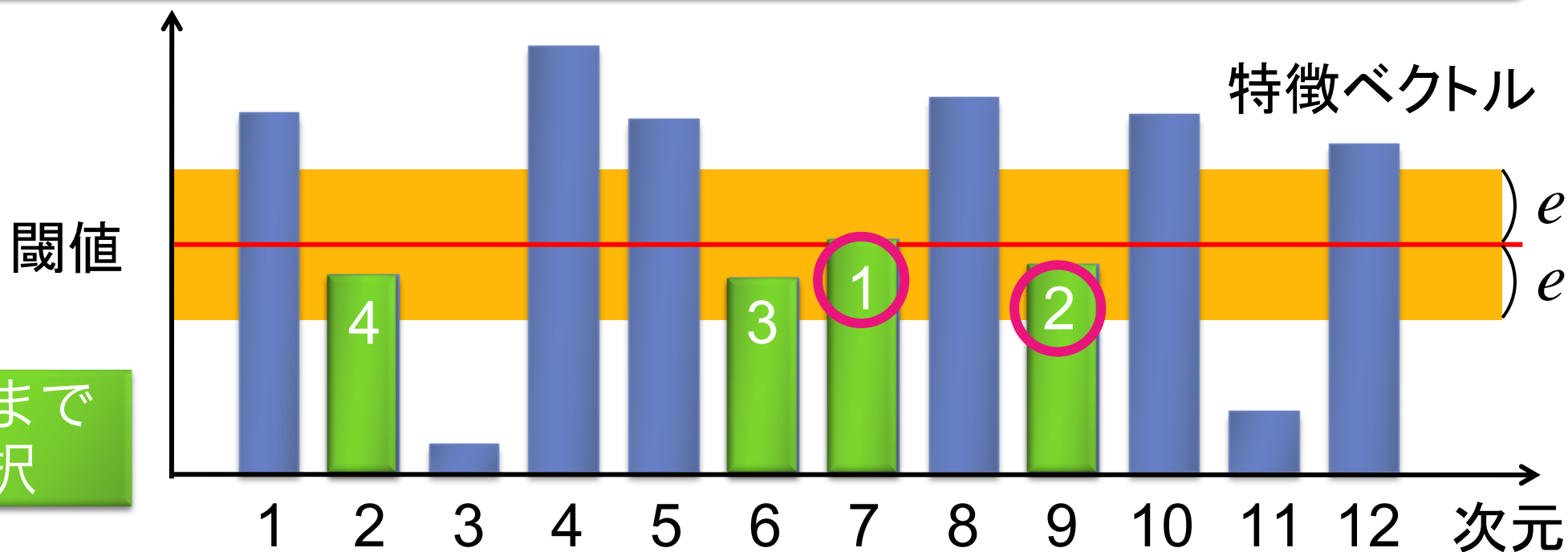
提案手法：

改良2：新たなクエリ特徴ベクトルの生成

新たに生成された二値ベクトル

(1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1)
(1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1)
(1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1)

二値ベクトル (1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1)



目次

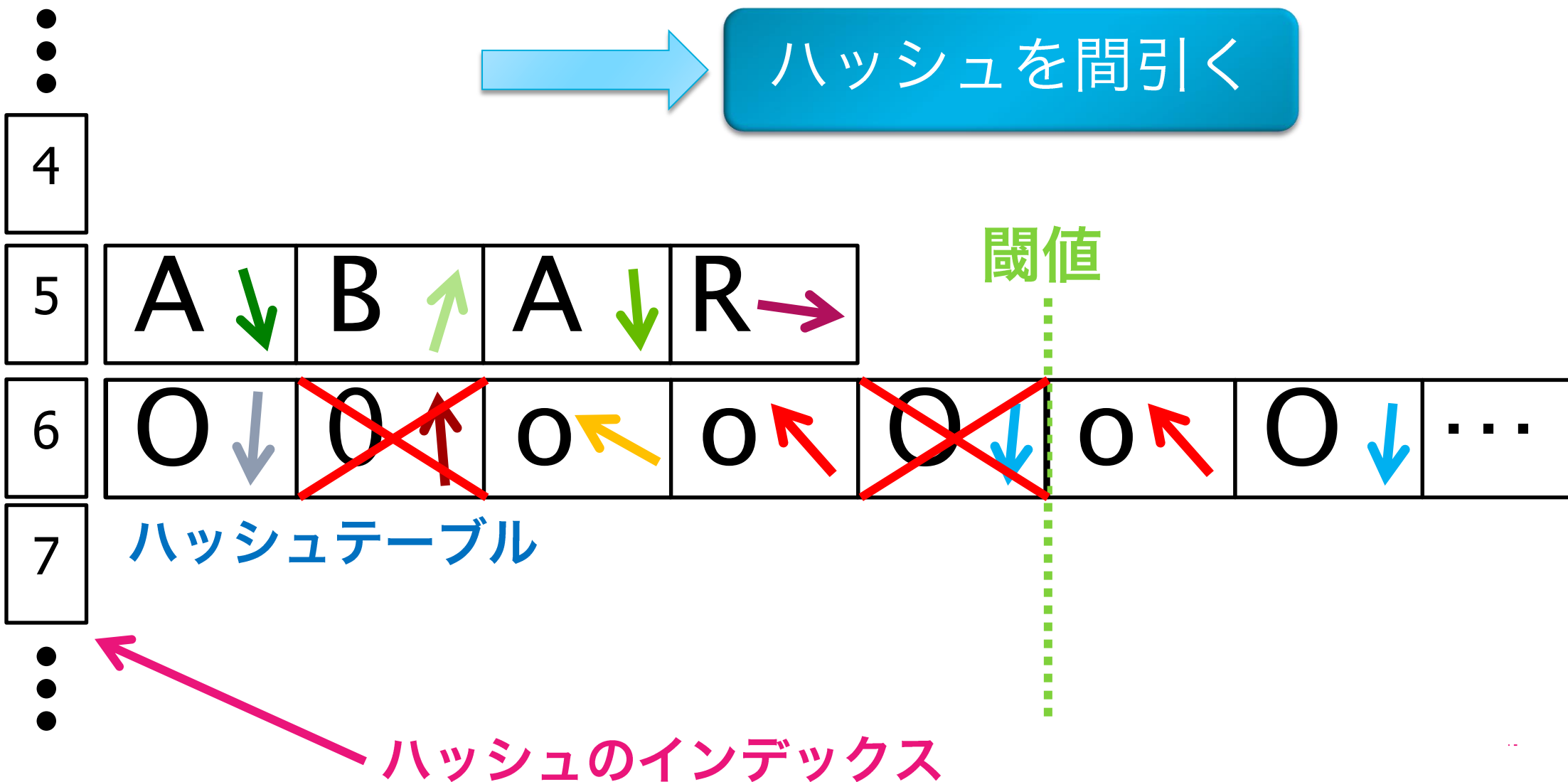
1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ



提案手法：

改良3：登録データの間引き

- ▶ ハッシュの衝突が多いと処理時間を要する



目次

1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ

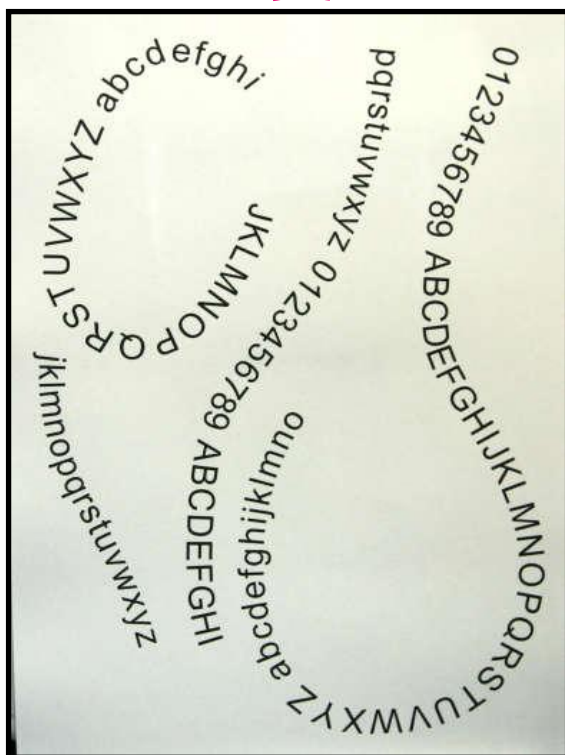


実験対象

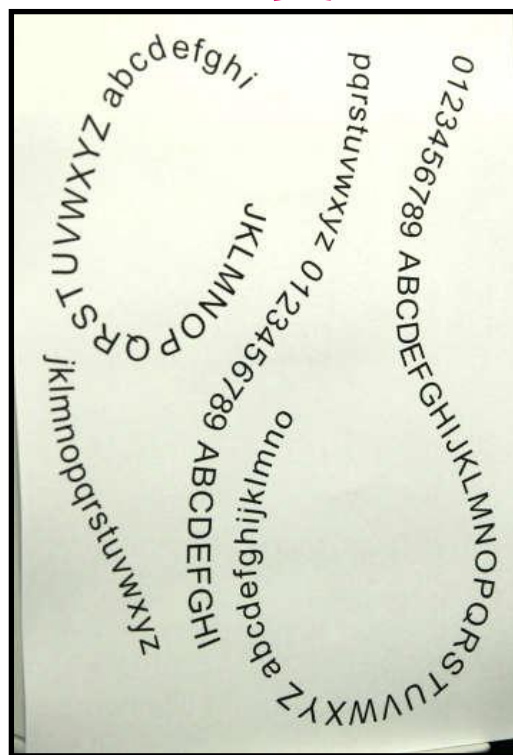
- ▶ 英数字が書かれた文書を3方向から撮影

1枚あたり124文字

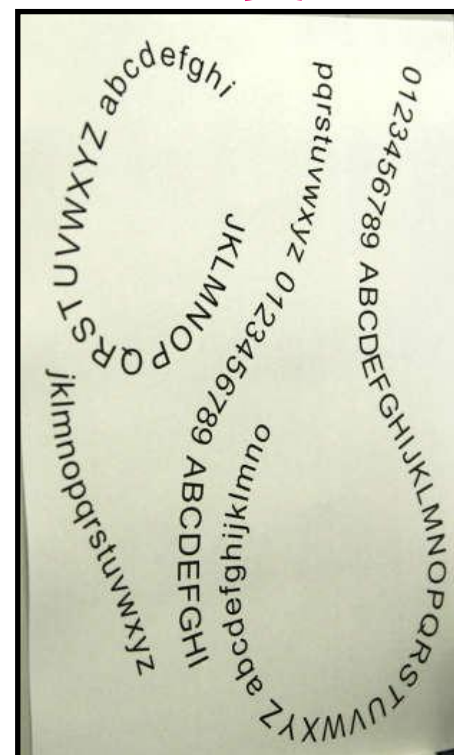
0度



30度



45度



実験条件

- 登録フォント数を増やし、
クラス識別率を計算した

最大100フォント

データベース

0123ABCDabcd

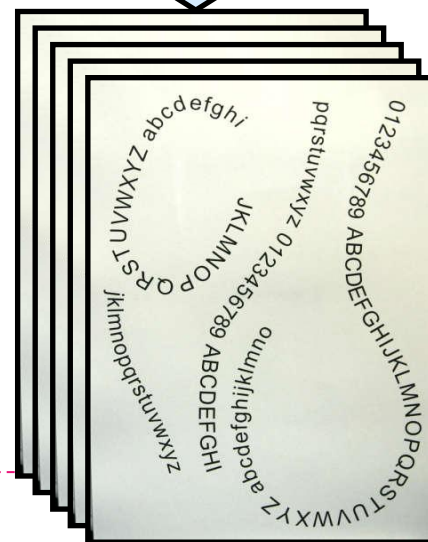
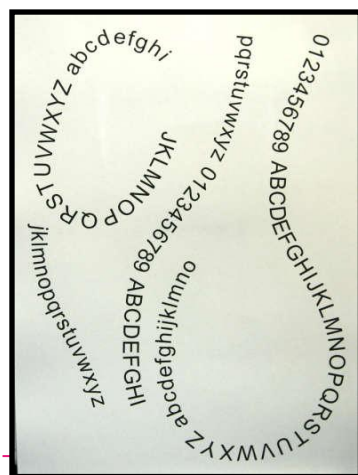
認識

従来手法と
提案手法を比較

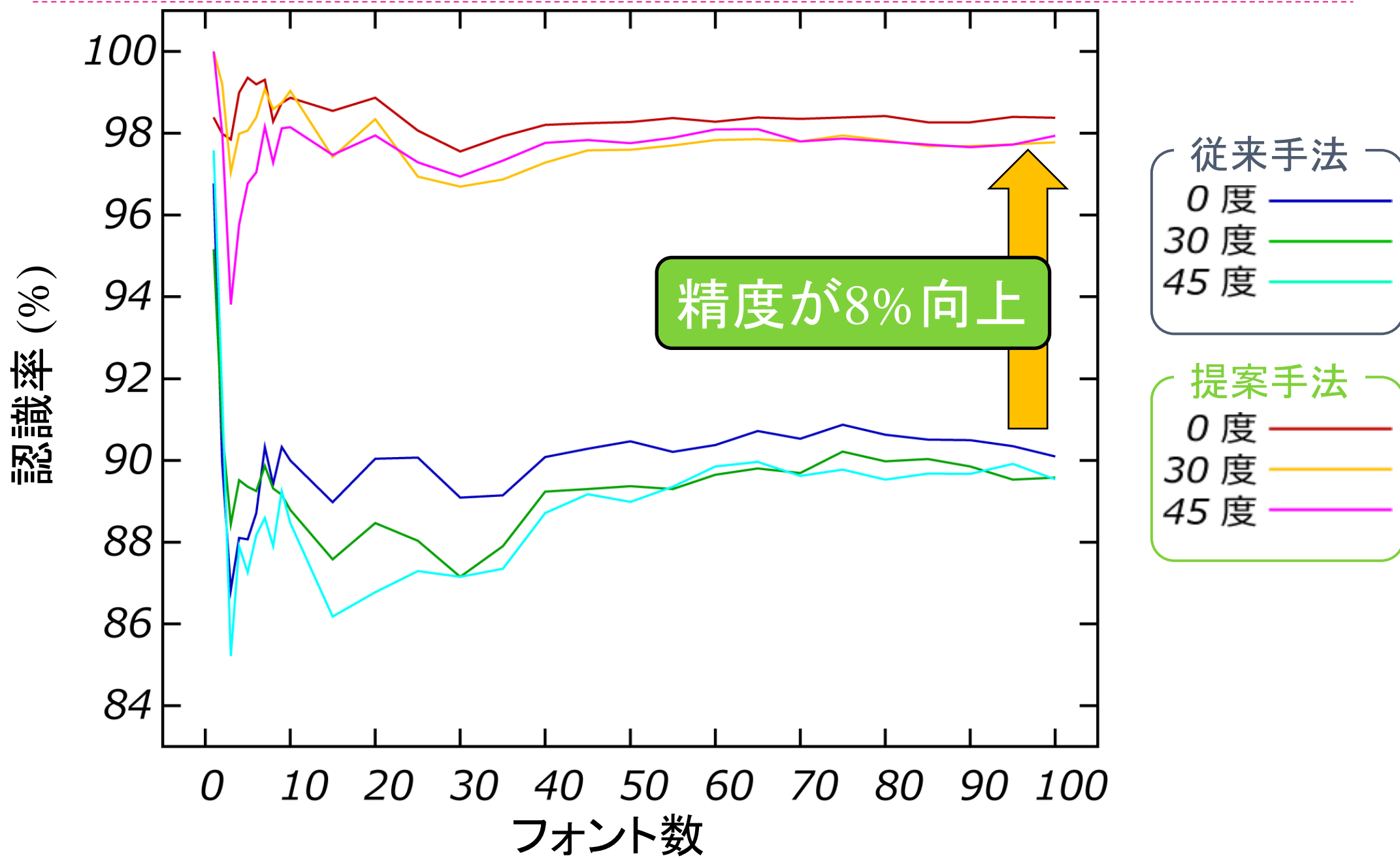
文書

認識

0123ABCDabcd
0123ABCDabcd
0123ABCDabcd
0123ABCDabcd
0123ABCDabcd

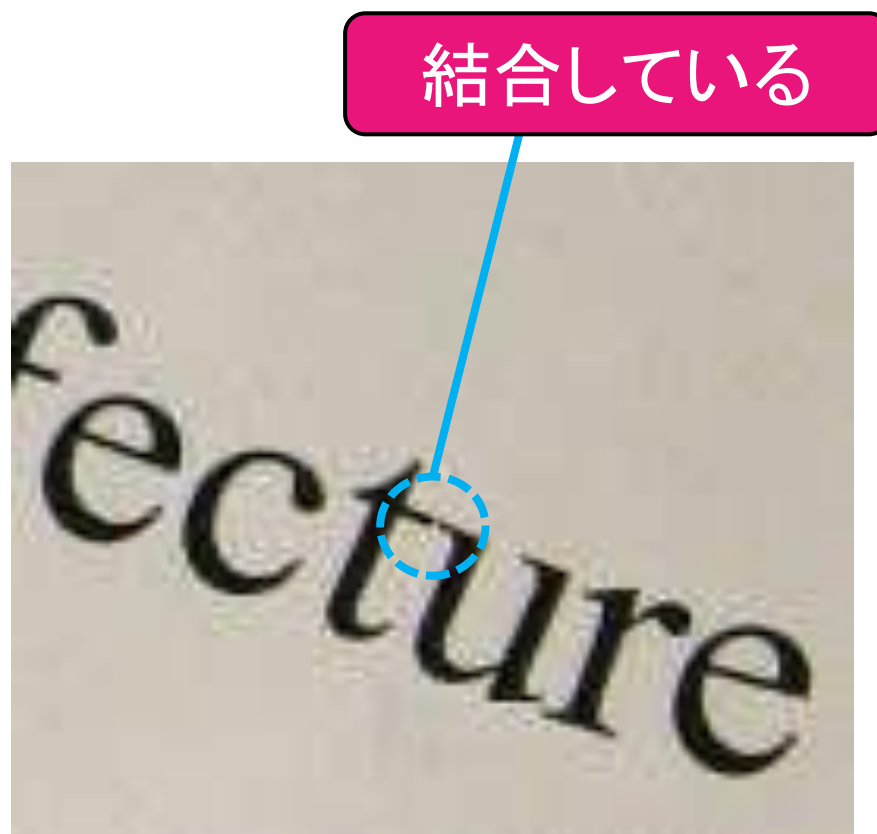


クラス認識率

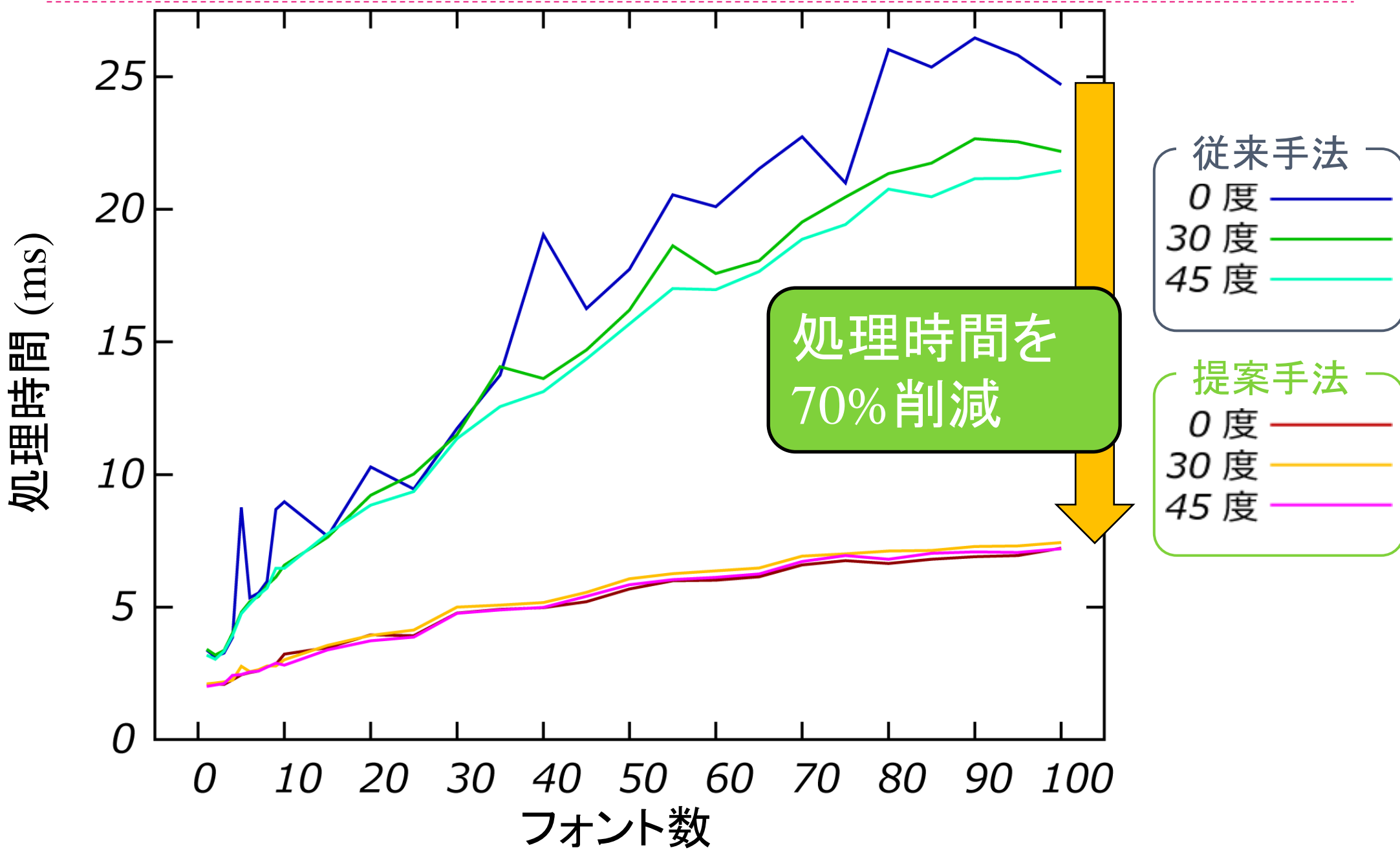


誤認識の例

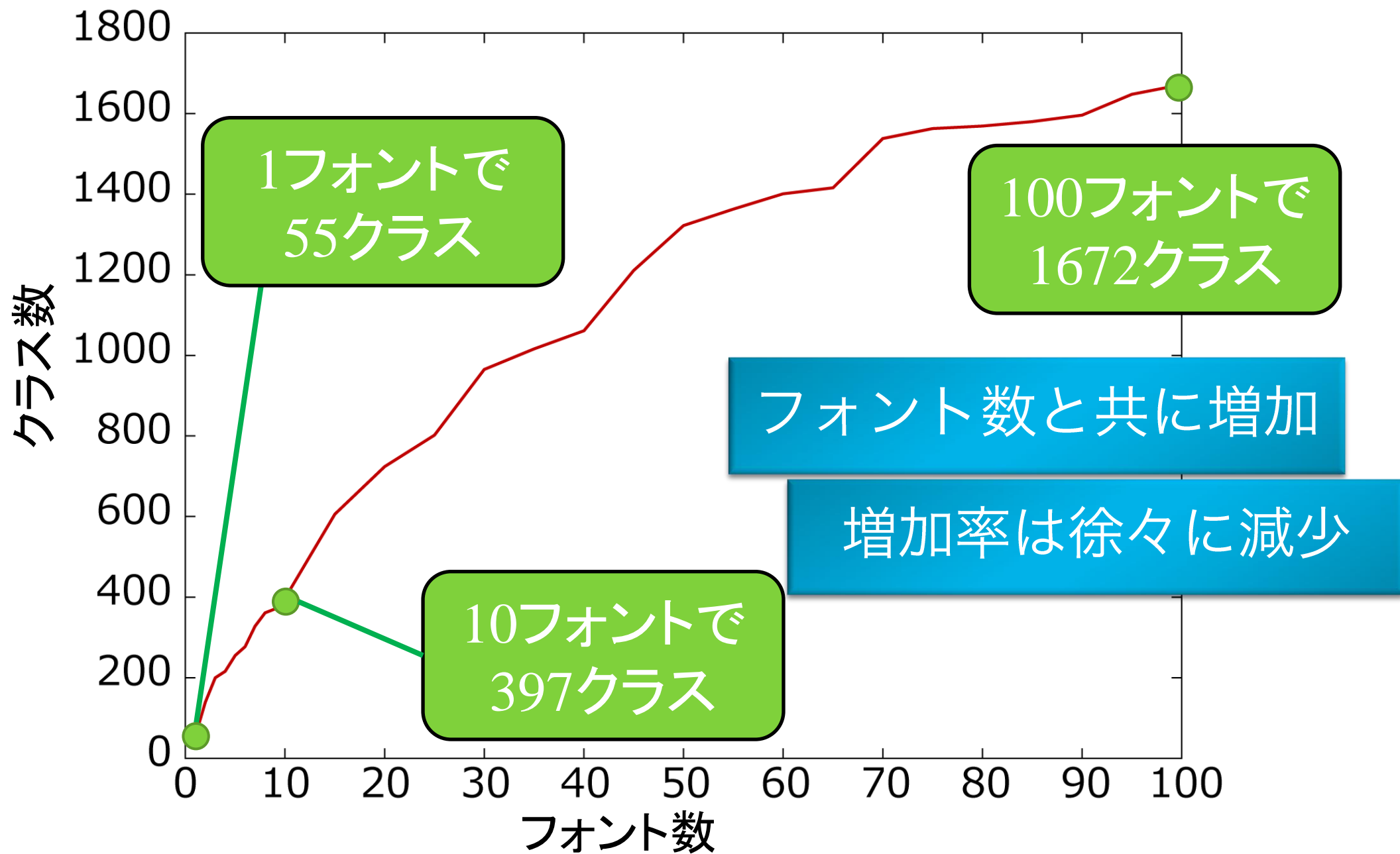
- ▶ 連結成分取得の失敗



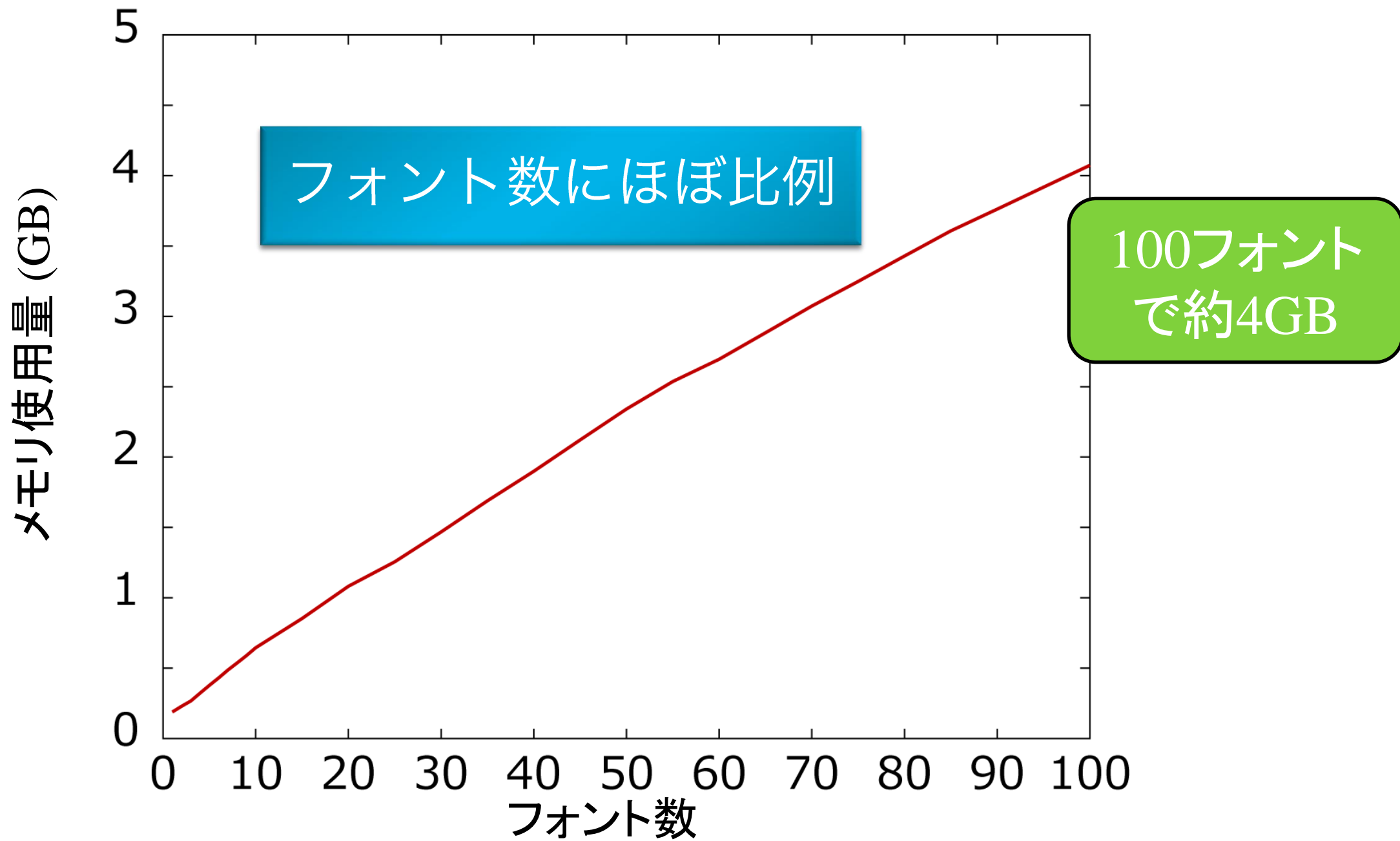
1文字あたりのクラス認識時間



クラス数



メモリ使用量



目次

1. 背景
2. 従来手法
 1. アフィン不変な図形の照合と高速化
 2. 分離文字の認識
 3. 姿勢推定
3. 提案手法
 1. 改良1：距離計算の導入
 2. 改良2：新たなクエリ特徴ベクトルの生成
 3. 改良3：登録データの間引き
4. 実験
5. まとめ



まとめ

- ▶ 100フォントに対応したカメラベース文字認識システムの実現
 - ▶ テンプレートマッチングによるカメラ撮影文字の認識
- ▶ 100フォントを登録したときの性能（正面から）
 - ▶ クラス認識率：98.4%
 - ▶ 計算時間：7.2ms / 1文字
- ▶ 今後の課題
 - ▶ メモリ使用量の削減
 - ▶ 日本語への対応





大阪府立大学
OSAKA PREFECTURE UNIVERSITY



カメラ撮影文字の 事例に基づく実時間認識

岩村雅一 辻智彦 黄瀬浩一