Vol.2017-MPS-112 No.7
2017/2/27

透過型 HMD 上での日本語文字切り出しの一手法

薗田小百合† 石川由羽† 高田雅美† 城和貴†

概要：本稿では，透過型 HMD 上での日本語文字切り出しの手法の提案を行う． OpenCV を用いてカスケード分類器を生成し，日本語文字切り出しを行うアプリケーション上で利用する．カスケード分類器の生成には，Haar-Like 特徴を利用する．Haar-Like 特徴は顔認識で使われている特徴量であり，リアルタイム処理を実現することができる．駅名標をポジティブサンプルとして学習を行い，透過型 HMD 上で動作するアプリケーションで日本語文字の切り出しを行う．学習は，ポジティブサンプル 150 枚，ネガティブサンプル 200 枚で行い，様々なテストデータで切り出しが行われることを確認した．駅名標でない写真をテストデータとした場合でも，日本語文字の切り出しが行えた．

キーワード：透過型 HMD，Haar-Like 特徴，カスケード分類器

A method of Japanese character segmentation on transparent HMD

SAYURI SONODA†
YU ISHIKAWA† MASAMI TAKATA† KAZUKI JOE†

1. はじめに

近年, 訪日外国人旅行者数は増加している[1]. また, 2020 年に東京にてオリンピックおよびパラリンピックの開催が決定しているため，さらなる増加が期待される．しかしながら，訪日外国人旅行者の多くは，日本語に対する知識が少なく，漢字などの文字による情報を理解することが困難である．中華人民共和国や大韓民国などの一部アジアの国々から訪日する外国人の中には，漢字をある程度理解できる旅行者がいるが，ひらがなやカタカナを習得していることは少ない．そのため，日本語と多言語間での翻訳ツールの開発が望まれている．

翻訳ツールとして, Google 翻訳[2]などのスマートフォン上で利用可能なアプリケーションが存在する．これらのアプリケーションは，スマートフォンに搭載されているカメラから得られる情報をもとに，リアルタイム翻訳することができる．この機能は 2015 年 2 月に搭載され，日本語からの翻訳に対応するまでは期間が空き，2017 年 1 月から可能となった．これは，情景画像中から，アルファベットを認識させることは容易であるが，日本語を切り出すことは，その文字の種類の多さから，困難であるためである．

そこで，本稿では，翻訳ツール開発への一助として，透過型ヘッドマウントディスプレイ（Head Mounted Display, HMD）上で動作可能な日本語切り出しアプリケーションを開発する．HMD は，ゴーグルやヘルメットのような形状をした表示装置である．この HMD に，拡張現実感 (Augmented Reality, AR)を実現することによって，スマートフォン上で利用する場合よりも直感的に情景中の文字の場所を理解することができるものと期待される．つまり，より自然な視界上に，翻訳情報を AR によって与える際に，実際の背景を透過させることによって，文字が存在する場所とその翻訳結果を対応させることが容易になり，直感性と利便性を向上させることができる．この際，片眼に対応する HMD にのみ，翻訳に関する情報を掲示するのではなく，両眼に付加することによって，より自然な視界を実現させる．開発する翻訳ツールでは，より利便性を高めるために，リアルタイム処理を目指す．そのため，日本語文字切り出しには，カスケード分類器を利用する．カスケード分類器は顔認識技術として用いられており，リアルタイム処理を可能としている．カスケード分類器の生成のための特徴量としては，Haar-Like 特徴[3]を利用する．Haar-Like 特徴を用いることによって，画像中の各矩形領域の輝度値の平均値の差を特徴量とし，その局所的な特徴を組み合わせることで物体を判別することができる．

開発するアプリケーションでは，両眼シースルー型の EPSON 社の MOVERIO BT-200[4]を採用する．また，顔認識ライブラリとして，OpenCV[5]を用いる．OpenCV を利用した顔認識技術の精度は高く，これを文字の切り出しに応用することで，精度と処理の速さが期待できる．学習データとしては，駅名標の写真を利用する．駅名標とは，鉄道駅にて，当該の駅名を記した標識の一種である．駅名標は，視覚認識されやすいようにデザインされている．テストデータとしては，学習データとは異なる視点から撮影された画像を用いる．これらの画像データは，インターネット上に公開されているものから入手する．

以下，2 章では日本語文字切り出しの既存研究について紹介する．3 章では日本語文字切り出しのためのアプリケーションについて提案する．4 章では，3 章で述べた手法を

† 奈良女子大学
Nara Women's University

用いて日本語文字切り出しを行う．

2. 既存研究

以下，情景画像からの文字切り出しに関連する研究を 2 つ述べる．

1 つ目は，最適 2 次元セグメンテーションによる情景内文字抽出に関する研究である[6]．画像の部分領域に対する文字・非文字の識別には OCR を用いながら，最適な 2 次元セグメンテーションを併用することで，高精度に文字抽出を行う手法を提案している．この研究では，画像を 2 値化する際にしきい値を段階的に変化させると，各文字が明瞭に表れる段階が含まれることに着目し，濃淡画像の各階調をしきい値とする 2 値画像の連結成分と，それらの包含関係から，コンポーネント・ツリーを作成している．評価実験では，英語の文字抽出を行っている．従来の手法を比較し，精度向上が確認できた．データ 1 枚あたりにかかる実行時間は，長いものでは数分に及ぶ．

2 つ目は，2 値化とエッジ抽出による情景画像からの高精度文字列検出に関する研究である[7]．この研究では，NAT(Noise Attribute Thresholding)法と色情報を用いた 2 値化及びエッジ情報を利用し，高精度に文字列を検出する手法を提案している．NAT 法はノイズの特性を利用した文書画像の 2 値化手法である．NAT 法は照明変化などによる色の変化に脆弱であるという問題があるが，色コントラストを考慮した濃淡画像を与えることで，高い精度の 2 値化を可能としている．また，色情報を処理することで．照明変化による誤検出数を抑制する．さらに，従来手法では検出が困難である接触文字や小さい文字列に対しても，文字エッジがもつ特徴を利用することで検出を可能としている．評価実験では，英語の文字の検出を行っている．従来の手法を比較し，精度向上が確認できた．データ 1 枚あたりにかかる実行時間は，従来の手法の 1/4 程度に短縮され，約 62 秒である．

以上の研究では，どちらも従来よりも高精度な結果となっているが，実行時間が長いためリアルタイム処理には利用できないこと，文字切り出しの実験対象が日本語よりも扱いやすい英語のみであるという問題点がある．

3. 日本語文字切り出しアプリケーション

3.1 概要

翻訳ツールの開発において，2 通りの仕様が考えられる．1 つ目は，写真を撮影し，その写真に対して処理を行う方法である．2 つ目は，かざされたカメラに映りこんでいる景色に対して処理を行う方法である．本アプリケーションは，情景画像の中に含まれる日本語を切り出し翻訳することが目的である．そのため，1 つ目の方法を採用する．

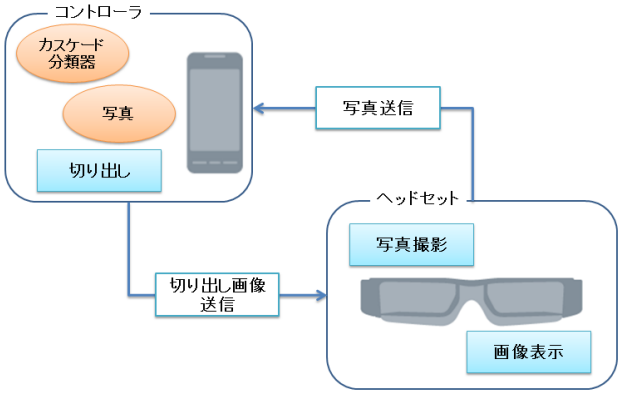

図 1：アプリケーションのシステム構成

リアルタイム処理を実現するために，カスケード分類器を適用する．カスケード分類器で必要となる特徴量として，Haar-Like 特徴を用いる．Haar-Like 特徴では，画像の一部分を切り出し，局所的な明暗差を算出する．そのため，検出率はそれほど高くないが，高速に処理することが可能である．そのため，リアルタイム処理を重視する本アプリケーションに適している．

図 1 にアプリケーションのシステム構成を示す．図 1 のカスケード分類器の作成方法については，3.2 節で述べる．ヘッドセット部分のカメラで現実環境の写真を撮影し，コントローラに送信する．ヘッドセットとコントローラ間のデータの送受信は有線で行う．コントローラ内のアプリケーションで，写真とカスケード分類器を入力として，切り出し画像の生成を行う．生成した画像はヘッドセット部分に送信され，ヘッドセットのディスプレイに表示される．この詳細については，3.3 節で述べる．

3.2 カスケード分類器

カスケード分類器を用いるためには，事前に学習させて，カスケード分類器を生成させておく必要がある．本節では，学習の手順について説明する．なお，基本的な学習の流れは，顔認識に準じる．

1. ポジティブサンプルの収集
2. ポジティブサンプル中の日本語部分を指定
3. ネガティブサンプルの収集
4. ネガティブサンプルを列挙した txt ファイル作成
5. vec ファイル作成
6. カスケード分類器を生成

本アプリケーションにおいて，ポジティブサンプルとは，日本語文字が画像中に含まれる画像で，ネガティブサンプルとは含まれない画像である．画像の拡張子はすべて bmp とする．

手順 1 では，複数枚のポジティブサンプルを収集する．

手順 2 では，ポジティブサンプルに含まれる日本語文字部分を指定する．これは，学習の際にどの部分が日本語文字であるかの情報が必要であるためである．指定には，日本語文字部分を四角で囲んだ際の左上を始点として，始点の座標と始点からの幅と高さが必要である．また，画像中に複数日本語文字が含まれる場合は，すべて指定する．これらの情報を txt ファイルで作成する．

手順 3 では，複数枚のネガティブサンプルを収集する．つまり，日本語文字が画像中に含まれていないことを確認し，収集する．

手順 4 では，ネガティブサンプルを手順 6 で利用可能にするために，ネガティブサンプルのファイル名を列挙した txt ファイルを作成する．

手順 5 では，ポジティブサンプルを手順 6 で利用可能にするために，手順 2 で作成された txt ファイルを，vec ファイルに保存する．なお，vec ファイルとは，ベクトルファイルのことを意味する．ベクトル化する際に，幅と高さの分割数を与える必要がある．

手順 6 ではカスケード分類器を生成する．生成には，Haar-Like 特徴を用いる．学習終了後，xml 形式のデータが出力される．このデータがカスケード分類器であり，アプリケーションで利用される．

3.3 アプリケーション

アプリケーションにおける処理の流れについて説明する．

a. テストデータ取得
b. カスケード分類器読み込み
c. 日本語文字検出
d. 検出範囲を描画
e. 結果画像表示

手順 a では，テストデータの画像を取得する．切り出しを行う画像は，カメラから取得する．ここで，テストデータは，学習に用いたものとは別の画像でなければならない点に注意が必要である．画像の形式は bmp とする．

手順 b では，カスケード分類器の読み込みを行う．

手順 c では，テストデータから日本語文字の検出を行う．検出された物体は，リスト構造で返される．

手順 d では，検出された結果の描画を行う．この際，HMD の描画形式に従う必要がある．たとえば，カメラで取得された画像のうち，日本語文字が検出された部分のみを切り出し，現実環境に合成する方法が考えられる．

手順 e では，得られた結果画像を HMD 上に，表示する．

図 2 : MOVERIO BT-200

4. 実験

4.1 環境

HMD として，透過型 HMD である MOVERIO BT-200 を用いる．図 2 に MOVERIO BT-200 を表す．MOVERIO BT-200 は EPSON 社から開発者用システムソフトウェアが提供されている[8]．これを適用することにより，adb(Android Debug Bridge)が利用できる．adb とは，Android のデバッグをサポートするツール群であり，これにより USB を経由するアプリケーションのインストールや端末ログの取得を行うことができる．特に，Android アプリケーションの開発において，端末ログの取得は重要である．端末ログを取得することで，画面に表示されないエラーや警告を知ることができる．一方，MOVERIO BT-200 に開発者用システムソフトウェアを適用することにより位置情報を取得するための GPS が利用できなくなるなどの問題が生じる．そこで，開発中の動作試験にはエミュレータのみを利用する．Android Studio 純正のエミュレータは動作が重く，リアルタイム処理が重要である動作確認には不十分なため，動作が比較的高速である Oracle VM VirtualBox[9] と Genymotion[10]をエミュレータとして利用する．

使用した OS は Windows7（64 ビット）である．CPU は，Intel®Core™2 Duo CPU E8400@3.00GHz，メモリは 4GB である．学習の際は，OpenCV2.4.9 と Visual Studio Community 2013[11]を用いる．アプリケーションの開発には，OpenCV2.4.9 と Android Studio1.4[12]を用いる．プラットフォームとして，学習には Windows を利用し，アプリケーションの動作には Android を利用する．これは，MOVERIO BT-200 が本体に Android™4.0 を搭載しているためである．アプリケーション実行より前の段階で学習を行うため，学習とアプリケーションのプラットフォームが分かれている．

実験で用いるポジティブサンプルおよびネガティブサンプルの画像は，本来は，MOVERIO BT-200 の画面から取得すべきである．しかしながら，本実験は，エミュレータ上で行うため，Google 画像検索で取得する．取得対象は，駅名標で，同じ角度から撮影されたと思われる画像は 1 枚のみ使用する．また，画像の拡張子は，bmp とする．

カスケード分類器を生成する際に必要となるベクトルファイルの作成では，OpenCV の createsamples ユーティリ

ティを用いる．createsamples ユーティリティはコンソールであるため，コマンドライン引数で情報を与える．使用するコマンドライン引数は-vec, -info, -num, -w, -h である．-vec は出力先のファイル名である．-info は与えるポジティブサンプルを記述する．-num は出力されるポジティブサンプルの数である．-w と-h とは出力されるポジティブサンプルの幅と高さである．ここではデフォルトと同様の 24 を与える．

カスケード分類器の生成には，OpenCV が用意している traincascade ユーティリティを使用する．traincascade ユーティリティはコンソールであるため，コマンドライン引数で情報を与える．使用するコマンドライン引数は-data, -vec, -bg, -numPos, -numNeg, -featureType, -w, -h である．-data は出力先のフォルダ名である．-vec にはポジティブサンプルから作成された vec ファイルを与える．-bg ではネガティブサンプルのリストを与える．-numPos は学習の際のそれぞれのステージで使用するポジティブサンプルの数を与える．ただし，ポジティブサンプルの数と同じ値を与えるとエラーとなる．これは，学習の際にポジティブサンプルがポジティブサンプルでないと誤認識される可能性があり，誤認識されたポジティブサンプルは学習に利用できないためである．ポジティブサンプルでないと誤認識された画像は，以降ポジティブサンプルのデータ群から除外される．すると，ポジティブサンプルの全体数が減り，新たに別のポジティブサンプルを補充し，利用することになる．そのため，あらかじめ補充することを予測し，手順 5 で作成するポジティブサンプルの数より少ない数値に設定する．そこで，以下の式(1)に従って，numPos の値を決定する．

$$\text{numPos} \leq \frac{num - S}{numStages + minHitRate(1 - numStages)} \quad (1)$$

num は手順 5 で作成するポジティブサンプルの数である．numStages は学習のステージ数であり，minHitRate は各ステージが満たす最少認識率である．式(1)は，各ステージでのポジティブサンプルが誤認識される数を予測する式をから導き出される．また，予測を超えて誤認識されることもあるため，S には適当な数値を代入して，numPos に余裕をもたせる．-numNeg はネガティブサンプルの数を与える．-featureType は，学習に利用する特徴の型を指定する．本稿では，Haar-Like 特徴を指定する．-w, -h はサンプルのサイズとして 24 を与える．

アプリケーションで用いる画像は，bmp 形式である．しかしながら，カスケード分類器として用いる OpenCV の CascadeClassifier では Mat クラスを用いる．Mat は，画像の幅，高さ，ビット深度などのデータを保持する．そこで，bitmapToMat メソッドを用いて画像変換を行う．また，画像比較において，色情報は不要であるため，cvtColor メソッドを利用してグレースケール変換する．


(1)ポジティブサンプル　(3)ネガティブサンプル

図 3：学習に利用するサンプルの例


(1)宇治駅　(2)新宿駅


(3)直江津駅　(4)春日大社千年の至宝展の紹介看板

図 4：テストデータの例

日本語文字の検出には，detectMultiScale メソッドを利用する．

4.2 切り出し結果

学習に利用するポジティブサンプルとネガティブサンプルの例を図 3 に示す．図 3(1)はポジティブサンプルである宇治駅の駅名標の写真である．図 3(2)はネガティブサンプルである日本語文字の含まれないロゴ画像である．学習に利用するポジティブサンプルの数は 150 枚，ネガティブサンプルの数は 200 枚である．

アプリケーション実行に利用するテストデータの例を図 4 に示す．図 4(1)は宇治駅の駅名標の写真である．学習に利用するポジティブサンプルにも宇治駅の駅名標の写真は含まれているため，別の角度から撮影している写真を扱う．図 4(2)は，新宿駅の駅名標の写真である．学習に利用しているポジティブサンプルには，新宿駅の駅名標の写真は含まれていない．図 4(3)は，直江津駅の駅名標の写真である．新宿駅と同様に，学習に利用しているポジティブサンプルには，直江津駅の駅名標の写真は含まれていない．図 4(4)は，春日大社千年の至宝展の紹介看板である．緑の

線で囲まれている部分が，日本語文字を検出した場所である．

4.3 考察

図 4(1)のように，学習に利用した駅名標については，別の角度から撮影されている写真でも検出が成功している様子が確認できる．また，図 4(2)のように，学習に利用していない駅名標についても，検出が成功することが確認できるものもあるが，図 4(3)のように，検出が成功しているとはいえないものもある．これは，ポジティブサンプルの中に検出したい部分と類似したものがある場合とない場合の差であると考えられる．

図 4(4)は，イラストの部分に誤検出が起きているが，一番大きく表示されている「春日大社」の文字を検出できている．駅名標のみの学習であっても，看板の切り出しに有効であることが分かる．さらに検出された日本語文字は，文字の背景がイラストとなっていることから，駅名標のように背景が単色のものでなくても，本手法は有効であることが分かる．

今回の学習データはポジティブサンプルの数は 150 枚，ネガティブサンプルの数は 200 枚と非常に少なく，写真の種類も駅名標のみであるが，図 4(4)のような看板の切り出しも行えることが確認できたため，今後学習データの数や種類を増やすことで更なる精度の向上が期待できる．

5. まとめ

本稿では，OpenCV を用いて日本語文字を検出するカスケード分類器を生成した．また，生成したカスケード分類器を利用して情景画像から日本語文字切り出しを行うアプリケーションの開発を行った. アプリケーションの動作は，スマートフォンより直感性と利便性の高い HMD 上で行う．また，アプリケーションにはリアルタイム処理が求められるため，高速な処理が可能な Haar-Like 特徴を利用して，日本語文字切り出しを行う. カスケード分類器の生成には，OpenCV を用いる．

開発したアプリケーションの実験では，日本語の学習に駅名標写真を利用したカスケードファイルを用いて，情景画像から日本語切り出しを行えることが確認できた. また，テストデータとして，学習に用いていない駅の駅名標写真を用いた場合や，駅名標でない写真を用いた場合でも，誤検出はあるものの，日本語切り出しが行えており，本アプリケーションの有効性が確認できた．

今後は，学習データの数や種類を増やすことで，更なる精度向上を目指す．また，現在は画像中から日本語検出を行った後，切り出しを正方形で行っているが，認識された範囲に合わせた切り出しを行う工夫を加えることで，より正しい切り出しを目指す．

参考文献

[1]訪日外国人旅行者数（平成 28 年 9 月）に関する配付資料, http://www.mlit.go.jp/kankocho/page01_000542.html（参照：2017/01/30）
[2]Google 翻訳, https://play.google.com/store/apps/details?id=com.google.android.apps.translate&hl=ja（参照：2017/01/30）
[3]Paul Viola, Michael Jones : Rapid Object Detection using a Boosted Cascade of Simple Features, IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, Vol.1, pp.511-518, 2001
[4]MOVERIO BT-200, http://www.epson.jp/products/moverio/bt200/（参照:2017/01/30）
[5]OpenCV, http://opencv.org/（参照:2017/01/30）
[6]武部浩明, 内田 誠一 : 最適 2 次元セグメンテーションによる情景内文字抽出, 電子情報通信学会論文誌, J97-D(3), pp.667-675, 2014
[7]松田友輔, 大町真一郎, 阿曽弘具 : 2 値化とエッジ抽出による情景画像からの高精度文字列検出, 電子情報通信学会論文誌, J93-D(3), pp.336-344, 2010
[8]MOVERIO BT-200 開発者用システムソフトウェア, https://tech.moverio.epson.com/ja/life/bt-200/pdf/bt200_rlnd112_ja.pdf_（参照：2017/01/30）
[9]Oracle VM VirtualBox, http://www.oracle.com/technetwork/jp/server-storage/virtualbox/overview/index.html（参照：2017/01/30）
[10]Genymotion, https://www.genymotion.com/（参照：2017/01/30）
[11]Visual Studio Community 2013, https://www.visualstudio.com/en-us/news/releasenotes/vs2013-community-vs（参照：2017/01/30）
[12]Android Studio, http://developer.android.com/intl/ja/sdk/index.html（参照：2017/01/30）