[情報通信プロジェクト]

実験レポート

レポート提出日：2019年1月25日

実験者：16173009 林田和磨

16173064 伊藤光太郎

18273002 平尾礼央

18173003 伊藤広樹

班名：B1班

1. **実験の目的と内容**

「キムタクに一番似ているのは誰か」という疑問や、「顔をパスワードの代わりとして利用したい」と言った要求に答えるシステムを開発する。代表的なパターン識別手法 (K 最近傍法、部分空間法等) を学びながら、高速で認識率のよいアルゴリズムを作成する。

　今回の実験では与えられたデータセットを用いておこなう。データセットはデータベースとクエリから構成され、データベースには20人の画像がそれぞれの人物に対し10枚ずつの顔画像が収録されている。一方、クエリにはデータベースに登録されていない人物の画像を含め合計58枚の顔画像が含まれている。

　本実験ではクエリから顔画像を任意に選び、その画像がデータベースのどの人物かを判定する顔認識システムを開発する。

1. **開発環境および、各手順の担当(役割分担)**

B1班ではMATLAB(R2015a)およびPythonを使用して実験をおこなった。

Python担当:伊藤広樹、平尾礼央

MATLAB担当:伊藤光太郎、林田和磨

各手順において、それぞれの担当分は以下のとおりである。

* 顔のトリミング、正規化を実装(Python)
* 学習データの水増し(Python)
* 画像データの入力部(MATLAB(林田和磨))
* Canny法によるエッジ検出(MATLAB(伊藤光太郎))
* 離散コサイン変換(DCT)で特徴抽出(MATLAB(伊藤光太郎),Python)
* HOG特徴量の抽出(MATLAB(伊藤光太郎))
* Dlibを用いた顔の部位検出(Python)
* 顔の部位から各部位の長さ等の特徴抽出(Python)
* 単純マッチング(MATLAB(伊藤光太郎),Python)
* k最近傍法(MATLAB(伊藤広樹、伊藤光太郎,)
* 部分空間法(MATLAB(林田和磨、伊藤光太郎))
* CNN(Python)
* LightGBM(Python)
* GUI作成(Python)
* データの分析(Python)
* スライドの作成(平尾礼央)
* 中間発表(伊藤広樹)
* ポスターの作成(平尾礼央)
* 最終発表(林田和磨)
* レポート作成(伊藤光太郎、)

1. **作成したプログラムの機能の説明**

まず、顔認識のシステムの全体像を図1に示す。

C:\Users\u6173064\AppData\Local\Microsoft\Windows\INetCache\Content.Word\プロジェクト実験中間発表 最終版-converted.emf

図1. 顔認識システムの全体像

ここでは図1の各手順について、採用した手法およびそれぞれの機能がどのプログラムに書かれているか、またそのプログラムについて説明する。なお、プログラムのコードは量が多いため、このレポートの末尾に添付した。

**3.1　入力**

与えられた画像を入力する手順。また、今回MATLABで作成したプログラムでは使用する特徴量及び識別アルゴリズムを選択する手順もここに分類する。

・SAISYUU.m … 3行目から26行目でパスを指定して、MATLABに画像を読み込む。また、27行目から34行目でどの特徴量と識別アルゴリズムの組み合わせで顔認識するかを選択する。

**3.2　前処理部**

**・顔検出**

与えられたデータセットの画像は人物のみでなく、背景も写ってしまっているため、識別する人物の顔だけを抽出する必要がある。

OpenCVによる顔の切り出し

正規化

なお、後述する「3.2 特徴抽出部」中の①、②の特徴量を使用する場合にはCanny法と呼ばれるエッジ検出をおこなった。なお、Canny法はエッジ検出の一種で、ガウシアンフィルタを使って平滑化の処理をした後に微分をする。その微分画像の勾配からエッジを検出する方法である。

実際のプログラム：

・SAISYUU.m … 13行目、24行目においてCanny法によるエッジ検出を行った。

**3.3　特徴抽出部**

今回の実験では3つの特徴量を画像データから抽出した。

1. 二次元離散コサイン変換(DCT)

画像を余弦波の周波数と係数成分へと変換する方法。画像のエネルギーは低域成分に集中しているため、変換後の周波数成分から低域の部分のみを特徴量として使用する。

実際のプログラム：

・funcDCT.m … 画像を引数として渡す関数。3行目でDCT変換をし、4行目で低周波成分を抜き出したものを特徴量として使用。

1. HOG(Histograms of Oriented Gradients)

局所領域 (セル) の画素値の勾配方向ヒストグラムを特徴量としたもの。

各ピクセルの輝度から勾配を求めた後、セル領域ごとにヒストグラムを求める。それをブロックごとに正規化し、特徴量を抽出する。今回の実験ではセルのサイズを8×8のもの(HOG8)と16×16のもの(HOG16)の二種類を使用した。

実際のプログラム：

・funcHOG.m … 画像を引数として渡す関数。3行目でHOG特徴量を求めて特徴量として使用。

1. Dilbを使った顔パーツの部位情報

なお、MATLABではPythonで書き出されたcsvファイルを読み込んで使用した。

実際のプログラム：

・SAISYUU.m …50、51行目、82行目、144行目など各識別アルゴリズム内で顔パーツの部位情報を使用する際にPythonで作成されたcsvファイルを読み込んで使用。

**3.4　識別部**

今回の実験では5つの識別アルゴリズムを用いてクエリがどの人物の画像かを識別した。

1. 単純マッチング

特徴量や画像の輝度値をピクセルごとに比較し、誤差が最も低いデータが属するクラス(今回は人物)を解答する方法。

実際のプログラム：

・SAISYUU.m … 35行目から71行目に相当。37行目から52行目の部分で入力部で指定した、単純マッチングに使用する特徴量や画像をそれぞれ準備する。そして54行目から70行目で、クエリに対する特徴量を求め、単純マッチングの関数を呼び出し、回答を受け取る。

・matching.m … 5行目から13行目で画像の輝度値や指定された特徴量を用いて、与えられたクエリと、200個のデータベースそれぞれに対しての距離計算をしている。そして15行目でクエリと最も距離が近くなったデータベースの画像を選び、23行目でその画像はどのクラス(人物)だったのかを判別する。

1. k最近傍法(kNN)

特徴量でデータをプロットし、クエリの特徴量と近い場所にあるk個のデータの中で多数決をしてどのクラスに属するか解答する方法。

実際のプログラム：

・SAISYUU.m … 72行目から110行目が対応。75行目から78行目で特徴量を用いてkNNモデルを作成し、80行目から97行目で作成したkNNモデルを使用して多数決の結果と、その内訳を出力し、多数決の結果を回答として使用。リジェクト機能を使用する場合は98行目から110行目で多数決の内訳をチェックし、それが条件を満たさない場合はリジェクトする(リジェクトの条件の詳細については後述)。

・labeling.m　… kNNモデルを作成するときに特徴量とラベルを引数として渡す必要があるため、その二つを作成するための関数。4行目から7行目と10行目から20行目で特徴量を計算し、25行目でラベルを作成する。

1. 部分空間法

特徴量を使ってベクトルを作り、クエリの特徴量を用いて作ったベクトルと最も類似しているクラスを解答する方法。

　実際のプログラム：

・SAISYUU.m … 111行目から201行目が対応。まず、113行目から126行目と143行目から146行目でデータベースの特徴量を取得し、128行目から140行目と148行目から154行目でクエリの特徴量を取得する。次に161行目から166行目でデータベースの特徴量から1人当たり10本のベクトルを作成し、167行目から174行目でその10本のベクトルの平均をとってその人物の代表ベクトルを決定する。これを20人分繰り返し、最後に176行目から186行目でその代表ベクトルとクエリの特徴量から作ったベクトルと、20人分の代表ベクトルを比較し、最も近似する代表ベクトルを持つ人物を回答する。リジェクト機能を使用する場合は188行目から201行目で類似度(２つのベクトル間の角度)が閾値よりも大きいかどうか判定し、類似度が小さい(角度が大きい)場合はリジェクトを行う。

1. 畳み込みニューラルネットワーク
2. LightGBM

**3.5　出力**

今回の実験では正答率を求める必要があるため、出力した結果が正しいかどうかを判定するプログラムを追加した。

・SAISYUU.m …　202行目から226行目が対応。204行目から211行目で正解ラベルを作成し、213行目から223行目で、各手法によって回答された答えと正解ラベルを比較して正答率とリジェクト機能によってはじかれなかった個数を算出し、225行目および226行目で表示した。

1. **実験結果**

各手法での正解率は表2のようになった。

表２. 各識別手法と特徴量の組み合わせによる正解率[%]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ピクセルマッチング | 単純マッチング | k最近傍法 | 部分空間法 | CNN | LightGBM |
| 顔の部位 |  |  |  |  |  |  |
| CannyありDCT15 |  |  |  |  |  |  |
| CannyなしDCT15 |  |  |  |  |  |  |
| HOG8 |  |  |  |  |  |  |
| HOG16 |  |  |  |  |  |  |

また、各手法での計測時間は表3のようになった。

表3. 各識別手法と特徴量の組み合わせによる正解率[秒]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ピクセルマッチング | 単純マッチング | k最近傍法 | 部分空間法 | CNN | LightGBM |
| 顔の部位 |  |  |  |  |  |  |
| CannyありDCT15 |  |  |  |  |  |  |
| CannyなしDCT15 |  |  |  |  |  |  |
| HOG8 |  |  |  |  |  |  |
| HOG16 |  |  |  |  |  |  |

1. **考察**

・リジェクト機能

MATLABで実装された各認識アルゴリズムにおいて、正答率が高かった特徴量を用いたものに対して「データベースに登録されていない人物をはじく機能(リジェクト機能)」を実装した。リジェクトの方法は「距離や角度に対して閾値を用いてその閾値を上回った場合リジェクトする」方法(単純マッチング、部分空間法)と「距離が近かった５つが同一でない場合にリジェクトする」方法をとった。その時の精度を表4に、再現率を表５に示す。なお、ここでの精度とは「識別器がリジェクトしなかったものに対して正しく分類できた割合」、再現率とは「データベースに登録されているクエリ(56枚)に対して正しく分類した割合」とする。また、今回の実験ではシステムが「顔認証」であるため、精度ができるだけ100%に近づけるようにし、その条件の下で最大となる再現率を表5に示した。

表4. リジェクト機能を使用した際の精度[%]

|  |  |  |  |
| --- | --- | --- | --- |
|  | 単純マッチング | k最近傍法 | 部分空間法 |
| CannyありDCT15 | 100 | 87.88 | 100 |
| HOG16 | 100 | 96.43 | 100 |

表５. リジェクト機能を使用した際の再現率[%]

|  |  |  |  |
| --- | --- | --- | --- |
|  | 単純マッチング | k最近傍法 | 部分空間法 |
| CannyありDCT15 | 46.43 | 57.14 | 25.00 |
| HOG16 | 48.21 | 50 | 37.93 |

参考のため、表6にリジェクト機能を使用していないときの再現率を示す。なお、精度に関しては4.結果の表３と同様の結果(リジェクト機能がない場合は正答率をそのまま精度として用いるため)になるので省略する。

表6. リジェクト機能を使用した際の再現率[%]

|  |  |  |  |
| --- | --- | --- | --- |
|  | 単純マッチング | k最近傍法 | 部分空間法 |
| CannyありDCT15 | 85.71 | 71.43 | 69.64 |
| HOG16 | 78.57 | 78.57 | 66.07 |

　表２と表４、表5と表6を比較すると、リジェクト機能を搭載することによって精度は向上しているが、その代わり再現率が下がるというトレードオフの関係が成り立っていることがわかる。また、表2の結果では単純マッチングおよび部分空間法ではDCTのほうが正答率が高く、良い特徴量であると考えられていた。しかし、表4、表５をみるとその２つの識別アルゴリズムではともに精度は100%であっても、HOGのほうが再現率は高くなっているため、リジェクト機能を使用する際にはHOG特徴量のほうが有用であると考えられる。

　また、k最近傍法では、精度を100%にすることができなかったため、リジェクト機能を搭載する場合に用いる識別アルゴリズムは単純マッチングか部分空間法が適しているのではないかと考える。

1. **作成したプログラム**

**6.1 MATLAB作成分のプログラム**

**・SAISYUU.m**

clc

clear

%% DBの実装

c = 20; % クラス総数

n = 10; % 1クラス当たりの学習パターン数

% データベースの読み込み

path = 'M:\project\dataset2\DB\jpeg\';

for i=1:c

for j=1:n

str = strcat(path, num2str(n\*(i-1)+j-1, '%03d'), '.jpg');

img = imread(str);

img = edge(img, 'Canny',[],2);

DB(:,:,n\*(i-1)+j) = img;

end

end

% クエリの読み込み

Qpath = 'M:\project\dataset2\Query\jpeg\';

D = dir('M:\project\dataset2\Query\jpeg\\*.jpg');

for i=1:length(D)

name = strcat(Qpath, D(i).name);

img = imread(name);

img = edge(img, 'Canny', [], 2);

Query(:,:,i) = img;

end

%% 使用する機能選択

fprintf('どのアルゴリズムを使用しますか?\n')

prompt = '0=単純マッチング(輝度値),1=単純マッチング(DCT),2=単純マッチング(HOG),3=単純マッチング(Face\_Parts),4=KNN(DCT),5=KNN(HOG),6=KNN(Face\_Parts),7=部分空間(DCT),8=部分空間(HOG),9=部分空間(Face\_Parts)\n';

hanbetu = input(prompt);

fprintf('リジェクト機能を使用しますか?\n')

prompt = '1=Yes,0=No\n';

RJ = input(prompt);

answer = zeros(1, 58);

%% 単純マッチング

if (hanbetu >= 0 && hanbetu <= 3)

%特徴量のDB作成

if (hanbetu >= 1 && hanbetu <= 2)

for i=1:200

img=DB(:,:,i);

if hanbetu == 1

m\_feature = funcDCT(img);

else

m\_feature = funcHOG(img);

end

m\_DB(i,:) = m\_feature;

end

end

if hanbetu == 3

m\_DB = csvread('M:\project\dataset4\DB\csv\FP.csv');

Query\_feature = csvread('M:\project\dataset4\Query\csv\QFP.csv');

end

%単純マッチングによる回答

for i=1:length(D)

if hanbetu == 0

Q\_feature = double(Query(:,:,i));

elseif hanbetu == 1

Q\_feature = funcDCT(Query(:,:,i)) ;

elseif hanbetu == 2

Q\_feature = funcHOG(Query(:,:,i)) ;

elseif hanbetu == 3

Q\_feature = Query\_feature(i,:) ;

end

if hanbetu == 0

answer(i) = matching(DB, Q\_feature, hanbetu, RJ);

else

answer(i) = matching(m\_DB, Q\_feature, hanbetu, RJ);

end

end

end

%% KNN

if(hanbetu >= 4 && hanbetu <=6)

%knnモデルの作成

[fDB, C] = labeling(DB, hanbetu); %特徴量のラベリング

model = fitcknn(fDB, C);

model.NumNeighbors = 5;

%knnモデルによる回答

if hanbetu == 6

Query\_filename = 'M:\project\dataset4\Query\csv\QFP.csv';

Query\_feature = csvread(Query\_filename);

end

for i=1:length(D)

testImg = Query(:,:,i); %クエリの取得

if hanbetu == 4

dctF = funcDCT(testImg);

[answer(i), score(i,:)] = predict(model, dctF);

elseif hanbetu == 5

hogF = funcHOG(testImg);

[answer(i), score(i,:)] = predict(model, hogF);

elseif hanbetu == 6

faceF = Query\_feature(i,:);

[answer(i), score(i,:)] = predict(model, faceF);

end

end

%リジェクト領域

if RJ == 1

score2 = score.';

[MA,I] = max(score2);

for i=1:58

if MA(i) > 0.9

answer(i) = I(i) - 1;

else

answer(i) = 999;

end

end

end

end

%% 部分空間(DCTを特徴量とする場合)

if(hanbetu >= 7 && hanbetu <=8)

if hanbetu == 7

feature\_num = 225;

else

feature\_num = 2916;

end

%DB部分

for i=1:200

img = DB(:,:,i);

if hanbetu == 7

db\_feature\_list(i,:) = funcDCT(img);

else

db\_feature\_list(i,:) = funcHOG(img);

end

end

%クエリ部分

for i=1:58

img = Query(:,:,i);

if hanbetu == 7

Query\_feature(i,:) = funcDCT(img);

else

Query\_feature(i,:) = funcHOG(img);

end

end

for i=1:58

Query\_feature(i, feature\_num+1) = sqrt(sumsqr(Query\_feature(i,:)));

end

end

%% 部分空間(顔のパーツを特徴量として扱う場合)

if hanbetu == 9

%DB部分

DB\_filename = 'M:\project\dataset4\DB\csv\FP.csv';

db\_feature\_list = csvread(DB\_filename);

feature\_num = 20;

%クエリ部分%

Query\_filename = 'M:\project\dataset4\Query\csv\QFP.csv';

Query\_feature = csvread(Query\_filename);

%クエリのサイズ計算

for i=1:58

Query\_feature(i, feature\_num+1) = sqrt(sumsqr(Query\_feature(i,:)));

end

end

%% 部分空間回答部

if(hanbetu >= 7 && hanbetu <= 9)

%平均値計算

sub\_space = zeros(20, feature\_num+1);

test = zeros(200, feature\_num);

for i=1:20 %i=人数

for j=1:10 % i人目について10枚ずつ処理

for k=1:feature\_num %特徴量が20個

test(j,k) = db\_feature\_list((i-1)\*10+j, k);

end

end

%平均値をとる

M = mean(test);

for k=1:feature\_num

sub\_space(i,k) = M(k);

end

zettai = sqrt(sumsqr(sub\_space(i,:)));

sub\_space(i, feature\_num+1) = zettai;

end

%部分空間法による回答

hairetu = zeros(58,20);

for j=1:58

for i=1:20

for k=1:feature\_num

hairetu(j,i) = hairetu(j,i) + Query\_feature(j,k) \* sub\_space(i,k);

end

hairetu(j,i) = acos(hairetu(j,i) / (Query\_feature(j,feature\_num+1) \* sub\_space(i,feature\_num+1)) );

end

end

[S, answer] = min(hairetu, [], 2);

%リジェクト領域

for i=1:58

if hanbetu == 8

sikiiti = 0.95;

elseif hanbetu == 7

sikiiti = 0.47;

end

if(RJ == 1 && S(i) > sikiiti)

answer(i) = 999;

else

answer(i) = answer(i) - 1;

end

end

end

%% Answer check

% Qラベルの作成

A = zeros(1, 8);

Qlabels = A;

nums = [3, 3, 4, 3, 3, 4, 2, 3, 3, 3, 3, 3, 3, 2, 1, 1, 1, 1, 2, 2];

for i=1:20

A = ones(1, nums(i) ) \* i;

Qlabels = horzcat(Qlabels, A);

end

% 正解率の判定

correctNum = 0;

PassNum = 0;

for i=1:length(D)

if(answer(i) > 19 && RJ == 1)

PassNum = PassNum + 1;

end

if(answer(i) == Qlabels(i))

correctNum = correctNum + 1;

end

end

sprintf('正解率: %.4f', correctNum / ( length(D) ) )

sprintf('精度 %.4f',correctNum / ( PassNum ))

**・matching.m**

function number = matching(DB, Query, hanbetu, RJ)

%単純マッチング関数(関数Mファイル)

%クエリ画像XとDBの画像をピクセル毎に比較し、二乗誤差が最も小さい人物を出力する

for i=1:200

if hanbetu == 0

A = double(DB(:,:,i));

elseif (hanbetu >= 1 && hanbetu <= 3)

A = DB(i,:);

end

D = (Query - A).^2;

distance(i) = sum(sum(D));

end

[minimum, index] = min(distance);

%リジェクト領域

if RJ == 1

if hanbetu == 1

sikiiti = 140;

elseif hanbetu == 2

sikiiti = 57;

end

if minimum > sikiiti

index = 10000;

end

end

number = ceil(index / 10) - 1;

end

**・funcDCT.m**

function dctFeature = funcDCT(img)

% DCT特徴量

img4 = dct2(double(img)); %2次元DCT

imgdctlow = img4(1:15, 1:15); %低周波成分の抜き出し

dctFeature = reshape(imgdctlow, [1, 15\*15]); %1次元化

end

**・funcHOG.m**

function hogFeature = funcHOG(img)

% DCT特徴量

hogFeature = extractHOGFeatures(img, 'CellSize', [16 16]);

end

**・labeling.m**

function [fDB,C] = labeling(DB,hanbetu)

%特徴量のDB作成関数

if hanbetu == 6

DB\_filename = 'M:\project\dataset4\DB\csv\FP.csv';

feature\_list = csvread(DB\_filename);

end

for i=1:200

%DBのi枚目を読み込む

img = DB(:,:,i);

%i枚目の特徴量計算

if hanbetu == 4

feature = funcDCT(img); %dctの場合

elseif hanbetu == 5

feature = funcHOG(img);

elseif hanbetu == 6

feature = feature\_list(i,:);

end

%特徴量のDB

fDB(:,i) = feature;

%正解ラベル

C(i) = fix( (i-1) / 10);

end

C = transpose(C);

fDB = transpose(fDB);

end

**6.2 Python作成分のプログラム**