Project #1 – 最近鄰居人臉辨識

1.方法描述

A.說明

a.原理

SAD (sum of absolute distance): 計算兩張圖之間數值的絕對值差總和，數值越小代表兩張圖越相近。

SSD (sum of square distance): 計算兩張圖之間數值的平方差總和，數值越小代表兩張圖越相近。

b.作法

1)讀取放置圖片的test與training資料夾，**注意程式本身和圖片資料夾必須放在同一個目錄下**

2)設置正確計算圖片的counter等會用到的參數

3)開始雙重迴圈

4)對於每張test圖片，遍歷training資料夾下的每張圖片，分別計算SAD與SSD，並記錄下最相近的圖片名稱

5)由於某些test圖片名稱少了一個m在前頭，所以會幫他補一個m，方便圖片名稱比較

6)比較記錄下的最相近圖片名稱，是否跟test圖片屬於同一個人，是的話正確判斷的counter數+1

7)結束雙重迴圈，計算SAD與SSD的正確率並印出

c.執行方式

將程式碼與放置圖片的test與training資料夾放在相同目錄底下，打開matlab，執行程式碼即可，執行結束會在Command Window顯示SAD與SSD的正確率

B.程式碼

```
%read data
test_path = './dataset/test/';
test_data_dir = dir(test_path);
train_path = './dataset/training/';
train_data_dir = dir(train_path);

%count for correctness
sad_correct = 0;
ssd_correct = 0;

%redundant length count
test_red = 0;

%loading bar
h = waitbar(0, 'please wait');

for i = 1:length(test_data_dir) %read test img
```

```
    test_name = test_data_dir(i).name; %fprintf('%s\n', test_name);
    if(isequal(test_name, '.') || isequal(test_name, '..') || isequal(test_name,
'.DS_Store'))
        test_red = test_red + 1;
        continue;
    end
    test = imread(strcat(test_path, test_name));
    test = im2double(test);
    %[m, n] = size(test); %get img size
    %new_test = reshape(test, 1, m*n); %reshape img size

    %initial value
    sad_min = realmax('single');
    sad_name = '';
    ssd_min = realmax('single');
    ssd_name = '';

    for j = 1:length(train_data_dir) %read train img

        train_name = train_data_dir(j).name; %fprintf('%s\n', train_name);
        if(isequal(train_name, '.') || isequal(train_name, '..') || isequal(test_name,
'.DS_Store'))
            continue;
        end
        train = imread(strcat(train_path, train_name));
        train = im2double(train);
        %[m, n] = size(train); %get img size
        %new_train = reshape(train, 1, m*n); %reshape img size

        %compute sad value
        sad = abs(test - train);
        sad_sum = sum(sad, 'all');
        if(sad_min > sad_sum)
            sad_min = sad_sum;
            sad_name = train_name;
        end

        %compute ssd value
        ssd = abs((test - train) .* (test - train));
        ssd_sum = sum(ssd, 'all');
        if(ssd_min > ssd_sum)
            ssd_min = ssd_sum;
            ssd_name = train_name;
        end

    end

    %strcat
```

```
    if(strncmp(test_name, 'm-00', 4))
        strcat('m', test_name);
    end

    %fprintf('test_name: %s\n', test_name);
    %fprintf('sad_name: %s\n', sad_name);
    %fprintf('ssd_name: %s\n', ssd_name);

    %update sad correct value
    if(strncmp(test_name, sad_name, 6))
        sad_correct = sad_correct + 1;
    end

    %update ssd correct value
    if(strncmp(test_name, ssd_name, 6))
        ssd_correct = ssd_correct + 1;
    end

    str=['processing...', num2str(i / length(test_data_dir) * 100), '%'];
    waitbar(i / length(test_data_dir), h, str)

end

%dalete bar
delete(h);

%compute sad result
sad_result = sad_correct / (length(test_data_dir) - test_red);
fprintf('sad = %3f\n', sad_result);

%compute ssd result
ssd_result = ssd_correct / (length(test_data_dir) - test_red);
fprintf('ssd = %3f\n', ssd_result);
```

2.實驗結果

A.每個階段的圖片

計算SAD與SSD之前會先對圖片做im2double()

原圖

im2double() 後的圖

B.執行結果

```
Command Window
  sad = 0.603730
  ssd = 0.541570
fx >>
```

3.結果討論

SAD 與 SSD 的實驗結果都大概落在 60% 左右，這兩種算法其實都很容易受到雜訊的影響。
就實驗數據集與訓練數據集來說，同一個人的圖片都有著不同的表情，就連臉都不一定全部露出來。有些會有布料阻擋，這些其實都是雜訊，會影響算法對相同人臉的判斷。尤其當雜訊在圖片上佔了大部分的比例時，算法會傾向被雜訊影響，自然無法判斷出相同人臉。

要準確判斷人像的輪廓訊息，我認為會需要使用機器學習做工具效果會更好，例如專門處理影像的CNN。

4.問題討論

這兩個演算法實作並不困難，重點是讀入影像之後，會需要先將數據做 im2double()，將數值壓縮到0和1之間，否則在計算的時候會有數值 overflow 無法表示的問題，也就是數值超過255，間接導致準確度大幅跌落至 10% 不到。