UFPB - Universidade Federal da Paraíba

Centro de Informática

SINAIS E SISTEMAS DINÂMICOS

Daniel de Sá Pires

Julio Leite Tavares Neto

Lucas Freitas de Barros

João Pessoa, 06 de abril de 2020

UFPB - Universidade Federal da Paraíba

Centro de Informática

Projeto realizado e concretizado sob orientação do professor Derzu Omaia como requisito avaliativo para a disciplina de Sinais e Sistemas Dinâmicos.

João Pessoa, 06 de abril de 2020

SUMÁRIO

1. INTRODUÇÃO

1.1 Fundamentação Teórica

Para realizar a identificação facial através de imagens foi calculado a distância euclidiana sobre uma sub-região retangular. Distância Euclidiana é a distância entre dois pontos que pode ser provada pela aplicação repetida do teorema de Pitágoras.

Fórmula:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \qquad \mathrm{DE} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Figura 1. Fórmula para calcular a distância Euclidiana

Também para realizar os cálculos utilizamos a transformada de Fourier:

A transformada de Fourier permite analisar de forma adequada funções não periódicas. A transformada de Fourier compete em algumas aplicações com a transformada de Laplace. Entretanto, a transformada de Fourier é mais útil que a transformada de Laplace em algumas aplicações relacionados com problemas de comunicações e processamento de sinais.

A forma exponencial da série de Fourier:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n \; e^{jn\omega_o t}$$

onde

$$c_n = \frac{1}{T}\int_0^T f(t) \; e^{-jn\omega_o t}\, dt$$

Figura 2. Fórmula para calcular a transformada de Fourier

1.2 O trabalho

Sistemas de reconhecimento facial vem a anos sendo criados e aprimorados, usando novas tecnologias e se tornando cada vez mais precisos e confiáveis, esses sistemas são sistemas biométricos que se utilizam de características do indivíduo que podem ser medidos e comparados a características equivalente de outros.

No corrente trabalho, conceitos definidos em sala de aula serão aplicados na prática com o objetivo de reconhecer 40 pessoas diferentes de um banco de imagens previamente definido. Desta forma, conceitos como transformada de Fourier Bidimensional e distância euclidiana são imprescindíveis para o desenvolvimento do mesmo.

2. DESENVOLVIMENTO

2.1 Ferramentas

As ferramentas utilizadas no projeto foram: Python, OpenCV e Numpy.

O uso de Python pode ser explicado pelas características de uma linguagem marcada por: Simplicidade, Interoperabilidade, Multiplataforma e Robustez. Além dessas vantagens foi utilizada uma extensão de Python, a NumPy. Ela é uma poderosa biblioteca Python que é usada principalmente para realizar cálculos em Arrays Multidimensionais. Como a representação de imagens em um Computador se dá através de Arrays Multidimensionais, o NumPy torna-se a escolha mais natural. Essa poderosa biblioteca fornece algumas excelentes funções(espelhamento,rotação e etc.) para rápida manipulação de imagens.

A parte de Visão Computacional tenta simular a visão natural com embasamento científico, e a OpenCV é uma biblioteca de programação de código aberto que atende bem aos requisitos da Visão Computacional. Atualmente possui mais de 500 funções que podem ser utilizadas em diversas linguagens de programação (C++, Python, Ruby, Java…) e é usada para diversos tipos de análise em imagens e vídeos, como detecção, tracking e reconhecimento facial, edição de fotos e vídeos, detecção e análise de textos, etc.

2.2 A lógica

O desenvolvimento do programa descrito segue os três seguintes passos:

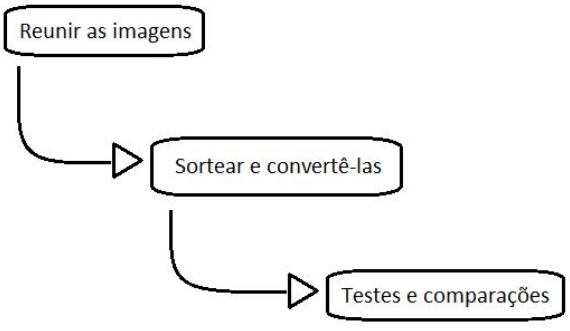

Figura 3. Passos seguidos no algoritmo

Essas são as saídas expressas pelo programa ao fim da execução, além do resultado de acertos e erros. Cada etapa supracitada possui um conjunto de instruções que serão descritas abaixo:

1. Reunir imagens:

   Inicializa cada um dos arrays declarados no escopo global do algoritmo, porém, não somente isso como também reúne todos os caminhos das imagens subjacentes através do comando glob.iglob, que recebe uma string como parâmetro e retorna um array com todos os objetos contido no diretório e especificado no caminho.

2. Sortear e convertê-las

   Nesta etapa, uma imagem de cada pessoa foi sorteada, e todas sofreram a Transformada de Fourier Bidimensional.

   Para definir as imagens aleatórias um comando da biblioteca Random foi utilizado: randint, que recebe como parâmetro um intervalo de números que serão sorteados.

   Para carregar as imagens, foi utilizado um comando da biblioteca da opencv, cv2, a função imread recebe como parâmetro o caminho da imagem e o parâmetro que define que será aberta em escala de cinzas.

   Após isso, a transformada de Fourier é aplicada nas linhas subjacentes através dos comandos numpy.fft.fft2 e numpy.fft.fftshift.

   Com as imagens em mãos, a imagem aleatória convertida é armazenada na lista “imagensSorteadas” e seus respectivos caminhos são salvos em “urlsImgsEscolhidas”.

   Da mesma forma como as imagens sorteadas são convertidas e guardadas, as demais imagens também são convertidas e guardadas em suas respectivas listas: “imagensDeTeste” (para armazenar as imagens) e “urlsImgsDeTeste” (para armazenar os caminhos dessas imagens).

3. Testes e comparações

   Foram realizados os testes definidos na descrição do projeto. A fim de esclarecer melhor há um tópico dedicado a explicar o algoritmo dos testes. Deve-se atentar, por hora, à descrição teórica de cada teste implementado e executado.

   - **Teste 1**

   A distância euclidiana é calculada tendo por base apenas a parte real das áreas, desta forma, extraída a parte real de cada imagem sorteada, sua distância em relação a parte real das demais imagens são calculadas, seguindo o seguinte:

$$DT(T0, C0) = D(R(T0),R(C0))$$

onde,
T0 equivale à imagem de teste da pessoa
C0 equivale à imagem de comparação
R equivale à parte real

- **Teste 2**

Idem ao Teste 1, neste caso a distância euclidiana é calculada tendo por base apenas as partes imaginárias das áreas em questão. Desta forma, as distâncias entre as partes imaginárinárias da imagem sorteada e das demais imagens são calculadas, no qual:

$$DT(T0, C0) = D(I(T0),I(C0))$$

onde,
T0 equivale à imagem de teste da pessoa
C0 equivale à imagem de comparação
I equivale à parte imaginária

- **Teste 3**

Esta é, basicamente, a soma entre o que é feito no Teste 1 e no Teste 2, logo é feito o cálculo da distância entre as partes reais das imagens envolvidas e, somado a esta, é o resultado do cálculo da distância entre as partes imaginárias envolvidas.

$$DT(T0, C0) = D(R(T0),R(C0)) + D(I(T0),I(C0))$$

onde,
T0 equivale à imagem de teste da pessoa
C0 equivale à imagem de comparação
R equivale à parte real
I equivale à parte imaginária

- **Teste 4**

Neste teste, basicamente, é calculada a menor distância entre quatro possibilidades: entre as partes reais da imagem escolhida e da imagem de comparação; entre as partes imaginárias da imagem escolhida e da imagem de comparação; entre a parte real da imagem escolhida e da parte imaginária da imagem de comparação; entre a parte imaginária da imagem escolhida e da parte real da imagem de comparação.

$$DT(T0, C0) = MENOR(D(R(T0),R(C0)) ;D(R(T0),I(C0)) ; D(I(T0),I(C0)) ; D(I(T0),R(C0)))$$

onde,
T0 equivale à imagem de teste da pessoa

C0 equivale à imagem de comparação
R equivale à parte real
I equivale à parte imaginária

- **Teste 5**

No teste presente é feita a combinação entre os módulos da parte real e imaginárias de cada imagem em comparação.

$$DT(T0, C0) = D(merge(R(T0),I(T0)),merge(R(C0),I(C0)))$$

onde,
T0 equivale à imagem de teste da pessoa
C0 equivale à imagem de comparação
R equivale à parte real
I equivale à parte imaginária

Ao final de todos os testes, porém, é imprescindível o cálculo dos erros e acertos que possuem em cada melhor resultado de cada teste.

2.3 Os testes

Antes de iniciar a execução dos testes propriamente ditos, é necessário calcular a área e a parte real e imaginária da imagem da pessoa sorteada. Após isso, uma função é chamada, se trata da *realiza_teste*, que recebe como parâmetro, respectivamente, o índice do teste que está sendo realizado (seguindo a ordem descrita no tópico 2.2), a parte real, a parte imaginária da imagem sorteada da pessoa corrente e o índice da pessoa.

A função, de forma sucinta, se responsabiliza por realizar o teste que for adequado, conforme o que for enviado por parâmetro. Desta forma, o primeiro parâmetro (índice do teste) define qual dos testes será realizado. Porém, uma sacada interessante vale ser mencionada: o código possui vários trechos que se repetem diversas vezes, portanto, ao invés de reescrever cada teste como algo independente dentro da função, tornando todo o código mais robusto e menos ágil, apenas as partes que diferem um teste de outro foram rearranjadas, incluídas em condições diferentes, que são atendidas conforme o teste que se deseja realizar no momento da execução.

Desta forma, parte do código é realizada independentemente do teste que se quer realizar, e somente aquilo que difere um teste de outro é incluído nas condições da função.

A parte que se repete para todos os testes se resume em percorrer todas as imagens através de dois loops de *for*, recuperar a imagem de teste (a que será comparada com a imagem sorteada de cada pessoa) e a extração da área da mesma.

Para a execução da função, outras funções são frequentemente utilizadas: *modulo* e *distancia_euclidiana*. A função de módulo recebe a quantidade de linhas e colunas, além da matriz do qual se quer calcular o módulo. A *distancia_euclidiana* é a função responsável por calcular a distância entre duas matrizes.

Para mais detalhes a respeito do código, o mesmo está devidamente documentado e disponível no github.

3. RESULTADOS E CONCLUSÕES

Para cada execução pode-se observar uma quantidade diferente de acertos e erros, porém isso se decorre da aleatoriedade da escolha das imagens de cada pessoa, o que resulta em taxas diferentes de erros e acertos.

A tabela seguinte é resultado de um dos testes, o que mostra um grande equilíbrio na quantidade de acertos, que chega a denotar um sistema quase perfeito de identificação de pessoas, por exceção ao Teste 5, onde observa-se uma taxa maior de erros.

	ACERTOS	ERROS	% ERROS
Teste 1	38	2	5%
Teste 2	38	2	5%
Teste 3	39	1	2,5%
Teste 4	39	1	2,5%
Teste 5	6	34	85%

Tabela 1. Taxa de erro de execução dos testes

4. CÓDIGO

```
"""
UNIVERSIDADE FEDERAL DA PARAÍBA
Engenheria de Computação
Sinais e Sistemas Dinâmicos

ALUNOS:
    Daniel de Sá Pires
    Júlio Leite Tavares Neto
    Lucas Freitas de Barros
"""



import cv2
import numpy
import glob
import random
import math

#Inicializo todas as listas criadas
def inicializando():
    for i in range(40):
        imagensOrlFaces.append([])
        imagensSorteadas.append([])
        imagensDeTeste.append([])
        urlsImgsEscolhidas.append([])
        urlsImgsDeTeste.append([])
        url = "orl_faces\orl_faces\s"+str(i+1)+"\*"
        imagensOrlFaces[i] = glob.iglob(url)

#Calcula o módulo
def modulo(linhas,colunas,matriz):
    modulo = 0
    for i in range(linhas):
        for j in range(colunas):
            modulo += (int(matriz[i][j]))**2
            if(modulo < 0):
                modulo *= -1
    return math.sqrt(modulo)
```

```
#Calcula a distância euclidiana entre duas matrizes
def distancia_euclidiana(matriz1, matriz2):
    retorno = 0
    for i in range(len(matriz1)):
        for j in range(len(matriz1[0])):
            retorno += (int(matriz2[i][j])- int(matriz1[i][j]))**2
            if (retorno < 0):
                retorno *= -1
    return math.sqrt(retorno)

#Realiza os 5 testes
def
realiza_teste(indiceDoTeste,parteReal,parteImaginaria,indiceI):
    #Inicializa variáveis que serão úteis no decorrer dos testes
    menorDistancia = None
    menorDistanciaAux = None
    distancia = None

    #Caso seja solicitada a execução do 5º teste, uma sequência
    #de passos deve ser seguida antes da execução do teste
    #propriamente dito. Isso consiste em determinar os módulos
    #real e imaginário da imagem sorteada corrente. Para isso
    #é feito o módulo das partes reais da imagem corrente, e
    #o módulo das partes imaginárias da imagem corrente. Lembrando
    #que o módulo se dá pela raiz quadrada da soma dos quadrados
    if(indiceDoTeste == 5):
        moduloTesteReal = 0
        moduloTesteImaginario = 0

        moduloArea = 0
        moduloTeste = 0

        #Cálcula o módulo da parte real da imagem escolhida
        moduloReal =
modulo(len(parteReal),len(parteReal[0]),parteReal)

        #Calcula o módulo da parte imaginária da imagem escolhida
        moduloImaginario =
modulo(len(parteImaginaria),len(parteImaginaria[0]),parteImaginari
a)
```

```
    #Percorre todas as imagens de teste
    for a in range(QNT):
        for j in range(9):
            #Pega a imagem de teste atual
            magnitudeTeste = imagensDeTeste[a][j]

            #Extrai a posição do quadrado na imagem
            linhas,colunas = magnitudeTeste.shape
            cLinha,cColuna = linhas//2,colunas//2

            #Pega a área da image
            areaTeste = magnitudeTeste[cLinha-QUAD:cLinha+QUAD,
cColuna-QUAD:cColuna+QUAD]

            #Caso seja feito o Teste 1, a distância calculada será
entre a parte real
            #da imagem aleatória corrente e a parte real da imagem
atual
            if(indiceDoTeste == 1):
                parteRealAreaTeste = numpy.real(areaTeste)
                distancia =
distancia_euclidiana(parteReal,parteRealAreaTeste)

            #Caso seja feito o Teste 2, a distância calculada será
entre a parte imaginária
            #da imagem aleatória corrente e a parte imaginária da
imagem atual
            elif(indiceDoTeste == 2):
                parteImaginariaTeste = numpy.imag(areaTeste)
                distancia =
distancia_euclidiana(parteImaginaria,parteImaginariaTeste)

            #Caso seja feito o Teste 3, a distância calculada será
a soma da distância entre
            #as partes reais e imaginárias entre as imagens
aleatória e correntes
            elif(indiceDoTeste == 3):
                parteRealAreaTeste = numpy.real(areaTeste)
                parteImaginariaTeste = numpy.imag(areaTeste)

                distancia =
```

```
distancia_euclidiana(parteReal,parteRealAreaTeste)
                distancia +=
distancia_euclidiana(parteImaginaria,parteImaginariaTeste)

            #Caso seja feito o Teste 4, a distância será a menor
distância entre uma das
            #opções:
            #1 - distância entre as partes reais das imagens
aleatórias correntes e da
            #imagem atual
            #2 - distância entre a parte real da imagem aleatória
atual e a parte imaginária
            #da imagem de teste atual
            #3 - distância entre a parte imaginária da imagem
aleatória atual e a parte
            #real da imagem de teste atual
            #4 - distância entre a parte imaginária da imagem
aleatória atual e a parte
            #imaginária da imagem de teste atual
            elif(indiceDoTeste == 4):
                parteRealAreaTeste = numpy.real(areaTeste)
                parteImaginariaTeste = numpy.imag(areaTeste)
                distancia =
distancia_euclidiana(parteReal,parteRealAreaTeste)
                if(menorDistanciaAux == None):
                    menorDistanciaAux = distancia
                if(distancia < menorDistanciaAux):
                    menorDistanciaAux = distancia
                distancia =
distancia_euclidiana(parteReal,parteImaginariaTeste)
                if(distancia < menorDistanciaAux):
                    menorDistanciaAux = distancia
                distancia =
distancia_euclidiana(parteImaginaria,parteRealAreaTeste)
                if(distancia < menorDistanciaAux):
                    menorDistanciaAux = distancia
                distancia =
distancia_euclidiana(parteImaginaria,parteImaginariaTeste)
                if(distancia < menorDistanciaAux):
                    menorDistanciaAux = distancia
                distancia = menorDistanciaAux
```

```
            #Caso seja feito o Teste 5, a distância será dada pela
raiz quadrada do
            #quadrado dadiferença da soma dos módulos de cada
imagem envolvida.
            elif(indiceDoTeste == 5):
                parteRealAreaTeste = numpy.real(areaTeste)
                parteImaginariaTeste = numpy.imag(areaTeste)

                moduloTesteReal =
modulo(len(parteRealAreaTeste),len(parteRealAreaTeste[0]),parteRea
lAreaTeste)
                moduloTesteImaginario =
modulo(len(parteImaginariaTeste),len(parteImaginariaTeste[0]),part
eImaginariaTeste)

                distancia = math.sqrt(((moduloTesteReal +
moduloTesteImaginario)-(moduloReal + moduloImaginario))**2)


            #Compara a distância obtida com o teste atual com a
menor distância
            #já calculada e atualiza os valores dos índices.
            if(menorDistancia == None):
                menorDistancia = distancia
                indiceI, indiceA, indiceJ = indiceI,a,j
            if(distancia < menorDistancia):
                menorDistancia = distancia
                indiceI, indiceA, indiceJ = indiceI,a,j
    return menorDistancia,indiceI,indiceA,indiceJ


#Constantes
QNT = 40
QUAD = 4

#Aqui conterá todas as URLS já no formato glob para que possam ser
acessadas
#(note que as URLS são string que foram convertidas para um objeto
do qual é
#possível pegar todos os arquivos de um diretório)
```

```
imagensOrlFaces    = []

#Guarda as imagens aleatórias sorteadas convertidas
imagensSorteadas   = []

#As demais imagens convertidas para que possam ser comparadas com
a sorteada
imagensDeTeste     = []

#Guarda as URLS das imagens aleatórias sorteadas e convertidas
urlsImgsEscolhidas = []

#Guarda a URL das demais imagens convertidas que serão comparadas
urlsImgsDeTeste    = []


#Guarda a quantidade de erros e acertos que terão em cada um dos 5
testes
erros   = [0]*5
acertos = [0]*5


print("Passo 1 - Pegando as URLs")
#Pega todas as URLS e inicializ as listas
inicializando()

print("Passo 2 - Sorteando e convertendo imagens")
#Sorteia uma imagem, pega todas as imagens e transforma, compara
se ela é a aleatória
for i in range(QNT):
    #Pega o caminho de uma imagem aleatória de cada uma das 40
pastas
    imagemAleatoria =
"orl_faces\orl_faces\s"+str(i+1)+"\\"+str(random.randint(1,10))+".
pgm"

    #Percorre todas as imagens da pasta atual
    for a in imagensOrlFaces[i]:
        #Carrega a imagem
        imagem = cv2.imread(a,cv2.IMREAD_GRAYSCALE)
```

```
        #Faz a transformada nelas
        imagemTransformada = numpy.fft.fft2(imagem)
        fftshift = numpy.fft.fftshift(imagemTransformada)

        #Se o caminho da imagem atual for o mesmo que foi sorteado
em "imagemAleatoria"
        #guarda a imagem convertida e sua URL nas listas
"urlsImgsEscolhidas" e
        #imagensSorteadas, respectivamente. Caso contrário, guarda
sua url e a imagem
        #convertida em "urlsImgsDeTeste" e "imagensDeTeste"
respectivamente, prontas
        #para serem testadas.
        if (a == imagemAleatoria):
            urlsImgsEscolhidas[i] = imagemAleatoria
            imagensSorteadas[i] = fftshift
        else:
            urlsImgsDeTeste[i].append(a)
            imagensDeTeste[i].append(fftshift)


print("Passo 3 - Testando e comparando")
#Iniciam os testes
for i in range(QNT):
    #Pega a imagem sorteada da pasta atual
    magnitudeEscolhida = imagensSorteadas[i]

    #Recupera a posição do quadrado
    linhas, colunas = magnitudeEscolhida.shape
    cLinha, cColuna = linhas//2, colunas//2

    #Calcula a área e pega a parte real e imaginária da mesma
    area = magnitudeEscolhida[cLinha-QUAD:cLinha+QUAD,
cColuna-QUAD:cColuna+QUAD]
    parteReal = numpy.real(area)
    parteImaginaria = numpy.imag(area)

    #Inicia o loop para a execução de cada um dos 5 testes com a
imagem corrente
    for testes in range(5):
        menorDistancia,indiceI,indiceA,indiceJ =
```

```
realiza_teste(testes+1,parteReal,parteImaginaria,i)
        #Compara com os índices das imagens e calcula os acertos e
erros que
        #existiram na comparação.
        if(indiceI == indiceA):
            acertos[testes] += 1
        else:
            erros[testes] += 1

print("\n\nRESULTADO\n")
for i in range(len(acertos)):
    print("#-- TESTE %d --#" %(i+1))
    print("Acertos: %d" %(acertos[i]))
    print("Erros: %d\n" %(erros[i]))
```

5. REFERÊNCIAS

[1] https://pt.wikipedia.org/wiki/Dist%C3%A2ncia_euclidiana
[2] https://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/mcap05.pdf
[3] https://github.com/BarrosLucas/ProjetoSSD/blob/master/sinais.py
[4] https://www.dropbox.com/s/heifyl4kpnwpmzp/Projeto.pdf?dl=0
[5] https://www.dropbox.com/s/u16sm4mt6vk69zn/SSD10_TransformadaFourier.pdf?dl=0