INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Linguagens de Programação

Tema do Trabalho: reconhecimentodefaces

Elaborado por:

Felizmelo Pereira Borja Nº 16709

Docente: José Jasnau Cae

1 Introdução

Nos anos mais recentes um dos tópicos considerados mais importantes na área da informática tem sido o uso de técnicas de aprendizagem por máquinas1 associadas às redes sociais. As grandes empresas colecionadoras de dados, caso da Google, Amazon e Facebook, entre outras, desenvolveram bibliotecas e aplicações informáticas que permitem aprender com a informação e, assim, extrair mais-valia económica. Osexemplosmaisconhecidosdetoolkit emcasosdemachine learning pormeiodaanálisequantidades maciças de dados são: TensorFlow2, patrocinado pela Google e PyTorch3 pela Facebook. Há, no entanto, outro tipo de pacotes com outros modelos de aprendizagem. O exemplo mais conhecido é o scikit-learn4. A procura de engenheiros e cientistas com conhecimentos nesta área tem aumentado proporcionalmente ao valor financeiro criado. Uma das aplicações da aprendizagem por computador encontra-se no reconhecimento de faces5. Esta aplicação reúne o processamento digital de imagem6 a uma das técnicas de aprendizagem por meio de máquinas designada por reconhecimento de padrões7. Um dos pacotes de software usados em visão por computador e processamento de imagem mais potentes é o OpenCV8. É objetivo do trabalho:

•a realização duma aplicação de reconhecimento de faces usando a linguagem programação Python 3;

• a instalação do pacote OpenCV com suporte para Python 3;

• a instalação do pacote scikit-learn;

• a aplicação deverá permitir reconhecer a partir duma imagem recolhida através dum telemóvel e passada para um computador, a quem pertence a face;

• o teste da aplicação será realizado com base em imagens dos alunos da disciplina, sendo que coletivamente os alunos devem construir a base de dados de imagem de treino;

• o trabalho será individual;

• uma parte da classificação será proporcional à taxa de sucesso no reconhecimento das faces.

Desenvolvimento :

O desenvolvimento deste sistema de reconhecimento facial utilizando uma plataforma "linguagem programação Python3 e pacote OpenCV uso comum, é uma alternativa viável para realizar o controle e ou registro de acesso de pessoas em um determinado meio, pois permite a portabilidade/mobilidade e não exige gastos elevados para montar a solução. Além disso, tal projeto viabiliza aprofundar os conhecimentos em Linux bem como a utilização de ferramentas para o tratamento de imagens e de vídeos, como a biblioteca OpenCV. O completo desenvolvimento da plataforma ainda envolve sólidos conhecimentos de programação em linguagem pytthon, além da configuração e instalação de IDEA, no meu caso usei Spyder que vem já na Anaconda.

Na primeira parte da projeto eu fiz filmagem dos videos com um telemovel e passei os videos para meu computador e fiz um programa que vai ler os videos confrome eu mandei ou dar nome da pessoa e este no vai ser dado por um aquivo de txt e faz a recortagem dos faceis e criar arquivo para cada cara do videos e chamei o arquivo de BaseDados :

```
# -*- coding: cp1252 -*-
import numpy as np
import cv2
import os

def carregaNomesASeremLidos(txt):
    listaNomeAlunos = []
    pFile = open(txt, "r")
    for line in pFile:
        listaNomeAlunos.append(line.rstrip())
    return listaNomeAlunos

def criaPastaComNomes(listaNomes):
    for nome in listaNomes:
```

```
        try:
            print("criou " + nome)
            os.mkdir(nome)
        except OSError:
            print("Não foi possível criar o diretório ou o mesmo já existe.")
def salvaFacesDetectadas(nome):
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    cap = cv2.VideoCapture(nome + ".mp4") #inicia captura da câmera
    counterFrames = 0;
    while(counterFrames < 30): #quando chegar ao milésimo frame, para
        print(counterFrames)
        ret, img = cap.read()

        #frame não pode ser obtido? entao sair
        if(ret == False):
            cap.release()
            return
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        #se nenhuma face for achada, continue
        if not np.any(faces):
            continue

        #achou uma face? recorte ela (crop)
        for (x, y, w, h) in faces:
            rostoImg = img[y:y+h, x:x+w]
        #imagens muito pequenas são desconsideradas
```

```
        larg, alt, _ = rostoImg.shape
        if(larg * alt <= 20 * 20):
            continue
        #salva imagem na pasta
        rostoImg = cv2.resize(rostoImg, (255, 255))
        cv2.imwrite(nome + "/" + str(counterFrames)+".png", rostoImg)
        counterFrames += 1


    cap.release()
#função principal da aplicação
def main():
    listaNomeAlunos = carregaNomesASeremLidos("input.txt")
    criaPastaComNomes(listaNomeAlunos)


    for nome in listaNomeAlunos:
        print("Analisando: " + nome)
        salvaFacesDetectadas(nome)
if __name__ == "__main__":
    main()
```

Na segunda parate do meu projeto tentei usar tudo mais do que fala de boas praticas de prograçacao usar metodos funçoes e .t.c

Nesta parte queria usar controle de versao mais nao estava a funcionar por isso nao usei perante todo o meu trabalho so quando ternimei o trablho que vem a funcionar e mais o meu programa funciona de seguinte forma ,pega nos dados ou fotos cortadas na primeira parte e faz a compraçao com car a de um video onde estao estas pessoas e captao comtodo o nome deles se personar (q)programa sai.

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 27 05:41:46 2019

@author: Felizmelo
"""

# -*- coding: cp1252 -*-

# PCA - Principal component analysis
# É uma técnica utilizada para comprimir informações sem perder a essência
# O PCA faz isso com matrizes utilizando autovalores e autovetores
# O AutoFaces compacta a imagem do rosto de uma maneira que fica fácil comparar

# Funciona assim:
# O ViolaJones é o programa responsável por detectar as faces no vídeo
# Uma vez detectadas, ele envia as fotos apenas do rosto para o AutoFaces
# O AutoFaces faz o reconhecimento do rosto utilizando aquela imagem enviada


import numpy as np
import cv2
import os

def criaArquivoDeRotulo(pasta):
    # cria o arquivo de treino, que é um .txt com várias linhas no formato:
```

```
    # 'caminho completo da imagem';'rótulo'
    # sendo label o que classifica a foto. É apenas um número inteiro,
    # para identificar que é a pessoa 1 ou pessoa 2
    label = 0
    f = open("TRAIN", "w+")
    for dirPrincipal, nomeDirs, nomeArqs in os.walk(pasta):
        for subDir in nomeDirs:
            caminhoPasta = os.path.join(dirPrincipal, subDir)
            for filename in os.listdir(caminhoPasta):
                caminhoAbs = caminhoPasta + "\\" + filename
                f.write(caminhoAbs + ";" + str(label) + "\n")
            label = label + 1
    f.close()


def criaDicionarioDeImagens(fPoint):
    # Abre o arquivo de treino e cria um dicionário de fotos (já com as fotos abertas):
    # dicionarioDeFotos = {
    # 0: [Imagem1, Imagem2, ...],
    # 1: [Imagem1, Imagem2, ...],
    # ... }
    # A chave é o número do rótulo e o conteúdo é uma lista de fotos
    lines = fPoint.readlines()
    dicionarioDeFotos = {}
    for line in lines:
        filename, label = line.rstrip().split(';')
        if int(label) in dicionarioDeFotos.keys():
            dicionarioDeFotos.get(int(label)).append(cv2.imread(filename, 0))
```

```
        else:
            dicionarioDeFotos[int(label)] = [cv2.imread(filename, 0)]


    # ao final, cria um dicionário que na posição 0 (rótulo
    # referente a camila, por exemplo, tem-se uma lista contendo
    # todas as fotos da camila que estão na base de teste
    # na posição 1 do dicionário, teremos uma lista com todas as
    # fotos do pirula
    return dicionarioDeFotos
def treinaModelo(dicionarioAlunos):
    # cria e treina as autofaces
    model = cv2.face.EigenFaceRecognizer_create()
    listkey = []
    listvalue = []
    for key in dicionarioAlunos.keys():
        for value in dicionarioAlunos[key]:
            listkey.append(key)
            listvalue.append(value)
    model.train(np.array(listvalue), np.array(listkey))
    # param 1: todas as imagens:
    # [Imagem1, Imagem2, Imagem3, Imagem4, Imagem5, ...]
    # param 2: lista com todos os rótulos, na mesma ordem e com o mesmo
tamanho do vetor das fotos:
    # [0, 0, 1, 0, 1, ...]
    return model
def reconheceVideo(modelo, arquivo):
    face_cascade                                                    =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
cap = cv2.VideoCapture(arquivo)  # inicia captura da câmera
counterFrames = 0;
while (counterFrames < 30):  # quando chegar ao milésimo frame, para
    ret, img = cap.read()
    # frame não pode ser obtido? entao sair
    if (ret == False):
        cap.release()
        return
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    # se nenhuma face for achada, continue
    if not np.any(faces):
        continue
    rostos = []
    # achou uma face? recorte ela (crop)
    for (x, y, w, h) in faces:
        rosto = img[y:y + h, x:x + w]
        # esse rosto é grande o bastante pra ser levado
        # em conta
        if w > 100 and h > 100:
            # modifica o tamanho dele pra se ajustar ao
            # treinamento e pinte pra tons de cinza
            rosto = cv2.resize(rosto, (255, 255))
            rosto = cv2.cvtColor(rosto, cv2.COLOR_BGR2GRAY)
            # aqui ele recebe a foto e diz qual rótulo
            # pertence (ou seja, quem é)
            label = modelo.predict(rosto)
```

```
            font = cv2.FONT_HERSHEY_SIMPLEX
            if (label[0] == 0):  # é o Felizmelo ?
                # então bota um texto em cima da caixinha
                cv2.putText(img, 'Felizmelo', (x - 20, y + h + 60), font, 3, (255, 0, 0), 5, cv2.LINE_AA)
                # (imagem, texto, posicao, fonte, tamanho da  fonte, cor, expessura, antialiasing)
                # pinte um retângulo ao redor do rosto do Felizmelo
                img = cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            if (label[0] == 1):  # é Luiz?
                # então bota um texto em cima da caixinha
                cv2.putText(img, 'Luiz', (x - 20, y + h + 60), font, 3, (0, 0, 255), 5, cv2.LINE_AA)
                # pinte um retângulo ao redor do rosto de  Luiz
                img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)

        # redimensione só pra ficar bonito na tela
        img = cv2.resize(img, (int(0.75 * img.shape[1]), int(0.75 * img.shape[0])))

        # exibir na tela!
        cv2.imshow("reconhecimento", img)
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
def main():
    # cria um arquivo que indica que aquela foto pertence a tal pessoa
    criaArquivoDeRotulo("data")
```

```
    # carrega o arquivo
    fPoint = open("TRAIN", "r")
    # constrói um dicionário dos dados lidos no texto
    dicionarioDeFotos = criaDicionarioDeImagens(fPoint)
    modelo = treinaModelo(dicionarioDeFotos)
    # DO THE F* MAGIC DUDE
    reconheceVideo(modelo, "Felizmelo_Luiz.mp4")
if __name__ == "__main__":
    main()
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 27 05:41:46 2019

@author: Felizmelo
"""

# -*- coding: cp1252 -*-

# PCA - Principal component analysis
# É uma técnica utilizada para comprimir informações sem perder a essência
# O PCA faz isso com matrizes utilizando autovalores e autovetores
# O AutoFaces compacta a imagem do rosto de uma maneira que fica fácil comparar

# Funciona assim:
# O ViolaJones é o programa responsável por detectar as faces no vídeo
# Uma vez detectadas, ele envia as fotos apenas do rosto para o AutoFaces
```

```
# O AutoFaces faz o reconhecimento do rosto utilizando aquela imagem enviada


import numpy as np
import cv2
import os


def criaArquivoDeRotulo(pasta):
    # cria o arquivo de treino, que é um .txt com várias linhas no formato:
    # 'caminho completo da imagem';'rótulo'
    # sendo label o que classifica a foto. É apenas um número inteiro,
    # para identificar que é a pessoa 1 ou pessoa 2
    label = 0
    f = open("TRAIN", "w+")
    for dirPrincipal, nomeDirs, nomeArqs in os.walk(pasta):
        for subDir in nomeDirs:
            caminhoPasta = os.path.join(dirPrincipal, subDir)
            for filename in os.listdir(caminhoPasta):
                caminhoAbs = caminhoPasta + "\\" + filename
                f.write(caminhoAbs + ";" + str(label) + "\n")
            label = label + 1
    f.close()


def criaDicionarioDeImagens(fPoint):
```

```
    # Abre o arquivo de treino e cria um dicionário de fotos (já com as fotos
abertas):
    # dicionarioDeFotos = {
    # 0: [Imagem1, Imagem2, ...],
    # 1: [Imagem1, Imagem2, ...],
    # ... }
    # A chave é o número do rótulo e o conteúdo é uma lista de fotos
    lines = fPoint.readlines()

    dicionarioDeFotos = {}
    for line in lines:
        filename, label = line.rstrip().split(';')
        if int(label) in dicionarioDeFotos.keys():
            dicionarioDeFotos.get(int(label)).append(cv2.imread(filename, 0))
        else:
            dicionarioDeFotos[int(label)] = [cv2.imread(filename, 0)]

    # ao final, cria um dicionário que na posição 0 (rótulo
    # referente a camila, por exemplo, tem-se uma lista contendo
    # todas as fotos da camila que estão na base de teste

    # na posição 1 do dicionário, teremos uma lista com todas as
    # fotos do pirula
    return dicionarioDeFotos



def treinaModelo(dicionarioAlunos):
    # cria e treina as autofaces
```

```
    model = cv2.face.EigenFaceRecognizer_create()
    listkey = []
    listvalue = []
    for key in dicionarioAlunos.keys():
        for value in dicionarioAlunos[key]:
            listkey.append(key)
            listvalue.append(value)


    model.train(np.array(listvalue), np.array(listkey))
    # param 1: todas as imagens:
    # [Imagem1, Imagem2, Imagem3, Imagem4, Imagem5, ...]
    # param 2: lista com todos os rótulos, na mesma ordem e com o mesmo tamanho do vetor das fotos:
    # [0, 0, 1, 0, 1, ...]
    return model



def reconheceVideo(modelo, arquivo):
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    cap = cv2.VideoCapture(arquivo)  # inicia captura da câmera
    counterFrames = 0;
    while (counterFrames < 30):  # quando chegar ao milésimo frame, para
        ret, img = cap.read()


        # frame não pode ser obtido? entao sair
        if (ret == False):
            cap.release()
```

```
    return

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

# se nenhuma face for achada, continue
if not np.any(faces):
    continue

rostos = []
# achou uma face? recorte ela (crop)
for (x, y, w, h) in faces:
    rosto = img[y:y + h, x:x + w]
    # esse rosto é grande o bastante pra ser levado
    # em conta
    if w > 100 and h > 100:

        # modifica o tamanho dele pra se ajustar ao
        # treinamento e pinte pra tons de cinza
        rosto = cv2.resize(rosto, (255, 255))
        rosto = cv2.cvtColor(rosto, cv2.COLOR_BGR2GRAY)

        # aqui ele recebe a foto e diz qual rótulo
        # pertence (ou seja, quem é)
        label = modelo.predict(rosto)

        font = cv2.FONT_HERSHEY_SIMPLEX
```

```
            if (label[0] == 0):  # é o Felizmelo ?
                # então bota um texto em cima da caixinha
                cv2.putText(img, 'Felizmelo', (x - 20, y + h + 60), font, 3, (255, 0, 0), 5, cv2.LINE_AA)
                # (imagem, texto, posicao, fonte, tamanho da  fonte, cor, expessura, antialiasing)
                # pinte um retângulo ao redor do rosto do Felizmelo
                img = cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            if (label[0] == 1):  # é Luiz?
                # então bota um texto em cima da caixinha
                cv2.putText(img, 'Luiz', (x - 20, y + h + 60), font, 3, (0, 0, 255), 5, cv2.LINE_AA)
                # pinte um retângulo ao redor do rosto de  Luiz
                img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)

        # redimensione só pra ficar bonito na tela
        img = cv2.resize(img, (int(0.75 * img.shape[1]), int(0.75 * img.shape[0])))

        # exibir na tela!
        cv2.imshow("reconhecimento", img)
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()


def main():
    # cria um arquivo que indica que aquela foto pertence a tal pessoa
```

```
    criaArquivoDeRotulo("data")

    # carrega o arquivo
    fPoint = open("TRAIN", "r")

    # constrói um dicionário dos dados lidos no texto
    dicionarioDeFotos = criaDicionarioDeImagens(fPoint)
    modelo = treinaModelo(dicionarioDeFotos)

    # DO THE F* MAGIC DUDE
    reconheceVideo(modelo, "Felizmelo_Luiz.mp4")



if __name__ == "__main__":
    main()# -*- coding: utf-8 -*-
"""
Created on Sun Jan 27 05:41:46 2019

@author: Felizmelo
"""

# -*- coding: cp1252 -*-

# PCA - Principal component analysis
# É uma técnica utilizada para comprimir informações sem perder a essência
# O PCA faz isso com matrizes utilizando autovalores e autovetores
# O AutoFaces compacta a imagem do rosto de uma maneira que fica fácil comparar
```

```
# Funciona assim:
# O ViolaJones é o programa responsável por detectar as faces no vídeo
# Uma vez detectadas, ele envia as fotos apenas do rosto para o AutoFaces
# O AutoFaces faz o reconhecimento do rosto utilizando aquela imagem enviada


import numpy as np
import cv2
import os


def criaArquivoDeRotulo(pasta):
    # cria o arquivo de treino, que é um .txt com várias linhas no formato:
    # 'caminho completo da imagem';'rótulo'
    # sendo label o que classifica a foto. É apenas um número inteiro,
    # para identificar que é a pessoa 1 ou pessoa 2
    label = 0
    f = open("TRAIN", "w+")
    for dirPrincipal, nomeDirs, nomeArqs in os.walk(pasta):
        for subDir in nomeDirs:
            caminhoPasta = os.path.join(dirPrincipal, subDir)
            for filename in os.listdir(caminhoPasta):
                caminhoAbs = caminhoPasta + "\\" + filename
                f.write(caminhoAbs + ";" + str(label) + "\n")
            label = label + 1
    f.close()
```

```
def criaDicionarioDeImagens(fPoint):
    # Abre o arquivo de treino e cria um dicionário de fotos (já com as fotos abertas):
    # dicionarioDeFotos = {
    # 0: [Imagem1, Imagem2, ...],
    # 1: [Imagem1, Imagem2, ...],
    # ... }
    # A chave é o número do rótulo e o conteúdo é uma lista de fotos
    lines = fPoint.readlines()

    dicionarioDeFotos = {}
    for line in lines:
        filename, label = line.rstrip().split(';')
        if int(label) in dicionarioDeFotos.keys():
            dicionarioDeFotos.get(int(label)).append(cv2.imread(filename, 0))
        else:
            dicionarioDeFotos[int(label)] = [cv2.imread(filename, 0)]

    # ao final, cria um dicionário que na posição 0 (rótulo
    # referente a camila, por exemplo, tem-se uma lista contendo
    # todas as fotos da camila que estão na base de teste

    # na posição 1 do dicionário, teremos uma lista com todas as
    # fotos do pirula
    return dicionarioDeFotos
```

```
def treinaModelo(dicionarioAlunos):
    # cria e treina as autofaces
    model = cv2.face.EigenFaceRecognizer_create()
    listkey = []
    listvalue = []
    for key in dicionarioAlunos.keys():
        for value in dicionarioAlunos[key]:
            listkey.append(key)
            listvalue.append(value)


    model.train(np.array(listvalue), np.array(listkey))
    # param 1: todas as imagens:
    # [Imagem1, Imagem2, Imagem3, Imagem4, Imagem5, ...]
    # param 2: lista com todos os rótulos, na mesma ordem e com o mesmo tamanho do vetor das fotos:
    # [0, 0, 1, 0, 1, ...]
    return model



def reconheceVideo(modelo, arquivo):
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    cap = cv2.VideoCapture(arquivo)  # inicia captura da câmera
    counterFrames = 0;
    while (counterFrames < 30):  # quando chegar ao milésimo frame, para
        ret, img = cap.read()
```

```
# frame não pode ser obtido? entao sair
if (ret == False):
    cap.release()
    return

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

# se nenhuma face for achada, continue
if not np.any(faces):
    continue

rostos = []
# achou uma face? recorte ela (crop)
for (x, y, w, h) in faces:
    rosto = img[y:y + h, x:x + w]
    # esse rosto é grande o bastante pra ser levado
    # em conta
    if w > 100 and h > 100:

        # modifica o tamanho dele pra se ajustar ao
        # treinamento e pinte pra tons de cinza
        rosto = cv2.resize(rosto, (255, 255))
        rosto = cv2.cvtColor(rosto, cv2.COLOR_BGR2GRAY)

        # aqui ele recebe a foto e diz qual rótulo
        # pertence (ou seja, quem é)
```

```
            label = modelo.predict(rosto)


            font = cv2.FONT_HERSHEY_SIMPLEX
            if (label[0] == 0):  # é o Felizmelo ?
               # então bota um texto em cima da caixinha
               cv2.putText(img, 'Felizmelo', (x - 20, y + h + 60), font, 3, (255, 0, 0), 5, cv2.LINE_AA)
               # (imagem, texto, posicao, fonte, tamanho da  fonte, cor, expessura, antialiasing)
               # pinte um retângulo ao redor do rosto do Felizmelo
               img = cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            if (label[0] == 1):  # é Luiz?
               # então bota um texto em cima da caixinha
               cv2.putText(img, 'Luiz', (x - 20, y + h + 60), font, 3, (0, 0, 255), 5, cv2.LINE_AA)
               # pinte um retângulo ao redor do rosto de  Luiz
               img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)


      # redimensione só pra ficar bonito na tela
      img = cv2.resize(img, (int(0.75 * img.shape[1]), int(0.75 * img.shape[0])))


      # exibir na tela!
      cv2.imshow("reconhecimento", img)
      if cv2.waitKey(10) & 0xFF == ord('q'):
         break
   cap.release()
   cv2.destroyAllWindows()
```

```
def main():
    # cria um arquivo que indica que aquela foto pertence a tal pessoa
    criaArquivoDeRotulo("data")

    # carrega o arquivo
    fPoint = open("TRAIN", "r")

    # constrói um dicionário dos dados lidos no texto
    dicionarioDeFotos = criaDicionarioDeImagens(fPoint)
    modelo = treinaModelo(dicionarioDeFotos)

    # DO THE F* MAGIC DUDE
    reconheceVideo(modelo, "Felizmelo_Luiz.mp4")


if __name__ == "__main__":
    main()
```

Conclusões

Ao término deste projeto e diante dos resultados obtidos, pode-se inferir que os objetivos iniciais propostos para o projeto foram alcançados. Utilizando-se *Linux*, python e opencv, desenvolveu-se uma solução de baixo custo e baixa complexidade que realiza a detecção e reconhecimento facial com o auxílio da biblioteca *OpenCV*. Além disso, empregou-se técnicas de tratamento de multimídia, estudou-se e aplicou-se conceitos e algoritmos de visão computacional e fez-se uma análise desta solução.

Observando os resultados adquiridos, considerando-se que tanto o tempo médio de processamento para identificação como a taxa de sucesso na identificação apresentaram melhores resultados com a utilização da detecção de imagens por meio do algoritmo de *LBP*, pode-se concluir que a aplicação embarcada na *Beaglebone Black* a qual realiza a identificação facial apresenta melhor desempenho quando comparado com o algoritmo de Viola Jones.

Além disso, pode-se concluir que o tempo exigido para identificação de uma face não é elevado

Referências bibliográficas

1Em Inglês machine learning, vidé https://en.wikipedia.org/wiki/Machine_learning.
2https://www.tensorflow.org/ 3https://pytorch.org/ 4http://scikit-learn.org/stable/index.html
5https://en.wikipedia.org/wiki/Facial_recognition_system
6https://en.wikipedia.org/wiki/Digital_image_processing.
7https://en.wikipedia.org/wiki/Pattern_recognition 8https://opencv.org/
9https://realpython.com/face-recognition-with-python/
10https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

[1] P.I. Okwu and I.N. Onyeje "Ubiquitous Embedded Systems Revolution: Applications and Emerging Trends", International Journal of Engineering Research and Applications (IJERA)
Vol. 3, Issue 4, Jul-Aug 2013, pp.610-616

[2] *Beagleboard.org, 'BeagleBoard.org - black', 2015.* [Online]. Disponível: http://beagleboard.org/black. [Acesso em 28 de maio de 2015].

[3] *Opencv.org, 'OpenCV | OpenCV', 2015*. [Online]. Disponível: http://opencv.org. [Acessado em 28 de maio de 2015].

[4] *M. Mitchell, J. Oldham and A. Samuel, Advanced Linux programming. Indianapolis, Ind.: New Riders Pub., 2001*.

[5] R. Silva and E. Alecrim, 'A tecnologia por trás da Rover *Curiosity | Tecnoblog', Tecnoblog*, 2012. [Online]. Disponível: https://tecnoblog.net/109317/tecnologia-rover-curiosity/. [Acesso em 01 de maio de 2015].