Aplicação para Reconhecimento Facial com algoritmo Local Binary Patern Histogram com biblioteca Opencv e Linguagem Python.

André Vieira da Silva
Bacharelado em Engenharia da Computação
Instituto federal do Ceará Campus Fortaleza
Fortaleza, Ceará, Brazil
sgavsnake@gmail.com

***Resumo*—Há algum tempo tem-se a Visão Computacional com o objetivo de emular ,simular a visão humana ,com o intuito de produzir elementos tecnológicos capazes de reconhecer padrões,objetos e ainda situações através de operações matemáticas estatísticas sobre imagens.Nisso é função deste documento aplicar algumas dessas operações disponíveis na biblioteca opencv com ajuda da linguagem python de forma ao final ser capaz de desenvolver uma aplicação para reconhecer e identificar uma pessoa com obviamente uma avaliação pontual sobre os resultados e os processos utilizados para alcançar o objetivo já citado além de uma breve introdução à Visão Computacional com posterior foco nos Atributos e características de uma imagem que forneceram a base para a construção da aplicação.**

***Index Terms*—Visão Computacional, Reconhecimento de Imagens, Reconhecimento Facial.**

I. Introdução

Ao longo dos anos a capacidade de processamento computacional tem aumentado vertiginosamente .Isso permitiu que novas ferramentas ,tecnologias e áreas de pesquisa fossem sendo desenvolvidas em um processo recursivamente interativo.Neste processo cada elemento dentre os citados produz ou aplica ,valida conceitos ,redescobre outras utilidades e fontes de pesquisa,recursos advindos dos já citados de forma que é um sistema realimentado de conhecimento um verdadeiro ciclo preso e retroalimentado.

Uma aplicação bastante comum desses recursos é a identificação de pessoas,onde basicamente, temos reconhecimento da face de uma pessoa ,se de fato a imagem apresentada é de uma pessoa ou não, e a identificação propriamente dita,onde agora a imagem adquirida é submetida a um processo que permite saber se a imagem é de uma determinada pessoa ou não.

II. Visão Computacional uma visão geral

A. Definições

Visão computacional é a ciência e tecnologia das máquinas que enxergam. Ela desenvolve teoria e tecnologia para a construção de sistemas artificiais que obtém informação de imagens ou quaisquer dados multidimensionais. Exemplos de aplicações incluem o controle de processos (como robôs industriais ou veículos autônomos), detecção de eventos, organização de informação, modelagem de objetos ou ambientes e interação (atrelado a interação homem-computador).

A visão computacional também pode ser descrita como um complemento da visão biológica. Na visão biológica, a percepção visual dos humanos e outros animais é estudada, resultando em modelos em como tais sistemas operam em termos de processos fisiológicos. Por outro lado, a visão computacional estuda e descreve sistemas de visão artificial implementados por hardware ou software. Subcampos de pesquisa incluem reconstrução de cena, detecção de eventos, reconhecimento de objetos, aprendizagem de máquina e restauração de imagens.

B. Exemplos de aplicações

Uma das mais difundidas aplicações da visão computacional é a Medicina, ou o processamento médico das imagens. Tal área é caracterizada pela extração de informação de imagens para realizar diagnósticos sobre os pacientes. Fontes de imagens incluem imagens de microscopia, de radiografia, de angioplastia, de ultrassonografia, de tomografia e de Ressonância magnética. Outra aplicação bastante difundida é a indústria. Aqui, a informação obtida auxilia processos como a inspeção de controle de qualidade e cálculo de posição e orientação de detalhes para um braço robótico por exemplo. As aplicações militares são talvez uma das maiores da visão computacional, ainda que apenas uma pequena parte desse trabalho esteja disponível ao público. Exemplos básicos incluem a detecção de unidades inimigas e ou mísseis teleguiados. Sistemas mais avançados enviam mísseis para uma área ao invés de um alvo específico, sendo que a seleção do alvo é feita no processamento da imagem do local feita pelo próprio míssil. Uma das novas aplicações são os veículos autônomos, cujo nível de autonomia varia entre total ou parcial, este último usado para somente auxiliar a tarefa de dirigir em situações diversas. A autonomia total usa a visão computacional para a navegação, isto é, para obter a localização, para produzir mapas do ambiente e para detectar obstáculos. Várias montadoras já demonstraram veículos totalmente autônomos, mas tal tecnologia ainda não atingiu maturidade suficiente para

estar no mercado. A exploração espacial também está usando veículos autônomos usando a visão computacional, como por exemplo a Mars Exploration Rover da NASA.

C. Aplicações Típicas

Cada uma das áreas de aplicação descritas abaixo utilizam um conjunto de tarefas de visão computacional.

a) Reconhecimento: O problema clássico da visão computacional e do processamento de imagens é determinar se uma imagem contém ou não um dado objeto, uma dada característica ou uma dada atividade. Tal tarefa pode ser resolvida de forma robusta e sem esforço humano, mas ainda não foi resolvida satisfatoriamente para o caso geral, objetos arbitrários em situações arbitrárias. Os métodos atuais conseguem, no máximo, resolver para objetos específicos, como poliedros, faces humanas, letras escritas à mão ou veículos; também em situações específicas, como iluminação bem definida, fundo fixo e pose dos objetos bem definida. Diferentes variedades de problemas de reconhecimento são descritas na literatura:

- **Reconhecimento**: uma ou várias classes pré-definidas ou aprendidas de objetos podem ser reconhecidas, geralmente em conjunto com sua posição em imagens bidimensionais ou com sua pose em imagens tridimensionais.
- **Identificação** : uma instância individual de um objeto pode ser reconhecida, como a identificação de uma face ou de impressão digital, ou até mesmo a identificação de um veículo.
- **Detecção**: a imagem é digitalizada para uma condição específica, como a detecção de células ou tecidos anormais.

b) Movimento: Várias tarefas estão relacionadas a estimativa do movimento, no qual uma sequência de imagens é processada para produzir uma estimativa da velocidade em cada ponto.

c) Reconstrução de cena: Dadas duas ou mais imagens de uma cena, ou um vídeo, a reconstrução de cena visa computar um modelo tridimensional da cena. No caso mais simples, o modelo consiste somente em um conjunto de pontos tridimensionais; métodos mais sofisticados reconstroem também texturas e cores.

d) Restauração de imagens: O objetivo da restauração de imagens é a remoção de ruídos.

D. Sistemas de Visão Computacional

A organização de um sistema de visão computacional é dependente da aplicação. A implementação específica de tal sistema depende também se sua funcionalidade é pré-especificada ou se existe alguma parte de aprendizagem durante a operação. Existem, entretanto, funções típicas encontradas vários sistemas de visão computacional:

a) Aquisição de imagem: Uma imagem digital é produzida por um ou vários sensores. Dependendo do tipo do sensor, o resultado pode variar entre uma imagem bidimensional, uma cena tridimensional ou ainda uma sequência de imagens. Os valores dos pixels geralmente indicam a intensidade da luz em uma ou várias faixas de cor (o que forma imagens em tom de cinza ou coloridas), mas também podem indicar valores físicos como profundidade e absorção ou reflexão das ondas eletromagnéticas.

b) Pré-processamento: Antes de um método de visão computacional ser aplicado em uma imagem para extrair informação, é geralmente necessário processar a imagem para assegurar-se que ela satisfaz as condições do método. Exemplos incluem remapeamento (para assegurar o sistema de coordenadas), redução de ruídos (para assegurar que as informações são verdadeiras) e aumento de contraste (para assegurar que as informações relevantes serão detectadas).

c) Extração de características: Características matemáticas da imagem em vários níveis de complexidade são extraídas. Exemplos básicos incluem detecção de bordas, cantos ou pontos. Exemplo sofisticados incluem a morfologia matemática, detecção de texturas, formatos e movimentos.

d) Detecção e segmentação: Em algum ponto do processo uma decisão é feita sobre a relevância de regiões da imagem para processamento posterior. Exemplos incluem a seleção de regiões de interesse específicos e segmentação de uma ou mais regiões que contém um objeto de interesse.

e) Processamento de alto nível: Neste ponto a entrada é geralmente um conjunto pequeno de dados. O processo posterior inclui a verificação da satisfação dos dados, a estimativa de parâmetros sobre a imagem e a classificação dos objetos detectados em diferentes categorias.

III. Atributos,Características de uma imagem

Na visão computacional e no processamento de imagens , atributos são informações relevantes à natureza de um problema de uma dada aplicação computacional. Esse conceito é muito empregado nas aplicações que fazem uso dos recursos de Machine Learning e reconhecimento de padrões.

O atributos ,características de uma imagem ,no processamento de imagens,são em geral uma coleção considerável de elementos de origem matemática ,probabilística ,heurística ,formas ,estruturas enfim muitos elementos e fontes de origem de operações aplicadas a imagem em processamento.Isto torna a conceituação de caracterização de uma imagem muito abrangente sendo necessário técnicas adequadas para a extração de atributos específcos,uma filtragem dos elementos descritores que são realmente relevantes ao domínio da aplicação em desenvolvimento.

Limitando um pouco o nível de abstração ,extrair os atributos ou características ,em visão computacional ,de uma imagem nada mais é que associar à imagem a ser processada,o resultado de um processamento computacional qualquer. Com todas essas afirmações chegamos a um problema recorrente,o de como representar uma imagem,que dados,que parâmetros que características podem identificar de forma única cada imagem dentro de um conjunto de elementos .O que os métodos de extração de atributos pretendem individualmente ou em conjunto com outros métodos computacionais disponibilizar ao mundo infinito das aplicações ,lembrando que é a natureza do problema que define o leque dos parâmetros da solução.

IV. Reconhecimento de Objetos

Umas das principais atividades da visão computacional diz respeito ao reconhecimento de objetos.Esta etapa da visão computacional se define como sendo o processo que identifica um ou mais elementos de uma imagem baseados um etapa de treinamento, onde o esforço computacional e dispendido literalmente ao aprendizado.Esta etapa de treinamento pode ser:

- **Supervisionado**:Os dados da base de treino são rotulados de que ja sabemos a saída correta do processamento de um algoritmo de reconhecimento.
- **Não Supervisionado** : Os dados da base de treino não permite um identificação dos resultados esperados pelo processamento de um algoritmo de reconhecimento.

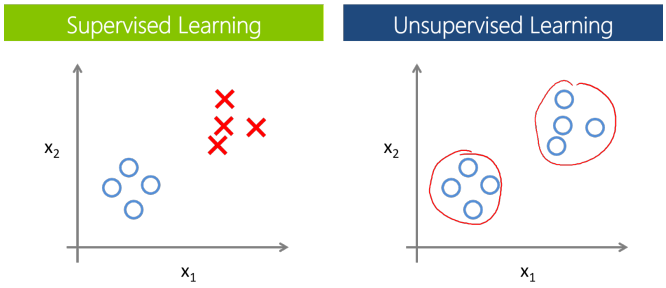

Figura 1: Diferentes abordagens de supervisão

V. Algoritmo de Classificação e identificação facial

A. LBPH - Local Binary Patern Histograms

O algoritmo de LBPH é um algoritmo que já data de alguns anos,muito usado e conhecido no reconhecimento de objetos na área de visão computacional para a classificação.O algoritmo tem a característica de não ser muito sensível a luminosidade.O LBP é um operador de textura simples, porém eficiente, que rotula os pixels de uma imagem ao limitar a vizinhança de cada pixel e considera o resultado como um número binário.Foi descrito pela primeira vez em 1994 (LBP) e, desde então, foi considerado um recurso poderoso para a classificação de textura. Ainda, quando o LBP é combinado com os Histograms of Oriented Gradients (HOG), ele melhora o desempenho da detecção consideravelmente em alguns conjuntos de dados.[4]

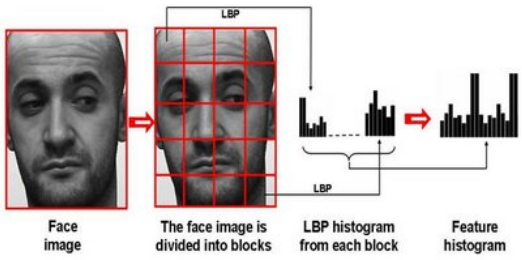

Figura 2: Abstração de trabalho do algoritmo LBP

O LBPH em termos de algoritmo possui 4 parâmetros sendo:

- 1. Raio : Adaptação feita para o algoritmo considerar uma distancia circular para construção do padrão binário,o padrão desse valor é um.
- 2. Vizinhos : tamanho da vizinhança ao redor do ponto na iteração corrente ,custo computacional é proporcional a esse parâmetro.
- 3. Grade X : O número de células na direção horizontal. Quanto mais células mais na é a grade e maior é a dimensionalidade do vetor de características resultante.Geralmente é definido como 8.
- 4. Grade Y : Igual a Grade X só que na vertical.

O LBPH em termos de algoritmo possui 4 parâmetros sendo: 1. Raio : Adaptação feita para o algoritmo considerar uma distancia circular para construção do padrão binário,o padrão desse valor é um. 2. Vizinhos : tamanho da vizinhança ao redor do ponto na iteração corrente ,custo computacional é proporcional a esse parâmetro. 3. Grade X : O número de células na direção horizontal. Quanto mais células mais na é a grade e maior é a dimensionalidade do vetor de características resultante.Geralmente é definido como 8. 4. Grade Y : Igual a Grade X só que na vertical. Abaixo[10]

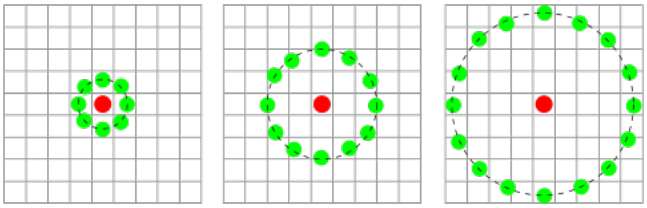

Figura 3: Exemplo de Vizinhança para cálculo no LBP

O algoritmo ainda possui a vantagem de ser invariante a posição dos objetos na imagem,e continua ainda a ser estudado de forma que há diversas versões disponíveis para a classificação de objetos em geral ou de classes de objetos específicos como pessoas por exemplo.A partir de agora começaremos a aplicação de reconhecimento e identificação de pessoas e mais detalhes sobre o algoritmo serão fornecidos quando assim convier. As próximas etapas serão:

B. Reconhecendo faces

Inicialmente precisamos criar um programa que seja capaz de reconhecer e identificar as faces humanas que estiverem em frente a uma câmera por conveniência também haverá a possibilidade de capturar as imagens baseando-se no valor da média das intensidades dos pixels da imagem isso serve para garantir um nível de iluminação adequado de forma que as etapas seguintes do processo de classificação não sejam prejudicadas simplesmente pela falta de iluminação nas imagens.

a) :

```
import cv2
import numpy as np

classificador = cv2.CascadeClassifier("haarcascade-frontalface-default.xml")
classificadorOlho = cv2.CascadeClassifier("haarcascade-eye.xml")
font = cv2.FONT_HERSHEY_COMPLEX_SMALL

camera = cv2.VideoCapture(0)

amostra = 1
numerodeAmostras = 20
id = input("Digite Seu Identificador:")
largura,altura = 220,220
print("Capturando Faces")

while(True):
        conectado,imagem = camera.read()
        imagemCinza = cv2.cvtColor(imagem,cv2.COLOR_BGR2GRAY)
        facesDetectadas = classificador.detectMultiScale(imagemCinza,scaleFactor=1.5,minSize
            =(100,100))
        print(np.average(imagemCinza))

        for(x,y,l,a) in facesDetectadas:
                cv2.rectangle(imagem,(x,y),(x+l,y+a),(0,0,255),2)
                regiao = imagem[y:y + a, x:x + l]
                regiaoCinzaOlho = cv2.cvtColor(regiao, cv2.COLOR_BGR2GRAY)
                olhosDetectados = classificadorOlho.detectMultiScale(regiaoCinzaOlho)

                for(oX, oY, oL, oA) in olhosDetectados:
                        cv2.rectangle(regiao, (oX, oY), (oX + oL,oY + oA), (0,255, 0), 2)

                        if(cv2.waitKey(1) % 0xFF == ord("q")):

                                if(np.average(imagemCinza) > 110):
                                        imagemFace = cv2.resize(imagemCinza[y:y+a,x:x+l],(
                                            largura,altura))
                                        cv2.imwrite("fotos/pessoa." + str(id)+"." + str(
                                            amostra) + ".jpg",imagemFace)
                                        print("foto " + str(amostra) + "capturada com sucesso"
                                            )
                                        amostra+=1

                cv2.putText(imagem, "Average", (x, y + (a + 30)), font, 2, (0, 0, 255))
                cv2.putText(imagem, str(np.average(imagemCinza)), (x, y + (a + 50)), font, 1,
                    (0, 0, 255))

        cv2.imshow("Camera",imagem)
        cv2.waitKey(1)

        if(amostra >= numerodeAmostras+1):

                break

print("Faces Capturadas com Sucesso")

camera.release()

cv2.destroyAllWindows()
```

Figura 4: Código Responsável pela captura de imagens e cadastro de indivíduos

O algoritmo acima permite a captura das imagens das faces e um cadastro de uma pessoa através de um número id informado pelo usuário no cadastro ao sistema, após informar este número é possível fazer a captura de 20 imagens para treino da base de dados para futuro reconhecimento e identificação.Para a captura devera ser pressionado a tecla 'q' e o algoritmo só permite a captura de uma imagem se a média que é informada na tela ,for superior a 110 ,isso á garante um bom limiar de iluminação na imagem para o não prejuízo.O resultado é um conjunto de imagens da face dos usuários, região de interesse para a aplicação(ROI), em escale de cinza onde são destaques a tês propriamente dita em especial a região dos olhos.[6]

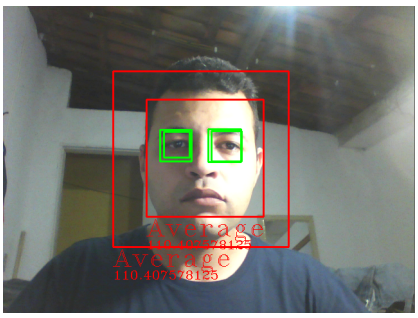

Figura 5: Visão da Camera com Grades de Cercando a ROI

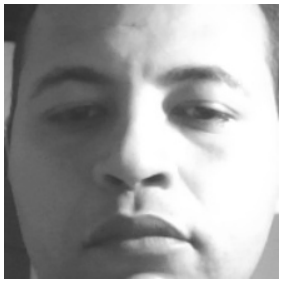

Figura 6: Visão da ROI processada pelo algoritmo.

C. Treinando com Base de Dados

Pra vermos a aplicação trabalhando vamos treinar o classificador LBPH com uma base de dados de faces de pessoas ,faces95,disponibilizado pelo Dr Libor Spacek no site http://cswww.essex.ac.uk/mv/allfaces a mesma como o autor mesmo coloca pode ser usada para este fim e seu comercial não é autorizado. A base citada possui 72 conjuntos de fotos coloridas de diversas pessoas do busto cada conjunto com 20 imagens de ambos os sexos de com uma variedade bastante adequada ao nosso proposito. A base foi pre-processada afim de chegar a um conjunto de imagens em tons de cinza de tamanho uniforme de maneira a evitar uma massa de dados e um esforço computacional desnecessário além de se poder estabelecer um plano mais seguro para posteriores avaliações.

No primeiro momento cada imagem do conjunto passará por uma binarização local da vizinhança do pixel central onde o valor de threshold será o valor de intensidade deste pixel na mascara que percorre a imagem.Essa mascara terá seu tamanho

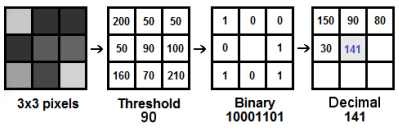

Figura 7: Forma de Trabalho do LBP.

definido de acordo com o parâmetro raio fornecido ao algoritmo.Após a limiarização local a vizinhança estará binarizada de forma que percorrendo ela no sentido predefinido pode-se formar um número de base dois que ao ser convertido para a base decimal substituirá o valor do pixel central antes o valor de threshold para a binarização.O resultado de todo o processo será uma imagem como a abaixo apresentad,uma textura visual parecida ser formada por completamente por ruído sal e pimenta.

Figura 8: Resultado da execução do algoritmo LBP.

A segunda etapa divide em regiões,janelas, com tamanho definido pelos parâmetros GridX e GridY a imagem resultante do processamento do LBP.O processo agora é calcular para cada região um histograma local e ao final concatenar todos esses histogramas geradas ,e ao final,o resultado será um histograma de características da imagem. Todos os valores nos histogramas gerados pela base de treino das imagens irão formar o base para a classificação e identificação posteriores.A imagem ilustra esse processo.

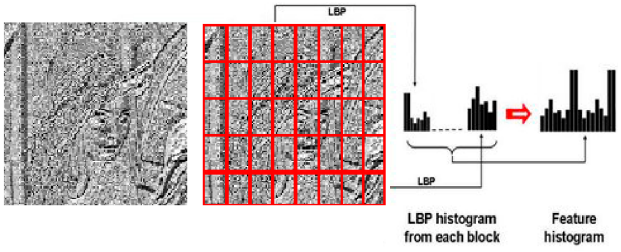

Figura 9: Extração do Histograma de Características.

Na seção seguinte teremos os códigos da implementação em linguagem python da aplicação de reconhecimento e identificação. [5] [2] [1] [3]

```
import cv2
import  os
import numpy as np
from PIL import Image
lbph = cv2.face.LBPHFaceRecognizer_create()
def formatImageData():
        size = 220, 220
        folder = [os.path.join('yale2/faces95/', f) for f in os.listdir('yale2/faces95')]
        caminhos = []
                for item in folder:
                        for p, _, files in os.walk(os.path.relpath(item)):
                                for file in files:
                                        caminhos.append(os.path.join(p, file))
        countIds = 1
        nomes = []

        for caminhosImagens in caminhos:
                try:
                        file = (os.path.split(caminhosImagens)[-1].split('.')[0:2])
                        basepath = (os.path.split(caminhosImagens))[0].split('\\')[0:3]

                        # basepath = "{param}\{param1}\\".format(param=basepath[0],param1="
                            fotos")
                        basepath = "{param}/".format (param="fotos")

                        if(not(file[0] in nomes) and "jpg" in caminhosImagens):
                                nomes.append(file[0])
                                countIds += 1
                        nome = "{nome1}.{id}.{nome2}.jpg".format(nome1=file[0], id=countIds,
                            nome2=file[1])
                        print("file = ",end=" ")
                        print(file,end='---')
                        print("nome = ",end=" ")
                        print(nome,end=" ---")
                        print(nomes)
                        # input()
                        if("jpg" in caminhosImagens):
                                image = Image.open(caminhosImagens).convert('L')
                                image.thumbnail(size)
                                nomeFile = "{param1}{param2}".format(param1=basepath,param2=
                                    nome)
                                image = np.array(image)
                                print(nomeFile)
                                # cv2.imshow("image",image)
                                # cv2.waitKey()
                        cv2.imwrite(nomeFile,image)

                except:
                        continue
```

Figura 10: Código Responsável pela adequação da base de dados para treinamento

```
import cv2

def getImageComId():
        caminhos = [os.path.join('fotos',f)for f in os.listdir('fotos')]
        faces = []
        ids = []
        for caminhosImagens in caminhos:
                imagemFace = cv2.cvtColor(cv2.imread(caminhosImagens),cv2.COLOR_BGR2GRAY)
                id = int(os.path.split(caminhosImagens)[-1].split('.')[1])
                ids.append(id)
                faces.append(imagemFace)
        print(ids)
        return np.array(ids),faces

ids,faces = getImageComId()

print("Treinamento...LBPH")
lbph.train(faces,ids)
lbph.write("classificadorLbph.yml")
```

Figura 11: Código Responsável pelo Treinamento do Algoritmo LBPH

```
import cv2

detectorFace = cv2.CascadeClassifier("haarcascade-frontalface-default.xml")

reconhecedor = cv2.face.LBPHFaceRecognizer_create()

reconhecedor.read("classificadorLbph.yml")

largura,altura = 220,220
font = cv2.FONT_HERSHEY_COMPLEX_SMALL

camera = cv2.VideoCapture(0)

while(True):
        conectado,imagem = camera.read()
        imagemCinza = cv2.cvtColor(imagem,cv2.COLOR_BGR2GRAY)
        facesDetectadas = detectorFace.detectMultiScale(imagemCinza,
        scaleFactor=1.5,
        minSize=(30,30))
        for(x,y,l,a) in facesDetectadas:
                imagemFace = cv2.resize(imagemCinza[y : y + a,x : x + l],(largura,altura))
                cv2.rectangle(imagem,(x,y),(x+l,y+a),(0,0,255),2)
                id,confianca = reconhecedor.predict(imagemFace)
                print("id = {id}".format(id=id))
                if (id == 1):
                        nome = "Identificado"
                else:
                        nome = "Nao Identificado"
                cv2.putText(imagem,nome,(x,y+(a+30)),font,2,(0,0,255))
                cv2.putText(imagem, str(int(confianca)), (x, y + (a + 50)), font, 1, (0, 0,
                    255))
        cv2.imshow("Face",imagem)
        if(cv2.waitKey(1) == ord('q')):
                break
camera.release()
cv2.destroyAllWindows()
```

Figura 12: Código Responsável pela Identificação com LBPH

D. Avaliação da Implementação para Reconhecimento e Identificação Facial

O algoritmo implementado atingiu o objetivo fim de identificar pessoas pre-cadastradas,como era o obetivo apresentado na seção anterior. Após o devido treinamento,implementação do algoritmo da figura 11, da base de dados formatada pelo algoritmo da Figura 10,podemos realizar o treinamento para a classificação e identificação de pessoas cujo código é apresentado na figura 10. A identificação é feita em tempo real e ainda como parâmetro de qualidade mostramos a confiança que mede o basicamente o grau de certeza essa métrica de confiança tenderá a zero quanto mais correto algoritmo estiver sobre a identificação do individuo.

Figura 13: Pessoa Cadastrada Identificada Corretamente

E. Testando aplicação de Reconhecimento Facial

Passaremos agora a valiar de um forma menos subjetiva a implementação utilizado nesse ponto utilizaremos uma base de dados menor,a da Universidade de Yale,sob os mesmos critérios de uso de licença da base de dados fornecida pelo site do Dr Libor Spacek. A base foi dividida em duas pastas uma pasta contendo 15 pessoas cada uma com 9 imagens de diferentes posições e até mesmo adereços, como óculos, em escala de cinza de ambos os sexos que será usada pra o treinamento do algoritmo e outra com as 2 imagens para cada uma das 15 pessoas já citadas que serão usadas para testar o reconhecimento da implementação realizada e descrita anteriormente.

Numero 0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	2	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	1	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1	0	0	0	0	0
11	0	0	0	0	0	0	0	0	2	0	0	0	0
12	0	0	0	0	0	0	0	1	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	2	0	0
14	0	0	0	0	0	0	0	0	0	0	0	2	0
15	0	0	0	0	0	1	0	0	0	0	0	0	1

Tabela I: Matriz de Confusão do Algoritmo LBPH

Descrição	Valor
Accuracy	0.63
Confiança	1.212

Tabela II: Tabela de Accuracy e Confiança obtidas no teste da implementação

A matriz de confusão permite aferir que a implementação para uma base de dados razoavelmente pequena ainda possui uma quantidade de erros considerável afinal houveram casos em que a implementação não conseguiu identificar corretamente o que nos leva a sermos críticos aonde devemos usar a implementação e de que forma podemos realizar melhorias otimizações para que se alcance um resultado melhor.Muito provavelmente uma base de dados maior já permitiria melhorar bastante a qualidade da classificação pois mais dados seriam produzidos para a devida distinção de cada individuo ,contudo,em contra partida teríamos o tempo de processamento sendo elevado proporcionalmente ao tamanho da base principalmente para a etapa de treinamento.

VI. Trabalhos Futuros

Levando em consideração os levamentos já mencionados pode-se propor para continuidade desses trabalhos basicamente :

- Uma base de dados mais adequada para o modelo da aplicação sugerida,nos testes 9 imagens para cada um em um conjunto de 15 pessoas foi verificado ser insuficiente na implementação chegou-se a usar uma base com 20 imagens para cada pessoa no conjunto sendo significativa a melhoria na confiança e na taxa de acerto.
- Etapas de pre-processamento nas imagens de entrada afim de otimizar a base de classificação melhorando os resultados obtidos,isto ainda a ser avaliado.
- Aumento no conjunto de regiões de interesse (ROI) para gtambém gerar uma base de classificação melhor adequada a aplicação.

Referências

[1] *Face recognition with local binary patterns*, 2002.
[2] *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*, 2002.
[3] *Face description with local binary patterns: Application to face recognition.*, 2006.
[4] A. V. da Silva. metodos de extração e classificação de atributos.
[5] gonzales e woods. *Processamento Digital de Imagens*. Pearson, 2010.
[6] J. Granatyr. reconhecimento-facial-com-python-e-opencv/. —.
[7] Manual. Lbph. Technical report, Opencv, 2018.
[8] prof.Pedrosa. Lab.lapisco. —.
[9] prof.Pedrosa. site. —.
[10] K. Salton. face-recognition-how-lbph-works.
[11] schoalarpedia. $Local_binary_patterns. Technical report, schoalarpedia$, 2018.