SISTEMA DE ANÁLISE FACIAL PARA CHAMADA ESCOLAR

Victor Hugo B. Tavares, Wilton Miro Barros Júnior

Programa de Graduação em Engenharia Eletrônica, Faculdade Gama
Universidade de Brasília
Gama, DF, Brasil
email: victorhugo.tavares@hotmail.com, wiltonjrfla@gmail.com

1. INTRODUÇÃO

Um dos problemas que os professores enfrentam com os alunos é saber a frequência que os alunos participam das suas aulas, pois há várias maneiras de o aluno marcar sua presença mesmo não estando em sala de aula. Pensando nisso, teve-se a ideia de criar um sistema que realizasse a chamada de uma turma através de um reconhecimento facial. Na Universidade de Brasília, a frequência miníma do aluno em uma matéria é de 75% e sua chamada é realizada de forma de assinatura, no qual cada aluno assina sua presença em uma folha. Esse sistema pode ser facilmente fraudado, pois um aluno pode assinar para um ou mais colegas de turma. Sabendo disso, alguns professores recorrem a chamada oral para tentar diminuir esse fraude, mas há casos de quando a turma é muito grande, o professor pode não conhecer todos os seus alunos e outra pessoa responder por ele, além de esse sistema perder um tempo considerável da aula.

Desta forma, teve-se a ideia de se criar um sistema que possa evitar ou diminuir a inassiduidade dos alunos, pois evitando as fraudes, os alunos terão que ter a presença miníma para ser aprovado na disciplina além de garantir que o aluno está presente na aula e obrigatoriamente interagindo com a matéria.

2. OBJETIVO

O objetivo deste projeto é criar um sistema que diminua a inassiduidade dos alunos através de um reconhecimento facial e sensor de temperatura infravermelho. Visando aumentar a frequência do alunos, uma menor probabilidade de evasão de alunos em uma instituição, além de ter um controle maior da instituição e dos pais [1].

3. REQUISITOS

Para a realização desse projeto, será construído um sistema inicialmente composto de um sensor de temperatura infravermelho, uma câmera e a Raspberry Pi B+. O sensor de temperatura sera usado como sistema de segurança para prevenir que se burle, com uma fotografia por exemplo, o sistema de detecção facial. Detectando a temperatura corporal validando assim que se trata de uma pessoa e não de uma fotografia. Já a câmera sera usada para identificar o aluno cadastrado no banco de dados correspondente à sua respectiva sala de aula, e assim podendo validar sua presença. Todo o sistema sera controlado pela Raspberry Pi.

4. BENEFÍCIOS

Com o emprego desse projeto será possível um maior controle de acesso dos alunos bem como a criação de um portfólio com esses respectivos dados que poderá ser usado para futuras analises. Podendo o sistema ser abrangido para diferentes setores correspondentes ao escopo do projeto.

5. DESENVOLVIMENTO

5.1. Descrição de Hardware

Para a realização deste projeto foi utilizada a seguinte lista de materiais:

Material	Quantidade
Raspberry Pi 3B+ (Micro-processador)	1
mlx90614esf (Sensor de temperatura)	1
Jumpers	4
Protoboard	1
Webcam	1
Estrutura do Sistema (A definir)	1

No hardware não houve basicamente quase nem uma alteração, a não ser na utilização da webcam no lugar do modulo pi câmera, pois a pi câmera adquirida veio com defeitos impossibilitando a sua utilização. O sistema em geral é composto por um sensor de temperatura infra-vermelho, uma câmera e a Raspberry Pi B+ (anexo 9.1). O sensor de temperatura é usado como sistema de segurança para prevenir que se burle, com uma fotografia por exemplo, o sistema de detecção facial, detectando a temperatura corporal e validando deste modo que se trata de uma pessoa e não de uma fotografia. Já a câmera será usada para identificar o aluno cadastrado no banco de dados correspondente à sua respectiva sala de aula, e assim podendo validar sua presença. Todo o sistema sera controlado pela Raspberry Pi.

O termômetro infravermelho da MLX90614esf foi desenvolvido para sensoriamento de temperatura sem contato. Um conversor ADC de 17-bits e um poderoso DSP contribuem para que este sensor tenha uma alta precisão e resolução. O MLX90614 possui dois métodos de saída: PWM e SMBus (ou seja, TWI, I2C). A saída PWM de 10-bit possui uma resolução de 0,14°C, enquanto que a interface TWI possui uma resolução de 0,02°C. Este sensor sai de fábrica calibrado em uma grande faixa de temperatura: -40 a 85°C para ambientes e -70 a 382,2°C para temperaturas de objetos. O valor medido é a média de temperaturas de todos os objetos do campo de visão do sensor. O MLX90614 oferece uma precisão padrão de 0,5°C em temperatura ambiente.

O sensor de temperatura possui um VCC, GND e os pinos SDA e SCL. Esses pinos fazem parte do protocolo de comunicação do sensor, esse protocolo de comunicação é o I2C, neste protocolo precisa dos dois pinos já citados, o SDA significa serial data e esse pino transfere todos os dados obtidos pelo sensor e o SCL significa serial clock e serve para a temporização entre os dispositivos, de modo que a comunicação pela SDA possa ter confiabilidade. Sabendo que os pinos 1, 3, 5 e 6 da raspberry pi são respectivamente 3,3 volts, SDA, SCL e ground, foi realizado a montagem conforme circuito em anexo 1.

Em relação a câmera escolhida para o projeto, como mencionado no ponto de controle 3, foi o modulo câmera pi 5MP. Porém o modulo comprado veio com defeito e foi devolvido, e não foi possível à aquisição de um novo módulo,foi usado então a webcam para a implementação do projet, que apesar de não ter a mesma qualidade de imagem do modulo que pretendemos usar, acreditamos que será suficiente para a validação. A webcam está representada na imagem do anexo 9.6.

5.2. Descrição de Software

Nesse ponto de controle foi desenvolvido o código os códigos presentes nos anexos 9.3, 9.4 e 9.5 para realizar a captura das fotos para o banco de dados, pra realizar o treinamento do software e para realizar o reconhecimento facial, respectivamente. Foi feito também uma calibração no sensor de temperatura, presente no anexo 9.2, para uma medição mais precisa.

A principal biblioteca usada para o funcionamento do sensor de temperatura foi a BCM 2835. Esta é uma biblioteca C para o Raspberry Pi que fornece acesso ao GPIO e outras funções de in / out no chip Broadcom BCM 2835, usado no RaspberryPi, permitindo acesso aos pinos GPIO no plugue IDE de 26 pinos da placa RPi para poder controlar e fazer a interface com vários dispositivos externos. Ela fornece funções para leitura de entradas digitais e configuração de saídas digitais, usando SPI e I2C, para acessar os timers do sistema. Vale salientar que quanto mais próximo do objeto o sensor de temperatura estiver, maior será sua precisão, pretendendo-se assim utilizar uma estrutura que favoreça essa maior precisão.

Para a execução do reconhecimento facial realizado com a câmera, foi realizada a biblioteca OpenCV (Open Source Computer Vision Library), que possui módulos de Processamento de Imagens e Video I/O, Estrutura de dados, Álgebra Linear, GUI (Interface Gráfica do Usuário) Básica com sistema de janelas independentes, Controle de mouse e teclado, além de mais de 350 algoritmos de Visão computacional como: Filtros de imagem, calibração de câmera, reconhecimento de objetos, análise estrutural e outros. O seu processamento é em tempo real de imagens. Para a realização do reconhecimento facial, dividiu-se o processo em três etapas: captura da imagem, processamento da imagem e posteriormente reconhecimento.

A primeira etapa consiste em capturar imagens do aluno matriculado e formar um banco de dados com essas imagens. Assim, tirou-se 30 imagens de cada aluno. Na segunda etapa, as imagens foram processadas com um método chamado Eigenfaces. Este método é uma otimização de um modelo matemático chamado de PCA(Análise de Componentes Principais) que descreve um conjunto de dados usando as principais componentes que representam da melhor maneira o conjunto de dados, reduzindo a dimensionabilidade dos dados, ou seja, quer obter as informações contidas nos dados usando menos dimensões. Desta forma, há um redimensionamento nas imagens e posteriormente é realizado um operação com os novos dados. A operação é feita seguindo estes passos:

- Monta-se uma matriz com os dados
- Calcula-se a média dos dados
- Subtrai a media na matriz de dados
- Calcula-se a matriz de covariância
- Calcula-se os valores Eigen e os valores Eigen da matriz de covariância

Pelo fato de usar os auto-vetores da matriz de covariância e usar dados de imagem de rosto que foi atribuído o nome Eigenface sendo que é similar ao método matemático PCA já citado, mas há uma otimização para reduzir a matriz de covariância, assim reduzindo o processamento necessário para fazer o calculo do auto-vetores e auto-valores. Desta forma, o método gera uma face media que será comparada com a imagem do aluno. Já a terceira etapa é o reconhecimento facial propriamente dito, a câmera fica em loop e quando o aluno aparece na imagem ele detecta a face e reconhece a pessoa.

6. RESULTADOS

Após vários testes com o sensor de temperatura foi possível realizar uma melhor calibração do mesmo chegando assim em um valor mais próxima da temperatura corpórea que desejamos aferir. A temperatura obtida após aferição foi de 35 graus Celsius, o que não é a temperatura ideal e nem muito diferente dos valores obtidos antes da calibração, mas nesses novos testes foi possível notar que a aferição variou muito pouco em relação aos valores obtidos no ponto de controle anterior, mas em relação a temperatura não ser ideal, notou-se que para que os valores estejam na faixa ideal, a distância do sensor com a face tem que ser muito pequena, assim influenciando na estrutura que teremos que fazer. Já em relação a câmera, foi possível realizar a detecção facial com o reconhecimento do aluno com um nível de exatidão considerável devido a baixa qualidade da câmera utilizada, um exemplo do reconhecimento está em anexo 7. Vale destacar que para a aquisição das imagens do banco de dados o ambiente tem que esta bem iluminado para uma maior precisão do algorítimo.

7. CONCLUSÃO

Em suma, com o algorítimo de temperatura e o de detecção facial, ambos funcionando, deve-se agora realizar a ligação entre os dois códigos, em seguida programar o algorítimo que irá realizar a escrita dos dados dos alunos que foram feitos os reconhecimentos faciais em um documento de texto e por fim realizar o embarque do protótipo em uma estrutura à se definir ainda.

8. REFERENCIAS

[1] https://oglobo.globo.com/brasil/sistema-digital-avisara-pais-sobre-presenca-de-alunos-em-escola-do-interior-de-sp-3138190
[2] SHAH, SAMARTH (2015) Ĺearning Raspberry Pi, 1ªedição, ISBN 9781783982820. Editora Packt Publishing Ltd.
[3] https://medicoresponde.com.br/qual-e-a-temperatura-normal-do-corpo-humano/
[4] https://www.learnopencv.com/principal-component-analysis/
[5] http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=4466
[6] https://opencv.org/
[7]https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/

9. ANEXOS

9.1. Anexo 1

Imagem do esquemático do circuito feito no Fritzing:

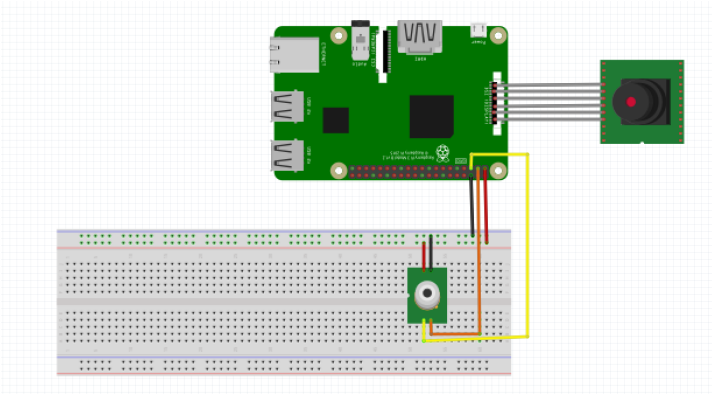

Fig. 1. Fluxo esquemático do sistema

9.2. Anexo 2

Código para o teste do sensor de temperatura:

```
#include <stdio.h>
#include <bcm2835.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <time.h>
#define AVG 1   //averaging samples


int main(int argc, char **argv)
{
    unsigned char buf[6];
    unsigned char i,reg;
    double temp=0,calc=0, skytemp,atemp;
    bcm2835_init();
    bcm2835_i2c_begin();
    bcm2835_i2c_set_baudrate(25000);
    // set address
    bcm2835_i2c_setSlaveAddress(0x5a);
```

```
    printf("\ndevice is working!!\n");

    calc=0;
    reg=7;

    for(i=0;i<AVG;i++)
{
bcm2835_i2c_begin();
bcm2835_i2c_write (&reg, 1);
bcm2835_i2c_read_register_rs(&reg,&buf[0],3);
temp = (double) (((buf[1]) << 8) + buf[0]);
temp = (temp * 0.02)-0.01;
temp = temp - 273.15;
calc+=temp;
sleep(1);
}

    skytemp=calc/AVG;


    if(skytemp > 33){

    printf("É Humano\n");
    }

    else{

    printf("Não é humano\n");
    }

    }

    return 0;
}
```

9.3. Anexo 3

Código para a captura de foto:

```
import cv2
import numpy as np

classificador = cv2.CascadeClassifier("
haarcascade-frontalface-default.xml")
classificadorOlho = cv2.CascadeClassifier("
haarcascade-eye.xml")
camera = cv2.VideoCapture(0)
amostra = 1
numeroAmostras = 30
id = input('Digite seu ID: ')
largura, altura = 220, 220
print("Capturando os rostos...")

while (True):
    conectado, imagem = camera.read()
    imagemCinza = cv2.cvtColor(imagem,
    cv2.COLOR_BGR2GRAY)
    #print(np.average(imagemCinza))
    facesDetectadas =
    classificador.detectMultiScale(
        imagemCinza,
        scaleFactor=1.5,
        minSize=(150,150))

    for (h, j, k, l) in facesDetectadas:
     cv2.rectangle(imagem, (h, j), (h + k, j
     + l), (0, 0, 255), 2)
     if cv2.waitKey(1) & 0xFF == ord('q'):
         #if np.average(imagemCinza) > 110:
             imagemFace = cv2.resize(
             imagemCinza[j:j + l, h:h + k], (
             largura, altura))
             cv2.imwrite("fotos/pessoa."+str(id)
             + "." + str(amostra) + ".jpg",
             imagemFace)
             print("[foto" + str(amostra) + "
             capturada com sucesso]")
             amostra += 1
    cv2.imshow("Face", imagem)
    cv2.waitKey(1)
    if (amostra >= numeroAmostras + 1):
        break
print("Faces capturadas com sucesso!")
camera.release()
cv2.destroyAllWindows()
```

9.4. Anexo 4

Código para o treinamento:

```
import cv2
import os
import numpy as np


eigenface =
cv2.face.EigenFaceRecognizer_create()

def getImagemComId():
 caminhos = [os.path.join('fotos', f)
 for f in os.listdir('fotos')]
```

```
 faces = []
 ids = []
 for caminhoImagem in caminhos:
     imagemFace = cv2.cvtColor(
     cv2.imread(caminhoImagem),
     cv2.COLOR_BGR2GRAY)
     id = int(os.path.split(caminhoImagem)
     [-1].split('.')[1])
     ids.append(id)
     faces.append(imagemFace)
 return np.array(ids), faces
ids, faces = getImagemComId()
print(ids)

print("treinando...")
eigenface.train(faces, ids)
eigenface.write('
classificadorEigen.yml')

print("Treinamento realizado
com sucesso!")
```

9.5. Anexo 5

Código para o reconhecimento facial:

```
import cv2
import os
import numpy as np


eigenface =
cv2.face.EigenFaceRecognizer_create()

def getImagemComId():
 caminhos = [os.path.join('fotos', f)
 for f in os.listdir('fotos')]
 faces = []
 ids = []
 for caminhoImagem in caminhos:
     imagemFace = cv2.cvtColor(
     cv2.imread(caminhoImagem),
     cv2.COLOR_BGR2GRAY)
     id = int(os.path.split(caminhoImagem)
     [-1].split('.')[1])
     ids.append(id)
     faces.append(imagemFace)
 return np.array(ids), faces
ids, faces = getImagemComId()
print(ids)
print("treinando...")
eigenface.train(faces, ids)
eigenface.write('classificadorEigen.yml')
print("Treinamento realizado com sucesso!")
```

9.6. Anexo 5

Imagem da webcam utilizada:

Fig. 2. Imagem webcam LG

9.7. Anexo 6

Imagem do reconhecimento facial:

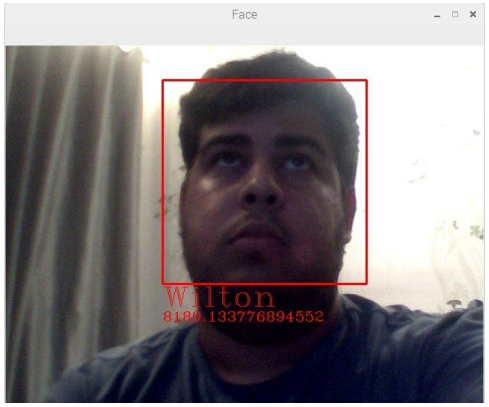

Fig. 3. Reconhecimento facial

9.8. Anexo 7

Imagem do sensor de temperatura utilizado:

Fig. 4. Sensor de temperatura

9.9. Anexo 8

Fluxograma do sistema:

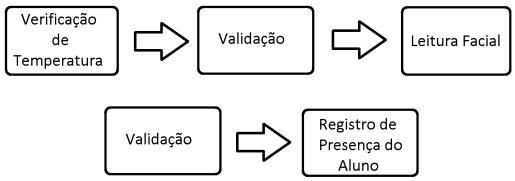

Fig. 5. Fluxo esquemático do sistema