Sistema de monitoramento com reconhecimento facial e fala utilizando Raspberry PI

Pedro da C. Cason, Pedro H. de A. Costa, Dra. Daniela V. Cunha

Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie (UPM)
São Paulo, SP – Brazil.

{31708854, 31716997}@mackenzista.com.br,
daniela.cunha@mackenzie.com.br

Abstract. *This article presents the development of a monitoring system for personal use with speech and face recognition, built using a Raspberry Pi single-board microcomputer, with a camera and a microphone. The project aims to demonstrate that it is possible to produce a low-cost monitoring system, being possible to obtain facial and speech recognition with Python programming and some libraries, as well as sending automatic notifications via email in case if pre-determined keywords are detected or a pre-registered face is recognized by the system. So, the monitoring of an environment is carried out with different purposes, such as the surveillance of the elderly and/or children, in order to avoid accidents. Initial tests showed that the system is capable of successfully detecting and recognizing registered faces, as well as identifying and recognizing the word "Help", sending a notification email to a pre-defined email.*

Keywords: *monitoring system; facial and speech recognition; automatic notifications.*

Resumo. *Este artigo apresenta o desenvolvimento de um sistema de monitoramento destinado ao uso pessoal com reconhecimento de fala e face, construído utilizando um microcomputador de placa única Raspberry Pi, em conjunto com uma câmera e um microfone. O projeto tem como objetivo demonstrar que é possível produzir um sistema de monitoramento de baixo custo, sendo possível obter reconhecimento facial e de fala com programação Python e algumas bibliotecas, assim como o envio de notificações automáticas via e-mail caso palavras-chave pré-determinadas sejam detectadas ou uma face pré-cadastrada seja reconhecida pelo sistema. Desta forma realiza-se o monitoramento de um ambiente com diferentes finalidades, como a vigilância de idosos e/ou crianças, em prol de evitar acidentes. Testes iniciais mostraram que o sistema é capaz de detectar e reconhecer com sucesso as faces cadastradas, bem como identificar e reconhecer a palavra "Socorro", enviando um e-mail de notificação ao e-mail pré-definido.*

Palavras-chave: *sistema de monitoramento; reconhecimento facial e de fala; notificações automáticas.*

1. Introdução

A tecnologia está cada vez mais presente na vida das pessoas. O que antes era considerado como "luxo" ou algo interessante, hoje é indispensável e muitas vezes obrigatório. Entre essas tecnologias, estão os computadores pessoais, os celulares e até mesmo os dispositivos IoT (*Internet of Things*), como lâmpadas e eletrodomésticos controlados pela internet.

Com os avanços da tecnologia, uma preocupação constante é a de monitoramento e segurança. Com inúmeros propósitos, um sistema de monitoramento, hoje, pode colaborar com a segurança de estabelecimentos bem como realizar o acompanhamento de idosos e crianças em ambiente residencial, por exemplo.

Todos os anos, cerca de 5 mil crianças vêm a óbito e outras 110 mil são hospitalizadas devido a acidentes domésticos, sem a supervisão de um responsável [Sociedade Brasileira de Pediatria 2018]. Da mesma forma, cerca de 4,3 milhões de idosos moram sozinhos no Brasil [G1 2020]. Dependendo da idade e de doenças comuns relativas aos idosos, e crianças que passam o dia sozinhas em casa, são necessários cuidados específicos e uma maior vigilância, que um sistema de monitoramento poderia auxiliar em diversas situações.

O objetivo geral deste projeto é o desenvolvimento de um sistema de monitoramento utilizando um microcomputador *Raspberry Pi*, com uma *PiCamera* e um microfone USB focado em uso doméstico, para ajudar a monitorar ambientes e ter o controle de quem esteve no cômodo monitorado. Em caso de acidentes, ao detectar a palavra "socorro", o sistema notifica por e-mail com uma mensagem os usuários que possuem a conta configurada. No caso de pessoas que entram no ambiente e são reconhecidas pelo monitoramento, o sistema envia outro e-mail, este com uma imagem tirada no momento da detecção facial, e a mensagem no corpo.

Para que a criação do projeto fosse possível, foram estudados os conceitos de *IoT*, reconhecimento facial e reconhecimento de fala. As bibliotecas do *Python* utilizadas foram: *VideoStream, face_recognition, speech_recognition, pickle, time e cv2*, e bibliotecas de integração com e-mail, MIME *e Smtplib*, que foram essenciais para o funcionamento do projeto.

Na seção 2 deste artigo, é apresentado o referencial teórico abordando tópicos importantes no contexto do trabalho, como Reconhecimento Facial, *Haar Cascade*, Histograma de Gradientes Orientados e Reconhecimento de Fala; a seção 3 apresenta o Desenvolvimento do Projeto, onde são abordados os materiais e métodos utilizados para seu correto funcionamento; a seção 4 apresenta os resultados obtidos e a seção 5 apresenta a conclusão do artigo.

2. Referencial Teórico

O referencial teórico do projeto apresentado neste artigo aborda conceitos importantes para entendimento do projeto, como o reconhecimento facial e as tecnologias necessárias para o funcionamento do sistema de monitoramento, assim como o reconhecimento de fala e sua implementação.

2.1. Reconhecimento Facial

Reconhecimento facial é o termo utilizado para a identificação de faces de pessoas por meio de vídeos ou imagens. Largamente utilizada em sistemas de segurança e monitoramento sofisticados, ficou mais acessível ao longo do tempo com a utilização em hardwares mais simples e projetos menores. Seu uso hoje é possível em dispositivos IoT, laptops e até mesmo celulares [AWS 2020].

Alguns trabalhos apresentaram formas diferentes para realizar a tarefa de reconhecimento facial, como o uso do LBP (*Local Binary Pattern*), que em alguns casos é mais rápido para o reconhecimento facial, porém é menos preciso. No entanto, a utilização em conjunto dos algoritmos *Haar Cascade* e *Histogram of Oriented Gradients* se mostrou muito eficiente e confiável para uso no Raspberry Pi. Ambos são apresentados a seguir."

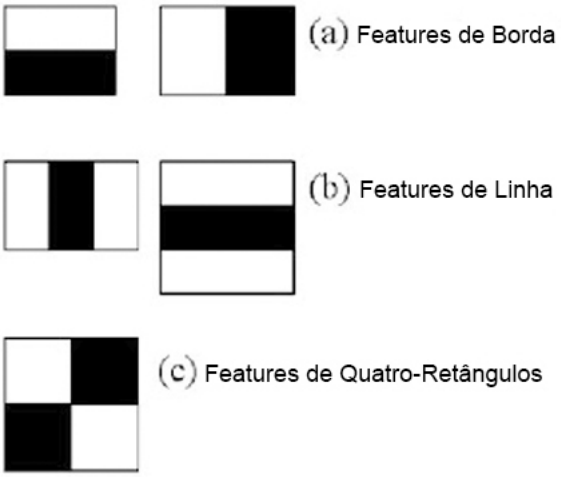
2.1.1. *Haar Cascade*

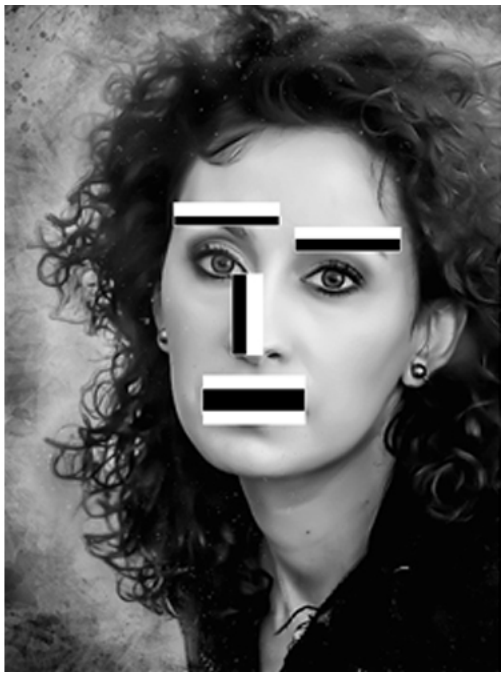
Haar Cascade é um algoritmo de detecção de objetos utilizado normalmente para identificar faces em imagens ou vídeos em tempo real. Este algoritmo foi criado por Viola e Jones [Behera 2020]. Sua utilização pode ser variada, para o reconhecimento não apenas de faces, mas também de objetos, carros e placas. Neste projeto, seu uso será apenas para o reconhecimento de rostos.

O uso do algoritmo para as faces inicialmente é constituído de um grande banco de imagens, chamadas de positivas e negativas [Medium 2020]. As imagens denominadas positivas são formadas pelos rostos de diversas pessoas, para que seja possível verificar os pontos em comum (todas as faces sem deficiências ou outros problemas possuem olhos, boca e nariz). As imagens negativas são quaisquer outras imagens, que não contenham nenhum elemento que deverá ser identificado. Neste caso, essas imagens não podem conter faces. A figura 1 apresenta exemplos de imagens positivas e negativas, utilizadas no processo de treinamento de modelos de detecção facial.

As imagens positivas e negativas provêm de um modelo para identificação de faces já existente, no formato *XML* (*eXtensible Markup Languag*e) utilizado para arquivos codificados. São necessárias fotos de várias pessoas e vários objetos para constituir este banco. Porém, essas informações já existem de forma padrão na internet, sendo necessário apenas utilizá-las junto ao *OpenCV (Open Source Computer Vision Library)*, biblioteca de visão computacional que funciona com a linguagem de programação *Python* e é compatível com os algoritmos de reconhecimento facial, como o próprio *Haar Cascades* explicado nesta seção, e o HOG*, Histogram of Oriented Objects,* algoritmo que será apresentado na seção 2.1.2.

O *Haar* também possui classificadores, que são utilizados nas imagens das faces que são detectadas, posicionados sobre as partes do rosto que o caracterizam, como boca, nariz e olhos, conforme mostra a Figura 2. As *features* são definidas como um valor numérico encontrado por meio da relação da intensidade dos pixels. Esse cálculo é realizado utilizando o conceito de imagem integral, melhorando o desempenho de processamento. As *features* de borda (a) procuram elementos da face que são horizontais ou verticais, com valores de referência dos pixels 0 e 1, com uma margem de valores e diferenças [Medium 2019].





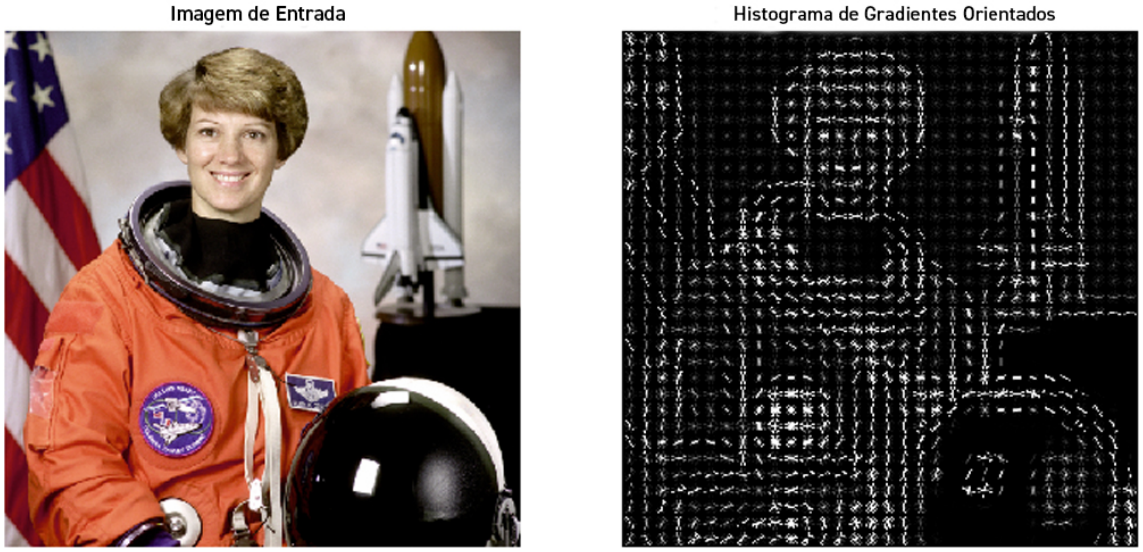
As *features* de linha (b) são utilizados em elementos onde a diferença de pixels brancos e pretos é 0, 1 e 0 novamente, como na boca de uma pessoa, onde antes do lábio e abaixo do lábio os pixels são mais claros. Já as *features* de "quatro-retângulos" (c) procuram por linhas diagonais, como linhas faciais externas, por exemplo.

Desta forma, torna-se possível no projeto realizar a identificação das faces que aparecem na câmera. Porém, ainda não há a distinção de faces, ou seja, a diferenciação da pessoa A de B. É necessário então o uso do HOG (*Histogram of Oriented Gradients*).

2.1.2. Histograma de Gradientes Orientados

O Histograma de Gradientes Orientados, chamado de HOG, é um descritor de recurso proposto por Dalal e Triggs em 2005, que representa imagens de forma simplificada, ignorando outras informações irrelevantes e identificando a direção das linhas nos pixels [Analytics Vidhya 2019]. O HOG conta o número de ocorrências (histogramas) dos gradientes de orientação para pequenas porções da imagem.

Este descritor, em conjunto com o *Haar Cascade*, permite que a face detectada possa ser reconhecida e diferenciada entre as faces cadastradas. O *Haar* identifica os olhos, boca e nariz, identificando que se trata de uma face, enquanto o HOG identifica as bordas e direções, ou seja, a silhueta da pessoa, entendendo o formato completo da face e assim gerando um reconhecimento mais confiável. A Figura 3 demonstra o histograma de gradientes orientados completo (à direita), realizado a partir da imagem de entrada, à esquerda.



Para obter o resultado à direita na Figura 3, é necessário obter o gradiente da imagem, um vetor que indica a variação dos níveis de cinza e a direção dessa escala. Este vetor gradiente é calculado utilizando as posições x e y da imagem (de acordo com sua altura e largura em pixels) com a seguinte equação [Santos and Carneiro 2020]:

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{bmatrix}.$$

$\frac{\partial f}{\partial x}$

$\frac{\partial f}{\partial y}$

diferença a cor entre os pixels adjacentes acima e abaixo do alvo. O resultado obtido é o vetor gradiente [Weng 2017]. Após a descoberta deste valor, é necessário descobrir a magnitude do gradiente (o quão rápido a imagem muda) e sua angulação, que resulta nas direções das linhas. A magnitude pode ser calculada pela equação:

$$g = \sqrt{g_x^2 + g_y^2}.$$

E o ângulo do gradiente pode ser obtido com:

$$\theta = \arctan\left(g_y / g_x\right).$$

Com estes dados, é possível realizar o pré-processamento da imagem. É necessário que a proporção seja 1:2, ou seja, se a imagem tiver seu tamanho (em pixels) 1080 x 1920, é necessário que ela seja redimensionada para 1000x2000. Além disso, em alguns casos, o descritor funciona de forma mais correta se a gama da imagem for transformada em preto e branco [Analytics Vidhya 2019].

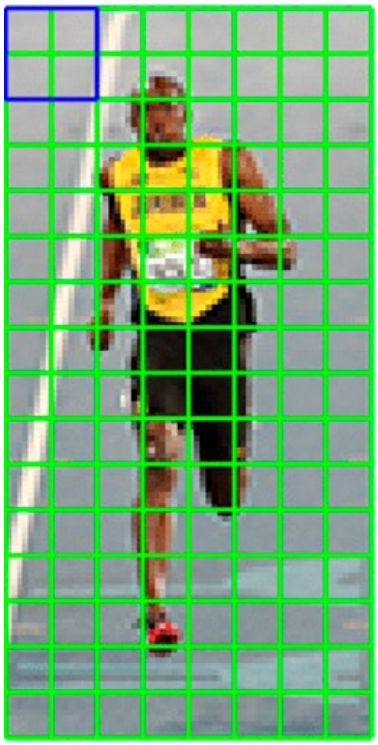


Figura 4 - Células demonstradas na imagem (quadriculados), onde serão calculadas.
Fonte: Mallick (2016).

A Figura 4 demonstra como são realizados os cálculos em pequenas seções da imagem, chamados de células. E o histograma calculado obtido para cada uma gera o histograma de gradientes orientados, com seu resultado sendo representado na Figura 3. Dessa forma, obtém-se a silhueta das faces que serão reconhecidas.

2.2. Reconhecimento de fala

Reconhecimento de fala é a capacidade de um programa de processar a voz humana falada e gerar um texto escrito [IBM 2020]. O texto escrito gerado é usado para que palavras específicas possam ser identificadas e consequentemente servem de gatilho para o acionamento de funções utilizando linguagens de programação. A Figura 5 representa o funcionamento deste processo.

Figura 5 - Como o reconhecimento de fala funciona. Fonte: TowardsDataScience (2019).

No projeto, o reconhecimento de fala foi utilizado para detectar a palavra "Socorro", configurada para que caso ouvida, o software processe o áudio e envie uma notificação por e-mail para contatos pré-definidos com uma mensagem de socorro em seu corpo [Let's Code 2019].

Para realizar essa transcrição do áudio para o texto, utilizou-se a biblioteca do Python denominada *Speech_Recognition*. Esta biblioteca faz integração com a biblioteca *Speech-to-Text,* que é uma biblioteca de reconhecimento de fala criada e mantida pela Google.

Não há a necessidade de treinamento para tipos de sotaque diferentes, já que a Google trabalha há anos com a língua Portuguesa e suas variantes dialéticas. Depois da transcrição de áudio para texto, o Python entende as letras dispostas como *Strings* (sequência de letras) e consegue assimilar que a palavra falada tem o mesmo significado da palavra "Socorro" escrita.

3. Desenvolvimento do Projeto

Neste item, descreve-se a arquitetura do projeto, assim como as tecnologias utilizadas na parte de hardware e software.

Para o monitoramento facial, existem três etapas necessárias para seu funcionamento, que serão explicadas ao longo desta seção:

- Obtenção grupos de imagens faciais de todas as pessoas que serão reconhecidas para formar um *dataset*;
- Realização do treinamento dos algoritmos utilizando as imagens das pessoas para que seja possível realizar o reconhecimento;
- Utilização do monitoramento facial final, ativo a todo momento e com a integração com e-mail para notificações com imagens;

No monitoramento de fala, não são necessários passos anteriores para o funcionamento. O software fica ativo a todo momento e notifica por e-mail usuários pré-configurados caso a palavra "Socorro" seja ouvida no ambiente, com texto e sem imagem.

3.1. Formação do Dataset

A primeira etapa para o funcionamento do monitoramento facial é a criação de um banco de imagens de cada pessoa que poderá ser reconhecida pelo programa. O primeiro software do projeto, desenvolvido em Python, tem a função de tirar várias fotos de uma pessoa, utilizando uma *PiCamera* que funciona no próprio *Raspberry Pi*.

No código, deve-se digitar o nome da pessoa a ser cadastrada, e então o programa deve ser iniciado. A câmera ficará ativa e ao apertar a barra de espaço no teclado, serão tiradas as fotos. O usuário pode tirar quantas fotos desejar, e ao terminar, é necessário apertar a tecla "esc" do teclado. Uma nova pasta com o nome da pessoa será criada, e as imagens serão automaticamente transferidas para este novo diretório, que fica no mesmo caminho dos programas do projeto, salvos no cartão de memória.

Assim, este banco de imagens, chamado de "*Dataset*", possui o material necessário para que as faces detectadas sejam comparadas com as cadastradas e possam ser ou não reconhecidas de acordo com o *dataset*. A biblioteca Python utilizada para cadastrar o *Dataset* é a "*PiCamera*", que habilita o uso específico da *PiCamera*, demonstrada na Figura 6, e conectada ao *Raspberry Pi*.

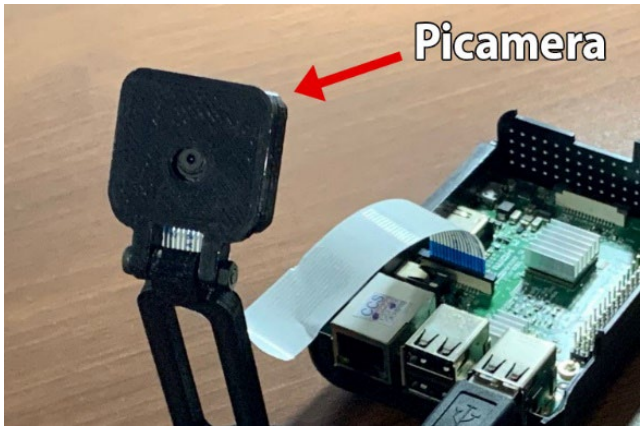


Figura 6 - *PiCamera* instalada no *Raspberry Pi*. Fonte: elaborado pelos autores (2021).

3.2. Treinamento das Faces

Com as faces cadastradas no *Dataset*, é necessário que as fotos sejam utilizadas pelos algoritmos para seu treinamento, ou seja, a análise de todas as imagens em que o sistema "aprende" a reconhecer as faces cadastradas. Este segundo software, também desenvolvido em Python, é responsável por analisar as faces utilizando o algoritmo HOG e as bibliotecas utilizadas foram a *"cv2",* que habilita o OpenCV para o processamento de imagem e a *face_recognition*, uma sub-biblioteca do "cv2" que efetivamente usa os algoritmos para o reconhecimento de faces. Também é utilizada a biblioteca *Pickle*.

A biblioteca *Pickle* é responsável pela serialização (realiza a tradução da estrutura de um arquivo para um formato que pode ser armazenado como bytes) e desserialização (realiza o inverso da serialização) da estrutura de um objeto dentro do *Python*. Essa biblioteca, no software para treinamento das faces, gera um arquivo com os dados que serão lidos pelo software de monitoramento e que permitem a interpretação do que foi treinado, para que seja possível monitorar as faces que os algoritmos passaram a conhecer.

3.3. Monitoramento facial

Com o *Dataset* montado e os algoritmos treinados para reconhecer as faces, o software de monitoramento passou a ser funcional, podendo reconhecer as pessoas que já foram cadastradas. Neste programa, são utilizadas as bibliotecas *face_recognition*, para realizar o reconhecimento com os algoritmos, *pickle*, utilizada para obter as informações das faces treinadas, e a *cv2* para o processamento dos dados.

Também foram utilizadas as bibliotecas *VideoStream*, necessária para rodar a câmera e mantê-la funcionando a todo momento. A *Time*, para marcar em qual horário a foto foi tirada (a foto é salva localmente com a data e hora), e as bibliotecas *MIME* e *Smtplib*, com foco nas notificações por e-mail. A biblioteca *MIME* é utilizada para permitir anexos com fotos, e a *smtplib* é responsável pela configuração do remetente e destinatário.

Seu funcionamento é da seguinte forma: a câmera, sempre ativa, procura por faces. Caso apareça na câmera, o algoritmo *Haar Cascade* procura pelos pontos em comum de qualquer face, como olhos, boca, nariz e outras linhas como maxilar e testa. Então, em caso de se tratar de uma face, o *HOG* faz uma análise com gradientes (Figura 3), e compara com todas as pessoas pré-cadastradas, realizando assim o reconhecimento da face. Caso o reconhecimento não seja realizado, o programa não realiza nenhum tipo de intervenção, conforme apresentado na Figura 7.

Figura 7 - Fluxograma de funcionamento do reconhecimento facial. Fonte: elaborado pelos autores (2021).

Na Figura 8, é possível observar o fluxo de funcionamento completo do sistema de monitoramento facial, com seu funcionamento inicial, reconhecimento e notificações:

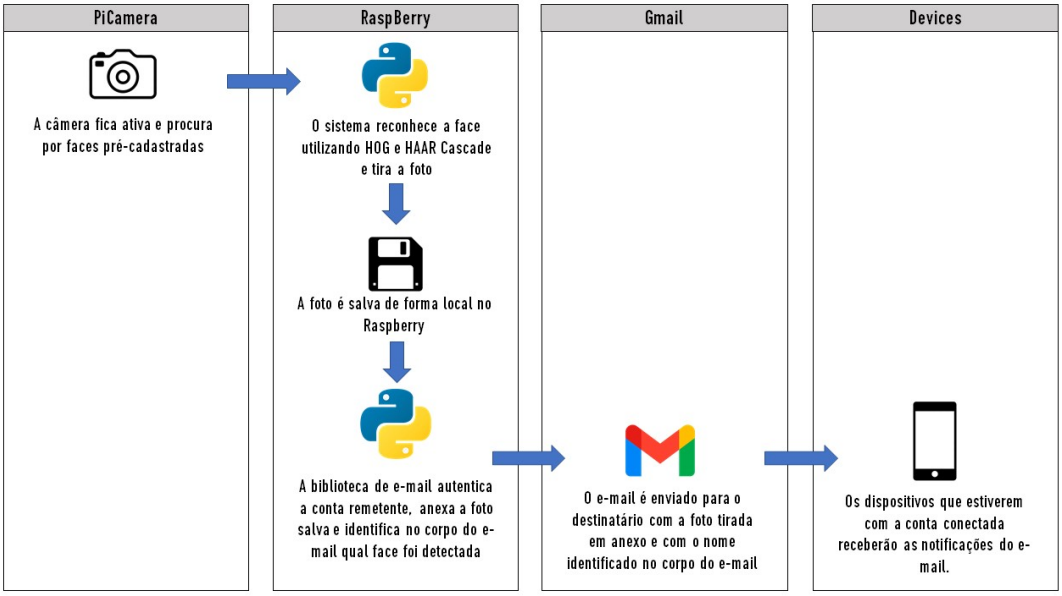


Figura 8 - Funcionamento do Monitoramento Facial. Fonte: elaborado pelos autores (2021).

3.4. Reconhecimento de fala

O segundo projeto, que fica separado do software de monitoramento facial, é o programa de reconhecimento de fala, que também fica ativo o tempo todo e foi desenvolvido em Python. Seu fluxo de funcionamento pode ser visualizado na Figura 9:

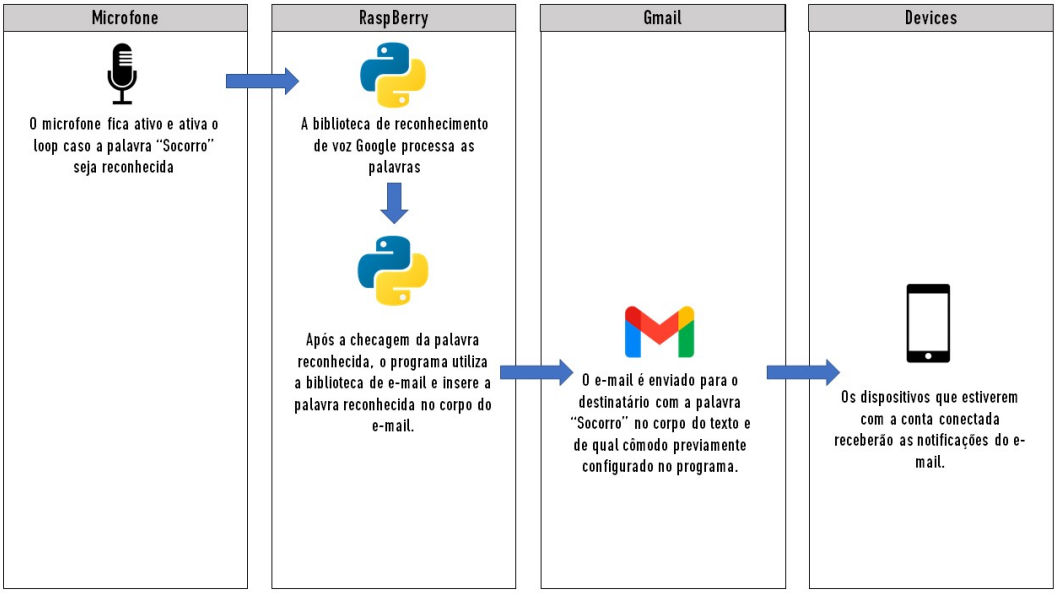


Figura 9 - Funcionamento do Reconhecimento de fala. Fonte: elaborado pelos autores (2021).

3.5. Notificações

Tanto para o sistema de monitoramento quanto para o sistema de detecção de fala, existem as notificações dos eventos; seja para o reconhecimento de uma pessoa; seja para a detecção da palavra socorro. A ferramenta escolhida para essas notificações é o e-mail.

A sua escolha se deve a simplicidade de configuração, sem a necessidade de grandes integrações e *APIs*, e com a facilidade de ser possível realizar a configuração de uma conta de e-mail em diversos dispositivos, sejam computadores ou celulares. Foi criada uma conta padrão utilizando o cliente Gmail, onde os softwares estão parametrizados para que a conta criada seja o remetente. O destinatário pode ser configurado para qualquer outra conta de e-mail.

A biblioteca *smtplib* permite a configuração de um remetente e destinatário, e o envio de e-mails com título e descrição a partir de um projeto *Python.* Mesmo assim, ainda não é possível adicionar anexos. As notificações de reconhecimento facial necessitam da utilização dos anexos para incluir a imagem tirada da pessoa reconhecida. Neste caso, foi utilizada a biblioteca *MIME*. Esta biblioteca permite que os e-mails enviados pelo *Python* possam incluir anexos, como imagens, para uso junto do sistema [Python Docs 2021]. Dessa forma, é possível obter notificações em diversos dispositivos em que a conta destinatária esteja configurada.

3.6 *Hardware*

Figura 10 - *Hardware* do projeto montado. Fonte: elaborado pelos autores (2021).

Neste item, serão indicados os componentes utilizados para o funcionamento total do projeto. O *hardware* montado e demonstrado na Figura 10 consiste em: uma placa microcontroladora *Raspberry Pi*, modelo de terceira geração com 1GB de memória *RAM*

e processador *BroadCom* com 4 núcleos. O armazenamento do dispositivo utiliza um cartão *MicroSD* de 16GB e é alimentado por uma fonte 5v.

Os periféricos que são necessários para o projeto são: um microfone USB, que não necessita ser um modelo específico, e uma câmera, que pode ser tanto uma *Webcam*, ou no caso específico do *Raspberry Pi*, uma *PiCamera*, ligada diretamente à placa e é o modelo utilizado no projeto.

4. Resultados

A seguir, são demonstrados os resultados obtidos com o projeto em funcionamento, posicionado em um quarto de uma residência, realizando o monitoramento do ambiente. O sistema faz o reconhecimento facial de quem entra no ambiente, envia uma notificação para a conta de e-mail configurada com a foto tirada em anexo. Da mesma forma, também será demonstrado o sistema de reconhecimento de fala, posicionado no mesmo ambiente para reconhecer a palavra “Socorro”.

Os testes iniciais foram realizados utilizando o cadastro de 4 pessoas diferentes que moram em uma mesma casa, sem o cadastro de pessoas externas, também por conta da pandemia. Os *datasets* foram montados com fotos tiradas pela própria *Picamera* em um ambiente com luminosidade controlada. Após os cadastros, o monitoramento foi iniciado e as pessoas passaram pela câmera, tanto de forma separada, quanto em um dos testes, realizado com duas pessoas ao mesmo tempo.

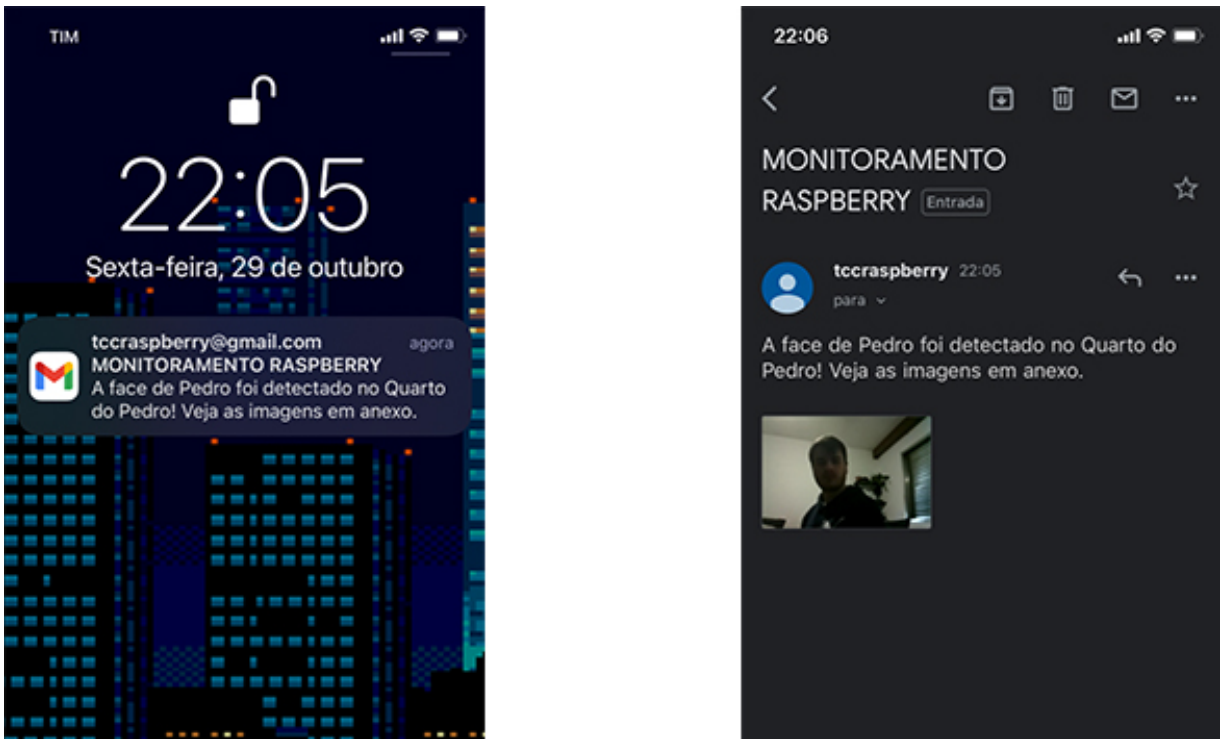


Figura 11 - Recebimento de notificação do monitoramento facial. Fonte: elaborado pelos autores (2021).

Como observado na Figura 11, ao reconhecer uma face, o sistema de monitoramento envia um e-mail ao destinatário com a imagem em anexo e a descrição de qual pessoa foi reconhecida, o cômodo que ela está e o e-mail recebido.

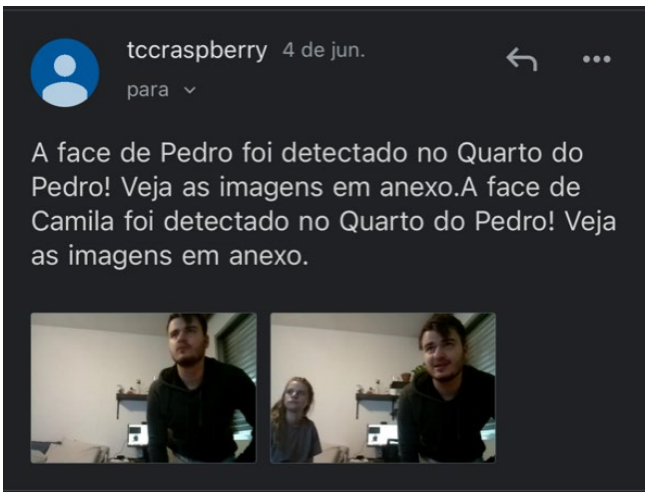


Figura 12 - Recebimento de notificação do monitoramento facial. Fonte: elaborado pelos autores (2021).

A Figura 12 demonstra o resultado do teste realizado com duas pessoas ao mesmo tempo. O sistema foi capaz de reconhecer e identificar corretamente as pessoas que apareceram na câmera. Entre os testes realizados, não houve inconsistências no reconhecimento, sem a existência de falsos-positivos ou o reconhecimento incorreto.

Já no monitoramento para detecção de voz, o funcionamento é parecido: no cômodo de uma residência, com o microfone conectado ao *Raspberry Pi*, ao detectar a palavra "Socorro" no ambiente, é enviada uma notificação para o mesmo e-mail, com a descrição de que a palavra foi ouvida no ambiente, conforme a Figura 13:

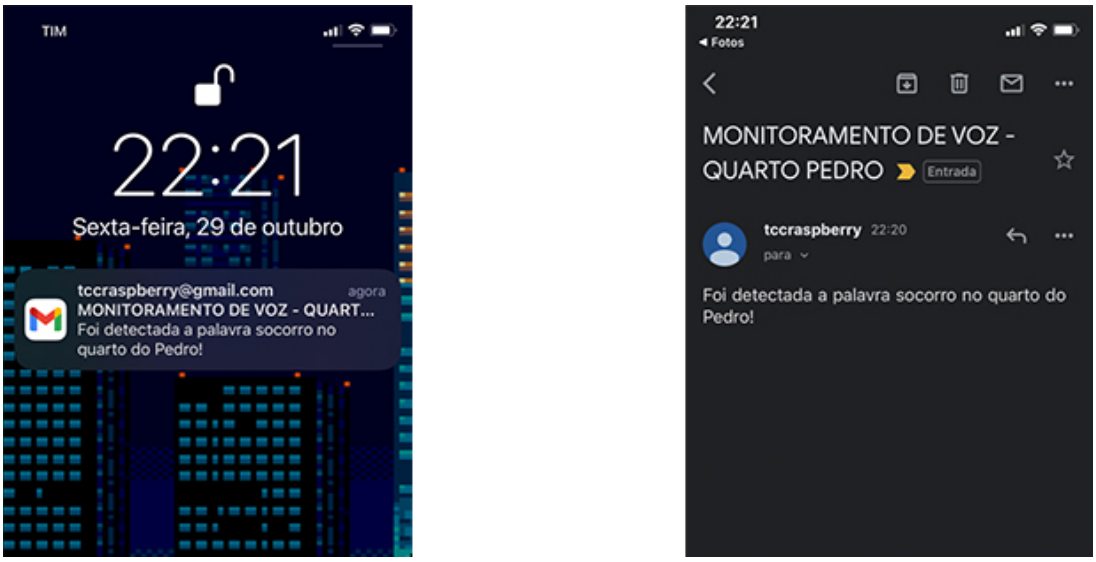


Figura 13 - Recebimento de notificação do monitoramento de voz. Fonte: elaborado pelos autores (2021).

Para validar a detecção de voz, foram realizados testes com a fala de outras palavras, não relacionadas a palavra “Socorro”, assim como homônimos, que podem confundir o sistema. Nos resultados, falando de forma clara e com som mais alto a palavra que gera a notificação, o reconhecimento sempre foi correto, não gerando erros ou uma notificação incorreta.

5. Conclusões

O projeto apresentado neste artigo possui ferramentas funcionais para realizar o monitoramento de crianças e idosos em ambientes residenciais, por exemplo, com o funcionamento do reconhecimento facial e o reconhecimento de fala, notificando por e-mail contas pré-cadastradas e assim podendo evitar acidentes e ajudar na definição de ações rápidas que podem auxiliar em situações de controle e perigo.

O projeto é relativamente fácil de ser mantido e não tem um alto custo. Sendo assim, pode ser instalado e configurado em mais de um cômodo, com as mesmas funcionalidades e abrangendo mais locais, para maior segurança e área de monitoramento mais amplo.

Para ser utilizado por outras pessoas, é necessário apenas ter conexão à internet com wireless, já que todo o processo de reconhecimento de voz e fala é realizado diretamente no *Raspberry Pi* de forma local e a conexão à internet é necessária para o envio das notificações. Para o cadastro das pessoas a serem reconhecidas, é necessário apenas que sejam tiradas fotos dos rostos utilizando o programa de cadastro de dataset do projeto.

Visando melhorias futuras, o projeto pode ser atualizado com uma câmera de mais qualidade, que pode ser desde a melhoria do sensor de imagem até uma câmera IP, que tenha visão noturna, por exemplo, e possa realizar o monitoramento com mais precisão. Além disso, o projeto pode sofrer incrementos que possibilitem que até mesmo quem não está cadastrado possa gerar uma notificação, expandindo as possibilidades de monitoramento para um foco maior em segurança.

Referências

Adakane, D. (2019) “What are Haar Features used in Face Detection?”, https://medium.com/analytics-vidhya/what-is-haar-features-used-in-face-detection-a7e531c8332b.

Behera, G. S. (2020) “Face Detection with Haar Cascade”, https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08.

Dalal, I. and Triggs, B. (2007) “Histograms of Oriented Gradients for Human Detection”, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://lear.inrialpes./fr/people/triggs/pubs/Dalal-cvpr05.pdf.

Dunn, C. (2020) “How to Train yout Raspberry PI for Facial Recognition”, https://www.tomshardware.com/how-to/raspberry-pi-facial-recognition.

Grandin, F. and Rodrigues, M. (2020) “Brasil tem 4,3 milhões de idosos vivendo sozinhos; coronavírus muda rotinas e impõe desafios”, https://g1.globo.com/fique-em-casa/noticia/2020/03/27/brasil-tem-43-milhoes-de-idosos-vivendo-sozinhos-coronavirus-muda-rotinas-e-impoe-desafios.ghtml.

Github (2018) “Haar Cascade Classifier”, https://jmlb.github.io/flashcards/2018/06/30/computer_vision_haar_cascade/.

IG Último Segundo (2021) “Menino de 4 anos morre em incêncio após ser deixado sozinho com vela acesa”, https://ultimosegundo.ig.com.br/brasil/2021-08-12/sp--menino-de-4-anos-morre-em-incendio-apos-ser-deixado-sozinho-com-vela-acesa.html.

Khan, T. (2019) “Computer Vision — Detecting objects using Haar Cascade Classifier”, https://towardsdatascience.com/computer-vision-detecting-objects-using-haar-cascade-classifier-4585472829a9#:~:text=Haar%20Cascade%20classifier%20is%20an,of%20Simple%20Features%E2%80%9D%20in%202001.&text=Based%20on%20the%20training%20it,objects%20in%20the%20other%20images.

Letscode (2019) “Speech Recognition com Python”, https://letscode.com.br/blog/speech-recognition-com-python.

Mallick, S. (2016) “Histogram of Oriented Gradients explained using OpenCV”, https://learnopencv.com/histogram-of-oriented-gradients/

OpenCV. (2021) “About OpenCV”, https://opencv.org/about/.

OpenCV. (2021) “Cascade Classifier”, https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html.

Python. (2021) “Creating MIME E-mail Objects”, https://docs.python.org/pt-br/3.7/library/email.mime.html.

Santos, M., Carneiro, M. (2020) “Detecção de Padrões em Imagens Através de Histogramas de Gradientes Orientados e Classificadores Lineares do Tipo SVM”, *Universidade Federal de Uberlândia*, p. 2.

Sociedade Brasileira de Pediatria (2018) “Acidentes domésticos estão entre principais causas de morte de crianças”, https://www.sbp.com.br/imprensa/detalhe/nid/acidentes-domesticos-estao-entre-principais-causas-de-morte-de-criancas/.

Singh, A. (2019) “Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor”, *Analytics Vidhya* https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/.

Tyagi, M. (2021) “HOG (Histogram of Oriented Gradients): An Overview. *Towards Data Science*”, https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f.

Weng, L. (2017) “Object Detection for Dummies Part 1: Gradient Vector, HOG, and SS”, https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html