Sistema de Chamada Automatizado com Reconhecimento Facial para Ambientes Educacionais: Uma Proposta de Implementação em Raspberry Pi 3

Mateus Borges Gonçalves Silveira
190093188@aluno.unb.br
Bacharelado em Engenharia Eletrônica
Faculdade de Ciências e Tecnologia
Universidade de Brasília

Matheus Alves Da Silva
matheus-alves.ma@aluno.unb.br
Bacharelado em Engenharia Eletrônica
Faculdade de Ciências e Tecnologia
Universidade de Brasília

I. Introdução

Este projeto propõe o desenvolvimento de um sistema de chamada automatizado inovador, utilizando tecnologia de reconhecimento facial em um sistema embarcado compacto baseado no Raspberry Pi 3 e uma câmera. O objetivo principal é otimizar o processo de registro de presença em ambientes educacionais, reduzindo a carga administrativa sobre os professores e aumentando a precisão e confiabilidade dos dados de frequência dos alunos. A solução visa oferecer uma alternativa de baixo custo e fácil implementação para instituições de ensino que buscam modernizar suas operações e garantir um controle de presença eficiente e à prova de fraudes. O sistema será projetado para ser autônomo, capturando imagens dos alunos ao entrarem na sala de aula, processando-as para identificar os indivíduos e marcando automaticamente sua presença em um banco de dados. A integração com sistemas existentes de gestão escolar será considerada para facilitar a exportação e o uso dos dados de frequência.

Esse trabalho é baseado na ideia do professor Diogo De Oliveira Costa, durante as aulas ele utiliza um sistema similar em sala, ao invés do rosto dos alunos o sistema utiliza a carteirinha estudantil para lançar a presença porem o sistema não e ideal e suscetível a fraudes.

Hoje já existem soluções similares como as desenvolvidas por Presença On-Line (FingerSec) [1], EduFace [2] entre outras, a maioria dessas soluções utilizam aplicativos para computadores ou celulares para realizar essa função nesse trabalho utilizaremos uma raspberry para atingir esse objetivo dado a necessidade de desempenho e armazenamento que outros microcontroladores simples não são capaz de fornecer.

A gestão da presença de alunos em instituições de ensino é uma tarefa fundamental, mas que frequentemente consomem tempo valioso de professores e administradores. Métodos tradicionais de chamada manual são suscetíveis a erros, demandam tempo em sala de aula e podem ser ineficientes, especialmente em turmas grandes. Com o avanço da tecnologia, surgem novas possibilidades para otimizar esse processo, tornando-o mais rápido, preciso e confiável. A aplicação dessa tecnologia em sistemas embarcados compactos, como o Raspberry Pi 3,

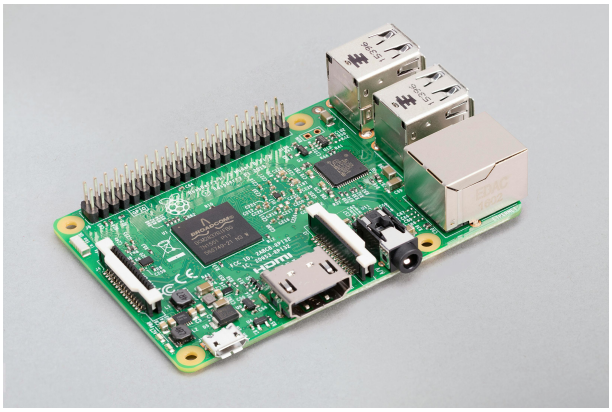

Figura 1. Raspberry Pi 3 Modelo B [3].

oferece uma solução de baixo custo e alta portabilidade. O Raspberry Pi 3, um computador de placa única, possui recursos de hardware que, embora limitados em comparação com computadores de mesa, são suficientes para a implementação de algoritmos de reconhecimento facial com o auxílio de bibliotecas otimizadas como o OpenCV [4]. Essa combinação permite o desenvolvimento de dispositivos autônomos que podem ser instalados em locais estratégicos, como entradas de salas de aula, para registrar a presença dos alunos de forma discreta e eficiente [5].

O controle de presença de alunos em instituições de ensino, desde o nível fundamental até o superior, é um processo crítico para a gestão acadêmica e administrativa. Tradicionalmente, esse controle é realizado de forma manual, com professores registrando a presença dos alunos em cadernos de chamada ou planilhas. Esse método, embora amplamente utilizado, apresenta uma série de desafios e ineficiências como: Consumo de Tempo, Susceptibilidade a Erros, Fraudes e Falsificações.

Diversas empresas oferecem sistemas de controle de presença baseados em reconhecimento facial [6], principalmente para ambientes corporativos e educacionais. Esses sistemas geralmente consistem em dispositivos de hardware [7] (terminais com câmeras e processadores) e software de

gestão [8]. Alguns exemplos notáveis incluem: Presença On-Line (FingerSec) [1], CHEGANDO (Diário Escola) [9] , EduFace [2],iDFace (Control iD) [10],Essas soluções comerciais demonstram a viabilidade e a demanda por sistemas de controle de presença automatizados, mas muitas vezes são soluções fechadas e de custo elevado, o que pode ser uma barreira para pequenas e médias instituições de ensino.

O método tradicional de chamada manual, embora familiar, é ineficiente, propenso a erros e consome um tempo valioso que poderia ser dedicado a atividades pedagógicas mais significativas. A precisão dos dados de frequência é crucial para a avaliação do desempenho acadêmico dos alunos, para a conformidade com regulamentações educacionais e para a comunicação eficaz com pais e responsáveis. A falta de um sistema automatizado e confiável pode levar a inconsistências nos registros, dificuldades na geração de relatórios e até mesmo a situações de fraude, comprometendo a integridade do processo educacional.

O objetivo geral deste projeto é desenvolver um sistema de chamada automatizado que utiliza reconhecimento facial [11] para marcar a presença dos alunos em um sistema embarcado compacto usando Raspberry Pi 3 e câmera. Para alcançar este objetivo geral, os seguintes objetivos específicos são propostos: desenvolver um módulo de detecção e reconhecimento facial, integrar o sistema com o hardware Raspberry Pi 3 e câmera, criar um banco de dados de alunos, implementar a funcionalidade de registro de presença, desenvolver uma interface de usuário e avaliar a performance do sistema.

Afim de alcançar o objetivo do projeto seria em ordem a detecção facial, reconhecimento facial, cadastro de alunos, registro de presença, geração de relatórios e interface de usuário.

II. Solução Proposta

A. Descrição de hardware

Componente	Função
Raspberry Pi 3	Computador principal que roda o sistema de reconhecimento facial e registra as presenças.
Câmera webcam	Captura as imagens dos alunos para análise facial.
Cartão microSD ou Pendrive	Armazena o sistema operacional, bibliotecas e banco de dados de imagens.
Fonte de alimentação 5V	Fornece energia para o Raspberry Pi e os periféricos.
Monitor e teclado (uso inicial)	Utilizados apenas na configuração do sistema e testes iniciais.
Case / suporte para câmera	Estrutura física para fixar e posicionar a câmera de forma adequada.
Botão	Realizar a captura da imagem e o processamento.
LEDs	Indicar o estado de captura.

Tabela I. Lista de hardware básico para sistema de chamada automatizada com Raspberry Pi

De acordo com o levantamento inicial de hardware, não foi necessária a utilização de circuitos eletrônicos adicionais para o funcionamento do projeto. Assim, apenas conhecimentos básicos de eletrônica e soldagem são suficientes para sua implementação.

A webcam USB escolhida não apresenta requisitos específicos além do fator custo, já que seu consumo de energia é desprezível. A resolução utilizada será relativamente baixa, em virtude das limitações de desempenho do Raspberry Pi.

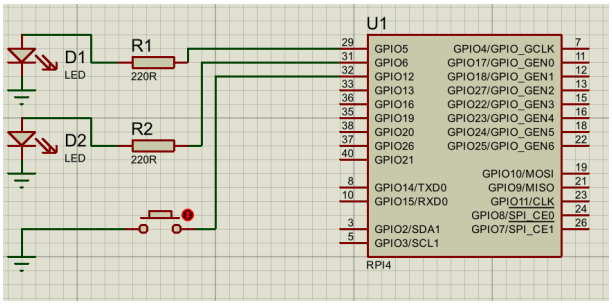

Figura 2. Esquemático Eletrônico Periférico.

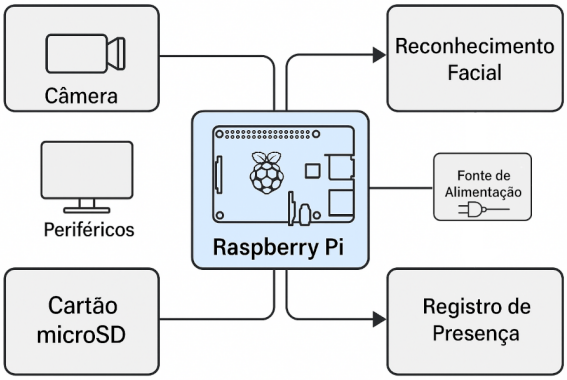

Figura 3. Diagrama de Blocos.

B. Descrição de software

Até o momento, o desenvolvimento do sistema de chamada automatizado concentrou-se na criação e validação da base de dados de faces por meio de dois programas modulares e independentes: o Módulo de Cadastro e o Módulo de Verificação.

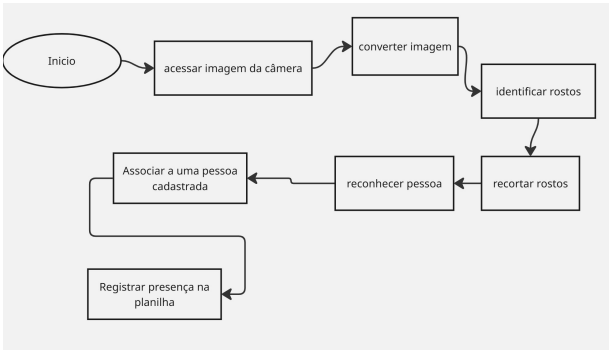

Figura 4. Diagrama De Blocos Geral Do Software.

O módulo de cadastro é a etapa fundamental que popula essa base de dados. Sua operação inicia ao receber o nome de um aluno e uma foto como entrada. A imagem é processada com um classificador Haar Cascade para detectar um único rosto, garantindo a qualidade do dado de entrada. Em seguida, a face detectada é recortada, convertida para escala de cinza e

padronizada em um tamanho fixo para consistência. As características extraídas deste rosto são então usadas para treinar ou atualizar o modelo de reconhecimento facial, baseado no algoritmo LBPH, que é salvo no arquivo reconhecedorFacial.yml. Concomitantemente, um ID numérico é associado ao nome do aluno e registrado no arquivo nomes.csv, completando o cadastro e fortalecendo o modelo a cada nova adição.

Para validar a precisão do modelo gerado, foi implementado o programa verificarReconhecimento. Ele carrega o modelo de inteligência do arquivo .yml e as identidades do arquivo .csv. Ao receber uma imagem de teste, o sistema detecta os rostos, os processa e os submete ao modelo para predição. O resultado inclui a identidade prevista e um índice de confiança, onde valores mais baixos indicam maior certeza. Finalmente, o programa exibe a imagem de teste com um retângulo ao redor do rosto, rotulado com o nome correspondente e a confiança. Se a confiança for baixa, o indivíduo é corretamente classificado como ”Desconhecido”, demonstrando a robustez do sistema.

Feito esse processo e associado um nome ao rosto que posteriormente e utilizado para marcar a presença. A estrutura de dados utilizada é composta por um vetor de registros (struct), no qual cada posição representa um aluno, contendo seu nome e o status de presença (presente ou ausente) assim como a data.

Inicialmente, todos os alunos são carregados a partir de um arquivo de texto (alunos.txt), onde cada linha corresponde a um nome. Após o carregamento, o sistema define a presença de todos como falta, garantindo que apenas os alunos efetivamente reconhecidos sejam marcados como presentes. Essa lista poderá ser convertida em um documento .csv para facilitar o acesso as informações.

III. Resultados Experimentais

Foram realizados testes das funcionalidades das partes do projeto desde o reconhecimento de face, captura da fotografia, entre outros, com objetivo de realizar a integração final do protótipo.

A respeito da captura de fotos foi utilizado a ferramenta de linha de comando fswebcam dado sua facilidade de uso e baixo custo de recursos, ela é amplamente utilizada em projetos embarcados.

Ao instalar esse recurso podemos fotografar usando o comando fswebcam -r 1920x1080 /local/nome.jpeg, porém esse comando resulta em uma imagem básica, acrescentando outras configurações ao código conseguimos pode-se obter resultados melhores da captura como mostras as imagens abaixo.

Com base nas imagens existe uma melhora da foto tirada com as modificações no código porem nada significativo, já em relação a referencia do Windows a imagem segue o mesmo padrão sem mudanças, considerando o fato que as configurações são fixas as fotos tiradas em condições de luminosidade diferentes podem ser obtidos resultados melhores ou piores.

Os principais problema encontrados ate em tao foram a dificuldade com foco automático, exposição e falta de pós-processamento, isso podeira ser corrigido por software porém utilizaria recursos limitados, por esse motivos optamos por manter a captura mais simplificada possível para manter o sistema responsivo.

Figura 5. Imagem referencia Windows.

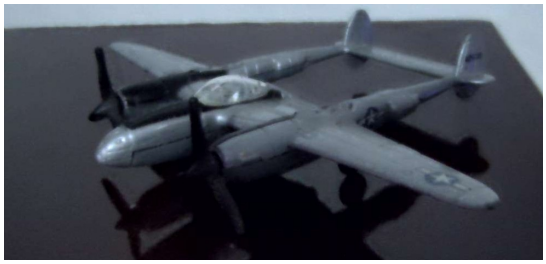

Figura 6. Imagem fswebcam original.

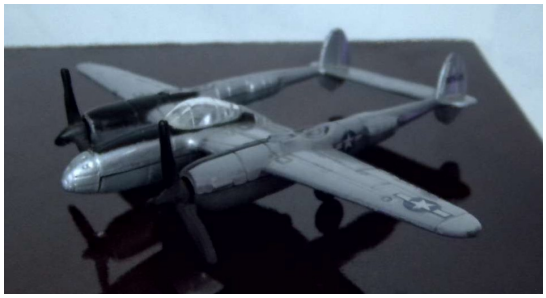

Figura 7. Imagem fswebcam Modificada.

Outras limitações em relação a captura foi a própria qualidade do sensor de captura que mesmo em condições ideias pode gerar imagens granuladas, o efeito dessa granulação no reconhecimento facial ainda devera ser analisadas posteriormente.

Referências

[1] Fingersec biometricy security, “Fingersec,” Disponível em: https://www.fingersec.com.br/presencaonline/ (03/09/2025).

[2] Eduface, “Eduface,” Disponível em https://eduface.com.br/ (03/09/2025).

[3] Raspberry Pi Foundation, “Raspberry Pi 3 Model B,” Disponível em: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/ (01/08/2025).

[4] Core Electronics, “Face Recognition With Raspberry Pi and OpenCV,” Disponível em https://core-electronics.com.au/guides/face-identify-raspberry-pi/ (03/09/2025).

[5] Hikvision, “Terminais de reconhecimento facial,” Disponível em: https://www.hikvision.com/pt-br/products/Access-Control-Products/Face-Recognition-Terminals/ (03/09/2025).

[6] New Science Publ, “Análise e avaliação de modelos de detecção e reconhecimento facial,” Disponível em https://periodicos.newsciencepubl.com/arace/article/download/3629/4691/13892 (03/09/2025).

[7] Dixi Ponto, “Relógio de Ponto Facial + Sistema de Ponto,” Disponível em: https://loja.dixiponto.com.br/relogio-de-ponto-facial-sistema-de-ponto-plano-mensal (03/09/2025).

[8] Jibble, “Presença por Reconhecimento Facial 100

[9] Diário Escola, “CHEGANDO e reconhecimento facial: revolucione a gestão escolar,” Disponível em https://diarioescola.com.br/organize-e-simplifique-sua-gestao-escolar com-chegando-e-reconhecimento-facial/ (03/09/2025).

[10] Control iD, “Controle de Acesso iDFace,” Disponível em https://www.controlid.com.br/controle-de-acesso/idface/ (03/09/2025).

[11] Kaspersky, “O que é reconhecimento facial e como ele funciona?” Disponível em https://www.kaspersky.com.br/resource-center/definitions/what-is-facial-recognition (03/09/2025).

APÊNDICE

Codigo: cadastrar_usuario

```
#include <opencv2/opencv.hpp>
#include <opencv2/face.hpp>
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;
using namespace cv;
using namespace cv::face;

// Função para ler o arquivo CSV
e determinar o próximo ID disponível
int getNextId(const string& csv_path) {
    ifstream file(csv_path);
    string line;
    int max_id = 0;

    while (getline(file, line)) {
        stringstream ss(line);
        string id_str;
        // Pega a primeira parte da linha (o ID)
        getline(ss, id_str, ';');
        try {
            int id = stoi(id_str);
            if (id > max_id) {
                max_id = id;
            }
        } catch (const std::invalid_argument& e)
        {
            // Ignora linhas mal formatadas
        }
    }
    file.close();
    return max_id + 1;
}

int main(int argc, char** argv) {


    if (argc != 3) {
        cout << "Uso: " << argv[0] << " \"
    <Nome do Aluno>\"
    <caminho_para_a_foto.jpg>" << endl;
    return -1;
}

string student_name = argv[1];
string image_path = argv[2];
string cascade_path =
"haarcascade_frontalface_default.xml";
string model_path =
"reconhecedor_facial.yml";
string csv_path = "nomes.csv";



Mat image = imread(image_path, IMREAD_COLOR);
if (image.empty()) {
    cout << "Erro: Não foi possível
    carregar a imagem em: "
    << image_path << endl;
    return -1;
}

CascadeClassifier face_cascade;
if (!face_cascade.load(cascade_path)) {
    cout << "Erro: Não foi possível
    carregar o classificador Haar Cascade."
    << endl;
    return -1;
}

Mat gray;
cvtColor(image, gray, COLOR_BGR2GRAY);
equalizeHist(gray, gray);

vector<Rect> faces;
face_cascade.detectMultiScale(gray,
faces, 1.1, 5, 0|CASCADE_SCALE_IMAGE,
Size(100, 100));

if (faces.size() != 1) {
    cout << "Erro: A foto de cadastro
    deve conter exatamente um rosto.
    Foram encontrados: "
    << faces.size() << endl;
    return -1;
}

cout << "Rosto detectado com sucesso.
Iniciando o processo de cadastro..."
<< endl;



// Recorta o rosto da imagem original
Mat face_roi = gray(faces[0]);

// Padroniza o tamanho da imagem
do rosto

(importante para o reconhecimento)
Mat resized_face;
```

```
resize(face_roi, resized_face,
Size(200, 200), 1.0, 1.0, INTER_CUBIC);

// Determina o ID para o novo aluno
int new_id = getNextId(csv_path);

vector<Mat> images_to_train;
vector<int> labels_to_train;

images_to_train.push_back(resized_face);
labels_to_train.push_back(new_id);

// Cria o reconhecedor LBPH
Ptr<LBPHFaceRecognizer> model =
LBPHFaceRecognizer::create();

// Se o modelo já existe, ele é
carregado eatualizado
(treinado com a nova foto).
// Se não, um novo modelo é
treinado do zero.
try {
    model->read(model_path);
    cout << "Modelo existente carregado.
    Atualizando com o novo rosto..."
    << endl;
    model->update
    (images_to_train, labels_to_train);
} catch (const cv::Exception& e) {
    cout << "Nenhum modelo encontrado.
    Criando um novo..."
    << endl;
    model->train
    (images_to_train, labels_to_train);
}



// Salva o modelo treinado/atualizado
model->save(model_path);

// Salva a associação ID ->
Nome no arquivo CSV
// O 'fstream::app' garante que a nova
linha seja adicionada ao final do arquivo
ofstream csv_file(csv_path, fstream::app);
if (!csv_file.is_open()) {
    cout << "Erro ao abrir o arquivo CSV
    para escrita." << endl;
    return -1;
}
csv_file << new_id << ";"
<< student_name << endl;
csv_file.close();

cout << "-------------" << endl;
cout << "Aluno cadastrado com sucesso!"
<< endl;
cout << "ID: " << new_id << endl;
cout << "Nome: " << student_name
<< endl;
cout << "Modelo salvo em: "
<< model_path << endl;
cout << "CSV atualizado em: "
<< csv_path << endl;
cout << "-------------" << endl;

return 0;
}
```

Codigo: verificar_reconhecimento

```
#include <opencv2/opencv.hpp>
#include <opencv2/face.hpp>
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <map>

using namespace std;
using namespace cv;
using namespace cv::face;

// Função para carregar o arquivo CSV com os
nomes dos alunos
map<int, string>
loadNames(const string& csv_path) {
    map<int, string> names;
    ifstream file(csv_path);
    string line;
    while (getline(file, line)) {
        stringstream ss(line);
        string id_str, name_str;
        if (getline(ss, id_str, ';')
        && getline(ss, name_str)) {
            try {
                names[stoi(id_str)] = name_str;
            } catch
            (const std::invalid_argument& e)
            {
                // Ignora linhas mal formatadas
            }
        }
    }
    return names;


int main(int argc, char** argv) {
    // --- 1. VERIFICAÇÃO DOS ARGUMENTOS ---
    if (argc != 2) {
        cout << "Uso: " << argv[0] <<
        " <caminho_para_a_foto_de_teste.jpg>
        " << endl;
        return -1;
    }

    string test_image_path = argv[1];
    string cascade_path =
    "haarcascade_frontalface_default.xml";
    string model_path =
    "reconhecedor_facial.yml";
    string csv_path = "nomes.csv";
```

```
// --- 2. CARREGAMENTO DO MODELO, NOMES
E CLASSIFICADOR ---
cout << "Carregando modelo de
reconhecimento..." << endl;
Ptr<LBPHFaceRecognizer> model =
LBPHFaceRecognizer::create();
model->read(model_path);

cout << "Carregando banco de dados
de nomes..." << endl;
map<int, string> names =
loadNames(csv_path);
if (names.empty()) {
    cout << "Erro:
    Nenhum nome encontrado no arquivo CSV
    ou arquivo não existe." << endl;
    return -1;
}

CascadeClassifier face_cascade;
if (!face_cascade.load(cascade_path)) {
    cout << "Erro: Não foi possível
    carregar o classificador Haar Cascade."
    << endl;
    return -1;                          }
}

// --- 3. CARREGAMENTO E PROCESSAMENTO
DA IMAGEM DE TESTE ---
Mat test_image =
imread(test_image_path, IMREAD_COLOR);
if (test_image.empty()) {
    cout << "Erro: Não foi possível carregar
    a imagem de teste." << endl;
    return -1;
}

Mat gray;
cvtColor(test_image, gray, COLOR_BGR2GRAY);

vector<Rect> faces;
face_cascade.detectMultiScale
(gray, faces, 1.1, 5, 0|CASCADE_SCALE_IMAGE,
Size(100, 100));

// --- 4. DETECÇÃO E RECONHECIMENTO ---
for (const auto& face : faces) {
    // Recorta e prepara o rosto para
    a predição (deve ser do mesmo
    tamanho usado no treino)
    Mat face_roi = gray(face);
    Mat resized_face;
    resize(face_roi, resized_face,
    Size(200, 200), 1.0, 1.0, INTER_CUBIC);

    int predicted_label = -1;
    double confidence = 0.0;
    model->predict(resized_face,
    predicted_label, confidence);

    string label_text;
    if (confidence < 80) {
        label_text = names[predicted_label] +
        (Conf: " + to_string(confidence) + ")
    } else {
        label_text = "Desconhecido (Conf: "
        + to_string(confidence) + ")";
    }

    // Desenha o resultado na imagem
    rectangle(test_image, face, Scalar
    (0, 255, 0), 2);
    Point text_pos(face.x, face.y - 10);
    putText(test_image, label_text,
    text_pos, FONT_HERSHEY_SIMPLEX,
    0.5, Scalar(0, 255, 0), 2);
}

// --- 5. EXIBIÇÃO DO RESULTADO ---
imshow("Resultado do Reconhecimento",
test_image);
cout << "Pressione qualquer tecla na
janela da imagem para sair." << endl;
waitKey(0);

return 0;
```