

UNIVERSIDADE ESTADUAL DO RIO GRANDE SUL  
ENGENHARIA DE COMPUTAÇÃO

ALLEFF DYMYTRY PEREIRA DE DEUS

**RECONHECIMENTO FACIAL PARA  
CLASSIFICAÇÃO E REGISTRO DE  
PRESENÇA EM SALA DE AULA**

Trabalho de Conclusão apresentado como  
requisito parcial para a obtenção do grau de  
Engenheiro em Engenharia de Computação

Profa. Dra. Letícia Vieira Guimarães  
Orientador

Guaíba, Dezembro de 2020

## FOLHA DE APROVAÇÃO

Monografia sob o título "*Reconhecimento Facial para Classificação e Registro de Presença em Sala de Aula*", defendida por Alleff Dymytry Pereira de Deus e aprovada em 3 de Julho de 2020, em Guaíba, estado do Rio Grande do Sul, pela banca examinadora constituída pelos professores:

---

Profa. Dra. Letícia Vieira Guimarães  
Orientadora

---

Profa. Dra. Adriane Parraga

---

Prof. Dr. João Leonardo Fragoso

*"Você deve entender que há mais de um caminho para o topo da montanha."*  
— MIYAMOTO MUSASHI

## **AGRADECIMENTOS**

Aos meus ....

À minha ...

Aos meus...

À Universidade, professores e funcionários que propiciaram um excelente ambiente de aprendizagem.

# SUMÁRIO

<b>LISTA DE ABREVIATURAS E SIGLAS</b>	<b>7</b>
<b>LISTA DE SÍMBOLOS</b>	<b>8</b>
<b>LISTA DE FIGURAS</b>	<b>9</b>
<b>LISTA DE TABELAS</b>	<b>10</b>
<b>RESUMO</b>	<b>11</b>
<b>ABSTRACT</b>	<b>12</b>
<b>1 INTRODUÇÃO</b>	<b>13</b>
<b>2 FUNDAMENTOS TEÓRICOS</b>	<b>14</b>
2.1 PROCESSAMENTO DIGITAL DE IMAGENS	14
2.1.1 Imagens	14
2.2 HAAR LIKE FEATURES	14
2.3 ANÁLISE DE COMPONENTES PRINCIPAIS	14
2.4 EIGENFACES	15
2.4.1 Reconhecimento Utilizando Eigenfaces	16
2.5 SISTEMAS EMBARCADOS	16
<b>3 MATERIAIS E MÉTODOS</b>	<b>18</b>
3.1 Aquisição da Imagem	21
3.2 Segmentação Facial	21
3.3 Obtenção Eigenfaces	22
3.4 Classificação Eigenfaces	25
3.5 Desenvolvimento da Interface	26
3.6 Integração Banco de Dados	27
3.7 Atestado de Presença	28
3.8 Integração do Sistema	29
<b>4 RESULTADOS E EXPERIMENTOS</b>	<b>30</b>
4.1 Eigenfaces	30
4.2 Armazenamento	35
4.3 Interface	37
<b>5 CONCLUSÃO</b>	<b>40</b>

<b>6 TRABALHOS FUTUROS . . . . .</b>	<b>41</b>
<b>REFERÊNCIAS . . . . .</b>	<b>42</b>

## **LISTA DE ABREVIATURAS E SIGLAS**

PCA	Principal Component Analysis
ROI	Region Object Interest

## LISTA DE SÍMBOLOS

$\Gamma$	Vetor unidimensional da Face
$\Psi$	Face Média do conjunto de faces
$\Phi$	Face com principais componentes ressaltadas
$\Omega$	Conjunto de pesos de cada Eigenface para determinada face $\Phi$
$e_{\Omega}$	Erro das diferenças dos pesos $\Omega$
$e_{\Phi}$	Erro das diferenças das faces $\Phi$
$\theta_{\Omega}$	Valor de Threshold dos pesos $\Omega$
$\theta_{\Phi}$	Valor de Threshold das faces $\Phi$



## LISTA DE FIGURAS

3.1	Diagrama das etapas de desenvolvimento. . . . .	18
3.2	Diagrama do processo de supervisão do sistema. . . . .	19
3.3	Diagrama das etapas de desenvolvimento. . . . .	20
3.4	Esquemático do método de classificação. . . . .	20
3.5	Segmentação Facial. . . . .	22
3.6	Faces sem tratamento pertencentes ao treino. . . . .	22
3.7	Face tratada pertencente ao treino. . . . .	23
3.8	Face média $\Psi$ . . . . .	23
3.9	Faces $\Phi$ . . . . .	24
3.10	Eigenfaces. . . . .	24
3.11	Diagrama do sistema. . . . .	29
4.1	Face de teste, Face reconhecida do banco e Face reconhecida do sistema: (a) Face que foi reconhecida, (b) Face não foi reconhecida . . .	30
4.2	Gráfico K x Acertos. . . . .	31
4.3	Gráfico K x Erros. . . . .	31
4.4	Desvio Padrão dos Omegas projetados no espaço de faces. . . . .	32
4.5	Desvio Padrão dos Omegas no espaço de faces. . . . .	32
4.6	Faces reconhecidas do treinamento utilizando Erro $\Phi = 30$ . . . . .	33
4.7	Faces reconhecidas do treinamento utilizando Erro $\Phi = 31$ . . . . .	33
4.8	Faces reconhecidas do treinamento utilizando Erro $\Phi = 32$ . . . . .	34
4.9	Faces reconhecidas do treinamento utilizando Erro $\Phi = 33$ . . . . .	34
4.10	Faces reconhecidas do treinamento utilizando Erro $\Phi = 34$ . . . . .	35
4.11	Faces reconhecidas do treinamento utilizando Erro $\Phi = 35$ . . . . .	35
4.12	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 30$ . . .	36
4.13	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 31$ . . .	36
4.14	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 32$ . . .	37
4.15	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 33$ . . .	37
4.16	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 34$ . . .	38
4.17	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 35$ . . .	38
4.18	Armazenamento das imagens a partir dos Bancos de Dados. . . . .	39
4.19	Recuperação das imagens a partir dos Bancos de Dados. . . . .	39

# LISTA DE TABELAS

4.1	Falsos positivos e negativos . . . . .	39
4.2	Tempo de recuperação das imagens . . . . .	39

## **RESUMO**

Este trabalho tem por objetivo o desenvolvimento de um sistema que detecta faces para o auxílio no registro de alunos presentes no ambiente acadêmico. Para a localização facial será utilizado o método de classificação Haar Like Features e para a classificação das faces encontradas será utilizado o método Eigen Faces. Para a facilidade de utilização do usuário final, os métodos irão ser integrados em um sistema que possibilita a inclusão dos dados diretamente no banco de dados, possibilitando a consulta dos mesmo com mais agilidade.

**Palavras-chave:** Eigenfaces, Reconhecimento Facial, Haar Like Features.

## **Facial Recognition for Classification and Register on Presence in Classroom**

### **ABSTRACT**

(COLOCAR EM INGLES DEPOIS)Este trabalho tem por objetivo o desenvolvimento de um sistema que detecta faces para o auxílio no registro de alunos presentes no ambiente acadêmico. Para a localização facial será utilizado o método de classificação Haar Like Features e para a classificação das faces encontradas será utilizado o método Eigen Faces. Para a facilidade de utilização do usuário final, os métodos irão ser integrados em um sistema que possibilita a inclusão dos dados diretamente no banco de dados, possibilitando a consulta dos mesmo com mais agilidade.(COLOCAR EM INGLES DEPOIS)

**Keywords:** Eigenfaces,Facial Recognition,Haar Like Features.

# 1 INTRODUÇÃO

Nos dias atuais pode-se notar que a biometria é algo amplamente utilizado e ajuda no cotidiano de todas as pessoas, já que pode-se fazer as mais diversas funções, sendo elas: pagamento de contas, liberação para áreas restritas, verificação em e-mails, para direção de um veículo, etc. A biometria tem por base a medição de características humanas de forma analógicas e transforma para o mundo digital, sendo uma delas a mais emergente o reconhecimento facial.

Segundo a ANSA, no carnaval de 2019, a polícia do Rio de Janeiro e do Salvador, conseguiram detectar e prender criminosos com a ajuda de câmeras equipadas com reconhecimento facial. Existe no mercado atual um crescimento de 20

O reconhecimento facial pode ser empregado nas mais diferentes áreas, sem ser a da segurança, com isso pode-se empregar o reconhecimento para verificar sentimentos, expressões, executar comandos configurados, check-in em eventos, entre outras aplicações.

Com a popularização dos meios de reconhecimento facial, algumas técnicas foram desenvolvidas conforme o tempo e a tecnologia se desenvolveu. Algumas das técnicas que são utilizadas, são elas: Eigenfaces, Fisherfaces, Kernel Direct Discriminant Analysis, K Nearest Neighbors, Local Binary Pattern.

Com esses avanços nas técnicas de reconhecimento facial, pode-se utilizar tais avanços para o reconhecimento de alunos em sala de aulas, já que os mesmos querem ter seus rostos reconhecidos para obter a presença em sala de aula, fazendo que o método tradicional de folha de chamada possa ser substituído e todo o sistema de presença seja diretamente integrado em um único sistema.

Desta forma, a construção de um sistema de reconhecimento facial integrado com um controle de presença se faz necessário para sanar o problema de ainda hoje, em um tempo de integração e avanço tecnológico, utilizar chamadas impressas para marcar se o aluno estava ou não em aula ou em um determinado evento.

## 2 FUNDAMENTOS TEÓRICOS

Neste capítulo serão abordados os conhecimentos necessários para a contextualização do presente trabalho, bem como seus métodos de desenvolvimento e funcionalidades específicas.

### 2.1 PROCESSAMENTO DIGITAL DE IMAGENS

Para poder abordar como são feitos os processamentos a cerca de imagens, primeiramente é necessário se fundamentar o que são imagens.

#### 2.1.1 Imagens

Uma imagem pode ser descrita como uma função bidimensional, que pode possuir dimensões de tamanhos diferentes. As coordenadas da imagem formam um plano e sua amplitude pode ser descrita como a intensidade do conjunto de coordenadas. As intensidades de uma imagem são processos físicos relacionados a energia irradiada pela fonte. Uma imagem pode ser captada a partir de sensores que conseguem traduzir a energia irradiada pela fonte em uma saída de tensão. Segundo (GONZALEZ, 2010), um dos sensores mais conhecidos para tal utilização, são os fotodiodos, sendo estes sensores construídos com materiais semicondutores que possuem uma saída de tensão proporcional a intensidade luminosa. Pode-se caracterizar uma imagem como digital, quando os valores dos elementos de coordenadas e amplitude são finitas. Os elementos que compoem uma imagem digítal são comumente chamados de *Pixels*.

### 2.2 HAAR LIKE FEATURES

Os Haar Like Features são características de imagens digitais utilizadas para o reconhecimento de objetos,

### 2.3 ANÁLISE DE COMPONENTES PRINCIPAIS

A Análise de Componentes Principais (Principal Component Analysis - PCA) é uma técnica matemática utilizada para a análise de dados de multivariáveis. Primeiramente concebido por Karl Pearson em 1901 e posteriormente desenvolvida por Harold Hotelling em 1930. O PCA não era utilizado tão comumente até os avanços na área da computação, se baseia na utilização de princípios matemáticos para transformar um número de variáveis correlacionais em quantidades menores de variáveis, sendo estas as principais componentes das variáveis. A principal transformação é a redução da dimensionalidade

do conjunto de dados, fazendo com que seja ressaltado as similaridades e diferenças dentro do conjunto de dados. Segundo (MISHRA et al., 2017) os autovetores e autovalores são os princípios fundamentais para a determinação dos PCAs, decorrido da decomposição da matriz de covariância. O processo de obtenção dos PCAs pode ser feito através dos seguintes passos:

### 1. Subtração da Média

Para que o cálculo do PCA funcione devidamente, é necessário realizar a subtração do conjunto dos dados de sua média em ambos os eixos, para que possa ser produzido uma média dos dados de valor zero;

### 2. Calcular a Matriz de Covariância

É necessário obter a matriz de covariância do conjunto de dados que já foram subtraídos da média;

### 3. Cálculo dos autovetores e autovalores

Calcular os autovetores e autovalores da matriz de covariância, sendo os autovetores uma parte muito importante para o PCA;

### 4. Escolher as componentes

Deve-se escolher os maiores eigenvalues dos quais serão obtidos os maiores eigenvectors como consequência, sendo estes os valores que mais representam as componentes principais do conjunto de dados.

Ainda segundo (MISHRA et al., 2017), o PCA pode ser considerado como a projeção do conjunto de dados em uma direção do espaço onde os dados tem uma grande variação, onde a direção é dada pelos eigenvectors da matriz de covariância correspondendo aos maiores eigenvalues.

## 2.4 EIGENFACES

Dentro dos algoritmos e cálculos de reconhecimento facial, um desses modelos de reconhecimento é o Eigenfaces, que consiste na obtenção de eigenvectors que melhor representam a face de cada indivíduo. A utilização do termo *Eigenface* foi apresentado por (TURK; PENTLAND, 1991), onde consiste "faces fantasmas" que possuem os maiores eigenvectors e eigenvalues do PCA do conjunto de dados. Segundo (TURK; PENTLAND, 1991) a ideia de se utilizar eigenfaces foi motivado na pesquisa de (SIROVICH; KIRBY, 1987), que utilizavam as melhores coordenadas para representar uma face, sendo denominado de *Eigenpictures*. Com isso podia-se reconhecer uma face com uma pequena componente, no caso de (SIROVICH; KIRBY, 1987) utilizava-se a parte dos olhos para se fazer o reconhecimento, contudo se pensou que poderiam utilizar uma pequena quantidade de características diferentes para reconhecer cada indivíduo, formando assim uma Eigenface. Sendo assim a utilização das eigenfaces é feito por meio da obtenção das PCAs da face de cada indivíduo de um conjunto de imagens previamente separado, onde os melhores eigenvectors serão a projeção dessas faces no conjunto de componentes denominado "espaço de faces".

### 2.4.1 Reconhecimento Utilizando Eigenfaces

Pode-se utilizar eigenfaces para reconhecimento facial, alguns passos devem ser feitos antes do reconhecimento concreto, levando em consideração que as imagens utilizadas estão centralizadas e possuem os mesmos tamanhos. Sendo assim pode-se dividir o processo em duas etapas, a primeira de treinamento e o segundo de reconhecimento. Para a etapa de treinamento os seguintes passos devem ser feitos:

- Adquirir uma coletânea de imagens para ser o conjunto de treinamento;
- Deixar as imagens no tamanho desejado, cortando somente a face como objeto de interesse;
- Transformar o vetor de imagens  $I$  ( $I = N \times N$ ) em um vetor gamma  $\Gamma$  ( $\Gamma = N^2 \times 1$ );
- Calcular a face média  $\Psi$  do vetor  $\Gamma$ ;
- Obter a matriz de covariância  $C$ ;
- Obter os autovalores e autovetores da matriz de Covariância  $C$ ;
- Selecionar os melhores  $M$  eigenvectors;
- Manter somente os  $K$  melhores autovetores ( $K$  com maiores autovalores);
- Calcular os pesos  $\Omega$  de cada face  $\Phi$  pelas eigenfaces.

A segunda etapa sendo a de reconhecimento, se utiliza uma face que esta fora do conjunto de treinamento para poder fazer a validação, além de que deve estar com o mesmo tamanho das imagens do conjunto de treino, seguindo os seguintes passos:

- Normalizar a imagem de teste  $\Phi = \Gamma - \Psi$
- Calcular pesos  $\Omega$  da imagem de teste pelas eigenfaces do conjunto de treinamento;
- Determinar se a imagem de entrada é uma face pertencente ao conjunto de treinamento pela distância Euclidiana;

## 2.5 SISTEMAS EMBARCADOS

Atualmente a tecnologia está cada vez mais popular e acessível, sendo assim possui-se diversos tipos diferentes de tecnologias para os mais diversos fins, sendo assim um conceito para que essas tecnologias possam ser classificadas e separadas para os seus específicos fins, pode-se utilizar o conceito de sistemas embarcados. Os sistemas embarcados são tecnologias que são construídas para devidos fins específicos com hardware específico que acaba não podendo ser reutilizado em outras aplicações que não possuam o mesmo hardware utilizado em seu desenvolvimento original.

Os sistemas embarcados podem normalmente desempenhar processos simples, que não geram nenhum tipo de risco para os usuários (exemplo de calculadoras, controles de videogames, telefones, etc.), contudo também podem ser utilizados para tarefas mais complexas que apresentam certos riscos se não forem projetados com um maior rigor (exemplo controle em aviões, controles industriais, monitoramento de saúde, etc.).



Neste âmbito um sistema desenvolvido para um hardware específico pode ser considerado um sistema embarcado, neste tipo de aplicação um hardware muito utilizado atualmente são as RaspberryPi, que são microprocessadores que rodam sistemas operacionais de diversos tipos.

### 3 MATERIAIS E MÉTODOS

Neste capítulo serão abordados os materiais e os métodos utilizados para a construção e desenvolvimento do presente trabalho. O desenvolvimento do presente trabalho possui uma construção paralela das funções para que se possa ser integrado cada parte do sistema, afim de chegar no final do desenvolvimento com todas as funções funcionando integralmente, sendo assim o início do desenvolvimento segue a etapa de aquisição das imagens, tendo em paralelo o desenvolvimento de uma interface que seja de simples acesso e o mais claro possível para o usuário final.



Figura 3.1: Diagrama das etapas de desenvolvimento.

O desenvolvimento do presente trabalho foi dividido em duas partes seguindo o modelo de metodologia da Figura 3.1, sendo agrupadas em relação aos processos necessários para o funcionamento das funções desenvolvidas, sendo assim pode-se dividir entre o processo de supervisão do sistema e o processo de atribuição da presença do aluno. Algumas funções possuem características em comum, mudando somente a utilização final de seus resultados, como pode ser notado na Figura 3.2 e 3.3. O processo de supervisão do sistema tem por objetivo conter todas as funções que somente o administrador do sistema poderá utilizar. O processo de atribuição de presença do aluno ocorre automaticamente

dentro do sistema, sendo necessário ao administrador analisar os logs do sistema para caso seja necessário ajustar ou inserir o aluno na base de dados.

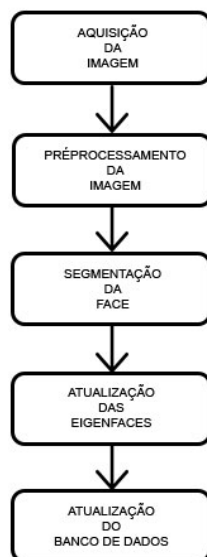


Figura 3.2: Diagrama do processo de supervisão do sistema.

A aplicação foi desenvolvida utilizando a linguagem de programação Python, sendo esta uma linguagem de programação considerada de alto nível, tendo o seu funcionamento interpretado, onde o programa é executado por um interpretador e após pelo sistema operacional ou processador. A linguagem também é orientada a objetos e funcional. Foi criada em 1991. Possui uma grande comunidade, já que a linguagem possui um modelo de desenvolvimento comunitário, onde várias pessoas podem ajudar a desenvolver novas bibliotecas para que toda a comunidade possa utilizar. O sistema proposto foi desenvolvido e testado em uma máquina que possui um processador AMD Ryzen 5 3400G de 3,7GHz, memória RAM de 16GB e uma placa de vídeo Radeon RX 590 de 8GB de VRAM, mas para a aplicação final será utilizado em uma Raspberry Pi 3, que possui um processador ARMv8 CORTEX A53 QUADCORE com velocidade de operação de 1.2GHz e 1GB de memória RAM. Utilizando o sistema operacional Linux para executar a aplicação desenvolvida.

As imagens utilizadas para o treinamento no presente trabalho foram retiradas do banco de imagens FEI face database, sendo este um banco de imagens desenvolvido pelo Laboratório de Processamento de Imagens do Departamento de Engenharia Elétrica do Centro Universitário da FEI. Esse banco contém 14 imagens diferentes de 200 indivíduos diferentes, obtendo um total de 2800 imagens. Dos 200 indivíduos pertencentes ao banco, para o treinamento do presente trabalho somente 49 indivíduos foram utilizados, sendo 1/4 do valor total. As imagens do banco de faces possuem uma diversidade de indivíduos, tendo diferenças de idade, diferenças de sexo, diferenças de etnias e etc, com isso é possível gerar uma diversidade de PCAs. Segundo (TURK; PENTLAND, 1991), existem problemas significantes de se possuir fundo nas imagens das faces, já que as eigenfaces não fazem distinção do que é ou não uma face. Sendo assim deste banco somente as imagens de face frontal foram utilizadas para o formar um conjunto de treinamento e teste. Sendo assim é necessário recortar somente o rosto de cada indivíduo.

Neste trabalho o processo de reconhecimento facial segue os processos básicos para o reconhecimento, tendo somente a modificação de inserção dos dados obtidos no banco

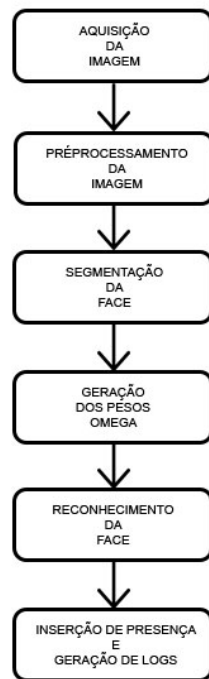


Figura 3.3: Diagrama das etapas de desenvolvimento.

de dados, esta ordem pode ser notada na figura 3.4.

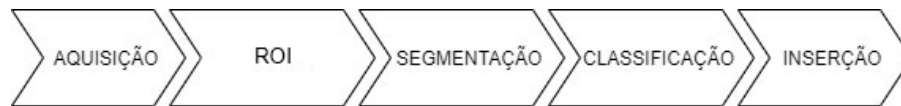


Figura 3.4: Esquemático do método de classificação.

- **AQUISIÇÃO:** O processo de aquisição se refere ao momento em que o usuário tem sua face adquirida por um dispositivo de captura (câmera, filmadora, webcam, etc.);
- **ROI:** O processo de determinação de região de interesse tem por funcionamento encontrar uma face na imagem adquirida no processo anterior;
- **SEGMENTAÇÃO:** O processo de segmentação por sua vez retira para o sistema somente a face localizada no processo anterior, facilitando a próxima etapa do sistema;
- **CLASSIFICAÇÃO:** O processo de classificação consiste em utilizar a face para o reconhecimento de qual usuário está no sistema;
- **INSERÇÃO:** O processo de inserção finaliza o sistema, inserindo qual usuário que foi reconhecido pelos processos anteriores.

Cada fase do reconhecimento facial tem uma parte dentro do sistema que executa uma função necessária para poder obter as informações necessária para conseguir obter um valor(face) aceitável para o reconhecimento.

### 3.1 Aquisição da Imagem

Para a aquisição da imagem dentro do sistema é necessário a utilização de imagens adquiridas a partir de uma webcam para captura, para ser feito o acesso a webcam a biblioteca OpenCV foi utilizada, sendo o OpenCV uma biblioteca de visão computacional e aprendizagem de máquina com código livre, possuindo mais 2500 algoritmos otimizados para uso de todos. Os algoritmos vão desde detectar objetos até extração de modelos 3D. A biblioteca pode ser utilizada nas linguagens de programação Python, C++, Java e MatLab, tendo seus códigos escritos nativamente em C++. Com a utilização do OpenCV pode-se utilizar diversos tamanhos de imagens diferentes, no presente trabalho foi utilizado as seguintes dimensões: Altura - 480 pixels e Largura - 640 pixels, poderia se utilizar o tamanho da imagem necessária para a obtenção das eigenfaces diretamente, contudo se perderia resolução no momento de aquisição da imagem, reduzindo a chance de reconhecimento das eigenfaces.

#### 1. Aquisição na Supervisão do Sistema

A aquisição da imagem durante a supervisão do sistema ocorre de forma a se obter imagens para cadastrar um novo aluno ou alterar uma imagem já existente. Sendo assim o administrador do sistema pode adquirir uma imagem quando achar necessário, durante a aquisição o processo de salvar uma imagem só pode ser feito se uma face for encontrada pelas características haar, como na Figura 3.5, caso não seja detectado uma face o sistema não permite que seja adquirido uma imagem, assim protegendo o sistema de possíveis falhas.

#### 2. Aquisição na Atribuição da presença

A aquisição da imagem durante o processo de atribuição da presença ocorre automaticamente, onde assim que uma face é encontrada, a imagem da mesma é adquirida e disponibilizada para o sistema, sem a supervisão de um usuário. A imagem adquirida é disponibilizada por um curto período de tempo para que o indivíduo possa ver que seu rosto foi adquirido pelo sistema e sua presença registrada.

### 3.2 Segmentação Facial

Para se obter somente a face de cada indivíduo, foi utilizado o Haar Like feature para faces frontais e realizar um processamento na imagem para se obter somente um único canal de escala de cinza (as imagens originais possuem três canais RGB), este tratamento e manuseio das imagens é feito utilizando o OpenCV. A segmentação das faces é dividida em duas situações:

#### 1. Segmentação na Supervisão do Sistema

A segmentação da face na supervisão do sistema utiliza as faces adquiridas previamente pelo administrador do sistema, sendo processadas utilizando as características haar para retirar somente a face frontal sendo esta a região de interesse. As faces segmentadas passam por um processamento para somente possuir um canal em escala de cinza, assim elas podem ser utilizadas para os cálculos necessários no sistema.

#### 2. Segmentação na Atribuição de presença

A segmentação da face durante a atribuição da presença, utiliza a face adquirida pelo sistema a partir da webcam, segmentando a face utilizando as características haar como na supervisão do sistema, contudo este processo ocorre sem que o administrador precise ter interação com o sistema. A face segmentada é encaminhada para as próximas funções necessárias para a atribuição da presença, um exemplo da segmentação na interface pode ser notado na Figura 3.5.

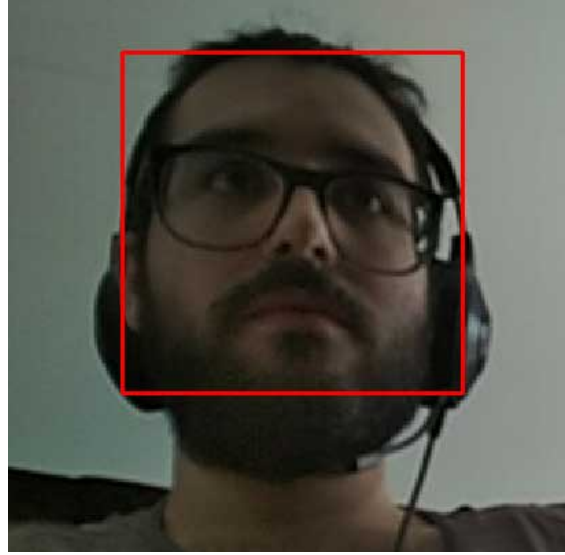


Figura 3.5: Segmentação Facial.

### 3.3 Obtenção Eigenfaces

Para os cálculos de obtenção da eigenfaces a biblioteca Numpy foi utilizada, sendo esta uma biblioteca de código aberto que consegue processar os mais diversos tipos de processos de computação numérica. Criada em 2005 e utilizada amplamente na comunidade científica para os mais diversos cálculos matemáticos e manipulação de dados.

Para os cálculos das eigenfaces deve-se considerar o tamanho das imagens que serão utilizadas no sistema, sendo assim, tanto as imagens do banco de faces e as imagens adquiridas pela webcam foram redimensionadas no tamanho  $N = 300 \times 300$  e possuem somente um canal em escala de cinza para a minimização dos cálculos (Imagem sem modificar  $I : (N \times N \times 3)$ , imagem modificada  $I : (N \times N \times 1)$ ) após deve-se transformar cada imagem  $I : (N \times N)$  em um vetor gama  $\Gamma : (N^2 \times 1)$ , onde é necessário transformar o vetor em uma dimensão, um exemplo da imagem de treinamento pode ser vista na figura 3.7. As imagens podem ser normalizadas dividindo pelo valor máximo de 255 para facilitar os cálculos.

Após este processo de achatamento da imagem, deve-se obter a face média do treinamento, sendo uma soma de todas as imagens presentes no treinamento e dividido pela quantidade de imagens, sendo obtida uma imagem  $\psi$  na equação 3.1.

$$\Psi = \frac{1}{M} \sum_{i=1}^M \Gamma_i \quad (3.1)$$

A imagem gerada é um amalgama de todas as faces de treino, possuindo um pouco de cada característica mais exuberante de cada face, o resultado da junção das faces pode ser

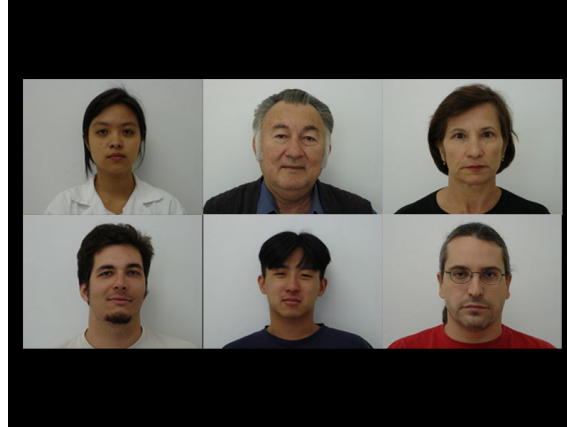


Figura 3.6: Faces sem tratamento pertencentes ao treino.

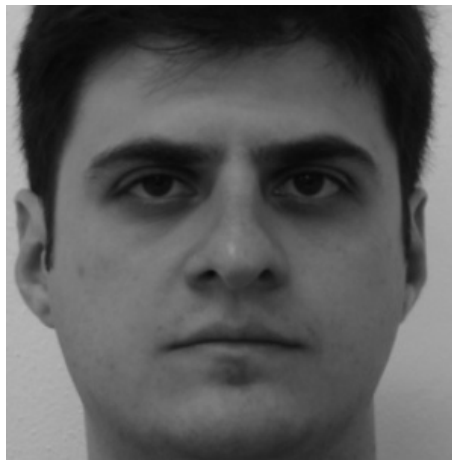


Figura 3.7: Face tratada pertencente ao treino.

notada na figura 3.8.

Cada face  $\Gamma$  pode ser subtraída da face média  $\Psi$  para assim formar um vetor de diferenças  $\Phi$  presente na equação 3.2, sendo estas as diferenças mais distintas de cada face, essa diferença pode ser notada nas figura 3.9.

$$\Phi_i = \Gamma_i - \Psi \quad (3.2)$$

Os autovetores e autovalores podem ser obtidas a partir da matriz de covariância do vetor  $AA^T$  como pode ser visto na equação 3.3, sendo este um vetor de  $\Phi$ , porém esta matriz possui um tamanho que pode ser um desperdício computacional ( $N^2 \times N^2$ ) então é necessário fazer o cálculo a partir da matriz transposta  $A^T$  que é computacionalmente menor ( $M \times M$ ).

$$C = \frac{1}{M} \sum_{i=1}^M \Phi_i \Phi_i^T = AA^T \quad (N^2 \times N^2) \quad (3.3)$$

Sendo assim é necessário calcular os autovetores a partir de  $A^T A$  e achar os autovetores de  $AA^T$  a partir da relação da equação 3.4.

$$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow C A v_i = \mu_i A v_i \Rightarrow \text{onde} \rightarrow u_i = A v_i \quad (3.4)$$



Figura 3.8: Face média  $\Psi$ .

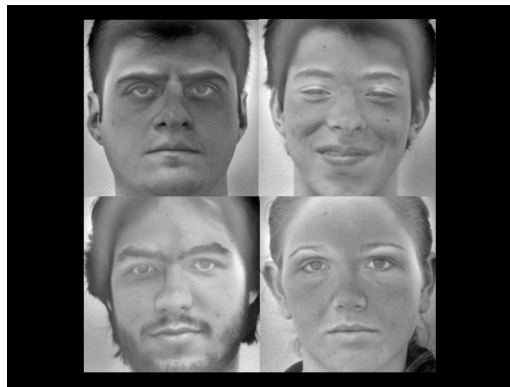


Figura 3.9: Faces  $\Phi$ .

Sendo assim é possível encontrar os autovetores da matriz de covariância  $AA^T$ , sendo estes os maiores autovetores de dos autovetores de  $A^T A$ . Os autovetores podem ser referenciados como as eigenfaces que devem ser redimensionadas para que seja possível ser visualizadas em uma imagem. Alguns exemplos de eigenfaces obtidas podem ser vistas na figura 3.10.

Com as eigenfaces geradas é necessário calcular os pesos de cada eigenface nas diferentes faces do treinamento, os pesos calculados para cada face são obtidos utilizando as eigenfaces transpostas vezes a face  $\Phi$  como na equação 3.5.

$$w_j = u_j^T \Phi_i \quad (3.5)$$

Os pesos  $w_j$  são distintos para cada face, esses pesos mais a face média  $\Psi$ , resulta na face original  $\Phi$ , contudo essa reconstrução pode não ocorrer do jeito esperado, já que cada rosto possui uma forma própria mesmo que elas sejam alinhadas. Para representar as faces no "espaço de faces" é necessário reunir cada peso  $w_j$  de cada face em um vetor  $\Omega$  como na equação 3.6





Figura 3.10: Eigenfaces.

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \\ \vdots \\ w_K^i \end{bmatrix}, \quad i = 1, 2, \dots, M \quad (3.6)$$

### 3.4 Classificação Eigenfaces

Para os cálculos e processos para classificação das eigenfaces, a biblioteca Numpy foi utilizada. As imagens obtidas previamente são tratadas como vetores numéricos, para poder ser utilizado pela Numpy. Com isso é necessário obter a nova imagem para teste e refazer todos os processos anteriores para normalizar (segmentar a face da imagem, transformar em um canal de escala de cinza, transformar em um vetor  $\Gamma$ ) a imagem para estar igualmente com as imagens do conjunto de treinamento. A nova face não possui os pesos calculados em relação as eigenfaces do sistema, sendo assim é necessário fazer este cálculo dos pesos  $\Omega$  para fazer o reconhecimento se a face pertence ou não ao sistema. Este processo é uma representação da face nova, de fora do conjunto de treinamento, no "espaço de faces" normalizando ela com a face média  $\Psi$  como na equação 3.7 (a nova face já deve estar no formato de  $\Gamma$  para ser representada).

$$\hat{\Phi} = \Gamma - \Psi \quad (3.7)$$

Para a projeção das eigenfaces é necessário calcular o vetor de pesos  $\Omega$  desta face  $\Gamma$ , utilizando as eigenfaces obtidas anteriormente, essa representação pode ser vista na equação 3.8

$$w_i = u_i^T \hat{\Phi} \quad \Omega = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_K \end{bmatrix} \quad (3.8)$$

Calculando o vetor  $\Omega$  da nova face, é necessário achar o menor erro euclidiano entre os pesos da nova face menos os pesos de cada face do conjunto de treinamento. Este erro deve ser o menor possível e estar abaixo de um valor de *threshold* para ser considerado a

mesma face, a equação 3.9 representa essa distância da face nova de cada face do conjunto de treino.

$$e_{\Omega} = \min_l \|\Omega - \Omega^l\| \quad (3.9)$$

O valor de *Threshold* pode ser escolhido arbitrário ou perante um critério de cada aplicação, segundo (SLAVKOVIC; JEVTIC, 2012), não há uma formula para o cálculo de um valor de *Threshold*, o que se pode fazer é calcular o erro mínimo da imagem de teste com o treinamento, obtido a partir da fórmula 3.9, e pode-se calcular utilizando a equação 3.10. Contudo este valor máximo acaba sendo um valor distante dos demais valores de erros da fórmula 3.9, não podendo ser utilizado para a comparação.

$$\theta = 0,8 * \max(rast) \quad (3.10)$$

Para o reconhecimento de uma face, as seguintes considerações foram feitas utilizando os valores de *Threshold* obtidos a partir de experimentação:

- Se o menor valor de  $e_{\Omega} < \theta_{\Omega}$  a face pode pertencer ao sistema;
- Se o menor valor de  $e_{\Phi} < \theta_{\Phi}$  a face é pertencente ao sistema;
- Se o valor de  $e_{\Omega}$  considera uma face diferente de  $e_{\Phi}$ , o valor de  $e_{\Phi}$  é levado em consideração;
- Se o menor valor de  $e_{\Omega} \geq \theta_{\Omega}$  a face não pertencer ao sistema;

As faces que são reconhecidas como não pertencentes ao sistema, podem ser adicionadas posteriormente ao sistema, recalculando a face  $\Psi$  e as eigenfaces com esta face nova.

### 3.5 Desenvolvimento da Interface

A interface do sistema foi desenvolvida utilizando a biblioteca Tkinter, sendo esta uma das bibliotecas padrões para desenvolvimento de interfaces, possui integração com a maioria dos sistemas operacionais Unix e Windows, fazendo com que seja possível possuir uma interface compatível com uma grande gama de sistemas operacionais. A interface pode ser dividida dentro dos dois processos que o presente trabalho foi dividido.

#### 1. Supervisão do Sistema

A interface da Supervisão do Sistema possui o intuito de manter as funções necessárias para o funcionamento correto do sistema podendo ser acessado somente pelo administrador do sistema, devido ser necessário fazer um login de acesso a esta tela, com isto nesta parte do sistema pode-se cadastrar novas faces ao banco de faces, cadastrar novos aula/eventos, verificar os estados das presenças.

- Cadastro de faces

Dentro da fase do reconhecimento facial, existe a possibilidade que uma face não seja reconhecida como pertencente ao sistema, sendo assim se faz necessário cadastrar o indivíduo no sistema, sendo assim a função para cadastro de novas faces possibilita a aquisição das imagens da face diretamente a partir da webcam integrada ao sistema, sendo

necessário obter um segmento de imagens e escolher a melhor entre elas. A imagem escolhida passa por um processo de normalização, para ser armazenada no banco de dados junto com as suas informações.

Devido ao cadastro de uma nova face ao banco de faces, se faz necessário recalculer a face  $\Psi$  juntamente com os pesos  $\Omega$  da conjunto de treinamento, o conjunto de treinamento é composto de todas as faces do sistema, onde é necessário sempre fazer os cálculos previamente para a utilização no reconhecimento, após os cálculos é necessário atualizar as informações no banco de dados e no Google drive.

- Cadastro de aulas e eventos

Com o reconhecimento da face de um indivíduo, deve-se colocar a sua presença em uma tabela que posteriormente é armazenada no banco de dados, assim é necessário que o administrador do sistema cadastre os eventos e aulas previamente, nesta parte do sistema o administrador pode colocar o nome da aula e do evento, a data.

- Verificar presença

Como existe a possibilidade de um indivíduo não estar cadastrado no sistema ou acabar sendo confundido com outro indivíduo, o administrador do sistema pode analisar a presença que foi atribuída no momento que um indivíduo foi ou não reconhecido, para que se necessário, retirar ou modificar a presença manualmente.

## 2. Atribuição de presença

A tela de atribuição de presença pode ser considerado como a tela principal do sistema, devido esta tela ser executada na maioria do tempo em contrapartida da tela de configuração. Nesta parte do sistema são utilizadas as funções de aquisição das imagens, processamento da imagem adquirida, reconhecimento e cadastro da presença.

A interface possui em sua tela principal a visualização da visão da câmera integrada ao sistema, quando uma face é encontrada pelo sistema, a mesma é ressaltada na tela principal além das informações do indivíduo da qual a face o sistema reconheceu.

## 3.6 Integração Banco de Dados

Dentro do desenvolvimento do trabalho proposto, foi necessário criar um meio de armazenar e recuperar as imagens e valores utilizados pelo sistema. No início da aplicação foram utilizadas as imagens localmente, sendo armazenadas e utilizadas somente durante a execução do sistema. Contudo este tipo de solução pode acabar esgotando o armazenamento interno do dispositivo que roda o sistema, sendo assim para que este problema fosse sanado, a forma de armazenamento das informações foi dividido em duas formas, sendo uma delas para o armazenamento das informações e outra para o armazenamento dos arquivos.

As imagens que são utilizadas pelo sistema projetado deve ser alocado em um local que seja fácil buscar e armazenar, sendo assim foi pensado em utilizar um banco que tivesse uma liberdade de inserir as informações das imagens sem a necessidade de um banco complexo. Sendo assim o banco de dados MongoDB foi utilizado, sendo o MongoDB um banco de dados não relacional que utiliza documentos para armazenar informações, utilizando informações na linguagem JSON. Podendo se conectar com as

mais diversas linguagens de programação (como por exemplo: Java, Python, Ruby, C++, etc). Pode ser utilizado com repositórios locais ou até mesmo com repositórios armazenados em nuvem de forma gratuita (conforme utilização do armazenamento e acesso repetidos dos dados). Dentro de algumas de suas vantagens, pode-se elencar a relação de não ser necessário as condições relacionais para começar a armazenar dados, obtendo melhor performance em relação as suas consultas, já que tudo esta dentro de um único documento. Contudo isso acaba sendo uma desvantagem também, pois caso seja necessário a modificação de algum atributo para todos as entradas, cada valor deve ser tratado um a um. Devido a quantidade de arquivos que são necessários para o funcionamento do projeto, as informações dos arquivos foram salvos no MongoDB e os arquivos propriamente ditos foram armazenados no Google Drive, sendo este um serviço de armazenamento e sincronização de arquivos desenvolvido pela empresa Google.

Para o acesso de ambos os bancos é necessário a configuração e utilização de arquivos de configuração, que contenham os usuários e senhas para acesso. Para o acesso do repositório do MongoDB é necessário repassar a linha de conexão do repositório. Para a conexão do Google Drive é necessário a configuração e o aceite na conta do usuário que irá hospedar os arquivos, neste caso foi configurado o arquivo *settings.yaml* para que não fosse necessário abrir uma janela de aceite a cada vez que o sistema fosse executado.

Para o armazenamento das informações no MongoDB, o seu acesso é feito a partir da biblioteca *Pymongo*, onde cada face tem as seguintes informações armazenadas:

1. ID do Mongo;
2. Nome da Pessoa;
3. ID do Google Drive.

O *ID do Mongo* é um identificador que é gerado automaticamente, sendo uma chave primária igual a de um banco de dados relacional, onde este é um identificador único. O *ID do Google Drive* é o identificador único que é gerado automaticamente quando o arquivo com as informações da imagem da face são salvos. Para o armazenamento dos arquivos no Google Drive, o seu acesso é feito a partir da biblioteca *Pydrive*, onde cada arquivo tem as seguintes informações armazenadas:

- Valor de  $\Phi$  da face ou valor de  $\Psi$  da face ou valor das eigenfaces;

Os valores são armazenados em um arquivo *JSON*, sendo o *JSON* uma notação de objetos JavaScript, onde é um formato de texto independente de linguagem de programação, pode ser interpretado pelas mais diversas linguagens. Para o armazenamento dos valores poderiam ser utilizadas outras formas de salvar o arquivo, contudo foi escolhido o arquivo *JSON* para a facilitação ao acesso das informações, já que as mesmas são armazenadas em formato de lista, sendo mais fácil de fazer a transformação para vetores da biblioteca *Numpy*.

### 3.7 Atestado de Presença

O processo de atestado de presença de um aluno ocorre sempre que sua face já foi tratada pelo sistema, contudo existem duas possibilidades dentro do sistema, que são tratadas de formas diferentes, sendo elas:

1. Inserção da presença no sistema

Após a aquisição da imagem, segmentação da face do aluno, cálculo de seus pesos  $\Omega$  e classificação de sua face, a presença é atribuída ao aluno se caso ele seja pertencente ao banco de faces do sistema.

## 2. Análise de presença pelo administrador

A análise da presença ocorre caso o aluno não seja reconhecido pelo sistema, sendo assim se faz necessário que o aluno coloque manualmente o seu nome no sistema, com isso sua presença pode ser computada e um log é gerado para o administrador do sistema, com as informações inseridas pelo aluno, juntamente com a foto adquirida no momento em que a presença foi realizada pelo sistema.

Para fins de registro e documentação, as presenças realizadas pelo sistema podem ser acessadas e baixadas pelo administrador do sistema.

### 3.8 Integração do Sistema

Na sua etapa final, todas as partes desenvolvidas serão integradas em uma versão final, já que ao longo do desenvolvimento do sistema as partes serão testadas para não ter nenhum tipo de problema na integração dos mesmos. O funcionamento do sistema pode ser visto na fig. 3.11.

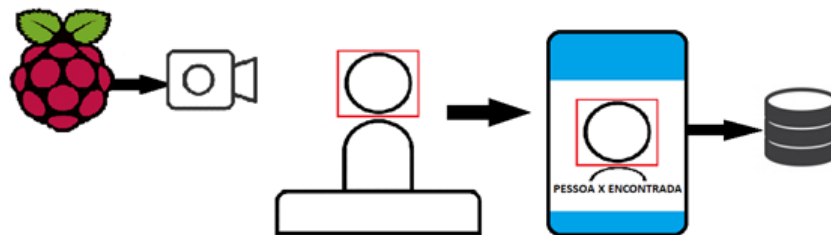


Figura 3.11: Diagrama do sistema.

Com isso o sistema final possui duas formas de acesso, a de administrador e a de usuário, possui funções de cadastro de face, de dados e de presença,

## 4 RESULTADOS E EXPERIMENTOS

Nesta seção serão discutidos os resultados obtidos a partir da aplicação construída e das eigenfaces obtidas.

### 4.1 Eigenfaces

Durante o teste de reconhecimento a partir das eigenfaces calculadas previamente, foram utilizadas 50 imagens (100 imagens totais, devido ao espelhamento de cada imagem) de faces diferentes que são pertencentes as da base de treinamento, porém possuem um distúrbio em cada imagem (o distúrbio da imagem são as modificações do rosto de cada pessoa, pois estão sorrindo). Para a classificação a fórmula 3.9 foi utilizada, obtendo um acerto de 80% das faces testadas, utilizando somente o erro mínimo para a seleção da face. Assim com o valor de erro de cada face obtido, foi necessário procurar qual a face que havia obtido o menor erro, os resultados podem ser observados na Figura 4.1

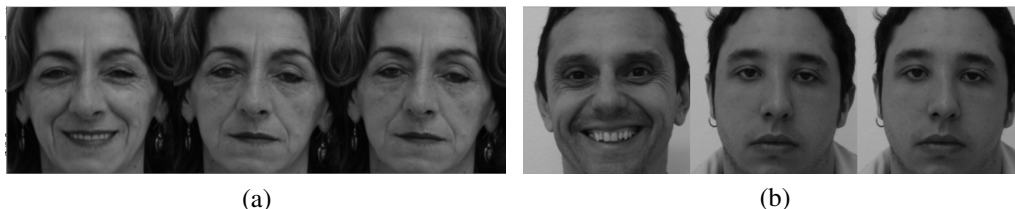


Figura 4.1: Face de teste, Face reconhecida do banco e Face reconhecida do sistema: (a) Face que foi reconhecida, (b) Face não foi reconhecida

A primeira face da Figura 4.1(a) é a face que foi utilizada como teste de reconhecimento, a segunda e terceira face são as faces que foram reconhecidas como as que possuíam o menor valor de erro, sendo a segunda obtida do banco de imagens armazenadas no banco de dados e a terceira a armazenada na execução do algoritmo. Porém a Figura 4.1(b) mostra uma face de teste que foi reconhecida erroneamente com outra face da base de treinamento.

Dentro dos testes de reconhecimento realizados com o algoritmo, foram utilizados diversas quantidades de eigenfaces, pois segundo (TURK; PENTLAND, 1991), utilizar somente as  $K$  melhores eigenfaces esta diretamente relacionado a eficiência computacional, já que o tamanho do banco de imagens que é utilizada pelo sistema pode aumentar conforme a necessidade de cada sistema. Neste trabalho foram utilizadas 196 imagens para o total da base de treinamento. A Figura 4.2 exemplifica a relação da utilização de  $K$  para a quantidade de acertos e a Figura 4.3 a relação de  $K$  para erros. A quantidade de

acertos e erros foram computadas observando manualmente cada imagem do teste com o as imagens da base de treinamento, um exemplo desta validação esta na Figura 4.1, onde a imagem era comparada manualmente caso obtivesse um um resultado satisfatório, foi computado como um acerto e caso não como um erro.

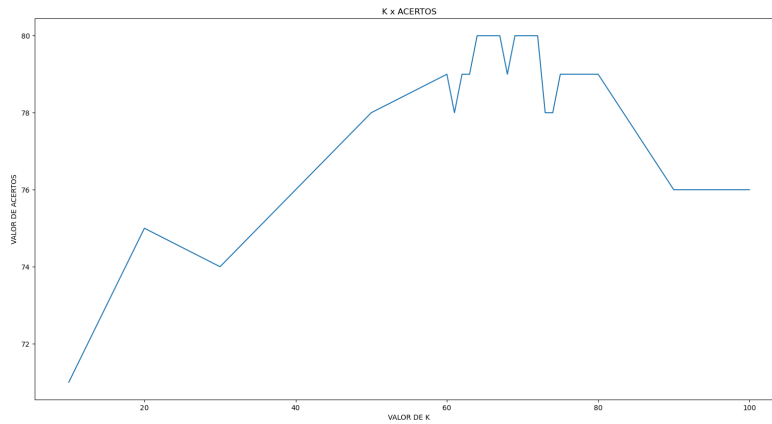


Figura 4.2: Gráfico K x Acertos.

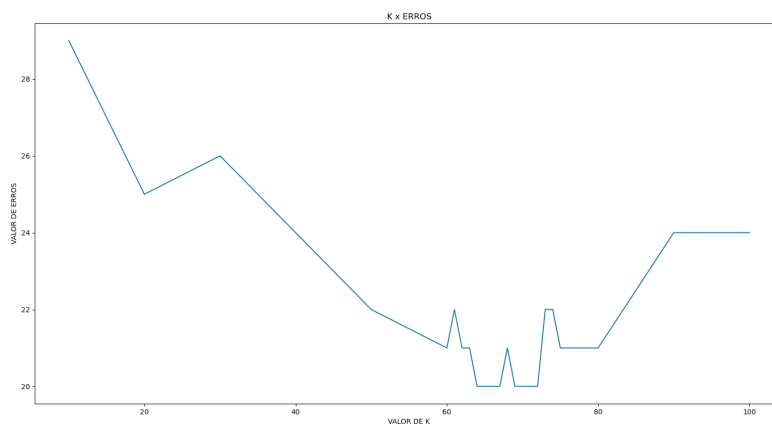


Figura 4.3: Gráfico K x Erros.

Com os resultados de  $K$ , observou-se que um dos melhores valores para a utilização para o algoritmo foi o de  $K = 70$ , sendo assim foram utilizadas as melhores 70 eigenfaces obtidas previamente no processo de reconhecimento. Os valores dos omegas de cada imagem de teste projetada no espaço de faces, possui um valor de desvio padrão que aparece estar próximo das faces que possuem características semelhantes. O efeito do desvio padrão pode ser notado na Figura 4.4.

Este efeito pode ser notado também no desvio padrão das imagens utilizadas no conjunto de treinamento, onde cada face que possui características semelhantes, acabam tendo um valor mais próximo do outro e fazendo um agrupamento no gráfico da Figura 4.5.

Contudo durante os testes para obter o melhor valor de  $K$ , notou-se que algumas faces estavam sendo comparadas e obtido o menor valor mas o rosto correto não estava dentro

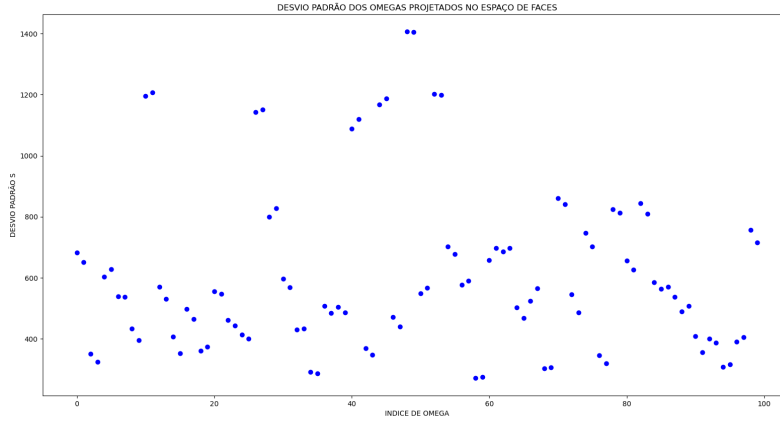


Figura 4.4: Desvio Padrão dos Omegas projetados no espaço de faces.

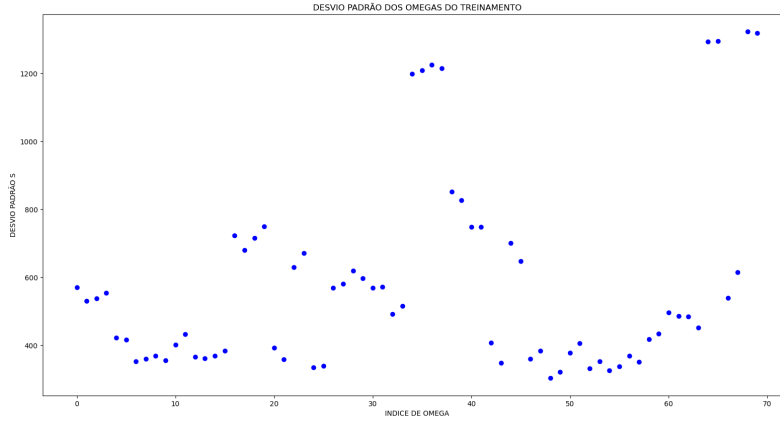


Figura 4.5: Desvio Padrão dos Omegas no espaço de faces.

do menor valor da fórmula 3.9, sendo assim foi necessário utilizar outro valor para ser uma segunda forma de comparação e validação. Com isso a fórmula 4.1 foi utilizada como forma de validação, onde o menor valor foi utilizado.

$$e_{\Phi} = \min \|\hat{\Phi} - \Phi_i\| \quad (4.1)$$

Com a utilização da fórmula 4.1, notou-se que houve um aumento de 10% de acertos em comparação com o teste anterior, este teste foi feito olhando cada imagem do teste comparada com a imagem do treinamento, validando manualmente, mas se fez necessário utilizar uma forma de fazer este reconhecimento automaticamente, utilizando os valores de *Erro mínimo de  $\Omega$*  e o *Erro mínimo de  $\Phi$* , sendo assim a Figura 4.6 possui uma comparação das faces reconhecidas manualmente com as faces reconhecidas sem nenhum tipo de verificação.

Cada valor de *Threshold de Phi* foi testado reconhecendo cada face novamente, para poder obter um resultado mais próximo do valor que foi reconhecido manualmente. Pode-se notar que em cada figura, existe uma diferença de erro entra a o reconhecimento manual e o sem supervisão, pode-se dizer que este efeito de falso e verdadeiro negativo.

Com os gráficos das Figuras 4.6 a 4.11 pode-se notar que as imagens possuem um



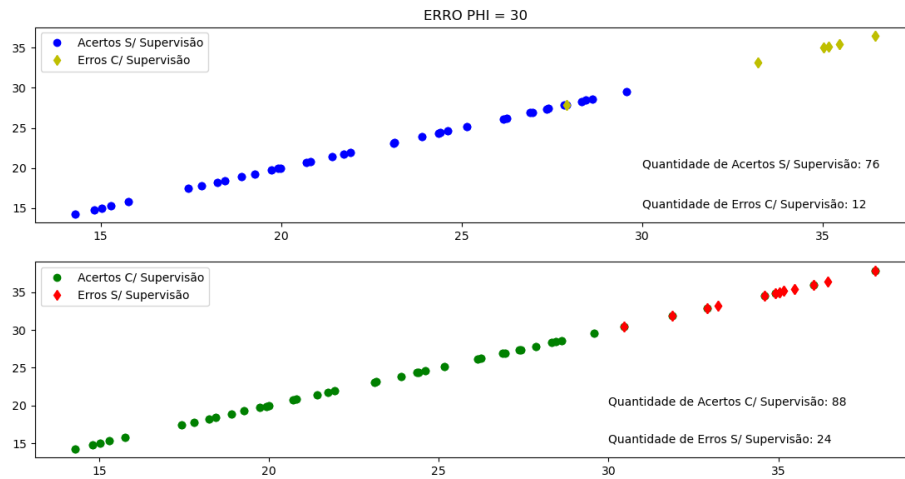


Figura 4.6: Faces reconhecidas do treinamento utilizando Erro  $\Phi = 30$ .

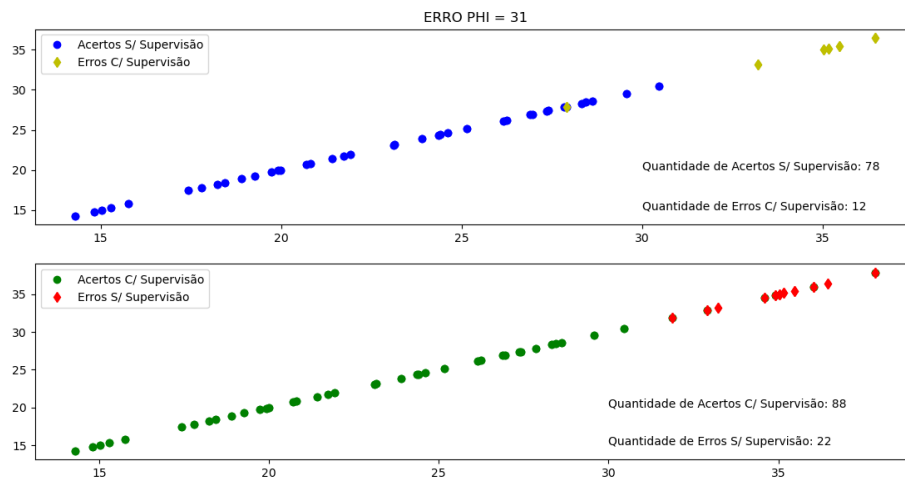


Figura 4.7: Faces reconhecidas do treinamento utilizando Erro  $\Phi = 31$ .

valor de *Erro Phi* abaixo de 30. Para poder obter uma confirmação neste valor, o teste que foi feito para garantir este valor é tentar reconhecer faces que estão fora das faces de treinamento. Os gráficos dos teste estão referenciados nas Figuras 4.12 a 4.17. Analisando cada gráfico pode-se notar que faces que são desconhecidas do sistema possuem um valor de *Erro Phi* acima de 30. Com isso é possível utilizar um valor entre 30 e 35 como *Threshold*.

A decisão foi feita com base em minimizar o efeto de falsos e verdadeiros negativos, já que com o aumento do valor, existe mais possibilidade de uma face ser atribuída erroneamente com outra.

O valor utilizado de trinta obtem também um erro atrelado as faces desconhecidas do sistema, onde erroneamente acaba considerando algumas faces como pertencentes ao conjunto de treinamento. A quantidade de erros diminui juntamente com a diminuição do valor de menor erro de  $\Phi$ , assim isto mostra que a fórmula 4.1 reproduz a diferença de cada face uma das outras, tendo a diferença de PCA de cada face.

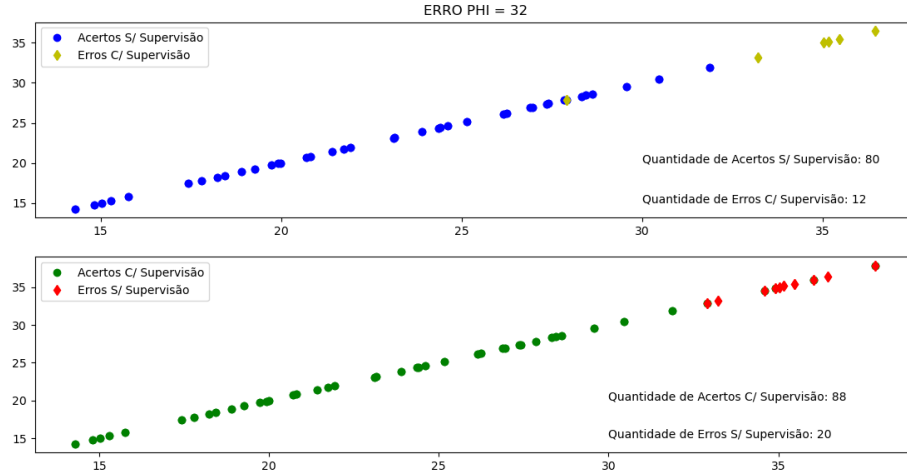


Figura 4.8: Faces reconhecidas do treinamento utilizando Erro  $\Phi = 32$ .

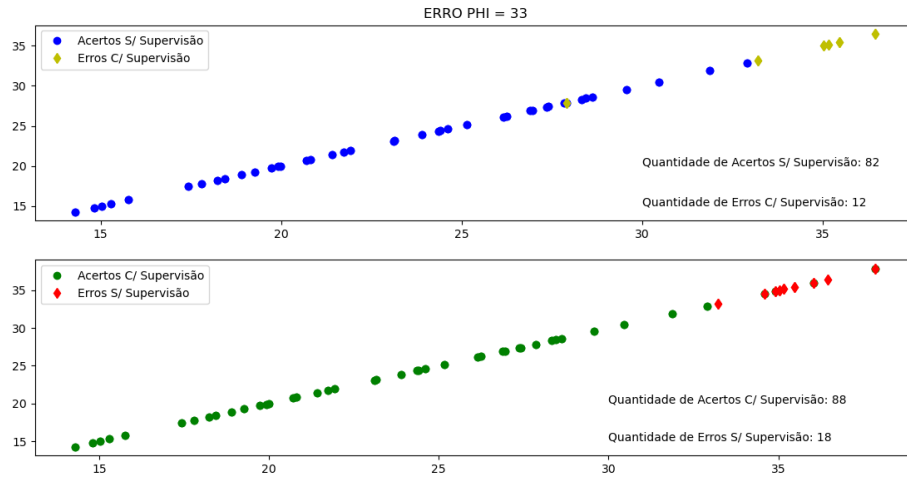


Figura 4.9: Faces reconhecidas do treinamento utilizando Erro  $\Phi = 33$ .

Pode-se elencar que algumas faces acabam sendo consideradas como pertencentes ao conjunto de treinamento devido a igualdade de certas características humanas, sendo este um fator que deve ser levado em consideração para fazer uma análise de grupos de faces.

Para se fazer uma análise dos valores de falsos e positivos negativos que podem ocorrer com o aumento do valor de *Erro Phi*, esses valores podem ser notados na tabela 4.1, onde **FN** é a quantidade de valores de "*Falsos Negativos*", **FP** é a quantidade de valores de "*Falsos Positivos*", **Iguais** é a quantidade de valores que são iguais aos valores comparados com o valor de acertos supervisionados, **Acertos** é a quantidade de valores que o sistema acertou de faces sem supervisão e **Erros** é a quantidade de valores que o sistema errou de faces sem a supervisão do sistema.

Pode-se notar que conforme o valor de  $e_\Phi$  vai aumentando, a quantidade de falsos negativos tende a diminuir, sendo assim pode-se notar que escolher somente um valor de *Threshold* é uma limitação, pode utilizar ao invés de somente um valor ser uma faixa, sendo assim a faixa vai de 30 a 35. Com todos os valores de *Threshold*, face  $\Psi$  e eigenfa-

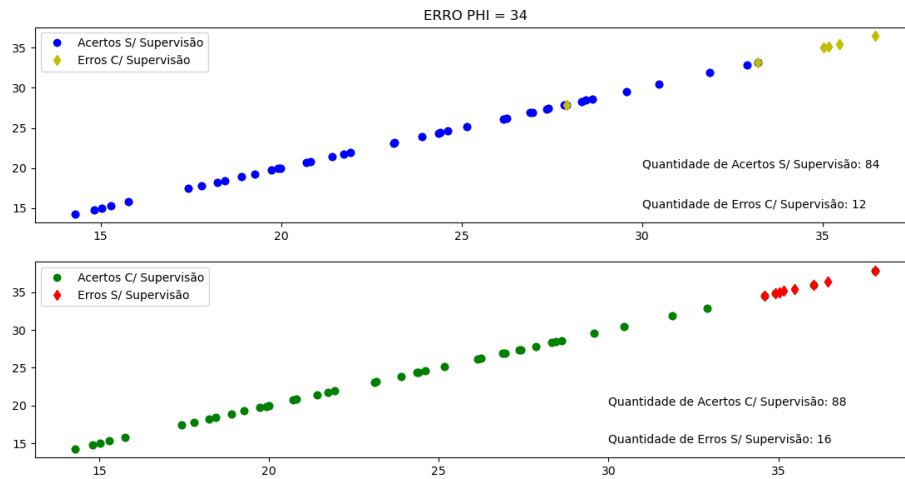


Figura 4.10: Faces reconhecidas do treinamento utilizando Erro  $\Phi = 34$ .

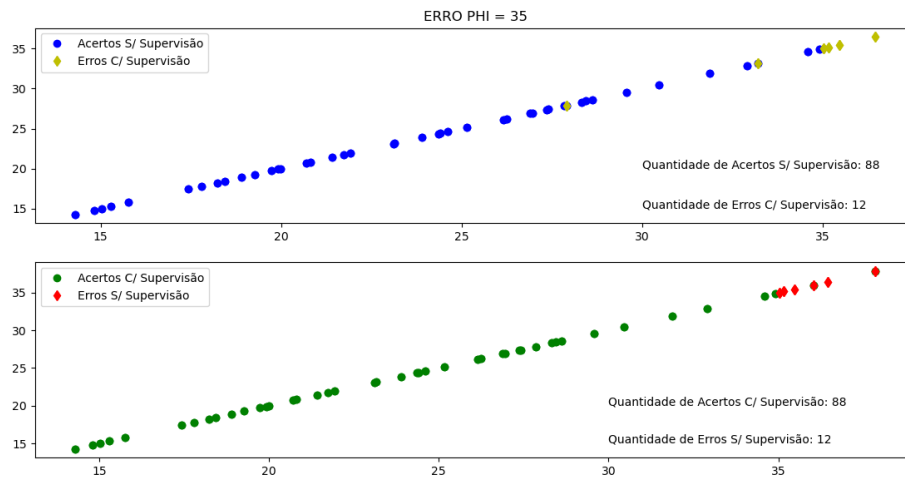


Figura 4.11: Faces reconhecidas do treinamento utilizando Erro  $\Phi = 35$ .

ces, pode-se utilizar isto para a verificação no sistema.

## 4.2 Armazenamento

Dentro do desenvolvimento do trabalho proposto, foi necessário criar um meio de armazenar e recuperar as imagens e valores utilizados pelo sistema. O acesso das imagens feito localmente não possui um atraso para a obtenção de cada imagem, contudo caso exista algum tipo de perda do sistema por meios físicos, as imagens serão perdidas e será necessário reconfigurar o sistema. Com isso em mente o armazenamento externo foi pensado, sendo esta uma forma mais prática de salvar as informações do sistema, contudo existe o atraso para a recuperação das informações que foram salvas externamente.

O processo de armazenamento em ambos os bancos ocorre em três etapas, sendo elas: Gerar as informações da face (valores de  $\Phi$  e  $\Omega$  de cada face individualmente), salvar essas informações no Google Drive e devolver o ID que foi salvo e salvar no Mongo o

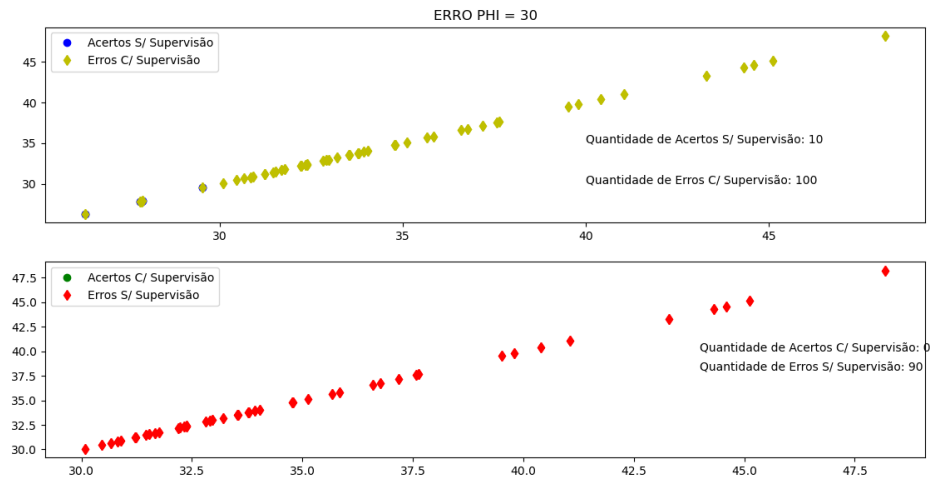


Figura 4.12: Faces reconhecidas de fora do treinamento utilizando Erro  $\Phi = 30$ .

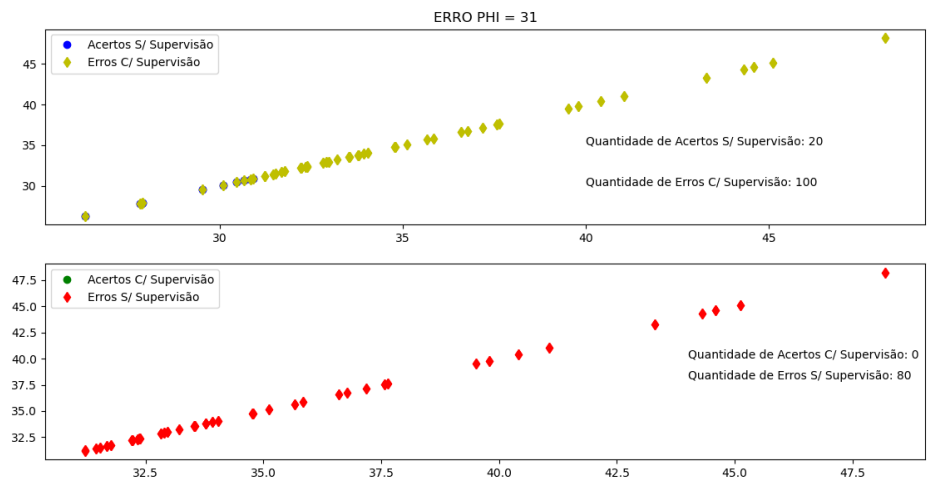


Figura 4.13: Faces reconhecidas de fora do treinamento utilizando Erro  $\Phi = 31$ .

ID do Google Drive juntamente com o nome de cada pessoa. Um exemplo deste processo pode ser notado na Figura 4.18.

Para a análise de uma nova face, é necessário buscar os valores de cada face no banco de imagens, a forma de busca é feita a partir dos valores de  $\Omega$ , que foram armazenados previamente, e devolvido o valor de  $\Phi$  da face reconhecida (deve-se após receber o valor de  $\Phi$  somar o valor de  $\Psi$  que também está armazenado no banco de imagens). O método de utilização de ambos os bancos pode ser notado na Figura 4.19

Para poder demonstrar que existe um atraso para a recuperação das imagens armazenadas o seguinte experimento foi feito: as imagens são armazenadas no Google Drive e o identificador único atrelado ao arquivo é gerado, este identificador é salvo posteriormente no banco de dados do MongoDB. Os arquivos foram divididos em três classes sendo elas, a face média  $\Psi$  (contém somente um arquivo), as eigenfaces (contém setenta arquivos diferentes) e as faces  $\Phi$  (contém cento e noventa e seis arquivos diferentes). O sistema recupera cada imagem a partir de seu identificador, com isso a tabela 4.2 foi gerada.

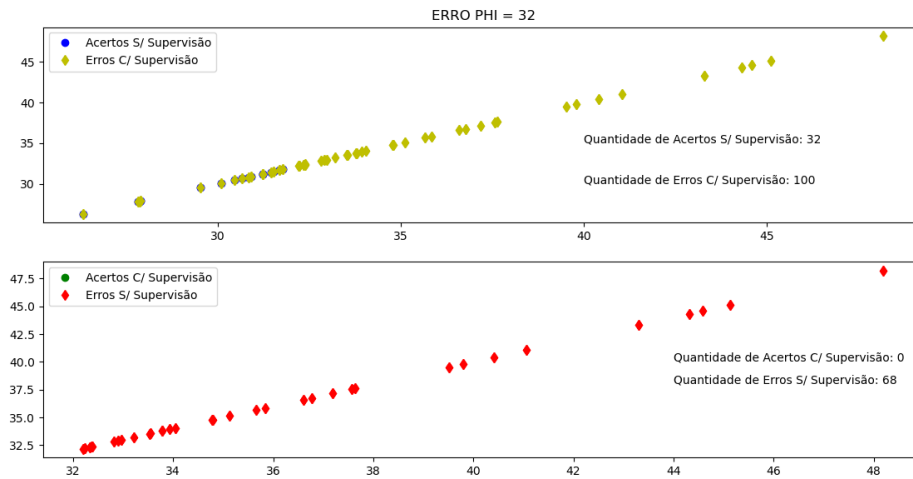


Figura 4.14: Faces reconhecidas de fora do treinamento utilizando Erro  $\Phi = 32$ .

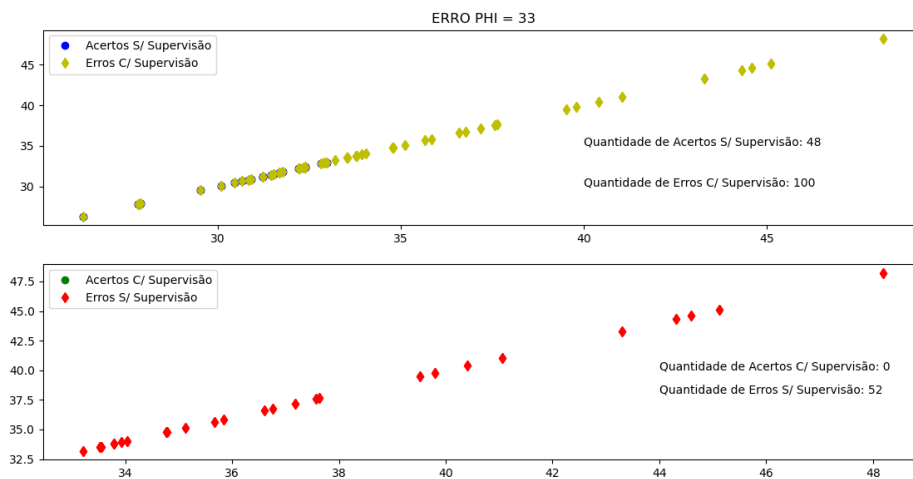


Figura 4.15: Faces reconhecidas de fora do treinamento utilizando Erro  $\Phi = 33$ .

Os tempos de aquisições dos arquivos estão diretamente atrelado ao acesso ao Google Drive, devido que o tempo para a obtenção de todos os identificadores que estão armazenados no MongoDB acaba não levando mais do que um segundo. Com um total de 635 segundos para obter todos os arquivos necessários para o funcionamento do sistema, estes arquivos devem ser carregados na inicialização do sistema e ser utilizado localmente, devido a esta quantidade de tempo necessária para fazer a requisição dos arquivos, pois para cada face que irá ser reconhecida é necessário todos os arquivos armazenados, sendo assim o tempo total mais o tempo para o reconhecimento seria o tempo total que o usuário deve esperar pra poder ser reconhecido.

### 4.3 Interface

Para a utilização das eigenfaces, foi desenvolvido uma interface utilizando a biblioteca Tkinter. O objetivo da interface é mostrar para o usuário que a sua face esta sendo cap-

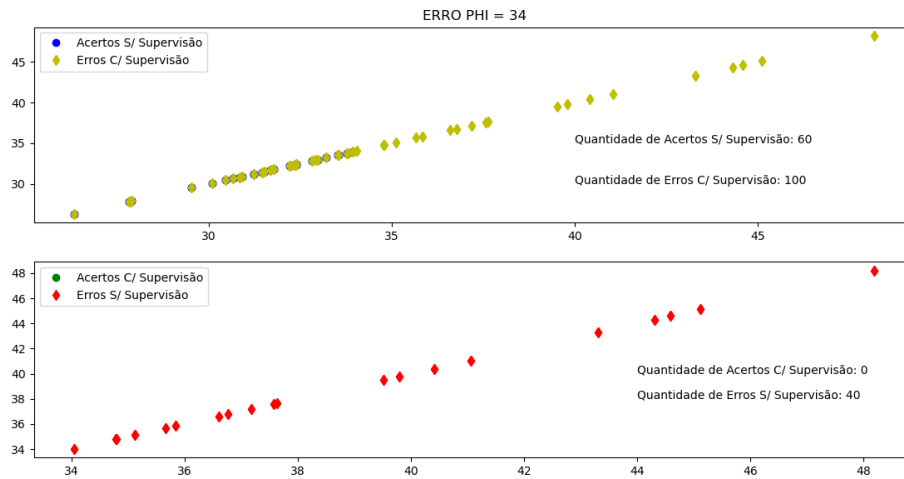


Figura 4.16: Faces reconhecidas de fora do treinamento utilizando Erro  $\Phi = 34$ .

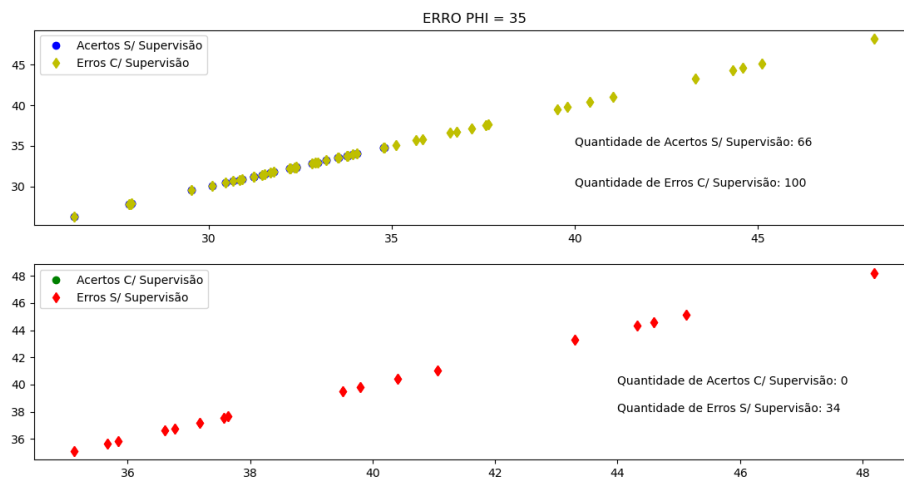


Figura 4.17: Faces reconhecidas de fora do treinamento utilizando Erro  $\Phi = 35$ .

turada e sendo calculado se a sua face será reconhecida, devido ao fato de que as pessoas que serão submetidas ao sistema tem o interesse de serem reconhecidas.

FN	FP	Iguais	Acertos	Erros
$e_{\Phi} = 30$				
14	2	84	76	24
$e_{\Phi} = 31$				
12	2	86	78	22
$e_{\Phi} = 32$				
10	2	88	80	20
$e_{\Phi} = 33$				
8	2	90	82	18
$e_{\Phi} = 34$				
8	4	88	84	16
$e_{\Phi} = 35$				
4	4	92	88	12

Tabela 4.1: Falsos positivos e negativos



Figura 4.18: Armazenamento das imagens a partir dos Bancos de Dados.

	$\Psi$	<b>Eigenfaces</b>	$\Phi$
Qnt Arquivos	1	70	196
Tempo Gasto	2,2s	157s	475s

Tabela 4.2: Tempo de recuperação das imagens

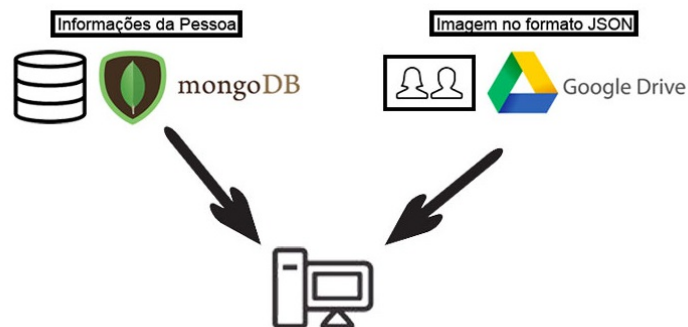


Figura 4.19: Recuperação das imagens a partir dos Bancos de Dados.

## **5 CONCLUSÃO**



## 6 TRABALHOS FUTUROS

Como trabalhos futuros, existe a possibilidade de utilizar Redes Neurais para poder modificar cada valor de  $\Omega$ , podendo assim reconhecer uma face diretamente pelo seu peso atrelado, sem necessidade de utilizar somente o valor de um *Threshold* mínimo e recalculá-lo para cada pessoa o seu peso da face.

## REFERÊNCIAS

GONZALEZ, R. C. **Processamento Digital de Imagens**. Av. Ermano Marchetti, 1453, São Paulo, São Paulo: Pearson Prentice Hall, 2010.

MISHRA, S. P.; SARKAR, U.; TARAPHDER, S.; DATTA, S.; SWAIN, D. P.; SAIKHOM, R.; PANDA, S.; LAISHRAM, M. **Multivariate Statistical Data Analysis-Principal Component Analysis (PCA)**. 2017.

SIROVICH, L.; KIRBY, M. **Low-dimensional procedure for the characterization of human faces**. 1987.

SLAVKOVIC, M.; JEVTIC, D. **Face Recognition Using Eigenfaces Approach**. 2012.

TURK, M.; PENTLAND, A. P. **Eigenfaces for Recognition**. 1991.