

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. Adriane Parraga
Orientadora

Prof. Dr. João Leonardo Fragoso

Profa. Dra. Letícia Vieira Guimarães

Prof. Dr. Roberto Ribeiro Baldino

"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

LISTA DE ABREVIATURAS E SIGLAS	7
LISTA DE FIGURAS	8
LISTA DE TABELAS	9
RESUMO	10
ABSTRACT	11
1 INTRODUÇÃO	12
2 FUNDAMENTOS TEÓRICOS	13
2.1 PROCESSAMENTO DIGITAL DE IMAGENS	13
2.1.1 Imagens	13
2.2 Análise de Componentes Principais	14
2.3 EIGENFACES	15
2.3.1 Reconhecimento Utilizando Eigenfaces	15
2.4 SISTEMAS EMBARCADOS	16
3 MATERIAIS E MÉTODOS	17
3.1 MATERIAIS	17
3.2 PYTHON	18
3.3 OPENCV	18
3.4 MONGODB	18
3.5 EIGENFACES	18
3.6 METODOLOGIA	19
3.6.1 Aquisição da Imagem	19
3.6.2 Segmentação Facial	19
3.6.3 Classificação Eigenfaces	19
3.6.4 Desenvolvimento da Interface	19
3.6.5 Integração Banco de Dados	19
3.6.6 Atestado de Presença	19
4 RESULTADOS E DISCUSSÕES	21
4.1 Obtenção das Eigenfaces	21
4.1.1 Reconhecimento utilizando as Eigenfaces	24
4.2 Armazenamento	30
4.3 Interface	33

5	CONCLUSÃO	35
6	TRABALHOS FUTUROS	36
	REFERÊNCIAS	37

LISTA DE ABREVIATURAS E SIGLAS

PCA	Principal Component Analysis
PDS	Processamento Digital de Sinais
RNA	Rede Neural Artificial
SLIT	Sistemas Lineares e Invariantes no Tempo
FPA	Filtro Passa-Alta
FPB	Filtro Passa-Baixa
FIR	<i>Finite Impulse Response</i> (Resposta ao Impulso Finita)
IIR	<i>Infinite Impulse Response</i> (Resposta ao Impulso Infinita)
ADC	<i>Analog-to-digital converter</i> (Conversor Analógico-Digital)
TF	Transformada de Fourier
TDF	Transformada Discreta de Fourier
STFT	<i>Short Time Fourier Transform</i> (Transformada de Fourier de Tempo Reduzido)

LISTA DE FIGURAS

2.1	Esquemático do método de classificação.	13
3.1	Diagrama das etapas de desenvolvimento.	19
3.2	Diagrama do sistema.	20
4.1	Faces sem tratamento pertencentes ao treino.	21
4.2	Face tratada pertencente ao treino.	22
4.3	Face média Ψ	22
4.4	Faces Φ	23
4.5	Eigenfaces.	23
4.6	Face de teste, Face reconhecida do banco e Face reconhecida do sistema: (a) Face que foi reconhecida, (b) Face não foi reconhecida . . .	25
4.7	Gráfico K x Acertos.	25
4.8	Gráfico K x Erros.	26
4.9	Desvio Padrão dos Omegas projetados no espaço de faces.	26
4.10	Desvio Padrão dos Omegas no espaço de faces.	27
4.11	Faces reconhecidas do treinamento utilizando Erro $\Phi = 30$	27
4.12	Faces reconhecidas do treinamento utilizando Erro $\Phi = 31$	28
4.13	Faces reconhecidas do treinamento utilizando Erro $\Phi = 32$	28
4.14	Faces reconhecidas do treinamento utilizando Erro $\Phi = 33$	29
4.15	Faces reconhecidas do treinamento utilizando Erro $\Phi = 34$	29
4.16	Faces reconhecidas do treinamento utilizando Erro $\Phi = 35$	30
4.17	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 30$. . .	30
4.18	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 31$. . .	31
4.19	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 32$. . .	31
4.20	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 33$. . .	32
4.21	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 34$. . .	32
4.22	Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 35$. . .	33
4.23	Recuperação das imagens a partir dos Bancos de Dados.	34
4.24	Recuperação das imagens a partir dos Bancos de Dados.	34
4.25	Interface do sistema.	34

LISTA DE TABELAS

4.1	Falsos positivos e negativos	33
-----	--	----

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 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. O processo de funcionamento do presente trabalho segue a ordem indicada na figura 2.1

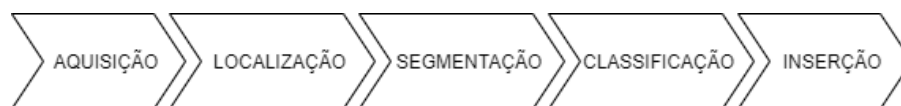


Figura 2.1: Esquemático do método de classificação.

- **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.);
- **LOCALIZAÇÃO:** O processo de localização 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.

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 digital são comumente chamados de *Pixels*.

2.2 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 eigenvectors e eigenvalues

Calcular os eigenvectors e eigenvalues da matriz de covariância, sendo os eigenvectors 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.

2.3 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".

2.3.1 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 Γ ($\Gamma = N^2 \times 1$);
- Calcular a face média Ψ do vetor Γ ;
- Obter a matriz de covariância C ;
- Obter os Eigenvalues e Eigenvectors da matriz de Covariância C ;
- Selecionar os melhores M eigenvectors;
- Manter somente os K melhores eigenvectors (K com maiores eigenvalues);
- Calcular os pesos Ω de cada face Φ 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 Ω 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;

2.4 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.

$$N = 300 \quad (2.1)$$

$$N^2 = 90000 \quad (2.2)$$

$$M = 196 \quad (2.3)$$

$$K = 50 \quad (2.4)$$

$$I \rightarrow N \times N \quad (2.5)$$

$$\Gamma \rightarrow N^2 \times 1 \quad (2.6)$$

$$\Psi \rightarrow N^2 \times 1 \quad (2.7)$$

$$\Phi_i \rightarrow N^2 \times 1 \quad (2.8)$$

$$C \rightarrow N^2 \times N^2 \quad (2.9)$$

$$A \rightarrow M \times N^2 \quad (2.10)$$

$$C_{mod} \rightarrow M \times M \quad (2.11)$$

$$u \rightarrow M \times N^2 \quad (2.12)$$

$$w_i \rightarrow 1 \times 1 \quad (2.13)$$

$$\Omega_i \rightarrow K \times 1 \quad (2.14)$$

$$\Omega_i = \lceil w_1 \rceil |w_2| \quad (2.15)$$

3 MATERIAIS E MÉTODOS

Neste capítulo serão abordados os materiais utilizados para a construção e desenvolvimento do presente trabalho, bem como os métodos discutidos anteriormente.

3.1 MATERIAIS

A aplicação será desenvolvida utilizando a linguagem de programação Python, sendo ela a linguagem escolhida devido ao seu maior acervo bibliográfico referente as funções necessárias para a o desenvolvimento de presente trabalho.

Para o tratamento e requisições das imagens, será utilizado a biblioteca OpenCV, devido ao seu alto grau de conceituação na literatura atual além de ser de código aberto e com uma ampla disponibilidade de documentação.

Para o tratamento dos vetores das imagens, será utilizado a biblioteca Numpy, sendo esta uma das bibliotecas que possui uma grande gama de operações que serão utilizadas para o tratamento e cálculo de cada imagem, além de ser difundida para o uso científico possui uma boa gama de documentação.

Para o armazenamento das informações necessária para gerar as presenças dentro da aplicação, será utilizado o banco de dados não relacional MongoDB, devido a não necessidade de relação entre as informações inseridas nas tabelas do banco criado, assim como a inserção das informações das imagens. Além de possuir conexão com a linguagem escolhida para a criação da aplicação. Para o desenvolvimento da interface gráfica, será utilizado a biblioteca de desenvolvimento de interfaces Tkinter.

O banco de imagens para o treinamento e o teste dos métodos previamente descritos, será o banco 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. Deste banco somente as imagens de face frontal foram utilizadas para o formar um conjunto de treinamento e teste, além de ser necessário cortar somente o rosto de cada indivíduo e transformar a imagem em um único canal de escala de cinza (as imagens originais possuem três canais).

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 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.

3.2 PYTHON

Python é 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. Possui versões para vários sistemas operacionais, tendo suporte para as mais diversas operações, possuindo uma grande gama de bibliotecas desenvolvidas pela própria comunidade que utiliza a linguagem. (FOUNDATION, 2001)

3.3 OPENCV

O OpenCV é 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++.

3.4 MONGODB

O MongoDB é 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.

3.5 EIGENFACES

Eigenface é um classificador de faces utilizado em visão computacional, utiliza auto-vetores para poder fazer a aproximação do valor de rosto médio com o rosto em questão. A criação do conceito de eigenfaces foi criada em 1987 por Sirovich e Kirby, tendo como primeira intenção de uma aproximação de uma representação de imagens em dimensões menores. As eigenfaces podem ser geradas a partir de um processo matemático chamado de Análise de Componente Principal (PCA) em um banco de imagens de faces consideravelmente grande. Pode-se utilizar as Eigenfaces para a classificação e reconhecimento facial, como feito por (TURK; PENTLAND, 1991), onde foram utilizadas para reconhecer faces de pessoas da equipe de pesquisa. Foram elencados certos problemas da utilização das mesmas, onde a iluminação, orientação e obstrução das imagens podem afetar diretamente no reconhecimento de cada face.

3.6 METODOLOGIA

Nesta seção serão apresentadas as etapas de desenvolvimento do presente trabalho seguindo o caminho da fig. 3.1.



Figura 3.1: Diagrama das etapas de desenvolvimento.

3.6.1 Aquisição da Imagem

3.6.2 Segmentação Facial

3.6.3 Classificação Eigenfaces

3.6.4 Desenvolvimento da Interface

3.6.5 Integração Banco de Dados

3.6.6 Atestado de Presença

O desenvolvimento do presente trabalho possui uma construção paralela das funções para que se possa ir integrando 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. As imagens serão adquiridas utilizando um algoritmo, contendo a biblioteca OpenCV, sendo executado na Raspberry Pi juntamente com uma câmera (podendo ser um webcam ou uma Raspicam).

Após a obtenção das imagens é necessário fazer a segmentação do rosto encontrado, considerando somente a face (do queixo até início da cabeça), em paralelo será construído e integrado o banco de dados com a interface, já que será necessário analisar como os dados das imagens adquiridas serão armazenadas no banco de dados (para que não seja

necessário armazenar nenhum tipo de informação na RaspberryPi, devido ao seu espaço limitado de armazenamento).

Assim que a face estiver segmentada, poderá ser aplicado o classificador Eigenfaces (a segmentação se faz necessária para que o fundo da imagem e qualquer outro objeto não influencie na classificação do indivíduo) para o reconhecimento do indivíduo em questão, como dito anteriormente, em paralelo será testado funções de frequência/presença no sistema, para que o objetivo final do sistema, que seria de por meio do reconhecimento facial a presença do indivíduo, da qual quer ser reconhecido, receberá presença dentro 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. 3.2.

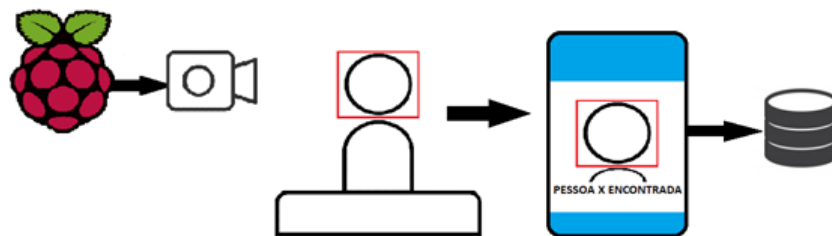


Figura 3.2: Diagrama do sistema.

4 RESULTADOS E DISCUSSÕES

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

4.1 Obtenção das Eigenfaces

A primeira etapa para o processo de obtenção das eigenfaces é obter um conjunto de imagens, estas imagens serão o grupo de treinamento um exemplo de algumas das imagens utilizadas nesta etapa pode ser notada na Figura 4.1, as imagens foram utilizadas para o treinamento, tiveram a face cortada, pois somente é necessário utilizar uma parte menor do rosto para o reconhecimento, sendo assim após o corte e transformação em um único canal de escala de cinza, para que a quantidade de cálculos diminuisse (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, para reduzir o tamanho dos cálculos as imagens foram redimensionadas no tamanho $N = 300 \times 300$ e possuem somente um canal em escala de cinza, um exemplo da imagem de treinamento pode ser vista na figura 4.2. As imagens podem ser normalizadas dividindo pelo valor máximo de 255 para facilitar os cálculos.

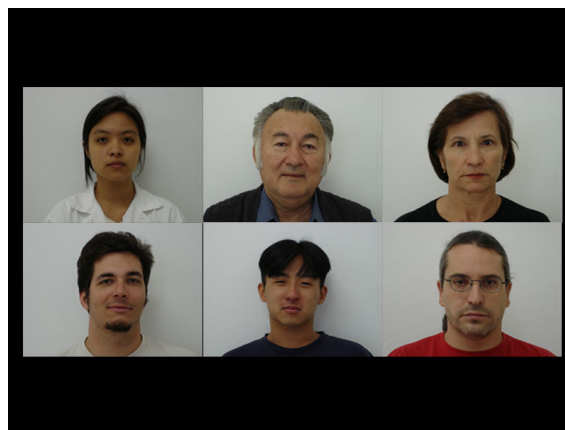


Figura 4.1: Faces sem tratamento pertencentes ao treino.

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 ψ na equação 4.1.

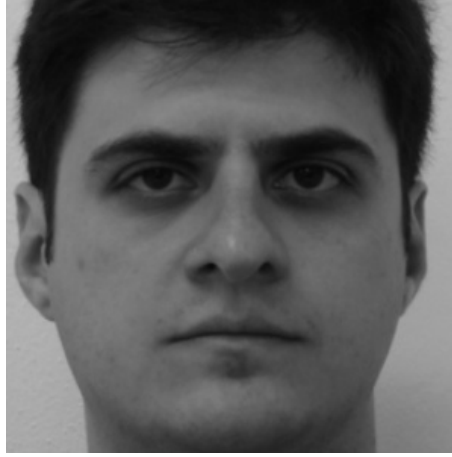


Figura 4.2: Face tratada pertencente ao treino.

$$\Psi = \frac{1}{M} \sum_{i=1}^M \Gamma_i \quad (4.1)$$

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 notada na figura 4.3.



Figura 4.3: Face média Ψ .

Cada face Γ pode ser subtraída da face média Ψ para assim formar um vetor de diferenças Φ presente na equação 4.2, sendo estas as diferenças mais distintas de cada face, essa diferença pode ser notada nas figura 4.4.

$$\Phi_i = \Gamma_i - \Psi \quad (4.2)$$

Os autovetores e autovalores podem ser obtidas a partir da matriz de covariância do vetor AA^T como pode ser visto na equação 4.3, sendo este um vetor de Φ , 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 ($M \times M$).

Figura 4.4: Faces Φ .

$$C = \frac{1}{M} \sum_{i=1}^M \Phi_i \Phi_i^T = AA^T \quad (N^2 \times N^2) \quad (4.3)$$

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 4.4.

$$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow C A v_i = \mu_i A v_i \Rightarrow \text{onde} \rightarrow u_i = A v_i \quad (4.4)$$

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 4.5.



Figura 4.5: Eigenfaces.

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 Φ como na equação 4.5.

$$w_j = u_j^T \Phi_i \quad (4.5)$$

Os pesos w_j são distintos para cada face, esses pesos mais a face média Ψ , resulta na face original Φ , 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 Ω como na equação 4.6

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \\ \vdots \\ w_K^i \end{bmatrix}, \quad i = 1, 2, \dots, M \quad (4.6)$$

4.1.1 Reconhecimento utilizando as Eigenfaces

Para a fase de reconhecimento é necessário representar uma face nova, de fora do conjunto de treinamento, no "espaço de faces" normalizando ela com a face média Ψ como na equação 4.7 (a nova face já deve estar no formato de Γ para ser representada).

$$\hat{\Phi} = \Gamma - \Psi \quad (4.7)$$

Para a projeção das eigenfaces é necessário calcular o vetor de pesos Ω desta face Γ , utilizando as eigenfaces obtidas anteriormente, essa representação pode ser vista na equação 4.8

$$w_i = u_i^T \hat{\Phi} \quad \Omega = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_K \end{bmatrix} \quad (4.8)$$

Calculando o vetor Ω 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 4.9 representa essa distância da face nova de cada face do conjunto de treino.

$$e_r = \min_l \|\Omega - \Omega^l\| \quad (4.9)$$

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 4.9, e pode-se calcular utilizando a equação 4.10. Contudo este valor máximo acaba sendo um valor distante dos demais valores de erros da fórmula 4.9, não podendo ser utilizado para a comparação.

$$\theta = 0,8 * \max(rast) \quad (4.10)$$

Nesta etapa do reconhecimento 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 4.9 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 4.6



Figura 4.6: 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 4.6(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 4.6(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 está 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 4.7 exemplifica a relação da utilização de K para a quantidade de acertos e a Figura 4.8 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 está na Figura 4.6, onde a imagem era comparada manualmente caso obtivesse um um resultado satisfatório, foi computado como um acerto e caso não como um erro.

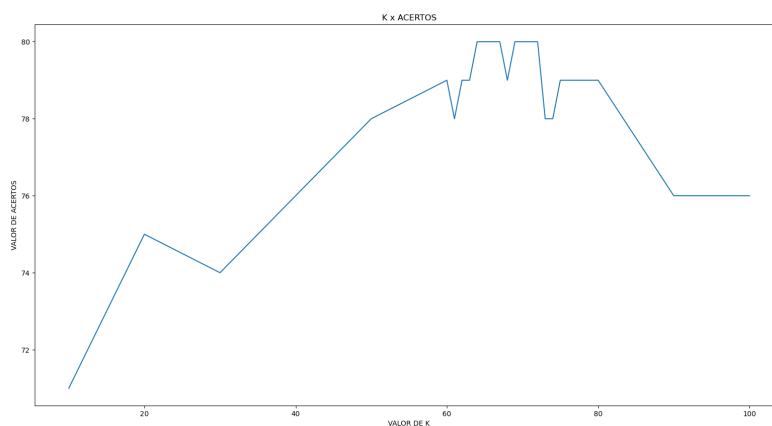


Figura 4.7: Gráfico K x Acertos.

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

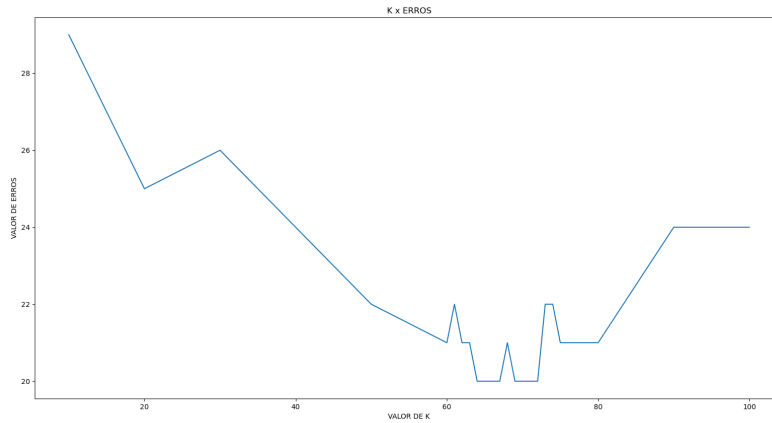


Figura 4.8: Gráfico K x Erros.

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 4.9.

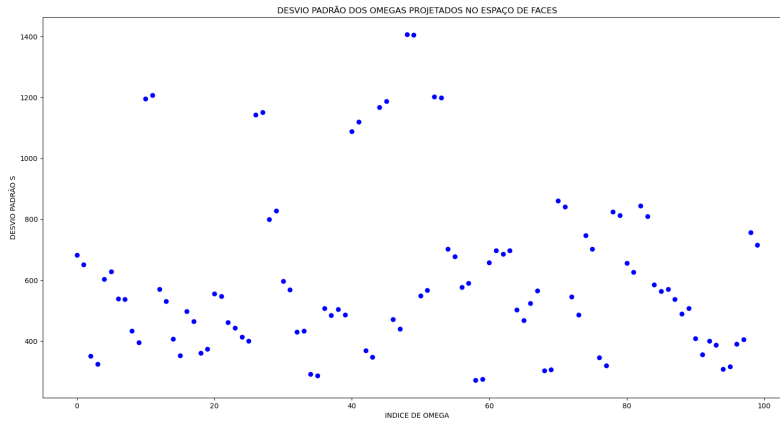


Figura 4.9: Desvio Padrão dos Omegas projetados no espaço de faces.

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 4.10.

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 do menor valor da fórmula 4.9, sendo assim foi necessário utilizar outro valor para ser uma segunda forma de comparação e validação. Com isso a fórmula 4.11 foi utilizada como forma de validação, onde o menor valor foi utilizado.

$$e_{\Phi} = \|\hat{\Phi} - \Phi_i\| \quad (4.11)$$

Com a utilização da fórmula 4.11, notou-se que houve um aumento de 10% de acertos

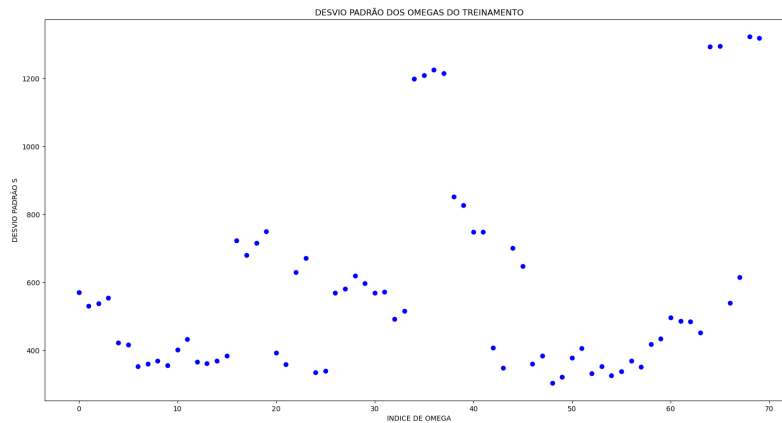


Figura 4.10: Desvio Padrão dos Omegas no espaço de faces.

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 Ω* e o *Erro mínimo de Φ* , sendo assim a Figura 4.11 possui uma comparação das faces reconhecidas manualmente com as faces reconhecidas sem nenhum tipo de verificação.

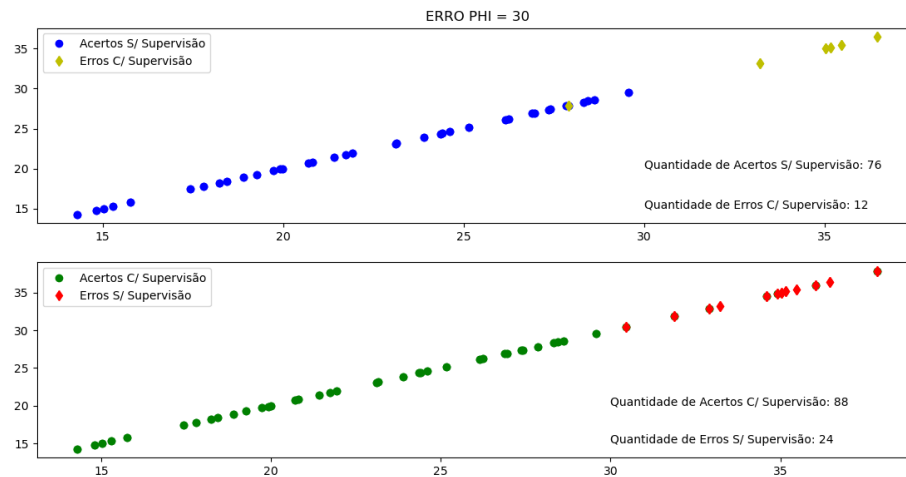


Figura 4.11: Faces reconhecidas do treinamento utilizando Erro $\Phi = 30$.

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 entre o reconhecimento manual e o sem supervisão, pode-se dizer que este efeito de falso e verdadeiro negativo.

Com os gráficos das Figuras 4.11 a 4.16 pode-se notar que as imagens possuem um 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 4.17 a 4.22. Analisando cada gráfico pode-se notar que faces que são desconhecidas do sistema possuem um valor

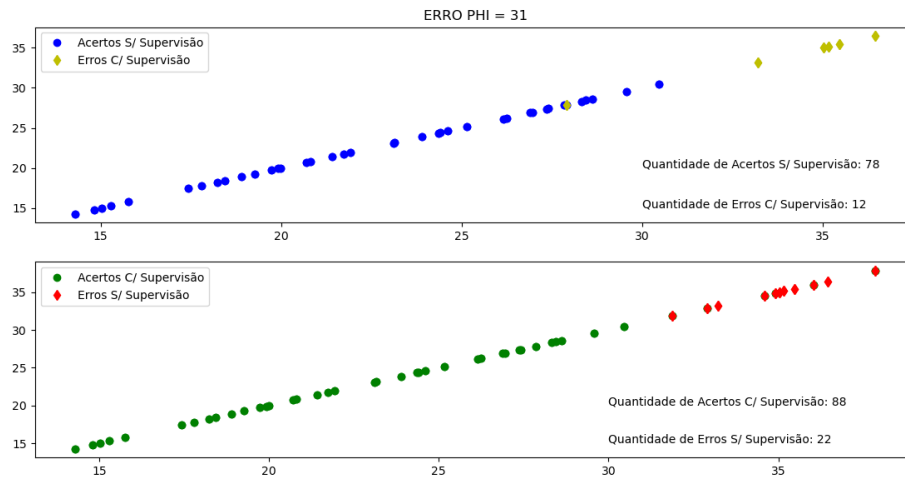


Figura 4.12: Faces reconhecidas do treinamento utilizando Erro $\Phi = 31$.

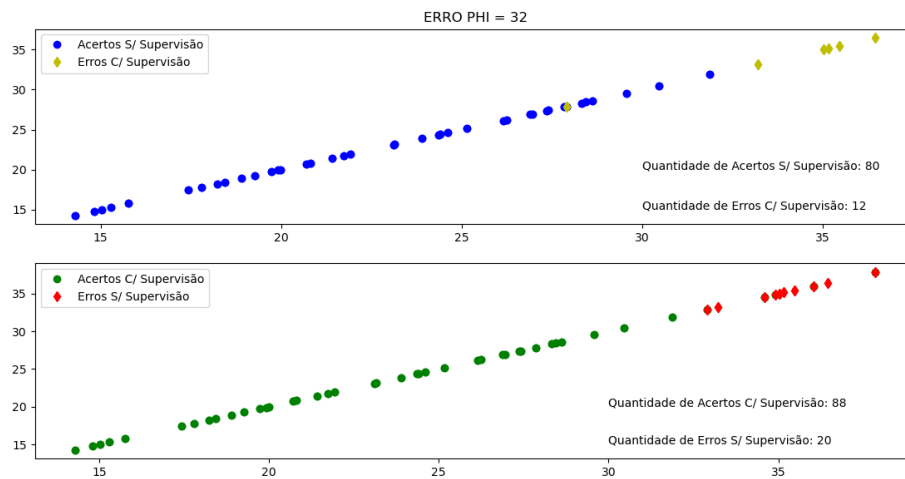


Figura 4.13: Faces reconhecidas do treinamento utilizando Erro $\Phi = 32$.

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 Φ , assim isto mostra que a fórmula 4.11 reproduz a diferença de cada face uma das outras, tendo a diferença de PCA de cada face.

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

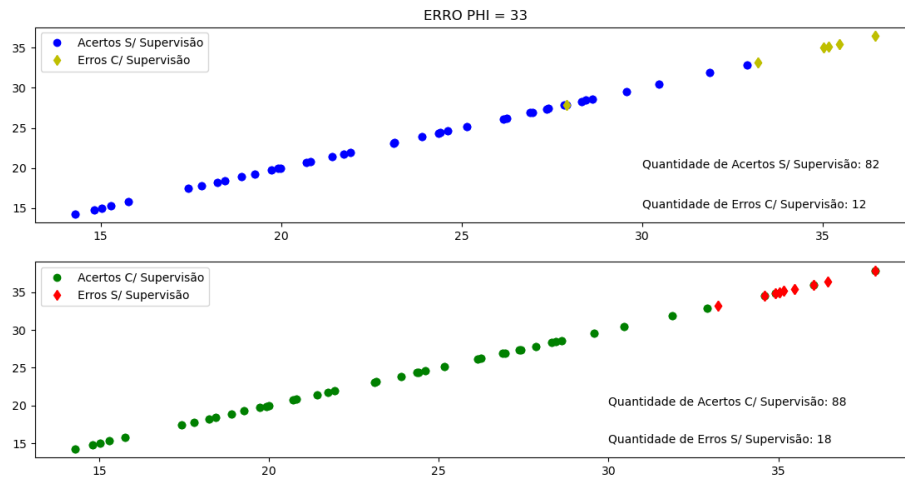


Figura 4.14: Faces reconhecidas do treinamento utilizando Erro $\Phi = 33$.

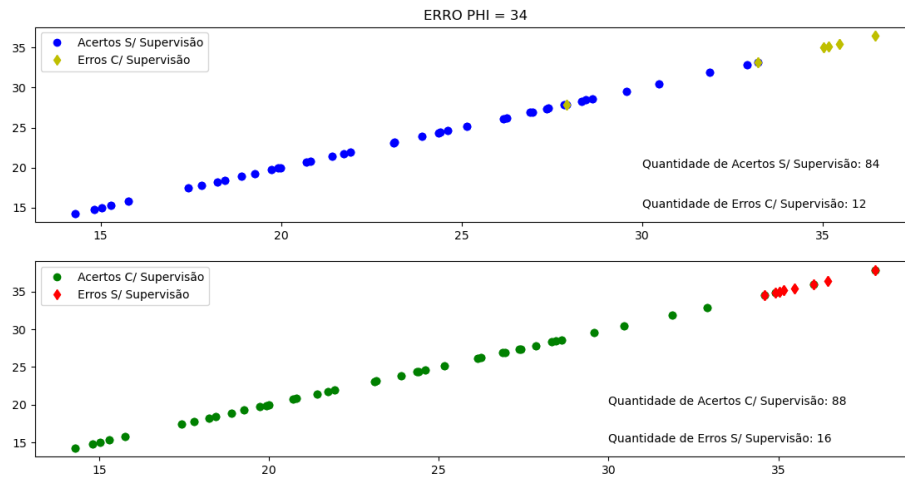


Figura 4.15: Faces reconhecidas do treinamento utilizando Erro $\Phi = 34$.

com o aumento do valor de *Erro Phi*, esses valor 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 ao 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_Φ 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 Ψ e eigenfaces, pode-se utilizar isto para a verificação no sistema.

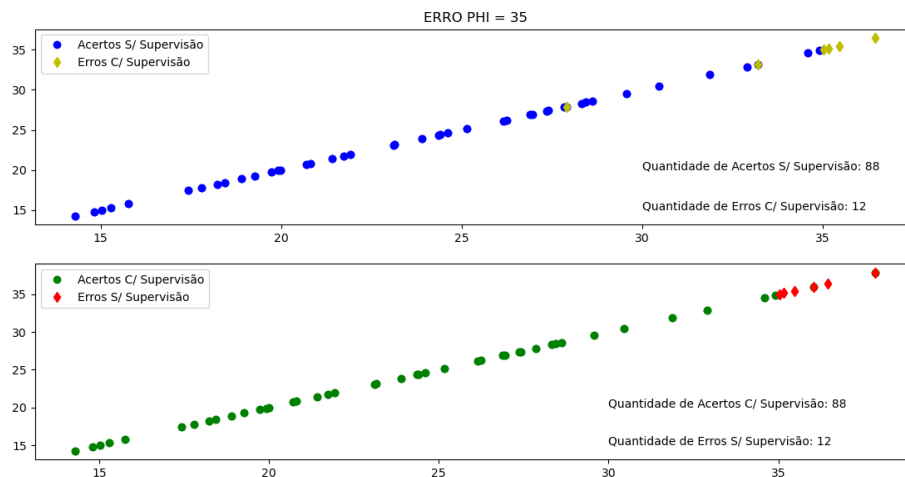


Figura 4.16: Faces reconhecidas do treinamento utilizando Erro $\Phi = 35$.

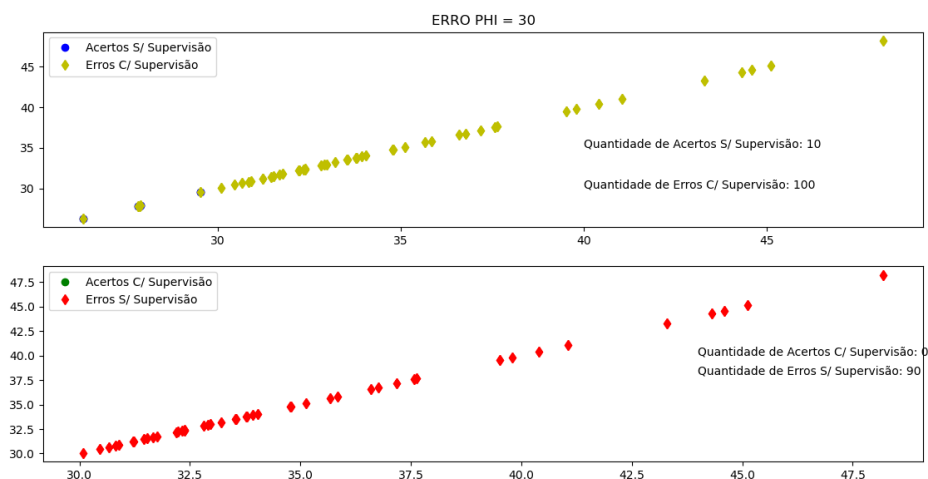


Figura 4.17: Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 30$.

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. 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 dividida em duas formas, sendo uma delas para o armazenamento das informações e outra para o armazenamento dos arquivos. Para o armazenamento das informações foi utilizado o MongoDB com um repositório em nuvem, o seu acesso é feito a partir da biblioteca *Pymongo*, onde cada face tem as seguintes informações armazenadas:

- ID do Mongo

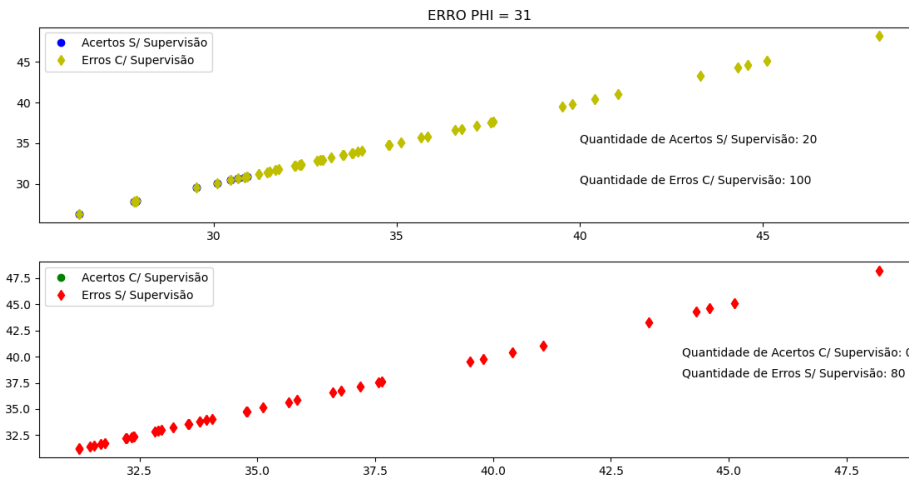


Figura 4.18: Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 31$.

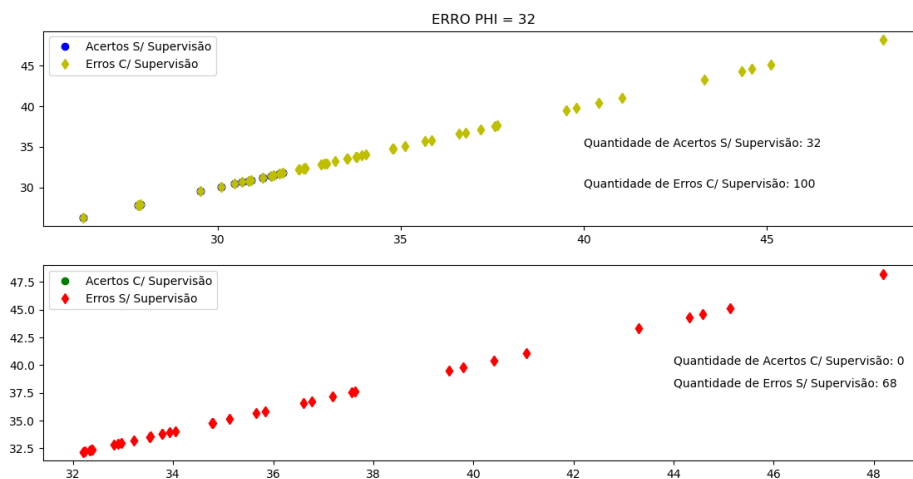


Figura 4.19: Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 32$.

- Nome da Pessoa
- 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 que é gerado automaticamente quando o arquivo com as informações da imagem da face são salvos.

Para o armazenamento de cada imagem foi utilizado o Google Drive, o seu acesso é feito a partir da biblioteca *Pydrive*, onde cada face tem as seguintes informações armazenadas:

- Valor de Φ da face;
- Valor de Ω da face.

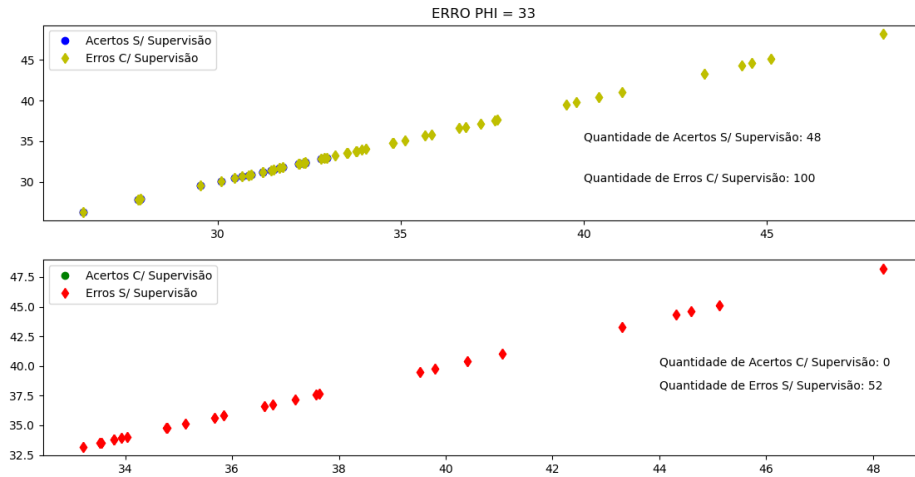


Figura 4.20: Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 33$.

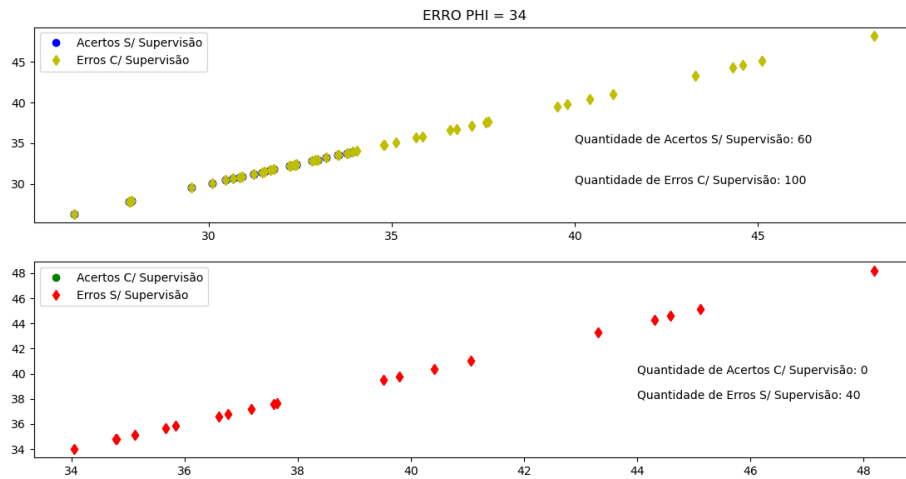


Figura 4.21: Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 34$.

Os valores são armazenados em um arquivo *JSON*, podendo ser utilizadas outras formas de salvar o arquivo, porém 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*.

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.

O processo de armazenamento em ambos os bancos ocorre em três etapas, sendo elas: Gerar as informações da face (valores de Φ e Ω de cada face individualmente), salvar essas informações no Google Drive e devolver o ID que foi salvo e salvar no Mongo o ID do Gogle Drive juntamente com o nome de cada pessoa. Um exemplo deste processo

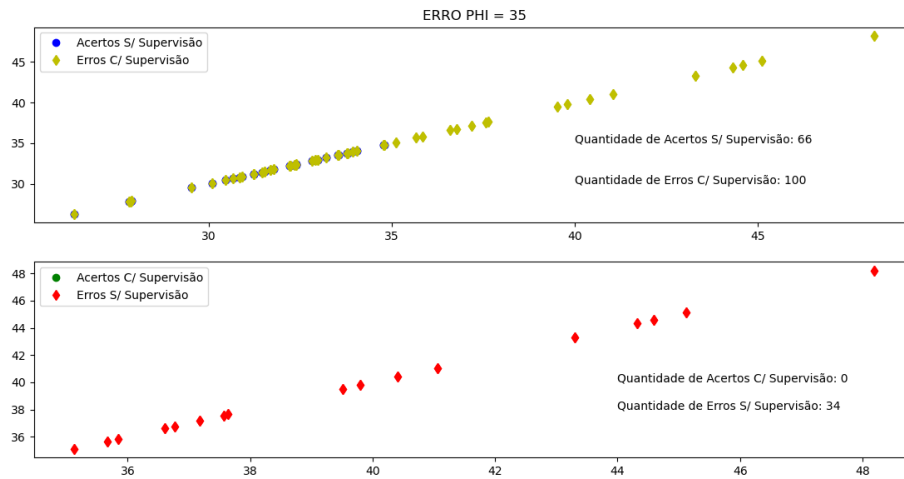


Figura 4.22: Faces reconhecidas de fora do treinamento utilizando Erro $\Phi = 35$.

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

pode ser notado na Figura 4.23.

Para a análise de um 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 Ω , que foram armazenados previamente, e devolvido o valor de Φ da face reconhecida (deve-se após receber o valor de Φ somar o valor de Ψ que também está armazenado no banco de imagens). O método de utilização de ambos os bancos pode ser notado na Figura 4.24

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 capturada 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.



Figura 4.23: Recuperação das imagens a partir dos Bancos de Dados.

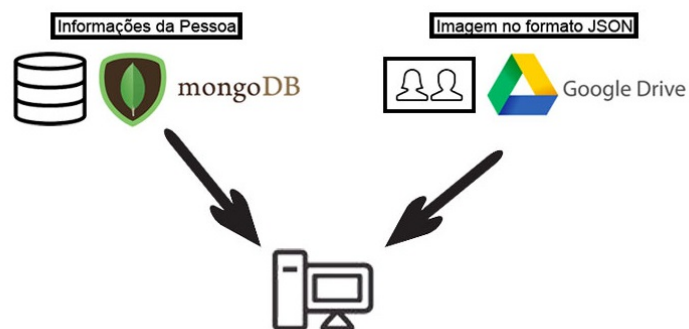


Figura 4.24: Recuperação das imagens a partir dos Bancos de Dados.

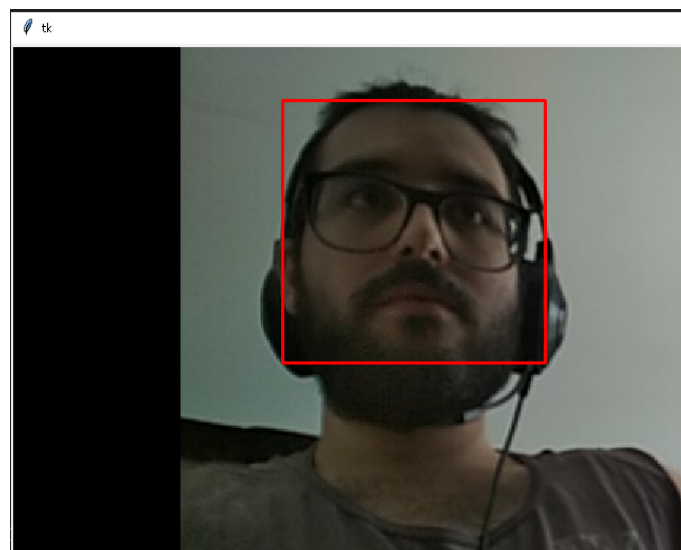


Figura 4.25: Interface do sistema.

5 CONCLUSÃO

6 TRABALHOS FUTUROS

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

REFERÊNCIAS

FOUNDATION, P. S. **Python**. (Acesso em 11 de Dezembro, 2020), <https://www.python.org>.

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.