UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE LORENA

VINICIUS RODRIGUES COSTA

Implementação Numérica da Técnica de Reconhecimento Facial Com Base no Método de Análise de Componentes Principais Utilizando as Plataformas MATLAB™ e EXCEL®

Lorena
2020

VINICIUS RODRIGUES COSTA

Implementação Numérica da Técnica de Reconhecimento Facial Com Base no Método de Análise de Componentes Principais Utilizando as Plataformas MATLAB™ e EXCEL®

Trabalho de conclusão de curso apresentado à Escola de Engenharia de Lorena da Universidade de São Paulo como requisito parcial para a conclusão do curso de Engenharia Física.

Orientador: Prof. Dr. Maurício Guimarães da Silva

Versão Final

Lorena

Julho / 2020

Ficha catalográfica elaborada pelo Sistema Automatizado
da Escola de Engenharia de Lorena,
com os dados fornecidos pelo(a) autor(a)

Costa, Vinicius Rodrigues
Implementação numérica da técnica de reconhecimento facial com base no método de análise de componentes principais utilizando as plataformas MATLAB™ e EXCEL®. / Vinicius Rodrigues Costa; orientador Maurício Guimarães da Silva. - Lorena, 2020.
73 p.

Monografia apresentada como requisito parcial para a conclusão de Graduação do Curso de Engenharia Física - Escola de Engenharia de Lorena da Universidade de São Paulo. 2020

1. Análise de componentes principais. 2. Matlab. 3. Excel. 4. Reconhecimento facial. I. Título. II. da Silva, Maurício Guimarães, orient.

Aos meus pais

AGRADECIMENTOS

Agradeço em primeiro lugar aos meus pais, Rogério Francisco Costa e Luciana Rodrigues Costa, que sempre colocaram a minha educação assim e a de meu irmão em primeiro lugar. Agradeço a eles por todo o amor, carinho e suporte recebidos durante minha jornada de graduação. A Maria Célia Galvão, que conheci durante a minha graduação, e desde então me apoia e ajuda a conquistar meus objetivos. Agradeço também a todos os meus outros familiares, em especial ao meu irmão Victor Rodrigues Costa, que continuava com a mesma idade durante toda minha graduação.

Ao meu orientador Prof. Dr. Maurício Guimarães da Silva que me orientou ao longo deste ano e guiou em meio a uma pandemia para a conclusão deste projeto. Muito obrigado pela oportunidade, paciência e confiança depositada. Agradeço por compreender minhas dificuldades e me ajudar nessa jornada.

Ao Prof. Dr. Fabiano Fernandes Bargos que me orientou como aluno de iniciação científica e me ajudou a trilhar pelos caminhos de hoje. Obrigado pelo apoio durante a graduação, sempre motivando a seguir em frente e por me apresentar o mundo da programação.

Agradeço também a todos os professores e técnicos da Escola de Engenharia de Lorena que ajudaram em todas as etapas da minha formação, não só profissional, mas também pessoal.

Aos meus amigos Amanda Akemy, Humberto Rigamonti, João Francisco, Júlio Cesar, Paulo Rodolpho, Leandro Kellermann, William Jugue pelas boas risadas, situações absurdas e comidas desastrosas que fizeram parte dos momentos mais divertidos da minha graduação.

RESUMO

RODRIGUES, V.C. **Implementação Numérica da Técnica de Reconhecimento Facial Com Base no Método de Análise de Componentes Principais Utilizando as Plataformas MATLAB™ e EXCEL®.** 2020. 73 p. Trabalho de Graduação – Escola de Engenharia de Lorena, Universidade de São Paulo, Lorena, 2020.

A análise das componentes principais é uma técnica antiga, descrita pela primeira vez como método estatístico em 1901. A técnica ressurgiu nos tempos modernos com o avanço da computação e com possibilidade de aplicações mais difundidas. Como uma técnica de modelagem multivariada da estrutura de covariância, ela pode ser amplamente utilizada para análises de sistemas de dados com múltiplas variáveis. Uma das análises possibilitadas pelo método das componentes principais é a classificação de padrões de faces (rostos), um tópico muito abordado e valorizado pelo mercado nos dias atuais. Neste trabalho é desenvolvido um código computacional que pode ser utilizado para reconhecimento facial. Embora se trate de uma análise preliminar, todos os conceitos utilizados na área de Reconhecimento Facial são abordados com rigor e didática. A metodologia de classificação é baseada no Método de Análise de Componentes Principais (PCA). A implementação numérica considera o uso das plataformas EXCEL® e MATLAB™. São desenvolvidos e disponibilizados três códigos computacionais, quais sejam: (i) Implementação preliminar da Metodologia PCA; (ii) Implementação inicial da técnica com imagens; (iii) Implementação preliminar do Método de Classificação para Reconhecimento Facial. Os códigos (i) e (iii) computacionais foram validados com dados *benchmark* disponibilizados na literatura técnica da área e foram considerados satisfatórios.

Palavras-chave: Análise de componentes principais. MATLAB. EXCEL. Reconhecimento Facial.

ABSTRACT

RODRIGUES, V.C. **Numerical implementation of the Facial Recognition Technique Based on the Principal Component Analysis Method using the MATLAB ™ and EXCEL® Platforms.** 2020. 73 p. Graduation Monograph – Escola de Engenharia de Lorena, Universidade de São Paulo, Lorena, 2020.

Principal component analysis is an old technique, first described as a statistical method in 1901. The technique has reappeared in modern times with the advancement of computing and with the possibility of more widespread applications. As a multivariate modeling technique of the covariance structure, it can be widely used for analysis of data systems with multiple variables. One of the analyzes made possible by the principal component method is the classification of face patterns (face recognition), a topic that has been widely addressed and valued by the market today. In this work, a computational code is developed that can be used for facial recognition. Although it is a preliminary analysis, all concepts used in the area of Facial Recognition are approached with rigor and didactics. The classification methodology is based on the Principal Component Analysis Method (PCA). The numerical implementation considers the use of EXCEL® and MATLAB ™ platforms. Three computational codes are developed and made available: (i) Preliminary implementation of the PCA Methodology; (ii) Initial implementation of the technique with images; (iii) Preliminary implementation of the Classification Method for Facial Recognition. The (i) and (iii) computational codes were validated with benchmark data available in the technical literature of the area and were considered satisfactory.

Keywords: Principal componente analysis. MATLAB. EXCEL. Facial recognition.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Durante a história da ciência, diversas técnicas e conceitos foram desenvolvidos para uma dada aplicação específica, ou mesmo sem qualquer aplicação pretendida para aquele momento e, com o passar do tempo, foram redescobertas e utilizadas em aplicações totalmente inesperadas. Atualmente, com os avanços da tecnologia, foi possível resgatar conceitos e técnicas antigas para aplicações mais modernas. Uma das técnicas redescoberta para aplicações modernas é a Análise dos Componentes Principais (do inglês, *Principal Components of Analysis* – PCA). Trata-se de um método de Estatística Multivariada, com histórico de mais de cem anos. Apesar de ser uma técnica antiga, o método apresenta diversas aplicações nos dias atuais, desde análises estatísticas na neurociência até a classificação de dados em análises matemáticas e gráficas, tendo em vista que o método viabiliza o desenvolvimento de técnicas de reconhecimento de padrões.

O método tem como finalidade a análise de uma grande quantidade de dados de forma a reduzir sua dimensão e revelar informações de sua estrutura. A análise explica a estrutura de variância e covariância do conjunto de dados, utilizando combinações lineares. As combinações facilitam a análise de dados, uma vez que removem as redundâncias dos dados originais para criar um novo conjunto de dados, mantendo as principais informações no novo conjunto criado. O novo conjunto é criado de forma que a variância das amostras seja maximizada e as covariâncias entre as dimensões, minimizadas. Assim, o método das componentes principais é capaz de fazer o reconhecimento de padrões em estruturas de dados complexas.

Um dos destaques da capacidade de classificação do método é a possibilidade de realizar reconhecimento facial. Programas de reconhecimento de faces e imagens podem ser feitos de diversas formas, e dentre estas, a técnica que utiliza componentes principais é uma das principais opções. O reconhecimento facial atualmente é alvo de inúmeras pesquisas e investimentos, sendo um tópico importante a ser estudado.

Um aspecto que deve ser salientado é que, embora existam diversas referências na literatura que tratam deste assunto, é difícil encontrar uma referência (publicação ou aula) que tenha abordado o problema em um contexto de Álgebra

Linear. E é mais difícil ainda, encontrar um código computacional que esteja fundamentado em uma teoria PCA e que esteja disponibilizado para o usuário em um mesmo trabalho. Neste trabalho, todos os conceitos matemáticos utilizados no método PCA são apresentados, o algoritmo de implementação numérica está devidamente explicado e, finalmente, o código computacional desenvolvido está disponibilizado.

1.1. Objetivos

- Estudo e Implementação do Método de Análise de Componentes Principais (PCA);
- Implementação do Método de Análise de Componentes Principais (PCA) no estudo de Reconhecimento de Imagens;

1.2. Contribuições

As principais contribuições deste trabalho estão na descrição, documentação e implementação do método PCA utilizando a plataforma MATLAB™ e no estabelecimento do *link* entre o MATLAB™ e EXCEL® no processo de classificação de imagens.

1.3. Divisão do Trabalho

Os estudos apresentados neste trabalho se iniciam com o capítulo 2 dedicado à revisão bibliográfica dos conceitos utilizados em Reconhecimento de Imagens; No Capítulo 3 é fornecida uma revisão de conceitos matemáticos utilizados no algoritmo de implementação do Método PCA, priorizando os conceitos estabelecidos em Álgebra Linear; No Capítulo 4 é descrito em detalhe a implementação numérica do Método PCA; No Capítulo 5 é disponibilizada uma aplicação do Método PCA em Reconhecimento de Imagem e, finalmente, no Capítulo 6 são estabelecidas as conclusões e propostas para trabalhos futuros.

2. REVISÂO BIBLIOGRÁFICA

2.1. Processamento de imagens

2.1.1. Imagens digitais

Uma imagem digital pode ser descrita como uma função discreta bidimensional, na qual os dois eixos principais são representados por duas coordenadas espaciais de números inteiros. Através dessa coordenada, a função vai representar a intensidade da imagem e também o número de linhas e colunas que vão corresponder as suas dimensões (GONZALEZ; WOODS, 2002). Os *pixels* são os componentes básicos de uma imagem digital. A forma como estes elementos são definidos pode variar dependendo do tipo da imagem analisada. Estes elementos podem representar imagens em escala de cinza ou mesmo como uma imagem colorida, mudando apenas a quantidade de informação armazenada por *pixel* (GONZALEZ; WOODS, 2002). Como um *pixel* representa uma coordenada nos eixos da imagem (x e y), podemos representar a função correspondente à intensidade da imagem da forma:

$$F(x,y) = [C_1 \quad C_2 \quad \cdots \quad C_n] \qquad (1)$$

sendo que os elementos C_n marcam os elementos do espaço de cores.

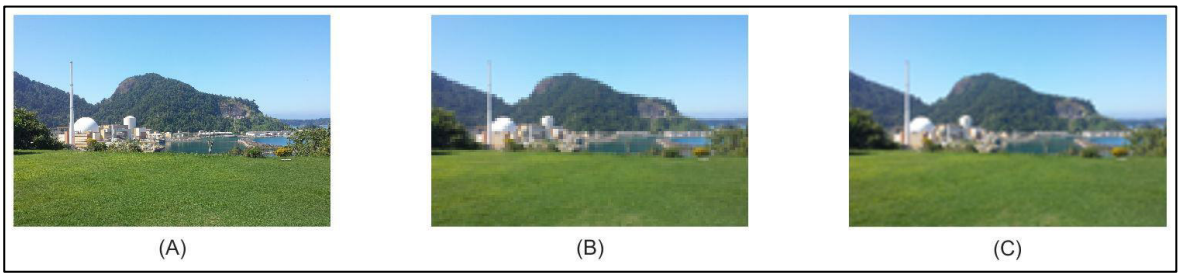
Quando utilizadas escalas de cinza, os valores para a função de representação normalmente estão somente dentro de um único elemento de C_n, e os valores para os níveis de cinza dentro do intervalo $[0, 2^k - 1]$, com um valor inteiro para k (GONZALEZ; WOODS, 2002). Já em uma imagem colorida, são utilizados múltiplos elementos C_n, onde os valores para cada componente (*pixel*) definido como uma tupla ternária com valores que variam de acordo com o espaço de cores utilizadas (GONZALEZ; WOODS, 2002).

No corpo humano, duas estruturas do aparelho visual são utilizadas para fazer perceber as cores: os cones e bastonetes na retina. Os bastonetes são utilizados para obter informações em ambientes escuros e os cones fornecem informação para ambientes mais iluminados (JUNQUEIRA; CARNEIRO, 2008). Os cones no olho humano podem ser divididos em três grupos de acordo com sua sensibilidade a um comprimento de onda azul, vermelho e verde (JUNQUEIRA;

CARNEIRO, 2008). Assim, as combinações dos comprimentos de onda são interpretadas pelo cérebro para formar as cores que vemos. Com intuito de representar as cores que vemos em um espaço de uma imagem digital, foram criados diversos espaços de cores. Para representar uma imagem existem diversos espaços compostos por diferentes elementos, com diferentes objetivos. Um dos mais conhecidos é o RGB (do inglês a sigla para vermelho, verde e azul), onde cada componente de imagem (*pixel*) é composto por um elemento de vermelho (R), verde (G) e azul (B). Assim, os valores para uma figura colorida composta por uma estrutura tridimensional possuem como valor máximo a cor branca e mínimo, a cor preta (SANTANA, 2014).

O processamento de imagens pode ser descrito, de forma simplificada em uma escala de três níveis (alto, médio e baixo) (GONZALEZ; WOODS, 2002). Processamentos considerados de baixo nível são marcados como processos que tem como entrada e saída imagens. Alguns exemplos destes processos de baixo nível são o tratamento de ruído, modificações nos níveis de contraste ou mesmo o redimensionamento de imagens digitais (GONZALEZ; WOODS, 2002). O redimensionamento é um processo simples de baixo nível que pode ser utilizado para reduzir o processamento necessário para trabalhar com a imagem em funções mais complexas. O redimensionamento de uma imagem funciona de forma a alterar o número de linhas e colunas que por sua vez afetam as dimensões da figura, mas mantendo o resultado próximo ao original (GONZALEZ; WOODS, 2002). Na Figura 1 são mostrados os efeitos do redimensionamento de uma imagem.

Figura 1: Exemplo de redimensionamento de uma imagem. Em (A) a figura na sua forma original. (B) a figura redimensionada com diminuição de 10x em suas dimensões. (C) a figura (B) redimensionada para retornar ao tamanho original com perda de dados.



Fonte: O Autor

Apesar de facilitar o processamento de tarefas mais complexas ao fazer modificações nas dimensões de uma imagem, linhas e colunas podem ser

apagadas, resultando em perda da informação original. Como visto na Figura 1, essa perda muitas vezes não pode ser revertida o que pode impedir o retorno da imagem para sua condição original.

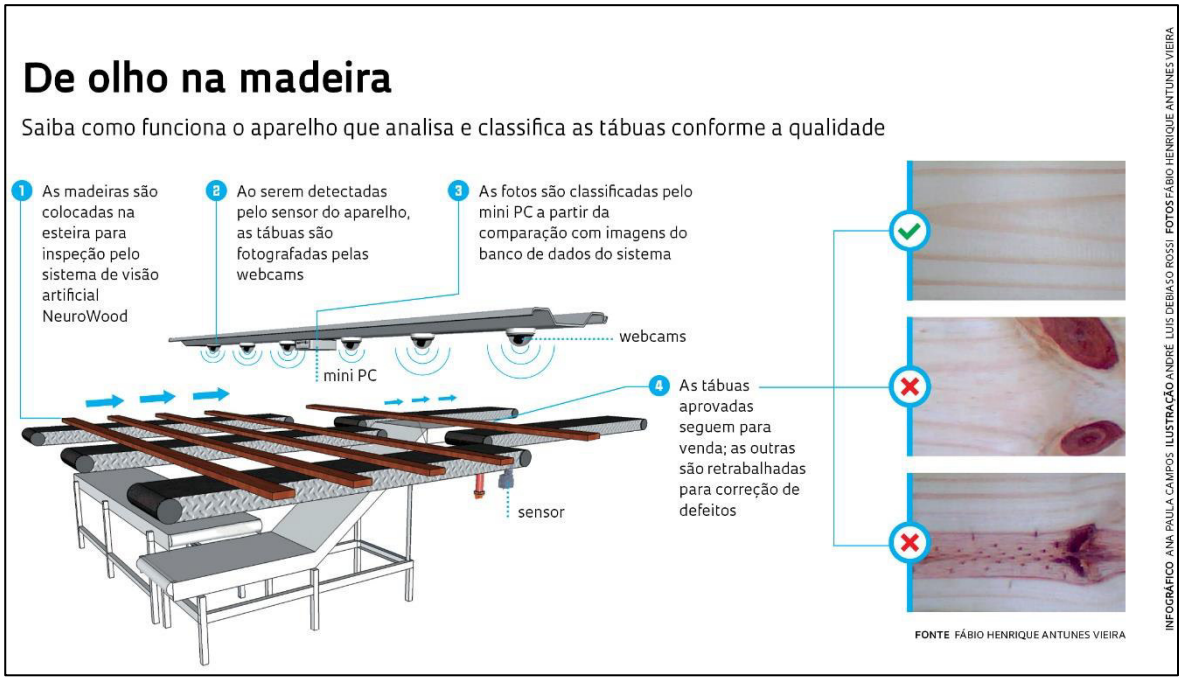
Os processamentos de níveis médios consistem em processos cujos resultados são um conjunto ou regiões extraídas das imagens originais, como um processo de segmentação de imagens (GONZALEZ; WOODS, 2002). Já os processos de alto nível são caracterizados pela obtenção de elementos semânticos da imagem analisada, com objetivo de executar funções de caráter cognitivo, como identificação de padrões, detecção de objetos e rostos, até mesmo a classificação de imagens (GONZALEZ; WOODS, 2002). Processamentos considerados de alto nível são responsáveis por gerar algoritmos de reconhecimento de faces ou objetos que atualmente são utilizados em diversas áreas do conhecimento, do entretenimento e sistemas de segurança. Existem diversos métodos para fazer o reconhecimento de imagens. Dentre eles, alguns métodos surgiram em outras áreas do conhecimento e puderam ser utilizadas para tais aplicações (SANTANA, 2014).

2.1.2. Reconhecimento de imagens

Como visto na seção anterior, o reconhecimento de imagens pode ser considerado um processo de alto nível, o que não impede a utilização de técnicas de níveis mais baixos no conjunto utilizado para fazer a classificação. O reconhecimento de imagens pode ser utilizado para diversas aplicações ao que é conhecido como campo de estudo da visão computacional (ANDRADE, 2019). A área da visão computacional trata de assuntos relacionados a máquinas interpretando imagens, que por sua vez podem ser observados em diversas áreas do conhecimento. Atualmente, existem diversas aplicações de visão computacional, com grandes investimentos dentro do mercado global de processamento de imagens (ANDRADE, 2019). Como dito anteriormente, diversas aplicações utilizam conceitos de reconhecimento de imagens, como utilização dentro da área médica para obter diagnósticos mais precisos, criação de instrumentos de comunicação para pessoas debilitadas, carros autônomos, dentre outros (ANDRADE, 2019). A utilização de tais métodos é feita inclusive no Brasil, onde ganha destaque na indústria madeireira como um método de controle de

qualidade, como observado no infográfico na Figura 2 (ANDRADE, 2019; SILVEIRA, 2017).

Figura 2: Exemplo de aplicação de reconhecimento de imagens no Brasil. Infográfico da utilização de reconhecimento de padrões de imagem para controle de qualidade na indústria madeireira.



Fonte: (SILVEIRA, 2017)

Uma das aplicações que compreende a visão computacional é o reconhecimento facial, que por sua vez é foco de inúmeras pesquisas. A identificação de rostos é alvo de estudos desde os anos 1950 na área de psicologia, e nos meados dos anos 1960 se tornaria também foco de estudo da engenharia (GOTTUMUKKAL; ASARI, 2004). A partir de 1970, equipes multidisciplinares começaram a utilizar o reconhecimento automático com máquinas, tratando o problema como o de reconhecimento de padrões em duas dimensões utilizando técnicas características para resolvê-lo (ZHAO, 2003). Somente nos anos 1990, com as melhorias na capacidade de processamento e outros avanços na computação houve um aumento no interesse pela técnica, uma vez que os avanços poderiam permitir a utilização em sistemas de segurança e vigilância (ZHAO, 2003). Existem muitas aplicações para o reconhecimento facial, como substituta da biometria; interações na robótica; entretenimento; e utilização como base de sistemas de segurança (ZHAO, 2003; OLIVER, 2020). Atualmente, sistemas de segurança têm um destaque para o uso de reconhecimento facial, focados para auxiliar a atuação de forças policiais (OLIVER, 2020). É evidente que tais

desenvolvimentos trazem questões relacionadas à privacidade dentre outros assuntos delicados que fogem ao escopo deste trabalho, mas por sua vez reforçam a importância do entendimento das técnicas de reconhecimento facial.

A identificação de faces é uma tarefa realizada sem muito esforço por parte dos humanos, mas quando comparado ao desenvolvimento em computadores, essa tarefa enfrenta grandes desafios. As variáveis do ambiente, como posição do rosto, foco, iluminação são alguns dos atributos que devem ser levados em consideração juntamente a aspectos individuais como a forma do cabelo, barba e assessórios faciais (GOTTUMUKKAL; ASARI, 2004). Para realizar o reconhecimento facial é necessário encontrar padrões para os indivíduos dentro de inúmeras variáveis, sendo necessário utilizar algum método para encontra-los e fazer a classificação (GOTTUMUKKAL; ASARI, 2004). Dentre os métodos de identificação de padrões, encontra-se a análise das componentes principais, que por si só não faz o reconhecimento de imagens, mas pode ser utilizado como uma das ferramentas principais para o tratamento da imagem de forma a obter os padrões na mesma, e assim fazer sua classificação. Nas próximas seções, são abordados tópicos fundamentais para o entendimento da utilização da técnica de componentes principais e suas aplicações na classificação de imagens.

3. FUNDAMENTAÇÃO TEÓRICA

Para entender a técnica de Análise de Componentes Principais (PCA), é necessário revisar conceitos fundamentais da Álgebra Linear. Dentre estes conceitos ressaltam-se os tópicos: Valores Singulares, Método de Fatoração de Matrizes baseado na Decomposição de Valores Singulares (DVS), e Componentes Principais. Esta seção tem como objetivo revisar estes conceitos desde que serão abordados nos tópicos subsequentes.

3.1. Álgebra Matricial

Dentre as metodologias mais importantes desenvolvidas para a aplicação de álgebra de matrizes na resolução de problemas de Engenharia, destacam-se os Métodos de Fatoração. Estes métodos são vastamente utilizados tendo em vista que vários modelos matemáticos representativos de fenômenos naturais, sejam eles nas áreas de transferência de calor, escoamento de fluidos, análise estrutural, dentre outros, deverão ser discretizados de forma a ser possível a sua resolução.

O objeto matemático obtido com esta discretização pode ser representado por um sistema de equações diferenciais ordinárias, cujas variáveis de estado (***x***) são função das variáveis que excitam o sistema (***u***), também denominadas de variáveis de entrada, no tempo (*t*). As variáveis de saída, ***y***, são calculadas a partir do conhecimento das variáveis de estado e das variáveis de entrada. Este tipo de representação é denominado de Modelagem no Espaço de Estados, qual seja:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{cases} \qquad (2)$$

sendo ***A*** a matriz de estados, ***B*** a matriz de entrada, ***C*** a matriz de saída e ***D*** a matriz de transferência direta. Observe que, para o caso de análise em regime permanente, o modelo matemático representativo do problema pode ser simplificado conforme segue:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{B}\boldsymbol{u} \qquad (3)$$

Ou seja, todos os problemas recaem na resolução de equações matriciais, seja, ou não, dependentes do tempo. Esta resolução é realizada por Métodos de Fatoração.

3.1.1. Métodos de Fatoração

O primeiro Teorema demonstrado em Álgebra Linear relacionado com Métodos de Fatoração pode ser enunciado conforme segue (POOLE , 2004):

Teorema 1 – Se A é ortogonalmente diagonalizável, então A é simétrica. Ou seja:

$$\boldsymbol{A = PDP^T} \qquad (4)$$

sendo ***P*** uma matriz ortogonal e ***D*** uma matriz diagonal formada pelos autovalores de ***A***.

Se ***A*** não é simétrica, não será possível realizar uma fatoração nos moldes apresentados anteriormente. Contudo, demonstra-se que, para estes casos, ***A*** pode ser fatorado. De acordo com POOLE (2004):

$$\boldsymbol{A = QDQ^{-1}} \qquad (5)$$

desde que ***Q*** seja uma matriz invertível. Com esta definição, ter-se-á um aumento significativo do número de problemas que poderão ser abordados. Salienta-se, porém, que esta solução ainda é restrita, tendo em vista as dimensões da matriz ***A*** e da necessidade de ***Q*** ter que ser invertível.

Então surge o Método de Decomposição por Valores Singulares (DVS), a partir do qual, toda matriz (simétrica ou não, quadrada ou não) possui uma fatoração da forma (POOLE, 2004):

$$\boldsymbol{A = PDQ^{\mathrm{T}}} \qquad (6)$$

sendo ***P*** e ***Q*** ortogonais e ***D*** uma matriz diagonal.

3.1.2. Decomposição por Valores Singulares

Quando átomos vibram, eles emitem luz. Demonstra-se por intermédio da Mecânica Quântica que as frequências desta vibração correspondem aos autovalores associados a um Operador matricial. Ou seja, pode-se, literalmente, "ver" os autovalores de um átomo a partir de seu espectro. Utilizando este conceito "ilustrativo", pode-se enunciar outro famoso Teorema da Álgebra Linear, o Teorema Espectral (POOLE, 2004):

Teorema 2 – Seja *B* uma matriz real (*n* x *n*). Então, *B* será simétrica, se e somente se, for ortogonalmente diagonalizável.

Pode-se aplicar este Teorema no processo de fatoração de uma matriz ***A*** qualquer. Considere uma matriz ***A*** (*m* x *n*). Sabe-se que a matriz $\boldsymbol{B} = \boldsymbol{A^T A}$, apresentará dimensão (*n* x *n*) e será simétrica. Portanto, de acordo com o Teorema Espectral, esta matriz, $\boldsymbol{B} = \boldsymbol{A^T A}$**,** poderá ser diagonalizada ortogonalmente. Dentro deste contexto, define-se o conceito de Valores Singulares (POOLE, 2004):

Definição: Se *A* é uma matriz (*m* x *n*), os valores singulares da *A* são as raízes quadradas dos autovalores de $\boldsymbol{A^T A}$.

Com base no acima exposto, pode-se enunciar o Teorema do Método DVS, qual seja (POOLE, 2004):

Teorema 3 – Seja *A* uma matriz (*m* x *n*). Então, existe uma matriz ortogonal *U* (*m* x *m*), uma matriz ortogonal *V* (*n* x *n*) e uma matriz Σ (*m* x *n*), de modo que:

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^{\mathrm{T}} \qquad (7)$$

As colunas de ***U*** são chamadas de vetores singulares à esquerda de ***A*** e as colunas de ***V*** são chamadas de vetores singulares à direita de ***A***. A matriz $\boldsymbol{\Sigma}$ contém os valores singulares de ***A***, ou seja, as raízes quadradas dos autovalores de $A^T A$. Onde a equação (7) é a decomposição de valores singulares na forma reduzida (BRUNTON; KUTZ, 2019).

3.1.3. Forma DVS por Produto Externo

Existe um outro método matemático para a decomposição por valores singulares. Considere a decomposição espectral definida pela Equação (7). Na forma aberta, obtém-se:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = [\boldsymbol{u_1} \quad \cdots \quad \boldsymbol{u_m}] \begin{bmatrix} \sigma_1 & \dots & 0 & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ 0 & \dots & \sigma_r & \\ & \mathbf{0} & & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{v_1^T} \\ \vdots \\ \boldsymbol{v_n^T} \end{Bmatrix} \tag{8}$$

A partir de manipulações algébricas da expressão acima, demonstra-se que (POOLE, 2004):

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = \sigma_1 \boldsymbol{u_1} \boldsymbol{v_1^T} + \cdots + \sigma_r \boldsymbol{u_r} \boldsymbol{v_r^T} \tag{9}$$

A Equação (9) é denominada de forma de DVS por produto externo. É interessante notar que esta equação representa a matriz ***A*** "ponderada" pelos valores singulares (σ_i) de ***A***. Considerando-se que os vetores ***U*** e ***V*** estão normalizados, pode-se concluir que, dependendo da magnitude de σ_i, os valores da matriz ***A*** não serão substancialmente alterados. Neste contexto, adotando-se $k \leq r$, a representação $\mathrm{A_k}$, Equação (10), pode ser considerada uma aproximação de A obtida a partir dos *k* primeiros valores singulares e os respectivos vetores singulares.

$$\boldsymbol{A_k} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = \sigma_1 \boldsymbol{u_1} \boldsymbol{v_1^T} + \cdots + \sigma_k \boldsymbol{u_k} \boldsymbol{v_k^T} \tag{10}$$

3.2. Componentes Principais

Com a finalidade de apresentar este conceito de forma mais didática, sem perda da generalidade e do significado matemático do problema, tratar-se-á a Equação (9) a partir de um formato equivalente, qual seja, a combinação linear exibida na Equação (11), qual seja:

$$y = a_1 x_1 + \cdots + a_r x_r \tag{11}$$

Considere que os valores de $x_i\,(i = 1,\ldots,r)$ são "dados" conhecidos de um determinado problema e que os parâmetros $a_i(i = 1,\ldots,r)$ são incógnitas a serem determinadas com a finalidade de se estimar o valor da variável y a partir da maximização de uma dada função objetivo.

Note que a variável y apresenta média ($\bar{y}$) e variância (S_y^2) dadas pelas expressões (12) e (13), .

$$\bar{y} = a_1\bar{x}_1 + \cdots + a_r\bar{x}_r \qquad (12)$$

$$S_y^2 = \sum_{j=1}^{r} a_{\mathrm{j}}^2\, S_{\mathrm{j}}^2 + 2\sum_{j=1}^{r-1}\sum_{k=1}^{r} a_j\, a_k S_{jk} \qquad (13)$$

sendo $\mathrm{S_{jk}}$ a covariância entre as variáveis $\mathrm{x_j}$ e $\mathrm{x_k}$, dada por:

$$\mathrm{S_{jk}} = \left(\mathrm{x_j} - \bar{\mathrm{x}}_{\mathrm{j}}\right)\left(\mathrm{x_k} - \bar{\mathrm{x}}_{\mathrm{k}}\right) \qquad (14)$$

e S_j^2 é a variância da variável $\mathrm{x_j}$.

A definição de Componente Principal é a combinação linear representada na Equação (11), cuja variância S_y^2 deve ser maximizada e está sujeita a restrição exibida na Equação (15):

$$\sum_{j=1}^{r} a_{\mathrm{j}}^2 = 1 \qquad (15)$$

Observe que a variância está associada ao produto interno entre dois vetores de dados iguais e a covariância é representada pelo produto interno entre dois vetores de dados diferentes. Em aplicações práticas, diferentes amostras podem estar bastante correlacionadas, ou seja, os vetores, mesmo sendo distintos podem estar muito próximos de serem linearmente dependentes (LD). Neste contexto, uma alternativa para encontrar o conjunto de coeficientes $a_i(i = 1,\ldots,r)$ é procurar descorrelacionar os dados originais encontrando direções (dentre as *r* possíveis estipuladas pelo problema) nas quais a variância é maximizada. Para isso, ao invés

de utilizar ***A***, considera-se a matriz de covariância dos dados de ***A*** no processo de maximização.

3.2.1. Exemplo de Obtenção das Componentes Principais

Neste tópico é apresentado um exemplo que aplica os conceitos vistos no item anterior. Considere a combinação linear entre duas variáveis:

$$y = a_1 x_1 + a_2 x_2 \qquad (16)$$

O problema de obtenção das componentes principais pode ser definido matematicamente como sendo um problema de otimização, ou seja:

$$\boldsymbol{Max} \quad S_y^2 = \sum_{j=1}^{r} a_j^2 \, S_j^2 + 2 \sum_{j=1}^{r-1} \sum_{k=1}^{r} a_j \, a_k S_{jk} \qquad (17)$$

$$\boldsymbol{s.a.}: \sum_{j=1}^{r} a_j^2 = 1$$

Em duas dimensões, tem-se:

$$\boldsymbol{Max} \quad S_y^2 = a_1^2 S_1^2 + a_2^2 S_2^2 + 2 a_1 a_2 S_{12} \qquad (18)$$

$$\boldsymbol{s.a.}: a_1^2 + a_2^2 = 1$$

Desde que se trata de um problema de otimização, este pode ser resolvido utilizando técnicas como a de Multiplicadores de Lagrange. Observe que a restrição do problema já está na forma para se aplicar este método. O problema é colocado conforme segue:

$$\nabla f = \lambda \nabla g \qquad (19)$$

Sendo $f = S_y^2(a_1, a_2)$ e $g(a_1, a_2) = a_1^2 + a_2^2$. A função lagrangeana (L) é dada por:

$$\mathrm{L}(a_1, a_2, \lambda) = f(a_1, a_2) - \lambda[g(a_1, a_2) - c] \qquad (20)$$

Sendo $c = 1$, Equação (18). Ou seja:

$$\mathrm{L}(a_1, a_2, \lambda) = a_1^2 S_1^2 + a_2^2 S_2^2 + 2a_1 a_2 S_{12} - \lambda[a_1^2 + a_2^2 - 1] \qquad (21)$$

O problema então consiste em encontrar os pontos críticos da função $\mathrm{L}(\mathrm{a}_1, \mathrm{a}_2, \lambda)$. Os pontos críticos de $\mathrm{L}(\mathrm{a}_1, \mathrm{a}_2, \lambda)$ são os valores de $\mathrm{a}_1, \mathrm{a}_2$ e λ que anulam as três derivadas parciais primeiras de $\mathrm{L}(\mathrm{a}_1, \mathrm{a}_2, \lambda)$, ou seja, o problema consiste em resolver o seguinte sistema:

$$\begin{cases} \mathrm{L}_{a_1} = 0 \\ \mathrm{L}_{a_2} = 0 \\ \mathrm{L}_{\lambda} = 0 \end{cases} \qquad (22)$$

Sendo $\mathrm{L}_{\xi} = \frac{\partial \mathrm{L}}{\partial \xi}$. Aplicando as derivadas, obtém-se:

$$\begin{cases} 2\mathrm{a}_1 \mathrm{S}_1^2 + 2\mathrm{a}_2 \mathrm{S}_{12} - 2\lambda \mathrm{a}_1 = 0 \\ 2\mathrm{a}_2 \mathrm{S}_2^2 + 2\mathrm{a}_1 \mathrm{S}_{12} - 2\lambda \mathrm{a}_2 = 0 \\ \mathrm{a}_1^2 + \mathrm{a}_2^2 - 1 = 0 \end{cases} \qquad (23)$$

Em notação matricial, tem-se:

$$\left[\begin{pmatrix} S_1^2 & S_{12} \\ S_{12} & S_2^2 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right] \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = 0 \qquad (24)$$

O que fornece a equação do tipo:

$$[\boldsymbol{C} - \lambda \boldsymbol{I}]\boldsymbol{a} = 0 \qquad (25)$$

Portanto, o procedimento para encontrar as componentes principais de uma conjunto de dados, definido pelo vetor $x_i\,(i = 1, \ldots, r)$, é resolvido a partir do problema de autovalores & autovetores associados à matriz ***C***, a qual é denominada de matriz de covariância. Essencialmente, as componentes principais do problema são representadas pelos autovetores de ***C***, desde que estes maximizam a variância da matriz de dados.

3.3. Obtenção das Componentes Principais Utilizando DVS

No tópico anterior foi apresentada a metodologia de obtenção das componentes principais, utilizando-se, como referência, o conceito matemático do significado de componentes principais. Pode-se concluir que o problema é, essencialmente, determinar os autovetores da matriz de covariância dos dados. Neste tópico, é apresentada uma metodologia, baseada no Método de Fatoração DVS, através da qual é possível calcular os autovetores & autovalores da matriz de covariância.

Partindo-se da mesma matriz principal $\boldsymbol{A}$ com dimensões $m \times n$, define-se uma nova matriz $\boldsymbol{Z}$, Equação (26).

$$\boldsymbol{Z} \equiv \frac{1}{\sqrt{n}} \boldsymbol{A^T} \tag{26}$$

Sendo *n,* procede-se então ao produto interno:

$$\boldsymbol{Z^T Z} = \left(\frac{1}{\sqrt{n}} \boldsymbol{A^T}\right)^T \left(\frac{1}{\sqrt{n}} \boldsymbol{A^T}\right) \tag{27}$$

A partir do qual, tem-se:

$$\boldsymbol{Z^T Z} = \frac{1}{n} \boldsymbol{A A^T} = C \tag{28}$$

Fazendo-se a decomposição de $\boldsymbol{Z}$ na forma reduzida da decomposição de valores singulares, Equação (7), tem-se:

$$\boldsymbol{Z} = \boldsymbol{U \Sigma V^T} \tag{29}$$

Utilizando-se as Equações (28) e (29), chega-se na seguinte relação:

$$\boldsymbol{Z^T Z\, V} = \boldsymbol{V \Sigma^2} \tag{30}$$

O qual se refere a um problema de autovalores & autovetores. Deste modo, a matriz $\boldsymbol{V}$ possui os autovetores de $\boldsymbol{Z^T Z}$, ou seja, os autovetores equivalentes a matriz de covariância (SHLENS, 2014). As colunas de $\boldsymbol{V}$ são os componentes principais da matriz de dados $\boldsymbol{A}$.

3.4. Síntese das Propriedades Algébricas das Componentes Principais

As propriedades algébricas das CP que garantem a sua grande aplicabilidade podem ser sintetizadas conforme segue (DUARTE, 1998):

- As CP representam direções perpendiculares no espaço das variáveis originais (os autovetores que as determinam são ortogonais); logo, os valores ("*scores*") de duas CP são não correlacionadas entre si. Neste contexto, o estudo da diferenciação ou similaridade entre os objetos torna-se independente das correlações entre as variáveis.
- Uma CP é normalizada (padronizada), ou seja, a soma dos quadrados dos coeficientes da combinação linear é igual à unidade.
- A primeira componente principal tem variância máxima em relação às demais (leva à maior diferenciação entre os objetos, por exemplo); logo, a primeira CP é a que encerra a maior parte da variação ocorrida nos dados. Isso confere a esta CP a propriedade de melhor preditora linear das variáveis originais, entre todas as CP.
- No Método PCA cada nova componente obtida explica cada vez menos a variabilidade total, até chegar ao ponto de tal informação ser desprezível; ou seja, a variabilidade explicada pelas CP decresce da 1ª CP para a 2ª, para a 3ª e assim por diante. Logo, se a maior parte da variação de indivíduo para indivíduo (ex.: acima de 80%) residir nas duas ou três primeiras CP, o estudo pode ser feito nestas dimensões, sem grandes distorções, ao invés de em p dimensões.

4. ANÁLISE DE COMPONENTES PRINCIPAIS

Dentre as técnicas de Estatística Multivariada, existe a Análise de Componentes Principais (do inglês PCA). Esta técnica foi descrita pela primeira vez por Karl Pearson em 1901, mas foi utilizada somente em 1933 com as descobertas de Harold Hotelling para analisar estruturas de correlação descrevendo métodos computacionais (HONGYU, 2016). A técnica transforma linearmente as variáveis originais do problema, de forma que um conjunto de muitas variáveis correlacionadas entre si acaba por se tornar um conjunto consideravelmente menor de variáveis sem correlação, contendo a maioria do conteúdo de informação original (HONGYU, 2016). Neste contexto, o principal motivo de sua aplicação é à redução de dados com uma menor perda de informação. Essencialmente, o procedimento consiste em redistribuir as variações no espaço original para obter um conjunto de eixos que não possuam correlação (HONGYU, 2012).

A utilização da técnica PCA pode gerar índices e agrupamentos. Os indivíduos são agrupados de acordo com suas variâncias de suas características (HONGYU, 2016). Ela utiliza conceitos de álgebra matricial, tais como DVS, assim como conceitos estatísticos.

Este capítulo descreve o algoritmo PCA a partir de sua implementação numérica. Foi escolhido um caso estudo que demonstra algumas das potencialidades do método. O Anexo A contém os programas desenvolvidos. Estes foram implementados de forma bastante didática. O objetivo principal foi disponibilizar uma ferramenta que possa ser adaptada para a resolução de diferentes tipos de problemas.

4.1. Algoritmo PCA

O algoritmo PCA estabelece um conjunto de etapas que seguem desde a obtenção de dados até a classificação dos resultados. O método também pode ser implementado de diferentes formas. As principais diferenças ocorrem na forma com a qual são obtidas as componentes principais.

Neste tópico é apresentada a implementação numérica destas etapas, já que boa parte da teoria necessária para este entendimento já foi apresentada no Capítulo 3. De forma geral, as etapas de execução do método PCA são as que seguem:

i. Dados necessários para Análise
ii. Normalização dos Dados
iii. Determinação da Matriz de Covariância
iv. Determinação dos Autovetores da Matriz de Covariância
v. Classificação de Dados: Projeção destes sobre os Autovetores

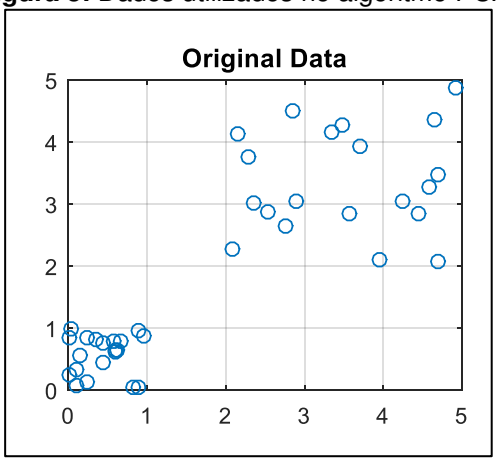
Nos próximos parágrafos são exibidas as etapas supramencionadas a partir de um exemplo de aplicação.

4.2. Dados a Serem Analisados: Matriz A

A metodologia PCA é aplicada a um conjunto de dados, dispostos na forma matricial, com o objetivo de se realizar a "classificação" dos referidos dados segundo um dado critério. Neste contexto, o passo inicial do algoritmo é a geração da matriz de dados, seja na forma experimental ou numérica. Os dados devem ser agrupados em uma matriz $\boldsymbol{A}$ composta de *n* vetores-coluna com *m* atributos.

A Figura 3 exibe uma amostra de dados gerada de forma aleatória que será utilizada para ilustrar a aplicação do método PCA. Neste caso, foram utilizadas m lotes de amostras, cada um dos quais com n medidas experimentais. A Figura 4 exibe o código computacional associado à esta fase. Observe que foi incluído um "distanciamento" entre os dados gerados com o objetivo de testar o código computacional desenvolvido.

Figura 3: Dados utilizados no algoritmo PCA.



Fonte: O Autor

Figura 4: Código Computacional Associado à Etapa 1.

```
% ////////////////////////////// INPUT
% Should be even
numdata= 40;

% Step 1, generating a dataset
x1= rand(numdata/2,1);
y1= rand(numdata/2,1);
x2= 3*rand(numdata/2,1)+2;
y2= 3*rand(numdata/2,1)+2;

% A matrix
x= [x1;x2];
y= [y1;y2];
```

Distanciamento aplicado aos dados analisados

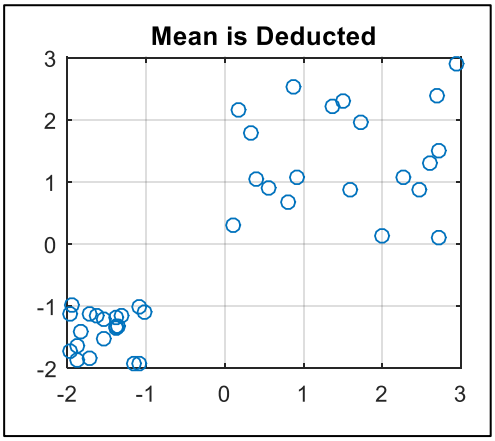
Fonte: O Autor

4.3. Normalização os Dados

Desde que os dados associados a um dado experimento possam apresentar diferentes ordens de grandeza, é prática comum se realizar uma transformação linear de forma a ser possível trabalhar com dados no formato normalizados. Uma das estratégias de normalização é garantir média 0 (zero). Neste contexto, esta etapa consiste em obter a média do conjunto de dados e subtrair o valor da média de cada uma de suas medidas.

No caso de dados com apenas duas dimensões, como visto na Figura 3,o conjunto de dados x deverá ser subtraído de sua respectiva média ($\bar{x}$) e o conjunto y de $\bar{y}$. O resultado final é um conjunto de dados cuja média é zero. O novo conjunto de dados está ilustrado na Figura 5 e o código computacional associado à esta fase está ilustrado na Figura 6.

Figura 5: Amostra aleatória deduzida da média.



Fonte: O Autor

Figura 6: Código Computacional Associado à Etapa 2.

```
% ////////////////////////////////// SOLVER
% Step 2, finding a mean and subtracting
xmean= mean(x);
ymean= mean(y);

xnew= x-xmean*ones(numdata,1);
ynew= y-ymean*ones(numdata,1);
```

Fonte: O Autor

4.4. Matriz de Covariância

Conforme mencionado no Capítulo 3, as componentes principais do problema são representadas pelos autovetores da matriz de covariância, desde que estes maximizam a variância da matriz de dados. A Figura 7 exibe o código computacional elaborado para esta etapa.

Figura 7: Código Computacional Associado à Etapa 3.

```
% Step 3, covariance matrix
covariancematrix= cov(xnew,ynew);
```

Fonte: O Autor

4.5. Autovetores & Autovalores da Matriz de Covariância

A etapa seguinte consiste em se obter os autovalores e autovetores da matriz de covariância. Este procedimento não foi descrito no Capítulo 3 por se tratar de um assunto largamente tratado em diversas referências da área. Na plataforma MATLAB™ são disponibilizadas diferentes metodologias de implementação numérica para a determinação dos autovalores e autovetores de matrizes, dentre estas, o Método de Fatoração de Cholesky (*help* MATLAB™). Nesta fase também são ordenados os autovalores da matriz de covariância com o objetivo de selecionar as componentes principais. A Figura 8 exibe o código computacional gerado para esta fase.

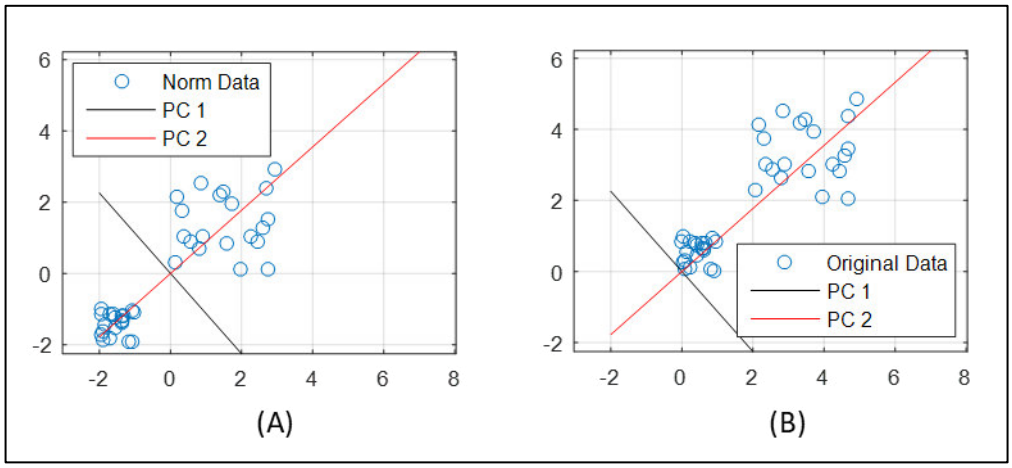
Figura 8: Código Computacional Associado à Etapa 4.

```
% Step 4, Finding Eigenvectors
[V,D] = eig(covariancematrix);
D= diag(D);
P= V(:,find(D==max(D))); % maxeigval
```

Fonte: O Autor

A Figura 9 exibe as componentes principais (CP) associadas aos dados em estudo. Estas componentes foram obtidas a partir dos autovetores gerados pela matriz de covariância dos dados originais e dos dados reduzidos. Salienta-se que não foram observadas diferenças entre os respectivos autovetores.

Figura 9: Componentes Principais da Matriz de Dados Normalizada (A) e Original (B).



Fonte: O Autor

4.6. Classificação dos Dados

A parte final do método PCA trata da classificação dos dados. Essencialmente, a metodologia consiste em se determinar as "distâncias" euclidianas entre os dados analisados e os eixos associados aos autovetores da matriz de covariância. Ou seja, os valores dos dados são expressos em termos de projeções nas direções dos autovetores já ordenados. Com base nestes resultados, procede-se à classificação.

A Figura 10 exibe o código computacional que determina a projeção do vetor dados (já normalizados) sobre os autovetores da matriz de covariância. A Figura 11 exibe o código computacional associado à classificação.

Figura 10: Código Computacional Associado à Etapa 5.

```
% Step 5, Deriving the new data set
% finding the projection onto the eigenvectors

finaldata= maxeigval'*[xnew,ynew]';
```

Fonte: O Autor

Figura 11: Código Computacional Associado à Classificação.

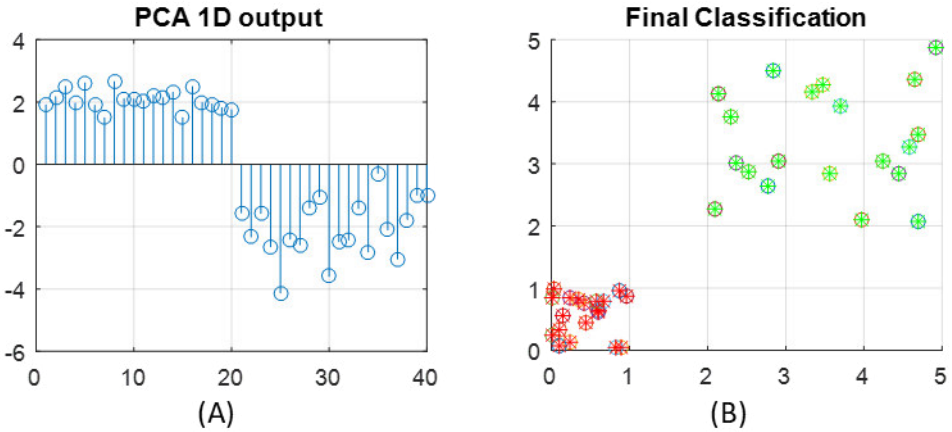
```
% We do a classification now

subplot(2,2,4);
title('Final Classification')
hold on

for i= 1:size(finaldata,2)
    if  finaldata(i)>=0
        plot(x(i),y(i),'o')
        plot(x(i),y(i),'r*')

    else
        plot(x(i),y(i),'o')
        plot(x(i),y(i),'g*')
    end
end
```

Fonte: O Autor

A Figura 12 exibe as projeções dos dados normalizados no sistema de autovetores. É importante salientar que é com estes resultados que é realizada a

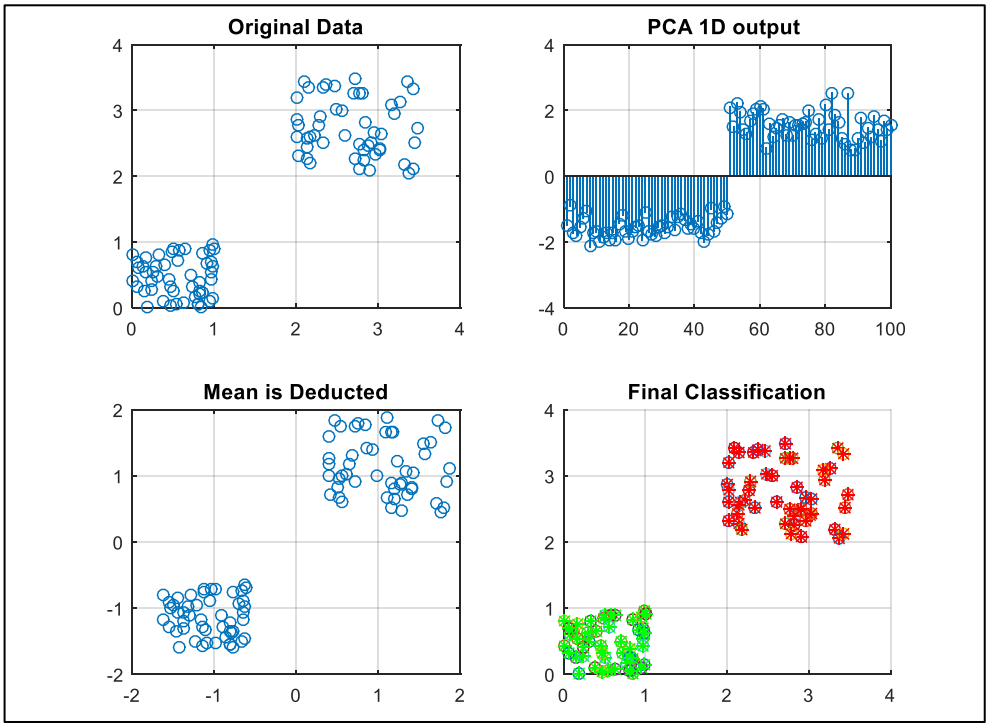
classificação. Finalmente, na Figura 13 são apresentados resultados para uma simulação com uma matriz 100 x 100. O tempo de simulação foi da ordem de 5 segundos com processador Intel core™ i3, e na ordem de 1 segundo com processador Ryzen 5 3600.

Figura 12: Projeções dos dados normalizados (A) e Classificação Final (B).



Fonte: O Autor

Figura 13: Classificação Final com Amostra de 100 elementos.



Fonte: O Autor

5. MÉTODO PCA APLICADO À IMAGENS

A metodologia DVS apresenta inúmeras aplicações nas áreas de Matemática, Engenharia, Estatística, e muitas outras. Dentre estas, destacam-se: (*i*) cálculo de "posto" de matrizes, muito utilizado em Tecnologias de Sistemas de Controle, (*ii*) Cálculo de Normas de Matrizes, utilizado em Mecânica Quântica; (*iii*) Cálculo de Pseudo Inversa e a Aproximação por Mínimos Quadrados, utilizado em Estatística, entre outras. Entre as aplicações mais publicadas na literatura técnica da área destacam-se as técnicas de "Compressão de Imagem" e "Reconhecimento de Imagens".

Um exemplo interessante para entender o problema de "Compressão de Imagem" está disponibilizado no livro devido a POOLE (2004, pg. 555). Considere uma imagem digital, preto e branco, de tamanho 340 x 280 pixels que deverá ser transmitida eletronicamente. Cada pixel é um dos 256 tons de cinza, cada um dos quais associados aos números pertencentes ao intervalo [0,255]. Observe que é possível armazenar esta informação em uma matriz ***A*** de dimensão 340 x 280, o que significaria transmitir (ou manipular) 95.200 números. Este processo é extremamente custoso em termos computacionais, caso esta operação tenha que ser repetida várias vezes. Então, surge a técnica de compressão de imagem, baseada no conceito DVS.

A ideia por trás da técnica de compressão de imagem é explorar o fato de que algumas partes de uma figura são "menos" importantes do que outras. Em termos matemáticos, o menor valor singular na DVS da matriz ***A*** provém de partes "menos interessantes" da imagem e, de fato, poderia ser ignorado no processo de reconstituição da imagem transmitida.

Ou seja, o método DVS é uma ferramenta matemática para representar a matriz ***A*** com um menor número de valores singulares, Equação (10), e ainda assim, manter grande parte do conteúdo informativo da imagem. Para o caso supramencionado, se tivéssemos somente 20 valores singulares de interesse, ao invés de transmitir 95.200 números, precisaríamos enviar somente os 20 valores singulares, mais 20 vetores $\mathrm{u}_1, \cdots, \mathrm{u}_{20}$ de R^{340} e 20 vetores $\mathrm{v}_1, \cdots, \mathrm{v}_{20}$ de R^{280}, ou seja, 20 + 20*340 + 20*280=12.420 números associados às cores da figura.

Contudo, ainda fica a pergunta, como selecionar os 20 valores singulares que deverão ser “transmitidos”? A resposta está no estudo das componentes principais, o que é realizado pelo Método PCA.

Neste trabalho foi dado ênfase à técnica de “Reconhecimento de Imagem”. O problema não será abordado de forma generalizada (métodos tridimensionais, por exemplo), contudo, permitirá que o leitor tenha um primeiro entendimento do significado do problema e da técnica de implementação numérica do método PCA na resolução deste problema. Salienta-se também que a plataforma MATLAB™ comporta várias ferramentas que auxiliam na implementação desta técnica. Neste contexto, muitos aspectos associados à implementação computacional não serão discutidos, tendo em vista que são *toolboxes* da plataforma MATLAB™. A versão utilizada para desenvolver e executar os códigos foi R2018b, versões anteriores da plataforma podem apresentar erros ao executar os programas desenvolvidos neste trabalho.

5.1. Descrição do Problema

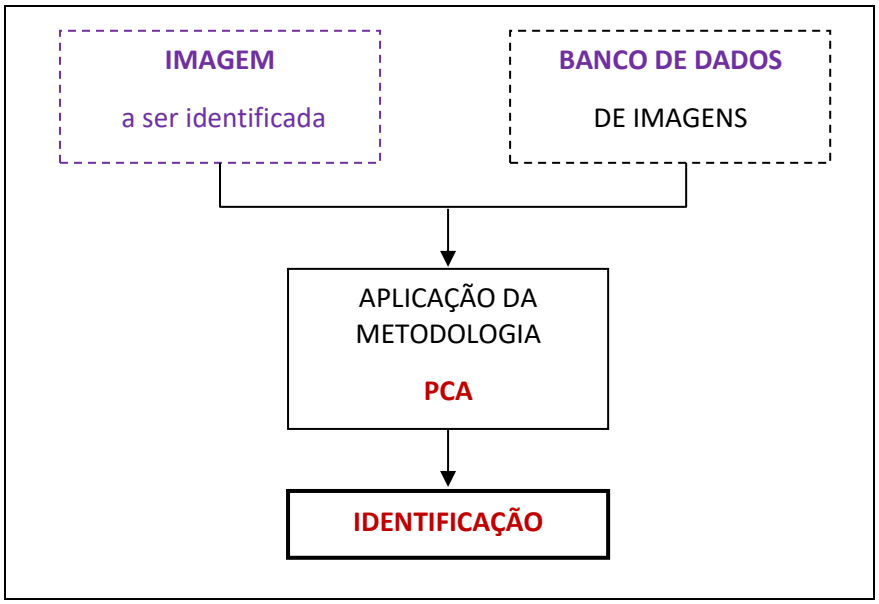
Essencialmente, o objetivo é identificar uma pessoa, com base em uma foto de sua face, utilizando um banco de dados que apresenta imagens de faces de diferentes pessoas. Neste banco de dados, também estão inclusas diferentes fotos da pessoa que se deseja identificar.

A Figura 14 exibe o diagrama de blocos do problema considerado. Os blocos com contorno pontilhado dizem respeito às entradas do problema. Neste caso, é necessário um banco de dados com imagens das pessoas que se deseja identificar para que o método seja mais eficiente. O bloco com contorno mais espesso está associado à saída do algoritmo, ou seja, a identificação da foto que foi trabalhada pelo SOLVER (bloco de linha contínua - ilustrado como “Aplicação da Metodologia PCA”).

5.2. Algoritmo do Programa Desenvolvido

Conforme explicitado anteriormente, na resolução deste problema pode ser aplicado, de forma direta, o método PCA. Neste tópico, é descrito com algum detalhe, o SOLVER do problema.

Figura 14: Descrição do Problema de Reconhecimento de Imagem.



Fonte: O Autor

A Figura 15 exibe o diagrama de blocos associado à resolução do problema de reconhecimento de imagem. A sequência de cálculo é definida conforme segue:

(i) Os dados de entrada no algoritmo são as imagens pertencentes ao banco de dados, denominadas “Imagens de Treino”, e a imagem a ser identificada, denominada de “Imagem Teste”;

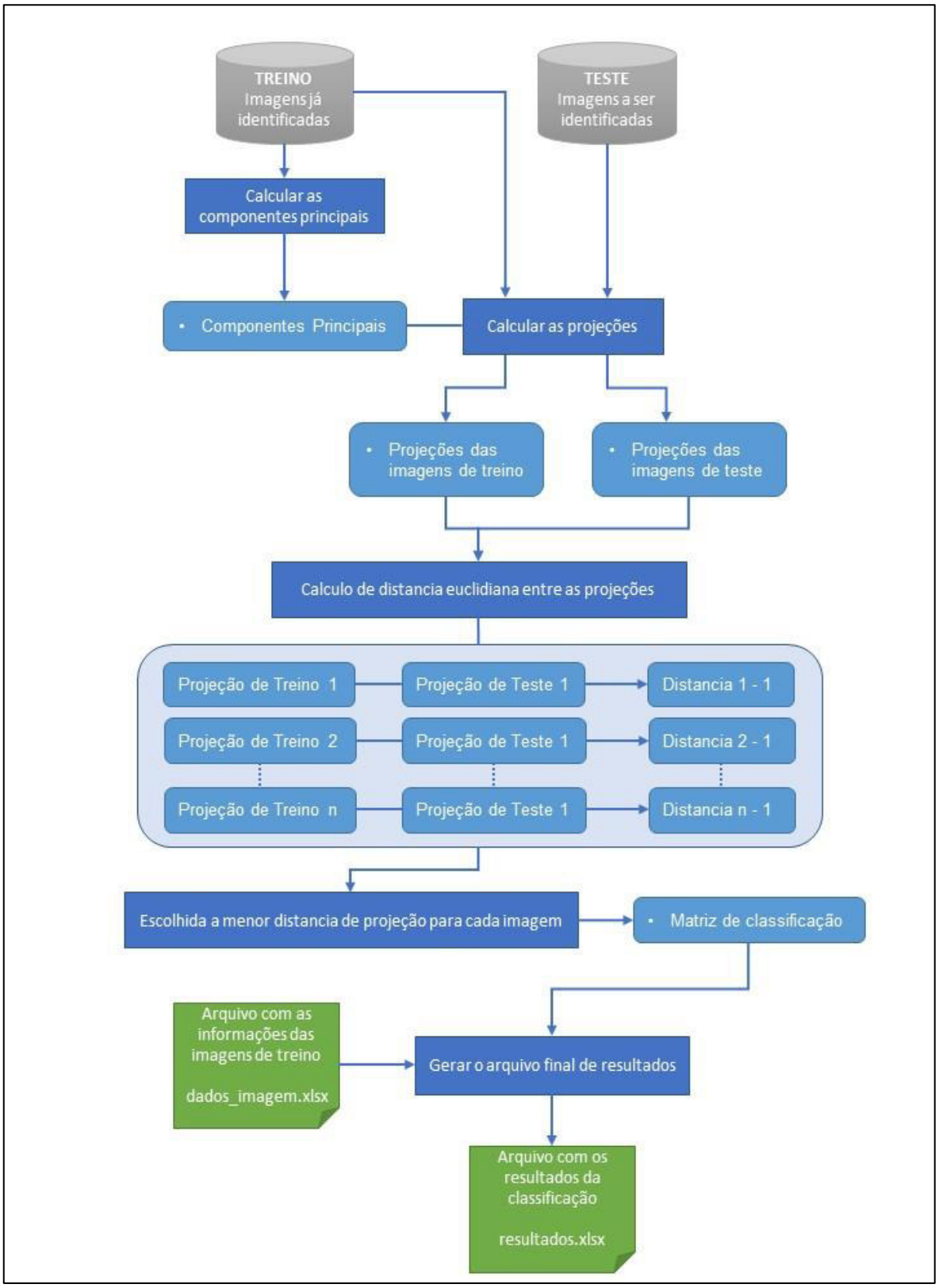
(ii) Com as Imagens de Treino alocadas no formato de matriz de *pixels*, são determinadas as componentes principais;

(iii) Procede-se então à projeção de todas as Imagens de Treino e Imagem de Teste nas componentes principais;

(iv) Com base nestas projeções, determina-se a distância euclidiana entre a Imagem de Teste e todas as Imagens de Treino;

(v) Finalmente, o processo de classificação é baseado na menor distância euclidiana verificada entre as duplas.

Figura 15: Algoritmo de Reconhecimento de Imagem.



Fonte: O Autor

5.3. Matriz de Dados

A leitura de dados foi realizada por intermédio de comandos próprios do MATLAB™. Com o objetivo de interpretar uma imagem de tipo ".jpg" e manter uma uniformidade nas escalas adotadas para as figuras que deverão ser alocadas na matriz de dados, procedeu-se em utilizar a escala de cor cinza e alterar o tamanho da figura para uma dimensão de 120×80.

A Figura 16 exibe os comandos utilizados para a leitura de dados (*imread*), transformação para cor cinza (*rgb2gray*) e redimensionamento da figura (*imresize*). O tratamento de uma figura colorida para uma imagem em tons de cinza reduz o custo computacional, desde que, ao invés de se utilizar uma matriz de imagem com três dimensões ($120 \times 80 \times 3$), representativas das cores vermelho, verde e azul, é utilizada uma imagem que apresenta dimensão $120 \times 80 \times 1$. Os comandos utilizados na Figura 16 dizem respeito à leitura de 4 (quatro) imagens da atriz Angelina Jolie.

Figura 16: Aquisição de dados.

```
% Carrega as imagens
% Angelina
A1 = imresize(double(rgb2gray(imread('angelina1','jpeg'))),[120 80]);
A2 = imresize(double(rgb2gray(imread('angelina2','jpeg'))),[120 80]);
A3 = imresize(double(rgb2gray(imread('angelina3','jpeg'))),[120 80]);
A4 = imresize(double(rgb2gray(imread('angelina4','jpeg'))),[120 80]);
```

Fonte: O Autor

Neste trabalho foram utilizadas quatro imagens, editadas com o objetivo de focar a região da cabeça e rosto, de cinco pessoas diferentes, perfazendo um total de vinte imagens. As imagens foram colocadas em uma matriz de dados $\boldsymbol{MD_i}$ de forma que cada linha da matriz esteja associada a todas as informações de apenas uma imagem. Esta matriz está representada na Equação (31).

$$\boldsymbol{MD_i} = \begin{bmatrix} F_1^1 & \cdots & F_{120*80}^1 \\ \vdots & \ddots & \vdots \\ F_{120*80}^{20} & \cdots & F_{120*80}^{20} \end{bmatrix} \begin{matrix} Figura\ 1 \\ \vdots \\ Figura\ 20 \end{matrix} \qquad (31)$$

Observe que esta matriz de dados contém todas as imagens que serão utilizadas na análise de reconhecimento. A matriz $\boldsymbol{MD_i}$ apresenta vinte linhas e número de colunas calculado em função das dimensões da figura, quais sejam, 120

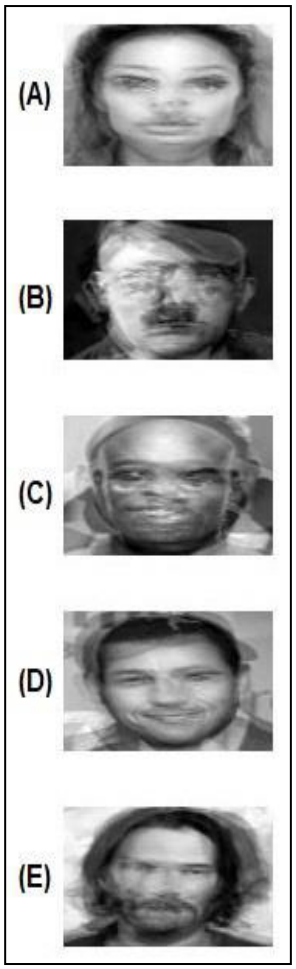
x 80. As imagens utilizadas para o desenvolvimento do código estão dispostas na Figura 17.

Figura 17: Faces utilizadas para o desenvolvimento do código.

Fonte: Composição do autor

Na implementação inicial do programa (Apêndice B), também foi gerada uma face “média” associada à cada uma das cinco pessoas. Esta matriz de dados será utilizada quando no emprego do Método PCA, nas projeções dos autovetores. Os rostos médios podem ser visualizados na Figura 18.

Figura 18: Faces “médias” das imagens utilizadas.



Fonte: O Autor

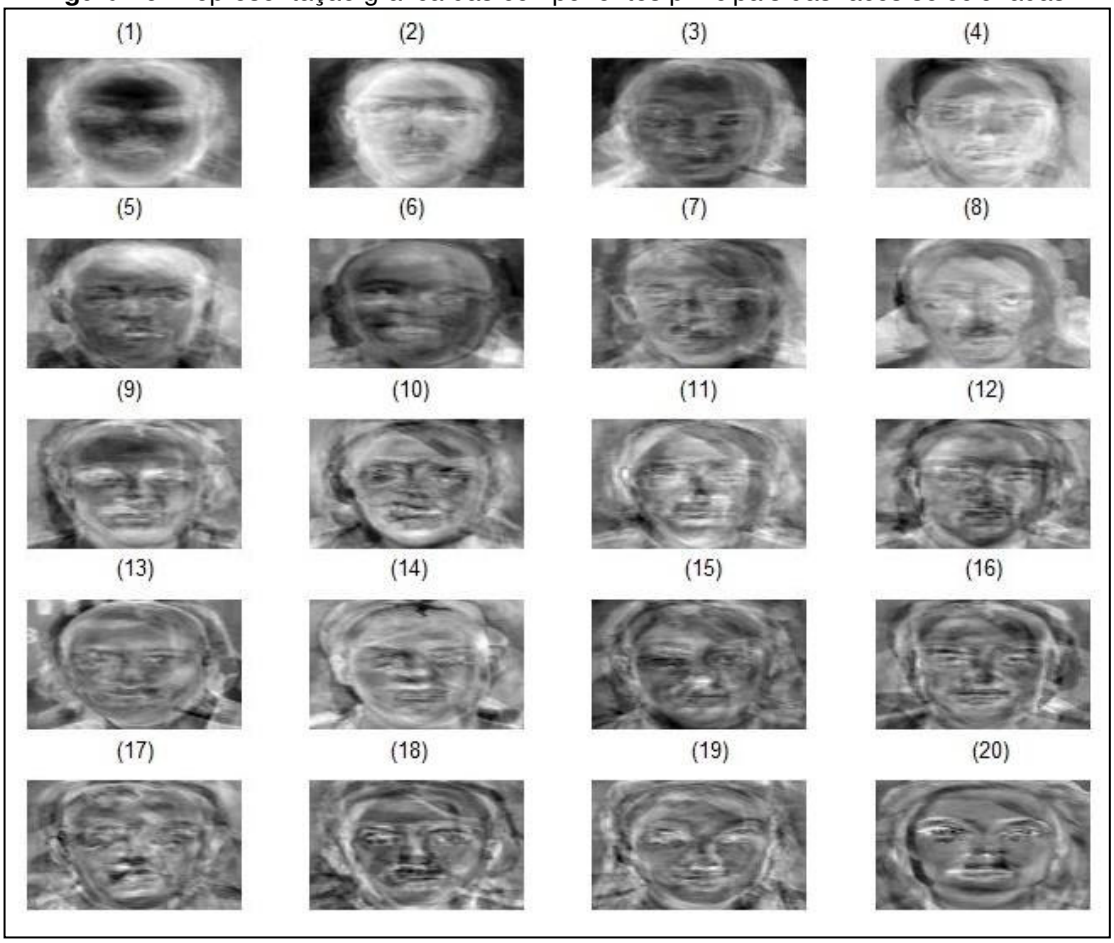
5.4. Componentes Principais e Valores Singulares

Após criadas as Imagens associadas às faces médias e a matriz com as imagens, foi gerada a matriz de correlação entre as imagens $\boldsymbol{MC}$.

$$\boldsymbol{MC} = \boldsymbol{MD_i}^{\boldsymbol{T}} * \boldsymbol{MD_i} \quad (32)$$

A partir da matriz de correlação foram obtidos os autovalores e autovetores na forma das componentes principais. A representação gráfica de cada um das componentes principais é observada na Figura 19, nas quais as componentes principais estão dispostas em ordem decrescente, começando com o maior componente no canto superior esquerdo.

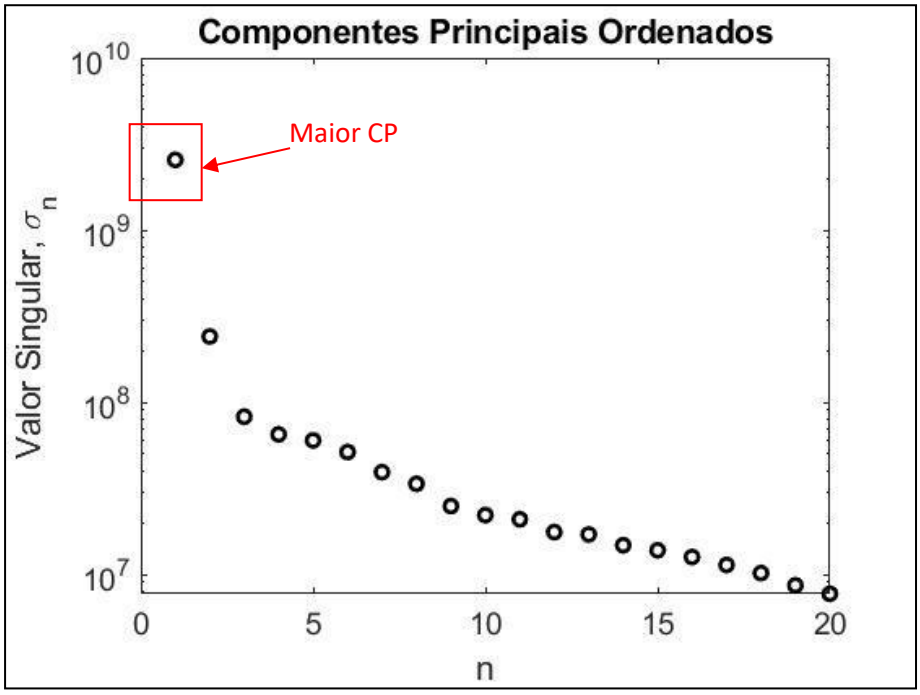
Figura 19: Representação gráfica das componentes principais das faces selecionadas.



Fonte: O Autor

A maior componente principal está relacionada a itens comuns a todas imagens. Sendo assim, o primeiro componente principal mostra as feições que são compartilhadas por todas as imagens. Para fazer uma classificação, este primeiro componente principal deve ser descartado ou as figuras devem ser centralizadas (retirando as médias) para remover estas características comuns. Fazendo a projeção dos componentes principais em um eixo logarítmico, fica visível a diferença do primeiro componente principal para os demais, Figura 20.

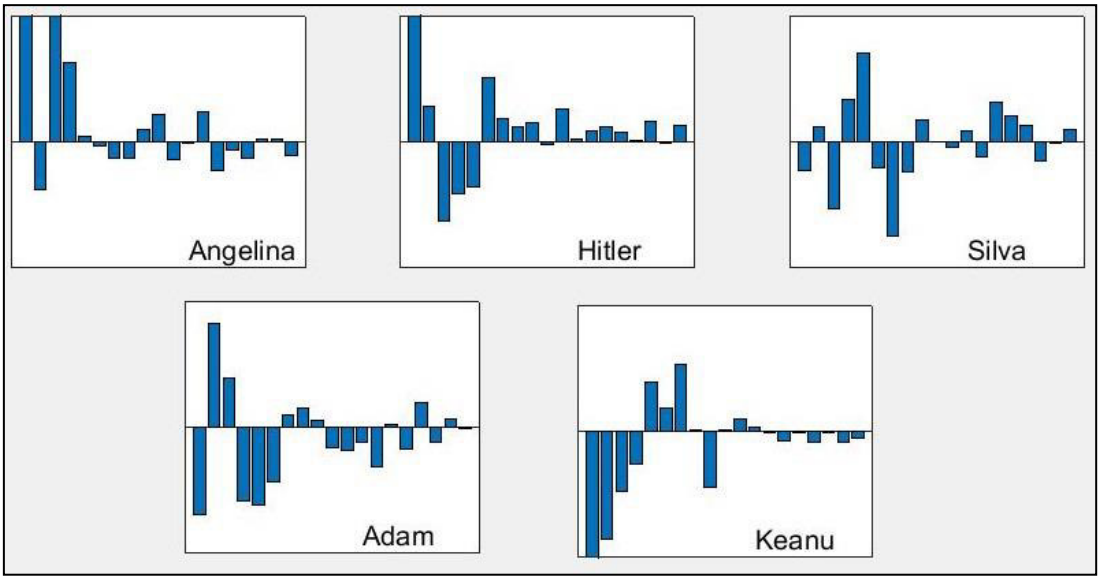
Figura 20: Componentes principais ordenadas de forma decrescente.



Fonte: O Autor

Em seguida, foi feita a projeção dos rostos médios no subespaço das componentes principais, retirando o valor da projeção da primeira componente. Foram obtidos os gráficos observados na Figura 21.

Figura 21: Projeções das faces médias nas componentes principais.

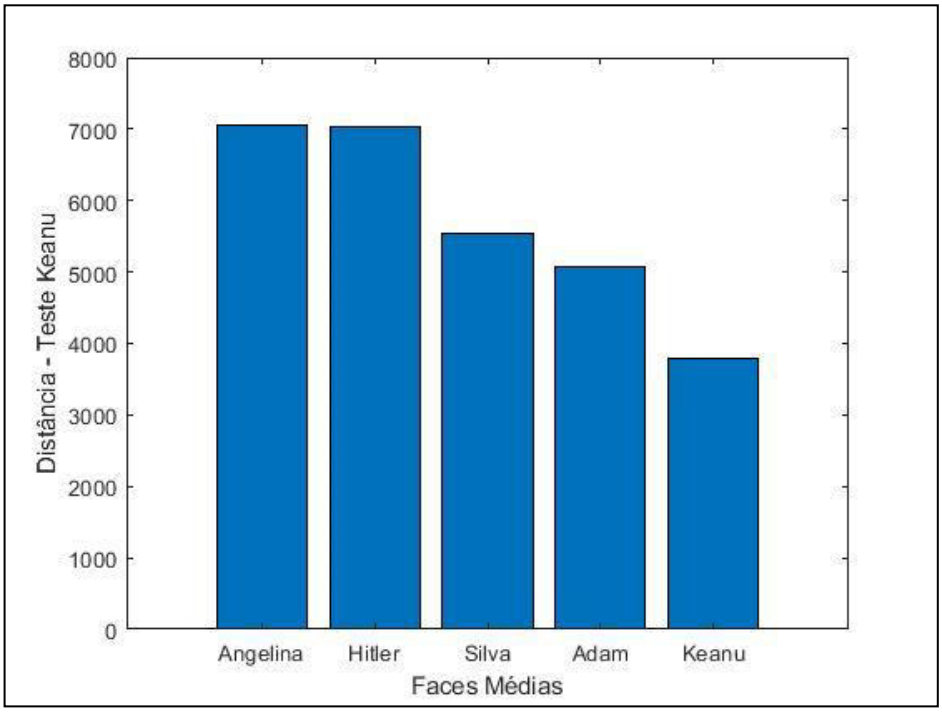


Fonte: O Autor

Cada projeção pode ser interpretada como uma chave única para identificar os rostos, ou seja, cada projeção é a “assinatura” da respectiva face. Para validar o funcionamento do código, foi colocada uma nova foto de rosto, de uma pessoa que já possuía um rosto médio (nova foto de Keanu Reeves). A nova imagem foi projetada no subespaço das componentes principais já existentes e calculou-se a distância euclidiana entre as projeções dos rostos médios (removendo o primeiro componente). O resultado do cálculo das distâncias é mostrado na Figura 22.

O resultado observado na Figura 22 mostra uma menor distância para a nova imagem de Keanu, sendo a projeção de seu próprio rosto médio. Assim, a classificação por análise de componentes principais pode ser feita através da distância euclidiana entre as projeções. A comparação entre as projeções também mostrou que a as projeções podem ser parecidas, mas possuem algumas diferenças significativas. Estas diferenças podem estar relacionadas a forma em que a figura foi editada para centralizar os rostos ou mesmo objetos que estavam ao fundo das imagens.

Figura 22: Distância euclidiana para nova imagem adicionada.



Fonte: O Autor

5.5. Caracterização da Classificação

Após a classificação da imagem teste, procedeu-se à sua caracterização. Cada Imagem apresenta características particulares e estão alocadas em uma planilha EXCEL®. De fato, são utilizados dois arquivos, quais sejam: (*i*) "*dados_imagem.xlsx*": nestas existem informações associadas à cada uma das imagens que serão utilizadas no processo de classificação e (*ii*) arquivo "*resultados.xlsx*": este arquivo é preenchido automaticamente ao executar o código MATLAB™ com as informações de classificação das imagens. A Tabela 1 exibe um exemplo de preenchimento do arquivo "*dados_imagem.xlsx*".

Tabela 1: Preenchimento do arquivo "dados_imagem.xlsx".

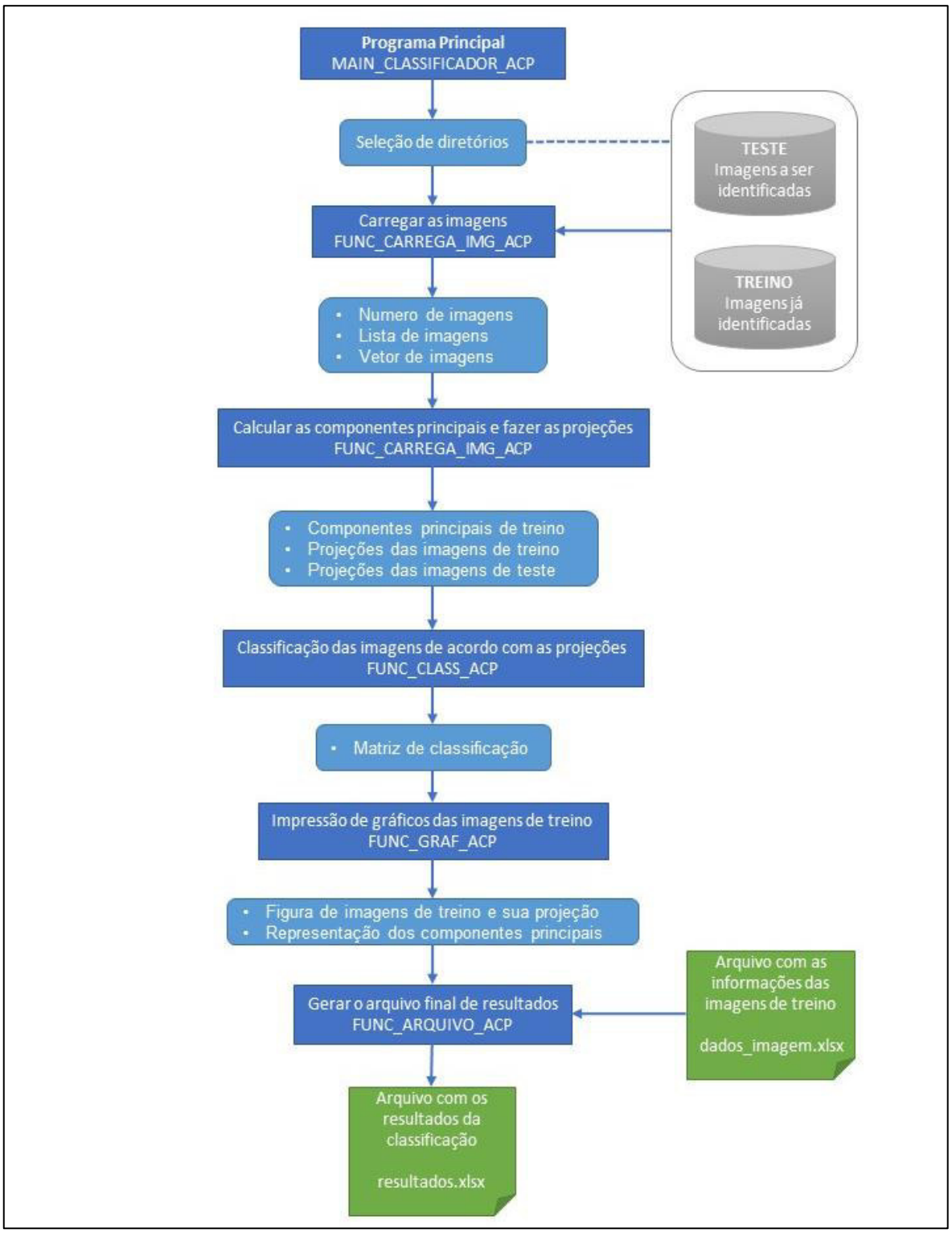
Classe	Peso	Altura	Sexo	Nacionalidade	Ano
angelina	35	1,69	F	Estados Unidos	2016
anderson	84	1,88	M	Brasil	2010
hitler	68	1,75	M	Áustria	1945
adam	75	1,77	M	Estados Unidos	2015
keanu	79	1,86	M	Líbano	2009

Fonte: O Autor

5.6. Implementação Numérica do Algoritmo

Neste tópico são apresentadas as funções desenvolvidas na plataforma MATLAB™. A Figura 23 exibe o algoritmo apresentado anteriormente com as respectivas funções desenvolvidas.

Figura 23: Programas Desenvolvidos na Implementação Numérica.



Fonte: O Autor

O código principal de implementação numérica é encontrado no Apêndice C (MAIN_CLASSIFICADOR_ACP.m), e se inicia com uma chamada dos diretórios das imagens que serão usadas para classificação e as imagens que serão classificadas (Figura 24). Desta forma, o programa tem uma etapa de “treino” na qual utiliza as imagens do diretório de treino para calcular os componentes principais.

Figura 24: Código para definir diretório e chamar função de carregar imagem.

```
% Selecionar o diretório de imagens para treino
dir_treino = uigetdir('C:\','Diretório de imagens para treino');
% Selecionar o diretório de imagens para teste
dir_classif = uigetdir('C:\','Diretório de imagens para classificar');
% Carrega as imagens
[n_img_T,D_img_treino,lista_img_T] = FUNC_CARREGA_IMG(dir_treino);
[n_img_C,D_img_class,lista_img_C] = FUNC_CARREGA_IMG(dir_classif);

fprintf('A lista de Imagens utilizadas para Classificação: \n\n')
fprintf('%s \n',lista_img_T(:))
```

Fonte: O Autor

Após a seleção dos diretórios, o programa executa uma função para carregar as imagens do local selecionado. A função utilizada é FUNC_CARREGA_IMG.m, cujo código se encontra no Apêndice D, e tem como meio de funcionamento o mesmo conceito utilizado na seção 5.3 observado na Figura 16. A função tem como saída uma matriz de imagem e uma lista com o nome dos arquivos. O código principal exibe na tela de comandos a lista de imagens utilizadas para classificação (Tabela 2)

Tabela 2: Exemplo de saída de dados após carregadas as imagens para classificação.

A lista de Imagens utilizadas para Classificação:
adam2.jpg
anderson2.jpg
angelina2.jpg
hitler1.jpg
keanu2.jpg

Fonte: O Autor

Após carregar as imagens, o código passa as matrizes de imagem para uma função responsável pelo cálculo das componentes principais e das projeções. A função de cálculo FUNC_CALC_ACP.m se encontra no Apêndice E. O cálculo das

componentes principais é feito utilizando as imagens de treino e as projeções são feitas utilizando as imagens de treino e as imagens a ser classificadas. A execução deste código difere do código desenvolvido inicialmente (Apêndice B) pelo fato de não utilizar um rosto médio para fazer as projeções, mas apenas as imagens colocadas na pasta de treino.

Após executada a função de cálculo de componentes principais, o programa segue para fazer a classificação das imagens utilizando a função FUNC_CLASS_ACP.m, encontrada no Apêndice F. Esta função faz a classificação a partir da distância euclidiana entre as projeções das imagens nos componentes principais. A função calcula a distância euclidiana da imagem a ser classificada com cada uma das imagens utilizadas como treino e associa a imagem com menor distância (Tabela 3).

Tabela 3: Exemplo de saída de dados de classificação na janela de comandos do MATLAB.

Classificação das imagens:			
Imagem analisada	**Classe**	**Distância**	**Imagem mais próxima**
adam1.jpg	adam	2.343,0597	adam2.jpg
adam2.jpg	adam	1.440,9257	adam2.jpg
anderson1.jpg	anderson	3.192,8660	anderson2.jpg
anderson2.jpg	anderson	4.561,0940	anderson2.jpg
anderson4.jpg	anderson	2.704,7230	anderson2.jpg
angelina1.jpg	angelina	2.031,0418	angelina2.jpg
angelina2.jpg	angelina	1.619,8517	angelina2.jpg
angelina3.jpg	angelina	1.870,2749	angelina2.jpg
angelina4.jpg	angelina	3.356,6803	angelina2.jpg
hitler1.jpg	hitler	1.056,1894	hitler1.jpg
hitler3.jpg	hitler	3.050,5147	hitler1.jpg
keanu2.jpg	keanu	6.914,1330	keanu2.jpg
keanu3.jpg	keanu	3.429,6057	keanu2.jpg

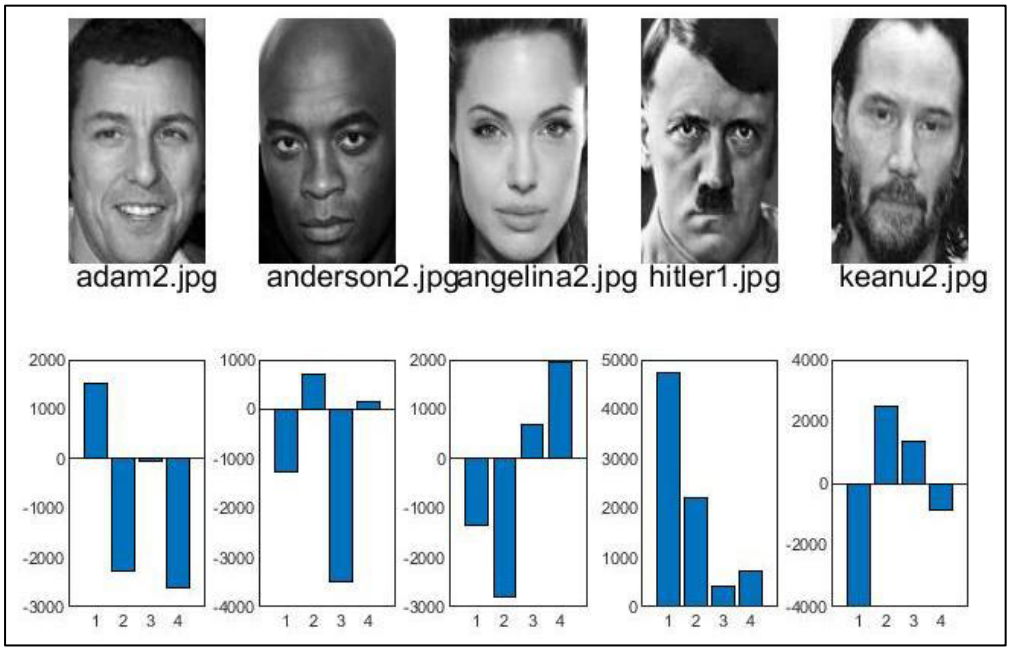
Fonte: O Autor

O cálculo da distância também desconsidera o efeito da primeira componente principal. A função dá como saída uma lista na janela de comando: uma tabela com a informação da imagem analisada, sua classificação (Classe), a distância e o nome da imagem mais próxima.

Feita a classificação, o código principal executa a função responsável pela saída dos gráficos FUNC_GRAF_ACP.m, encontrada no Apêndice G. Essa função cria duas figuras, uma com as imagens utilizadas para classificação junto a sua projeção nos componentes principais (Figura 25) e outra com a representação

gráfica dos componentes principais junto a um gráfico logarítmico dos mesmos (Figura 26).

Figura 25: Exemplo de saída de gráficos do programa final com as figuras utilizadas para classificação e as projeções das imagens nas componentes principais (excluindo a primeira componente).



Fonte: O Autor

Figura 26: Exemplo de saída de gráficos do programa final com a representação gráfica das componentes principais junto aos valores singulares das componentes principais dispostos em escala logarítmica.

Fonte: O Autor

As representações gráficas são formadas com base nos mesmos conceitos observados na seção 5.4, onde o primeiro rosto (da esquerda para direita) na Figura 26 representa os elementos comuns entre todos os demais.

A última etapa do programa principal é a utilização da função que gera o arquivo com todas as informações de classificação FUNC_ARQUIVO_ACP.m, que se encontra no Apêndice H. O código utiliza a planilha de Excel dados_imagem.xlsx para obter os dados e classificar na planilha (sobrescrevendo o arquivo) resultados.xlsx. A classificação feita com os prefixos de nomes garante uma classe

única para várias imagens e assim, a classificação fica associada à classe e não somente à imagem mais próxima. O exemplo de saída de dados da função é mostrado na Tabela 4.

Tabela 4: Exemplo de saída de dados para o arquivo resultados.xlsx após a execução do programa final de classificação de imagens. Algumas distâncias de imagens com mesmo nome não são iguais a zero, pois foram modificadas antes de serem colocadas na pasta treino.

Imagem analisada	Distancia	Imagem mais próxima	Classe	Peso	Altura	Sexo	Nacionalidade	Ano
adam1.jpg	2.343,0597	adam2.jpg	adam	75	1.77	M	Estados Unidos	2015
adam2.jpg	1.440,9257	adam2.jpg	adam	75	1.77	M	Estados Unidos	2015
anderson1.jpg	3.192,8660	anderson2.jpg	anderson	84	1.88	M	Brasil	2010
anderson2.jpg	4.561,0940	anderson2.jpg	anderson	84	1.88	M	Brasil	2010
anderson4.jpg	2.704,7230	anderson2.jpg	anderson	84	1.88	M	Brasil	2010
angelina1.jpg	2.031,0418	angelina2.jpg	angelina	35	1.69	F	Estados Unidos	2016
angelina2.jpg	1.619,8517	angelina2.jpg	angelina	35	1.69	F	Estados Unidos	2016
angelina3.jpg	1.870,2749	angelina2.jpg	angelina	35	1.69	F	Estados Unidos	2016
angelina4.jpg	3.356,6803	angelina2.jpg	angelina	35	1.69	F	Estados Unidos	2016
hitler1.jpg	1.056,1894	hitler1.jpg	hitler	68	1.75	M	Áustria	1945
hitler3.jpg	3.050,5147	hitler1.jpg	hitler	68	1.75	M	Áustria	1945
keanu2.jpg	6.914,1330	keanu2.jpg	keanu	79	1.86	M	Líbano	2009
keanu3.jpg	3.429,6057	keanu2.jpg	keanu	79	1.86	M	Líbano	2009

Fonte: O Autor

O resultado final na Tabela 4 é uma junção das informações preenchidas na Tabela 1 com as informações obtidas da Tabela 3. Para executar o código novamente, é recomendado apagar os dados da tabela de resultados antes de executar o programa. Os resultados obtidos da utilização do programa são discutidos na próxima seção.

5.7. Saídas do Programa

A utilização do método para classificação apresentou resultados positivos para sua primeira utilização, vista na Figura 22. A partir deste primeiro teste, foram feitos mais testes utilizando o programa final. Realizaram-se alguns testes nos quais foram alteradas algumas condições das imagens para entender como o classificador iria reagir. O primeiro teste com o programa final foi feito colocando

uma figura de cada pessoa no diretório de treino e testando a inclusão das outras imagens. O espaço amostral de imagens se manteve o mesmo desde os primeiros testes com figuras. Todas as imagens utilizadas (exceto a imagem de teste do Keanu Reeves) são mostradas na Figura 17. O Tempo de execução do programa ficou na faixa de 6 segundos (Ryzen 5 3600).

O primeiro teste realizado utilizou as figuras sem modificação do seu formato inicial. Isso resultou em uma classificação ruim por parte do programa: de um total de quinze figuras, apenas quatro foram classificadas corretamente (acurácia de 27%). Os resultados da primeira classificação se encontram na Tabela 5.

Tabela 5: Primeiro teste de classificação de imagem com código final.

Classificação das imagens:			
Imagem analisada	**Classe**	**Distância**	**Imagem mais próxima**
adam1.jpg	angelina	4.811,9440	angelina11.jpg
adam3.jpg	angelina	4.492,2137	angelina11.jpg
adam4.jpg	keanu	4.509,9369	keanu11.jpg
anderson1.jpg	adam	4.923,5925	adam11.jpg
anderson3.jpg	hitler	4.269,9022	hitler11.jpg
anderson4.jpg	keanu	3.781,9556	keanu11.jpg
angelina1.jpg	adam	4.438,2289	adam11.jpg
angelina3.jpg	adam	4.427,4300	adam11.jpg
angelina4.jpg	angelina	3.824,9127	angelina11.jpg
hitler2.jpg	adam	3.611,5031	adam11.jpg
hitler3.jpg	hitler	3.039,6668	hitler11.jpg
hitler4.jpg	hitler	2.669,4870	hitler11.jpg
keanu1.jpg	anderson	4.274,6591	anderson11.jpg
keanu3.jpg	keanu	4.560,5440	keanu11.jpg
keanu4.jpg	anderson	5.189,1518	anderson11.jpg

Fonte: O Autor

O resultado da classificação está diretamente relacionado ao número de imagens utilizadas para o treino, assim como a forma de edição da figura antes da realização dos testes. Um segundo teste foi realizado invertendo as imagens utilizadas para classificação (treino) com as imagens utilizadas para teste, possibilitando ver o impacto de várias figuras sendo utilizadas para treinar o programa. Ao realizar o segundo teste os números estariam invertidos, haveriam cinco imagens para classificação e quinze imagens para criar os componentes principais e ser utilizadas para classificação. O resultado do segundo teste é observado na Tabela 6 e mostrou que a classificação das imagens continuava com

uma taxa de erro muito alta, onde dessa vez duas das cinco imagens foram classificadas corretamente (acurácia de 40%).

Tabela 6: Resultado do segundo teste invertendo as imagens de treino com as imagens de classificação.

Classificação das imagens:			
Imagem analisada	**Classe**	**Distância**	**Imagem mais próxima**
adam11.jpg	hitler	3.528,8715	hitler2.jpg
anderson11.jpg	keanu	4.021,7897	keanu4.jpg
angelina11.jpg	angelina	4.303,4011	angelina4.jpg
hitler11.jpg	hitler	4.557,8371	hitler2.jpg
keanu11.jpg	anderson	4.511,1838	anderson4.jpg

Fonte: O Autor

O resultado observado na Tabela 6 mostra que apenas aumentar o número de imagens não garante uma classificação correta. O resultado já era esperado, uma vez que as imagens mais próximas da Tabela 6 também possuíam as menores distâncias em relação as imagens analisadas no primeiro teste. Sendo assim, foi testada uma centralização melhor dos rostos (editando as imagens) para verificar a saída dos resultados. Para fazer as alterações, foi necessário adaptar consideravelmente as imagens de modo que a imagem de fundo fosse parcialmente removida. Feitas as alterações, foi realizado um novo teste utilizando as mesmas imagens dos testes anteriores (tanto para o treino como para classificação) editadas. O resultado do novo teste com as figuras editadas é observado na Tabela 7.

Tabela 7: Terceiro teste de classificação de imagem com código final com figuras editadas para dar ênfase nos rostos.

Classificação das imagens:			
Imagem analisada	**Classe**	**Distância**	**Imagem mais próxima**
adam1.jpg	adam	2.941,6660	adam22.jpg
adam3.jpg	adam	4.228,5824	adam22.jpg
adam4.jpg	adam	4.292,6987	adam22.jpg
anderson1.jpg	anderson	3.675,5414	anderson22.jpg
anderson3.jpg	anderson	4.219,5216	anderson22.jpg
anderson4.jpg	anderson	3.668,2739	anderson22.jpg
angelina1.jpg	angelina	1.674,4436	angelina22.jpg
angelina3.jpg	angelina	1.406,6592	angelina22.jpg
angelina4.jpg	angelina	3.680,3873	angelina22.jpg
hitler2.jpg	adam	3.250,2530	adam22.jpg

hitler3.jpg	hitler	2.642,2910	hitler22.jpg
hitler4.jpg	hitler	4.145,6445	hitler22.jpg
keanu1.jpg	angelina	3.354,3839	angelina22.jpg
keanu3.jpg	angelina	1.964,8424	angelina22.jpg
keanu4.jpg	keanu	3.815,4788	keanu4.jpg

Fonte: O Autor

Fazendo as modificações para reduzir a influência de objetos externos aos rostos das imagens utilizadas, houve uma melhora significativa nos resultados obtidos. Doze dentre as quinze imagens foram classificadas corretamente (acurácia de 80%). Essa melhora significativa mostra a sensibilidade do programa com os as imagens colocadas. Outros testes menores evidenciaram um impacto significativo nas distâncias observadas, através de uma pequena mudança na área recortada. Fatores como plano de fundo e cabelo foram observados influenciando diretamente nos resultados da classificação. Sendo assim, para fazer uma classificação por imagem utilizando o método das componentes principais é necessário ter um conjunto de dados de imagens uniformes, uma vez que a área da figura analisada tem grande impacto no resultado final. Uma melhoria possível para o programa é a utilização de uma função para identificar e separar os rostos ou objeto de estudo a ser analisado, permitindo melhorar a implementação do método para classificação. Em resumo, o método das componentes principais pode ser usado para diversas aplicações e o mesmo se mostra viável para fazer a classificação de imagens, desde que a área de interesse de classificação esteja claramente visível para o programa.

CONCLUSÕES E CONSIDERAÇÕES FINAIS

Neste trabalho a Metodologia PCA foi descrita e implementada numericamente. Dois códigos computacionais foram desenvolvidos, quais sejam: (*i*) Implementação Didática do Método (PCA), descrevendo todas as etapas utilizadas no algoritmo numérico previstas para o referido método e (*iii*) Implementação do Método (PCA) no estudo de Reconhecimento de Imagens.

Uma das principais preocupações na elaboração deste trabalho foi disponibilizar um código computacional que auxilie, de forma direta, o desenvolvedor iniciante. O primeiro programa disponibilizado no Anexo A atende à este requisito.

Outra preocupação deste trabalho foi o entendimento dos principais conceitos envolvidos na área de Reconhecimento Facial utilizando o Método PCA. No Capítulo 3 estão descritos, com certo rigor, todos os conceitos utilizados na implementação do Método PCA.

A partir da pesquisa realizada e dos resultados dos programas desenvolvidos, pode-se constatar que o Método PCA é uma técnica versátil e útil nos dias atuais. Ela apresenta diversas aplicações dentro da análise de dados com múltiplas variáveis. Dentre suas aplicações, um dos destaques pode ser dado para sua utilização como classificador de imagens, um tema amplamente abordado nos tempos atuais e que já apresenta aplicações no mercado.

Nos primeiros testes utilizados no tratamento de imagens, foi possível constatar que existe uma correspondência biunívoca as imagens utilizadas como referência e suas respectivas projeções no espaço das componentes principais. Utilizando este conceito, foi possível estabelecer um método de classificação baseado nas distâncias euclidianas entre as projeções das imagens teste e as imagens de referência, deixando claro que a metodologia é eficiente e pode ser facilmente aprimorada.

Como sugestão de trabalhos futuros, destacam-se, entre outros:

i. Estudo e Implementação de Método de Classificação em um contexto Tridimensional utilizando o Método PCA;

ii. Implementação do Método PCA aplicado à determinação de geometria de veículos baseado em Imagem;

iii. Implementação do Método de Reconhecimento de Imagens aplicado ao estudo de requisitos de artefatos bélicos;

REFERÊNCIAS

ANDRADE, Rodrigo de Oliveira. As máquinas que tudo veem. **Revista Fapesp**, v. 277, p. 70-73, 2019.

BRUNTON, Steven L.; KUTZ, J. Nathan. **Data-driven science and engineering: Machine learning, dynamical systems, and control**. Cambridge University Press, 2019.

DUARTE, João Batista. INTRODUÇÃO À ANÁLISE DE COMPONENTES PRINCIPAIS. **URL: https://files.cercomp.ufg.br/weby/up/396/o/ACP.pdf,** 1988.

GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing. 2nd edn Prentice Hall. **New Jersey**, v. 793, 2002.

GOTTUMUKKAL, Rajkiran; ASARI, Vijayan K. An improved face recognition technique based on modular PCA approach. **Pattern Recognition Letters**, v. 25, n. 4, p. 429-436, 2004.

HONGYU, Kuang. **Distribuição empírica dos autovalores associados à matriz de interação dos modelos AMMI pelo método bootstrap não-paramétrico**. Tese de Doutorado. Universidade de São Paulo, 2012.

HONGYU, Kuang; SANDANIELO, Vera Lúcia Martins; DE OLIVEIRA JUNIOR, Gilmar Jorge. Análise de componentes principais: resumo teórico, aplicação e interpretação. **E&S Engineering and Science**, v. 5, n. 1, p. 83-90, 2016.

JUNQUEIRA, Luiz C.; CARNEIRO, José. **Histologia Básica. Texto e Atlas**. 11ª edição. Rio de Janeiro: Guanabara, p. 459, 2008.

MATHWORKS Matlab, R2018a, MathWorks, 1994. Conjunto de Programas.

OLIVER, J. 1 Vídeo (21 min). **Facial Recognition: Last Week Tonight with John Oliver (HBO).** Publicado pelo canal LastWeekTonight, 2019. Disponivel em: <https://www.youtube.com/watch?v=jZjmlJPJgug>. Acesso em: 15 jun. 2020

POOLE, David. **Álgebra linear**. Cengage Learning Editores, 2004.

SANTANA, Deise Maia. **Detecção e reconhecimento de face utilizando Matlab**. Trabalho de conclusão de curso (graduação) - Universidade Estadual do sudoeste da Bahia, Curso de Ciência da Computação, BA, 2014.

SHLENS, Jonathon. A tutorial on principal component analysis. **arXiv preprint arXiv:1404.1100**, 2014.

ZHAO, Wenyi et al. Face recognition: A literature survey. **ACM computing surveys (CSUR)**, v. 35, n. 4, p. 399-458, 2003.

APÊNDICE A – Programa do Capítulo 3

```
%
/////////////////////////////////////////////////////////////////////
/////
% Principal Components of Analysis - PCA
% Vinícius Rodrigues Costa - USP - Engenharia Física
%
% Based on tutotial
%
www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components
.pdf
% Core Code by SIAMAK FARIDANI
%
% Autors: Vinicius - 07/2020
%         Maurício - 07/2020
%
% Status: Ok !
%
/////////////////////////////////////////////////////////////////////
/////
clc;
close all;
clear all;

% ////////////////////////////////// INPUT
% Should be even
numdata= 100;

tic
% Step 1, generating a dataset
x1= rand(numdata/2,1);
y1= rand(numdata/2,1);
x2= 1.5*rand(numdata/2,1)+2;
y2= 1.5*rand(numdata/2,1)+2;

% A matrix
x= [x1;x2];
y= [y1;y2];
```

```
% ////////////////////////////////// SOLVER
% Step 2, finding a mean and subtracting
xmean= mean(x);
ymean= mean(y);

xnew= x-xmean*ones(numdata,1);
ynew= y-ymean*ones(numdata,1);

% Step 3, covariance matrix
covariancematrix= cov(xnew,ynew);

% Step 4, Finding Eigenvectors
[V,D] = eig(covariancematrix);
D= diag(D);
P= V(:,find(D==max(D))); %maxeigval

% Step 5, Deriving the new data set
% finding the projection onto the eigenvectors
finaldata= P'*[xnew,ynew]';
```

```
% ////////////////////////////////// OUTPUT
subplot(2,2,1);
plot(x,y, 'o');
title('Original Data');
grid on

% Mean is deducted
subplot(2,2,3);
plot(xnew,ynew, 'o');
title('Mean is Deducted')
grid on

subplot(2,2,2);
stem(finaldata, 'DisplayName', 'finaldata', 'YDataSource',
'finaldata');
title('PCA 1D output ')
grid on

% We do a classification now
subplot(2,2,4);
title('Final Classification')
hold on

for i= 1:size(finaldata,2)
    if  finaldata(i)>=0
        plot(x(i),y(i),'o')
        plot(x(i),y(i),'r*')

    else
        plot(x(i),y(i),'o')
        plot(x(i),y(i),'g*')
    end
end
grid on
```

APÊNDICE B – Implementação inicial de ACP para imagens (imagem_acp.m)

```
%% base de imagens
% Implementação Inicial acp com imagens (Código de teste inicial)
% Vinicius Rodrigues Costa
clear all; close all; clc
% A = Angelina, H = Hitler, S = Anderson Silva, D = Adam Sandler, K =
Keanu
% Carrega as imagens
% Angelina
A1 = imresize(double(rgb2gray(imread('angelina1','jpeg'))),[120 80]);
A2 = imresize(double(rgb2gray(imread('angelina2','jpeg'))),[120 80]);
A3 = imresize(double(rgb2gray(imread('angelina3','jpeg'))),[120 80]);
A4 = imresize(double(rgb2gray(imread('angelina4','jpeg'))),[120 80]);
% Adolf Hitler
H1 = imresize(double(rgb2gray(imread('hitler1','jpeg'))),[120 80]);
H2 = imresize(double(rgb2gray(imread('hitler2','jpeg'))),[120 80]);
H3 = imresize(double(rgb2gray(imread('hitler3','jpeg'))),[120 80]);
H4 = imresize(double(rgb2gray(imread('hitler4','jpeg'))),[120 80]);
% Anderson silva
S1 = imresize(double(rgb2gray(imread('anderson1','jpeg'))),[120 80]);
S2 = imresize(double(rgb2gray(imread('anderson2','jpeg'))),[120 80]);
S3 = imresize(double(rgb2gray(imread('anderson3','jpeg'))),[120 80]);
S4 = imresize(double(rgb2gray(imread('anderson4','jpeg'))),[120 80]);
% Adam Sandler
D1 = imresize(double(rgb2gray(imread('adam1','jpeg'))),[120 80]);
D2 = imresize(double(rgb2gray(imread('adam2','jpeg'))),[120 80]);
D3 = imresize(double(rgb2gray(imread('adam3','jpeg'))),[120 80]);
D4 = imresize(double(rgb2gray(imread('adam4','jpeg'))),[120 80]);
% Keanu Reeves
K1 = imresize(double(rgb2gray(imread('keanu1','jpeg'))),[120 80]);
K2 = imresize(double(rgb2gray(imread('keanu2','jpeg'))),[120 80]);
K3 = imresize(double(rgb2gray(imread('keanu3','jpeg'))),[120 80]);
K4 = imresize(double(rgb2gray(imread('keanu4','jpeg'))),[120 80]);
K5 = imresize(double(imread('keanu4','jpeg')),[120 80]);
% Plotar as imagens

subplot(5,4,1),  pcolor(flipud(A1)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,2),  pcolor(flipud(A2)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,3),  pcolor(flipud(A3)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,4),  pcolor(flipud(A4)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,5),  pcolor(flipud(H1)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,6),  pcolor(flipud(H2)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,7),  pcolor(flipud(H3)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,8),  pcolor(flipud(H4)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,9),  pcolor(flipud(S1)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,10), pcolor(flipud(S2)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,11), pcolor(flipud(S3)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,12), pcolor(flipud(S4)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
```

```
subplot(5,4,13), pcolor(flipud(D1)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,14), pcolor(flipud(D2)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,15), pcolor(flipud(D3)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,16), pcolor(flipud(D4)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,17), pcolor(flipud(K1)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,18), pcolor(flipud(K2)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,19), pcolor(flipud(K3)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,20), pcolor(flipud(K4)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])

%% Rosto médio


rm_A = (A1 + A2 + A3 + A4)/4;
rm_H = (H1 + H2 + H3 + H4)/4;
rm_S = (S1 + S2 + S3 + S4)/4;
rm_D = (D1 + D2 + D3 + D4)/4;
rm_K = (K1 + K2 + K3 + K4)/4;

figure(2),
subplot(5,1,1),  pcolor(flipud(rm_A)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,1,2),  pcolor(flipud(rm_H)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,1,3),  pcolor(flipud(rm_S)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,1,4),  pcolor(flipud(rm_D)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
subplot(5,1,5),  pcolor(flipud(rm_K)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])

%% Matriz de dados

DD = [
reshape(A1,1,120*80)
reshape(A2,1,120*80)
reshape(A3,1,120*80)
reshape(A4,1,120*80)
reshape(H1,1,120*80)
reshape(H2,1,120*80)
reshape(H3,1,120*80)
reshape(H4,1,120*80)
reshape(S1,1,120*80)
reshape(S2,1,120*80)
reshape(S3,1,120*80)
reshape(S4,1,120*80)
reshape(D1,1,120*80)
reshape(D2,1,120*80)
reshape(D3,1,120*80)
reshape(D4,1,120*80)
reshape(K1,1,120*80)
reshape(K2,1,120*80)
reshape(K3,1,120*80)
```

```
reshape(K4,1,120*80)
];

A = (DD.')*DD;
[V,D] = eigs(A,20,'lm') ;

% primeira figura é o rosto médio de todas as imagens
figure(3)
subplot(5,4,1), facel=reshape(V(:,1),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,2), facel=reshape(V(:,2),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,3), facel=reshape(V(:,3),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,4), facel=reshape(V(:,4),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,5), facel=reshape(V(:,5),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,6), facel=reshape(V(:,6),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,7), facel=reshape(V(:,7),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,8), facel=reshape(V(:,8),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,9), facel=reshape(V(:,9),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,10), facel=reshape(V(:,10),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,11), facel=reshape(V(:,11),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,12), facel=reshape(V(:,12),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,13), facel=reshape(V(:,13),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,14), facel=reshape(V(:,14),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,15), facel=reshape(V(:,15),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,16), facel=reshape(V(:,16),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,17), facel=reshape(V(:,17),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,18), facel=reshape(V(:,18),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,19), facel=reshape(V(:,19),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
subplot(5,4,20), facel=reshape(V(:,20),120,80); pcolor(flipud(facel)),
shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])

figure(7)
subplot(1,1,1), semilogy(diag(D),'ko','linewidth',[2])
set(gcf,'color','w')
set(gca,'Fontsize',[14])
xlabel('n')
ylabel('Valor Singular, \sigma_{n}')
title('Componentes Principais Ordenados')
vecA = reshape(rm_A,1,120*80);
vecH = reshape(rm_H,1,120*80);
vecS = reshape(rm_S,1,120*80);
vecD = reshape(rm_D,1,120*80);
vecK = reshape(rm_K,1,120*80);
```

```
projA = vecA*V;
projH = vecH*V;
projS = vecS*V;
projD = vecD*V;
projK = vecK*V;

% Projeção dos rostos médios nos autovetores
% O primeiro autovetor é retirado para tirar a influencia dos rostos
médios
% O resultado é uma chave para identificar novas imagens.
figure(4)
subplot(1,5,1), bar(projA(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
2000],'Xtick',[],'Ytick',[])
text(12,-1700,'Angelina','Fontsize',[15])
subplot(1,5,2), bar(projH(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
2000],'Xtick',[],'Ytick',[])
text(12,-1700,'Hitler','Fontsize',[15])
subplot(1,5,3), bar(projS(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
2000],'Xtick',[],'Ytick',[])
text(12,-1700,'Silva','Fontsize',[15])
subplot(1,5,4), bar(projD(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
2000],'Xtick',[],'Ytick',[])
text(12,-1700,'Adam','Fontsize',[15])
subplot(1,5,5), bar(projK(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
2000],'Xtick',[],'Ytick',[])
text(12,-1700,'Keanu','Fontsize',[15])

% adicionando uma nova imagem para teste
% projeção em cima dos componentes principais utilizados para gerar a
base
% original (primeiras 20 figuras)
% T = imagem teste (Keanu)

T = imresize(double(rgb2gray(imread('keanu_teste','jpeg'))),[120 80]);
vecT = reshape(T,1,120*80);
projT = vecT*V;

figure(5)
subplot(1,2,1), bar(projT(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
2000],'Xtick',[],'Ytick',[])
text(12,-1700,'Keanu Teste','Fontsize',[15])
subplot(1,2,2), bar(projK(2:20)), set(gca,'Xlim',[0 20],'Ylim',[-2000
2000],'Xtick',[],'Ytick',[])
text(12,-1700,'Keanu Médio','Fontsize',[15])

% distancia Euclidiana - A menor distancia indica o mais próximo
dist1 = sqrt(sum((projT-projA).^2));
dist2 = sqrt(sum((projT-projH).^2));
dist3 = sqrt(sum((projT-projS).^2));
dist4 = sqrt(sum((projT-projD).^2));
dist5 = sqrt(sum((projT-projK).^2));

figure(6)
dist =[dist1,dist2,dist3,dist4,dist5];
bar(dist)
xticklabels({'Angelina','Hitler','Silva','Adam','Keanu'})
xlabel('Faces Médias')
ylabel('Distância - Teste Keanu')
```

APÊNDICE C – Código principal de classificação de imagens por ACP (MAIN_CLASSIFICADOR_ACP.m)

```
%% Programa principal para classificação de imagens por ACP
% Vinicius Rodrigues Costa
% Para rodar basta selecionar os diretórios com
% as imagens de treino e imagens de teste.

clear all; close all; clc;

% Para testes rapidos basta comentar os campos de selecionar diretório e
% colocar o caminho para o diretório das imagens no mesmo formato a
baixo:
% dir_treino = uigetdir('C:\Users\T-
Gamer\Desktop\TCC\Códigos\classificador_acp\imagens_treino');
% dir_classif = uigetdir('C:\Users\T-
Gamer\Desktop\TCC\Códigos\classificador_acp\imagens_teste');
% Assim basta apertar select quando o programa rodar.


% Selecionar o diretório de imagens para treino
dir_treino = uigetdir('C:\','Diretório de imagens para treino');

% Selecionar o diretório de imagens para teste
dir_classif = uigetdir('C:\','Diretório de imagens para classificar');

% Carrega as imagens
[n_img_T,D_img_treino,lista_img_T] = FUNC_CARREGA_IMG(dir_treino);
[n_img_C,D_img_class,lista_img_C] = FUNC_CARREGA_IMG(dir_classif);

fprintf('A lista de Imagens utilizadas para Classificação: \n\n')
fprintf('%s \n',lista_img_T(:))

% Calcula a ACP e as projeções
[V,D,Proj_T,Proj_C] = FUNC_CALC_ACP(D_img_treino,D_img_class);

% Faz a classificação
[class] = FUNC_CLASS_ACP(Proj_T,lista_img_T,Proj_C,lista_img_C);

% Plota os graficos das imagens de treino
FUNC_GRAF_ACP(V,D,Proj_T,lista_img_T,D_img_treino)

% Gera o Arquivo final com os dados (resultados.xlsx)
FUNC_ARQUIVO_ACP(class)
```

APÊNDICE D – Código de função criada para carregar as imagens (FUNC_CARREGA_IMG.m)

```
% Função de carregar imagens em uma matriz
% Vinicius Rodrigues Costa
function [n_img,D_img,lista_img] = FUNC_CARREGA_IMG(diret)

diretorio = dir(strcat(diret,'\','*.jpg'));
n_img = 0;
lista_img = [];
D_img = [];

%Define gera uma lista de imagens
for iter1 = 1 : size(diretorio,1)
           lista_img = [lista_img; string(diretorio(iter1).name)];
           n_img = n_img + 1;
end

% Gera a matrix com as imagens
for iter2 = 1 : n_img
    str = strcat(diret,'\',lista_img(iter2));
    img  = reshape(imresize(double(rgb2gray(imread(str{1}))),[120
80]),1,120*80);
    D_img =[D_img;img];
end
end
```

APÊNDICE E – Código de função criada calcular as componentes principais (FUNC_CALC_ACP.m)

```
% Função para calcular as componentes principais nas imagens de treino
% Vinicius Rodrigues Costa
function [V,D,Proj_T,Proj_C] = FUNC_CALC_ACP(D_img_treino,D_img_class)

[n_img_1,~]= size(D_img_treino);
[n_img_2,~]= size(D_img_class);
Proj_T = [];
Proj_C = [];

%ACP
A = (D_img_treino.')*D_img_treino;
[V,D] = eigs(A,n_img_1,'lm');

% Calcula as projeções
%Projeção de Treino
for iter_1 = 1 : n_img_1
    Proj_img_1 = D_img_treino(iter_1,:)*V;
    Proj_T =[Proj_T;Proj_img_1];
end

%Projeção de imagens de teste (Para classificar)
for iter_2 = 1 : n_img_2
    Proj_img_2 = D_img_class(iter_2,:)*V;
    Proj_C =[Proj_C;Proj_img_2];
end
end
```

APÊNDICE F – Código de função criada para fazer a classificação das imagens (FUNC_CLASS_ACP.m)

```
% Função para classificar as imagens de teste
% Vinicius Rodrigues Costa
function [class] = FUNC_CLASS_ACP(Proj_T,lista_img_T,Proj_C,lista_img_C)

[n_T,~] = size(lista_img_T);
[n_C,~] = size(lista_img_C);
dist_num = [];
class = [];

% Interações de Classificação
for iter1 = 1 : n_C
    dist = [];

   for iter2 = 1: n_T
       dist = [dist, [ lista_img_T(iter2);...
       num2str(sqrt(sum((Proj_C(iter1,2:n_T)-
Proj_T(iter2,2:n_T)).^2)));...
       lista_img_C(iter1)]];
   end

   dist_num = [str2double(dist(2,:))];
   M = dist_num==min(dist_num);
   class = [class; dist(3,M),dist(2,M),dist(1,M)];
end

%remove caracteres especiais para criar lista de classe
tcl = regexprep(class(:,3),'.jpg','');
tcl = regexprep(tcl,'\d[0-9_]+\d','');
tcl = regexprep(tcl,'\d+(?:_(?=\d))?', '');

%Tablea com resultados
class = table(class(:,1),tcl,class(:,2),class(:,3),'VariableNames',...
       {'Imagem_analisada','Classe','Distancia','Imagem_mais_proxima'});

fprintf('\n\n Classificação das imagens: \n\n')
disp(class)
end
```

APÊNDICE G – Código de função criada para criar os gráficos de saída do programa principal (FUNC_GRAF_ACP.m)

```
% Função para criar graficos
% Vinicius Rodrigues Costa
function FUNC_GRAF_ACP(V,D,Proj_T,lista_img_T,D_img_treino)

% Graficos
[n_T,~] = size(lista_img_T);

%plotar imagens de treino junto a sua projeção
figure(1)
for iter1 = 1: n_T
    subplot(2,n_T,iter1),
pcolor(flipud(reshape(D_img_treino(iter1,:),120,80))),...
        shading interp, colormap(gray), set(gca,'Xtick',[],'Ytick',[])
    text(5,-4,string(lista_img_T{iter1}),'Fontsize',[15])
    subplot(2,n_T,iter1+n_T), bar(Proj_T(iter1,2:n_T)), set(gca,'Xlim',[0
n_T])
end

% A primeira imagem é o rosto médio das imagens de treino
figure(2)
for iter2 = 1: n_T
    subplot(1,n_T+1,iter2), facel=reshape(V(:,iter2),120,80);
pcolor(flipud(facel)), shading interp, colormap(gray),
set(gca,'Xtick',[],'Ytick',[])
end
subplot(1,n_T+1,n_T+1), semilogy(diag(D),'ko','linewidth',[2]),
set(gca,'Fontsize',[14])
end
```

APÊNDICE H – Código de função gerar o arquivo de saída da classificação (FUNC_ARQUIVO_ACP.m)

```
% Função para classificar as imagens de teste
% Vinicius Rodrigues Costa
function FUNC_ARQUIVO_ACP(class)

% Leitura da tabela de dados
tab_dados = readtable('dados_imagem.xlsx');

[Nc,~]= size(class{:,:});
[Nd1,Nd2]= size(tab_dados(:,:));

% Interações para fazer a união dos dados
T2 = [];
for iter1 = 1 : Nc
T1 = [];
    for iter2 = 1 : Nd1
        T0 = [];
        if class{iter1,2} == tab_dados{iter2,1}
            for iter3 = 1 : Nd2
                T0 = [T0, string(tab_dados{iter2,iter3})];
            end
            T1 = [class{iter1,1},class{iter1,3},class{iter1,4},T0];
        end
    end
    T2 = [T2;T1];
end

% Gera a primeira linha da tabela
T3 = ["Imagem_analisada","Distancia","Imagem_mais_proxima"];
for iter4 = 1 : Nd2
    T3 = [T3,string(tab_dados.Properties.VariableNames(iter4))];
end

% Gera a tabela final de resultados (resultados.xlsx)
T = [T3;T2];
writetable(table(T),'resultados.xlsx','WriteVariableNames',false)
fprintf('\n\n O Arquivo resultados.xlsx foi atualizado. \n\n')
end
```