



**UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

DEISE SANTANA MAIA

**DETECÇÃO E RECONHECIMENTO DE FACE UTILIZANDO O
MATLAB**

VITÓRIA DA CONQUISTA - BA

2014

DEISE SANTANA MAIA

**DETECÇÃO E RECONHECIMENTO DE FACE UTILIZANDO O
MATLAB**

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Estadual do Sudoeste da Bahia, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.
Orientador(a): Prof. Dr. Roque Mendes Prado Trindade

VITÓRIA DA CONQUISTA - BAHIA

2014

DEISE SANTANA MAIA

**DETECÇÃO E RECONHECIMENTO DE FACE UTILIZANDO O
MATLAB**

Aprovada em 08/08/2014

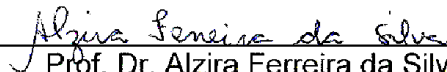
BANCA EXAMINADORA



Prof. Dr. Roque Mendes Prado Trindade
Universidade Estadual do Sudoeste da Bahia – UESB
Orientador



Prof. Msc. Alexsandra Oliveira Andrade
Universidade Estadual do Sudoeste da Bahia – UESB



Prof. Dr. Alzira Ferreira da Silva
Universidade Estadual do Sudoeste da Bahia – UESB

RESUMO

O objetivo deste trabalho foi analisar pesquisas publicadas sobre os temas detecção e reconhecimento de face e propor uma abordagem e implementação de um algoritmo para identificação de faces. Trata-se de um estudo do tipo bibliográfico exploratório com vista ao desenvolvimento experimental. Diferentes algoritmos de detecção de face, olhos e lábios foram comparados e posteriormente utilizados na delimitação e alinhamento das faces. Foram empregadas técnicas baseadas na distribuição de cores dos componentes da face com o auxílio das bibliotecas *Image Processing ToolBox* e *Neural Network ToolBox* do Matlab R2014a *Student Version*. No algoritmo de reconhecimento de face, as imagens resultantes da detecção de face foram classificadas utilizando a média de duas amostras de imagens de cada indivíduo do banco de dados. O percentual de acerto dos algoritmos de detecção face, localização de olhos, detecção de lábios e reconhecimento de face foi de 95,8%, 85%, 90% e 70,74%, respectivamente.

Palavras-chave: Processamento de imagens. Segmentação de pele. Detecção dos olhos. Detecção dos lábios. Reconhecimento de face.

ABSTRACT

The goal of this study was analyzing published studies about face detection and face recognition, and propose an approach and implementation of an algorithm for face identification. This is an exploratory, bibliographical and experimental study. Different algorithms for face, eyes and lips detection were compared and subsequently used for the delimitation and alignment of faces. The techniques applied were based on the color distribution of the face components using the Matlab R2014a *Student Version* libraries Image Processing Toolbox and Neural Network Toolbox. In the face recognition algorithm, the resulting images of the face detection were classified using the average of two sample images of each person of the dataset. The percentage of correct answers for the face detection, eyes localization, lips detection and face recognition algorithms were 95.8%, 85%, 90% and 70.74%, respectively.

Keywords: Image Processing. Skin Segmentation. Eye Detection. Lips Detection. Face recognition.

SUMÁRIO

1 INTRODUÇÃO	7
2 OBJETIVOS.....	10
2.1 OBJETIVO GERAL	10
2.2 OBJETIVOS ESPECÍFICOS	10
3 REFERENCIAL TEÓRICO.....	11
3.1 PROCESSAMENTO DE IMAGENS DIGITAIS	11
3.2 ESPAÇOS DE CORES.....	14
3.3 REDES NEURAIS ARTIFICIAIS	16
3.4 PROCESSAMENTO DE IMAGENS E REDES NEURAIS NO MATLAB.....	18
3.5 TÉCNICAS UTILIZADAS NA DETECÇÃO E RECONHECIMENTO DE FACE	19
4 METODOLOGIA.....	22
4.1 TIPO DE ESTUDO	22
4.2 FERRAMENTAS UTILIZADAS	22
5 DESENVOLVIMENTO DOS ALGORITMOS.....	24
5.1 DETECÇÃO DE FACE.....	24
<i>5.1.1 Localização da face</i>	<i>24</i>
5.1.1.1 Algoritmo de Localização de Face I	24
5.1.1.2 Algoritmo de Localização de Face II	25
5.1.1.3 Algoritmo de Localização de Face III.....	27
5.1.1.4 Algoritmo de Localização de Face IV	27
<i>5.1.2 Localização dos olhos</i>	<i>30</i>
5.1.2.1 Algoritmo de Localização dos Olhos I	30
5.1.2.2 Algoritmo de Localização dos Olhos II	31
5.1.2.3 Algoritmo de Localização dos Olhos III	33
<i>5.1.3 Alinhamento e delimitação da face</i>	<i>33</i>
<i>5.1.4 Localização dos lábios e delimitação da face</i>	<i>34</i>
5.2 RECONHECIMENTO DE FACE	35
5.2.1 Redimensionamento e alteração na iluminação	35
5.2.2 Descrição do algoritmo de reconhecimento de face.....	37
6.1 Resultados dos algoritmos de detecção de face	39
6.2 Resultados do Algoritmo de Reconhecimento de Face	41
7 CONCLUSÃO	42
8 TRABALHOS FUTUROS	43
9 REFERÊNCIAS.....	44
APÊNDICE A – TABELA DOS RESULTADOS DO ALGORITMO DE RECONHECIMENTO DE FACE	48
APÊNDICE B – ALGORITMOS DE LOCALIZAÇÃO DE FACE EM MATLAB	49
A) CÓDIGO DO ALGORITMO DE LOCALIZAÇÃO DE FACE I	49
B) CÓDIGO DO ALGORITMO DE LOCALIZAÇÃO DE FACE II	53

c) CÓDIGO DO ALGORITMO DE LOCALIZAÇÃO DE FACE III	55
d) CÓDIGO DO ALGORITMO DE LOCALIZAÇÃO DE FACE IV	57
APÊNDICE C – CÓDIGOS DO ALGORITMOS DE LOCALIZAÇÃO DE OLHOS EM MALAB	60
A) ALGORITMO DE LOCALIZAÇÃO DOS OLHOS I	60
B) ALGORITMO DE LOCALIZAÇÃO DOS OLHOS II	62
C) ALGORITMO DE LOCALIZAÇÃO DOS OLHOS III	64
APÊNDICE D – CÓDIGO DO ALGORITMO DE ALINHAMENTO DA FACE	66
APÊNDICE E – CÓDIGO DO ALGORITMO DE DETECÇÃO DE LÁBIOS	68
APÊNDICE F – CÓDIGO DO ALGORITMOS DE ALTERAÇÃO DA ILUMINAÇÃO	69
APÊNDICE G – CÓDIGO DO ALGORITMO RECONHECIMENTO DE FACE	70

1 INTRODUÇÃO

A área de detecção e reconhecimento automático de face é alvo de uma série de estudos e pesquisas. Entre as suas aplicações está a biometria, que é o estudo das características físicas e comportamentais individuais, o que também inclui o reconhecimento da íris ou das digitais. Porém, a vantagem da utilização da face como meio de identificação biométrica é que esse método é menos invasivo e, em geral, não são necessários equipamentos sofisticados para a obtenção da imagem, podendo-se utilizar câmeras digitais ou *webcams* (ZHAO, 2003).

São diversas as áreas que se beneficiam de sistemas de reconhecimento de face, incluindo entretenimento, como na interação humano-robô/computador e em vídeo games; segurança da informação, como em *login* de dispositivos pessoais, segurança na Internet e segurança em aeroportos; e a área de vigilância, como na análise de vídeos de câmeras de segurança (ZHAO, 2003).

Um sistema de reconhecimento de face pretende reproduzir uma das capacidades naturais dos seres humanos: reconhecer as características de uma face em diferentes ambientes e associá-las a informações já armazenadas na memória.

O que parece algo simples e automático no ser humano, representa um desafio em sistemas computacionais devido às diversas variáveis do ambiente: posição facial e distância com relação à(s) câmera(s), iluminação do ambiente e possíveis sombras projetadas sobre a face, e alguns aspectos individuais, como o uso de barba, variações de peso e cortes de cabelo, etc. Contudo, geralmente algumas características básicas são mantidas independentemente do ambiente e o reconhecimento é possível através da análise dos padrões encontrados em diferentes imagens de um mesmo indivíduo (GOTTUMUKKAL e ASARI, 2003; ZHAO, 2003).

Os primeiros estudos na área de reconhecimento de face foram realizados no campo da Psicologia, por volta de 1950 e, por volta de 1960, no campo da Engenharia. Mas, as primeiras pesquisas envolvendo o reconhecimento automático de faces através de máquinas se deu na década de 1970. Desde então, essa área de pesquisa tem sido conduzida por psicofísicos, neurocientistas, engenheiros e cientistas da computação (ZHAO, 2003).

A neurociência tem estudado o reconhecimento de face pelos seres humanos, as áreas do cérebro associadas ao reconhecimento e os fatores que nos fazem distinguir faces de pessoas diferentes (ZHAO, 2003).

Um estudo comparativo foi realizado para analisar a diferença entre o reconhecimento de faces e objetos por seres humanos, e o resultado dos testes mostrou que alguns parâmetros, como a fonte de iluminação e distorções na imagem, afetam de forma diferente o reconhecimento das faces e objetos. Em geral, o reconhecimento de faces é mais sensível às variações de direção da iluminação, rotação em profundidade, rotação no plano, polaridade de contraste e métrica de variação (proporção ou grau de curvatura). Essas mesmas variáveis devem ser consideradas no estudo de reconhecimento automático de faces (BIEDERMAN, 1998).

As primeiras pesquisas no campo da Engenharia e Computação, na década de 1970, trataram o reconhecimento de face como um problema de reconhecimento de padrões em 2D e utilizaram técnicas típicas de reconhecimento de padrões. Mas, foi apenas na década de 1990 que os estudos nessa área se desenvolveram significativamente, devido ao crescente interesse comercial, à disponibilidade de *hardware* compatível com as necessidades desses sistemas e à sua utilização em sistemas de vigilância (ZHAO, 2003).

Como em qualquer outro sistema computacional, o objetivo de desenvolver sistemas para reconhecer faces é agilizar uma tarefa que poderia ser desenvolvida por pessoas, com a vantagem de que o desempenho de uma máquina pode ser superior ao desempenho humano na realização de tarefas repetitivas e que exigem atenção. Nos seres humanos, o reconhecimento de face é muito mais complexo e apurado do que qualquer sistema automático, pois o contexto do ambiente e o aprendizado adquirido durante a vida é levado em consideração. Entretanto, em sistemas de vigilância, por exemplo, a atenção de um indivíduo na tentativa de identificar faces pode diminuir com o tempo, devido à fadiga e elementos de distração do meio. O mesmo não acontece em sistemas automáticos, que têm uma capacidade superior à dos seres humanos para armazenar faces na memória e depois recuperá-las, ainda que as características armazenadas da face sejam limitadas.

Nos próximos capítulos deste trabalho serão abordados os objetivos desta pesquisa, o referencial teórico dos conteúdos estudados, a metodologia e as

técnicas utilizadas na implementação dos algoritmos e a análise dos resultados. O Capítulo 2 (Objetivos) abordará os objetivos gerais e específicos da presente pesquisa. O Capítulo 3 (Referencial teórico) foi dividido em sessões que abordam os tópicos referentes ao processamento de imagens digitais, espaços de cores, redes neurais artificiais, Matlab, e técnicas utilizadas na detecção de reconhecimento de faces. O Capítulo 4 (Metodologia) abordará o tipo de pesquisa e as ferramentas utilizadas para desenvolver a mesma. O Capítulo 5 (Desenvolvimento dos Algoritmos) irá contemplar a descrição dos algoritmos implementados para a detecção e reconhecimento de face. O Capítulo 6 (Resultados e Discussão) irá apresentar e discutir os resultados dos algoritmos implementados. Por fim, a conclusão e os trabalhos futuros serão apresentados nos Capítulos 7 e 8, respectivamente.

2 OBJETIVOS

2.1 Objetivo Geral

Analisar pesquisas publicadas sobre o tema e propor uma abordagem e implementação de algoritmos para detecção e reconhecimento de faces.

2.2 Objetivos Específicos

- Investigar as técnicas mais utilizadas na detecção e reconhecimento de faces;
- Modelar algoritmos para detecção e reconhecimento de faces;
- Definir as pré-condições dos algoritmos;
- Definir o banco de dados de imagens a ser utilizado nos testes;
- Implementar o algoritmo em Matlab;
- Fazer testes e análises estatísticas dos resultados.

3 REFERENCIAL TEÓRICO

3.1 Processamento de Imagens Digitais

Imagens digitais podem ser definidas por uma função discreta bidimensional $f(x,y)$, na qual x e y representam as coordenadas espaciais com valores inteiros e o valor de f corresponde à intensidade da imagem na coordenada de entrada, e pelo seu número de linhas e colunas (GONZALES e WOODS, 2002).

Os componentes ou elementos de uma imagem digital são conhecidos como *pixels*, que são definidos de formas diferentes a depender do tipo de imagem. Nas imagens em escala de cinza, os valores dos *pixels* representam seus níveis de cinza, que usualmente se encontram no intervalo $[0, 2^k - 1]$, sendo k um valor inteiro (GONZALES e WOODS, 2002). Nas imagens coloridas, os valores dos *pixels* são definidos como tuplas ternárias, sendo que os seus valores variam com o espaço de cores utilizado. A Figura 1 ilustra a definição de um *pixel* em imagens em escala de cinza e coloridas com valores no intervalo $[0, 255]$.

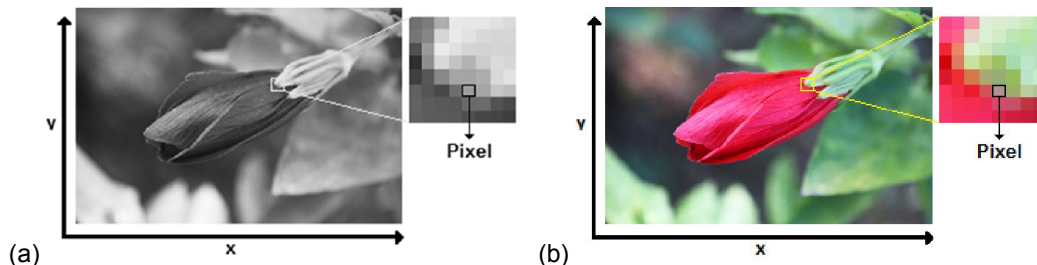


Figura 1: (a) Definição de um pixel em uma imagem em escala de cinza, na qual o valor de $f(x_i, y_i) = 158$, sendo (x_i, y_i) a posição do pixel em destaque; (b) Definição de um pixel em uma imagem colorida, na qual o valor de $f(x_i, y_i) = (176, 148, 147)$ no espaço de cores RGB, sendo (x_i, y_i) a posição do pixel em destaque. Fonte: Elaborada pelo autor.

Para descrever o conceito de processamento de imagens, Gonzales e Woods (2002) consideraram uma escala com três níveis de processamento (baixo, médio e alto). Os processos de baixo nível são caracterizados pelo fato de que suas entradas e as saídas são imagens, como as operações de redução de ruído e realce do contraste. O nível médio inclui operações que têm como resultado atributos extraídos das entradas, como a segmentação de imagens. Por sua vez, os processos de alto nível incluem a análise de imagens, que extrai a semântica da

entrada para executar funções cognitivas, como, por exemplo, a detecção de faces e objetos em imagens digitais.

A seguir serão descritos os processos de redimensionamento, dilatação e segmentação de imagens, que foram utilizados no desenvolvimento deste trabalho.

Redimensionar uma imagem significa alterar suas dimensões, reduzindo ou aumentando o seu número de linhas e colunas, tendo como resultado uma imagem semelhante à original. Quando se reduz as dimensões de uma imagem, é necessário eliminar algumas linhas ou colunas, provocando a perda de informações da imagem. Por isso, se tentarmos reverter esse processo, a imagem resultante não será igual à imagem original. Porém, existem algumas funções de suavização que minimizam o efeito de bordas irregulares resultantes do aumento das dimensões de uma imagem (GONZALES e WOODS, 2002). A Figura 2 ilustra o resultado do redimensionamento de uma imagem com e sem a utilização de filtros de suavização.

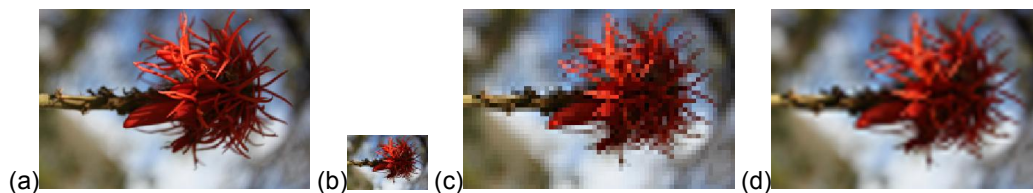


Figura 2: (a) imagem original (250x166 *pixels*); (b) imagem redimensionada com 25% do número de linhas e colunas da imagem original (63x42 *pixels*); (c) imagem redimensionada com suas dimensões originais sem filtro; (d) imagem redimensionada com suas dimensões originais com filtro. Fonte: Elaborada pelo autor.

A dilatação faz parte do conjunto de operadores morfológicos, que se baseiam na “extração da geometria e topologia de uma imagem através do uso de outra imagem completamente definida, chamada elemento estruturante” (ANDRADE *et al.*, 2013, p. 1). No contexto de processamento de imagens, a dilatação é geralmente utilizada para explorar e ampliar as formas contidas na imagem de entrada ou reparar possíveis descontinuidades (GOYAL, 2011). A Figura 3 apresenta os resultados da dilatação realizada no Matlab de uma imagem binária utilizando dois elementos estruturantes diferentes, ambos quadrados e brancos, sendo que o primeiro tem dimensões 8x8 *pixels* e, o segundo, 12x12 *pixels*. Podemos observar que o nível de alteração da imagem resultante depende da forma e das dimensões do elemento estruturante utilizado.

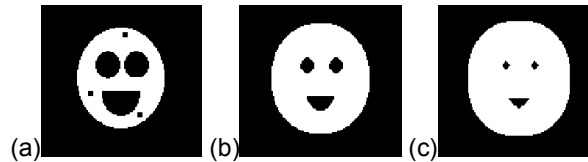


Figura 3: (a) Imagem original; (b) Imagem resultante da dilatação utilizando um elemento estruturante quadrado e branco de dimensões 8x8 *pixels*; (c) Imagem resultante da dilatação utilizando um elemento estruturante quadrado e branco de dimensões 12x12 *pixels*. Fonte: Elaborada pelo autor.

Segundo Luccheseyz e Mitray (2001), a segmentação é um processo que subdivide uma imagem em regiões, que devem ser homogêneas com relação à algumas características, como cor ou textura. A semântica das regiões segmentadas e o método utilizado na segmentação dependem do tipo de aplicação. Os autores descrevem ainda diferentes técnicas utilizadas no processo de segmentação de imagens: *clustering*, limiarização de histograma, *split-and-merge*, *region growing*, detecção de bordas e redes neurais.

O *clustering* consiste em uma classificação não supervisionada de objetos, na qual os padrões de classificação são pré-definidos e independentes da imagem de entrada (LUCCHESEYZ e MITRAY, 2001).

A técnica de limiarização de histograma baseia-se nos picos e vales do histograma de uma imagem para separar os seus objetos do plano de fundo. Em imagens monocromáticas, os picos e vales podem ser facilmente identificados a partir do histograma de luminosidade da imagem. No caso de imagens coloridas, é necessária a análise dos histogramas das três componentes de cores da imagem (LUCCHESEYZ e MITRAY, 2001).

Na técnica *split-and-merge*, a imagem de entrada é subdividida até que sejam obtidas partições homogêneas. Subsequente à etapa de particionamento, os fragmentos resultantes são agrupados às regiões vizinhas que apresentam características semelhantes, segundo alguns critérios pré-definidos. O agrupamento é executado até que sejam encontradas as maiores regiões homogêneas da imagem (LUCCHESEYZ e MITRAY, 2001).

A técnica *region growing* consiste em utilizar um fragmento pré-selecionado da imagem, e aglomerar *pixels* vizinhos à região inicial obedecendo a um critério de homogeneidade até que não haja mais *pixels* a serem adicionados à região. Essa técnica visa a segmentação de imagens com uma única região, mas, combinada a outras técnicas, pode ser aplicada a imagens com múltiplas regiões (LUCCHESEYZ e MITRAY, 2001).

Por fim as técnicas de detecção de bordas e redes neurais utilizam as fronteiras entre as regiões da imagem e classificação de *pixels*, respectivamente, para processar a segmentação (LUCHESEY e MITRAY, 2001).

A Figura 4 mostra o resultado da segmentação de duas imagens utilizando o Matlab utilizando as técnicas de limiarização de histograma e *clustering*, respectivamente.

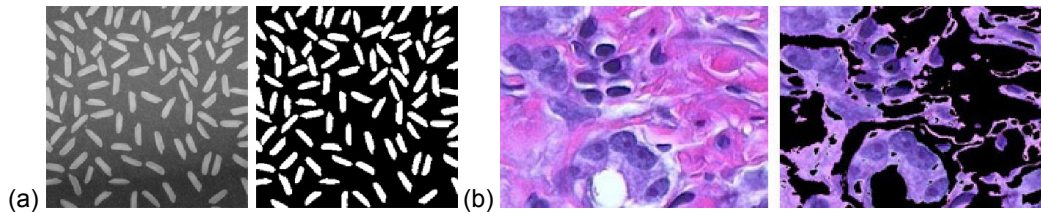


Figura 4: (a) imagem original de grãos de arroz e sua segmentação utilizando um método de limiarização; (b) imagem original de um tecido corado e sua segmentação utilizando a técnica *clustering*. Fonte: THE MATHWORKS, [s.d.].

3.2 Espaços de Cores

As cores são percebidas pelo aparelho visual humano através de dois tipos de estruturas localizadas na retina: os cones e os bastonetes. Os bastonetes atuam principalmente quando há pouca luminosidade, pois os cones se tornam quase insensíveis à luz nessas condições. Porém, a níveis normais de luminosidade, os cones provêm a maior parte da informação visual ao cérebro (MUSTAFI, ENGEL e PALCZEWSK, 2009).

Os cones subdividem-se em três grupos, de acordo com a sensibilidade aos diferentes intervalos de comprimento de onda, os quais encontram-se na área das cores primárias azul, verde ou vermelha do espectro. Portanto, a percepção das inúmeras cores pelo cérebro é resultado da combinação de três comprimentos de onda em proporções variadas (MUSTAFI, ENGEL e PALCZEWSK, 2009).

Nesse contexto, diversos espaços de cores foram criados para representar imagens digitais, incluindo o RGB e o YCbCr, que serão utilizados durante o desenvolvimento deste trabalho.

Um dos espaços mais conhecidos é o RGB (*Red, Green and Blue*), no qual os *pixels* são representados pela intensidade dos componentes Vermelho, Verde e Azul. Assim, os valores dos *pixels* se limitam à estrutura tridimensional ilustrada na Figura 5, na qual cada dimensão representa um componente, sendo a cor branca

representada pelos valores máximos dos três componentes e a cor preta pelos valores mínimos.

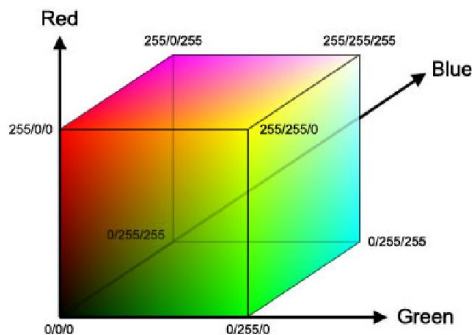


Figura 5: Cubo do espaço de cores RGB. Fonte: ZAPATA, 2013.

A Figura 6 mostra uma imagem e a representação de suas matrizes R, G e B. As cores das matrizes representam a intensidade das cores vermelho, verde e azul em cada imagem. Assim, as regiões mais escuras em R, G e B são as que apresentam os *pixels* com os menores valores para as componentes vermelho, verde e azul, respectivamente. Nota-se que, as regiões da imagem com a cor mais próxima ao branco apresentam valores altos nas três matrizes.

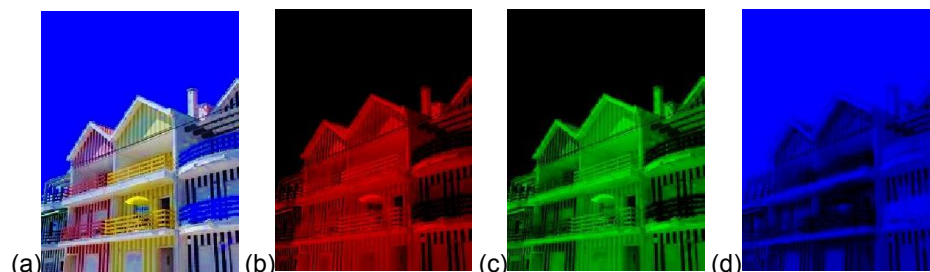


Figura 6: (a) imagem original; (b) matriz R; (c) matriz G; (c) matriz B.

Fonte: Elaborada pelo autor

O espaço de cores YCbCr não é intuitivo como o RGB mas é útil em aplicações de compressão de imagem e transmissão de sinais de vídeo, dividindo-se nas componentes de luminância e croma. A luminância (componente Y) mede a densidade de intensidade da luz de um ponto na imagem, e a croma (componentes Cr e Cb) medem o valor das cores. O espaço YCbCr pode ser representado pela Figura 7, no qual os eixos x e y são a croma e, o eixo z é a luminância.

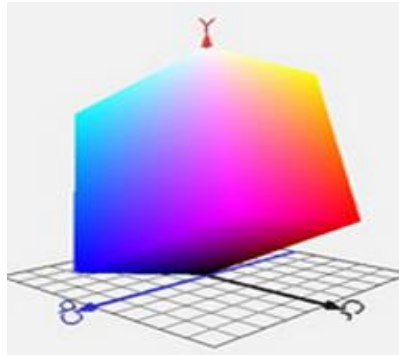


Figura 7: Gráfico do Espaço YCbCr. Fonte: BRIGGS, 2012.

A Figura 8 mostra as três componentes de cores de uma imagem no espaço YCbCr. Ressalta-se que, a componente Y é basicamente a imagem original em escala de cinza.

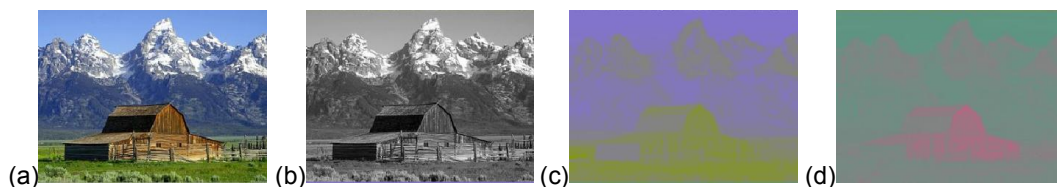


Figura 8: (a) imagem original; (b) matriz Y; (c) matriz Cb; (d) matriz Cr.
Fonte: WIKIPEDIA, 2014.

A seguinte equação converte os valores RGB em YCrCb (ASMARE, ASIRVADAM e IZNITA, 2009):

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & 0.081 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Ambos os espaços de cores mostrados anteriormente foram utilizados na implementação dos algoritmos de detecção de reconhecimento de face, ressaltando-se que, em cada situação, um dos espaços ou a combinação dos dois se mostrou mais apropriado

3.3 Redes Neurais Artificiais

Redes neurais artificiais são processadores capazes de aprender através da experiência e, então, utilizar o conhecimento adquirido em situações novas no mesmo escopo de sua aprendizagem. As redes neurais foram criadas com base no modelo de aprendizado humano, que inclui os neurônios e as suas transmissões

sinápticas, bem como as propriedades de plasticidade e adaptabilidade (HAYKIN, 2001).

O modelo de uma rede neural artificial é composto por neurônios (unidades de processamento), pesos sinápticos (fatores de multiplicação dos sinais de entrada), somadores para reunir os sinais de entrada e as funções de ativação, que restringem a amplitude do sinal de saída (HAYKIN, 2001).

A estrutura dos neurônios na rede está relacionada ao algoritmo de aprendizagem utilizado para o treinamento. O modelo que foi utilizado neste trabalho é do tipo alimentado adiante (*Feedforward*) ou acíclico, composto por camadas ocultas, nas quais os neurônios têm como entrada apenas os sinais da camada precedente. Os neurônios ocultos têm a função de extrair estatísticas de ordem elevada e, conseqüentemente, intervir entre a entrada externa e a saída da rede (HAYKIN, 2001). A Figura 9 apresenta um modelo de uma Rede Neural *Feedforward*.

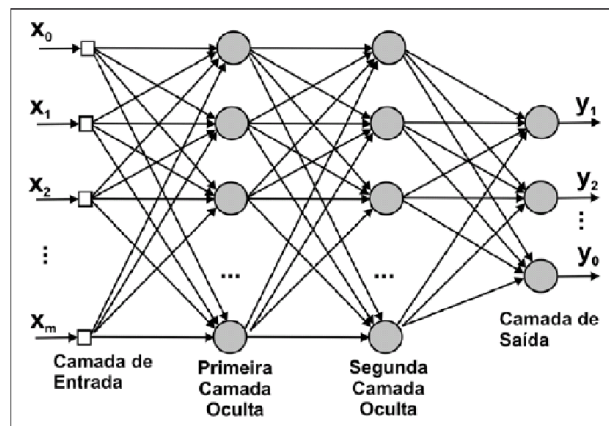


Figura 9: Modelo de uma Rede Neural *Feedforward* com duas camadas ocultas.
Fonte: BARBOSA, FREITAS e NEVES, 2005.

Entre as aplicações das redes neurais está o reconhecimento de padrões, que é realizado através do treinamento de uma rede utilizando entradas que representam o padrão a ser identificado. O número de entradas depende dos parâmetros da função que a rede pretende simular. Em sistemas de segmentação de pele no espaço RGB, que será abordado mais adiante, a rede pode ser implementada com três neurônios na camada de entrada, que representam as componentes R, G e B de cada *pixel*. A saída seria a probabilidade de um *pixel* pertencer ou não à pele.

3.4 Processamento de Imagens e Redes Neurais no Matlab

O Matlab é uma ferramenta com uma linguagem de alto nível e um ambiente interativo voltada para a computação numérica. O software trabalha com matrizes e inclui bibliotecas de funções relacionadas ao processamento de imagens e redes neurais (THE MATHWORKS, [s.d.]).

Os arquivos de imagem podem ser manipulados através do *Image Processing Toolbox*, que inclui as funções *imread* e *imwrite*, utilizadas para abrir e salvar arquivos de imagem, respectivamente.

A função *imread* permite diferentes sintaxes, mas, a mais utilizada neste trabalho foi $A = \text{imread}(\text{filename}, \text{fmt})$, na qual A recebe a matriz de cores correspondente à imagem *filename.fmt*. A matriz de cores tem o mesmo número de linhas e colunas da imagem de entrada e o valor de cada posição na mesma representa a cor de um *pixel*. Para imagens binárias, os valores da matriz de cores são restritos a 0 e 1, que representam as cores preta e branca, respectivamente. Para imagens coloridas, a matriz apresenta três dimensões relativas ao espaço RGB, com valores inteiros no intervalo [0,255] (THE MATHWORKS, [s.d.]).

A função *imwrite* também permite diferentes sintaxes, entre elas destaca-se a *imwrite(A,filename)*, na qual a matriz de cores A é armazenada no arquivo de imagem *filename*.

As matrizes de cores podem ser manipuladas utilizando operações matriciais básicas, como soma, subtração e multiplicação, ou funções específicas do processamento de imagens, como erosão, dilatação e alteração de contraste (THE MATHWORKS, [s.d.]).

As Redes Neurais são criadas e manipuladas através do *Neural Network Toolbox*, que inclui diversas funções de aprendizado como a *feedforwardnet* e *cascadeforwardnet*, utilizadas para criar redes neurais do tipo *FeedForward* e *Cascade-Forward*, respectivamente.

As redes criadas no Matlab podem ser configuradas quanto ao número de camadas escondidas, tipo de função e número de épocas do treinamento, erro final de aprendizado, taxa de treinamento e formato de saída. O treinamento das redes com aprendizado supervisionado é feito utilizando a função *train* com a seguinte sintaxe $\text{net} = \text{train}(\text{net}, P, T)$, na qual *net* é a rede neural a ser treinada a partir da matriz de entrada P e valores de saída esperados em T (THE MATHWORKS, [s.d.]).

3.5 Técnicas utilizadas na detecção e reconhecimento de face

Um sistema de reconhecimento de face automático tenta responder à alguma (s) das seguintes questões: de quem são as faces que aparecem na imagem? A imagem de entrada contém a face esperada pelo sistema? As faces detectadas na imagem são novas ou já são “conhecidas” pelo sistema?

A entrada de um sistema de reconhecimento automático de face é uma imagem de uma ou mais faces, e, a saída, sua identificação/verificação a partir de um banco de dados. A imagem não é composta unicamente pela face, mas também por um plano de fundo. Primeiramente, deve-se determinar o padrão das imagens e o procedimento necessário para a padronização das imagens de entrada como, por exemplo, resolução e escala de cores. Em seguida, deve-se definir os parâmetros a serem utilizados no reconhecimento das faces e, assim, classificar as imagens do banco de dados segundo esses parâmetros. Por último, a imagem de entrada é identificada ou verificada. Na identificação, a face é pesquisada em um banco de dados e o resultado é a identidade correspondente ao indivíduo na foto (caso o mesmo esteja no banco de dados). Enquanto isso, na verificação, o sistema deve responder se a face de entrada pertence a um determinado indivíduo ou não (ZHAO, 2003).

A depender do tipo de aplicação, a eficiência do sistema terá diferentes parâmetros. Em um sistema que analisa vídeos de crimes capturados por câmeras de vigilância, por exemplo, o tempo de execução do sistema não é crucial pois, assim como as investigações policiais, a precisão do resultado é mais importante do que se ter um sistema em tempo real. Por outro lado, o resultado em tempo real é essencial em sistemas de *logon*, pois há um tempo máximo ideal de espera pelo usuário. Dessa forma, os diferentes parâmetros e requisitos de cada sistema geram restrições que irão definir a sua forma de implementação.

Para implementar esses sistemas faz-se necessário decidir sobre: resolução e tipo da(s) câmera(s) utilizada(s); número de câmeras; capacidade de processamento dos computadores utilizados; tipo da imagem (2D ou 3D, infravermelho, a cores ou em escala de cinza, etc.); uso de imagens estáticas ou vídeos; uso de outros recursos em paralelo, como a detecção de voz; e se será um sistema de verificação ou identificação. Todas essas decisões interferem no custo dos sistemas e nas técnicas e algoritmos a serem utilizados (ZHAO, 2003).

Diversas técnicas de detecção e reconhecimento de faces foram descritas nos artigos revisados. Dentre estas, a detecção de pele foi a mais citada como forma de delimitar a área da face em uma imagem colorida. Singh (2003) descreve três algoritmos de detecção de pele, sendo que em cada algoritmo um diferente espaço de cores é utilizado. O resultado final é a interseção dos resultados para os espaços RGB, YCbCr e HSI (*hue, saturation and intensity*), seguido da extração dos olhos e lábios, que devem ser as áreas mais escuras dentro da área da face. Sidib, Montesinos e Janaqui (2006) descrevem um algoritmo de detecção de pele no espaço RGB normalizado e utiliza uma função elíptica Gaussiana de densidade de probabilidade para modelar a distribuição de cores da pele. Depois, os olhos são detectados como duas áreas dentro da área da face que não pertençam à faixa de cores da pele humana e que obedeçam alguns padrões quanto às suas localizações e dimensões. Freitas *et al.* (s.d.) utilizaram o espaço RGB no algoritmo de detecção de pele, que define intervalos de possíveis valores de vermelho, azul e verde para os diferentes tons de pele. Em contrapartida, Lakshmi e Patilkulakarni (2010), além da detecção de pele nos espaços de cores YCbCr e HSI, utilizaram dois algoritmos de detecção de borda de forma a melhorar sua eficácia quando a cor do plano de fundo pertence à faixa de cores da pele humana.

Ito *et al.* (2012) descreve uma técnica para detecção de olhos a partir de uma transformada de Hough bidimensional, que detecta possíveis centros dos olhos, e depois extrai pares que obedeçam a determinadas condições. A partir da localização dos olhos é possível derivar alguns pontos característicos do rosto e extrair sua área da imagem (CAMPADDELLI, LANZAROTTI e LIPORI, 2007). A ponta do nariz pode ser caracterizada pelo ponto com maior luminosidade do rosto e, os lábios são detectados a partir do extremo das suas laterais e dos seus extremos inferiores e superiores.

O passo seguinte à detecção é o treinamento da base de dados de faces. A técnica PCA (*Principal Component Analysis*) é amplamente utilizada em algoritmos de reconhecimento de padrões e se baseia no cálculo de vetores e valores próprios a partir de um banco de dados de imagens. No PCA, as imagens do banco de dados são representadas como vetores, através dos quais se calcula uma base vetorial com o menor número de vetores possíveis. Nesse sentido, cada imagem poderá ser representada com uma combinação linear da base encontrada (GOTTUMUKKAL e ASARI, 2003). Essa técnica é genérica e pode ser utilizada de diferentes formas

combinada a outras técnicas. Gottumukkal e Asari (2004) subdividiram as imagens e aplicaram o PCA à cada subimagem, o que se mostrou mais eficiente a depender do número de subimagens. A desvantagem dessa técnica é que a variação de iluminação, posição e expressão facial em cada imagem podem influenciar sensivelmente os seus resultados. Para contornar esse problema, pode-se combinar o PCA a um algoritmo de alinhamento de face, que é aplicado antes do PCA e se baseia em pontos chaves da imagem para padronizá-las em uma mesma posição canônica (GENG e JIANG, 2012).

Além do PCA, sistemas de reconhecimento de face podem utilizar técnicas que analisem a semântica da imagem, ou seja, que detectem componentes das faces (olhos, nariz, etc.), podendo ser mais precisos no reconhecimento de face, com a desvantagem de ter um maior custo computacional. Um estudo comparando as técnicas descritas mostrou que o reconhecimento baseado em componentes e na utilização de SMV (*Support Vector Machine*) pode superar um sistema que utiliza técnicas globais, como o PCA, para a classificação das imagens (HEISELE, 2003).

Atingir um equilíbrio entre performance e custo total deve ser o objetivo de um sistema de reconhecimento de faces, uma vez que poderá ser utilizado em ambientes que exigem um tempo real de resposta e um alto grau de confiabilidade.

4 METODOLOGIA

4.1 Tipo de estudo

O presente trabalho caracteriza-se como um estudo do tipo bibliográfico exploratório com vista ao desenvolvimento experimental, ou seja, a implementação de algoritmos tomando como base artigos e outras produções científicas sobre o tema (GIL, 2010).

Na maioria dos trabalhos acadêmicos, a pesquisa bibliográfica se faz necessária para fornecer uma fundamentação teórica ao trabalho e o estado da arte do tema proposto. Esse tipo de metodologia confere a possibilidade de investigar de forma mais ampla uma série de fenômenos que não estão ao alcance do pesquisador (GIL, 2010).

O desenvolvimento experimental é definido como:

Um trabalho sistemático, que utiliza conhecimentos derivados da pesquisa ou experiência prática com vistas à produção de novos materiais, equipamentos, políticas e comportamentos, ou à instalação ou melhoria de novos sistemas e serviços (GIL, 2010, p. 27).

4.2 Ferramentas utilizadas

Neste trabalho, o desenvolvimento experimental foi realizado utilizando o ambiente interactivo Matlab R2014a *Student Version*, o banco de dados faciais *Helen*, contendo imagens de faces em diferentes ângulos, resoluções e planos de fundo e, o banco de dados *Faces 1999*, que contém faces frontais de 28 indivíduos sob diferentes níveis de iluminação, plano de fundo e expressões faciais (CALTECH, 1999; LE, [s.d.]).

Foram selecionadas 93 imagens do banco de dados *Helen* e 447 imagens do banco de dados *Faces 1999* no formato JPEG, totalizando 540 imagens, das quais 500 imagens foram utilizadas nos testes dos algoritmos de detecção de face e 40 imagens foram utilizadas na implementação de um dos algoritmos de localização da face utilizando redes neurais. Para testar o algoritmo de reconhecimento de face, apenas o banco de dados *Faces 1999* foi utilizado, pois o mesmo, diferentemente do *Helen*, contém várias imagens de um mesmo indivíduo. A maioria das imagens selecionadas é colorida e contém apenas uma face em posição frontal, considerando que esses são os pré-requisitos do sistema. Para minimizar o tempo

de resposta dos algoritmos, as imagens foram redimensionadas sem alterar suas proporções. As dimensões das imagens do banco de dados *Faces 1999* foram reduzidas em 50%, resultando em imagens de 448x196 *pixels*, enquanto que as imagens do *Helen*, que possuem dimensões variadas, resultou em imagens de dimensões entre 200x210 e 450x449 *pixels*.

Ressalta-se que os banco de dados *Faces 1999* e *Helen* foram escolhidos por obedecerem ao pré-requisito do sistema e por terem sido empregado em outros artigos com abordagem semelhante.

A implementação das técnicas de detecção de face se deu por etapas, subdividindo-se em técnicas de localização de face, detecção de olhos, detecção de lábios e alinhamento da face. Foram implementados mais de um algoritmo para localização de faces e detecção de olhos de modo a encontrar o mais adequado para as imagens testadas.

Para o reconhecimento de face, foi implementada uma técnica global de classificação e os testes foram realizados utilizando as imagens do banco de dados *Faces 1999*.

As técnicas utilizadas na detecção e reconhecimento de face serão descritas no capítulo seguinte.

5 DESENVOLVIMENTO DOS ALGORITMOS

5.1 Detecção de face

Na implementação, a detecção de face foi subdividida em localização da face, localização dos seus componentes (olhos e lábios) e alinhamento da face. As próximas seções irão detalhar a(s) técnica(s) utilizada(s) em cada etapa.

5.1.1 Localização da face

Nessa etapa, quatro algoritmos de localização de face foram implementados e comparados, utilizando diferentes abordagens para a segmentação de pele e delimitação da área da face nos espaços de cores RGB e YCbCr. O código desses algoritmos em Matlab encontra-se no Apêndice B.

5.1.1.1 Algoritmo de Localização de Face I

No primeiro algoritmo implementado, empregou-se um método simples na segmentação de pele, que consiste em analisar os valores de R, G e B de cada *pixel* a partir do seguinte classificador (NASCIMENTO *et al*, 2007):

Dados os valores do pixel p, R, G e B pertencentes ao intervalo [0, 255], se $R > 95$ e $G > 40$ e $B > 20$ e $Max \{R, G, B\} - Min \{R, G, B\} > 15$ e $|R - G| > 15$ e $R > G$ e $R > B$, então p é um pixel pertencente à pele.

O classificador descrito acima foi utilizado gerando uma imagem binária para cada imagem no banco de dados, nas quais os *pixels* brancos representam os pontos classificados como pele. Após a segmentação, foi realizada uma dilatação com um elemento estruturante quadrado e branco de dimensões 8x8 em cada imagem resultante com o objetivo de apagar os pontos no interior da face que não foram classificados como pertencentes à pele. A partir da imagem resultante, a posição da face foi estimada através dos maiores segmentos de *pixels* brancos contínuos verticais e horizontais. A área da face foi definida da seguinte forma: *dados os pontos (x_1, y_1) e (x_2, y_2) , que representam o início da maior linha e maior coluna, respectivamente, e os valores v_1 e v_2 , que, por sua vez, representam a*

largura da linha e a altura da coluna encontrada, a área da face é definida pelo retângulo com os seguintes vértices: (x_2, y_1) , (x_2, y_1+v_1) , (x_2+v_2, y_1) e (x_2+v_2, y_1+v_1) .

A entrada do Algoritmo de Localização de Face I é o nome da imagem de entrada e o mesmo retorna uma imagem binária resultante da segmentação de pele e uma imagem colorida com a face da imagem de entrada delimitada. Alguns resultados podem ser vistos na Figura 10.

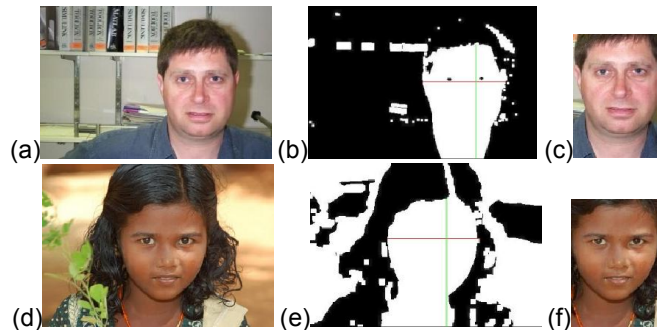


Figura 10: (a) e (d): imagens originais; (b) e (e): imagens binárias após a segmentação, dilatação e localização da maior linha (vermelho) e maior coluna (verde); (c) e (f) imagens resultantes do Algoritmo de Localização de Face I. Fonte: CALTECH, 1999; LE, [s.d.].

Com a utilização desse método, encontrou-se um problema relacionado à sua sensibilidade às variações de luz da imagem, o que gera muitos falsos negativos em imagens com pouca iluminação, como mostrado na Figura 11.

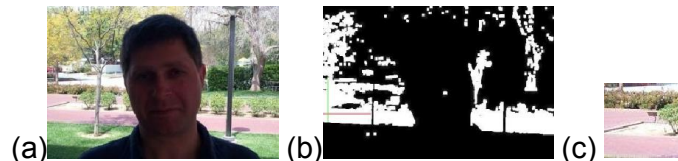


Figura 11: (a): imagem original; (b): imagem binária após a segmentação, dilatação e localização da maior linha (vermelho) e maior coluna (verde); (c) imagem resultante da localização incorreta de face. Fonte: CALTECH, 1999.

5.1.1.2 Algoritmo de Localização de Face II

Para contornar o problema das variações de luminosidade nas imagens, foi implementada uma variação do Algoritmo de Localização de Face I utilizando o espaço de cores YCbCr, no qual o valor de Y pertence ao intervalo $[0, 1]$ e os valores de Cb e Cr pertencem ao intervalo $[-0.5, 0.5]$. A conversão entre os valores de RGB para YCbCr no Matlab foi realizada da seguinte forma:

$$Y = (0.299 * R + 0.587 * G + 0.114 * B) / 255$$

$$Cb = (-0.14713 * R - 0.28886 * G + 0.436 * B) / 255$$

$$Cr = (0.615 * R - 0.51498 * G - 0.10001 * B) / 255$$

Para equilibrar a luminosidade dos *pixels*, todos os valores da matriz Y foram igualados ao valor 0.7, que mostrou-se mais adequado após alguns testes com valores no intervalo [0,1], e então convertidos novamente para RGB:

$$R = (Y + 1.13983 * Cr) * 255$$

$$G = (Y - 0.39465 * Cb - 0.58060 * Cr) * 255$$

$$B = (Y + 2.03211 * Cb) * 255$$

A entrada do Algoritmo de Localização de Face II é o nome da imagem de entrada e a saída inclui uma imagem colorida resultante da alteração da iluminação da imagem original, uma imagem binária resultante da segmentação de pele e uma imagem colorida com a face da imagem de entrada delimitada. Alguns resultados são mostrados na Figura 12.

O resultado desse algoritmo foi um aumento na taxa de verdadeiros positivos na segmentação de pele, mas também falsos positivos. Além disso, em algumas imagens a delimitação da face foi sutil com relação às suas dimensões, como na Figura 12(d).

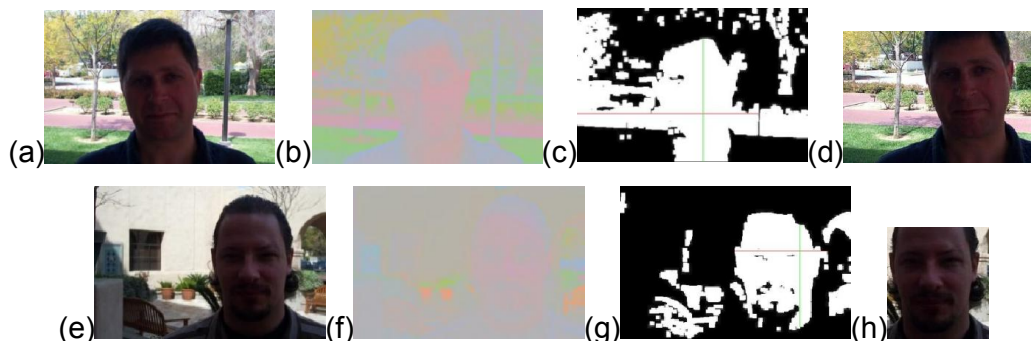


Figura 12: (a) e (e): imagens originais; (b) e (f): imagens com o valor da luminância Y igual à 0.7; (c) e (g): imagens binárias após a segmentação, dilatação e localização da maior linha (vermelho) e maior coluna (verde); (d) e (h) imagens resultantes do Algoritmo de Localização de Face II..
Fonte: CALTECH, 1999.

5.1.1.3 Algoritmo de Localização de Face III

O percentual do plano de fundo contido na imagem resultante do processamento de localização de face foi crucial na detecção dos componentes da face, pois observou-se que o espaço de busca das próximas etapas precisaria ser delimitado à área da face. Para minimizar esse problema apresentado pelo algoritmo anterior, um novo algoritmo foi implementado utilizando o mesmo método de segmentação de pele do Algoritmo de Localização de Face II, mas um método diferente na delimitação da face. Analisou-se as imagens contidas no banco de dados e o valor médio encontrado para a largura e altura das faces foi 200 e 120 *pixels*, respectivamente. Diante disso, percorreu-se as imagens binárias resultantes da segmentação de pele de modo a encontrar o retângulo de dimensões 200x120 *pixels* contido nas imagens com o maior percentual de 'pele', ou seja, com a maior densidade de *pixels* brancos. Essa abordagem se mostrou mais eficiente para esse banco de dados do que as anteriores, porém, esse algoritmo é mais limitado em razão das dimensões máximas das faces serem pré-definidas.

A entrada do Algoritmo de Localização de Face III é o nome da imagem de entrada e a saída é uma imagem colorida com a face da imagem de entrada delimitada. Alguns resultados podem ser vistos na Figura 13.



Figura 13: (a) e (c): imagens originais; (b) e (d): imagens resultantes do Algoritmo de Localização de Face III. Fonte: LE, [s.d.].

5.1.1.4 Algoritmo de Localização de Face IV

O resultado do Algoritmo III foi satisfatório, mas ainda havia o problema de que o plano de fundo de algumas imagens continha muitos pixels classificados como pertencente à região da pele. Por isso foi implementado um novo algoritmo de segmentação de pele, substituindo o classificador de pixels em RGB dos algoritmos anteriores por uma rede neural *Feedforward* com duas camadas ocultas, sendo que a primeira camada oculta possui cinco neurônios e a segunda possui três neurônios.

A entrada da rede possui três parâmetros, que correspondem aos valores de R, G e B de um *pixel* e a saída é um valor no intervalo $[-1,1]$, sendo que os *pixels* com valores de saída mais próximos à zero têm uma menor probabilidade de pertencer à uma região da pele. A Figura 14 ilustra o modelo da rede neural utilizada, que é retornado pelo Matlab após o treinamento da rede.

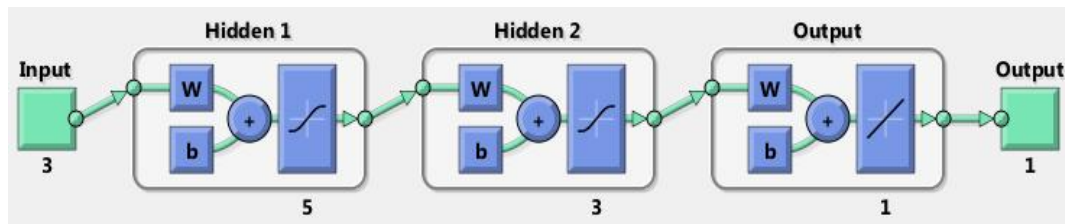


Figura 14: Modelo da rede neural utilizada na segmentação de pele do Algoritmo de Localização de Face IV. Fonte: THE MATHWORKS.

O conjunto de entrada foi composto por valores em RGB de 546.310 *pixels* e a saída esperada foi uma matriz binária de dimensões 1×546.310 , na qual os *pixels* pertencentes à pele das faces tinham o valor de saída 1, e os demais 0.

O conjunto de entrada da rede foi composto pelos *pixels* de 40 imagens contendo pessoas de diferentes etnias do banco de dados *Helen*, ressaltando-se que esse conjunto de imagens não possui interseção com as 500 imagens de teste. Em cada imagem, os *pixels* da pele foram pintados de azul e, a partir dessas imagens pintadas e das imagens originais, foram criadas as matrizes de entrada e a saída esperada da rede neural, na qual os *pixels* em azul receberam o valor de saída 1. Algumas amostras são apresentadas na Figura 15.

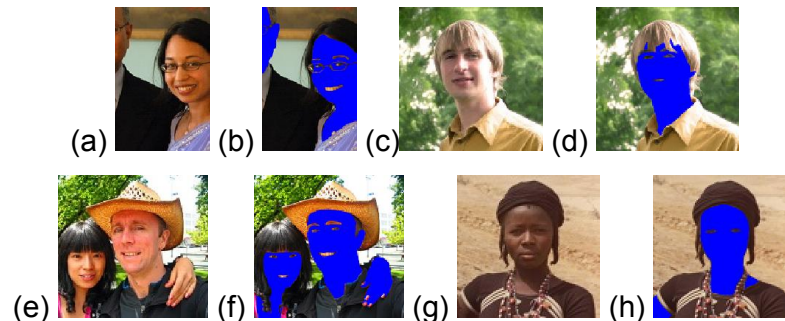


Figura 15: (a), (c), (e) e (g): imagens originais; (b), (d), (f) e (h): imagens segmentadas manualmente. Fonte: LE, [s.d.].

Na criação da rede, foi utilizada a função do Matlab *feedforwardnet* e o treinamento foi realizado com 100 épocas utilizando uma função baseada na

otimização de Levenberg-Marquardt. Depois de treinada, a rede foi salva e aplicada à todas as imagens de teste.

O resultado da aplicação da rede para cada imagem foi uma matriz com valores no intervalo $[-1,1]$, na qual cada valor representava a saída de cada *pixel*. As matrizes resultantes foram então convertidas em matrizes positivas, nas quais os valores mais próximos à 1 representavam os *pixels* da pele, e em seguida redimensionadas de acordo com as dimensões originais de cada imagem. A função do Matlab $f(x) = abs(x)$ foi utilizada para converter as matrizes resultantes em matrizes positivas, pois retorna o módulo da entrada.

Posteriormente, o mesmo método descrito no Algoritmo de Localização de Face III, que busca na imagem um retângulo de 200×120 *pixels* com o maior percentual de pele, foi aplicado. As imagens resultantes foram, então, redimensionadas para 100 colunas e um número de linhas proporcional ao aumento/diminuição das colunas.

A entrada do Algoritmo de Localização de Face IV é o nome da imagem de entrada e a saída inclui uma imagem em escala de cinza, na qual as áreas mais escuras representam os valores mais próximos a zero da matriz resultante, e uma imagem colorida com a face da imagem de entrada delimitada.

A precisão desse algoritmo foi superior aos algoritmos anteriores, pois apresentou menos falsos positivos, e, por isso, os seus resultados foram utilizados nos testes dos algoritmos de localização dos olhos. Algumas imagens resultantes desse algoritmo podem ser vistas na Figura 16.

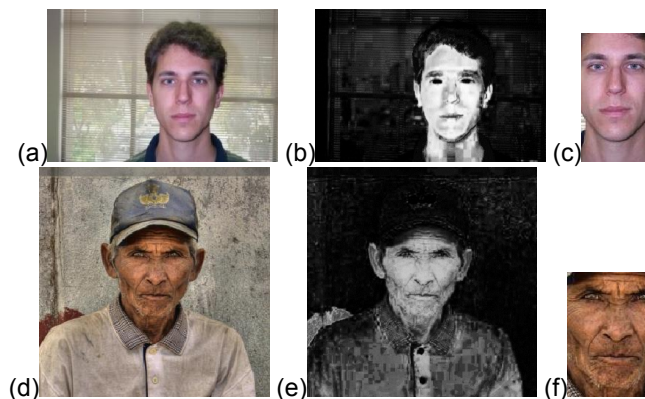




Figura 16: (a), (d) e (g): imagens originais; (b), (e) e (h): imagem resultante da aplicação da rede neural; (c), (f) e (i): imagens resultantes do Algoritmo de Localização de Face IV. Fonte: CALTECH, 1999; LE, [s.d.].

5.1.2 Localização dos olhos

Nessa etapa, foram implementados e comparados três algoritmos de localização dos olhos, que foram aplicados às imagens resultantes do Algoritmo de Localização de Face IV. A implementação desses algoritmos em Matlab encontra-se no Apêndice C.

5.1.2.1 Algoritmo de Localização dos Olhos I

O primeiro algoritmo teve o objetivo de identificar a posição aproximada dos olhos na imagem, buscando encontrar uma linha que intersecte os olhos. Considerando uma imagem na qual os olhos estão abertos, espera-se que as linhas que intersectam essa região apresentem mudanças consideráveis das cores dos seus *pixels* em pelo menos oito pontos, que correspondem aos cantos dos olhos e da íris. Assim, para cada linha da primeira metade da imagem, foi calculada a soma dos ângulos entre *pixels* vizinhos, que representa a diferença entre a tonalidade de dois pixels. Quanto maior for a diferença entre duas cores, maior será o ângulo entre os *pixels*.

A entrada do Algoritmo de Localização dos Olhos I é uma face delimitada e a saída é a mesma face de entrada contendo uma linha vermelha na região dos olhos. A linha resultante do algoritmo é a que apresenta o maior valor para a soma dos ângulos dos seus *pixels* vizinhos e dos *pixels* vizinhos das duas linhas imediatamente superiores e inferiores. Alguns resultados podem ser vistos na Figura 17.



Esse algoritmo se mostrou eficiente para imagens com diferentes níveis de iluminação, tamanhos dos olhos e cores da íris. O algoritmo falhou para imagens nas quais o cabelo cobria uma parte da face ou quando a detecção da face estava incorreta, como nas imagens da Figura 18. Ressalta-se que a presença de franja geralmente cria vários intervalos de regiões de pele e cabelo alternadas, o que eleva a soma dos ângulos entre os *pixels* das linhas da testa.



5.1.2.2 Algoritmo de Localização dos Olhos II

No segundo algoritmo, os olhos foram localizados separadamente, executando-se o mesmo código para cada metade vertical da face. Assim como no algoritmo anterior, a busca foi realizada apenas na primeira metade horizontal da imagem.

Considerando o espaço de cores YCrCb, se igualarmos o valor da luminância para todos os *pixels* de uma imagem contendo uma face, pode-se observar que as regiões dos olhos e dos cabelos apresentam cores mais próximas do azul e do verde, enquanto as regiões da pele são sempre mais alaranjadas/avermelhadas. Isso pode ser visto nas imagens resultantes do Algoritmo de Localização de Face II, descrito neste trabalho. Com isso, espera-se que os *pixels* da região dos olhos apresentem valores maiores para verde e azul do que os *pixels* da pele e, os

menores valores para vermelho no espaço RGB. Conseqüentemente, essa região apresentará os menores valores para Cr e os maiores valores para Cb. Porém, apenas os valores máximos para Cb foram utilizados no algoritmo, pois, a partir de alguns testes realizados, mostrou-se mais eficiente do que o uso dos valores de Cb e Cr em conjunto.

O objetivo do algoritmo foi encontrar a área de 20×10 pixels com a maior soma dos valores de Cr. As dimensões da área foram definidas a partir da média da área dos olhos das faces do banco de dados.

A entrada do Algoritmo de Localização dos Olhos II é uma imagem com uma face delimitada e a saída é a face de entrada com as regiões do olhos delimitadas por retângulos com bordas azuis.

A vantagem desse algoritmo com relação ao anterior é que, quando correta, a localização dos olhos é mais precisa e pode ser utilizada no alinhamento da face. Alguns resultados corretos podem ser vistos na Figura 19.

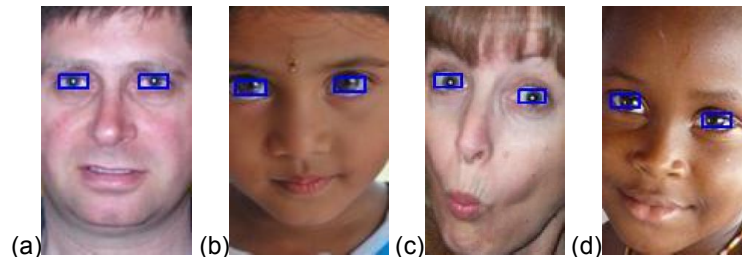


Figura 19: Imagens resultantes do Algoritmo II de localização dos olhos.
Fonte: CALTECH, 1999; LE, [s.d.].

Esse algoritmo se mostrou pouco eficiente para as faces com sobrancelhas e cabelos muito escuros e espessos, pois, quanto mais se aproxima da cor preta, menores são os valores para os três componentes do espaço RGB, e o mesmo acontece para imagens muito ou pouco iluminadas. A Figura 20 apresenta alguns resultados incorretos.

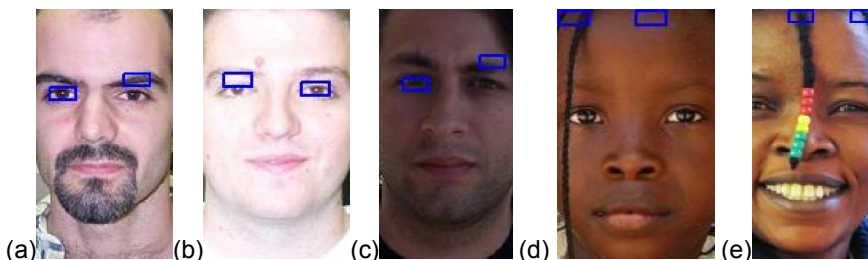


Figura 20: (a), (c), (e): imagens originais; (b), (d), (f): imagens resultantes da localização incorreta dos olhos. Fonte: CALTECH, 1999; LE, [s.d.].

5.1.2.3 Algoritmo de Localização dos Olhos III

Devido as falhas dos Algoritmos de Localização dos Olhos I e II, um terceiro algoritmo foi criado combinando os dois primeiros. Assim, além da soma dos valores de Cr , a soma dos ângulos entre os *pixels* de uma área de 30×5 *pixels* foi considerada.

A entrada do Algoritmo de Localização dos Olhos III é uma imagem com uma face delimitada e a saída é a face de entrada com as regiões do olhos delimitadas por retângulos com bordas azuis, e duas linhas vermelhas, que representam os limites superior e inferior do espaço de busca do algoritmo.

O resultado, mostrado na Figura 21, foi superior aos algoritmos anteriores, sendo utilizado em seguida para o alinhamento das faces.

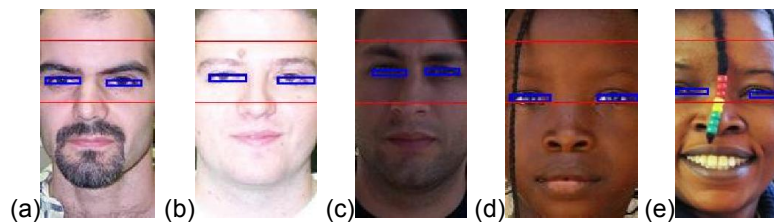


Figura 21: Imagens resultantes do Algoritmo de Localização dos Olhos III, nas quais os retângulos azuis definem as regiões dos olhos e as linhas vermelhas delimitam a área de busca do algoritmo. Fonte: CALTECH, 1999; LE, [s.d.].

5.1.3 Alinhamento e delimitação da face

O passo seguinte à localização dos olhos foi o alinhamento da face com base nas posições das regiões dos olhos. O algoritmo de alinhamento da face simplesmente calcula a inclinação da reta que passa pelos pontos centrais das regiões de cada olho e depois utiliza a função *imrotate(imagem, α)* do Matlab, que rotaciona *imagem* em α graus no sentido anti-horário em torno do seu centro. A implementação desse algoritmo encontra-se no Apêndice D.

A maioria das imagens desse banco de dados já estavam alinhadas, mas algumas apresentaram uma sutil diferença, como podemos ver na Figura 22.



Figura 22: (a), (c), (e): imagens originais; (b), (d), (f): imagens resultantes do algoritmo de alinhamento. Fonte: CALTECH, 1999; LE, [s.d.].

A partir das imagens alinhadas, as faces foram delimitadas acima da região dos olhos, como mostra a Figura 22. O objetivo foi delimitar ao máximo as faces, mantendo os três componentes principais: olhos, nariz e lábios.



Figura 23: (a), (c), (e): imagens originais; (b), (d), (f): faces delimitadas acima dos olhos. Fonte: CALTECH, 1999; LE, [s.d.].

5.1.4 Localização dos lábios e delimitação da face

A localização dos lábios se deu de forma semelhante à dos olhos. A diferença é que, ao contrário da região dos olhos, espera-se que a região dos lábios apresente os maiores valores para a componente vermelho. Dessa forma, foi considerado a soma dos componentes Cr e Cb do espaço de cores YCrCb, sendo que a área (20x50 pixels) com a maior soma foi definida como a região dos lábios. Dado que: $Cb + Cr = (-0.14713*R - 0.28886*G + 0.436*B) + (0.615*R - 0.51498*G - 0.10001*B) = 0.46787*R - 0.80386*G + 0.33599*B$, podemos concluir que a soma Cb + Cr será maior para os maiores valores de R.

O algoritmo foi aplicado às imagens resultantes do algoritmo de alinhamento da face, sendo que a busca foi realizada apenas na metade inferior da imagem. O código do algoritmo de localização dos lábios encontra-se no Apêndice E e alguns resultados podem ser vistos na Figura 24.

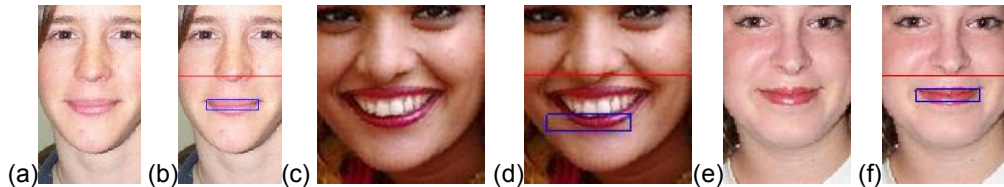


Figura 24: (a), (c), (e): imagens originais; (b), (d), (f): imagens resultantes do algoritmo de localização dos lábios, nas quais a área delimitada pelo retângulo azul representa a região dos lábios e a reta vermelha representa o início do espaço de busca. Fonte: CALTECH, 1999; LE, [s.d.].

Utilizando o resultado do algoritmo anterior, as faces foram delimitadas na região inferior aos lábios, eliminando parte do queixo e o pescoço das imagens, como ilustrado na Figura 25.



Figura 25: (a), (c), (e): imagens originais; (b), (d), (f): imagens delimitadas abaixo dos lábios. Fonte: CALTECH, 1999; LE, [s.d.].

5.2 Reconhecimento de face

Nesta fase, apenas as imagens do banco de dados *Faces 1999* foram utilizadas pois, para se testar a eficiência de um algoritmo de reconhecimento de face, é necessário haver mais de uma imagem de cada indivíduo.

5.2.1 Redimensionamento e alteração na iluminação

Esta etapa teve o objetivo de equilibrar o nível de iluminação das imagens. Em primeiro lugar, foi preciso estabelecer um padrão para a dimensão das imagens, assim, todas foram redimensionadas para 100x100 *pixels*. No primeiro algoritmo implementado, algumas imagens foram escolhidas para determinarem o padrão de iluminação para o restante das faces. Foram escolhidas oito imagens com um nível de luminância (Y) intermediário. Então, foi calculada a média das oito matrizes de luminância que, por sua vez, foi utilizada como parâmetro para alterar a iluminação de cada imagem do banco de dados.

As oito matrizes de luminância e a média estão representadas na Figura 26 sob a forma de imagens em escala de cinza, nas quais as regiões mais claras representam os maiores valores de Y .

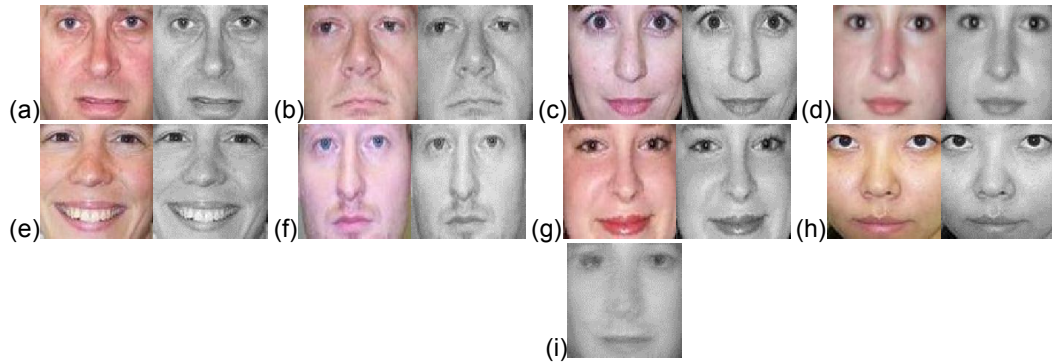


Figura 26: (a), (b), (c), (d), (e), (f), (g), (h): imagens escolhidas e suas matrizes de luminância representadas em escala de cinza, (i): média das matrizes de luminância. Fonte: CALTECH, 1999; LE, [s.d.].

Nesse algoritmo, para cada imagem, a componente luminância resultante foi definida como $Y_i/2 + Y_\beta/2$, sendo Y_i e Y_β a luminância da imagem e a luminância da média, respectivamente. O resultado foi a diminuição da diferença de iluminação e da nitidez das imagens, como mostra a Figura 27.

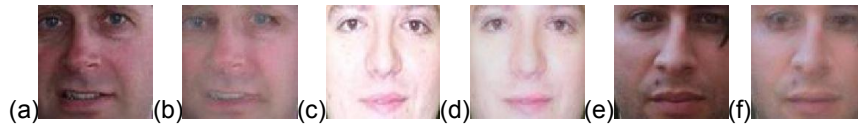


Figura 27: (a),(c), (e): imagens originais; (b), (d), (f): imagens resultantes do primeiro algoritmo de alteração da iluminação. Fonte: CALTECH, 1999; LE, [s.d.].

O segundo algoritmo para equilibrar a iluminação das imagens utilizou a média dos valores da matriz média de luminância calculada anteriormente e o código encontra-se no Apêndice E. Para cada imagem, a componente de luminância resultante foi definida como $Y_i * (\lambda_i/\lambda_\beta)$, sendo λ_i e λ_β a média dos valores das matrizes de luminância da imagem e da matriz de luminância média, respectivamente. O resultado foi superior ao algoritmo anterior, pois não alterou a nitidez das imagens, como mostra a Figura 28.



Figura 28: (a),(c), (e): imagens originais; (b), (d), (f): imagens resultantes do segundo algoritmo de alteração da iluminação. Fonte: CALTECH, 1999; LE, [s.d.].

5.2.2 Descrição do algoritmo de reconhecimento de face

A detecção das faces e padronização das imagens nas etapas anteriores influenciaram diretamente nos resultados do algoritmo de reconhecimento de face. Esperava-se que, quanto maior a acurácia dos resultados da detecção de face, maior a probabilidade de as faces serem corretamente classificadas.

A identificação das faces do banco de dados se baseou no padrão de distribuição das cores em cada imagem, que é uma técnica global de classificação, pois não analisa separadamente os componentes da face. Para cada indivíduo foram escolhidas duas imagens com suas faces corretamente detectadas, exceto para dois indivíduos dos quais só havia uma foto no banco de dados. As médias das faces escolhidas serviram como base para a classificação do restante das faces e serão denominadas face média α , sendo α o identificador do indivíduo. Nenhuma das imagens escolhidas para as médias foram utilizadas no teste.

No algoritmo, cada face foi comparada às faces médias dos 28 indivíduos. O algoritmo utiliza o espaço YCbCr para calcular o desvio padrão das matrizes Cb e Cr da diferença entre a face e as faces médias, e então retorna o identificador da face correspondente ao menor desvio padrão. A Figura 29 ilustra uma parte do resultado do algoritmo para uma das imagens do Indivíduo 16, na qual as matrizes Cb e Cr estão representadas como imagens em escala de cinza.

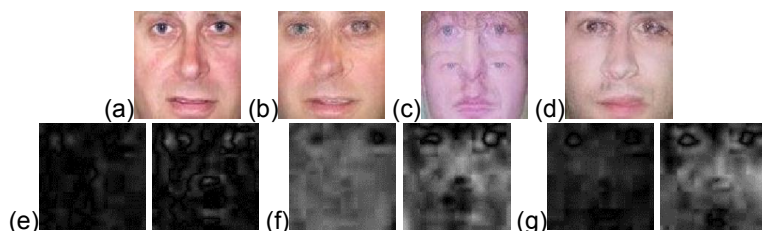


Figura 29: (a) Face do Indivíduo 1; (b) Face média 1; (c) Face média 16; (d) Média 19; (e) Matrizes Cb e Cr da diferença entre (a) e (b); (f) Matrizes Cb e Cr da diferença entre (a) e (c); (g) Matrizes Cb e Cr da diferença entre (a) e (d). Fonte: CALTECH, 1999; LE, [s.d.].

Pode-se notar na figura acima que a variação das tonalidades de cinza das matrizes Cb e Cr é maior quando a face média não corresponde à face do indivíduo. Isso ocorre devido às propriedades particulares de cada face, como as dimensões dos olhos, lábios e nariz e a tonalidade da pele. Assim, quanto menor for a variação dos valores em Cb e Cr, menor será a soma dos desvios padrão de Cb e Cr e maior será a probabilidade de a face pertencer àquele indivíduo. A face na Figura 29 (a) foi

corretamente detectada, sendo as somas dos desvios padrão das matrizes na Figura 29(e), Figura 29(f) e Figura 29(g) iguais a 0.0307, 0.0481 e 0.0419, respectivamente.

Contudo, para que esse método funcione, é necessário que os componentes da face estejam bem localizados e alinhados em todas as imagens, pois assim as regiões dos olhos, nariz e lábios podem ser subtraídas das regiões correspondentes nas faces médias.

6 RESULTADOS E DISCUSSÃO

6.1 Resultados dos algoritmos de detecção de face

A seguir serão apresentados os resultados quantitativos dos algoritmos relacionados à detecção de face, descritos no capítulo anterior.

A Tabela 1 apresenta os resultados dos algoritmos de localização de face para as 500 imagens de teste. Os resultados corretos incluem as imagens nas quais as faces foram totalmente detectadas e sem cortes. Os resultados com parte da face detectada incluem as imagens nas quais uma área significativa de face tenha sido recortada. As faces com plano de fundo são as imagens nas quais as faces foram completamente detectadas, mas com uma grande área do plano de fundo incluída no resultado. Já os resultados incorretos são as imagens nas quais apenas parte do plano de fundo foi detectado como uma face.

Tabela 1: Resultados dos algoritmos de localização de face com segmentação de pele

Algoritmos de Localização de Face	Resultados corretos (%)	Face com plano de fundo (%)	Parte da face detectada (%)	Resultados incorretos (%)
Algoritmo I	67,4	17	8,8	6,8
Algoritmo II	48,6	46,8	3,6	1
Algoritmo III	88,4	1,6	5,4	4,6
Algoritmo IV	95,8	1,6	2,4	0,2

Um dos desafios encontrados na implementação da localização de face por meio da segmentação de pele foi identificar uma face em uma imagem na qual o plano de fundo apresentava cores muito próximas às da face. Além disso, a falta ou excesso de iluminação interferiu diretamente nos resultados, o que foi mostrado na descrição do Algoritmo de Localização de Face I.

Para minimizar a interferência da iluminação sobre o resultado, o Algoritmo de Localização de Face II foi utilizado, todavia a precisão dos resultados diminuiu significativamente, o que pode ser visto comparando-se o número de resultados corretos dos dois primeiros algoritmos.

Em contrapartida, a utilização de uma área de dimensões pré-definidas para as faces, definida no Algoritmo de Localização de Face III, aumentou significativamente

o percentual de acerto pois, na maioria das imagens, a área da face era maior do que a área do plano de fundo detectado como pertencente à pele.

Porém, para tornar o método de detecção de face independente das cores do plano de fundo, uma rede neural foi implementada no Algoritmo de Localização de Face IV utilizando imagens com diversos planos de fundo para o treinamento. Apesar de o Algoritmo IV apresentar bons resultados para as imagens de teste, ainda há várias melhorias a serem implementadas, incluindo a detecção de várias faces com dimensões variadas em uma mesma imagem.

Os resultados dos algoritmos de detecção de olhos são mostrados na Tabela 2. A pré-condição dos três algoritmos foi que a face na imagem de entrada deveria estar bem delimitada e, por isso, a maioria das entradas que continham uma parte do plano de fundo apresentaram resultados incorretos. O Algoritmo de Detecção dos Olhos III apresentou os melhores resultados e foi utilizado no alinhamento das faces, todavia apresenta limitações quanto ao ângulo de inclinação da face e precisão da localização da pupila. Para tornar o algoritmo mais genérico, seria necessário utilizar um método que localize precisamente o centro dos olhos.

Tabela 2: Resultados dos algoritmos de detecção de olhos a partir das imagens resultantes do Algoritmo de Localização de Face IV de detecção de face

Algoritmos de Detecção dos Olhos	Resultados corretos (%)	Resultados parcialmente corretos (com apenas um dos olhos detectados) (%)
Algoritmo I	81	0
Algoritmo II	64,2	15,2
Algoritmo III	85	4

A percentagem de acerto do algoritmo de detecção de lábios foi de 90%. Apesar dos bons resultados para as imagens de teste, o algoritmo apresentaria erros caso os lábios estivessem pintados de cores não convencionais, como verde ou azul. Nesses casos, seria necessário um método adicional para detectar os lábios, como a detecção de borda ou a estimativa com base na localização dos olhos.

6.2 Resultados do Algoritmo de Reconhecimento de Face

O índice de acerto do algoritmo de reconhecimento de face foi de 70,74% para todas as imagens testadas e de 77,81% se considerarmos apenas as faces detectadas corretamente. A Tabela do Apêndice A mostra o resultado para todos os indivíduos do banco de dados.

No conjunto das faces corretamente detectadas, os erros de identificação ocorreram principalmente devido aos variados níveis de delimitação das faces e devido à assimetria da iluminação em algumas imagens, nas quais havia sombras projetadas sobre uma parte da mesma. Todavia, o índice de acerto também dependeu das imagens escolhidas para representar cada indivíduo, pois a probabilidade de identificar uma face corretamente aumenta de acordo com o nível de semelhança entre a face de entrada e a média correspondente ao indivíduo ao qual pertence aquela face.

O percentual de acerto desse algoritmo de reconhecimento de face poderia aumentar caso fosse utilizado um algoritmo de detecção de olhos mais acurado pois, com a localização precisa da pupila, as faces poderiam ser delimitadas lateralmente de forma mais padronizada. Além disso, se fossem utilizadas mais imagens para calcular a face média de cada indivíduo, seria possível abranger um número maior de possibilidades quanto ao tipo de iluminação e expressões faciais, o que provavelmente aperfeiçoaria o resultado do algoritmo.

7 CONCLUSÃO

A área de detecção e reconhecimento de face envolve diversas técnicas de processamento de imagens. Faces podem ser detectadas em imagens digitais através da segmentação de pele utilizando classificadores ou redes neurais no espaço de cores RGB e YCbCr. As regiões dos olhos e lábios podem ser encontradas por meio da análise da crominância e nível de variação das cores nas diferentes regiões da face. Por sua vez, o reconhecimento de face pode ser implementado através de uma abordagem global de classificação, que não considera a semântica dos componentes da face, utilizando o desvio padrão da diferença entre a face de entrada e as faces no banco de dados na classificação da mesma.

Aplicando-se as técnicas acima, foram implementados algoritmos relacionados à detecção de face que atingiram resultados satisfatórios, com percentuais de acerto entre 85 e 95,8%. O resultado do algoritmo de reconhecimento de face foi razoável, com 70,74% de acerto.

A principal dificuldade encontrada no desenvolvimento da detecção de face foi adaptar os algoritmos aos diferentes níveis de iluminação e tonalidades de pele das faces, o que foi resolvido por meio da utilização de uma rede neural com imagens de treinamento contendo pessoas de diferentes etnias e por algoritmos que equilibram a iluminação das imagens de entrada. No que diz respeito ao reconhecimento de face, o percentual de acerto foi afetado pela precisão dos resultados da detecção de face, pois, nem todas as faces corretamente detectadas apresentaram o mesmo padrão de proximidade e delimitação da face.

Diante disso, pode-se concluir que há necessidade de aprimoramento dos algoritmos para que estes possam ser utilizados em sistemas para ambientes reais.

8 TRABALHOS FUTUROS

Visando a melhoria dos resultados obtidos, os algoritmos relacionados à detecção de face poderiam ser aprimorados caso múltiplas faces pudessem ser detectadas em uma mesma imagem. Além disso, a rede neural utilizada na segmentação de pele poderia ser treinada com mais entradas, tornando-a menos propícia à detecção incorreta de regiões faciais.

O algoritmo de reconhecimento de face poderia ser melhor avaliado caso fossem utilizadas faces de um número maior de indivíduos com todas as faces delimitadas com precisão. Desta forma, os requisitos da detecção de face também poderiam ser aperfeiçoados.

Por último, seria necessária a implementação de uma interface gráfica para que o sistema pudesse ser utilizado por usuários comuns.

9 REFERÊNCIAS

ANDRADE, A. O. *et al.* **Analysing some R-Implications and its application in Fuzzy Mathematical Morphology**. Journal of Intelligent and Fuzzy Systems. IOS Press. 2013. Disponível em: <<http://iospress.metapress.com/content/3667k988742p9313/>>. Acesso em: 21 jul. 2014.

ASMARE, M.; H; ASIRVADAM, V. S.; IZNITA, L. **Color Space Selection for Color Image Enhancement Application**. Signal Acquisition and Processing. 2009. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5163856&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D5163856>. Acesso em: 29 jun. 2014.

BARBOSA, A. H.; FREITAS, M. S. da R.; NEVES, F. de A. **Confiabilidade estrutural utilizando o método de Monte Carlo e redes neurais**. Revista Escola de Minas. 2005. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0370-44672005000300011>. Acesso em: 16 jun. 2014.

BIEDERMAN, I.; KALOCSAI, P. **Neural and Psychophysical Analysis of Object and Face Recognition**. Springer. 1998. Disponível em: <http://www.kalocsai.net/publications/kalo_nato2.pdf>. Acesso em: 08 abr. 2014.

BRIGGS, D. **The Dimensions of Color**. Disponível em: <<http://www.huevaluechroma.com/012.php>>. Acesso em: 11 jul. 2014.

CALTECH. **Computational Vision**. Disponível em: <<http://www.vision.caltech.edu/html-files/archive.html>>. Acessado em: 21 jul. 2014.

CAMPADELLI, P.; LANZAROTTI, R.; LIPORI, G.. **Automatic facial feature extraction for face recognition**. **Face Recognition**. Capítulo 3. 2007. Disponível em: <<http://cdn.intechweb.org/pdfs/194.pdf>>. Acesso em: 29 out. 2013.

FREITAS, R. F. *et al.* **Algoritmos para segmentação da pele utilizando modelos de cores RGB em ambiente MATILAB/SIMULINK**. VII Encontro de Pesquisa e Pós-Graduação (VII ENPPG). [s.d]. Disponível em: <<http://conexoes.ifce.edu.br/index.php/conexoes/article/download/141/132>>. Acesso em: 12 dez. 2013.

GENG, C.; JIANG, X. **Fully automatic face recognition framework based on local and global features**. Machine Vision and Applications. *Springer*. 2012. Disponível em: <<http://www3.ntu.edu.sg/home/EXDJiang/JiangX.D.-MVA-13.pdf>>. Acesso em: 30 dez. 2013.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5ª edição. São Paulo: Atlas, 2010.

GONZALES, R. C.; WOODS, R. E. **Digital Image Processing**. 2ª edição. Upper Saddle River: Prentice Hall. 2002. Disponível em: <http://users.dcc.uchile.cl/~jsaavedr/libros/dip_gw.pdf >. Acesso em: 19 jul. 2014.

GOTTUMUKKAL, R.; ASARI, V. **An improved face recognition technique based on modular PCA approach**. *Elsevier*. 2003. Disponível em: <<http://www.utdallas.edu/~sxb027100/dock/25-429.pdf> >. Acesso em: 29 out. 2013.

GOYAL, M.. **Morphological Image Processing**. International Journal of Science and Technology. 2011. Disponível em: <<http://www.ijcst.com/vol24/1/megha.pdf>>. Acesso em: 18 jul. 2014.

HAYKIN, S. **Introdução**. In: **Redes Neurais: Princípios e Prática**. Hamilton: bookman. 2001. Disponível em: <http://www.ncdd.com.br/livros/redes_neurais_simon_haykin.pdf >. Acesso em: 16 jun. 2014.

HEISELE, Bernd. *et al.* **Face recognition component-based versus global approaches**. Computer Vision and Image Understanding. *Elsevier*. 2003. Disponível em: <<http://cbcl.csail.mit.edu/projects/cbcl/publications/ps/heisele-cviu-03.pdf>>. Acesso em: 15 jan. 2014.

ITO, Y. et al. **Detection of eyes by circular Hough transform and histogram of gradient**. 21st International Conference on Pattern Recognition (ICPR 2012). 2012. Disponível em: <<http://vision.unipv.it/VA/VA-En/TEST/Circuit%20board%20MM/Eyes.pdf> >. Acesso em: 26 dez. 2013.

LAKSHMI, H. C. V.; PATILKULAKARNI, S. **Segmentation algorithm for multiple face detection in color images with skin tone regions using color spaces and edge detection techniques**. International Journal of Computer

Theory and Engineering. 2010. Disponível em: <<http://www.ijcte.org/papers/200-H080.pdf>>. Acesso em: 28 dez. 2013.

LE, V. **Helen dataset**. Disponível em: <<http://www.ifp.illinois.edu/~vuongle2/helen/>>. Acessado em: 21 jul. 2014.

LUCHESEY, L.; MITRAY, S.K. **Color Image Segmentation: A State-of-the-Art Survey**. 2001. Disponível em: <<http://ultra.sdk.free.fr/docs/Image-Processing/filters/Color%20Image%20Segmentation-A%20State-of-the-Art%20Survey.pdf>>. Acesso em: 21 jul. 2014.

MUSTAFIA, D.; ENGEL, A. H.; PALCZEWSKI, K.. **Structure of cone photoreceptors**. Progress in Retinal and Eye Research. Elsevier. 2009. Disponível em: <http://www.researchgate.net/publication/223387833_Structure_of_cone_photoreceptors/file/50463529cfdd350c5c.pdf>. Acesso em: 12 jul. 2014.

NASCIMENTO, A. V. *et al.* **Comparação de Clusters para detecção de pele**. Biblioteca Digital Brasileira de Computação. 2007. Disponível em: <<http://www.lbd.dcc.ufmg.br/colecoes/wvc/2007/0029.pdf>>. Acesso em: 17 mar. 2014.

SIDIB, D.; MONTESINOS, P.; JANAQUI, S. **A simple and efficient eye detection method in color images**. International Conference Image and Vision Computing New Zealand. 2006. Disponível em: <http://hal.inria.fr/docs/00/37/43/66/PDF/article_ivcnz2006.pdf>. Acesso em: 28 dez. 2013.

SINGH, S. K. *et al.* **A robust skin color based face detection algorithm**. **Tamkang Journal of Science and Engineering**. 2003. Disponível em: <<http://www.csee.wvu.edu/~richas/papers/tkjse.pdf>>. Acesso em: 11 dez. 2013.

THE MATHWORKS, INC. **Matlab and Statistics Toolbox Release 2011a**. Natick, Massachusetts, United States.
YCbCr. In: WIKIPÉDIA: The Free Encyclopedia. Disponível em: <http://en.wikipedia.org/wiki/Image_scaling>. Acesso em: 18 jul. 2014.

ZAPATA, C. **Modelo RGB**. Disponível em <<http://cindy2906.blogspot.com.br/2013/02/modelo-rgb.html>>. Acesso em: 15 jul. 2014.

ZHAO, W. *et al.* **Face recognition: A literature survey**. Journal ACM Computing Surveys (CSUR). 2003. Disponível em:
<<http://www.cs.ucf.edu/~dcm/Teaching/COT4810-Spring2011/Literature/DiegoVelasquez-FaceRecognitionLiteratureSurvey.pdf>>.
Acesso em: 06 abr. 2014.

APÊNDICE A – Tabela dos Resultados do Algoritmo de Reconhecimento de Face

Tabela: Resultado do algoritmo de reconhecimento/classificação de face

Indivíduos	Número de imagens	Faces detectadas corretamente	Faces reconhecidas/classificadas corretamente
Indivíduo 1	19	17	11
Indivíduo 2	18	18	15
Indivíduo 3	3	3	1
Indivíduo 4	20	20	16
Indivíduo 5	19	17	17
Indivíduo 6	21	19	18
Indivíduo 7	18	18	18
Indivíduo 8	3	1	1
Indivíduo 9	19	18	12
Indivíduo 10	5	5	5
Indivíduo 11	3	3	2
Indivíduo 12	3	1	0
Indivíduo 13	18	16	6
Indivíduo 14	19	18	17
Indivíduo 15	23	23	19
Indivíduo 16	20	19	12
Indivíduo 17	3	2	2
Indivíduo 18	17	17	12
Indivíduo 19	18	16	6
Indivíduo 20	27	8	4
Indivíduo 21	18	18	13
Indivíduo 22	18	18	18
Indivíduo 23	20	20	20
Indivíduo 26	3	3	2
Indivíduo 27	18	18	17
Indivíduo 28	20	20	14
Total	393	356	278

APÊNDICE B – Algoritmos de Localização de Face em Matlab

a) Código do Algoritmo de Localização de Face I

```

% A entrada dessa função é o nome da imagem de entrada.
% A saída inclui a imagem binária resultante da segmentação de pele e uma
% imagem colorida com a face da imagem de entrada delimitada.
function [imagem, imagemDelimitada] = detectarFace1( nomeImagem )

% A função segmentarPele1 cria uma imagem binária com a região da pele
% segmentada. O código dessa função será apresentado mais adiante.
segmentarPele1(nomeImagem);
imagemBinaria = imread(strcat(nomeImagem, '_skinBW.jpg'));
elemento = strel('square', 8);~

% A imagem binária é dilatada utilizando um elemento estruturante quadrado
% e branco de 8x8 pixels.
imagemBinaria = imdilate(imagemBinaria, elemento);
linhasImagem = size(imagemBinaria, 1);
colunasImagem = size(imagemBinaria, 2);

% Procura a maior sequência horizontal de pixels brancos na imagem. Espera-
% se que esta sequência esteja no interior da região do rosto.
inicioLinhaHorizontal = 1;
inicioColunaHorizontal = 1;
larguraMaiorLinha = 0;
for i=1:linhasImagem
    for j=1:colunasImagem
        larguraMaiorLinhaAux = 0;
        while (imagemBinaria(i,j,1) == 255)
            larguraMaiorLinhaAux = larguraMaiorLinhaAux + 1;
            j = j+1;
            if (j>colunasImagem)
                break;
            end
        end
        if (larguraMaiorLinhaAux > larguraMaiorLinha)
            inicioLinhaHorizontal = i;
            inicioColunaHorizontal = j - larguraMaiorLinhaAux;
            larguraMaiorLinha = larguraMaiorLinhaAux;
        end
    end
end

% Procura a maior sequência vertical de pixels brancos na imagem. Espera-se
% que esta sequência esteja no interior da região do rosto.
inicioLinhaVertical = 1;
inicioColunaVertical = 1;
alturaMaiorColuna = 0;
for j=1:colunasImagem
    for i=1:linhasImagem
        alturaMaiorColunaAux = 0;
        while (imagemBinaria(i,j,1) == 255)
            alturaMaiorColunaAux = alturaMaiorColunaAux + 1;
            i = i+1;
            if (i>linhasImagem)
                break;
            end
        end
    end
end

```

```

end
if (alturaMaiorColunaAux > alturaMaiorColuna)
    inicioLinhaVertical = i - alturaMaiorColunaAux;
    inicioColunaVertical = j;
    alturaMaiorColuna = alturaMaiorColunaAux;
end
end
end

% Verifica se as retas das maiores sequências horizontal e vertical se
% cruzam ou não.
if ~(inicioLinhaHorizontal>inicioLinhaVertical &&
    inicioLinhaHorizontal<inicioLinhaVertical+alturaMaiorColuna &&
    inicioColunaVertical>inicioColunaHorizontal &&
    inicioColunaVertical<inicioColunaHorizontal+larguraMaiorLinha)

% Caso não se cruzem, calcula-se a maior linha que intersecte a
% maior coluna e a maior coluna que intersecte a maior linha encontrada.

% Procura a maior linha com pixels brancos consecutivos que corte a maior
% coluna encontrada anteriormente.
inicioLinhaHorizontal2 = 1;
inicioColunaHorizontal2 = 1;
larguraMaiorLinha2 = 0;
for k=inicioLinhaVertical:inicioLinhaVertical+alturaMaiorColuna-1
    for l=1:colunasImagem
        larguraMaiorLinhaAux = 0;
        while (imagemBinaria(k,l,1) == 255)
            larguraMaiorLinhaAux = larguraMaiorLinhaAux + 1;
            l = l+1;
            if (l>colunasImagem)
                break;
            end
        end
        % Verifica se a nova linha encontrada corta a maior coluna
        % encontrada anteriormente
        if (larguraMaiorLinhaAux > larguraMaiorLinha2 && l -
            larguraMaiorLinhaAux<inicioColunaVertical && l >
            inicioColunaVertical )
            inicioLinhaHorizontal2 = k;
            inicioColunaHorizontal2 = l - larguraMaiorLinhaAux;
            larguraMaiorLinha2 = larguraMaiorLinhaAux;
        end
    end
end

% Procura a maior coluna com pixels brancos consecutivos que corte a maior
% linha encontrada anteriormente.
inicioLinhaVertical2 = 1;
inicioColunaVertical2 = 1;
alturaMaiorColuna2 = 0;
for l=inicioColunaHorizontal:inicioColunaHorizontal+larguraMaiorLinha-1
    for k=1:linhasImagem
        alturaMaiorColunaAux = 0;
        while (imagemBinaria(k,l,1) == 255)
            alturaMaiorColunaAux = alturaMaiorColunaAux + 1;
            k = k+1;
            if (k>linhasImagem)
                break;
            end
        end
    end
end

```

```

    % Verifica se a nova coluna encontrada corta a maior linha
    % encontrada anteriormente
    if (alturaMaiorColunaAux > alturaMaiorColuna2 && k -
        alturaMaiorColunaAux < inicioLinhaHorizontal && k >
        inicioLinhaHorizontal)
        inicioLinhaVertical2 = k - alturaMaiorColunaAux;
        inicioColunaVertical2 = 1;
        alturaMaiorColuna2 = alturaMaiorColunaAux;
    end
end
end

% A linha mais larga ou a coluna mais comprida é substituída pelo novo
% valor encontrado.
if (larguraMaiorLinha2 >= alturaMaiorColuna2)
    inicioLinhaHorizontal = inicioLinhaHorizontal2;
    inicioColunaHorizontal = inicioColunaHorizontal2;
    larguraMaiorLinha = larguraMaiorLinha2;
else
    inicioLinhaVertical = inicioLinhaVertical2;
    inicioColunaVertical = inicioColunaVertical2;
    alturaMaiorColuna = alturaMaiorColuna2;
end
end

% Desenha uma linha vermelha na maior sequência horizontal de pixels
% brancos.
for j=inicioColunaHorizontal:inicioColunaHorizontal+larguraMaiorLinha
    imagem(inicioLinhaHorizontal, j, 1) = 255;
    imagem(inicioLinhaHorizontal, j, 2) = 0;
    imagem(inicioLinhaHorizontal, j, 3) = 0;
end

% Desenha uma linha verde na maior sequência vertical de pixels brancos.
for i=inicioLinhaVertical:inicioLinhaVertical+alturaMaiorColuna
    imagem(i, inicioColunaVertical, 1) = 0;
    imagem(i, inicioColunaVertical, 2) = 255;
    imagem(i, inicioColunaVertical, 3) = 0;
end

imwrite(imagemBinaria, strcat(nomeImagem, '_skinBW.jpg'));

% Cria uma nova imagem com apenas o 'rosto' detectado na imagem original
imagemOriginal = imread(strcat(nomeImagem, '.jpg'));
imagemDelimitada = imcrop(imagemOriginal, [inicioColunaHorizontal,
    inicioLinhaVertical, larguraMaiorLinha, alturaMaiorColuna]);
imwrite(imagemDelimitada, strcat(nomeImagem, '_face.jpg'));
end

```

```

% A entrada dessa função é o nome da imagem de entrada e a saída é uma
% imagem binária resultante da segmentação de pele por meio de um
% classificador de pixels em RGB
function [imagem] = segmentarPele1(nomeImagem)

imagem = imread(strcat(nomeImagem, '.jpg'));
linhasImagem = size(imagem, 1);
colunasImagem = size(imagem, 2);

% Segmenta a imagem de entrada de acordo com um classificador de pixels em
% RGB, separando os pixels que pertencem do restante dos pixels.
for i=1:linhasImagem
    for j=1:colunasImagem
        ehPele = false;
        if (imagem(i,j,1) > 95 && imagem(i,j,2) > 40 && imagem(i,j,3) > 20)
            if (max([imagem(i,j,1), imagem(i,j,2), imagem(i,j,3)]) -
min([imagem(i,j,1), imagem(i,j,2), imagem(i,j,3)])) > 15
                if abs(imagem(i,j,1) -imagem(i,j,2)) > 15
                    if (imagem(i,j,1) > imagem(i,j,2) && imagem(i,j,1) >
imagem(i,j,3))
                        ehPele = true;

                        % Os pixels classificados como pertencentes à pele
                        % recebem o valor 255, correspondente à cor branca.
                        imagem(i,j,1) = 255;
                        imagem(i,j,2) = 255;
                        imagem(i,j,3) = 255;
                    end
                end
            end
        end
    end

    % Os pixels que não são classificados como pertencentes à pele
    % recebem o valor zero, correspondente ao preto.
    if ehPele == false
        imagem(i,j,1) = 0;
        imagem(i,j,2) = 0;
        imagem(i,j,3) = 0;
    end
end
end

% Armazena o resultado em uma imagem binária
imwrite(imagem, strcat(nomeImagem, '_skinBW.jpg'));

end

```

b) Código do Algoritmo de Localização de Face II

% Após a execução desse código para todas as imagens do banco de dados, o
% mesmo código de delimitação de pele do Algoritmo de Localização de Face I
% é aplicado.

% A entrada dessa função é o nome da imagem de entrada e a saída é uma
% imagem binária resultante da segmentação de pele por meio de um
% classificador de pixels em RGB

```
function [imagem] = segmentarPele2(nomeImagem)
```

% Lê a imagem de entrada

```
imagem = imread(strcat(nomeImagem, '.jpg'));
```

```
linhasImagem = size(imagem, 1);
```

```
colunasImagem = size(imagem, 2);
```

% Converte os valores em RGB de cada pixel para valores no intervalo [0,1]

```
R = double(imagem(:, :, 1))/255;
```

```
G = double(imagem(:, :, 2))/255;
```

```
B = double(imagem(:, :, 3))/255;
```

% Converte os valores em RGB dos pixels para valores no espaço YCbCr.

% Todos os valores de luminância são igualados à 0.7.

```
Y = 0.7*(ones(linhasImagem, colunasImagem));
```

```
U = -0.14713*R - 0.28886*G + 0.436*B;
```

```
V = 0.615*R - 0.51498*G - 0.10001*B;
```

% Os valores dos pixels são, então, novamente convertidos em valores no
% espaço RGB.

```
imagem(:, :, 1) = (Y + 1.13983*V)*255;
```

```
imagem(:, :, 2) = (Y -0.39465*U - 0.58060*V)*255;
```

```
imagem(:, :, 3) = (Y + 2.03211*U)*255;
```

```
imwrite(imagem, strcat(nomeImagem, '_pele.jpg'));
```

% Executa a segmentação de pele por meio de um classificador de pixels em
% RGB.

```
for i=1:linhasImagem
```

```
    for j=1:colunasImagem
```

```
        ehPele = false;
```

```
        if (imagem(i,j,1) > 95 && imagem(i,j,2) > 40 && imagem(i,j,3) > 20)
```

```
            if (max([imagem(i,j,1), imagem(i,j,2), imagem(i,j,3)]) -  
                min([imagem(i,j,1), imagem(i,j,2), imagem(i,j,3)])) > 15
```

```
                if abs(imagem(i,j,1) -imagem(i,j,2)) > 15
```

```
                    if (imagem(i,j,1) > imagem(i,j,2) && imagem(i,j,1) >  
                        imagem(i,j,3))
```

```
                        ehPele = true;
```

```
                    % Os pixels classificados como pertencentes à pele  
                    % recebem o valor 255, correspondente à cor branca.
```

```
                    imagem(i,j,1) = 255;
```

```
                    imagem(i,j,2) = 255;
```

```
                    imagem(i,j,3) = 255;
```

```
                end
```

```
            end
```

```
        end
```

```
    end
```

```
% Os pixels que não são classificados como pertencentes à pele
% recebem o valor zero, correspondente ao preto.
if ehPele == false
    imagem(i,j,1) = 0;
    imagem(i,j,2) = 0;
    imagem(i,j,3) = 0;
end
end
end

% Armazena o resultado em uma imagem binária
imwrite(imagem, strcat(nomeImagem, '_skinBW.jpg'));
end
```

c) Código do Algoritmo de Localização de Face III

```

% A entrada dessa função é o nome da imagem de entrada e a saída é uma
% imagem colorida com a face delimitada.
function [faceDelimitada] = detectarFace3(nomeImagem)

% Lê a imagem de entrada
imagem = imread(strcat(nomeImagem, '.jpg'));
linhasImagem = size(imagem, 1);
colunasImagem = size(imagem, 2);

% A função segmentarPele3 retorna uma matriz binária resultante da
% segmentação de pele por meio de um classificador de pixels em RGB.
matrizPeleSegmentada = segmentarPele3(nomeImagem);

linhasFace = 200;
colunasFace = 120;

faceX = 1;
faceY = 1;
menorSoma = inf;

% Procura um retângulo com uma área de 200x120 pixels com o maior
% percentual de pixels brancos, ou seja, pixels que foram classificados
% como pertencentes à região da pele.
for i=1:linhasImagem-linhasFace
    for j=1:colunasImagem-colunasFace
        recorte = matrizPeleSegmentada(i:i+linhasFace, j:j+colunasFace);
        if (sum(sum(recorte)) < menorSoma)
            faceX = i;
            faceY = j;
            menorSoma = sum(sum(recorte));
        end
    end
end

faceDelimitada = imagem(faceX:faceX+linhasFace, faceY:faceY+colunasFace, :);
imwrite(faceDelimitada, strcat(nomeImagem, '_faceRecorte.jpg'));

```



```

% A entrada dessa função é o nome da imagem de entrada e a saída é uma
% imagem binária resultante da segmentação de pele por meio de um
% classificador de pixels em RGB
function [imagem] = segmentarPele2(nomeImagem)
imagem = imread(strcat(nomeImagem, '.jpg')); % Lê a imagem de entrada
linhasImagem = size(imagem, 1);
colunasImagem = size(imagem, 2);

% Converte os valores em RBG de cada pixels para valores no intervalo [0,1]
R = double(imagem(:, :, 1))/255;
G = double(imagem(:, :, 2))/255;
B = double(imagem(:, :, 3))/255;

% Converte os valores em RBG dos pixels para valores no espaço YCbCr.
% Todos os valores de luminância são igualados à 0.5. Essa é a única
% diferença entre essa função e a função de segmentação utilizada no
% Algoritmo II.
Y = 0.5*(ones(linhasImagem, colunasImagem));
U = -0.14713*R - 0.28886*G + 0.436*B;
V = 0.615*R - 0.51498*G - 0.10001*B;

% Os valores dos pixels são, então, novamente convertidos em valores no
% espaço RGB.
imagem(:, :, 1) = (Y + 1.13983*V)*255;
imagem(:, :, 2) = (Y -0.39465*U - 0.58060*V)*255;
imagem(:, :, 3) = (Y + 2.03211*U)*255;
imwrite(imagem, strcat(nomeImagem, '_pele.jpg'));

% Executa a segmentação de pele por meio de um classificador de pixels em
% RGB.
for i=1:linhasImagem
    for j=1:colunasImagem
        ehPele = false;
        if (imagem(i,j,1) > 95 && imagem(i,j,2) > 40 && imagem(i,j,3) > 20)
            if (max([imagem(i,j,1), imagem(i,j,2), imagem(i,j,3)] -
                min([imagem(i,j,1), imagem(i,j,2), imagem(i,j,3)])) > 15
                && abs(imagem(i,j,1) -imagem(i,j,2)) > 15
                && (imagem(i,j,1) > imagem(i,j,2) && imagem(i,j,1) >
                    imagem(i,j,3)))
                ehPele = true;
                % Os pixels classificados como pertencentes à pele
                % recebem o valor 255, correspondente à cor branca.
                imagem(i,j,1) = 255;
                imagem(i,j,2) = 255;
                imagem(i,j,3) = 255;
            end
        end
    end
    % Os pixels que não são classificados como pertencentes à pele
    % recebem o valor zero, correspondente ao preto.
    if ehPele == false
        imagem(i,j,1) = 0;
        imagem(i,j,2) = 0;
        imagem(i,j,3) = 0;
    end
end
end
% Armazena o resultado em uma imagem binária
imwrite(imagem, strcat(nomeImagem, '_skinBW.jpg'));
end

```

d) Código do Algoritmo de Localização de Face IV

```

% A função retorna a localização da face na imagem
function[faceX,faceY,linhasFace,colunasFace]=detectarFace4(nomeImagem)

imagem = imread(strcat(nomeImagem, '.jpg'));
linhasImagem = size(imagem, 1);
colunasImagem = size(imagem, 2);

% Aplica a rede neural à todos os pixels da imagem de entrada para a
% segmentação de pele
matrizPeleSegmentada = aplicarRedeNeural2(nomeImagem);

% Inicialmente, considera-se um retângulo com 200 linhas e 120 colunas
% para delimitar a face.
linhasFace = 200;
colunasFace = 120;

% Procura o retângulo com o maior valor para a soma dos seus pixels.
faceX = 1;
faceY = 1;
maiorSoma = 0;
for i=1:linhasImagem-linhasFace
    for j=1:colunasImagem-colunasFace
        recorte= matrizPeleSegmentada(i:i+linhasFace,j:j+colunasFace);
        if (sum(sum(recorte)) > maiorSoma)
            faceX = i;
            faceY = j;
            maiorSoma = sum(sum(recorte));
        end
    end
end

media = mean(mean(matrizPeleSegmentada(faceX:faceX+linhasFace,
faceY:faceY+colunasFace, :)));

% Delimita ainda mais o rosto à esquerda.
% Percorre o retângulo da esquerda para a direita eliminando as
% colunas nas quais a soma dos pixels seja inferior à um valor de
% corte.
for j=faceY:faceY + colunasFace/2
    if (sum(matrizPeleSegmentada(faceX:faceX+linhasFace, j, 1)) < ...
        0.7*media*(linhasFace))
        faceY = faceY+1;
        colunasFace = colunasFace-1;
    else
        break;
    end
end

% Delimita ainda mais o rosto à direita
% Percorre o retângulo da direita para a esquerda eliminando as
% colunas nas quais a soma dos pixels seja inferior à um valor de
% corte.
for j=faceY+colunasFace:-1:faceY+colunasFace/2
    if (sum(matrizPeleSegmentada(faceX:faceX+linhasFace, j, 1)) < ...
        0.7*media*(linhasFace))
        colunasFace = colunasFace-1;
    else

```

```

        break;
    end
end

% Delimita o rosto em cima
% Percorre o retângulo de cima para baixo eliminando as linhas nas
% quais a soma dos pixels seja inferior à um valor de corte.
for j=faceX:faceX + linhasFace/2
    if (sum(matrizPeleSegmentada(j, faceY:faceY+colunasFace, 1)) < ...
        0.7*media*(colunasFace))
        faceX = faceX+1;
        linhasFace = linhasFace - 1;
    else
        break;
    end
end

% Delimita o rosto em baixo
% Percorre o retângulo de baixo para cima eliminando as linhas nas
% quais a soma dos pixels seja inferior à um valor de corte.
for j=faceX + linhasFace:-1: faceX + linhasFace/2
    if (sum(matrizPeleSegmentada(j, faceY:faceY+colunasFace, 1)) <
        0.7*media*(colunasFace))
        linhasFace = linhasFace - 1;
    else
        break;
    end
end

% Cria uma nova imagem com apenas o 'rosto' detectado na imagem
% original
imagemColorida = imagem(faceX:faceX+linhasFace, ...
                        faceY:faceY+colunasFace, :);
colunasImagem = size(imagemColorida,2);
escala = 100/colunasImagem;
imagemColorida = imresize(imagemColorida, escala);

imwrite(imagemColorida, strcat(nomeImagem, ...
    '_faceRecorte4Delimitado.jpg'));
imwrite(matrizPeleSegmentada(faceX:faceX+linhasFace, ...
    faceY:faceY+colunasFace, :), strcat(nomeImagem, ...
    '_faceRecorte4DelimitadoBW.jpg'));
end

```

```

% Esta função retorna uma matriz em escala de cinza resultante da aplicação
% da rede neural na imagem 'nomeImagem'. Quando mais próximos da cor
% branca, maior a probabilidade de um pixel pertencer à região da pele
function [matrizEscalaDeCinza] = aplicarRedeNeural2(nomeImagem)

```

```

% net6 é a rede treinada para segmentar a pele em imagens coloridas
load net6;
imagem = imread(strcat(nomeImagem, '.jpg'));
linhas = size(imagem, 1);
colunas = size(imagem, 2);
matriz1 = reshape(imagem(:,:,1), [size(imagem,1)*size(imagem, 2), 1]);
matriz2 = reshape(imagem(:,:,2), [size(imagem,1)*size(imagem, 2), 1]);
matriz3 = reshape(imagem(:,:,3), [size(imagem,1)*size(imagem, 2), 1]);
matrizEntrada = [matriz1 matriz2 matriz3];

% A função sim do Matlab aplica uma rede neural à um conjunto de dados
resultado = sim (net6 , double(matrizEntrada)');
matrizEscalaDeCinza = abs(resultado);
matrizEscalaDeCinza = reshape(matrizEscalaDeCinza, linhas, colunas);
imwrite(matrizEscalaDeCinza, strcat(nomeImagem, '_RN2.jpg'));

end

```

```

% Função utilizada para treinar a rede neural que será utilizada na
% segmentação de pele
function [] = inicializarETreinarRedeNeural( )

% A matriz de entrada corresponde ao conjunto de todos os pixels das 40
% imagens utilizadas no treinamento da rede.
matrizEntrada = dlmread('matrizEntrada.txt');
% A matriz de saída contém os valores de saída esperados para o conjunto de
% entrada
matrizSaida = dlmread('matrizSaida.txt');

% Cria uma rede do tipo Feedforward com duas camadas ocultas, uma com cinco
% e outra com três neurônios.
net6 = feedforwardnet([5 3], 'trainbr');
net6.divideParam.trainRatio = 1; % training set [%]
net6.divideParam.valRatio = 0; % validation set [%]
net6.divideParam.testRatio = 0; % test set [%]
net6.trainParam.epochs = 100; % número de épocas de treinamento
tamanhoCadaImagem = 132608;
net6 = train(net6,matrizEntrada',matrizSaida');
view(net6)
save('net6.mat', 'net6'); % armazena a rede treinada em 'net6.mat'

end

```

APÊNDICE C – Códigos do Algoritmos de Localização de Olhos em Malab

a) Algoritmo de Localização dos Olhos I

```

% Esta função tem como entrada uma imagem com uma face delimitada e a saída
% é a imagem de entrada com uma linha vermelha na região dos olhos.
function [imagemLinhaOlhos] = detectarOlhos1(nomeImagem)

% Lê a imagem de entrada
imagemLinhaOlhos = imread(strcat(nomeImagem, '.jpg'));

% Encontra as linhas e colunas da imagem
linhasImagem = size(imagemLinhaOlhos, 1);
colunasImagem = size(imagemLinhaOlhos, 2);

% Calcula a soma dos ângulos de todos os pixels vizinhos horizontalmente
vetorSomaDosAngulos = [];
for i=1:(linhasImagem/2)
    somaDosAngulos = 0;
    for j=1:colunasImagem-1
        anguloEntreVetores(imagemLinhaOlhos(i,j,:),
                            imagemLinhaOlhos(i,j+1,:));
        somaDosAngulos = somaDosAngulos +
            anguloEntreVetores(imagemLinhaOlhos(i,j,:),
                                imagemLinhaOlhos(i,j+1,:));
    end
    somaDosAngulos = somaDosAngulos/(colunasImagem-1);
    vetorSomaDosAngulos = [vetorSomaDosAngulos somaDosAngulos];
end

% Percorre a primeira metade horizontal da imagem de forma a encontrar
% cinco linhas consecutivas com a maior soma total dos ângulos entre pixels
% vizinhos.
linhaDosOlhos = 1;
maxSomaDosAngulos = 0;
for i = 1:(linhasImagem/2)-4
    somaDeCincoAngulos =vetorSomaDosAngulos(i) + vetorSomaDosAngulos(i+1) +
        vetorSomaDosAngulos(i+2)+ vetorSomaDosAngulos(i+3)+
        vetorSomaDosAngulos(i+4);
    if somaDeCincoAngulos > maxSomaDosAngulos
        linhaDosOlhos = i+2;
        maxSomaDosAngulos = somaDeCincoAngulos;
    end
end

% Desenha uma linha vermelha na região dos olhos
imagemLinhaOlhos(linhaDosOlhos-1:linhaDosOlhos+1, :, 1) = 255;
imagemLinhaOlhos(linhaDosOlhos-1:linhaDosOlhos+1, :, 2) = 0;
imagemLinhaOlhos(linhaDosOlhos-1:linhaDosOlhos+1, :, 3) = 0;

% Armazena a imagem resultante
inwrite(imagemLinhaOlhos, strcat(nomeImagem, '_linhaDosOlhos.jpg'));

end

```

```
% Essa função calcula o ângulo (em graus) entre dois vetores no espaço.
% Os vetores são representados por matrizes.
% As duas matrizes de entrada tem dimensões 1x1x3.
function [resultado] = anguloEntreVetores(matriz1, matriz2)
cossenoEntreVetores = produtoEscalar(matriz1,
matriz2)/(moduloVetor(matriz1)*moduloVetor(matriz2));
if (cossenoEntreVetores>1)
    cossenoEntreVetores = 1;
end
resultado = (acos(double(cossenoEntreVetores))/pi)*180;
end

% Calcula o produto escalar de dois vetores no espaço.
% Os vetores são representados por matrizes.
% As duas matrizes de entrada tem dimensões 1x1x3.

function [resultado] = produtoEscalar(matriz1, matriz2)
u = double([matriz1(1,1,1); matriz1(1,1,2); matriz1(1,1,3)]);
v = double([matriz2(1,1,1); matriz2(1,1,2); matriz2(1,1,3)]);
resultado = u'*v;
end

% Calcula o módulo de um vetor no espaço.
% Os vetores são representados por matrizes.
% As duas matrizes de entrada tem dimensões 1x1x3.
function [resultado] = moduloVetor(matriz)
u = double([matriz(1,1,1); matriz(1,1,2); matriz(1,1,3)]);
resultado = sqrt(u'*u);
end
```

b) Algoritmo de Localização dos Olhos II

```

% Essa função tem como entrada o nome de uma imagem com uma face delimitada
% e retorna a imagem de entrada com as regiões dos olhos delimitadas por
% retângulos com bordas azuis.
function [imagemOlhos] = detectarOlhos4(nomeimagemOlhos)

% Lê a imagem de entrada
imagemOlhos = imread(strcat(nomeimagemOlhos, '.jpg'));
linhasimagemOlhos = size(imagemOlhos, 1);
colunasimagemOlhos = size(imagemOlhos, 2);

% Converte os valores em RGB para valores no intervalo [0,1]
R = double(imagemOlhos(:, :, 1))/255;
G = double(imagemOlhos(:, :, 2))/255;
B = double(imagemOlhos(:, :, 3))/255;

% Encontra a matriz Cb da imagem, que será a única a ser utilizada pelo
% algoritmo
Cb = -0.14713*R - 0.28886*G + 0.436*B;

% As regiões dos olhos serão delimitadas por retângulos de 20x10 pixels
linhasOlho = 10;
colunasOlho = 20;

% Procura a região do olho esquerdo
olhoEsquerdoX = 1;
olhoEsquerdoY = 1;
maiorSoma = sum(sum(Cb(1:1+linhasOlho, 1:1+colunasOlho)));

% Percorre a metade vertical esquerda da face de modo a procurar um
% retângulo com uma área de 20x10 pixels com o maior valor para a soma dos
% valores de Cb de cada pixel
for i=1:int32(linhasimagemOlhos/2)-linhasOlho
    for j=1:int32(colunasimagemOlhos/2)-colunasOlho
        recorteFace = Cb(i:i+linhasOlho, j:j+colunasOlho);
        somaRecorte = sum(sum(recorteFace));
        if (somaRecorte > maiorSoma)
            olhoEsquerdoX = i;
            olhoEsquerdoY = j;
            maiorSoma = somaRecorte;
        end
    end
end

% Desenha na imagemOlhos um retângulo azul na posição dos olho esquerdo
shapeInserter =
vision.ShapeInserter('Shape','Rectangles','BorderColor','Custom',
'CustomBorderColor', uint8([0 0 255]));
retangulo = int32([olhoEsquerdoX olhoEsquerdoY linhasOlho colunasOlho]);
imagemOlhos = step(shapeInserter, imagemOlhos, retangulo);
retangulo = int32([olhoEsquerdoX-1 olhoEsquerdoY-1 linhasOlho+2
colunasOlho+2]);
imagemOlhos = step(shapeInserter, imagemOlhos, retangulo);

% Procura a região do olhos direito
olhoDireitoX = 1;

```

```

olhoDireitoY = int32(colunasimagemOlhos/2);
maiorSoma = sum(sum(Cb(1:1+linhasOlho,
int32(colunasimagemOlhos/2):int32(colunasimagemOlhos/2)+colunasOlho)));

% Percorre a metade vertical direita da face de modo a procurar um
% retângulo com uma área de 20x10 pixels com o maior valor para a soma dos
% valores de Cb de cada pixel
for i=1:int32(linhasimagemOlhos/2)-linhasOlho
    for j=int32(colunasimagemOlhos/2):colunasimagemOlhos-colunasOlho
        recorteFace = Cb(i:i+linhasOlho, j:j+colunasOlho);
        somaRecorte = sum(sum(recorteFace));
        if (somaRecorte > maiorSoma)
            olhoDireitoX = i;
            olhoDireitoY = j;
            maiorSoma = somaRecorte;
        end
    end
end

% Desenha na imagemOlhos um retângulo azul na posição dos olho direito
shapeInserter =
vision.ShapeInserter('Shape','Rectangles','BorderColor','Custom',
'CustomBorderColor', uint8([0 0 255]));
retangulo = int32([olhoDireitoX olhoDireitoY linhasOlho colunasOlho]);
imagemOlhos = step(shapeInserter, imagemOlhos, retangulo);
retangulo = int32([olhoDireitoX-1 olhoDireitoY-1 linhasOlho+2
colunasOlho+2]);
imagemOlhos = step(shapeInserter, imagemOlhos, retangulo);

% Armazena o resultado em um arquivo de imagem
imwrite(imagemOlhos, strcat(nomeimagemOlhos, '_olhos.jpg'));
end

```


c) Algoritmo de Localização dos Olhos III

```

% A função retorna a posição dos dois olhos encontrados em
% 'nomeImagem'
function [olhoEsquerdoX, olhoEsquerdoY, olhoDireitoX, olhoDireitoY] =
detectarOlhos3(nomeImagem)

% Abre a imagem
imagem = imread(strcat(nomeImagem, '.jpg'));
linhasImagem = size(imagem, 1);
colunasImagem = size(imagem, 2);

R = double(imagem(:, :, 1))/255;
G = double(imagem(:, :, 2))/255;
B = double(imagem(:, :, 3))/255;

Y = 0.299*R + 0.587*G + 0.114*B;
U = -0.14713*R - 0.28886*G + 0.436*B;
V = 0.615*R - 0.51498*G - 0.10001*B;

% Calcula a matriz com todos os ângulos entre os pixels vizinhos
% horizontais da imagem
matrizAngulos = zeros(int32(linhasImagem/2), colunasImagem-1);
for i=1:int32(linhasImagem/2)
    for j=1:colunasImagem-1
        matrizAngulos(i,j) = anguloEntreVetores(imagem(i,j,:),
imagem(i,j+1,:));
    end
end

% Define as dimensões do retângulo da área dos olhos
linhasOlho = 5;
colunasOlho = 30;

linhaInicial = 1 + int32((linhasImagem/2)*0.3);
linhaFinal = int32(linhasImagem/2) - int32((linhasImagem/2)*0.1);

% Encontra o olho esquerdo
olhoEsquerdoX = 1;
olhoEsquerdoY = 1;
maiorSomaU = sum(sum(U(1:1+linhasOlho, 1:1+colunasOlho)));
maiorSomaAngulos = 0;
for i=linhaInicial:linhaFinal-linhasOlho
    for j=1:int32(colunasImagem/2)-colunasOlho
        recorteFaceU = U(i:i+linhasOlho, j:j+colunasOlho);
        somaRecorteU = sum(sum(recorteFaceU));

        somaAngulos = sum(sum(matrizAngulos(i:i+(linhasOlho-2), ...
j:j+(colunasOlho-2))));

        if ((somaRecorteU > maiorSomaU && somaAngulos > ...
0.9*maiorSomaAngulos) || (somaAngulos > maiorSomaAngulos))
            olhoEsquerdoX = i;
            olhoEsquerdoY = j;
            maiorSomaU = somaRecorteU;
            maiorSomaAngulos = somaAngulos;
        end
    end
end
end

```

```

end

% Desenha na imagem um retângulo azul na posição dos olho esquerdo
shapeInserter =
vision.ShapeInserter('Shape','Rectangles','BorderColor','Custom', ...
                    'CustomBorderColor', uint8([0 0 255]));
retangulo = int32([olhoEsquerdoX olhoEsquerdoY linhasOlho ...
                 colunasOlho]);
imagem = step(shapeInserter, imagem, retangulo);

% Encontra o olho direito
olhoDireitoX = 1;
olhoDireitoY = 1;
maiorSomaU = sum(sum(U(1:1+linhasOlho,
int32(colunasImagem/2):int32(colunasImagem/2)+colunasOlho)));
maiorSomaAngulos = 0;
for i=linhaInicial:linhaFinal-linhasOlho
    for j=int32(colunasImagem/2):colunasImagem-colunasOlho
        recorteFaceU = U(i:i+linhasOlho, j:j+colunasOlho);
        somaRecorteU = sum(sum(recorteFaceU));

        somaAngulos = sum(sum(matrizAngulos(i:i+(linhasOlho-2), ...
                                           j:j+(colunasOlho-2))));

        if ((somaRecorteU > maiorSomaU && somaAngulos > ...
            0.9*maiorSomaAngulos) || (somaAngulos > maiorSomaAngulos))
            olhoDireitoX = i;
            olhoDireitoY = j;
            maiorSomaU = somaRecorteU;
            maiorSomaAngulos = somaAngulos;
        end
    end
end
end

% Desenha na imagem um retângulo azul na posição dos olho direito
shapeInserter =
vision.ShapeInserter('Shape','Rectangles','BorderColor','Custom', ...
                    'CustomBorderColor', uint8([0 0 255]));
retangulo = int32([olhoDireitoX olhoDireitoY linhasOlho colunasOlho]);
imagem = step(shapeInserter, imagem, retangulo);

% Desenha na imagem as linhas que delimitam o espaço de busca do algoritmo
for i=1:colunasImagem
    imagem(linhaInicial, i, 1) = 255;
    imagem(linhaInicial, i, 2) = 0;
    imagem(linhaInicial, i, 3) = 0;

    imagem(linhaFinal, i, 1) = 255;
    imagem(linhaFinal, i, 2) = 0;
    imagem(linhaFinal, i, 3) = 0;

end

% Salva a imagem com os olhos detectados
imwrite(imagem, strcat(nomeImagem, '_olhos.jpg'));
end

```

APÊNDICE D – Código do Algoritmo de Alinhamento da Face

```

% Essa função tem como entrada uma imagem com a face delimitada, a imagem
% original da face delimitada e o número da imagem no banco de dados.
% A saída dessa função é a face alinhada e delimitada acima dos olhos.
function [imagemAlinhada] = alinharDelimitarFace(nomeImagem,
                                                nomeImagemOriginal,
                                                numeroImagem)

% Lê a imagem com a face delimitada
imagem = imread(strcat(nomeImagem, '.jpg'));

% Encontra a posição dos olhos a partir do resultado do Algoritmo de
% Localização de Olhos III
[olhoEsquerdoX, olhoEsquerdoY, olhoDireitoX, olhoDireitoY] =
    detectarOlhos3(nomeImagem);

linhasOlho = 5;
colunasOlho = 30;

% Coordenadas dos olhos no plano cartesiano
x1 = olhoEsquerdoY + (linhasOlho/2);
x2 = olhoDireitoY + (linhasOlho/2);
y1 = olhoEsquerdoX + (colunasOlho/2);
y2 = olhoDireitoX + (colunasOlho/2);

% Calcula o ângulo da reta que passa pelos pontos centrais das regiões dos
% olhos
inclinacaoDaReta = double(y1-y2)/double(x1-x2);
angulo = (atan(double(inclinacaoDaReta))/pi)*180;

% Abre o arquivo no qual foram armazenadas as posições das faces em cada
% imagem do banco de dados a partir do Algoritmo de Localização de Face IV
matrizPosicoes = dlmread('matrizResultadosTeste.txt');
faceX = matrizPosicoes(numeroImagem, 1);
faceY = matrizPosicoes(numeroImagem, 2);
linhasFace = matrizPosicoes(numeroImagem, 3);
colunasFace = matrizPosicoes(numeroImagem, 4);

% Lê a imagem original do banco de dados
imagemOriginal = imread(strcat(nomeImagemOriginal, '.jpg'));

% Rotaciona a imagem original de acordo com o ângulo encontrado
% Considerou-se apenas os ângulos entre 5° e 15°, pois, caso o ângulo seja
% menor do que 5 o resultado do alinhamento será sutil, e caso o ângulo
% seja maior do que 15, há uma probabilidade maior de que o resultado da
% localização dos olhos esteja errado.
if (abs(angulo) > 5 && abs(angulo)<15)
    imagemOriginalRotacionada = imrotate(imagemOriginal, angulo);
else
    imagemOriginalRotacionada = imagemOriginal;
end

% Armazena a imagem original rotacionada.
imwrite(imagemOriginalRotacionada, strcat(nomeImagem, '_rotacionado.jpg'));
faceX = faceX + int32(((size(imagemOriginalRotacionada,1)) -
    (size(imagemOriginal,1)))/2);
faceY = faceY + int32(((size(imagemOriginalRotacionada,2)) -
    (size(imagemOriginal,2)))/2);

```

```
% Delimita a imagem original rotacionada a partir da posição da face
% encontrada em 'matrizPosicoes'
imagemResultante = imagemOriginalRotacionada(faceX:faceX+linhasFace,
                                              faceY:faceY+colunasFace, :);

escala = 100/(colunasFace+1);
imagemResultante = imresize(imagemResultante, escala);
imwrite(imagemResultante, strcat(nomeImagem, '_alinhado.jpg'));

% Delimita a face resultante
linhaDosOlhos = abs(min([olhoEsquerdoX, olhoDireitoX]) - 10);

% Armazena a imagem alinhada e delimitada acima dos olhos
imagemAlinhada = imagemResultante(linhaDosOlhos:size(imagemResultante,1),
                                   :, :);
imwrite(imagemAlinhada, strcat(nomeImagem, '_alinhadoCortado.jpg'));

end
```

APÊNDICE E – Código do Algoritmo de Detecção de Lábios

```

% O resultado da função é a imagem 'nomeImagem com os lábios delimitados
% por um retângulo azul.
function [] = detectarBoca(nomeImagem)

imagem = imread(strcat(nomeImagem, '.jpg'));
linhasImagem = size(imagem, 1);
colunasImagem = size(imagem, 2);

R = double(imagem(:, :, 1))/255;
G = double(imagem(:, :, 2))/255;
B = double(imagem(:, :, 3))/255;

Cb = -0.14713*R - 0.28886*G + 0.436*B;
Cr = 0.615*R - 0.51498*G - 0.10001*B;

linhasBoca = 10;
colunasBoca = 50;

% Encontra a área lábios considerando um retângulo com 10 de altura e 50 de
% comprimento com o maior valor para a soma das componentes Cb e Cr
bocaX = 1;
bocaY = 1;
maiorSoma = 0;
for i=int32(1.2*linhasImagem/3):linhasImagem-linhasBoca
    for j=1:colunasImagem-colunasBoca
        recorteFaceCb = Cb(i:i+linhasBoca, j:j+colunasBoca);
        recorteFaceCr = Cr(i:i+linhasBoca, j:j+colunasBoca);
        somaRecorte = recorteFaceCb+recorteFaceCr;
        if (sum(sum(somaRecorte)) > maiorSoma)
            bocaX = i;
            bocaY = j;
            maiorSoma = sum(sum(somaRecorte));
        end
    end
end

imagemBoca = imagem;

% Desenha a um retângulo azul na área dos lábios
shapeInserter =
vision.ShapeInserter('Shape','Rectangles','BorderColor','Custom',...
    'CustomBorderColor', uint8([0 0 255]));
retangulo = int32([bocaX bocaY linhasBoca colunasBoca]);
imagem = step(shapeInserter, imagem, retangulo);

imwrite(imagemBoca, strcat(nomeImagem, '_Boca.jpg'));
limiteInferior = bocaX + 15;
if (limiteInferior > linhasImagem)
    limiteInferior = linhasImagem;
end
imagemDelimitadaAbaixo = imagem(1:limiteInferior,:,:);
imwrite(imagemDelimitadaAbaixo, strcat(nomeImagem, ...
    '_BocaDelimitada.jpg'));
end

```

APÊNDICE F – Código do Algoritmos de Alteração da Iluminação

```

% Essa função tem como entrada uma imagem de 100x100 e a saída é a imagem
% de entrada após a alteração da iluminação.
function [imagem] = equilibrarIluminacao( nomeImagem )

% Lê a imagem de entrada
imagem = imread(strcat(nomeImagem, '.jpg'));

% Converte as matrizes de cores no espaço RGB para matrizes com valores no
% intervalo [0,1]
R = double(imagem(:, :, 1))/255;
G = double(imagem(:, :, 2))/255;
B = double(imagem(:, :, 3))/255;

% Encontra as componentes de cores do no espaço YCbCr
Y = (0.299*R + 0.587*G + 0.114*B);
U = -0.14713*R - 0.28886*G + 0.436*B;
V = 0.615*R - 0.51498*G - 0.10001*B;

% Lê a matriz resultante da iluminação média de 8 imagens do banco de dados
media = dlmread('media2.txt');
Y = Y*0.5 + media*0.5;

% Converte as matrizes encontradas novamente para o espaço RGB
R = (Y + 1.13983*V)*255;
G = (Y -0.39465*U - 0.58060*V)*255;
B = (Y + 2.03211*U)*255;

imagem(:, :,1) = R;
imagem(:, :,2) = G;
imagem(:, :,3) = B;

% Armazena a imagem resultante
imwrite(imagem, strcat(nomeImagem, 'Y.jpg'));

end

```

APÊNDICE G – Código do Algoritmo Reconhecimento de Face

```

% A função identifica os indivíduos calculando o desvio padrão da
% diferença entre a face de entrada 'nomeImagem' e todas as médias
% calculadas para cada indivíduo. A função retorna os dois menores
% valores para o desvio padrão e o número das faces correspondentes.
function [numeroDaFace1, numeroDaFace2, menorValor, segundoMenorValor] =
reconhecerFace(nomeImagem)

face = imread(strcat(nomeImagem, '.jpg'));
R = double(face(:, :, 1))/255;
G = double(face(:, :, 2))/255;
B = double(face(:, :, 3))/255;
Y = (0.299*R + 0.587*G + 0.114*B);
U = -0.14713*R - 0.28886*G + 0.436*B;
V = 0.615*R - 0.51498*G - 0.10001*B;
vetorDesvioPadrao = zeros(28,1);

% Calcula o desvio padrão corespondente aos 28 individuos do banco de
% dados
for i=1:28
    faceMedia = imread(strcat('Faces3\media_', int2str(i), '.jpg'));
    mediaR = double(faceMedia(:, :, 1))/255;
    mediaG = double(faceMedia(:, :, 2))/255;
    mediaB = double(faceMedia(:, :, 3))/255;
    mediaY = (0.299*mediaR + 0.587*mediaG + 0.114*mediaB);
    mediaU = -0.14713*mediaR - 0.28886*mediaG + 0.436*mediaB;
    mediaV = 0.615*mediaR - 0.51498*mediaG - 0.10001*mediaB;
    diferencaY = Y-mediaY;
    diferencaU = U-mediaU;
    diferencaV = V-mediaV;
    vetorDiferencaY = reshape(diferencaY, ...
        [size(diferencaY,1)*size(diferencaY, 2),1]);
    vetorDiferencaU = reshape(diferencaU, ...
        [size(diferencaU,1)*size(diferencaU, 2), 1]);
    vetorDiferencaV = reshape(diferencaV, ...
        [size(diferencaV,1)*size(diferencaV, 2), 1]);
    vetorDesvioPadrao(i) = std(vetorDiferencaU)+std(vetorDiferencaV);
end

% Encontra os dois menores valores para o desvio padrão.
% Considera-se que, quanto menor for o desvio padrão, maior será a
% probabilidade de a face de entrada pertencer ao indivíduo correpondente
% àquele desvio padrão.
menorValor = min(vetorDesvioPadrao);
numeroDaFace1 = find(vetorDesvioPadrao == menorValor);
vetorDesvioPadrao(numeroDaFace1) = Inf;
segundoMenorValor = min(vetorDesvioPadrao);
numeroDaFace2 = find(vetorDesvioPadrao == segundoMenorValor);

end

```