UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

E-MOTIV: PROTÓTIPO DE SISTEMA PARA RECONHECIMENTO DE EXPRESSÕES FACIAIS

WILLIAM LEANDER SEEFELD

BLUMENAU
2016

WILLIAM LEANDER SEEFELD

EMOTIV: PROTÓTIPO DE SISTEMA PARA RECONHECIMENTO DE EXPRESSÕES FACIAS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Daniel Theisges dos Santos, Mestre - Orientador

BLUMENAU
2016

EMOTIV: PROTÓTIPO DE SISTEMA PARA RECONHECIMENTO DE EXPRESSÕES FACIAS

Por

WILLIAM LEANDER SEEFELD

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Daniel Theisges dos Santos, Mestre – Orientador, FURB
Membro:	Prof. Roberto Heinzle, Doutor – FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB

Blumenau, 08 de Julho de 2016

Dedico este trabalho à minha família e à prof.ª Maria de Fátima de Oliveira, por desde tão cedo terem me incentivado aos estudos.

AGRADECIMENTOS

À minha família, por me incentivar e criar condições para os estudos desde cedo.

Aos meus amigos e colegas de trabalho, pela compreensão e apoio no desenvolvimento deste trabalho. Em especial, aos meus colegas de faculdade André V. Bampi, Eli T. de Souza, Gustavo Sabel, Maicon M. Gerardi, Reinoldo Krause Júnior e Vivian de L. Panzenhagen, por todos os momentos divididos nesta jornada.

Ao meu orientador, Daniel Theisges dos Santos, por ter acreditado na minha capacidade e ter prestado imenso apoio e incentivo, mesmo nos momentos mais difíceis.

If we crave for some cosmic purpose, then let us find ourselves a worthy goal.

Carl Sagan

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo para reconhecimento de expressões faciais universais. Esta técnica fornece uma forma alternativa de coleta de dados do usuário, podendo servir como um método de entrada para sistemas de informação. Para a detecção de faces e extração de suas características são utilizadas técnicas de Visão Computacional e Processamento de Imagens Digitais, implementados pela biblioteca dlib. A classificação em expressões faciais é realizada através de uma Rede Neural Artificial do tipo *perceptron* de multicamadas. O protótipo desenvolvido foi capaz de reconhecer até duas expressões faciais, além da expressão neutra, com até 89% de precisão, demonstrando a viabilidade das técnicas empregadas.

Palavras-chave: Visão computacional. Processamento de imagens. Redes neurais artificiais. Expressões faciais.

ABSTRACT

This work presents the development of a prototype for the recognition of universal facial expressions. This technic provides an alternative way of gathering data from the user, being able for usage as an input way for information systems. For faces detection and extraction of their characteristics technics of Computer Vision and Digital Image Processing are employed, implemented by the dlib library. The classification into facial expressions is performed by an Artificial Neural Network, of the multilayer perceptron kind. The developed prototype was able to recognize up to two facial expressions, besides the neutral one, with up to 89% of precision, showing the feasibility of the employed techniques.

Key-words: Computer vision. Image processing. Artificial neural network. Universal facial expressions.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente os dispositivos computacionais se encontram em estágio de transição para a Computação Ubíqua, fase na qual é esperado que estes aparatos tenham grande presença no ambiente, se integrem facilmente entre si e, frequentemente, passem despercebidos pelos seres humanos (SINDHURI; RAJU, 2013). É esperado que uma quantidade maior destes dispositivos resulte em mais informações disponíveis para tornar o cotidiano humano mais simples, eficiente e seguro.

Para tanto, faz-se necessário o desenvolvimento e aprimoramento de técnicas que permitam aos computadores compreender emoções humanas. Este é um dos objetivos da Computação Afetiva (CA) que, além do reconhecimento, visa dotar os computadores da capacidade de estimular emoções em seres humanos (PICARD, 1997).

A aplicação da CA traz benefícios para diversas áreas. Por exemplo, na área de educação se pode analisar quais partes de uma vídeo-aula os alunos não gostaram ou mostraram confusão. Na indústria automotiva, as fabricantes de carros podem instalar uma câmera na frente do motorista para dizer se o mesmo está perigosamente cansado e propenso a sofrer uma colisão. Na área da saúde, pode-se monitorar as expressões faciais de um paciente e ter um indicativo de que o mesmo apresenta sinais de depressão (GELLER, 2014).

Para o reconhecimento de emoções, pode-se recorrer à análise de expressões faciais. Isto porque, segundo Ekman e Friesen (1969), existe um conjunto de seis emoções básicas universais para as quais as expressões faciais apresentam sinais únicos entre si, porém iguais entre todos os seres humanos, independentemente de sexo e fatores sociais. Este conjunto é composto pelas emoções nojo, medo, alegria, surpresa, tristeza e raiva.

Diante do exposto, este trabalho apresenta o desenvolvimento de um protótipo utilizando técnicas de processamento de imagens e visão computacional para reconhecer e classificar, usando uma rede neural artificial, um subconjunto das expressões de emoções básicas universais em imagens estáticas.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um sistema capaz de reconhecer expressões faciais de emoção em imagens estáticas, usando técnicas de visão computacional e inteligência artificial.

Os objetivos específicos do trabalho são:

a) utilizar técnicas de visão computacional para extração de características faciais;
b) implementar um classificador de emoções utilizando uma rede neural artificial;

c) reconhecer as expressões faciais das emoções neutra, felicidade e tristeza.

1.2 ESTRUTURA

Este trabalho é composto por quatro capítulos, tendo sido o primeiro deles destinado à apresentação da introdução e objetivos do presente trabalho.

O capítulo seguinte aborda conceitos teóricos necessários para o desenvolvimento deste trabalho, iniciando com expressões faciais universais e visão computacional. Em seguida, são apresentadas técnicas de processamento de imagens, seguido pelo conceito de redes neurais artificiais. Por fim, são apresentados os trabalhos correlatos.

O terceiro capítulo descreve o desenvolvimento do protótipo, iniciando pela apresentação dos requisitos, especificação, implementação e operacionalidade da solução. São apresentados, por fim, os resultados obtidos.

O quarto e último capítulo se destina à exposição das conclusões do desenvolvimento deste trabalho, apontando também possíveis extensões do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os assuntos e técnicas utilizados para o desenvolvimento do sistema de classificação de expressões faciais. Na seção 2.1 são apresentadas as expressões faciais universais. Na seção 2.2 são introduzidos conceitos de visão computacional. Na seção 2.3 são apresentadas técnicas de processamento de imagens digitais. Na seção 2.4 são apresentados os principais conceitos de redes neurais artificiais aplicados neste trabalho. Por fim, na seção 2.5 são apresentados trabalhos correlatos.

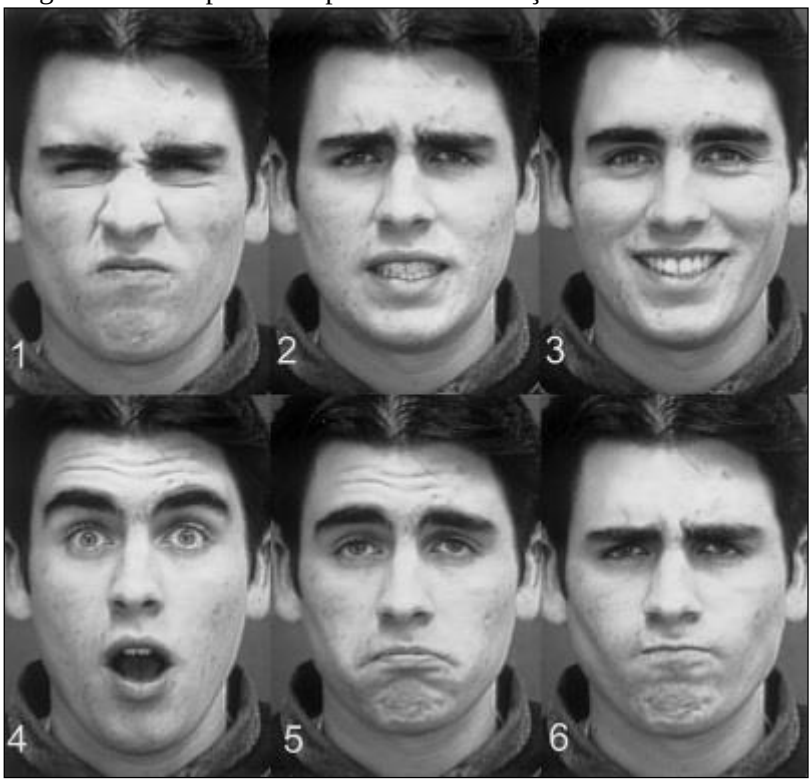
2.1 EXPRESSÕES FACIAIS UNIVERSAIS

De um ponto de vista evolucionário, as emoções básicas possuem grande valor para os primatas, em especial os seres humanos, por estarem relacionadas com tarefas fundamentais da vida. Estas tarefas abrangem desde escapar de situações de perigo para sobreviver até a realização de objetivos individuais. Esta relação se dá pois, para cada ação de impacto na vida de um indivíduo, como nos exemplos citados, o mesmo tende a demonstrar uma emoção específica (EKMAN, 1999a).

Sendo assim, é possível afirmar que o reconhecimento de emoções através de voz, expressões e gestos faz parte da interação social humana, ainda que inconscientemente. Dentre as formas de reconhecimento emocional citadas, o reconhecimento de emoções através de expressões faciais é considerado uma das formas mais primitivas e frequentemente utilizadas pelo homem (EKMAN, 1999b).

Atualmente há consenso sobre um conjunto de seis emoções que são expressas pelas mesmas expressões faciais entre os seres humanos, independentemente de gênero, idade, fatores econômicos, sociais, históricos, etc. (PORTER; BRINKE, 2008). Estas emoções, visíveis na Figura 1, foram propostas por Ekman e Friesen (1969), sendo elas: (1) repulsa, (2) medo, (3) alegria, (4) surpresa, (5) tristeza e (6) raiva.

Figura 1 - Exemplo das expressões das emoções básicas universais



Fonte: Schmidt e Cohn (2001).

É necessário ficar atento, entretanto, que diferentes culturas e contextos podem mascarar a forma como as emoções são expressas pelas chamadas “regras de exibição” (EKMAN; SORENSON; FRIESEN, 1969). Assim, apesar de uma emoção universal se manifestar de forma idêntica assim que é sentida, estas regras logo em seguida suavizam, intensificam ou neutralizam a expressão facial.

O reconhecimento das expressões de cada emoção pode ser feito utilizando o Sistema de Codificação da Ação Facial (*Facial Action Coding System* – FACS), um sistema de codificação baseado em anatomia para registro de mudanças na aparência causadas pela ação de músculos individuais da face. A representação FACS se dá através de Unidades de Ação (*Action Units* – AU), composta pela identificação do músculo e, às vezes, por uma letra que representa a intensidade do movimento (EKMAN; FRIESEN, 1978).

Graças à concepção deste sistema, tornaram-se evidentes as distinções entre as diversas expressões faciais de emoção. Com isso, foi possibilitado também que o reconhecimento das mesmas fosse automatizado por computadores, através de técnicas de visão computacional (SANDBACH et al., 2012).

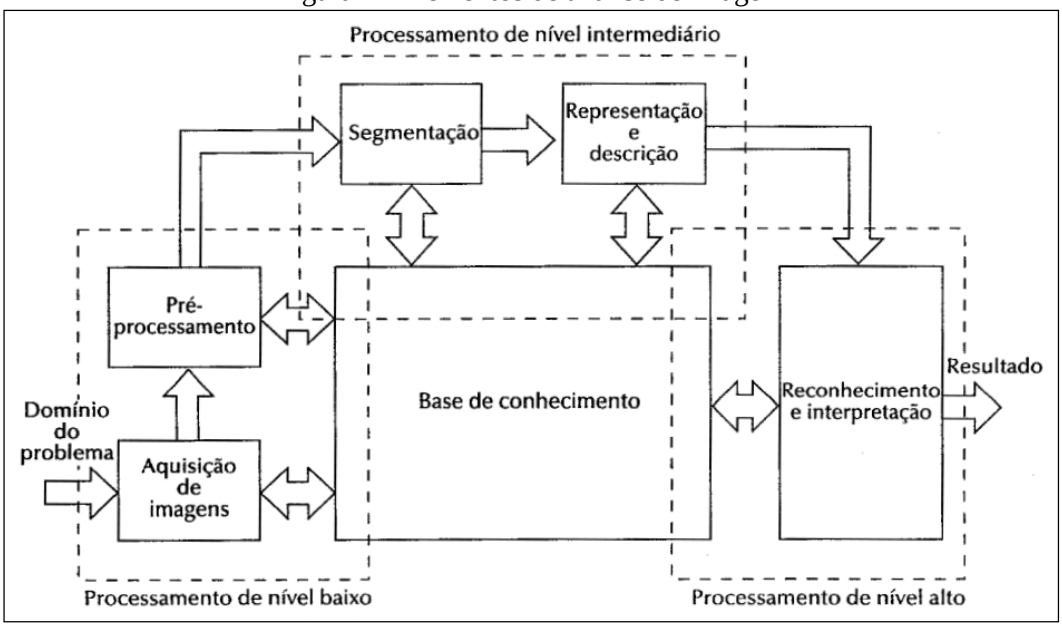
2.2 VISÃO COMPUTACIONAL

O reconhecimento de faces humanas por computadores, assim como outros objetos e características do ambiente, é tratado pela área da visão computacional (SHIRAI, 1987, p. 1). Esta área pode ser considerada a união das áreas de processamento de imagens e de inteligência artificial, com o objetivo de realizar tarefas semelhantes à visão humana (SZELISKI, 2011, p. 731).

Para tanto, é empregada análise de imagens, que é "[...] um processo de descobrimento, de identificação e de entendimento de padrões que sejam relevantes à performance de uma tarefa baseada em imagens" (GONZALEZ; WOODS, 2000, p. 407). Este processo vem sido utilizado desde reconhecimento de caracteres até autenticação por imagem facial. Conforme a Figura 2, Gonzalez e Woods (2000, p. 408) destacam que as etapas do processamento de imagens podem ser agrupadas em três níveis:

a) processamento de baixo nível: neste nível não é requerida inteligência, visto que as etapas tratam apenas do processo de formação da imagem (ex.: captura por uma câmera) e da utilização de funções de baixo nível para compensação, como suavização e redução de ruído;
b) processamento de nível intermediário: estas etapas tratam da segmentação e a descrição da imagem, requerendo algum nível de inteligência para, por exemplo, corrigir uma fronteira segmentada;
c) processamento de alto nível: responsável pelo reconhecimento e interpretação da imagem resultante, sendo o nível que mais requer inteligência.

Figura 2 - Elementos de análise de imagem



Fonte: Gonzalez e Woods (2000, p. 408).

Marques Filho e Vieira Neto (1999, p. 9) exemplificam as etapas destes processos utilizando como domínio do problema o reconhecimento de Código de Endereçamento Postal (CEP) de um lote de envelopes.

No processamento de nível baixo, a primeira etapa consiste na *aquisição de imagens* dos envelopes através de um sensor e um digitalizador e das configurações da captura, como iluminação, resolução da imagem e número de níveis de cinza da imagem digitalizada. Na etapa de *pré-processamento* são corrigidos o brilho e contraste e removidos ou suavizados os *pixels* ruidosos, resultando em uma imagem de melhor qualidade.

O processamento de nível intermediário inicia com a etapa de *segmentação*, que consiste na execução de algoritmos que irão localizar o CEP do restante das informações, posteriormente trabalhando sobre esta subimagem para segmentar cada dígito individualmente. Neste sentido, seria gerada uma subimagem para cada dígito do CEP. Na etapa seguinte, a *extração de características* é feita para cada subimagem, resultando em descritores que representem numericamente as características relevantes da imagem para análise. Neste mesmo exemplo, um descritor para um dígito poderia ser uma estrutura de dados que armazene as coordenadas normalizadas *x* e *y* de seu centro de gravidade e a razão entre suas dimensões.

Por fim, no processamento de nível alto, a etapa de *reconhecimento* é caracterizada pela rotulação de um objeto com base nos seus descritores. A etapa de *interpretação* é a responsável por atribuir um significado para todos os objetos rotulados - neste exemplo, os objetos seriam dígitos e o significado resultante seria um CEP. Ambas as etapas costumam ser realizadas através de técnicas de aprendizagem de máquina, dispensando a codificação de regras específicas de reconhecimento.

É relevante notar que nem todos os sistemas de visão computacional empregam todas as etapas aqui descritas. Também é válido observar que diversas combinações de etapas podem existir. Por exemplo, para reconhecer expressões em uma imagem com várias pessoas, primeiro é necessário identificar todas as faces, e então utilizar a saída deste processo para extrair as características faciais de cada rosto encontrado.

As técnicas de processamento de imagens geralmente envolvidas na identificação de faces e de características faciais são discutidas na próxima seção.

2.3 PROCESSAMENTO DE IMAGENS DIGITAIS

As técnicas de processamento de imagens digitais têm por objetivo, geralmente, aprimorar imagens para interpretação humana e a análise automática por computadores, extraindo informações de cenas (MARQUES FILHO; VIEIRA NETO, 1999, p. 1). Estas

imagens, em sua forma digital, são representadas por matrizes de *pixels* (uma abreviação de *picture elements*), elemento no qual as técnicas se baseiam para atingir seus resultados (GONZALEZ; WOODS, 2000, p. 5).

Conforme citado na seção anterior, é necessária a aplicação de técnicas do gênero para normalização, eliminação de ruídos ou dimensionamento da entrada de algoritmos de visão computacional. Uma forma amplamente utilizada para realizar estes processamentos é através de convolução com máscaras (MARQUES FILHO; VIEIRA NETO, 1999, p. 34).

Considerando uma matriz de intensidade de *pixels* adjacentes[1] $Z_1, \dots, Z_9$ de uma imagem em tons de cinza, conforme ilustrado na Figura 3a, e uma máscara 3 x 3 de coeficientes genéricos $W_1, \dots, W_9$, vista na Figura 3b. A máscara definida irá percorrer todos os *pixels* da imagem original, da esquerda para a direita, de cima para baixo, alinhando o *pixel* em questão com o *pixel* central das matrizes. O valor do pixel correspondente na imagem resultante é calculado pelo somatório $\sum_{i=1}^{9} W_i . Z_i$.

Figura 3 - Matrizes de convolução

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

(a)

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

(b)

Fonte: adaptado de Marques Filho e Vieira Neto (1999, p. 34)

Quando a "janela" estiver em algum *pixel* das extremidades da imagem (por exemplo, Z_5 na coordenada $x = 0$ e $y = 0$), os vizinhos inexistentes podem ser representados por zero ou assumir o valor do último *pixel* da linha ou coluna, sendo esta decisão variável entre diferentes técnicas.

O conceito de máscara de convolução é importante para o entendimento de algumas das técnicas mais comuns associadas à extração de características. Estas técnicas são abordadas nas próximas subseções.

2.3.1 Pré-processamento

A etapa de pré-processamento tem por finalidade "melhorar a imagem de forma a aumentar as chances para o sucesso dos processos seguintes" (GONZALEZ; WOODS, 2000, p. 6). Para tanto, são empregadas técnicas como realce de contraste, correção de gama, remoção

[1] Diz-se que um *pixel* é adjacente a outro quando a distância máxima entre eles, em quaisquer direções, é igual a um (GONZALEZ; WOODS, 2000, p. 26). *Pixels* adjacentes também são denominados vizinhos.

de ruído. A seguir são apresentadas algumas das técnicas comuns no domínio de extração de características faciais.

2.3.1.1 Realce de contraste

O contraste em uma imagem pode ser considerado como o grau de variações em seus níveis de cinza (MARQUES FILHO; VIEIRA NETO, 1999, p. 300). Desta forma, quanto maior o contraste de uma imagem, mais diferenças de intensidades são percebidas, geralmente realçando as formas presentes na mesma. Por este motivo este é um passo importante para sistemas de análises de imagem.

Para realizar a alteração de contraste, pode ser empregada a técnica de processamento ponto-a-ponto, onde a intensidade de cada pixel é alterada individualmente, sem considerar a intensidade dos *pixels* vizinhos. A forma clássica descrita por Gonzalez e Woods (2000, p. 119) consiste na aplicação de uma função matemática que transforme a intensidade de um pixel da imagem original, sendo esta função específica para cada domínio de aplicação.

2.3.1.2 Correção de gama

Segundo Szeliski (2011, p. 87), os primeiros monitores da era das TVs preto e branco, que utilizavam fósforo para exibição de imagens, não respondiam de forma linear aos impulsos elétricos recebidos. A relação entre a tensão elétrica (T) e o brilho resultante (B) foi denominado *gama* (γ), já que era matematicamente representada por $B = T^{\gamma}$. Para compensar este efeito, as câmeras da época faziam um pré-mapeamento da imagem capturada através de um gama inverso, alterando a intensidade dos *pixels*.

Embora os aparelhos de TV tenham evoluído, a relação *gama* ainda é considerada na compressão de imagens. Portanto, para adequadamente interpretar uma imagem, é necessário aplicar a correção gama.

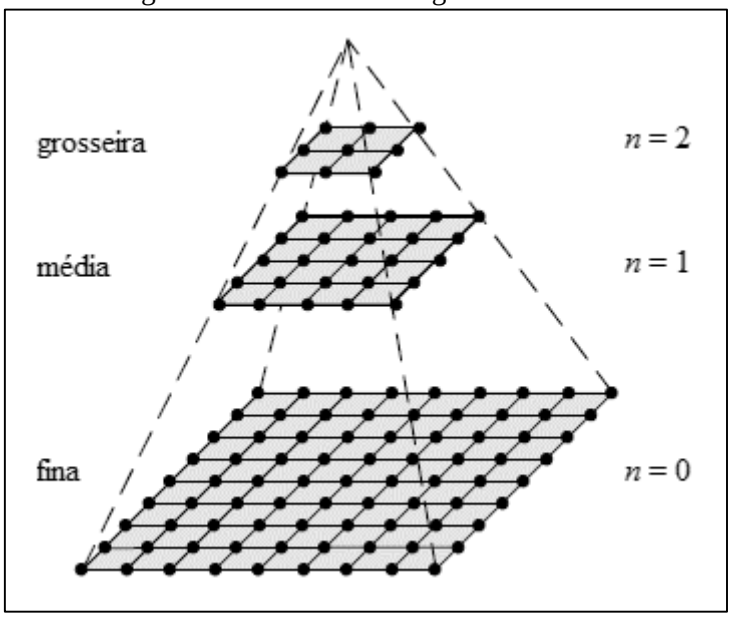
2.3.1.3 Pirâmide de imagem

Para otimizar a etapa de análise, pode-se ainda utilizar a técnica de pirâmide de imagem para reduzir suas dimensões, consequentemente preservando apenas as características mais predominantes do objeto. Assim, o custo computacional para analisar a imagem é reduzido (SZELISKI, 2011, p. 144).

> Uma pirâmide é uma coleção de representações de uma imagem. O nome pirâmide vem de uma analogia visual. Tipicamente, cada camada da pirâmide possui metade da largura e metade da altura da camada anterior, e se fossemos empilhar as camadas umas sobre as outras resultaria em uma pirâmide. (FORSYTH; PONCE, 2003, p. 205).

Esta descrição está exemplificada na Figura 4, onde os níveis de cada camada são indicados por *n*. Quanto maior o nível da camada, menor a quantidade de detalhes na imagem, resultando em uma imagem mais grosseira (*coarse*, na literatura inglesa). Camadas mais baixas são mais próximas da imagem original e, portanto, mais finas (*fine*) em detalhes.

Figura 4 - Pirâmide de imagem tradicional



Fonte: adaptado de Szeliski (2011, p. 151).

Para a obtenção de cada camada, pode ser adotada a estratégia de alternância de linhas e colunas como representação da camada anterior. Alguns filtros podem ser aplicados para suavizar estas representações, de forma a manter um certo nível de detalhe dos *pixels* descartados da imagem anterior. Uma destas técnicas é denominada *pirâmide Gaussiana*, que resulta em uma imagem suavizada (FORSYTH; PONCE, 2003, p. 205).

A técnica de pirâmide de imagem também pode ser utilizada para aumentar a dimensão da imagem. Desta forma, características sutis se tornam mais expressivas, melhorando as chances de serem corretamente identificadas nas etapas posteriores.

2.3.2 Segmentação

Após o pré-processamento da imagem, o primeiro passo em relação à sua análise é a sua segmentação, isto é, a subdivisão de objetos que a constituem. A quantidade de subdivisões a serem feitas depende do problema. Por exemplo, para a detecção de características faciais primeiro é necessário identificar uma face humana na imagem, e em seguida dividir a mesma em regiões como boca, nariz e olhos – sendo que estes últimos podem ainda ser divididos em regiões esquerda e direita.

Em uma imagem em tons de cinza, “uma borda é o limite entre duas regiões com propriedades relativamente distintas de nível de cinza” (GONZALEZ; WOODS, 2000, p. 297). Uma das técnicas mais utilizadas para o realce de bordas é o gradiente da função de luminosidade da imagem, introduzido a seguir.

2.3.2.1 Gradiente de luminosidade

É possível utilizar máscaras de convolução para se obter aproximações do cômputo da magnitude de um gradiente para um ponto (x, y) em uma imagem, o qual resulta em um nível de intensidade de pixel maior para diferenças maiores nas vizinhanças (GONZALEZ; WOODS, 2000, p. 299). Algumas dessas máscaras, também denominadas de operadores, são apresentadas no Quadro 1.

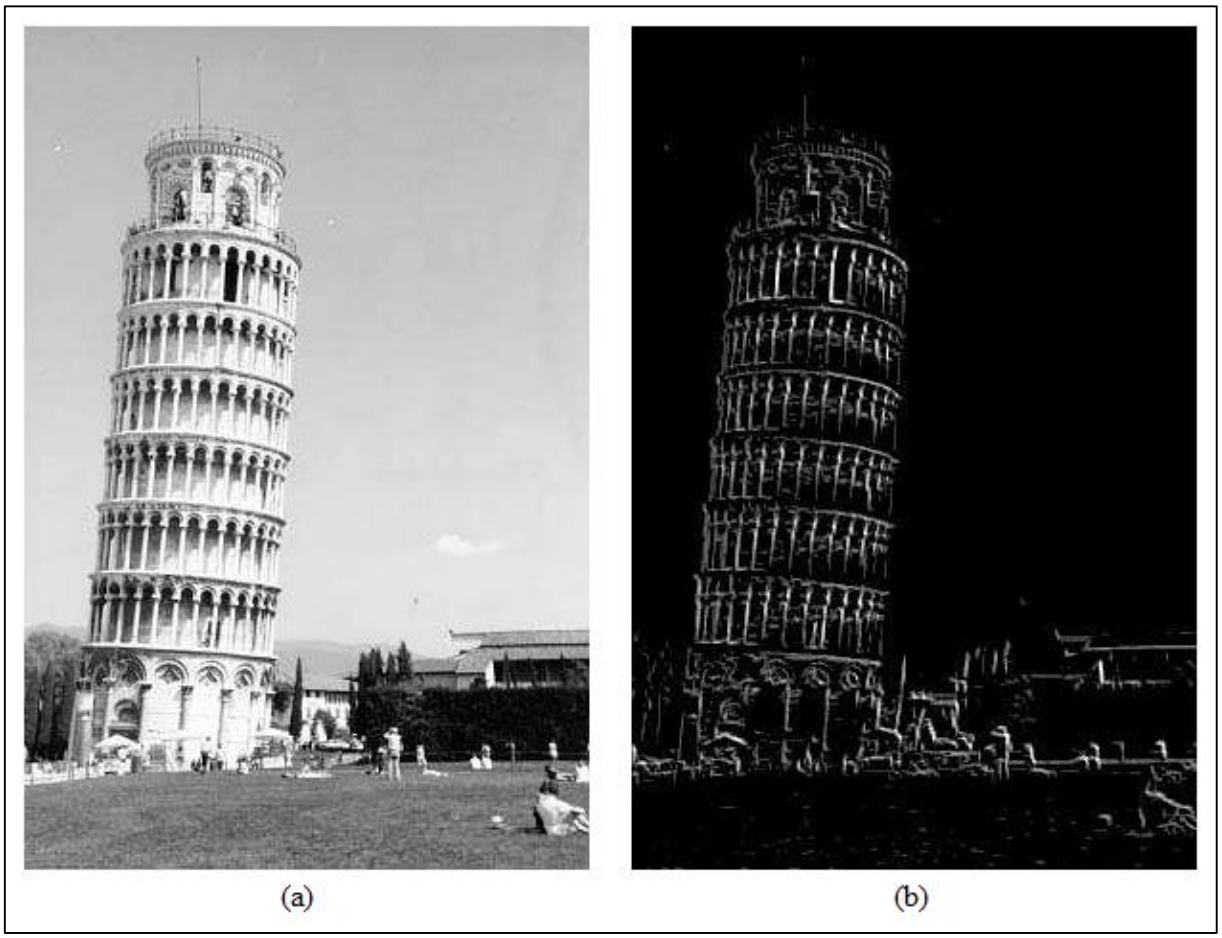
Quadro 1 - Operadores de aproximação de gradiente

Operador	Vertical	Horizontal
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Sobel	$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Prewitt	$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Fonte: adaptado de Marques Filho e Vieira Neto (1999, p. 38)

Na Figura 5 são apresentados uma imagem (a) e a combinação do resultado da aplicação dos operadores de Prewitt vertical e horizontal sobre a mesma (b).

Figura 5 - Exemplo de realce e detecção de bordas usando os operadores de Prewitt vertical e horizontal



Fonte: Marques Filho e Vieira Neto (1999, p. 38).

Conforme visto no Quadro 2, os operadores de aproximação de gradiente podem variar de acordo com a direção que se deseja realçar. Marques Filho e Vieira Neto (1999, p. 41) apresentam em seu livro operadores para realce de bordas com base em gradientes para 8 direções. Os mesmos podem ser conferidos no Quadro 2.

Quadro 2 - Operadores de gradiente orientado

Direção da borda	Direção grad.	Prewitt		
0	N	1 1 -1	1 -2 -1	1 1 -1
1	NO	1 1 1	1 -2 -1	1 -1 -1
2	O	1 1 1	1 -2 1	-1 -1 -1
3	SO	1 1 1	-1 -2 1	-1 -1 1
4	S	-1 1 1	-1 -2 1	-1 1 1
5	SE	-1 -1 1	-1 -2 1	1 1 1
6	E	-1 -1 -1	1 -2 1	1 1 1
7	NE	1 -1 -1	1 -2 1	1 1 1
Fator de escala		1/5		

Fonte: Marques Filho e Vieira Neto (1999, p. 41).

2.3.2.2 Janela de detecção deslizante

Este método é utilizado para detectar múltiplos objetos em uma mesma imagem. Segundo Forsyth e Ponce (2012, p. 519), seu conceito é simples: essencialmente, aplicar um classificador em “subjanelas” da imagem. Para isso, constrói-se uma base de dados de imagens rotuladas com tamanho fixo $n \times m$. Os exemplos rotulados como positivos devem conter imagens amplas com os objetos centralizados, enquanto as janelas negativas, não. É então treinado um classificador para diferenciar estas imagens. Em seguida, o classificador é

executado para cada janela $n \times m$ da imagem. Janelas classificadas como positivas contém o objeto, e as negativas não. Devido à possibilidade de o objeto possuir tamanhos diferentes do conjunto de treinamento, pode-se usar uma pirâmide Gaussiana para pesquisar as janelas $n \times m$ em cada camada.

Outro problema a ser contornado é quando uma mesma instância do objeto é detectada em janelas adjacentes ou mesmo parcialmente sobrepostas. Para contornar este problema, pode-se selecionar a janela que apresentar o melhor resultado de classificação, descartando as demais do conjunto de objetos detectados. Forsyth e Ponce (2012, p. 520) descrevem esta técnica através de um algoritmo, apresentado no Quadro 3.

Quadro 3 - Algoritmo de janela de detecção deslizante

Treine um classificador em janelas de imagem $n \times m$. Exemplos positivos contém o objeto, exemplos negativos não.
Escolha um limiar *t* e passos de incremento Δx e Δy nas direções *x* e *y*.

Construa uma pirâmide de imagem.

Para cada nível da pirâmide
  Aplique o classificador em cada janela $n \times m$, deslocando-a por Δx e Δy, neste nível, para obter uma resposta de força *c*.
  Se $c > t$
    Insira um ponteiro para a janela em uma lista *L*, ordenada por *c*.

Para cada janela *W* em *L*, iniciando pela de maior resposta
  Remova todas as janelas $u \neq W$ que sobreponham *W* significativamente,
    onde a sobreposição é computada na imagem original expandindo as janelas em escalas mais grosseiras.

L agora é a lista de objetos detectados.

2.3.3 Extração de características

Uma vez que a imagem esteja processada, com ruídos removidos e características visualmente realçadas, é necessário extrair os dados para análise. Esta análise é feita sobre *descritores* de imagem. Segundo Gonzalez e Woods (2000, p. 345), estes descritores são estruturas de dados que podem descrever características externas (fronteira dos segmentos) ou internas (*pixels* que compõem a região).

Um dentre vários tipos de descritores consiste na análise de histograma para este fim.

> O histograma de uma imagem [monocromática] é simplesmente um conjunto de números indicando o percentual de pixels naquela imagem que apresentam um determinado nível de cinza. Estes valores são normalmente representados por um gráfico de barras que fornece para cada nível de cinza o número (ou o percentual) de pixels correspondentes na imagem. Através da visualização do histograma de uma imagem obtemos uma indicação de sua qualidade quanto ao nível de contraste e

> quanto ao seu brilho médio (se a imagem é predominantemente clara ou escura). (MARQUES FILHO e VIEIRA NETO, 1999, p. 55)

Conforme demonstrado por Dalal e Triggs (2005), para a detecção de faces em uma imagem, é possível utilizar o Histograma de Gradiente Orientado (*Histogram of Oriented Gradient* – HOG). Esta técnica é descrita na seção 2.5.

2.3.4 Reconhecimento e interpretação

A etapa de reconhecimento de padrões geralmente ocorre através de métodos de decisão teórica (GONZALEZ; WOODS, 2000, p. 412-413), como o método de casamento por distância mínima ou correlação, classificadores estatísticos ótimos ou redes neurais artificiais. Qualquer que seja o classificador, para um vetor *n*-dimensional de características $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$, e um conjunto de classes $C = (\omega_1, \omega_2, \cdots, \omega_M)$, devem ser encontradas *M funções de decisão* $d_1(\boldsymbol{x}), d_2(\boldsymbol{x}), \cdots, d_M(\boldsymbol{x})$, tal que, se o padrão $\boldsymbol{x}$ pertencer à classe ω_i, então $d_i(\boldsymbol{x})$ deverá resultar no maior valor dentre as funções.

Por fim, a etapa de interpretação visa atribuir significado ao conjunto de elementos reconhecidos na imagem – no caso do presente trabalho, da determinação da expressão facial. Portanto, é necessária a aplicação de conhecimento específico sobre o domínio do problema, utilizando-se da "formulação de restrições e idealizações com a intenção de reduzir a complexidade da tarefa a um nível tratável" (GONZALEZ; WOODS, 2000, p. 409). Isto é possível através de técnicas como lógica de predicados e sistemas de produção.

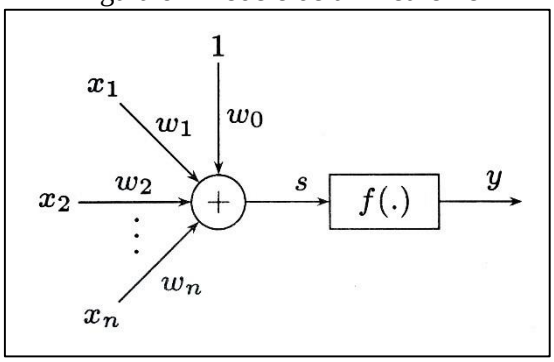
O protótipo desenvolvido neste trabalho, entretanto, explora outra alternativa para realizar esta interpretação. Trata-se do emprego de um método neuronal para o reconhecimento de padrões.

2.4 REDES NEURAIS ARTIFICIAIS

De acordo com Marques (2005, p. 163), "as redes neurais artificiais são constituídas por elementos simples interligados, com capacidade de aprendizagem a partir dos dados". A utilização de um número elevado destas unidades permite a execução de tarefas complexas. Com isso, apesar de os computadores digitais não replicarem o funcionamento do cérebro humano, é possível que os mesmos realizem tarefas que seres humanos fazem continuamente, como reconhecimento de objetos, cenas e fala.

Esta unidade de processamento simples descreve o funcionamento de um neurônio dada uma combinação linear de entradas produzidas por outras unidades, seguida de uma decisão binária. A Figura 6 ilustra este modelo.

Figura 6 - Modelo de um neurônio

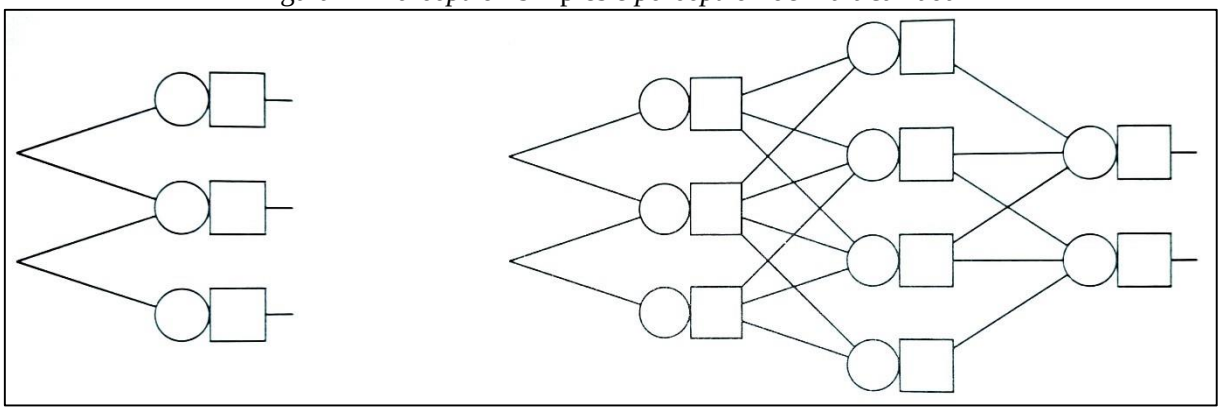


Fonte: Marques (2005, p. 164).

Na Figura 6, as entradas dos neurônios são indicadas pelo vetor $\tilde{x} = [1, x_1, \ldots, x_n]^T$ e seus pesos pelo vetor $\omega = [\omega_0, \omega_1, \ldots, \omega_n]^T$. O valor representado por s é o produto dos vetores $\tilde{x}$ e ω. A *função de ativação*, representada por f, consiste em uma função não linear que, dado o valor de s, gera uma saída y situada entre 0 e 1. Em termos matemáticos, f é definido por $f : \mathbb{R} \rightarrow [0, 1]$. A saída y é geralmente utilizada para determinar a classe que as entradas pertencem.

Uma das primeiras redes, e também uma das mais comuns, foi proposta no final da década de 50, o *perceptron*. Trata-se de uma rede onde normalmente as unidades estão divididas em camadas, de forma que "as entradas das unidades de uma camada são saídas das unidades da camada anterior" (MARQUES, 2005, p. 165). Quando a rede possui apenas uma camada é chamada de *perceptron simples* e, quando possui mais, de *perceptron de multicamada*. A Figura 7 ilustra ambas as formas.

Figura 7 - *Perceptron* simples e *perceptron* de multicamada



Fonte: Marques (2005, p. 166).

Na Figura 7 cada unidade é indicada por um círculo e um quadrado, representando a soma ponderada das entradas e a não linearidade, respectivamente. A última camada é designada por *camada de saída*. Na segunda rede podem ser observadas as *camadas escondidas*, para as quais as entradas são variáveis internas. A *arquitetura das redes*

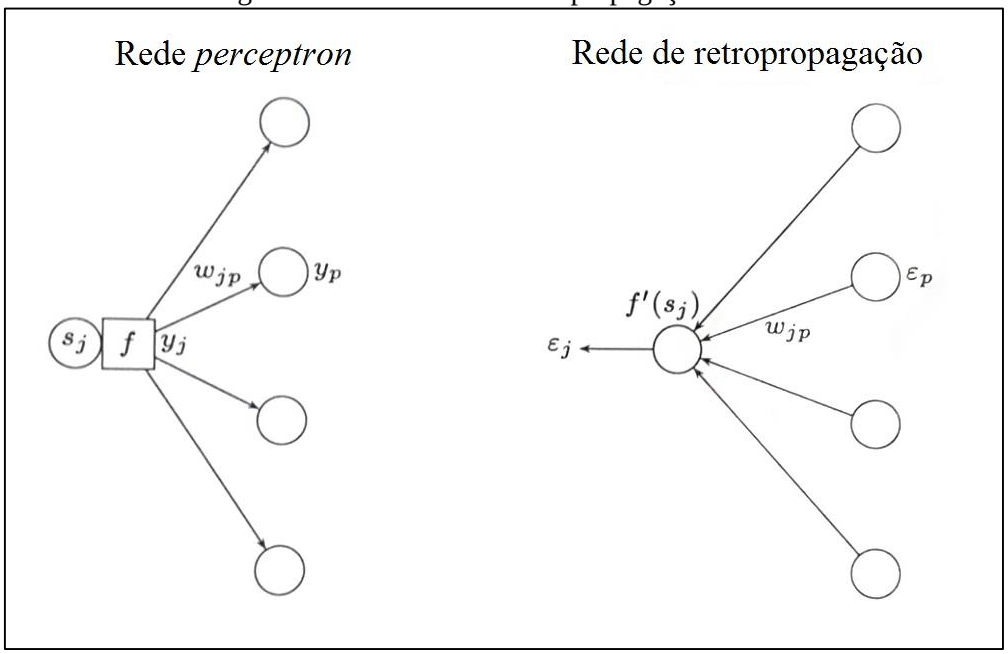
apresentadas, como é chamada a informação sobre quantidade de unidades de uma rede e suas ligações, pode ser representada de forma abreviada por 2-3 e 2-3-4-2, respectivamente.

O bom funcionamento de uma rede *perceptron* está associado à sua arquitetura e ao peso dado às entradas. A definição da arquitetura geralmente é feita manualmente, com base em critérios heurísticos e experiência passada do utilizador. Os pesos, por outro lado, podem ser obtidos através do *treinamento* da rede.

Para os *perceptrons simples*, este treino é feito de forma independente para cada unidade. Um algoritmo para escolha de pesos para redes deste tipo consiste na inicialização dos pesos da rede com valores aleatórios e repetitivos testes e ajustes destes pesos usando um padrão de treino previamente classificado. O treino é encerrado ao atingir uma condição de parada especificada.

O treinamento das redes *perceptron de multicamada* é possível através de um algoritmo recursivo designado por *método de retropropagação do erro* (referenciado na literatura inglesa por *error backpropagation*). Este mecanismo consiste na criação de uma rede transposta, denominada rede de retropropagação. Esta rede pode ser obtida invertendo o sentido dos nós da rede *perceptron*, convertendo os pontos de divergência em pontos de soma e substituindo as funções de ativação por suas respectivas derivadas (f'). Um exemplo desta transposição pode ser visto na Figura 8.

Figura 8 - Mecanismo de retropropagação de erro



Fonte: adaptado de Marques (2005, p. 177).

Na etapa de treinamento, primeiramente é feita a inicialização dos pesos da rede com valores aleatórios e então, na rede *perceptron*, são calculadas as saídas dos neurônios conforme

descrito anteriormente. Na rede de retropropagação, é calculado o *erro visto na unidade j* (ε_j), o qual determina, nas camadas de saída, o erro entre o resultado esperado para o treino e o resultado obtido. Nas demais camadas, este valor é obtido através do produto da derivada da função de ativação para a entrada do neurônio correspondente, multiplicado pela soma ponderada do erro percebido nas camadas seguintes – de forma semelhante ao cálculo das entradas dos neurônios das camadas ocultas na rede *perceptron*. Em seguida, os pesos da rede *perceptron* são corrigidos ao se aplicar a equação apresentada no Quadro 4.

Quadro 4 - Fórmula da correção do peso de um neurônio

$$\omega_{ij}(t) = \omega_{ij}(t-1) - \frac{\alpha}{N} y_i \varepsilon_j$$

Onde:
t = época atual do treinamento
α = coeficiente que define a magnitude da correção dos pesos dos neurônios

O treinamento da rede pode ser repetido até que uma taxa de reconhecimento satisfatória ou um número limite de iterações sejam atingidos. Após isto, com os pesos devidamente ajustados, as execuções seguintes da rede ocorrem sem que estes sejam alterados, apenas resultando em um valor que corresponda à entrada fornecida.

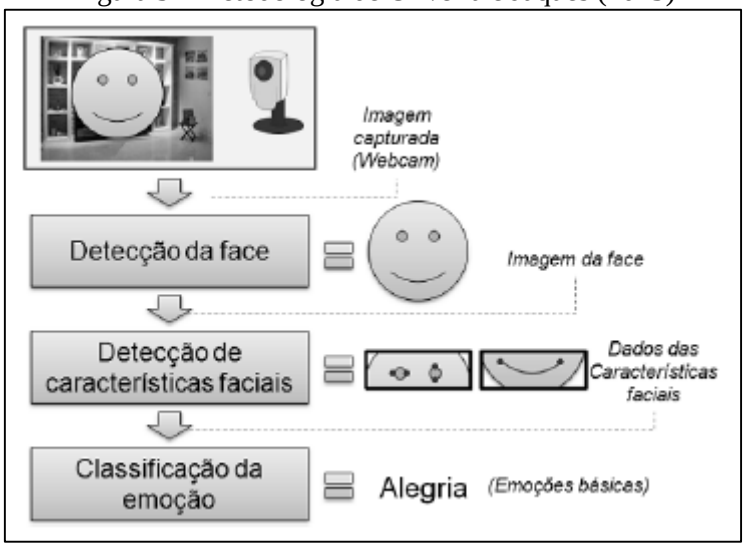
2.5 TRABALHOS CORRELATOS

Nesta seção são abordados três trabalhos correlatos. Na seção 2.6.1 é abordado o trabalho de Oliveira e Jaques (2013) para classificação de expressões faciais em vídeos. Na seção 2.6.2 é tratado o artigo de Filko e Martinović (2013) sobre reconhecimento de expressões faciais usando múltiplas redes neurais. Por fim, na seção 2.6.3, o trabalho desenvolvido no artigo de Tang e Huang (2008) para reconhecimento de expressões em uma base 3D é apresentado.

2.5.1 Classificação de emoções básicas através de imagens capturadas em vídeos de baixa resolução

O trabalho de Oliveira e Jaques (2013) apresenta um sistema que classifica as seis emoções básicas universais por meio de expressões faciais em vídeos capturados por uma *webcam*. O sistema aplica técnicas de Visão Computacional para extração de características e classificação das emoções. As principais etapas da execução do sistema podem ser vistas na Figura 9.

Figura 9 - Metodologia de Oliveira e Jaques (2013)



Fonte: Oliveira e Jaques (2013).

Como primeira etapa, é utilizado um classificador baseado em *Haar-like features* (método Viola-Jones)[2] para a localização da face em um quadro do vídeo. Existe a necessidade do utilizador posar para a câmera com uma expressão neutra para que o sistema possa classificar corretamente as demais expressões.

Na segunda etapa, ocorrem cinco subprocessos sequenciais: busca pelo centro dos olhos, correção da inclinação da face, aplicação de modelo antropométrico, identificação de pontos extremos sobre as características faciais e avaliação desses pontos extremos. É utilizado um modelo antropométrico para demarcação das regiões dos olhos, boca e sobrancelhas a partir do centro dos olhos. Isto implica em uma otimização para a tarefa de extração de características.

Para a identificação de 16 pontos extremos utilizados para classificação, são aplicados métodos de processamento de imagens, como conversão da imagem para tons de cinza, correção de histograma e de contraste, entre outros. Com isto, os pontos sobre as extremidades das características faciais são demarcados e armazenados em um vetor, chamado de vetor de pontos extremos (VPE). Estes pontos são utilizados para rastreamento em quadros seguintes, permitindo a classificação das emoções em tempo real.

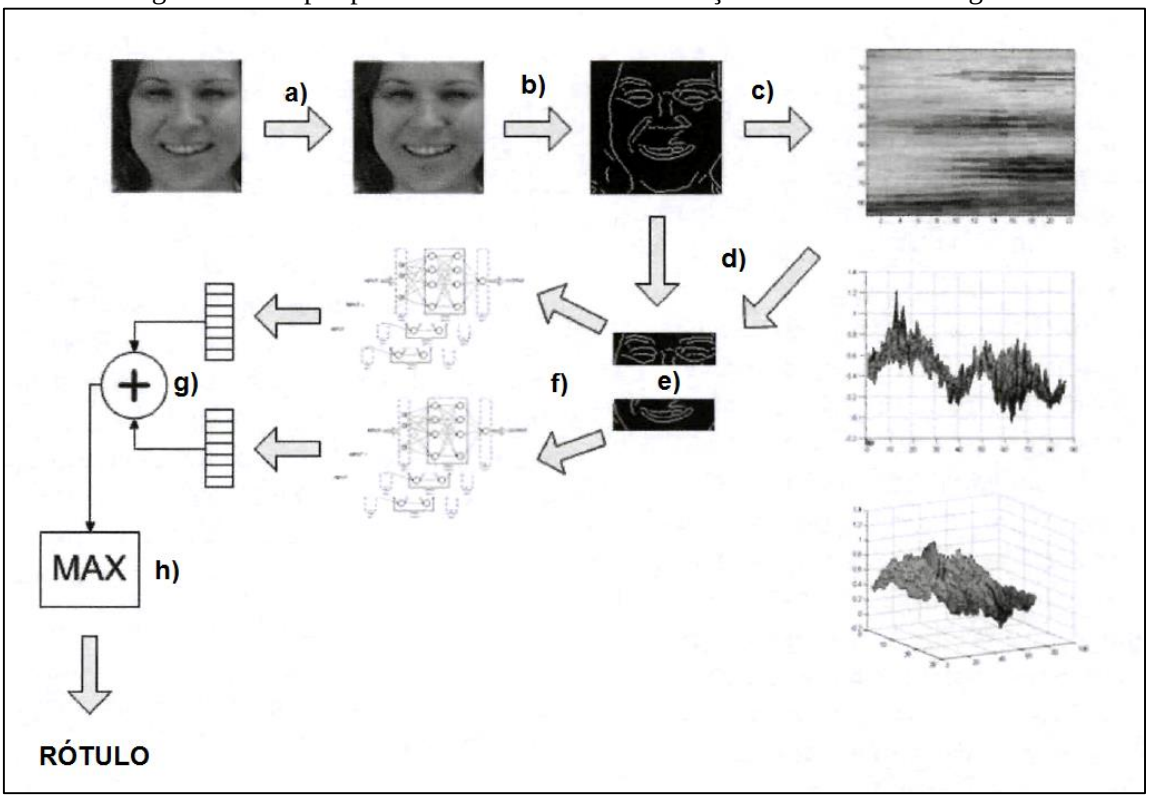
Após a extração das características, os dados são submetidos para a etapa de classificação das emoções por meio de uma rede neural artificial do tipo *multilayer perceptron feedforward*. A taxa de 63,33% de reconhecimento foi atingida com os experimentos, entretanto, a rede neural foi capaz de atingir uma taxa de 89,87% em testes isolados.

[2] Trata-se de um método rápido de detecção de objetos a partir de características simples extraídas de quadros do vídeo (VIOLA; JONES, 2001).

2.5.2 Emotion recognition system by a neural network based facial expression analysis

O sistema desenvolvido por Filko e Martinović (2013) no ambiente Matlab reconhece as emoções básicas universais a partir de imagens estáticas utilizando regiões faciais dos olhos e boca. Além das seis emoções já citadas, a expressão neutra também é reconhecida. A metodologia utilizada é ilustrada na Figura 10.

Figura 10 - Etapas para o reconhecimento de emoções humanas em imagens



Fonte: adaptado de Filko e Martinović (2013).

As etapas ilustradas consistem em:

a) carregamento da imagem e conversão para escala de cinza;
b) utilização do algoritmo *Canny* para detecção de bordas, pois o mesmo permite maior controle para este processo;
c) divisão da imagem binária resultante em retângulos de 94x30 *pixels*, sendo cada um deles analisado por uma rede neural para identificação de olhos e bocas;
d) análise para identificação dos retângulos com maiores chances de conter olhos ou boca;
e) geração dos vetores de características das regiões escolhidas no passo anterior – usando o algoritmo *Principal Component Analisys* (PCA), que trata a imagem como um padrão em um espaço linear para geração de vetores estatísticos denominados *eigenfaces*;

f) pontuação de cada emoção a partir de redes neurais, sendo uma rede para cada combinação entre região facial e emoção (totalizando quatorze somente para esta etapa);
g) soma dos pesos determinados pelas redes neurais de cada emoção para as regiões analisadas;
h) determinação da emoção presente na imagem a partir da melhor pontuação da etapa anterior.

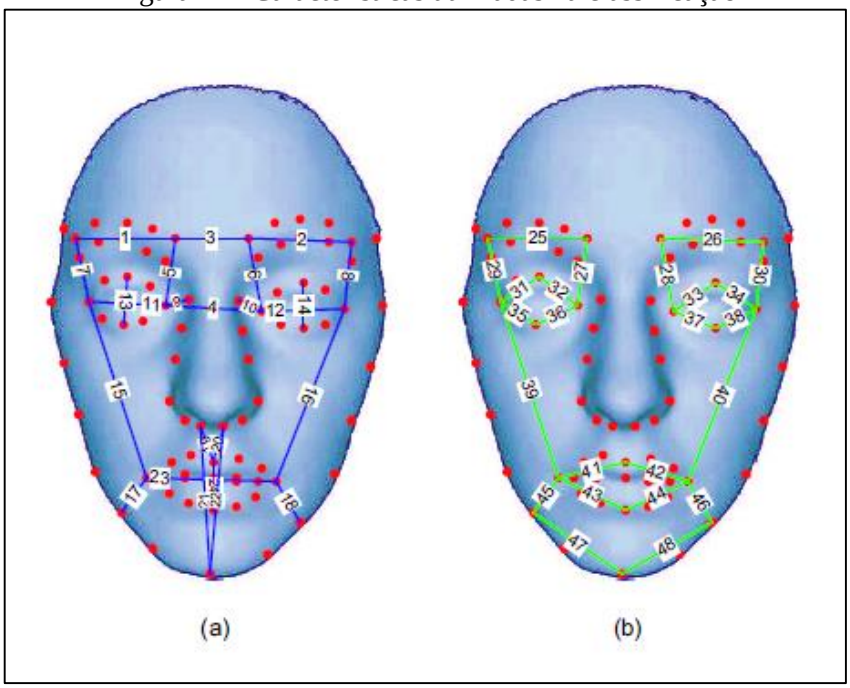
Para testar o sistema foi utilizada uma base com 15 imagens para cada uma das 6 emoções, além da expressão neutra. A taxa de acerto de classificação variou entre 46% e 80%, ficando em média em 70%. O sistema mostrou-se capaz de reconhecer as emoções na maioria dos casos testados.

2.5.3 3D facial expression recognition based on properties of line segments connecting facial feature points

O trabalho desenvolvido por Tang e Huang (2008) consiste no reconhecimento das seis emoções universais citadas neste trabalho, utilizando geometria facial em 3D. Foram consideradas, como características faciais, propriedade de segmentos de linhas conectando certos pontos 3D de características faciais.

As distâncias normalizadas (Figura 11a) e declives (Figura 11b) destes segmentos de linha compõem um conjunto de 96 características distintas para o reconhecimento das emoções.

Figura 11 - Características utilizadas na classificação



Fonte: Tang e Huang (2008).

Foi utilizado um classificador SVM[3], com o qual se obteve um desempenho médio de 87,1% de reconhecimento na base de testes BU-3DFE. A maior taxa de reconhecimento obtida foi de 99,2% para o reconhecimento de expressões de surpresa.

[3] Em suma, Norvig e Russel (2013, p. 648) definem SVM (*Support Vector Machine*, ou *Máquina de Vetor de Suporte*) como uma abordagem para aprendizagem supervisionada. A técnica é atraente por construir um separador de margem máxima (aumentando o poder de generalização), possibilidade de incorporar dados em um espaço de dimensão superior (permitindo classificações lineares e não-lineares) e por dispensarem parametrização (ao custo de manter alguns ou todos os exemplos de teste).

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são apresentadas as etapas do desenvolvimento do protótipo, fundamentado nos conceitos apresentados no capítulo anterior. São abordados os requisitos do protótipo, seguidos de sua especificação, implementação e discussão dos resultados obtidos.

3.1 REQUISITOS

Nesta seção são apresentados os Requisitos Funcionais (RF) e os Requisitos Não Funcionais (RNF) atendidos pelo protótipo.

O protótipo deverá:

a) utilizar as técnicas apresentadas para detecção de múltiplas faces em uma imagem (RF);
b) utilizar as técnicas apresentadas para extração de características faciais (RF);
c) extrair as características relevantes para o reconhecimento das expressões faciais (RF);
d) utilizar redes neurais artificiais para a classificação de expressões faciais (RF);
e) ser implementado utilizando o ambiente de desenvolvimento Microsoft Visual Studio (RNF);
f) ser implementado utilizando a linguagem de programação C++ (RNF);
g) ser implementado utilizando a biblioteca *dlib* (RNF);
h) reconhecer adequadamente imagens no formato JPG e PNG (RNF);
i) otimizar o uso do processador através de paralelismo para extrair as características faciais (RNF);
j) disponibilizar uma interface de linha de comando para interação (RNF).

3.2 ESPECIFICAÇÃO

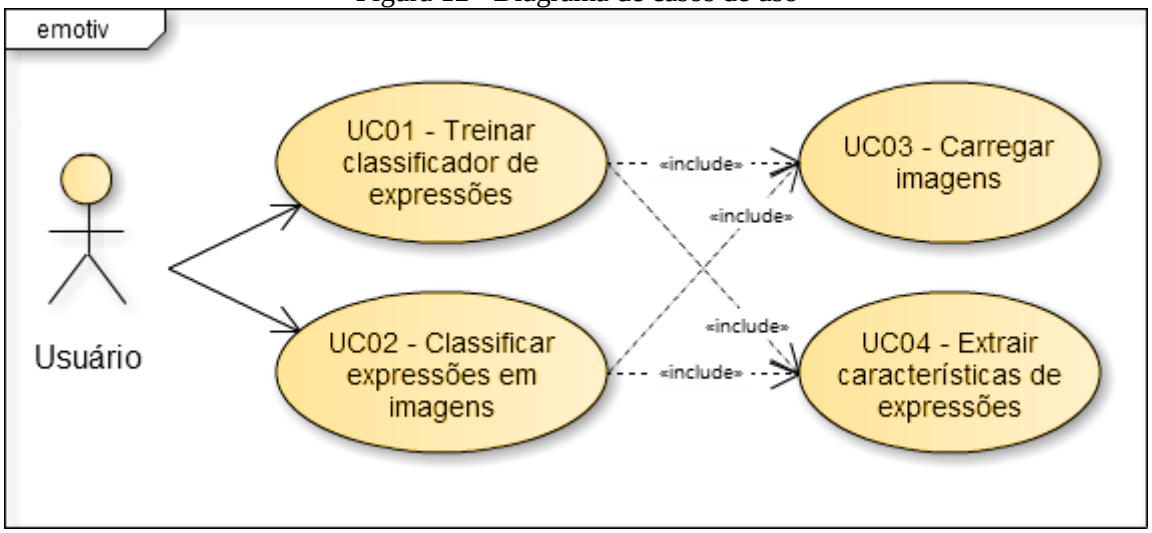
Esta seção apresenta a especificação do protótipo e suas funcionalidades, modelada com base em uma abordagem orientada a objetos na ferramenta Cacoo[4], utilizando-se a *Unified Modeling Language* (UML). Foram utilizados os diagramas de casos de uso e de classe para representar a metodologia de desenvolvimento do protótipo.

[4] Ferramenta baseada em navegador *web* para criação e edição de diagramas de variados tipos, incluindo modelos UML. Criada e mantida pela empresa Nulab (CACOO, 2016).

3.2.1 Diagramas de casos de uso

O protótipo apresenta quatro casos de uso, visualizados na Figura 12 e detalhados a seguir.

Figura 12 - Diagrama de casos de uso



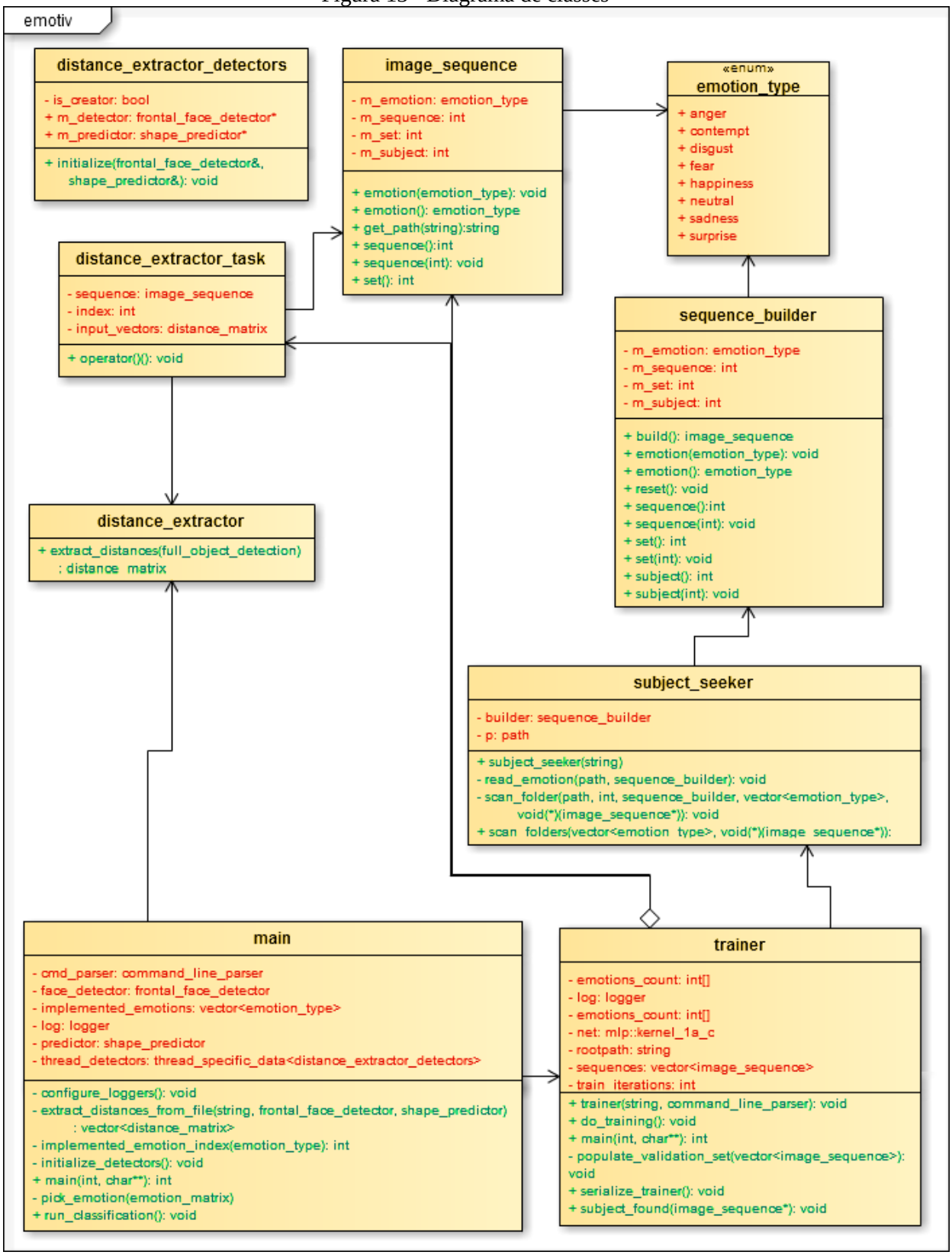
O `UC01 – Treinar classificador de expressões` é responsável pelo treinamento da rede neural que classifica as expressões. Para tanto, é necessário carregar as imagens de teste, função desempenhada pelo `UC02 – Carregar imagens`. Após o carregamento, a identificação de faces presentes nas imagens e extração de suas características é executada no `UC03 – Extrair características de expressões`.

É possível executar o protótipo em imagens especificadas pelo usuário, através do `UC02 – Classificar expressões em imagens`. Neste caso, é necessário que tenha ocorrido ao menos um treinamento da rede previamente. Este caso de uso se assemelha ao `UC01` por ambos dependerem dos casos de uso `UC03` e `UC04` para carregamento de imagens a serem analisadas e extração de suas características.

3.2.2 Diagramas de classes

Nesta seção é apresentado o diagrama de classes do protótipo, visto na Figura 13. Cabe notar que se optou por seguir o padrão de nomenclatura encontrado na biblioteca dlib, onde todas as letras dos nomes de classe são minúsculas e as palavras são separadas por um *underscore* (_).

Figura 13 - Diagrama de classes



A classe `main` representa o ponto no qual a execução do protótipo é iniciada. Esta classe é responsável pela leitura de argumentos de linha de comando fornecidos pelo usuário e inicialização do detector de faces e extrator de características. Esta mesma classe, com base nos

parâmetros informados pelo usuário, direciona a execução para a classificação de imagens fornecidas ou para o treinamento do classificador.

A classe `trainer` tem por responsabilidade orquestrar a busca de imagens e o treino do classificador. Esta busca acontece na forma de uma varredura de diretórios, realizada pela classe `subject_seeker`. Para cada arquivo de imagem de teste encontrado, uma representação é gerada na forma de uma instância da classe `image_sequence`. A criação desta representação é facilitada através da classe `sequence_builder`, que torna fácil a reutilização de parâmetros entre a construção de uma representação e outra.

A classe `distance_extractor_task` é utilizada para o treinamento da rede de forma paralela. Para tanto, é necessário que cada *thread* (linha de execução paralela) tenha uma cópia do detector de faces e do extrator de características faciais, sendo esta cópia gerenciada e encapsulada na classe `distance_extractor_detectors`.

Para a extração das características de expressões faciais, a classe `distance_extractor` é utilizada tanto no treinamento quanto na classificação de imagens fornecidas pelo usuário. Durante as etapas de treinamento e classificação, as emoções detectadas ou esperadas são representadas através da enumeração `emotion_type`.

3.3 IMPLEMENTAÇÃO

Nesta seção são detalhadas as técnicas, bibliotecas, ferramentas utilizadas e questões específicas do protótipo. Por fim, é detalhada a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do protótipo deste trabalho foi utilizado o ambiente de desenvolvimento *Microsoft Visual Studio Community 2015*, para codificação na linguagem *C++*. Tomou-se o cuidado para utilizar apenas bibliotecas multiplataforma, porém a implementação foi realizada na plataforma *Microsoft Windows 10* com arquitetura de 64 bits, não tendo sido averiguada a compatibilidade com outros sistemas operacionais.

O protótipo faz amplo uso da biblioteca dlib (DLIB, 2016), versão 18.18, a qual é multiplataforma e implementada em C++. Dentre as funcionalidades disponibilizadas, estão algoritmos de aprendizado de máquina, processamento de imagens, algoritmos numéricos e outros utilitários genéricos.

A rede neural disponibilizada pela biblioteca também foi utilizada. Esta implementação é do tipo *perceptron* de multicamada com algoritmo de retropropagação de erro. São disponibilizados os parâmetros *alpha* (correspondente ao coeficiente α, visto na seção 2.4) e

momentum. Através deste segundo, determina-se a fração de uma atualização de peso anterior a ser levada adianta no próximo treino.

As técnicas utilizadas no desenvolvimento do protótipo são apresentadas nas subseções a seguir, adotando a ordem do caso de uso `UC01` e então `UC02`, contemplando todas as funcionalidades.

3.3.1.1 Interpretação de argumentos de linha de comando

O protótipo disponibiliza uma interface de linha de comando para sua execução. É através de argumentos específicos que o usuário informa sua intenção e a forma como deseja que o protótipo se comporte.

Para realizar a interpretação dos argumentos foi utilizada a classe `command_line_parser`, um utilitário disponibilizado pela biblioteca dlib.

Os argumentos de linha de comando suportados pelo protótipo serão apresentados juntamente com a funcionalidade correspondente.

3.3.1.2 Carregamento de imagens

Dado o caminho de uma imagem, o carregamento da mesma é feita pelo método `load_image(image_type, string)`, disponibilizado pela biblioteca dlib. O primeiro argumento é uma estrutura que permita armazenar os componentes dos *pixels* carregados, e o segundo é o caminho absoluto da imagem, no formato JPEG ou PNG. Para armazenar a imagem em memória, é utilizada a forma sugerida pela biblioteca, através de um objeto do tipo `array2d<rgb_pixel>`, conforme visto no Quadro 5.

Quadro 5 - Carregamento de imagem em memória

```
array2d<rgb_pixel> img;
load_image(img, path);
```

No carregamento de imagens para a classificação, o caminho da imagem é fornecido pelo próprio usuário. Entretanto, para o treinamento é esperado uma estrutura de diretórios correspondente ao Quadro 6, onde os níveis de cada pasta são indicados pelo espaço em branco em frente ao nome do diretório.

Quadro 6 - Estrutura de diretórios da base de teste

Raiz	`/dataset`
Modelo	`/S001`
Conjunto	`/001`
Imagem neutra	`/S001_001_00000001.png`
Imagem com expressão	`/S001_001_00000014.png`
Rótulo da emoção	`/S001_001_00000014_emotion.txt`
Demais conjuntos	`/...` `/...`

Conforme visto, é esperado que a pasta raiz informada pelo usuário possua subpastas com imagens de uma mesma pessoa (modelo). Cada subpasta deve estar separada por conjuntos que contenham uma imagem de expressão neutra e outra com uma expressão da emoção. Ambas imagens devem respeitar a nomenclatura conforme estabelecido no Quadro 6, devendo a imagem de emoção neutra possuir numeração 1 (um). A imagem com expressão não possui restrição quanto à sua numeração, porém apenas será reconhecida se no diretório houver um arquivo de mesmo nome, porém com prefixo `_emotion.txt`. O conteúdo do arquivo, deve estar um código que represente a emoção da imagem, conforme Tabela 1.

Tabela 1 - Código das expressões faciais de emoção

Emoção	Código
Neutra	0
Raiva	1
Desprezo	3
Medo	4
Felicidade	5
Tristeza	6
Surpresa	7

Pode-se observar que não há emoção mapeada para o código 2. Isto se dá por a base utilizada para testes do protótipo utilizar este código para indicar uma emoção não contemplada neste trabalho.

Esta varredura de diretórios é feita pela classe `subject_seeker`, com auxílio de um objeto `sequence_builder` para coletar as informações de modelo, conjunto, sequência e emoção, para enfim gerar uma representação da imagem na forma de uma instância da classe `image_sequence`. O construtor da classe `subject_seeker` possui um parâmetro que indica o diretório raiz da busca.

A busca, entretanto, é feita ao se executar o método `scan_folders`, o qual recebe por parâmetro uma lista de emoções a serem aceitas para treinamento e uma função *call-back* que define o que fazer com as sequências criadas (neste caso, o objeto `trainer` armazena as

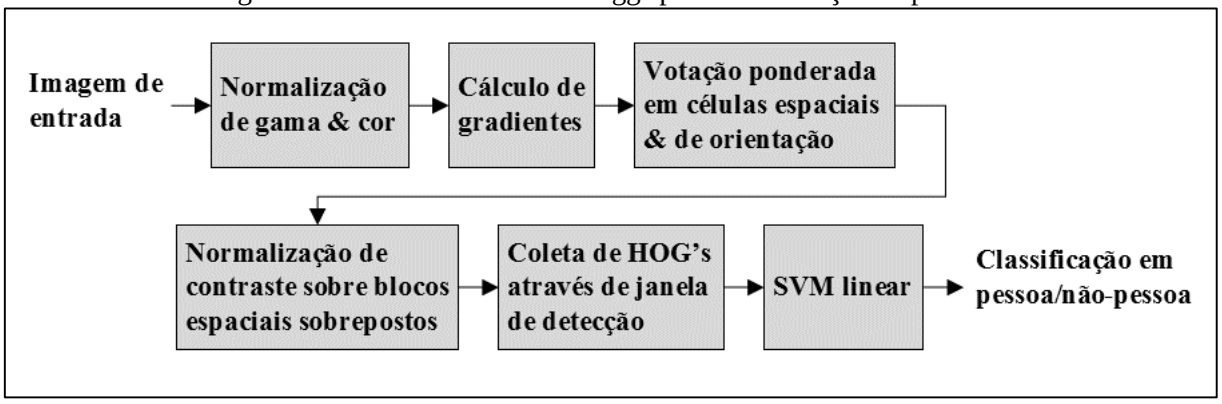
sequencias em uma lista interna). Por padrão, a lista de expressões aceitas inclui as opções neutra, felicidade e tristeza, porém o usuário pode definir sua própria lista.

Embora as imagens sejam mapeadas em memória, elas apenas são carregadas na etapa de detecção de faces, conforme apresentado a seguir.

3.3.1.3 Detecção de faces

Para a detecção de faces é utilizada a classe `frontal_face_detector`, disponibilizada pela dlib. Esta classe se trata de uma implementação do modelo desenvolvido por Dalal e Triggs (2005), o qual se destaca por se basear na análise de Histograma de Gradiente, descrito na seção 2.3.2.1. O mesmo é representado na Figura 14**.**

Figura 14 - Modelo de Dalal e Triggs para identificação de pessoas



Fonte: adaptado de Dalal e Triggs (2005).

A primeira etapa utiliza algumas técnicas de normalização de propriedades de imagem apresentadas na seção 2.3.1. Já a etapa de cálculo de gradientes é explicada na seção 2.3.2.1. Na etapa seguinte, cada *pixel* calcula um peso para um canal de histograma de gradiente, baseado na magnitude e orientação do seu gradiente. Estes votos são acumulados em *cestas de orientação* sobre regiões espaciais chamadas de *célula***s**.

Devido à variação de contraste presente dentro do gradiente de uma mesma imagem, é realizada a normalização desta propriedade localmente, isto é, dentro das células. Na etapa seguinte, os histogramas de gradientes de oito direções são coletados através do esquema de janela de detecção deslizante (explicado na seção 2.3.2.2). Por fim, estes gradientes são classificados como pertencentes ou não à classe buscada por um classificador SVM.

No Quadro 7 é possível ver como obter uma instância do detector (linha 01), seguido de um exemplo de como extrair as faces (linha 06).

Quadro 7 - Detecção de faces

```
01  frontal_face_detector f_detector = get_frontal_face_detector();
02  // ...
03  array2d<rgb_pixel> img;
04  load image(img, path);
05
06  std::vector<dlib::rectangle> dets = f_detector(img);
```

O retorno da execução do detector de faces é um vetor de retângulos, representando as coordenadas e tamanho das faces encontradas na imagem fornecida como parâmetro. A partir destas imagens é possível extrair as características faciais.

3.3.1.4 Extração de características faciais

Nesta etapa, é utilizado o objeto `shape_predictor`, da biblioteca dlib, para a extração de pontos sobre as faces detectadas na etapa anterior. Esta classe é a implementação da biblioteca para o modelo de Kazemi e Sullivan (2014) utilizam um arranjo "cascata" de árvores de regressão[5] Não são utilizadas técnicas de pré-processamento ou segmentação de imagens. Os autores utilizam, no entanto, uma indexação de *pixels* baseada pela sua intensidade e proximidade com a forma esperada. Eles recomendam por este motivo, que o algoritmo seja executado para a saída de um detector de faces, já que o classificador depende de a forma buscada (uma face humana) ocupar as mesmas dimensões das imagens de treinamento na imagem sendo classificada.

O classificador desenvolvido necessita de poucos treinos para reduzir a margem de erro, além de ser executado muito rapidamente (na ordem de poucos milissegundos) com coeficientes de erro tão baixos quanto 0.049.

Este objeto é de uso genérico, podendo ser treinado para reconhecer diferentes formas, conforme mostrado por Kazemi e Sullivan (2014). Entretanto, no site oficial da biblioteca é possível obter o arquivo `shape_predictor_68_face_landmarks.dat`, que é uma versão serializada de um `shape_predictor` treinado para o reconhecimento de 68 pontos sobre faces humanas.

No Quadro 8 é demonstrado como é feito o carregamento deste arquivo para utilização no protótipo, seguido de um trecho onde o mesmo é utilizado na saída da etapa anterior para extração das características faciais.

---

[5] Árvores de regressão e classificação são descritas por Timofeev (2004) como um método que constrói árvores de decisão com base em informações históricas de um conjunto de dados. Estas árvores de decisão, por sua vez, podem ser usadas para classificação de novas informações.

Quadro 8 - Carregamento do detector de pontos faciais

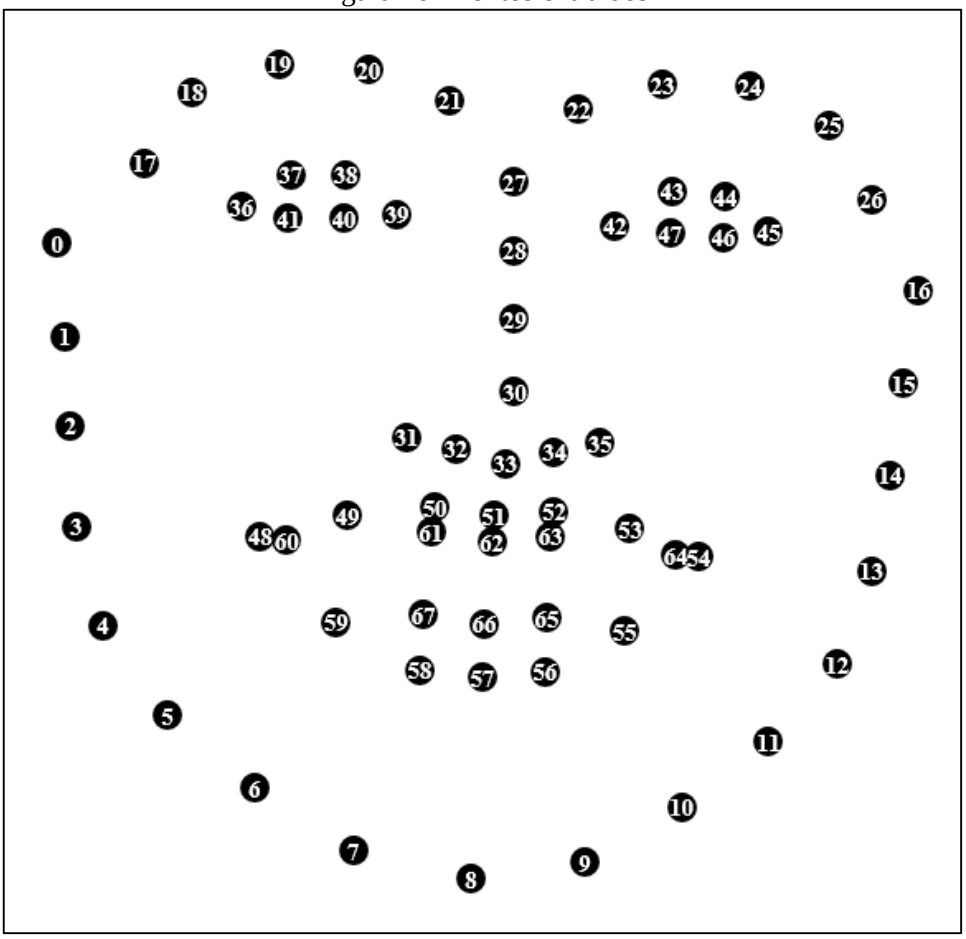
```
01  shape_predictor predictor;
02  deserialize("shape_predictor_68_face_landmarks.dat")
03    >> predictor;
```

Embora o protótipo não apresente saída gráfica, a Figura 15 ilustra um traçado obtido a partir da saída dos 68 pontos retornados pelo `shape_predictor`. A imagem foi obtida ao se executar o exemplo de aplicação deste detector no site da biblioteca sobre uma das imagens da base de testes.

Figura 15 - Traçado obtido a partir dos pontos faciais rertornados pelo `shape_predictor`

Na Figura 16 é possível observar todos os 68 pontos individualmente para esta mesma imagem, juntamente com seus índices.

Figura 16 - Pontos extraídos



No Quadro 9 é listado o código que une as etapas anteriores à atual. O mesmo é explicado a seguir.

Quadro 9 – Extração das características visuais

```
01  std::vector<distance_matrix> extract_distances_from_file(
02    string &path,
03    frontal_face_detector &f_detector,
04    shape_predictor &s_predictor) {
05
06    array2d<rgb_pixel> img;
07    load_image(img, path);
08
09    std::vector<dlib::rectangle> dets = f_detector(img);
10
11    std::vector<distance_matrix> distances;
12    for (int i = 0; i < dets.size(); i++) {
13      auto face = dets.at(i);
14      full_object_detection shape = s_predictor(img, face);
15      distance_matrix matrix = extract_distances(shape);
16      distances.push_back(matrix);
17    }
18    return distances;
19  }
20
```

O algoritmo inicia com o carregamento da imagem na linha 07 – podendo esta ter sido fornecida pelo usuário ou descoberta pelo treinador. Em seguida, na linha 09, é executado o

detector de faces, sendo as faces detectadas armazenadas em um vetor. Para cada uma destas, o extrator de características faciais é executado. O objeto retornado, `full_object_detection`, possui as coordenadas dos 68 pontos mencionados anteriormente.

For fim, as características relevantes para o reconhecimento de expressões faciais são extraídas pelo método `extract_distances`, abordado a seguir.

3.3.1.5 Extração de características de expressões faciais

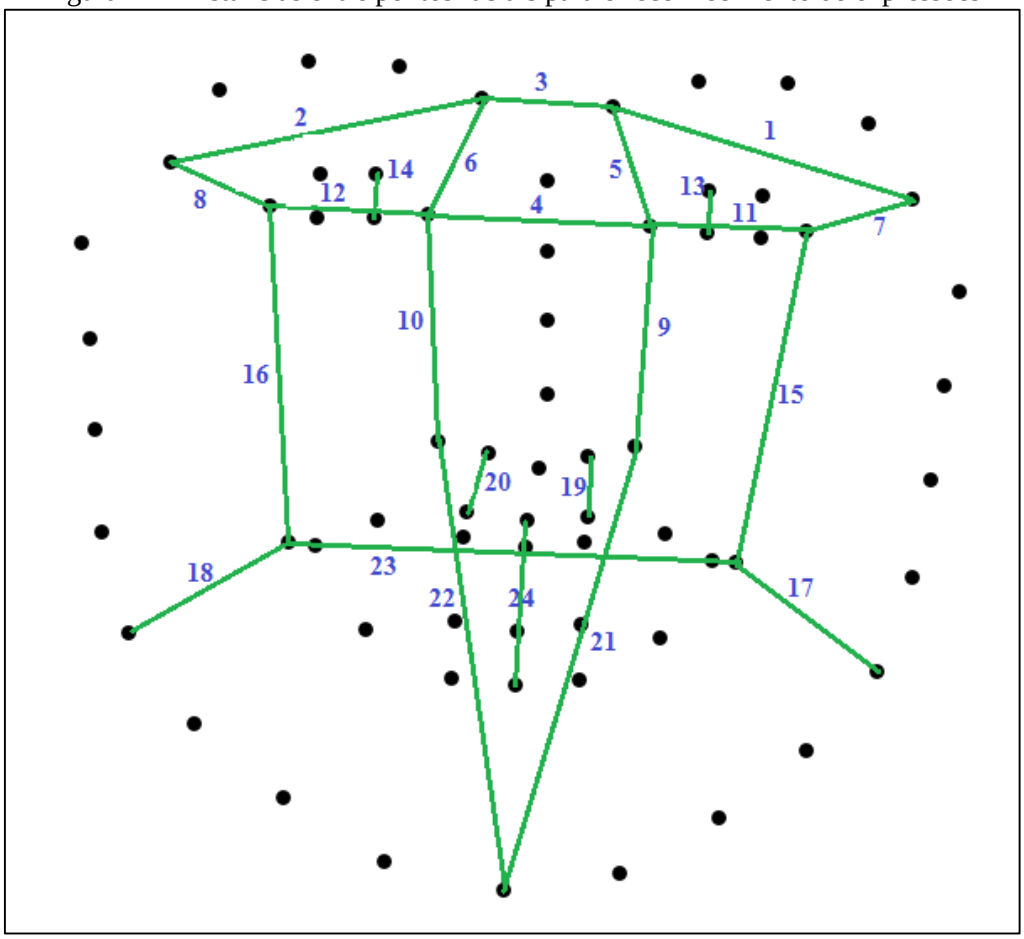
Após a obtenção dos pontos faciais na etapa anterior, são calculadas 24 distâncias entre estes, representando o relaxamento ou contração muscular envolvidos na expressão das emoções através da face. As distâncias consideradas foram inspiradas pelo trabalho de Tang e Huang (2008) e podem ser visualizadas na Tabela 2.

Tabela 2 - Características de expressão facial

Identificação	Descrição
1	Comprimento da sobrancelha direita.
2	Comprimento da sobrancelha esquerda.
3	Distância entre a parte interna das sobrancelhas.
4	Distância entre a parte interna dos olhos.
5	Distância entre o canto interno da sobrancelha direita e o canto interno do olho direito.
6	Distância entre o canto interno da sobrancelha esquerda e o canto interno do olho esquerdo.
7	Distância entre o canto externo da sobrancelha direita e o canto externo do olho direito.
8	Distância entre o canto externo da sobrancelha esquerda e o canto externo do olho esquerdo.
9	Distância entre o canto interior do olho direito e a narina direita.
10	Distância entre o canto interior do olho esquerdo e a narina esquerda.
11	Largura do olho direito.
12	Largura do olho esquerdo.
13	Altura do olho direito.
14	Altura do olho esquerdo.
15	Distância entre o canto externo do olho direito e o canto externo direito da boca.
16	Distância entre o canto externo do olho esquerdo e o canto externo esquerdo da boca.
17	Distância entre o canto externo direito da boca e a origem da mandíbula direita.
18	Distância entre o canto externo esquerdo da boca e a origem da mandíbula esquerda.
19	Distância entre a narina direita e o lábio superior médio.
20	Distância entre a narina esquerda e o lábio superior médio.
21	Distância entre a narina direita e o queixo.
22	Distância entre a narina esquerda e o queixo.
23	Largura da boca.
24	Abertura da boca.

Para melhor entendimento, a Figura 17 apresenta estas distâncias com base nos pontos da Figura 16.

Figura 17 - Distâncias entre pontos faciais para o reconhecimento de expressões



Conforme citado na subseção anterior, o cálculo destas distâncias é realizado pelo método `extract_distances`, listado parcialmente no Quadro 10.

Quadro 10 - Cálculo das distâncias referentes a expressões faciais

```
01 distance_matrix extract_distances(full_object_detection &shape) {
02   distance_matrix distances;
03   int index = 0;
04
05
06 double reference = get_distance(shape.part(42), shape.part(45));
07   double greatestDistance = -1;
08   double smallestDistance = 10e100;
09
10   // 01 - the length of the right eyebrow
11   define(shape, distances, index++, 22, 26, reference,
12     greatestDistance, smallestDistance);
13   //...
14   // 24 - the height of the mouth
15   define(shape, distances, index++, 51, 57, reference,
16     greatestDistance, smallestDistance);
17
18   greatestDistance -= smallestDistance;
19   for (int i = 0; i < index; i++) {
20     distances(i) =
21       (distances(i) - smallestDistance) / greatestDistance;
22   }
23   return distances;
24 }
```

Inicialmente, é calculada a largura do olho direito para que esta medida seja usada na normalização dos demais valores. Isto é, ao invés de retornar as distâncias absolutas entre os pontos, o método normaliza todas distâncias com base em uma que não se modifica independente da expressão manifestada.

Em seguida, é realizado o cálculo da distância normalizada em relação à distância descrita anteriormente. É feita também a coleta da maior e da menor distâncias calculadas, de forma que, entre as linhas 18 e 22, todas as distâncias se encontrem entre 0 e 1. Esta etapa é crucial para o correto funcionamento do classificador, descrito na próxima subseção.

No Quadro 11 são listados os métodos `get_distance` e `define`, que calculam as distâncias e as armazenam na matriz de resultados.

Quadro 11 - Métodos utilitários para o cálculo de distâncias faciais

```
01 inline double get_distance(point &point_a, point &point_b) {
02   long xd = point_b.x() - point_a.x();
03   long yd = point_b.y() - point_a.y();
04   return sqrt(xd * xd + yd * yd);
05 }
06
07 inline double define(full_object_detection &shape,
08   distance_matrix &distances,
09   int index, int point_a, int point_b, double reference,
10   double &greatestDistance, double &smallestDistance) {
11
12   point p_a = shape.part(point_a);
13   point p_b = shape.part(point_b);
14   double distance = get_distance(p_a, p_b) / reference;
15   distances(index) = distance;
16
17   if (distance > greatestDistance) {
18     greatestDistance = distance;
19   }
20   if (distance < smallestDistance) {
21     smallestDistance = distance;
22   }
23   return distance;
24 }
25
```

O método `get_distance` é responsável por calcular a distância euclidiana entre dois pontos da imagem. O segundo método, `define`, calcula a distância entre dois elementos do vetor de pontos faciais dados apenas os seus índices, realiza a normalização da distância com base no parâmetro `reference` (a largura do olho direito) e armazena o valor no índice indicado da matriz de distâncias. Adicionalmente, o método verifica se distância recém-calculada é a maior ou a menor até o momento, armazenando o valor para posterior normalização das demais distâncias.

3.3.1.6 Treinamento da rede neural

Com as distâncias extraídas e já normalizadas, o treino da rede ocorre de maneira simples. Primeiramente, são interpretados os argumentos de linha de comando `--no1` e `--no2` fornecidos pelo usuário, que denotam, respectivamente, a quantidade de nós da primeira e da segunda camada oculta. É interpretado também o argumento `-i`, que determina a quantidade de iterações de treinamento. Com isto, é instanciado um objeto do tipo `mlp::kernel_1a_c`, disponibilizado pela dlib, com as configurações indicadas pelo usuário.

Caso o usuário não forneça nenhum parâmetro de treinamento, a rede segue a arquitetura 24-26-24-3 e executa 500 iterações de treinamento, para as emoções neutra, felicidade e tristeza.

A interpretação de argumentos e instanciação da rede podem ser vistos no Quadro 12. Conforme visto no diagrama de classes (Figura 13), a rede é representada pelo atributo `net`. Entretanto, por uma peculiaridade da biblioteca, não é possível realizar uma atribuição direta para atributos desse tipo. Uma alternativa é o uso de ponteiros, e outra, mais simples, é a invocação do método `swap`, conforme visto na listagem de código, que troca as configurações da rede com as de outra rede criada.

Quadro 12 - Configuração da rede neural

```
01  void run_training() {
02    train_iterations = get_option(cmd_parser, "i", train_iterations);
03    int hidden_nodes_first = get_option(cmd_parser, "no1", 26);
04    int hidden_nodes_second = get_option(cmd_parser, "no2", 24);
05
06    mlp::kernel_1a_c custom_net(24,
07          hidden_nodes_first,
08          hidden_nodes_second,
09          enabled_emotions.size());
10    net.swap(custom_net);
11    //...
12  }
```

As imagens de testes coletadas na primeira etapa são divididas em um grupo de treinamento e um de validação, sendo reservado um terço da quantidade total de imagens para este segundo.

Para a obtenção da matriz de distância, procedimento este descrito nas etapas anteriores, o treinador utiliza a quantidade de *threads* suportada pelo processador da máquina do usuário. Estas *threads* executam a operação parênteses da classe `distance_extractor_task` (`operator()()`), a qual executa todas as etapas de extrações de características e alimentam um vetor contendo as matrizes de distância das imagens de teste.

O treinamento da rede inicia apenas após a extração das características de todas as imagens e se dá conforme visto no Quadro 13.

Quadro 13 - Treinamento da rede

```
01  for (int iteration = 0;
02    iteration < train_iterations;
03    iteration++) {
04
05    for (int i = 0; i < sequences.size(); i++) {
06      image_sequence sequence = sequences.at(i);
07      distance_matrix input = input_vectors[i];
08      emotion_type e = sequence.emotion();
09
10      matrix<double> output(enabled_emotions.size(),1);
11      for (int emi = 0; emi < output.size(); emi++)
12        output(emi) = 0;
13
14      output(implemented_emotion_index(e)) = 1;
15      net.train(input, output);
16    }
17  }
```

O método `implemented_emotion_index` simplesmente busca o índice de uma emoção na lista de emoções definidas para treino pelo usuário. A variável `input_vectors` contém as matrizes de distância obtidas para cada `image_sequence` selecionada para o treinamento.

Após a execução do treinamento, são realizados testes sobre o grupo de validação. Cada imagem de validação tem suas características extraídas e classificadas pela rede treinada. Em seguida, o resultado da rede é comparado com o esperado. A partir destes testes é montada uma matriz de confusão, impressa no terminal de comando.

A execução de uma classificação, tanto para validação quanto para uso final do protótipo, é realizada simplesmente alimentando a rede com uma matriz de distâncias, e analisando a matriz retornada. Nessa matriz resultante, cada índice representa a pontuação de uma emoção. Como resultado final desta classificação, é escolhido o índice que possui o maior valor, e então é consultada a emoção na lista de emoções definidas pelo usuário.

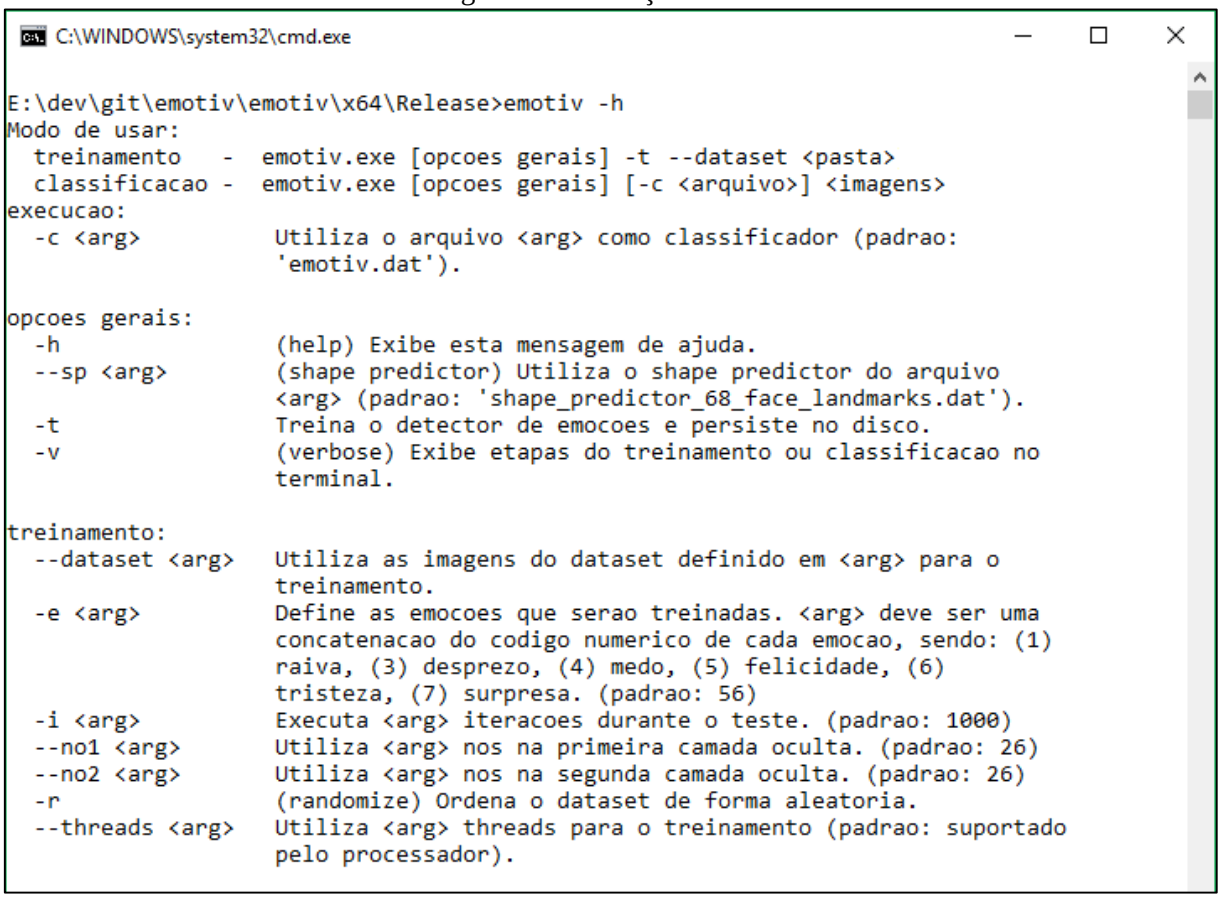
Ao final do treinamento, a rede é serializada em um arquivo `emotiv.dat`, para que a mesma possa ser utilizada na posterioridade apenas para classificação.

3.3.2 Operacionalidade da implementação

O protótipo é disponibilizado na forma de um executável, `emotiv.exe`, juntamente com os arquivos `emotiv.dat` e `shape_predictor_68_face_landmarks.dat`, explicados na seção anterior. O uso do protótipo é feito através de instruções passadas pelo terminal de comandos do sistema operacional. Para as demonstrações a seguir, foi utilizado o terminal nativo do sistema operacional Windows 10.

Visando manter um padrão com o que é esperado de programas de linha de comando, foi disponibilizada a opção `-h` (*help*), que exibe as instruções de uso do protótipo. A saída deste comando pode ser vista na Figura 18. Com base nela, serão descritas as formas de uso.

Figura 18 - Instruções de uso

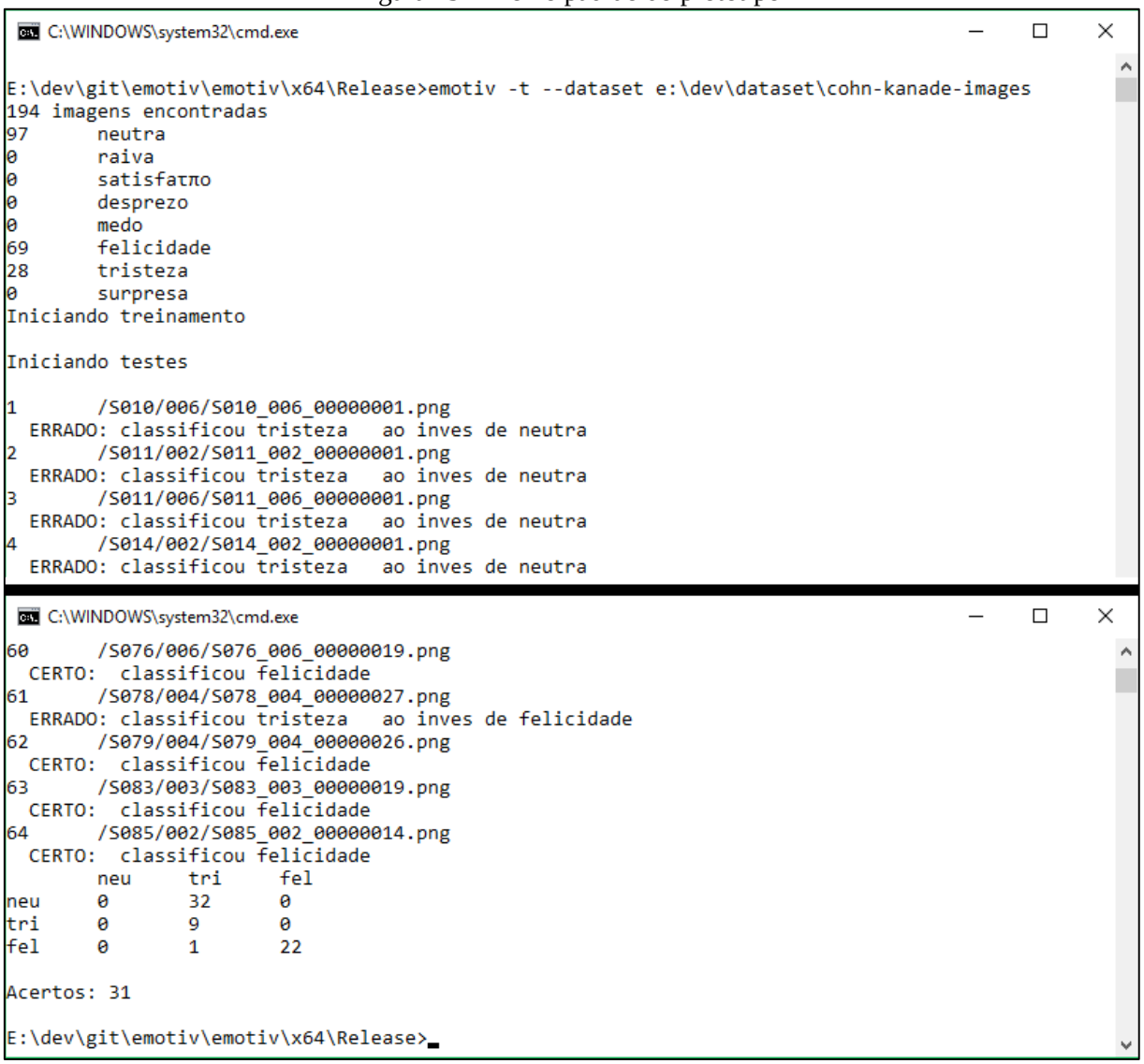


Conforme visto, existem duas formas de uso, que mapeiam os casos de uso `UC01` e `UC02` (Figura 12). Ambas as formas são descritas a seguir.

3.3.2.1 Treinamento do classificador

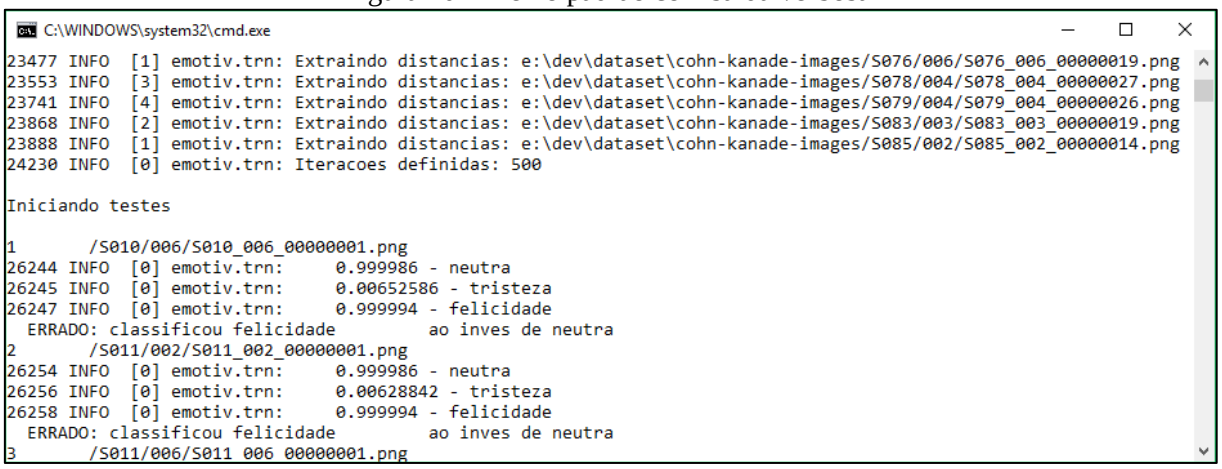
Para executar o treinamento da rede, e assim gerar um novo arquivo `emotiv.dat`, é necessário informar as opções `-t` e `--dataset`, sendo esta última seguida pela pasta raiz da base de testes, conforme descrito no Quadro 6. Um exemplo de saída da execução do programa apenas com estas opções é ilustrado na Figura 19.

Figura 19 - Treino padrão do protótipo



É possível, entretanto, executar o classificador com a opção `-v` para exibir um rastreamento das etapas de classificação, com informações de duração de tempo e *thread* em execução. A saída para a mesma instrução da figura anterior, porém com a opção `-v`, pode ser vista na Figura 20.

Figura 20 - Treino padrão com saída verbosa

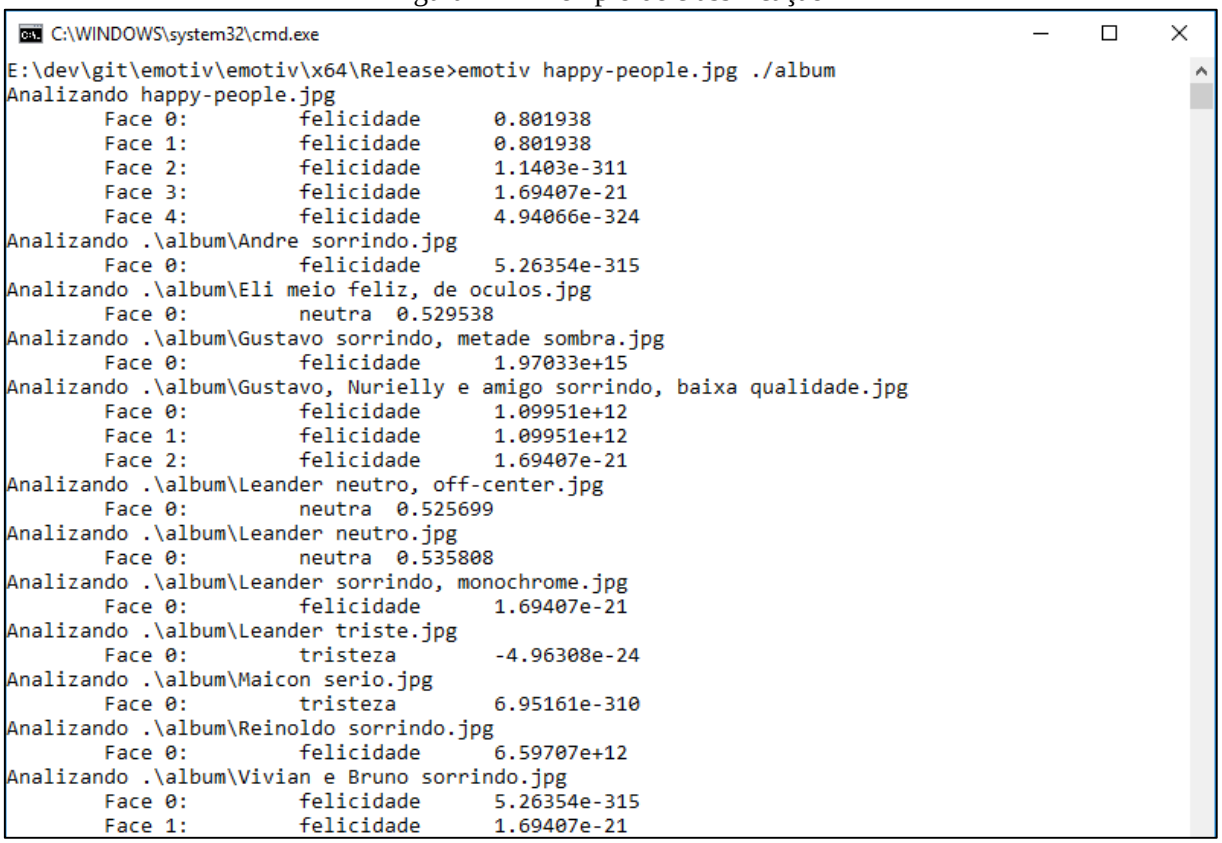


Nesta forma é exibido o tempo em milissegundos desde o início da execução do programa até o momento em que a mensagem foi emitida, possibilitando ter uma estimativa de desempenho do sistema. Informações como quantidade de iterações e nós ocultos nas camadas também são exibidas, assim como a pontuação de cada emoção na etapa de classificação.

3.3.2.2 Execução do classificador

Uma vez que um classificador tenha sido treinado, ou obtido de alguma outra fonte, é possível executar a classificação simplesmente fornecendo um ou vários caminhos de imagens e diretórios de imagens para o protótipo. Na Figura 21 foi executada a classificação em imagens não rotuladas, havendo entre as classificações imagens com múltiplas faces e tamanhos, com formato de arquivo diferente do usado no treinamento e com uma imagem monocromática.

Figura 21 - Exemplo de classificação



3.4 RESULTADOS E DISCUSSÕES

Para o treinamento do classificador foi utilizada a base de imagens CK+ (*Extended Cohn-Kanade Dataset*), a qual é composta por 593 sequências de imagens de 123 indivíduos (LUCEY et al., 2010). As sequências de imagens variam em duração de 10s até 60s, iniciando com uma expressão facial neutra e terminando com uma expressão universal. Além das imagens, são inclusos também rótulos de expressões faciais, de códigos de ação facial e de pontos das faces, fazendo desta uma base bem abrangente para o estudo de detecção de expressões faciais de indivíduos.

As imagens que compõem a base são, em sua maioria, monocromáticas, com resolução de 640 x 490 *pixels*. O restante das imagens são coloridas e possuem resolução de 640 x 480 *pixels*. A estrutura de diretórios na qual as imagens se encontram teve forte influência no desenvolvimento do protótipo, tendo sido descrita na seção 3.3.1.2 (Quadro 6).

Para a avaliação do protótipo, foi gerado um arquivo de comando em lotes do Windows (*batch file*) no qual várias configurações de classificação foram testadas. As configurações testadas são uma combinação de:

a) quantidade de emoções: foram testadas as combinações (1) neutra e felicidade, (2) neutra, felicidade e tristeza, e (3) neutra, raiva, felicidade e tristeza;

b) quantidade de nós ocultos na primeira camada: valores pares iniciando em 12 (metade do tamanho do vetor de entrada da rede) até o mínimo entre 24 e quatro vezes a quantidade de emoções;
c) quantidade de nós ocultos na segunda camada: valores pares iniciando com a mesma quantidade de emoções até a quantidade de nós da primeira camada;
d) quantidade de iterações: treinos com 200, 500 e 1200 iterações.

Devido ao grande volume de dados produzido e a ausência de uma ferramenta para processamento automático destes, optou-se por apresentar a combinação das três quantidades de iterações, juntamente com a maior e menor quantidade de neurônios em cada camada oculta. Na Tabela 3 é apresentado o desempenho da rede para as expressões neutra e felicidade.

Tabela 3 - Desempenho do protótipo para expressões Neutra e Felicidade

Iterações	Nós na 1ª camada	Nós na 2ª camada	Taxa de acerto
200	12	2	50%
200	12	12	50%
200	24	2	50%
200	24	24	50%
500	12	2	50%
500	12	12	50%
500	24	2	50%
500	24	24	98%
1200	12	2	50%
1200	12	12	98%
1200	24	2	50%
1200	24	24	50%

Os resultados para as emoções neutra, felicidade e tristeza são mostrados na Tabela 4.

Tabela 4 - Desempenho da rede para expressões Neutra, Felicidade e Tristeza

Iterações	Nós na 1ª camada	Nós na 2ª camada	Taxa de acerto
200	12	4	50%
200	12	12	50%
200	24	4	50%
200	24	24	50%
500	12	4	50%
500	12	12	50%
500	24	4	84%
500	24	24	89%
1200	12	4	50%
1200	12	12	50%
1200	24	4	50%
1200	24	24	50%

Por fim, a Tabela 5 demonstra o desempenho para classificação destas mesmas expressões, além da expressão facial de raiva.

Tabela 5 - Desempenho da rede para expressões Neutra, Raiva, Felicidade e Tristeza

Iterações	Nós na 1ª camada	Nós na 2ª camada	Taxa de acerto
200	12	4	74%
200	12	12	74%
200	24	4	50%
200	24	24	74%
500	12	4	67%
500	12	12	67%
500	24	4	67%
500	24	24	74%
1200	12	4	50%
1200	12	12	50%
1200	24	4	50%
1200	24	24	51%

Observa-se que a rede apresentou desempenho idêntico para diversas combinações. Nestes casos, os valores de saída correspondentes a cada emoção também não variaram entre uma classificação e outra (ficaram “fixos”). No Quadro 14 é apresentada a matriz de confusão para um caso de treinamento correto, das emoções neutra, felicidade e tristeza, com 24 neurônios em cada camada oculta, treinados em 500 iterações e que alcançou a taxa de reconhecimento de 89%.

Quadro 14 - Matriz de confusão para um treinamento bem-sucedido

	Neu.	Fel.	Tri.
Neu.	**28**	0	4
Fel.	1	**22**	0
Tri.	2	0	**7**

Para fins comparativos, é apresentada no Quadro 15 a matriz de confusão resultante do treinamento das emoções neutra, raiva, felicidade e tristeza, com 24 neurônios na primeira camada oculta e 4 na segunda, treinados em 500 iterações e com taxa de reconhecimento de 67%.

Quadro 15 - Matriz de confusão para um treinamento mal-sucedido

	Neu.	Rai.	Fel.	Tri.
Neu.	**47**	0	0	0
Rai.	15	**0**	0	0
Fel.	23	0	**0**	0
Tri.	9	0	0	**0**

O padrão apresentado no Quadro 15 se repete por uma grande parcela dos testes realizados, onde todas as avaliações são classificadas como uma única expressão facial. Não obstante, constatou-se que ao executar os testes com a configuração de sucesso novamente não há garantia que o desempenho resultante do classificador seja replicado, podendo até mesmo cair no padrão que resulta sempre na mesma classificação.

Foi considerada a possibilidade de existir algum erro na implementação da extração de características em múltiplas *threads*, porém o mesmo padrão de erro se repete quando o protótipo é executado com uma única *thread*.

3.4.1 Comparação com trabalhos correlatos

O Quadro 16 apresenta um comparativo entre o protótipo desenvolvido e os trabalhos correlatos.

Quadro 16 - Comparativo com os trabalhos correlatos

Características / Trabalhos	Oliveira e Jaques (2013)	Filko e Martinović (2013)	Tang e Huang (2008)	Protótipo desenvolvido
Emoções reconhecidas	6	6	6	2
Classificador	Redes neurais	Redes neurais	SVM	Redes neurais
Dimensão das imagens	2D	2D	3D	2D
Características classificadas	Distâncias entre pontos	*Eigenfaces*	Distâncias entre pontos	Distâncias entre pontos
Taxa de reconhecimento	89,87%	70%	87,1%	89%

Cabe notar que a contagem de emoções reconhecidas não inclui a expressão neutra, onde todos os trabalhos, bem como o protótipo, foram capazes de reconhecer. Para a taxa de reconhecimento deste trabalho foi utilizado o resultado do cenário no qual a classificação ocorreu normalmente. Nenhum dos trabalhos correlatos cita erros no classificador, como ocorrido no protótipo.

Pode-se observar que a classificação com base em distâncias entre pontos faciais específicos apresentou resultados melhores que a classificação por *eigenfaces*, realizada no trabalho de Filko e Martinović (2013).

A utilização de redes neurais para a classificação de expressões em imagens 2D se mostrou levemente superior à classificação por SVM em imagens 3D, que possuem mais dados disponíveis para análise.

É importante ressaltar, entretanto, que foram comparados os resultados da classificação de apenas duas expressões faciais universais deste trabalho contra o conjunto completo classificado pelos demais trabalhos. Seria necessário terem sido classificadas as mesmas emoções para poder realizar afirmações mais precisas.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um protótipo para a classificação de um subconjunto das expressões faciais universais. Foram utilizadas técnicas de processamento de imagens e visão computacional para a identificação de faces e reconhecimento de características, além da classificação em emoções através de uma rede neural artificial (RNA) multicamadas.

O protótipo resultante foi capaz de reconhecer as expressões neutra, felicidade e tristeza, mostrando a viabilidade das técnicas empregadas. Observou-se, também, que para a classificação de mais emoções se fazem necessários ajustes no classificador. Isto possivelmente se dá por conta de algum erro na utilização de ponteiros durante a implementação do protótipo – resultando em impurezas nos dados usados para treino – ou por existir um erro na implementação RNA utilizada.

Este trabalho serviu, também, como um estudo de caso da biblioteca dlib, onde sua aplicação para extração de características faciais se mostrou satisfatória. Não é possível concluir o mesmo a respeito da sua implementação de RNA, conforme citado anteriormente. Embora exista um exemplo de uso da rede no *site* oficial do desenvolvedor, este contempla a utilização da rede com apenas um nó de saída (DLIB, 2016). Não foram encontradas utilizações da rede com múltiplos nós de saída em outros trabalhos, nem mesmo em repositórios *online* de código aberto.

O protótipo desenvolvido foi capaz de atender os casos de uso propostos e, caso seja resolvido o impedimento em relação ao classificador, se mostra promissor para sua utilização genérica. Adicionalmente, o protótipo disponibiliza uma interface de linha de comando abrangente e relativamente simples de ser utilizada, facilitando o uso por usuários humanos e também por sistemas automatizados, seja para experimentação de diferentes configurações de treino ou para a classificação das expressões faciais em imagens.

4.1 EXTENSÕES

Dentre as possíveis extensões para este trabalho, pode-se citar:

a) alterar o classificador de características de expressões faciais para uma implementação mais confiável ou mesmo outra técnica – a própria biblioteca utilizada disponibiliza outros algoritmos de aprendizagem de máquina mais modernos (KING, 2009);
b) permitir que o protótipo, em seu modo de classificação, continue em execução até que o utilizador, humano ou robótico, informe o contrário, assim se torna mais viável

a utilização do protótipo como um componente de outros sistemas;

c) exibir as coordenadas e dimensões das faces detectadas pelo protótipo;
d) realizar a classificação das demais expressões faciais universais;
e) embutir a rede serializada de uma etapa de treinamento de sucesso junto ao executável final, dispensando o utilizador de ter que realizar o treinamento ou de utilizar um arquivo `emotiv.dat`.

REFERÊNCIAS

CACOO. Your ideas. Our canvas. **CACOO**. Disponível em: <https://cacoo.com/>. Acesso em: 24 junho 2016.

DALAL, N.; TRIGGS, B. Histograms of Oriented Gradients for Human Detection. In: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'05), 5., 2005, San Diego. **Proceedings…** [S/l]: IEEE Computer Society. 2005. v. 2. p. 886-893.

DLIB. **dlib C++ Library**, 2016. Disponível em: <http://dlib.net>. Acesso em: 3 de junho 2016.

EKMAN, P. Basic Emotions. In: DALGLEISH, T.; POWER, M. **Handbook of Cognition and Emotion**. Sussex: John Wiley & Sons, 1999a. p. 45-60.

_____. Facial expressions. In: DALGLEISH, T.; POWER, M. **Handbook of Cognition and Emotion**. Sussex: John Wiley & Sons, 1999b. p. 301-320.

EKMAN, P.; FRIESEN, W. V. Nonverbal Leakage and Clues to Deception. **Journal for the Study of Interpersonal Process**, [Washington, DC, USA], v. 32, n. 1, Feb. 1969. 88-106.

_____. **Facial action coding system**. Palo Alto, CA: Consulting Psychologists Press, 1978.

EKMAN, P.; SORENSON, E. R.; FRIESEN, W. V. Pan-Cultural Elements in Facial Display of Emotions. **The American Association for Advancement of Science**, [S/l], v. 164, Apr. 1969. 86-88.

FILKO, D.; MARTINOVIĆ, G. Emotion Recognition System by a Neural Network Based Facial Expression Analysis. **Automatika**, [S/l], v. 4, 2013. 263-272.

FORSYTH, D. A.; PONCE, J. **Computer Vision:** A Modern Approach. 1ª. ed. London, UK: Prentice Hall, 2003.

_____. **Computer Vision:** A Modern Approach. 2ª. ed. Boston, USA: Pearson, 2012.

GELLER, T. How do you feel? Your computer knows. **Communications of the ACM**, New York, USA, v. 57, p. 24-26, Jan. 2014. ISSN 1.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais**. Tradução de Roberto Marcondes Cesar Junior e Luciano da Fontoura Costa. São Paulo, SP: Edgard Blücher, 2000.

KAZEMI, V.; SULLIVAN, J. **One Millisecond Face Alignment with an Ensemble of Regression Trees**. 2014 IEEE Conference on Computer Vision and Pattern Recognition. Columbus, OH: IEEE. 2014. p. 1867-1874.

KING, D. E. Dlib-ml: A Machine Learning Toolkit. **Journal of Machine Learning Research**, [S/l], v. 10, Jul. 2009. 1755-1758.

LUCEY, Patrick et al. The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. In: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION - WORKSHOPS, 2010, San Francisco. **Proceedings...** [S/l]: IEEE Computer Society, 2010. p. 94-101.

MARQUES FILHO, O.; VIEIRA NETO, H. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999.

MARQUES, J. S. **Reconhecimento de Padrões:** métodos estatísticos e neuronais. 2ª. ed. Lisboa: IST Press, 2005.

NORVIG, P.; RUSSEL, S. **Inteligência Artificial**. 3ª. ed. [S.l.]: CAMPUS, 2013.

OLIVEIRA, E.; JAQUES, P. A. Classificação de emoções básicas através de imagens capturadas em vídeos de baixa resolução. **Revista Brasileira de Computação Aplicada**, Passo Fundo, RS, v. 5, out. 2013. 40-54.

PICARD, R. W. **Affective computing**. Cambridge, USA: MIT Press, 1997.

PORTER, S.; BRINKE, L. T. Reading between the Lies: identifying concealed and falsified emotions in universal facial expressions. **Psychological Science**, Washington, DC, US, v. 19, n. 5, May 2008. 508-514.

SANDBACH, G. et al. Static and dynamic 3D facial expression recognition: A comprehensive survey. **Image and Vision Computing**, [S/l], 30, Oct. 2012. 681-796.

SCHMIDT, K. L.; COHN, J. F. Human facial expressions as adaptations: Evolutionary questions in facial expression research. **Yearbook of Physical Anthropology**, [S/l], v. 44, 2001. 3-24.

SHIRAI, Y. **Three-dimensional computer vision**. Berlin: Springer-Verlag, 1987.

SINDHURI, K.; RAJU, K. V. Pervasive Computing. **International Journal of Engineering Trends and Technology (IJETT)**, [S/l], v. 4, May 2013. 1601-1608.

SZELISKI, R. **Computer Vision:** Algorithms and Applications. London: Springer-Verlag, 2011.

TANG, H.; HUANG, T. S. 3D facial expression recognition based on properties of line segments connecting facial feature points. In: 8TH IEEE INTERNATIONAL CONFERENCE ON AUTOMATIC FACE & GESTURE RECOGNITION (FG'08), 8., 2008, Amsterdam. **Proceedings…** [S/l]: IEEE Computer Society. 2008. p. 1-6.

TIMOFEEV, R. **Classification and Regression Trees (CART) Theory and Applications**. Humboldt University. Berlin. 2004.

VIOLA, P. A.; JONES, M. J. Rapid object detection using a boosted cascade of simple features. **IEEE Computer Society**, [S/l], v. 1, 2001. 511-518.