UNIVERSIDADE DE SANTA CRUZ DO SUL
DEPARTAMENTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Rodrigo Wink

SISTEMA DE RECONHECIMENTO DE FACES EM VÍDEO

Santa Cruz do Sul, novembro de 2015.

Rodrigo Wink

SISTEMA DE RECONHECIMENTO DE FACES EM VÍDEO

Trabalho de Conclusão II apresentado ao Curso de Ciência da Computação da Universidade de Santa Cruz do Sul, para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Rolf Fredi Molz.

Santa Cruz do Sul, novembro de 2015.

Dedico este trabalho a minha família, a meu pai Flávio, exemplo de dedicação e fortaleza, a meus amados irmãos César e Fernanda, e em especial a minha amada mãe Rosane, por toda a força, carinho, dedicação e apoio durante os momentos difíceis.

AGRADECIMENTOS

Agradeço o apoio de amigos, colegas e professores que participaram nesta jornada até a conclusão do curso. Agradeço ao orientador deste trabalho, Rolf, que apesar de todos os atrasos e contratempos de minha parte, sempre teve palavras de apoio e incentivo, que foram fundamentais para chegar até aqui.

RESUMO

O reconhecimento de faces de forma automatizada é uma área tecnológica cuja importância cresce com a criação de sistemas de segurança e monitoria em resposta a atentados terroristas. Também possui utilidade nas áreas de entretenimento e comércio, onde permite automatizar e agilizar processos. Este trabalho visa utilizar *Support Vector Machines* (SVM) na tarefa de reconhecimento de faces automatizado em vídeo. Tal processo engloba três etapas principais, a detecção de rostos, extração de características e o posterior reconhecimento, que são aprofundados neste trabalho, embora o foco se encontre no problema do reconhecimento e aplicação de SVM para o mesmo. A função do SVM é classificar corretamente uma face detectada, a partir de imagens de rostos obtidas de vídeo. Para tanto, são abordados três métodos de classificação multi-classe baseados em SVM, respectivamente: um-contra-um, um-contra-todos e SVM de Grafo Dirigido Acíclico DAGSVM (*Directed Asiclic Graph SVM*). É explorado para detecção de faces uma implementação do método Detector de faces Viola-Jones chamado *Haar Feature-based Cascade Classifier*, que foi utilizado na criação do sistema de reconhecimento. Especificamente, foi utilizada a versão implementada pela biblioteca *Open Source Computer Vision* (OpenCV). Para à extração de características, foram utilizados métodos de pré-processamento sobre a imagem inteira da face. Para a etapa de reconhecimento, foi utilizada a biblioteca LIBSVM, que possui a implementação de métodos de treinamento e previsão para classificadores binários SVM. Testes gerados através do sistema, sobre base de imagens de faces *The AT&T Database of Faces*, mostraram que DAGSVM é superior às alternativas.

Palavras chave: *Support Vector Machines*, classificação multi-classes, Reconhecimento de faces, Detecção de faces.

ABSTRACT

Automated face recognition is a technological area whose importance grows with the creation of security and monitoring systems in response to terrorist attacks. Its utility also lies in the areas of entertainment and commerce, where it allows for the automatization and simplification of processes. This work approaches the application of Support Vector Machines (SVM) to the task of automated facial recognition in video. Such process can be divided in three main parts, detection, feature extraction and recognition. Those three are further detailed in this work, although the main focus is in the problem of recognition and the application of SVM to it. The function of the SVM is to classify correctly a detected face from face images obtained from video. For that end, three multi-class classification methods based in SVM are studied, respectively: one-against-one, one-against-all and DAGSVM (Directed Acyclic Graph SVM). For face detection the Face Detector Viola-Jones method is explored, whose implementation called *Haar Feature-based Cascade Classifier* was used in the development of the recognition system. Specifically was utilized the version implemented in the Open Source Computer Vision (OpenCV) library. For the feature extraction step, preprocessing methods were used against the whole face image. For the recognition stage the LIBSVM library was used. It contains the implementation of the training and prediction methods for the SVM binary classifier. Tests using *The AT&T Database of Faces*, showed that DAGSVM was superior to the alternatives.

Keywords: Support Vector Machines, multi-class classification, Face Recognition, Face Detection.

SUMÁRIO

1. INTRODUÇÃO

O reconhecimento de faces por computador é um assunto que traz interesse, visto que não só está presente no meio acadêmico, comercial e militar, como em diversos itens de entretenimento, de filmes a seriados, como por exemplo, *Person of Interest*, que tem por foco especifico esse assunto. No meio acadêmico é uma área madura, cuja aplicabilidade como tecnologia se torna viável, com crescente redução de margens de erros de reconhecimento (ZHAO et al, 2003).

O reconhecimento de faces em vídeo é similar ao de imagens estáticas, porém tende a possuir menor qualidade. A diferença se encontra em ter acesso a múltiplas imagens sequenciais, e utilizar da diferença entre cada uma para encontrar padrões. Outro fato é a tendência a uma menor qualidade e controle das imagens obtidas. (ZHAO et al, 2003)

De acordo com (DEGTYAREV, 2010), detectar faces e reconhecê-las é um problema de crescente importância, visto que está relacionado diretamente ao desenvolvimento de sistemas de segurança pública, criados em resposta a recentes atentados terroristas. Outras áreas como entretenimento e mídias sociais, encontram utilidade na tecnologia para facilitar a identificação de pessoas automaticamente, por exemplo, em álbuns de fotografia. (HUA et al, 2011).

O objetivo do trabalho é reconhecer indivíduos através de sua face, a partir de imagens obtidas de vídeos de câmeras. Basicamente, o processo de reconhecimento facial pode ser dividido nas etapas detecção de face, extração de características e reconhecimento. Considerando tratar-se de uma área madura, múltiplas técnicas de resolução existem para cada etapa. Para detecção, será utilizado uma variante do método detector de faces Viola-Jones, a partir de uma biblioteca pré-existente (OpenCV), chamado *Haar Feature-based Cascade Classifier*. A base do mesmo é um método padrão da área, sendo responsável por tornar viável a detecção de faces em sistema comerciais, como expresso por (Zhang, C; Zhang, Z, 2010). Com relação a etapa de reconhecimento, este trabalho explorará a utilização de *Support Vector Machines*, criada por Vapnik e Cortez (1995). Tal técnica trata de um classificador linear, cujo diferencial está em permitir a separação de dados em espaços de dimensões superiores, causando baixo custo computacional adicional.

Tendo em vista estes fatores, o trabalho irá comparar as taxas de acerto entre os três diferentes classificadores multi-classes: um-contra-um, um-contra-todos e DAGSVM.

Também será explorada a utilização de pré-processamentos para preparação das imagens para o reconhecimento.

Para tanto, o trabalho está assim distribuído: no capítulo 2 é apresentada a justificativa para este trabalho, a importância da área e algumas das aplicações possíveis. O capítulo 3 apresenta o objetivo deste trabalho. No capítulo 4 é apresentado o referencial teórico. Primeiramente é aprofundada a área de detecção de faces em 4.1, seus desafios e classes de métodos existentes. É explicado o método Detector de Faces Viola-Jones. Em seguida, em 4.2, é tratado reconhecimento de faces, desafios da área e exemplos de métodos para sua realização. O subcapítulo 4.3 explica o que é o classificador linear binário *Support Vector Machines*. Já 4.4 explica métodos para utilizar o SVM em classificações multi-classe. O capítulo 4.5 apresenta trabalhos relacionados. O capítulo 5 apresenta o sistema desenvolvido. Em 5.1 é comentada a proposta do sistema e os detalhes de implementação e processos de execução. Em 5.2 são apresentados os resultados obtidos e após, a conclusão deste trabalho é apresentada.

2. JUSTIFICATIVA

O reconhecimento de faces é uma área em constante desenvolvimento, e madura principalmente no que diz respeito a fotos controladas (pose, iluminação, qualidade, distância) (ZHANG et al, 2015).

Existe demanda por algoritmos mais robustos para reconhecimento de faces em imagens "na natureza". Esta demanda é levada por dois grandes focos. O primeiro é a vontade de usuários de anotar ou marcar suas fotos digitais, para fins de facilitar a organização, o acesso e o compartilhamento online de seus álbuns pessoais. O segundo é a necessidade por companhias de segurança em reconhecer pessoas a partir de câmeras de vigilância e em situações não controladas, de forma robusta e confiável (HUA et al, 2011).

O reconhecimento de faces também possui utilidade na área de biometria, onde sugerido por (ZHAO et al, 2003), traz facilidades, pois não depende da cooperação e conhecimento do usuário, ao contrário de reconhecimento por íris ou digitais, simplificando o processo.

Esta facilitação do processo biométrico leva a aplicabilidade do reconhecimento facial para a área de controle de acesso, que consiste na verificação da identidade de indivíduo a fim de restringir o acesso a recursos. (SENIOR, 2002).

Porém (Senior, 2002) ressalta que a utilização para controle de acesso tem se restringindo a poucos domínios. Isso pelos sistemas de verificação não serem confiáveis e previsíveis o suficiente para substituir a segurança trazida pelas combinações de senhas e outros mecanismos físicos. No caso de serviços onde a tolerância a erros de identificação precisa ser minimizada, como bancos, a tecnologia atual não é confiável o suficiente. Para ser aplicável a estes casos, sistemas robustos de aquisição passiva e com probabilidade mínima de rejeitar falsamente um indivíduo, precisam ser desenvolvidos.

Outra aplicação são os sistemas de identificação. (SENIOR, 2002) comenta sobre o uso destes sistemas por dois estados americanos (Massachusetts e Connecticut) para controle de distribuição de benefícios ao cidadão pelo estado (*Welfare*). O problema enfrentado é o múltiplo cadastro do mesmo indivíduo. O reconhecimento de faces é utilizado para comparar o rosto recém registrado contra os já presentes em suas bases, a fim de impedir duplo cadastro. O reconhecimento por si ainda não é robusto o suficiente, o que leva a utilização de informações extras, como *Zip Code*, idade e nome, para otimizar a busca, além da presença de um operador humano para verificar a possibilidade de falso alarmes.

Serviços de vigilância estão começando a se tornar uma realidade presente, não sendo incomum a presença de câmeras de monitoria em estabelecimentos e ruas. Para realmente possuir efetividade em tempo real, tais serviços requerem a presença de operadores, que são responsáveis por reconhecerem indivíduos ou acontecimentos de interesse. Essa necessidade implica em restrições físicas e financeiras para o monitoramento em tempo real em larga escala, apesar da ampla existência de dispositivos de captura.

Devido a riqueza de informações que produz, vídeo é o meio de escolha para sistemas de vigilância. A aplicação de reconhecimento facial como biometria em vídeo, para sistemas que necessitem de identificação, é a melhor escolha, devido ao tipo de dados fornecidos pelo meio. É por isso que vigilância é um campo onde se demonstra maior interesse pela área de reconhecimento faces (SENIOR, 2002).

A necessidade crescente por monitoramento, aliada a ampla disponibilidade de captura, faz com que seja interessante permitir que computadores possam realizar o reconhecimento de pessoas em tempo real. Isto auxiliaria no rápido reconhecimento de indivíduos, não necessitando de operadores humanos.

O reconhecimento automático de faces pode ser utilizado para monitorar "pessoas de interesse", ao vivo, e sem o conhecimento e participação ativa do sujeito. Também possui utilidade em descobrir a identidade pós fato, utilizando registros de vídeo de um crime e vasculhando por uma base de suspeitos (SENIOR, 2002).

Outra área de aplicação é a computação pervasiva ubíqua. Trata-se do desenvolvimento de dispositivos dotados de sensores e aplicações capazes de comunicarem-se e existindo de forma difusa e interconectada no ambiente. Parte de seu funcionamento pode exigir a percepção do meio ambiente em que se encontra, a fim de fornecer o contexto adequado a possíveis usuários. A necessidade de que a experiência com tal forma de computação aconteça de forma produtiva, controlada, prática e pouco invasiva, traz consigo a necessidade de os sistemas serem "cientes da presença de humanos", do inglês *human-aware*, capazes de identificar cada usuário. Neste contexto, dentro dos sistemas de biometria existentes, o reconhecimento de faces possibilita uma alternativa de maior passividade para a identificação de pessoas, e por consequência, ter o contexto adequado fornecido as mesmas (SENIOR, 2002).

Em referência às tecnologias existentes, os avanços na área de reconhecimento de faces levaram a capacidades de acerto, contra álbuns de fotos obtidos em natureza (do inglês *in the wild*), de mais de 80% (ZHANG et al, 2015). Métodos utilizados podem requerer apurados

processos para segmentação (localização e obtenção) de faces nas fotos, aprendizado de máquina guiado contra múltiplas poses pré-definidas diferentes (do inglês *poselets*), utilização de *convolutional neural networks* para gerar um identificador global sobre os *poselets*, e aprendizado com *Support Vector Machines* sobre múltiplas *features* (carcterísticas, dados relevantes extraídos da imagem) relacionadas a imagem que contém as faces (ZHANG et al, 2015). Já o reconhecimento de faces em vídeo ainda não está tão apurado, devido a dificuldades quanto à qualidade e natureza da situação não controlada dos vídeos em natureza (WOLF et al, 2011).

Dentre as técnicas possíveis para detecção ou reconhecimento de faces, existem as que se utilizam de Aprendizado de Máquina. *Support Vector Machines* (SVM) é uma destas, as quais tem por natureza a capacidade de modelar uma equação matemática subjacente, um padrão, extraído do conjunto de dados de objetos de interesse. Este pode então ser utilizado para classificar novos objetos apresentados. Entende-se que faces humanas em imagens possuem padrões que podem ser modelados matematicamente e, portanto, são passíveis de serem utilizadas com sucesso por essas técnicas para a classificação.

A SVM possui a capacidade inerente de resolver problemas de classificação de padrões (*pattern-matching*), tendendo a se aproximar da melhor solução possível, sem necessitar da utilização de conhecimento de domínio do problema de interesse em seu design (KABEER, 2011).

3. OBJETIVOS

O principal objetivo deste trabalho é desenvolver uma aplicação para testar o reconhecimento de faces a partir de imagens obtidas de vídeo empregando SVM.

Especificamente, são pesquisadas referências bibliográficas e trabalhos relacionados que circundam o Trabalho de Conclusão. Também são definidas bases de faces e vídeos para uso como *benchmark*, e as tecnologias necessárias para extração de imagens do vídeo e subsequente segmentação das faces.

4. FUNDAMENTAÇÃO TEÓRICA

Primeiramente será abordada a questão da detecção de faces, suas dificuldades e a classificação de métodos possíveis para sua resolução. Em seguida, é explicado um dos métodos mais utilizados na área, o detector de faces Viola-Jones.

4.1. Detecção de faces e imagens em vídeo

Yang et al 2002, p. 13 fornece a seguinte definição para o problema de detecção de faces: "Dada uma imagem arbitraria, o objetivo de detecção de faces é determinar se existe ou não alguma face na imagem e, se presente, retornar a localização e dimensão de cada face na imagem" (13, Yang, 2002).

De acordo (DEGTYAREV, 2010, p.1), detecção de faces é uma área de pesquisa madura, onde a maioria dos pesquisadores considera o problema resolvido. Sua aplicação ocorre em áreas como interface com usuários, serviços de entretenimento, a indústria de propaganda e sistemas de segurança. Ainda, (Zhang, 2010, p.12) comenta que a detecção de faces é uma das técnicas fundamentais que permitem a realização da interação humano-computador de forma natural.

Além da utilização no reconhecimento de faces, onde é necessariamente o primeiro estágio do processo (DEGTYAREV, 2010, p.1), a detecção também é crucial para algoritmos de análise facial. Nestes, incluem-se os de alinhamento, modelagem, reiluminação e verificação de faces, além de rastreamento da posição da cabeça, reconhecimento de expressões faciais e de idade e gênero (ZHANG, 2010).

Uma forma simplificada do problema de detecção, e que pode ser considerado um subproblema do mesmo, é a localização. Esta consiste em encontrar um rosto em uma imagem, onde sabe-se que esta imagem possui um, e apenas um rosto. A detecção em si abarca encontrar rostos sem conhecimento de quantidade e pré-existência dos mesmos.
(YANG et al, 2002)

As dificuldades encontradas na detecção de faces podem ser classificadas, de acordo com Yang et al 2002, em seis categorias principais:

Pose: a posição relativa de uma face em relação a câmera pode gerar imagens diferentes do mesmo rosto (frontal, 45 graus, perfil, de cabeça para baixo). Também pode ocorrer de feições estarem apenas parcialmente vivíveis, ou mesmo totalmente encobertas (ocluídas).

Presença ou ausência de componentes estruturais: além de detalhes estruturais próprios do rosto, existem características extras que podem estar presentes, como barbas, bigodes e óculos. Existe igualmente grande variação destas, seja em cor, tamanho e formato.

Expressões faciais: estas afetam diretamente a aparência de uma pessoa.

Oclusões: imagens podem conter rostos que estão apenas parcialmente visíveis, ou mesmo tapados (ocluídos). Além de objetos em frente ao rosto, pode comumente ocorrer em fotos de grupos de pessoas, onde um oculta a outra de forma parcial.

Orientação da imagem: diferentes ângulos do eixo óptico de uma câmera levam diretamente a variações na imagem de rostos.

Condições de obtenção da imagem: a aparência de um rosto pode ser alterada por fatores de formação da imagem. Estes incluem características da câmera, como resposta do sensor e lentes, e de iluminação, como espectro, distribuição e intensidade da origem.

Ainda, detectores podem fazer dois tipos de erros. *Falso negativo*, onde faces não são detectadas, resultando em baixas taxas de detecção e *falso positivo*, onde uma região da imagem é declarada como sendo uma face, porém não é.
(YANG et al, 2002)

Para lidar com esses problemas, uma série de métodos foram desenvolvidos. Geralmente, são técnicas para detectar faces em imagens de intensidade única ou em cores. Estas podem, de acordo com (YANG et al, 2002), ser classificadas em quatro categorias principais, sendo que alguns métodos pertencem a mais de uma.

Métodos baseados em conhecimento: são métodos baseados em regras que codificam conhecimento humano do que constitui uma face típica. Geralmente as regras capturam as relações entre características faciais. Esses métodos foram projetados principalmente para o problema de localização de faces.

Abordagens a partir de características invariantes: esses métodos visam encontrar características estruturais que existem mesmo quando a pose, ponto de vista ou condições de iluminação variam, e então utilizá-las para localizar faces. Esses métodos foram projetados principalmente para o problema de localização de faces.

Métodos de correspondência de modelo: diversos padrões *standard* de um face são guardados para descrever a face como um todo ou os atributos faciais separadamente. As correlações entre uma imagem de entrada e os padrões guardados são computadas para realizar a detecção. Esses métodos têm sido utilizados tanto para problemas de localização quanto detecção.

Métodos baseados em aparência: em contraste com os métodos de correspondência de modelo, os modelos são aprendidos de um conjunto de imagens que deveriam capturar a variabilidade representativa da aparência facial. Esses modelos aprendidos são então utilizados para a detecção. Estes métodos são projetos principalmente para problemas de detecção facial. (YANG et al, 2002)

4.1.1. Detector de faces Viola-Jones

Dos métodos descobertos, (ZHANG, 2010) comenta que aquele que tornou viável e competitiva a detecção de faces por sistemas é o "detector de faces Viola-Jones", desenvolvido por (VIOLA, P; JONES, M. 2001). Novas melhorias foram realizadas sobre o método, como por exemplo em Lienhart e J. Maydt, 2002, que adicionaram *Haar-like features* e Imagens Integrais diagonais, sendo esta a versão disponível pelo pacote OpenCV que será utilizado neste trabalho (Comentado na sessão Sistema Proposto).

Existem quatro pontos importantes para o algoritmo:

4.1.1.1. *Haar-like features*

São estruturas retangulares utilizadas para reconhecer *features* (características específicas) em uma imagem. Cada uma é dividida internamente em retângulos menores e, ao ser utilizada sobre uma imagem, os valores dos pixels pertencentes a área de cada parte são somados. Então é calculada a diferença entre as partes (subtração), e se o valor ultrapassar determinado limite, é considerado que foi encontrada a *feature* em questão. Exemplos de *haar-like features* propostas por (VIOLA, P; JONES, M. 2001) estão na figura 1 a seguir.

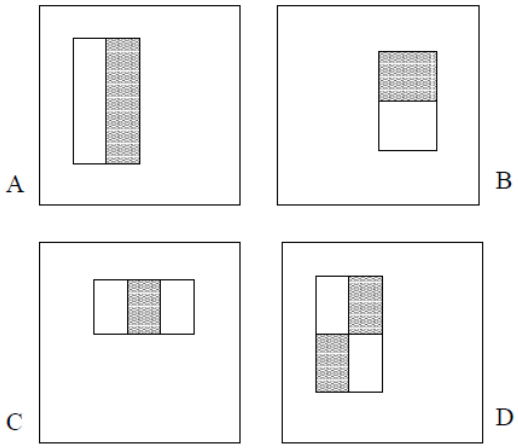


Figura 1. Exemplos de *Haar-like features* propostos por (VIOLA, P; JONES, M. 2001). Cores preto e branco são utilizadas apenas para diferenciar cada área interna. Fonte (VIOLA, P; JONES, M. 2001).

Sua utilização no processo consiste em passá-las, com escalas e posições diferentes, múltiplas vezes contra a imagem integral, para tentar reconhecer uma característica.
O conceito de imagem Integral é explicado em seguida, sendo que sua utilidade se encontra em otimizar os múltiplos cálculos somatórios de áreas das *Haar-like features*.

4.1.1.2. Imagem integral

A imagem Integral permite calcular o valor do somatório de uma área de pixels de um retângulo, necessitando apenas acessar quatro valores em memória. Basicamente, é criada uma versão da imagem onde cada pixel passa a conter o somatório da área retangular de pixels acima e a esquerda, até a origem. (VIOLA, P; JONES, M. 2001).

Tendo esta nova imagem, o cálculo do somatório de valores de pixels para diversas áreas retangulares é agilizado, como explicado a seguir.

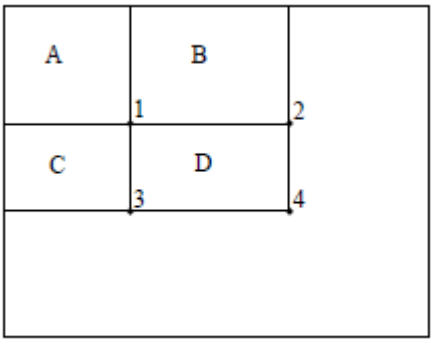


Figura 2 – Fonte: (Viola; Jones, 2001).
"A soma dos pixels dentro do retângulo D pode ser computada com apenas 4 referências a *arrays*. O valor da Imagem Integral no local 1 é a soma dos pixels no retângulo A. O valor no local 2 é A + B. no local 3 é A + C, e no local 4 é A + B + C + D. A soma dentro de D pode ser computada como 4 + 1 – (2 + 3)".

Como observado na Figura 2, a soma da área D pode ser calculada apenas utilizando os valores nos locais 1, 2, 3 e 4. Basicamente, é subtraído de 4 os valores de 2 e 3, sendo 1 adicionado pois é removido duas vezes (ambos 2 e 3 o removem).

4.1.1.3. AdaBoost e *Cascading*

É um algoritmo utilizado para melhorar o desempenho de classificação de algoritmos de aprendizado simples (também chamados de *weak*, fracos em português). Para o detector de faces Viola-Jones, é utilizado tanto para treinar o classificador quanto para selecionar um conjunto pequeno de *features*. No caso, é necessária a seleção de poucas pois o número de cálculos de áreas possíveis excede o número de pixels da imagem (Viola, Jones, 2011 comentam 180000 cálculos), tornando a computação proibitivamente cara.

O último ponto relevante é o uso de *Cascading*, que consiste uma técnica que permite selecionar apenas as *features* promissoras para treinamento, reduzindo o custo operacional.

4.2. Reconhecimento de faces

O reconhecimento de faces em imagens ou vídeos é uma das bem sucedidas aplicações de análise e entendimento de imagem, com múltiplas conferências voltadas ao tema (ZHAO et al, 2003). Um dos motivos para este sucesso está nas aplicações possíveis, que contemplam as áreas de segurança e comércio, possibilitadas graças aos avanços tecnológicos, frutos de décadas de pesquisa na área (ZHAO et al, 2003).

O problema do reconhecimento facial pode ser resumido, de forma generalizada, como citado em (ZHAO et al, 2003): “Dadas imagens de fotografias ou de vídeos de uma cena, identificar ou verificar uma ou mais pessoas na cena utilizando uma base de dados de faces”. Informações extras podem ser utilizadas para melhorar a identificação, tais como características da pessoa a ser identificada: gênero, idade, etnia, expressão facial e padrão de fala. Estes dados extras permitem diminuir o espaço de pesquisa na base de dados por faces associadas a tais características.

A solução para o problema passa por fases distintas, como mostrado na Figura 3. Primeiro, a detecção de faces a partir de cenas, onde as faces podem se encontrar em posições variadas, parcialmente visíveis ou em meio a diferentes objetos, o que dificulta sua obtenção. Após, é realizada a extração de características das regiões de faces obtidas. Por fim, a partir das características, realiza-se a identificação ou verificação (ZHAO et al, 2003). Identificação é o problema de submeter a face desconhecida e suas características a uma comparação com faces conhecidas na base de dados. Já verificação consiste em comparar a face obtida a uma face específica, validando ou rejeitando a correspondência.

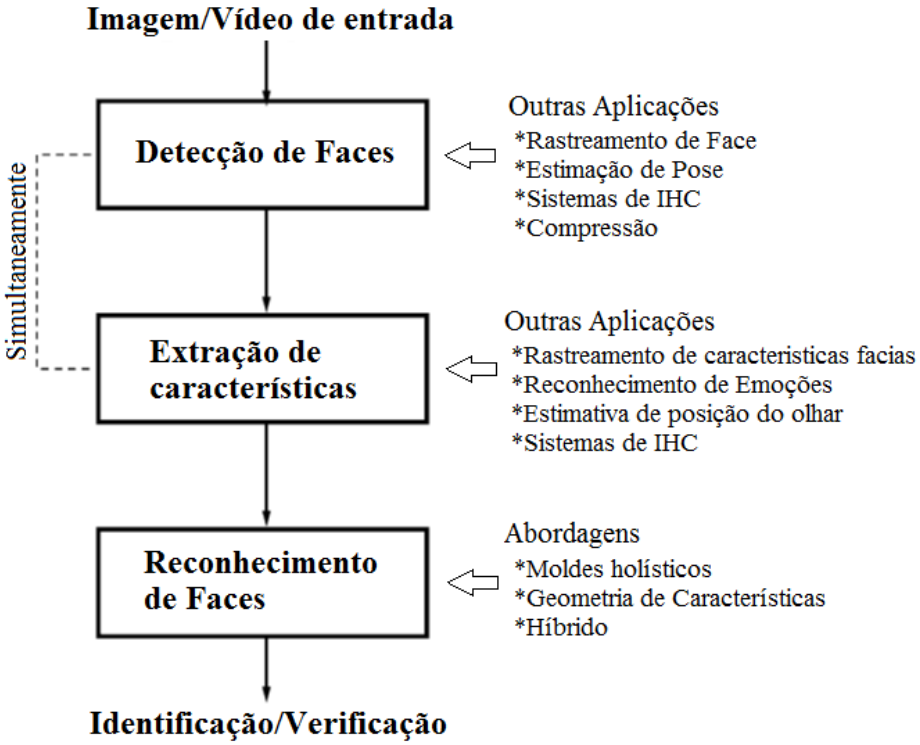


Figura 3 - Configuração de um sistema de reconhecimento de faces genérico.
Fonte traduzida de (ZHAO et al, 2003)

Aplicações envolvem tanto reconhecimento em fotos, com pose, distância e iluminação controladas, até imagens obtidas de vídeo, sem nenhum mecanismo de controle. Esta ampla variação nas condições do problema requer o uso de muitas técnicas distintas, de áreas como processamento, análise, entendimento e reconhecimento de padrões de imagens. As dificuldades e técnicas envolvidas com as aplicações podem ser classificadas em dois grandes grupos, o corresponder (do inglês "*matching*") para imagens estáticas e o para dinâmicas (vídeo). Cada um dos grupos também possui variações internas, como a qualidade da imagem, a quantidade de desordem do fundo (que afeta o desempenho de algoritmos de segmentação), se existe um critério bem definido para o *matching* e o tipo, natureza e disponibilidade de *input* pelo usuário (ZHAO et al, 2003).

No artigo apresentado por (Guo et al, 2000) há a explicação da possibilidade de que variações de expressão, iluminação e posição de uma mesma face em imagens distintas, podem chegar a ser "superiores a variação entre faces diferentes". Esta situação acaba sendo um grande desafio para a tarefa de reconhecimento.

Dela também surgem dois problemas principais. O primeiro é referente à quais características extrair da face, e como representá-las de modo a conseguir lidar com todas as variações possíveis na imagem. O segundo é como classificar uma imagem utilizando a nova representação.

O problema do reconhecimento de faces pode ser formulado como reconhecer objetos de três dimensões (3D) a partir de imagens de duas dimensões (2D). Reconhecer um objeto 3D a partir de suas imagens 2D trás muitos desafios. Os problemas de pose ou iluminação são duas dificuldades proeminentes para abordagens baseadas em aparência ou imagem. Muitas abordagens foram propostas para lidar com esses problemas, sendo que a maioria deles utiliza de conhecimento de domínio. (ZHAO et al, 2003)

4.2.1 Tipos de algoritmos de reconhecimento

Técnicas de reconhecimento podem ser classificadas em três métodos principais. O primeiro engloba métodos de correspondência holística, que utilizam de toda a região da face para o reconhecimento. O segundo são métodos estruturais, onde componentes da face, como olhos, são obtidos da imagem e então utilizados. O último são métodos híbridos, que utilizam de ambas abordagens holística e estrutural (ZHAO et al, 2003).

4.2.1.1 Métodos de correspondência holística

Esses métodos utilizam toda a região da face como o material de entrada para o sistema de reconhecimento. Uma das representações para a região da face mais utilizada é a *Eigenpictures*, que são baseadas em Análise por Componentes Principais, PCA (*Principal Component Analysis*).

Uma metodologia muito utilizada é a *Eigenfaces*. Este processo busca reduzir o tamanho das matrizes que formam as imagens de faces, onde a quantidade de pixels equivale a quantidade de dimensões da matriz. Considerando que algoritmos requerem uma grande quantidade de amostras para terem efeito, é necessário o processamento de uma quantidade excessiva de dados.

Para resolver esse problema, é utilizada a técnica PCA. Esta tem por objetivo descobrir os eixos dimensionais que explicam a maior parte da variação da matriz, efetivamente reduzindo a quantidade de dimensões necessárias. Estes eixos são os componentes principais. Como resultado, obtém-se uma matriz de pesos, que ao multiplicar novas matrizes, as transforma para a dimensão do componente principal. Essa transformação é aplicada a todas imagens utilizadas para treinamento, e posteriormente para cada imagem utilizada para reconhecimento. O resultado é uma redução expressiva na quantidade de dados processados, ao custo de passos adicionais de processamento.

Exemplos de outros métodos que têm por base o PCA incluem: *Probabilistic eigenfaces, Fisherfaces/subspace LDA, Evolution pursuit, Feature Lines e Independent component analysis*.

4.2.1.2. Métodos de correspondência baseados em características (estruturais)

Nesses métodos características locais como olhos, nariz e boca são primeiro extraídos e suas posições e estatísticas locais (geometria e/ou aparência) são fornecidos a um classificador estrutural.

Exemplos: *Pure geometry methods*, *Dynamic link architecture*, *Hidden Markov model*, *Convolution Neural Network*. (ZHAO et al, 2003)

4.2.1.3 Métodos híbridos

Assim como o sistema de percepção humano utiliza ambas características locais e toda a região da face para reconhecer um rosto, um sistema de reconhecimento de uma máquina deveria também utilizar ambos. Zhao continua o argumento de que, "pode ser argumentado que esses métodos podem potencialmente oferecer o melhor dos dois tipos de métodos". Exemplos: *Modular eigenfaces, Hybrid LFA, Shape-normalized, Component-based* (ZHAO et al, 2003).

Comparado à abordagens holísticas, métodos baseados em características são menos sensíveis à variações na iluminação e ponto de vista, e para inexatidão na localização da face. Porém, de acordo com (ZHAO et al, 2003), as técnicas de extração de características requeridas para este tipo de abordagem ainda não são confiáveis ou precisas o suficiente. Por exemplo, a maioria das técnicas de localização de olhos assumem alguns modelos geométricos e de textura e não funcionam caso o olho esteja fechado.

4.3. *Support Vector Machines*

Segundo (HEISELE et al, 2001), SVM, é um método de classificação que pertence à classe de classificadores por margem. Vetores de suporte são os pontos do conjunto de dados de treino que definem as margens de classificação do hiperplano. São os pontos para os quais a resolução do problema de equação quadrática fornece alfas com valor maior que zero. O SVM utiliza de vetores de suporte para encontrar um hiperplano, que é a "superfície de decisão", a qual permite classificar em duas classes distintas. O objetivo final é encontrar o hiperplano de decisão ótimo, OSH (do inglês *optimal separating hyperplane*), o qual maximiza a distância até os vetores de suporte.

A Figura 4 a seguir apresenta um exemplo de OSH encontrado para um espaço de duas dimensões.

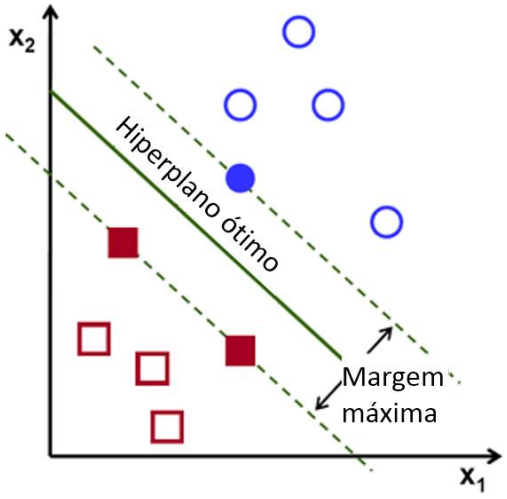


Figura 4 –Espaço de dimensões x_1 e x_2, onde existem pontos de duas classes, quadrados vermelhos e círculos azuis. A linha divisória é o hiperplano de separação encontrado, que apresenta a margem máxima de separação entre as classes, e os pontos preenchidos são os vetores de suporte do hiperplano. Fonte traduzida de: http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html.

De (HEISELE et al, 2001), segue a explicação. Dado o conjunto de pontos $x_i \in \mathbb{R}^n$, com $i = 1,2,\ldots,N$, onde cada ponto x_i pertence a uma de duas classes distintas. Tais classes são rotuladas $y_i \in /\{-1,1\}$, o OSH terá a seguinte forma:

$$\boldsymbol{f(x) = \sum_{i=1}^{\ell} \alpha_i y_i x_i \cdot x + b} \qquad (1)$$

Os coeficientes α_i e b na Eq.(1) são as soluções para um problema de programação quadrática e a classificação de um novo ponto x é realizada pela resolução da equação e observação do sinal do resultado.

Em seguida (HEISELE et al, 2001) explica o fato de o método poder ser utilizado em superfícies não linearmente separáveis. Para isso, é necessário mapear os pontos x de input para um espaço de maior dimensionalidade, chamado de "espaço de características", representado por $z = \Phi(x)$. A função de mapeamento é $\Phi(\cdot)$ e z o ponto em um espaço de maior dimensionalidade. Cita ainda que essa transformação é possível sempre que não se viole a restrição de que deve ser possível representar o produto escalar da transformação $\Phi(x) \cdot \Phi(y)$

de dois pontos no plano como uma função K = (x, y), chamada função *Kernel*. A Figura x exibe a transformação de um espaço de características não linearmente separável pela aplicação de um *Kernel*.

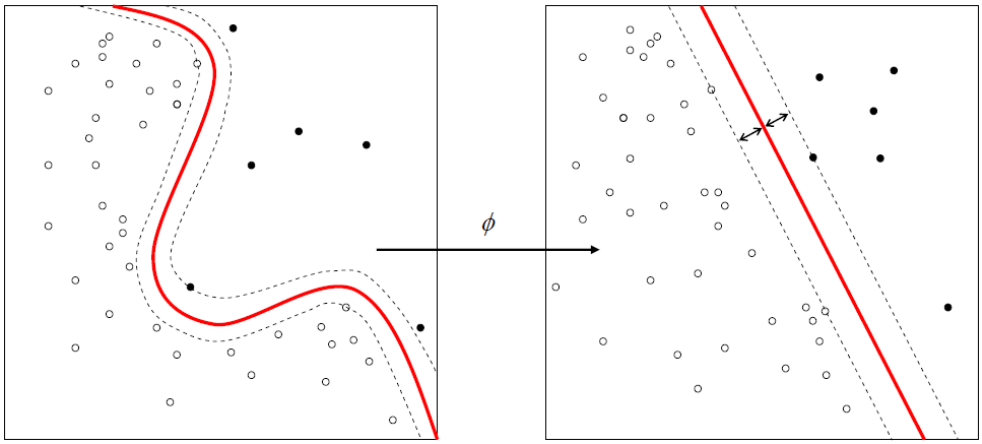


Figura 5 –Exemplo de utilização de *Kernel*. A função *Kernel* Φ transforma o espaço de características onde pontos não são linearmente separáveis a esquerda para um linearmente separável a direita. Fonte da imagem: https://upload.wikimedia.org/wikipedia/commons/1/1b/Kernel_Machine.png

Após a adição do Kernel, a equação classificatória passa a ser

$$\boldsymbol{f(x) = \sum_{i=1}^{\ell} y_i \alpha_i K(x, x_i) + b} \qquad (2)$$

Os coeficientes α_i e b continuam sendo as soluções para um problema de programação quadrática, e a equação não depende da dimensionalidade do espaço de características. (HEISELE et al, 2001) cita que uma importante família de funções *Kernel* é o *Kernel* polinomial:

$$\boldsymbol{K(x, y) = (1 + x \cdot y)^d}, \qquad (3)$$

Onde d é o grau do polinômio. Neste caso os "componentes do mapeamento $\Phi(x)$ são todos os monômios possíveis para componentes de *input* até o grau d". (HEISELE et al, 2001)

4.3.1 *Kernel trick*

Kernel Trick é uma técnica que permite aplicar a SVM em um espaço de dimensão Z diferente, sem precisar executar os cálculos dentro do mesmo. Consiste em substituir o produto interno $x^t x$ por $K(x^t, x)$, que apenas retorna o valor do produto interno no espaço Z.
A restrição para seu uso, que impede que uma função qualquer seja utilizada, é a necessidade de que seja um produto interno do vetor z no espaço Z de N dimensões.

4.4. Algoritmos para classificação multi-classe com *Support Vector Machines*

(HSU, C.-W; LIN, C.-J. 2002) Cita que existem dois modos de fazer classificação multi-classe com SVM. Um é criar múltiplos classificadores binários e uni-los, tentando resolver o problema realizando múltiplas classificações binárias. O outro é utilizar todos os dados diretamente na otimização. Destes, os mesmos chegam a conclusão de que os métodos mais práticos e adequados são o um-conta-um e o DAGSVM.

Para exemplos de classificação multi-classe nesta sessão, serão utilizados os indivíduos s1, s2, s3 e s4, cada um representado por uma cor, como apresentado na Figura 6 a seguir.

Figura 6 –Indivíduos s1, s2, s3 e s4, representado por respectivas cores, para exemplos multi-classe nesta sessão.

4.4.1. Um-contra-todos

Este método consiste em criar um modelo SVM para cada classe, onde compara a classe em si contra todas restantes. Cada classificador tem por resultado informar se o item avaliado pertence ou não a classe. A Figura 7 abaixo demostra um exemplo deste tipo de classificação multi-classe.

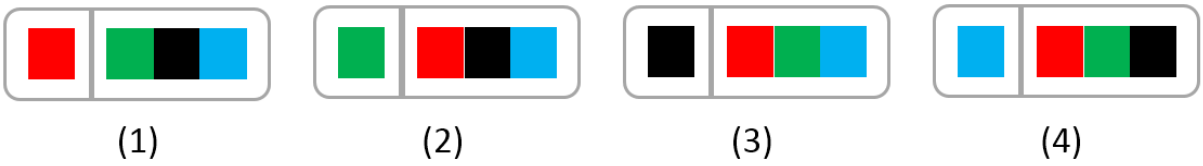


Figura 7 – Exemplo de SVMs construídas para classificação de quatro sujeitos distintos, representados pelas cores vermelho, verde, preto e azul. (1) é o classificador que separa vermelho das cores restantes. (2) separa azul, (3) separa preto e (4) separa azul das restantes. Fonte: do autor.

Tecnicamente, um-conta-todos constrói k modelos SVM, onde k é o número de classes. Para cada classe i, é treinada uma SVM, onde todos os elementos pertencentes a classe recebem o rótulo positivo, e o restante, negativo. Considerando l dados para treinamento $(x_1, y_1), \ldots, (x_l, y_l)$, onde $x_i \in \mathbb{R}^n$, $i = 1,2, \ldots, l$ e $y_i \in \{1, \ldots, k\}$ é a classe de x_i, a i-ésima SVM resolve o seguinte problema:

$$\underset{w^i, b^i, \xi^i}{\boldsymbol{min}} \quad \mathbf{1/2}(\boldsymbol{w^i})^{\boldsymbol{T}}\boldsymbol{w^i} + \boldsymbol{C}\sum_{\boldsymbol{j=1}}^{\boldsymbol{l}} \boldsymbol{\xi_j^i}(\boldsymbol{w^i})^{\boldsymbol{T}} \tag{4}$$

com restrições:

$$(\boldsymbol{w^i})^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x_j}) + \boldsymbol{b^i} \geq \mathbf{1} - \boldsymbol{\xi_j^i}, \quad \boldsymbol{se\ y_j = i}$$

$$(\boldsymbol{w^i})^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x_j}) + \boldsymbol{b^i} \leq -\mathbf{1} + \boldsymbol{\xi_j^i}, \quad \boldsymbol{se\ y_j \neq i}$$

$$\boldsymbol{\xi_j^i} \geq \mathbf{0}, \quad \boldsymbol{para\ j = 1, \ldots, l}$$

onde os dados x_i para o treinamento são mapeados para um espaço com mais dimensões pela função ϕ e C é o parâmetro de penalidade.

Minimizar $1/2(w^i)^T w^i$ significa, em realidade, maximizar $2/\|w^i\|$, que é a margem entre dois grupos de dados. O termo de penalidade $\boldsymbol{C}\sum_{\boldsymbol{j=1}}^{\boldsymbol{l}} \boldsymbol{\xi_j^i}(\boldsymbol{w^i})^{\boldsymbol{T}}$ existe para quando dados não forem linearmente separáveis, tendo por função reduzir o número de erros de treinamento. "O conceito básico por trás da SVM é buscar por um balanço entre o termo de regularização $1/2(w^i)^T w^i$ e os erros de treinamento."

Após resolver (1), existem k funções de decisão:

$$(\boldsymbol{w^1})^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x}) + \boldsymbol{b^1}$$

$$\vdots$$

$$(\boldsymbol{w^k})^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x}) + \boldsymbol{b^k}.$$

A classe de x será aquela que tiver o maior valor para a função de decisão

$$\boldsymbol{classe\ de\ x = arg\ \max_{i=1,\ldots,k}((w^i)^T \phi(x) + b^i).} \quad (5)$$

O problema resolvido em (5) é dual e o número de variáveis é o mesmo que o número de dados. Por isso, a quantidade de problemas de programação quadrática resolvidos é k vezes l.

4.4.2. Um-contra-um

Esta classificação multi-classe gera um classificador SVM para cada par de classes, de modo que todas classes são comparadas entre si, apenas uma vez. Um item a ser classificado é dito pertencer a um classe quando a maioria dos classificadores tem como resultado ela. A Figura 8 abaixo demonstra um exemplo deste tipo de classificação multi-classe.

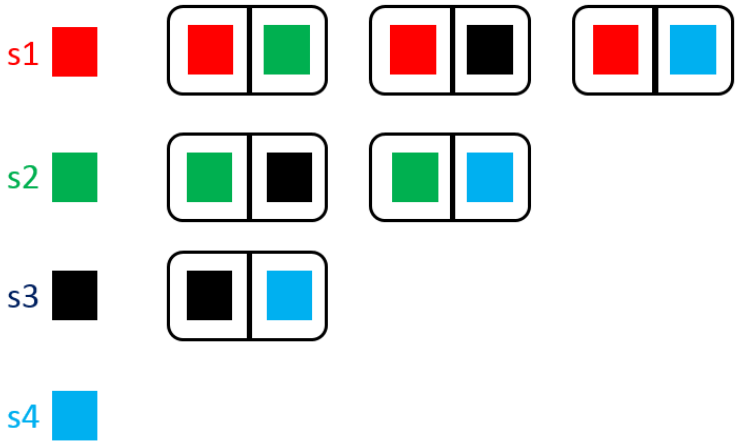


Figura 8 – Exemplo de classificador multi-classe um-contra-um. Para cada sujeito s1, s2, s3, s4, são gerados classificadores que os separam contra cada um dos restantes. S4 não apresenta nenhum classificador a sua direita, pois já é comparado contra todos os outros sujeitos nos classificadores anteriores. Fonte: do autor.

Tecnicamente este método constrói $k(k-1)/2$ classificadores, onde cada um é treinado com dados de duas classes. Para dados de treinamento da i-ésima e j-ésima classes, o seguinte problema de classificação binária é resolvido:

$$\min_{w^{ij},b^{ij},\xi^{ij}} \quad \mathbf{1/2}(\boldsymbol{w^{ij}})^{\boldsymbol{T}}\boldsymbol{w^{ij}} + \boldsymbol{C}\sum_{\boldsymbol{j=1}}^{\boldsymbol{l}} \boldsymbol{\xi_t^{ij}}(\boldsymbol{w^{ij}})^{\boldsymbol{T}} \tag{6}$$

com restrições:

$$(\boldsymbol{w^{ij}})^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x_t}) + \boldsymbol{b^{ij}} \geq \mathbf{1} - \boldsymbol{\xi_t^{ij}}, \quad \boldsymbol{se}\ \ \boldsymbol{y_t} = \boldsymbol{i}$$

$$(\boldsymbol{w^{ij}})^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x_t}) + \boldsymbol{b^{ij}} \leq -\mathbf{1} + \boldsymbol{\xi_t^{ij}}, \quad \boldsymbol{se}\ \ \boldsymbol{y_t} = \boldsymbol{j}$$

$$\boldsymbol{\xi_t^{ij}} \geq \mathbf{0}.$$

Após todos os $k(k-1)/2$ classificadores serem construídos, existem diferentes métodos possíveis para realizar o teste. (HSU, C.-W; LIN, C.-J. 2002) cita que um deles é uma estratégia de tipo votação chamada *Max Wins*, sugerida em (FRIEDMAN, J. 1996). Se $sinal\ de\ ((w^{ij})^T\phi(x_t) + b^{ij})$ diz que x pertence a i-ésima classe (gera sinal positivo), então os votos para a i-ésima classe são acrescidos em 1. Senão, a j-ésima tem seu voto acrescido. Ao final do processo, x acaba pertencendo a classe que possuir mais votos. Caso aconteça um empate, (HSU, C.-W; LIN, C.-J. 2002) optaram por escolher a classe de menor índice, apesar de possivelmente não ser a melhor heurística.

A resolução do problema dual em (6) envolve um número de variáveis que é igual ao número de dados em duas classes. Sendo assim, se em média cada classe possui l/k pontos de dados, é necessário resolver $k(k-1)/2$ problemas de programação quadrática onde cada um possui em média $2l/k$ variáveis.

4.4.3. DAGSVM – *directed asiclic graph SVM*

O algoritmo de classificação multi-classes SVM de Grafo Dirigido Acíclico DAGSVM (*Directed Asiclic Graph SVM*) foi proposto por (PLATT et al, 2002). De forma geral, esta estrutura de classificação permite arranjar classificadores um-contra-um de forma a requerer poucas comparações para obtenção da classificação. A partir do nodo raiz, o resultado obtido em um nodo leva a comparação em apenas um dos nodos filhos, repetindo o processo até chegar a um nodo folha, que é tido como resultado. A Figura 9 a seguir apresenta um exemplo de DAGSVM para classificação de três classes distintas.

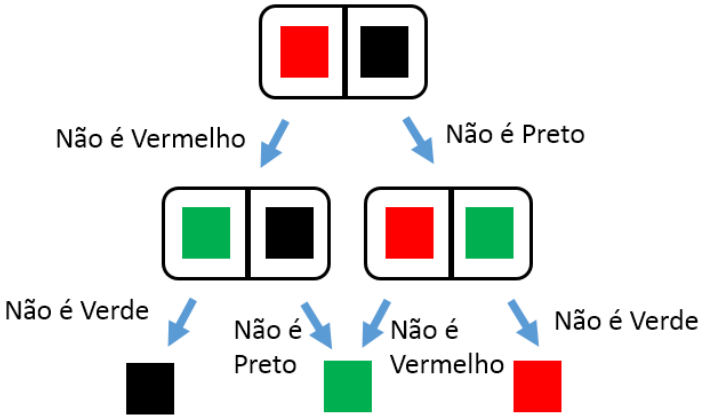


Figura 9 – Exemplo de classificador multi-classe DAGSVM. São classificadas três classes distintas, vermelho, verde e preto. O grafo gerado possui altura 3, e requer a execução de duas classificações para obter o resultado, a partir do nodo raiz (nodo classificador vermelho ou preto, no topo). Fonte: do autor.

Como (PLATT et al, 2002) explicam, Grafo Dirigido Acíclico, ou DAG (*Directed Asiclic Graph*) é um grafo que não possui ciclos e as arestas possuem uma orientação. A este, são acrescidos outras duas denominações, enraizado e binário. A primeira significa que o DAG possui um nodo especial, tal que este nodo é o único que não possui arestas apontando para o mesmo. A segunda que o DAG é composto por nodos que possuem apenas 0 ou 2 arestas saindo dos mesmos. Este é a base utilizada para o algoritmo.

(PLATT et al, 2002) fornece a seguinte definição para Decision DAGs (DDAGs):

> "Dado um espaço X e um conjunto de funções booleanas $F = \{f : X \to \{0,1\}\}$, a classe DDAG(F) de DAGs de Decisão em N classes sobre F são funções que podem ser implementadas utilizando um DAG Enraizado Binário com N folhas rotuladas pelas classes, e onde cada um dos $K = N(N-1)/2$ nodos internos é rotulado com um elemento de F. Os nodos são arranjados em um triangulo com o único nodo raiz no topo, dois nodos no segundo nível e assim por diante até o ultimo nível de N folhas. O i-ésimo nodo no nível $j < N$ é conectado ao i-ésimo e ao $(i+1)$-ésimo nodos no $(j+1)$-ésimo nível."

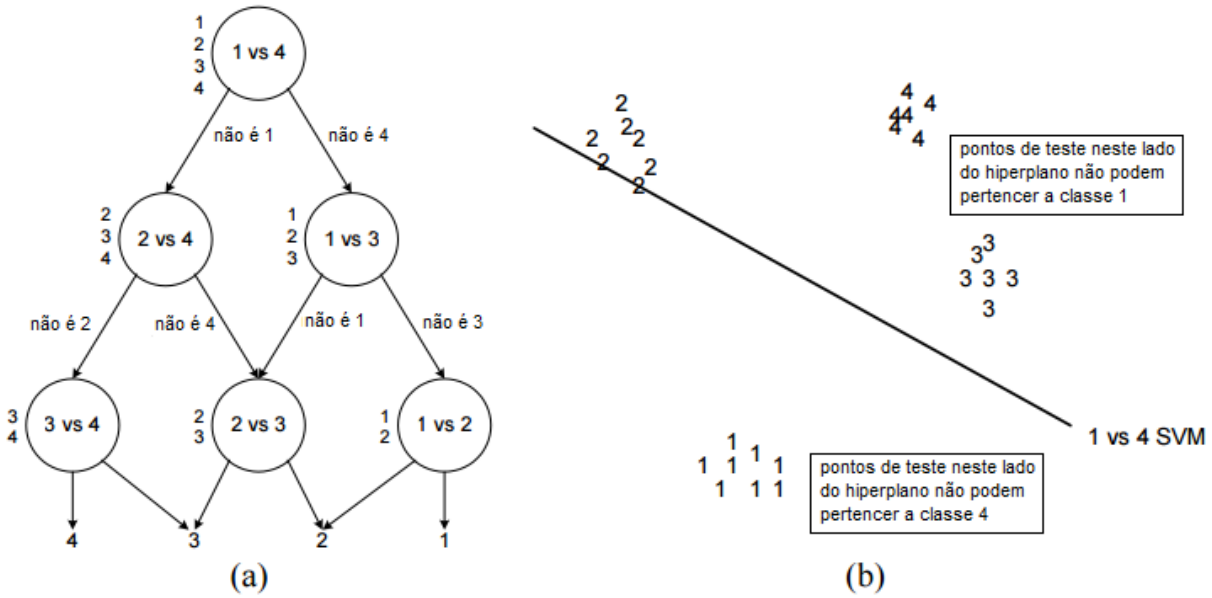


Figura 10 - (a) *Decision DAG* para encontrar a melhor classe a partir de quatro classes. A lista de estados equivalente para cada nodo é exibida a esquerda de cada. (b) Um diagrama do espaço de entrada para um problema com quatro classes. Uma SVM um-contra-um pode apenas excluir uma classe da consideração. Traduzida de fonte (PLATT et al, 2002)

A Figura 10 mostra um exemplo de DAGSVM para 4 classes, e um exemplo de classificação realizado por cada nodo. Para N classes, N-1 nodos são avaliados até chegar a uma conclusão. O grafo possui apenas um nodo raiz, de onde sempre inicia, e é formado por múltiplas SVMs um-contra-um, que fornecem como resultado a exclusão de uma das classes comparadas. Cada SVM é treinada apenas com os pontos de dados pertencentes as classes que classifica. Nenhum nodo classificador é repetido e as folhas do grafo são as classes possíveis para resultado. Quanto a posição das classes nas folhas ou ordem dos nodos, não há uma específica.

O algoritmo inicia comparando as classes i e j, tendo como resultado a exclusão de uma das classes, que leva a outro nodo de comparação. O processo se repete até chegar a uma das folhas, e o valor contido nessa é o resultado do algoritmo.

4.5. Trabalhos relacionados

Nessa seção serão apresentados alguns trabalhos pesquisados que possuem relação com o presente trabalho.

Em (HEISELE et al, 2001) foi testada a utilização de SVMs para reconhecimento de faces com poses distintas. Foi utilizada de duas abordagens baseadas no rosto inteiro (global) e uma baseada em componentes extraídos da face. Das globais, a primeira utilizou-se de uma SVM um-contra-todos para cada indivíduo a ser reconhecido. A outra consistiu em classificar a posição da face através da técnica de clusterização divisiva, para então utilizar um classificador de acordo com a posição da face.

O reconhecedor por componentes consiste em um detector formado por SVMs, capazes de localizar e extrair componentes faciais, e um classificador geométrico, que verifica se a configuração dos componentes extraídos corresponde com modelos geométricos previamente aprendidos. Os componentes então são transformados para respectivos tamanhos padrões, e transformados em um vetor de *features* de forma sequencial, para então serem utilizados nos classificadores SVM.

O primeiro teste consistiu em treinar 8593 imagens de faces, e testar contra 974, para 5 indivíduos. Resultados obtidos foram expressos através de gráficos ROC, dos quais pode-se extrair os dados a seguir. Para o classificador baseado em componentes, com SVM linear, taxa de mais de 95% de acerto com menos de 5% de falso positivos. Para global com clusterização e SVM linear, 95% de acerto com 5,5% de falso positivos. Para global com SVM linear, permitiu 50% de acerto com 4% de falsos positivos, e aumentando o limiar, 78% de acerto com 22% de falsos positivos. O de pior desempenho foi o global com SVM polinomial de grau 2, com 54% de acertos para 26% de falsos positivos.

O segundo teste utilizou apenas 1383 imagens de faces frontais para treino e 974 para testes, para 5 indivíduos. O classificador baseado em componentes, com SVM linear, resultou em 90% de acerto para 10% de falsos positivos. Para global com clusterização e SVM linear, 70% de acerto para 10% de falsos positivos, e aumentando o limiar, 78% de acerto para 22% de falsos positivos. Para global com SVM linear, 62% de acerto para 10% de falsos positivos, e 76% de acerto para 23% de falsos positivos. De todos testes realizados, a abordagem por componentes se mostrou superior.

Em (GUO et al, 2000) comparou-se o desempenho do reconhecimento de faces com duas técnicas, *eingenfaces* com critério de classificação por centro mais próximo (NCC) e SVMs. Foram realizados dois experimentos, um com a base de faces *The AT&T Dabase of Faces* (anteriormente *The ORL Database of Faces*), com 400 imagens de 40 indivíduos distintos, e outro com uma base composta, com 1079 imagens de 137 indivíduos. Está última é constituída por *The AT&T Dabase of Faces, The Bern database* (30 indivíduos), *The Yale database* (15 indivíduos) e uma própria, contento 179 imagens de faces frontais de 47 estudantes chineses.

O primeiro teste utilizou 5 imagens de cada indivíduo, escolhidas aleatoriamente, para treinamento e o restante para testes. Realizou-se o processo 4 vezes, e os resultados obtidos foram de 3% em média de erro, em média, para SVM, e 5,25% para NCC. Menor taxa de erro obtida com SVM foi de 1,5%.

No segundo teste, para treinamento, foram utilizadas 544 imagens e, para teste, 535. Foram utilizadas arvores binárias para classificação multi-classe, modificadas para serem otimizadas para muitos indivíduos. O experimento foi realizado apenas uma vez e teve como menor taxa de erro para SVM de 8,75%, e para NCC, de 15,14%.

Analisando-se ambos o trabalhos percebe-se que o primeiro trabalho demonstrou que a utilização de componentes da face é melhor para o reconhecimento do que a face inteira, porém requer detectores precisos, capazes de encontrar corretamente as partes. Esta dificuldade técnica torna a implementação por componentes uma boa alternativa para um trabalho futuro. A segunda pesquisa utilizou a técnica PCA como redutora de dimensionalidade e extração dos padrões mais relevantes das imagens das faces, obtendo um bom resultado.

5. IMPLEMENTAÇÃO

Foi desenvolvido um sistema para permitir testes de reconhecimento de faces a partir de imagens, realizando detecção de face nas mesmas, pré-processamentos, treinamento de classificadores SVM para reconhecimento e realização de testes de reconhecimento.

5.1 Sistema proposto

O sistema tem por requisito principal permitir o treinamento e teste de SVMs multi-classe para serem utilizadas na tarefa de reconhecimento de faces. Para tanto, torna-se necessário também ser capaz de extrair as faces de imagens. Sendo assim, pode-se resumir o sistema nos seguintes passos principais:

1- Obter uma imagem de entrada ou um conjunto de imagens;

2- Instanciar classes para detecção de faces do OpenCV, que utilizam de *Haar Feature-based Cascade Classifier;*

3- Detectar faces e pré-processar essas;

4- Instanciar classe para reconhecimento de faces;

4.1- Utilizar instancia de reconhecimento sobre face pré-processada;

4.2-Retornar lista de indivíduos potenciais, em ordem de probabilidade.

Para detecção de faces foi utilizada a biblioteca OpenCV, inicialmente desenvolvida pela Intel, porém mais tarde tornada *open source*. Essa biblioteca possui diversos algoritmos relacionados ao processamento de imagens, incluindo detecção e reconhecimento de faces. Devido ao foco deste trabalho ser o reconhecimento, a etapa de detecção será realizada por um método disponível nesta biblioteca, o *Haar Feature-based Cascade Classifier* (Sua base foi comentada no referencial teórico, o Detector de faces Viola-Jones).

O sistema foi desenvolvido em C#, e para permitir compatibilidade, será utilizado um *Wrapper* para OpenCV, chamado *EmGU*, acessível em: < http://www.emgu.com/wiki/index.php/Main_Page >.

Para os passos 4 e 4.1 foi utilizada uma classe de reconhecimento, que utilizará de métodos da LIBSVM, uma biblioteca desenvolvida por (CHANG, C.-C; LIN, C.-J, 2013), e que possui métodos de classificação que utilizam de SVM. Ela implementa os métodos de

classificação comentados neste trabalho. Especificamente, foi utilizado uma interface de LIBSMV para C#, disponível em https://github.com/ccerhan/LibSVMsharp.

O processo é dividido em 3 partes principais, exibidas no fluxograma da figura 11: carregamento de *dataset* e pré-processamento, treinamento de classificadores e reconhecimento.

Figura 11 – Fluxograma do processo geral do sistema. Fonte: do autor.

O sistema acessa um *dataset* de imagens contendo faces como entrada, as quais passarão pelas etapas do fluxograma. Para manter os dados gerados em cada etapa, definiu-se uma organização de arquivos de forma a separar dados de cada etapa do processo para cada *dataset* distinto. Cada um possui um diretório próprio, podendo por isso coexistir, pois persistem em domínios separados. O *dataset* em si consiste em um diretório com pastas, onde cada uma representa um indivíduo. Imagens contendo um mesmo indivíduo devem ficar dentro de uma mesma pasta. Esta recebe o nome do mesmo indivíduo, para identificação nas etapas de treinamento e reconhecimento.

Ao carregar um *dataset* para pré-processamento, o sistema gera uma estrutura de diretórios, com o intuito de separar o pré-processamento, o treinamento e o reconhecimento. As imagens pré-processadas são salvas, permitindo executar o treinamento sobre as mesmas sem necessitar executar os pré-processamentos e processamentos novamente.

Após a leitura do *dataset* de entrada, este é carregado no sistema, e para cada imagem, tenta-se localizar a face e em seguida os olhos. Caso não detecte algum destes, o usuário pode marcar o rosto e o ponto central de cada olho.

Obtendo-se o retângulo que compreende a face e as posições centrais dos olhos, passa então pela etapa de criação de variantes da imagem original. É realizada para lidar com diferenças na detecção em imagens futuras, e fornecer maior variação ao modelo de reconhecimento. A etapa seguinte é a de pré-processamento de cada imagem, com resultados salvos para não precisar repetir o processo novamente na etapa de treinamento de SVMs.

Para treinamento das SVM, para classificador multi-classe, pode-se selecionar entre três opções de estrutura: Um-contra-todos, Um-contra-um e DAGSVM.

Cria-se então os arquivos necessários para treinamento e testes, e o processo gera arquivos de modelo necessários de acordo com o tipo de classificação multi-classe selecionada.

Por fim, pode-se proceder para o reconhecimento. Uma imagem é fornecida ao sistema, e o mesmo processo de detecção e processamento realizado nas etapas anteriores é executado. O reconhecimento é realizado testando-se todos os arquivos de modelo que compõem o classificador, e todos resultados positivos são retornados.

5.1.1 Carregamento de *dataset* e pré-processamento

Dado o caminho para um *dataset*, cuja estrutura de pastas respeita o modelo explicado anteriormente, gera-se a estrutura de diretório necessária, e pode-se prosseguir com o pré-processamento. O fluxograma da Figura 12 expõe os passos para a execução do pré-processamento.

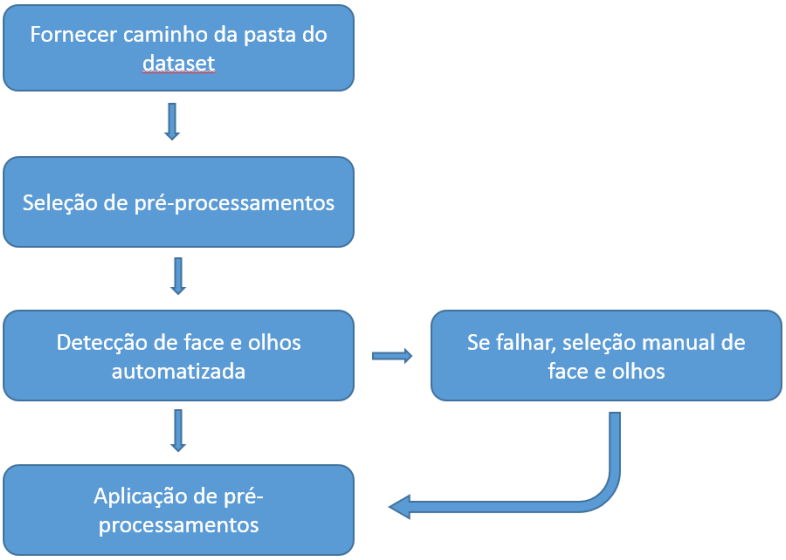


Figura 12 – Fluxograma do processo geral do sistema. Fonte: do autor.

O próximo passo é a seleção de pré-processamentos para aplicação em cada imagem. Estes incluem: definir o valor para a altura e a largura final do quadrado que corresponde a face pré-processada; optar pela criação de variantes da imagem; aplicar equalização por histograma; filtro bilateral e máscara elíptica. Após essa seleção, passa-se a etapa de detecção e de faces e olhos para cada imagem do *dataset*, explicada a seguir.

5.1.1.1 Detecção de face e olhos

De (BAGGIO et al, 2012) foi obtido o código e a forma de como se executa a detecção da face e dos olhos através da biblioteca OpenCV. O código, disponível em linguagem C, foi adaptado e alterações foram realizadas para permitir sua utilização no sistema.

A imagem é convertida para escalas de cinza, como requerido pelo algoritmo de detecção, e reduzida uniformemente para um tamanho específico máximo, de 320 pixels de altura ou largura, caso maior que tal valor, para otimizar a detecção.

Utilizou-se da biblioteca OpenCV com *wrapper* c# Emgu. Ela dispõe do método de detecção DetectMultiScale(), pertencente a classe CascadeClassifier, que foi utilizada tanto para detecção de faces quanto olhos. Este método recebe parâmetros de entrada, cujos valores finais escolhidos seguem abaixo.

Para a face, utilizou-se de:

- Arquivo de detecção: haarcascade_frontalface_default.xml.
- Fator de Escala: 1.1.
- Vizinhos: 2.
- Tamanho mínimo: 20x20 pixels.
- Tamanho máximo: String.Empty (ignorado).

Para olhos:

- Arquivo de detecção: de haarcascade_eye.xml, haarcascade_righteye_2splits.xml e haarcascade_lefteye_2splits.xml.
- Fator de Escala: 1.1.
- Vizinhos: 2.
- Tamanho mínimo:1x1 pixels.
- Tamanho máximo: tamanho da imagem de entrada.

Neste estágio de detecção, caso encontre um retângulo com o rosto, prossegue para a detecção dos olhos. Se encontrar mais de um retângulo para o rosto, escolhe aquele de maior área. Se algum dos anteriores não for encontrado, a imagem é retornada para o usuário, que então necessita selecionar o retângulo que compreende a face e a posição central de cada olho. Estas informações são então salvas em um arquivo, para que, caso a mesma imagem venha a ser reutilizada, não seja necessário encontrá-las ou fornecê-las novamente. As Figuras 13 e 14

demonstram, respectivamente, a detecção de faces e seleção automática de uma única, e seleção de face seguida de detecção de olhos e seleção também automática de cada um.

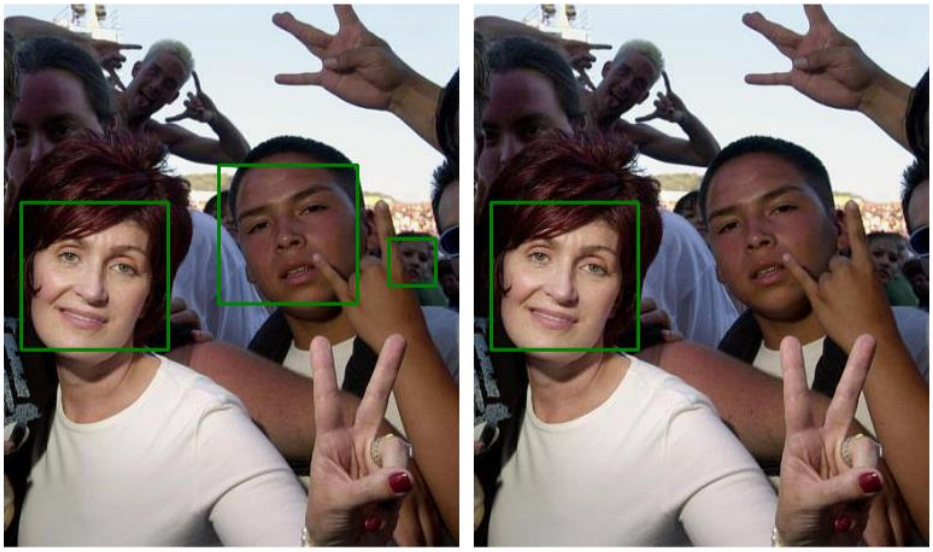


Figura 13 – Exemplo de detecção e escolha de faces automatizado. Fonte: do autor.

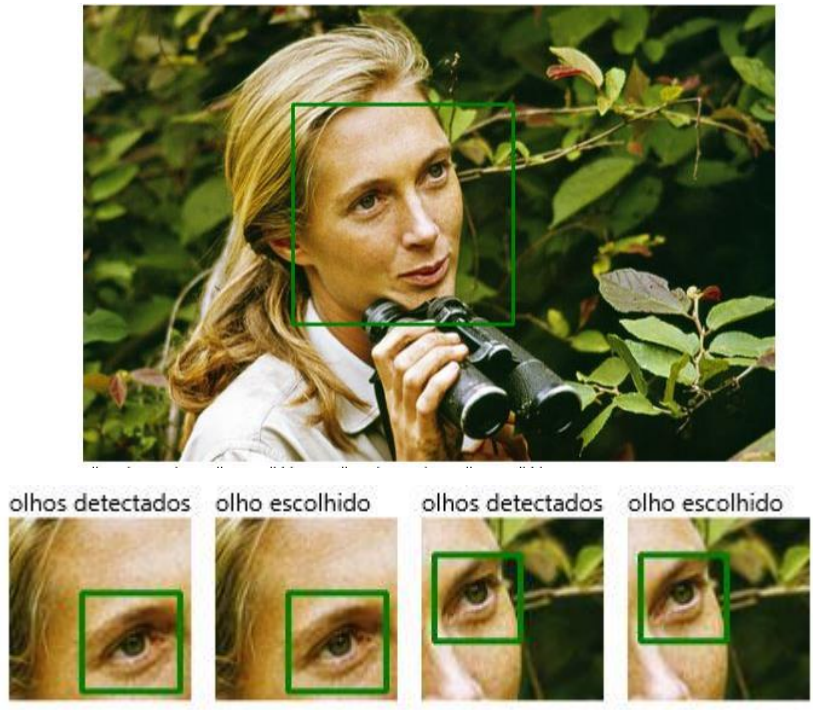


Figura 14 – Exemplo de detecção de faces e olhos automatizado. Fonte: do autor.

Quando não detecta face ou olhos automaticamente, é necessária a intervenção do usuário, como pode ser visto na Figura 15. As coordenadas escolhidas são salvas para posterior reuso.

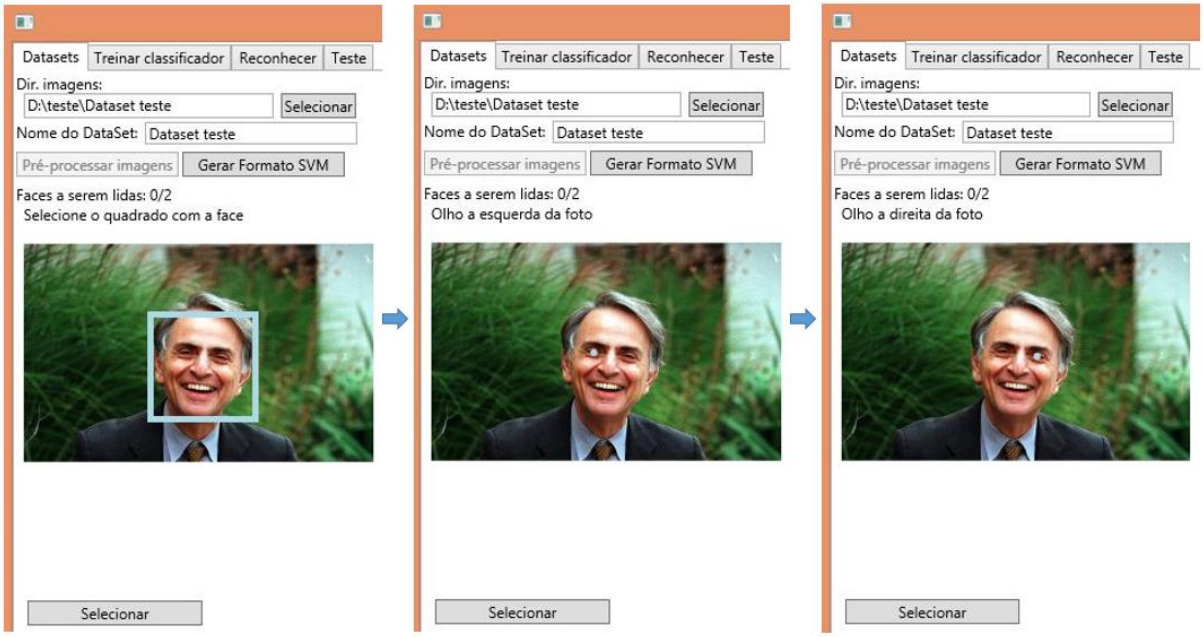


Figura 15 – Exemplo de seleção manual de face e olhos. Fonte: do autor.

Após a devida seleção das faces e seus olhos, dá-se o seguimento para a etapa de pré-processamento.

5.1.1.2 Pré-processamento

Esta etapa existe para uniformizar o tamanho das imagens de entrada e utilização de transformações sobre as imagens que possibilitem uma melhoria no resultado do reconhecimento. Um diagrama dos pré-processamentos disponíveis e sua ordem de aplicação é exibido na Figura 16.

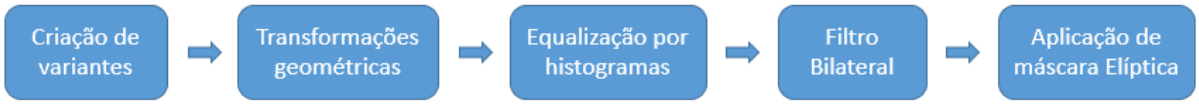


Figura 16 – Diagrama de pré-processamentos disponíveis e sua ordem de aplicação. Fonte: do autor.

O sistema permite a seleção de quais pré-processamentos se deseja aplicar. O primeiro é a criação de variações da imagem original. Para tanto, a posição central de cada olho é movimentada exclusivamente horizontalmente ou verticalmente, gerando 24 variantes da face. Posteriormente, a posição dos olhos determina a área da imagem da face que será mantida e o quanto é necessário realizar uma rotação na imagem para deixar os olhos nivelados (em mesma linha). Com isso, essas aumentam a variabilidade do modelo de reconhecimento, permitindo maiores diferenças na detecção da face (tamanho do rosto selecionado, posição dos olhos) para um mesmo indivíduo. Exemplos de variações geradas podem ser visualizadas na figura 17.

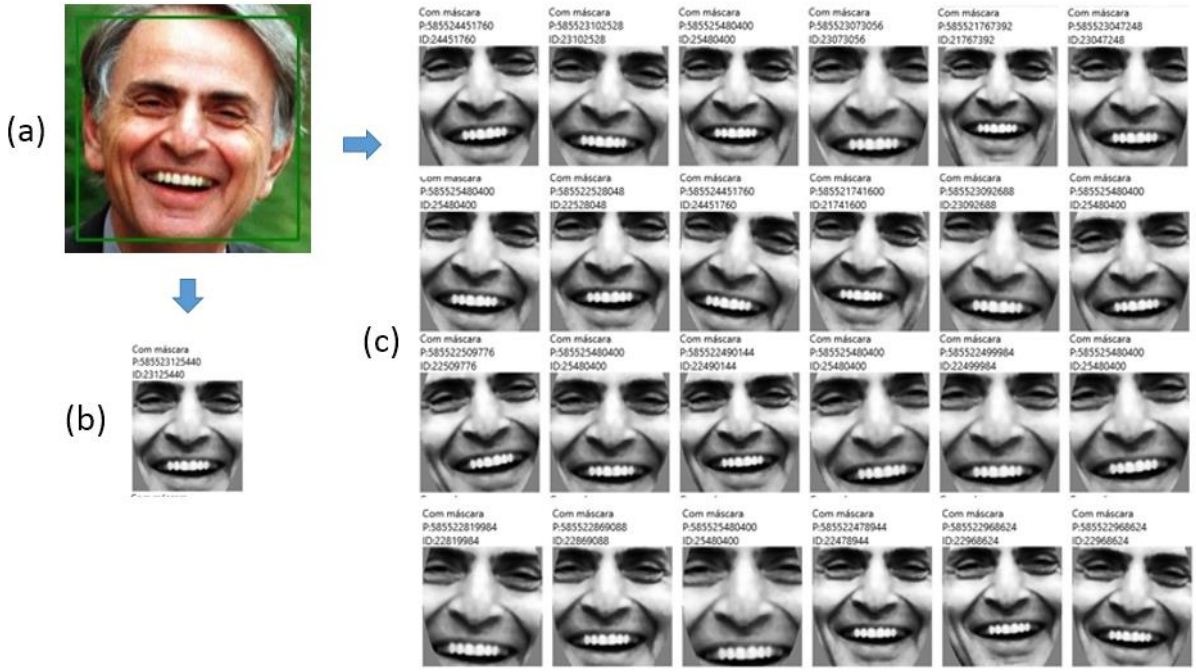


Figura 17 – Geração de variantes a partir de imagem de face detectada (a). A face em (b) mostra o resultado do pré-processamento de (a). Em (c) são exibidas as variações de imagens pré-processadas, geradas a partir de alterações nas coordenadas das posições dos olhos na imagem (a). Fonte: do autor.

Cada variante então passa pelos processamentos descritos a seguir, cuja sugestão de uso e implementação em código C foi obtida de (BAGGIO et al, 2012).

O primeiro passo é a transformação para escalas de cinza, para melhorar o desempenho do reconhecedor. Para o LIBSVM, permite diminuir o tamanho do vetor de *features* que servirá de entrada. Cada pixel passa a ser representado por uma dimensão (valor de 0 à 255), comparado a três dimensões por pixel, caso colorida. A redução de dimensões implica em melhoria de desempenho.

A próxima etapa é a de transformações geométricas, que engloba alterações de tamanho, área selecionada e rotações na imagem. É realizada em uma única operação com o método *warpAffine* da biblioteca *OpenCV*. Obtém-se a posição central e as diferenças de altura e de largura entre os olhos. Com as diferenças, obtém-se o ângulo x entre os olhos, a partir do arco tangente. Com estas informações, é possível então girar em x graus a imagem a partir do centro entre os olhos, deixando-os em mesmo nível (altura). Ainda da imagem são removidas a testa, queixo e laterais do rosto, através da definição da proporção de distância fixa que os olhos devem possuir das laterais e topo da face. Definido no trabalho como 0.16 de cada lateral e 0.14 a partir do topo. Por fim a imagem é reduzida para o tamanho especificado nos parâmetros de pré-processamento. O efeito final é a seleção da área central da face e padronização de todas as imagens de faces para um mesmo tamanho.

A equalização por histograma em três partes, utilizada para diminuir as variações de iluminação entre os lados da face. De acordo com (BAGGIO et al, 2012), diferenças de iluminação podem ser o suficiente para classificar uma mesma face como pertencendo a pessoas diferentes. A imagem da face é dividida em duas, e uma cópia inteira é mantida. Utiliza-se a equalização por histograma sobre cada uma das 3 imagens geradas. Esta serve para melhorar o contraste da imagem, espalhando faixas de intensidade concentrada de pixels para uma maior utilização do espectro de valores de 0 à 255. O próximo passo é obter a média dos valores de pixels das três imagens equalizadas, de acordo com a coluna do pixel. Detalhes de implementação podem ser lidos em (BAGGIO et al, 2012) páginas 243 à 245. Este último processamento tende a gerar ruído na imagem, o que leva a aplicação do próximo passo, o filtro bilateral, para suavizar a imagem.

Por fim ocorre a aplicação de máscara elíptica, para remover cantos da bochecha e da testa. A Figura 18 a na próxima página apresenta exemplos de resultados das etapas de pré-processamento, em sequência.

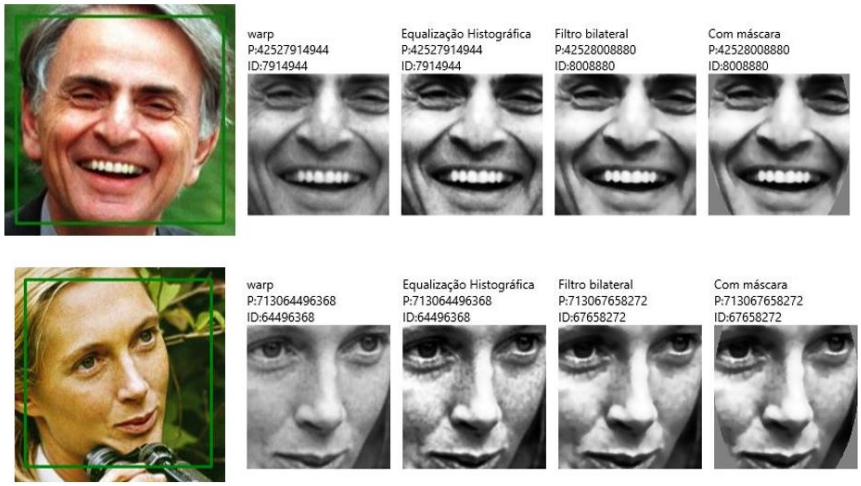


Figura 18 – Pré-processamentos realizados sobre duas faces detectadas (a esquerda). O primeiro é as transformações geométricas, em seguida equalização por histograma em três partes, após filtro bilateral e por fim aplicação de máscara elíptica. Fonte: do autor.

A Figura 19, a seguir, exibe a tela de pré-processamentos durante execução.

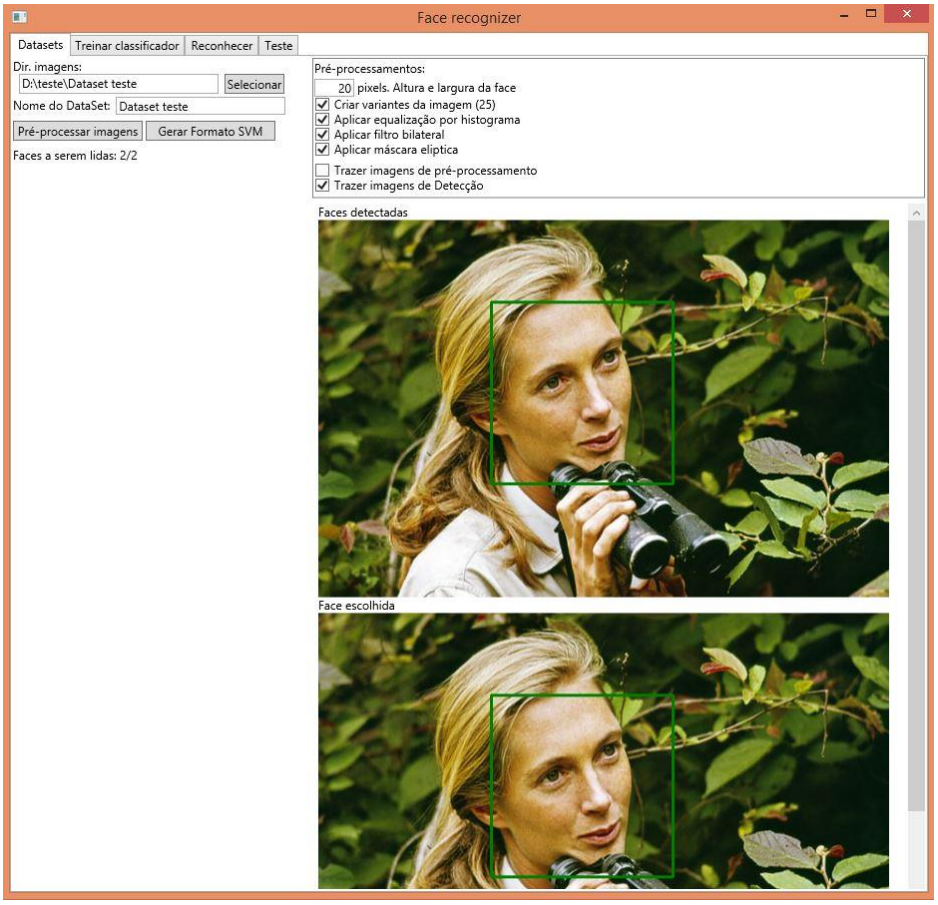


Figura 19 – Tela responsável pelo carregamento de *dataset*, detecção de faces, aplicação de pré-processamentos e transformação para vetor de *features*. Fonte: do autor.

Com todas as imagens pré-processadas salvas, é gerado um arquivo de formato .bin, com todas transformadas no formato de vetor de *features*, este sendo o formato de entrada aceito pelo LIBSVM, explicado na próxima seção.

5.1.2 Treinamento do reconhecedor

Para realizar o reconhecimento, é necessário treinar os classificadores multi-classe baseados em SVM, cujo processo de execução é exemplificado na Figura 20.

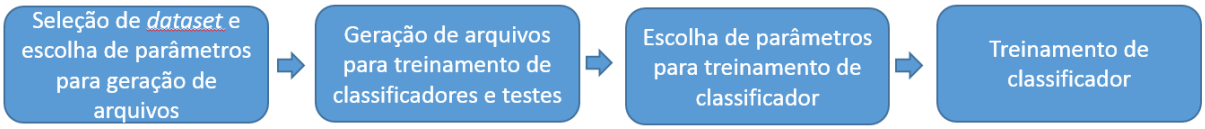
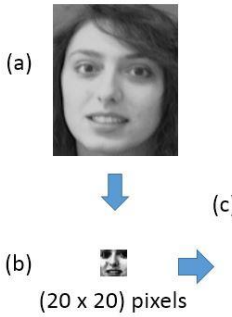


Figura 20 – Pré-processamentos, primeiro transformações geométricas, equalização pro histogramas, filtro bilateral e máscara elíptica. Fonte: do autor.

Após selecionado o *dataset*, define-se a quantidade das imagens originais de cada sujeito que deverá ser utilizada para o treinamento. Se existirem, as variantes de cada imagem selecionada são automaticamente utilizadas. Existe a opção de seleção aleatória de imagens para treinamento. Caso não se opte por ela, as primeiras, em ordem alfabética, na pasta são utilizadas. O restante das imagens são reservadas para o teste de reconhecimento de faces automatizado sobre o *dataset*.

A próxima etapa é a geração de arquivos para treinamento e testes. Para realizar o reconhecimento, é necessário gerar classificadores multi-classe baseados em SVM. O processo de treinamento dos mesmos requer arquivos de entrada contendo as imagens, distribuídos de acordo com os nodos classificadores a serem gerados. As imagens são transformadas para um vetor de *features*, onde a primeira coluna recebe o *label* da classificação. As colunas restantes são numeradas sequencialmente de 1 até a quantidade total de pixels que constituem a imagem, e o valor de cada pixel é escrito a direita de sua coluna correspondente (e.g.,1 1:208 2: 200 3:208).

Exemplo de imagem transformada em vetor de *features* pode ser observada na figura 21 a seguir.



(c)

1 1:128 2:96 3:96 4:91 5:73 6:55 7:51 8:49 9:67 10:172 11:189 12:173 13:72 14:60 15:47 16:56 17:75 18:104 19:96 20:88 21:128 22:89 23:98 24:96 25:86 26:65 27:57 28:52 29:69 30:194 31:242 32:172 33:70 34:53 35:63 36:81 37:110 38:123 39:100 40:84 41:128 42:95 43:100 44:98 45:84 46:47 47:44 48:47 49:43 50:244 51:245 52:166 53:60 54:50 55:45 56:56 57:117 58:133 59:83 60:18 61:128 62:41 63:21 64:12 65:11 66:16 67:22 68:52 69:56 70:243 71:245 72:162 73:58 74:72 75:18 76:18 77:16 78:20 79:19 80:15 81:23 82:13 83:13 84:10 85:13 86:17 87:46 88:77 89:144 90:243 91:245 92:156 93:96 94:83 95:63 96:18 97:13 98:12 99:11 100:10 101:134 102:65 103:27 104:16 105:21 106:44 107:132 108:176 109:191 110:244 111:247 112:157 113:100 114:163 115:160 116:64 117:31 118:38 119:43 120:47 121:245 122:241 123:228 124:197 125:195 126:216 127:216 128:210 129:213 130:240 131:248 132:167 133:104 134:142 135:221 136:224 137:217 138:208 139:207 140:213 141:246 142:247 143:245 144:243 145:238 146:227 147:226 148:205 149:213 150:246 151:246 152:205 153:104 154:131 155:216 156:230 157:235 158:234 159:234 160:232 161:244 162:246 163:245 164:236 165:196 166:217 167:191 168:194 169:241 170:246 171:247 172:233 173:114 174:92 175:212 176:232 177:235 178:235 179:233 180:229 181:229 182:243 183:242 184:222 185:183 186:147 187:115 188:220 189:178 190:239 191:242 192:175 193:83 194:129 195:180 196:221 197:228 198:232 199:228 200:223 201:237 202:235 203:207 204:180 205:134 206:139 207:104 208:96 209:14 210:160 211:101 212:33 213:42 214:84 215:170 216:167 217:212 218:222 219:218 220:208 221:194 222:163 223:137 224:122 225:130 226:189 227:196 228:154 229:80 230:59 231:31 232:37 233:29 234:145 235:168 236:171 237:157 238:195 239:199 240:191 241:151 242:128 243:113 244:127 245:189 246:227 247:226 248:216 249:194 250:96 251:85 252:135 253:158 254:151 255:161 256:175 257:170 258:175 259:175 260:132 261:128 262:131 263:119 264:184 265:229 266:228 267:218 268:192 269:188 270:194 271:159 272:150 273:160 274:161 275:162 276:162 277:168 278:170 279:130 280:125 281:128 282:154 283:155 284:107 285:30 286:26 287:21 288:25 289:27 290:26 291:24 292:24 293:20 294:21 295:53 296:112 297:159 298:149 299:132 300:119 301:128 302:192 303:157 304:153 305:17 306:15 307:63 308:157 309:211 310:136 311:192 312:125 313:61 314:18 315:13 316:12 317:153 318:165 319:139 320:113 321:128 322:187 323:157 324:192 325:176 326:75 327:63 328:54 329:132 330:84 331:107 332:49 333:24 334:20 335:22 336:155 337:155 338:170 339:132 340:78 341:128 342:128 343:164 344:191 345:192 346:87 347:80 348:94 349:103 350:104 351:124 352:76 353:37 354:30 355:91 356:156 357:146 358:127 359:74 360:128 361:128 362:128 363:142 364:165 365:166 366:128 367:79 368:51 369:43 370:44 371:37 372:34 373:45 374:76 375:125 376:107 377:114 378:76 379:68 380:128 381:128 382:128 383:128 384:164 385:179 386:149 387:138 388:102 389:88 390:80 391:81 392:83 393:88 394:106 395:119 396:124 397:86 398:66 399:128 400:128

Figura 21 – Transformação de imagem contendo a face em um vetor de *features*. (a) Imagem original. (b) Face extraída e pré-processada. (c) Vetor de *features* gerado a partir de (b). O primeiro caractere, "1", é o valor do *label* para a imagem. Em seguida, cada pixel da imagem é numerado sequencialmente de acordo com sua posição, e o valor de cada pixel é escrito a direita de sua coluna correspondente, separado por dois pontos. Fonte: do autor.

Gera-se então arquivos necessários para o treinamento dos três classificadores multi-classe implementados. A etapa seguinte envolve selecionar os parâmetros do classificador baseado em SVMs, o parâmetro de regularização C (permite diminuir *overfitting*), o tipo de *Kernel* utilizado (Linear, Polinomial, RBF e Sigmoide), e o parâmetro Gama, que afeta *Kernels* à exceção de Linear, o tipo de classificador multi-classe desejado (um-contra-um, um-contra-todos ou DAGSVM) e por fim, se deseja normalizar os valores dos vetores de *features*.

Arquivos que serão utilizados pela LIBSVM podem ter seus valores normalizados. Caso esse procedimento seja utilizado nesta etapa, precisa ser realizado também para arquivos que serão utilizados para o reconhecimento (sistema o faz automaticamente).

A normalização dos arquivos é recomendada para a boa classificação com a biblioteca LIBSVM. A normalização executada consiste em dividir cada valor de pixel no vetor de *features* pela raiz da soma dos quadrados de todos valores de pixels do vetor.

5.1.2.1 Implementação de classificadores

Para DAGSVM, o nome de cada indivíduo é adicionado à uma lista ordenada, e a posição i de cada nodo na lista é utilizada para a construção do grafo direcionado acíclico. Esta inicia com o nodo raiz, que classifica os indivíduos de menor (i_0) e maior índice i (i_n). A construção é feita em profundidade, de forma recursiva, primeiro gerando nodos à esquerda e

então a direita. O primeiro nodo à esquerda recebe o valor $i_0 + 1$ e i_n. Prossegue-se a geração até que $i_x + 1 = i_n$. Então, passa-se à criar nodos a direita, onde o primeiro valor é o i_x atual do nodo e o segundo valor recebe $i_n - 1$, respeitando também o critério de parada $i_x + 1 = i_n$.

Por otimização, nodos a esquerda são criados apenas até encontrar a primeira folha. Ao executar o reconhecimento inicia-se avaliando a face contra o nodo raiz. Se não for a classe a esquerda, segue para o nodo filho a esquerda. Se não for a classe a direita, segue para o nodo a direita. Ao chegar a uma folha, retorna o valor da classificação.

Para classificação multi-classe um-contra-um, cria-se um classificador binário para cada par de indivíduos no *dataset*. A imagem então é classificada contra todos os nodos, e escolhe-se o resultado por votação (estratégia *Max wins*). A classe com mais votos é o indivíduo reconhecido.

Para um-contra-todos, cada indivíduo possui um classificador que o diferencia de todos os outros. Todos classificadores com identificação positiva retornam o resultado.

Após a geração de classificadores, é possível realizar o reconhecimento, explicado em seguida.

A Figura 22 a seguir apresenta a tela de treinamento de classificadores.

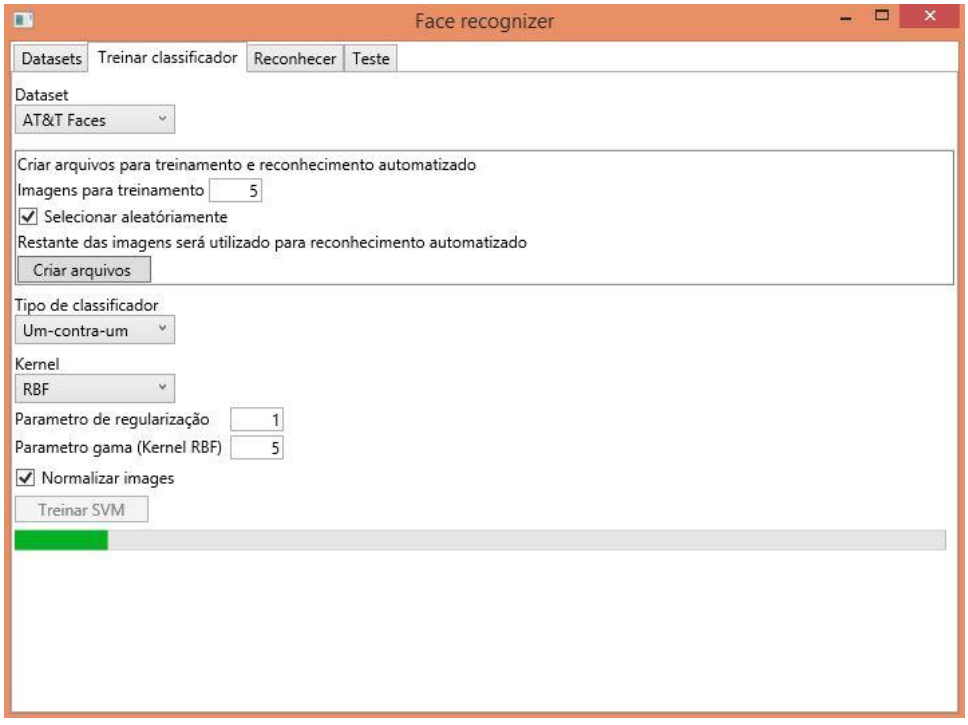


Figura 22 – Tela de treinamento de classificadores, realizando criação de classificador multi-classe um-contra-um. Fonte: do autor.

5.1.3 Reconhecimento

O fluxograma da Figura 23 apresenta os passos necessários para a realização da etapa de reconhecimento.

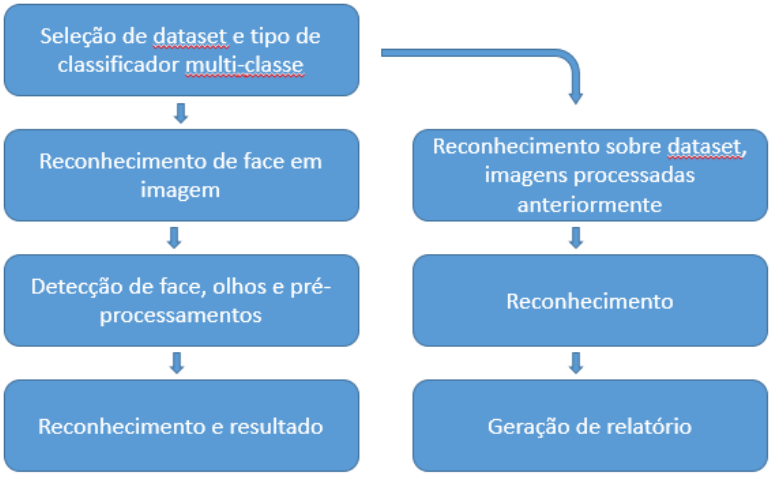


Figura 23 – Fluxograma da etapa de reconhecimento. Fonte: do autor.

É necessário escolher o *dataset* e o tipo de classificação multi-classe desejada. Em seguida é possível fornecer uma imagem diretamente para o reconhecimento, ou utilizar do conjunto de imagens do *dataset* separadas em etapas anteriores para o teste automatizado.

Caso se forneça uma nova imagem, esta passa por todas as etapas anteriores, e então é utilizada para o reconhecimento. Os parâmetros utilizados em pré-processamento anterior e treinamento de classificador mutli-classe específico estão salvos, não requerendo intervenção do usuário para fornecê-los novamente. Se for realizado o teste automatizado, este utiliza-se de imagens separadas na etapa de treinamento, e gera um arquivo texto com o resultado dos reconhecimentos. Informações do mesmo incluem: percentual de acerto total, percentual de falsos positivos, o percentual de imagens sem reconhecimento, percentual de falsos positivos com mais de uma pessoa reconhecida e tempo total transcorrido. Para cada indivíduo a ser testado, percentual de falsos positivos, percentual de imagens sem reconhecimento e o tempo de reconhecimento para todas as imagens do mesmo. Para cada imagem em si, retorna quais os indivíduos reconhecidos e a probabilidade do mesmo, caso aplicável. Exemplo de trechos do relatório podem ser observados na Figura 24. Em seguida, a Figura 25 exibe a tela de reconhecimento.

(a)

Dataset: AT&T Faces
Acertos: 93,000%
Sem reconhecimento:0,000%
Falsos positivos: 7,000%
Falsos positivos com mais de um resultado: 0,000%
Falsos positivos com mais de um resultado, incluindo o correto: 0,000%
Tempo: 00:07:48

(b)

Nome: s1
Acertos: 80,000%
Falsos positivos: 20,000%
Sem reconhecimento: 0,000%
Tempo: 00:00:13
C - Imagem: 1.pgm -> s1 | Matches: 1 | Probabilidade: 0,00000%
C - Imagem: 5.pgm -> s1 | Matches: 1 | Probabilidade: 0,00000%
C - Imagem: 7.pgm -> s1 | Matches: 1 | Probabilidade: 0,00000%
C - Imagem: 9.pgm -> s1 | Matches: 1 | Probabilidade: 0,00000%
X - Imagem: 10.pgm -> s6 | Matches: 1 | Probabilidade: 0,00000%

Figura 24 – Exemplo de retornos do relatório. (a) exibe resultados gerais e (b) um exemplo de resultados sobre um indivíduo. Fonte: do autor.

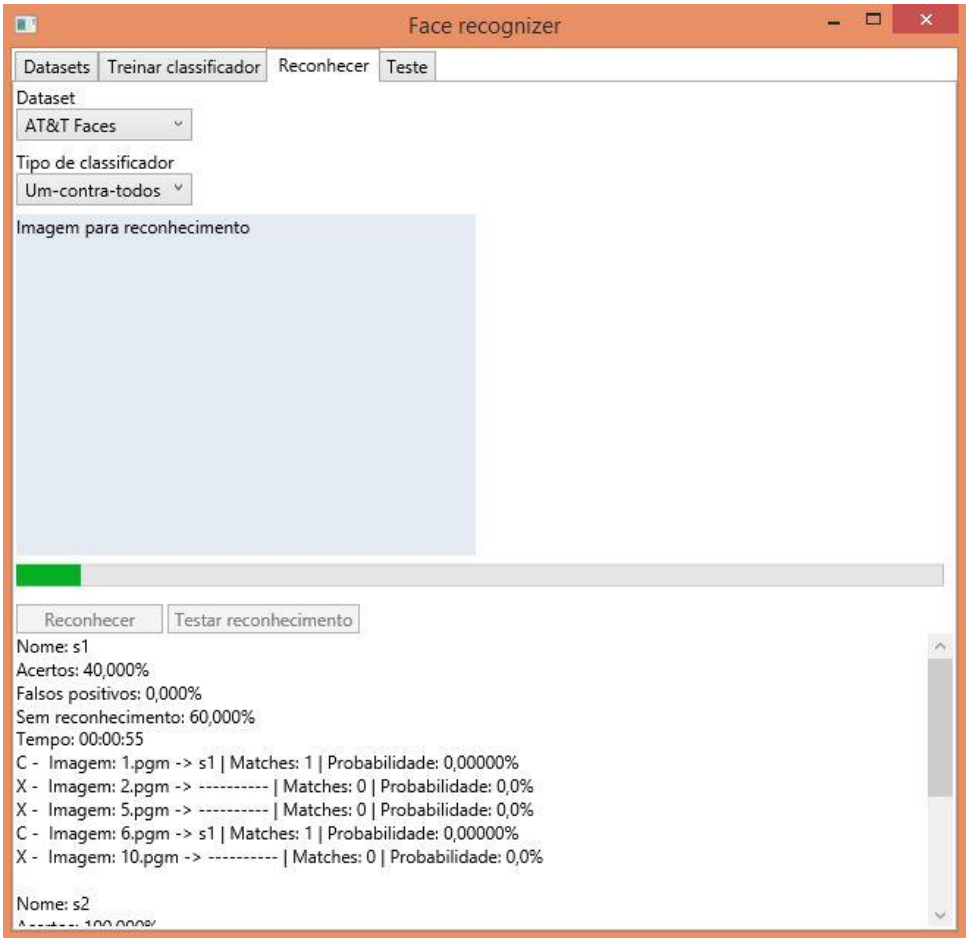


Figura 25 – Tela de reconhecimento, executando sobre imagens do dataset. Fonte: do autor

5.2 Manual de utilização

A tela inicial é a de carregamento de *datasets* e realização de pré-processamentos. Seu funcionamento é dado pelos passos a seguir.

1-Seleciona-se a pasta que contém o *dataset*. Este deve ser uma pasta de arquivos que possui como subdiretórios uma pasta para cada indivíduo, identificada com o nome do mesmo e que contém as imagens para treinamento e testes. O sistema aceita a maioria dos formatos comuns de imagens como entrada.

2-Em seguida escolhe-se os pré-processamentos desejados.

3-Então deve-se clicar no botão "Pré-processar imagens", que executa os pré-processamentos.

4-Após o pré-processamento, clica-se em gerar formato SVM, para transformar imagens no formato adequado para leitura nos algoritmos na próxima tela.

A Figura 26, a seguir, exibe os passos anteriores na tela de pré-processamentos.

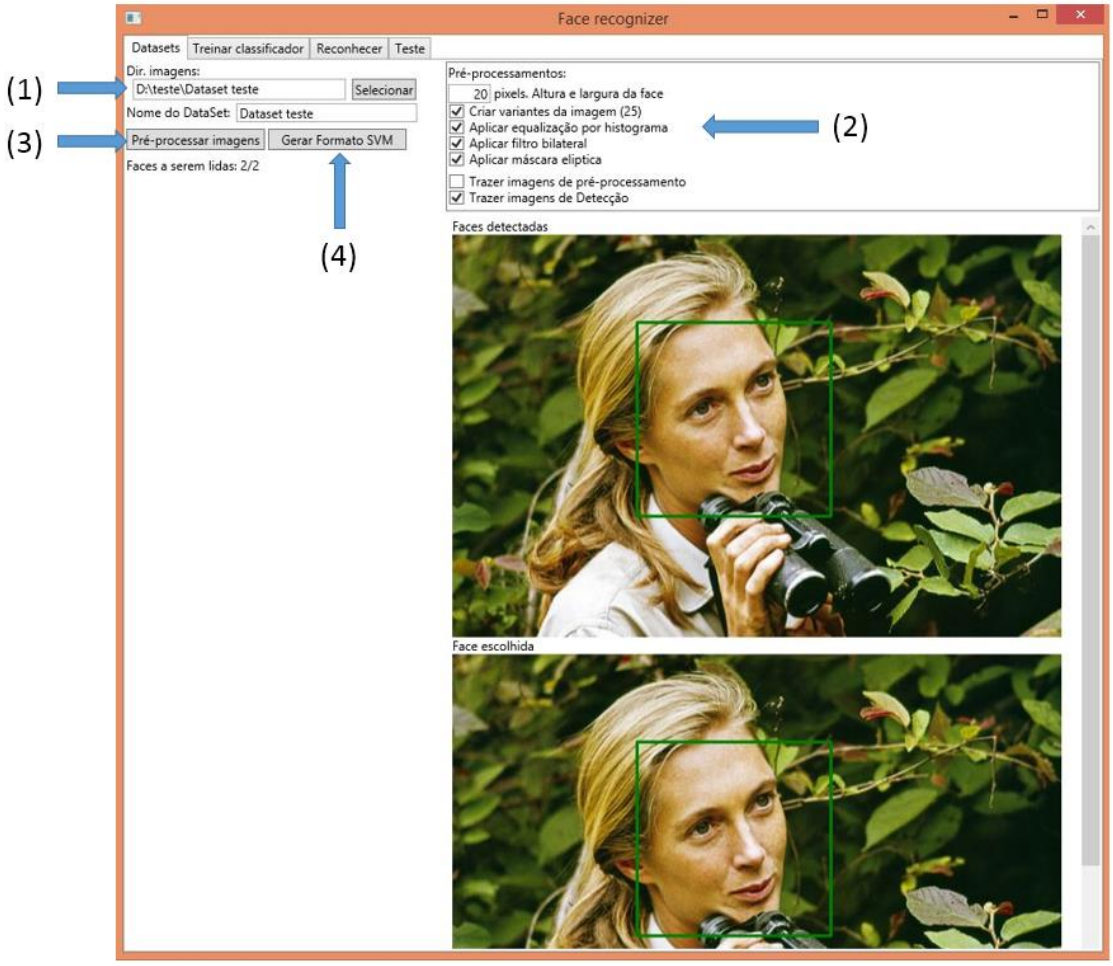


Figura 26 – Tela de pré-processamentos, com passos para execução. Fonte: do autor.

5- Cada imagem do *dataset* tem o rosto e olhos detectados. Caso não encontre automaticamente, é necessário selecionar a face, então o centro do olho a esquerda e então do olho a direita. Para a face, clicar na imagem com o botão esquerdo do mouse, e mantê-lo pressionado. Ao arrastar o mouse, permite escolher o tamanho do quadrado que servirá de delimitador da face. Ao soltar o botão, este tem sua posição e tamanhos finais definidos. Caso esteja correto, pode-se pressionador o botão direito do mouse ou clicar em "Selecionar", para passar para a próxima etapa. Caso não esteja correto, pode-se selecionar um novo quadrado, e o anterior será removido automaticamente. A etapa seguinte é a seleção do centro do olho esquerdo, que, caso não seja o centro adequado, basta clicar novamente que o anterior será removido automaticamente. Após selecionado, resta ainda, de mesma maneira, selecionar o centro do olho à direita da foto. As Figuras 27 e em seguida 28 abaixo, exibem os passos descritos anteriormente.

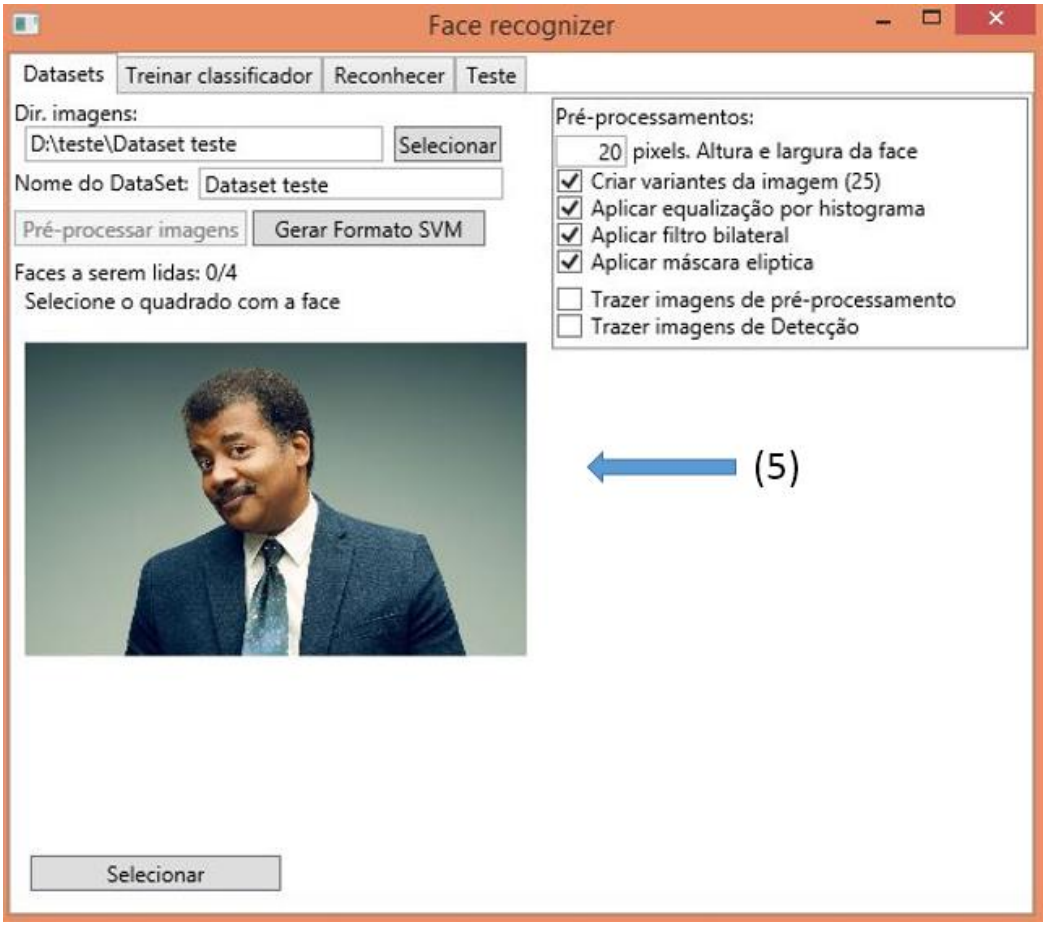


Figura 27 – Exemplo de tela de pré-processamento com seleção manual de face e olhos. Fonte: do autor.

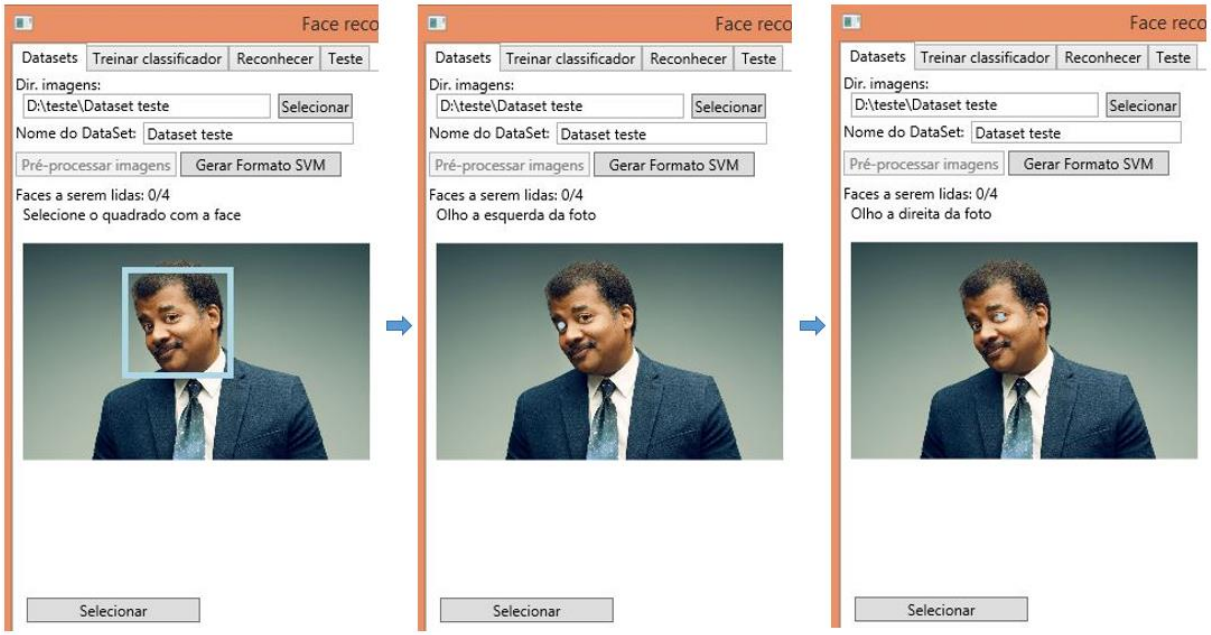


Figura 28 – Processo de seleção manual de face e olhos. Primeiro seleciona-se a face, em seguida centro do olho à esquerda, e por fim centro do olho à direita. Fonte: do autor.

O próximo passo é o treinamento de classificadores, na tela "Treinar classificador". Os passos a seguir são referentes a Figura 29, após estes.

1-Seleciona-se um dos *datasets* pré-processados anteriormente.

2-Então a quantidade de imagens, por individuo, que serão utilizadas para treinar o classificador. As restantes serão utilizadas na próxima tela, caso realize-se o teste automatizado de reconhecimento sob *dataset*.

3-Escolhe-se se deseja-se utilizar de seleção aleatória de quais imagens por indivíduos serão utilizadas para treinamento. Caso opte por não, são utilizadas as primeiras em ordem alfabética no nome do arquivo, na pasta.

4-Realiza-se a geração de arquivos necessários ao treinamento de classificadores.

5-Seleciona-se o tipo de classificador para ser treinado.

6-O tipo de *Kernel*.

7-Para todos classificadores, o valor do parâmetro de regularização, um número real.

8-Parametro Gama do *Kernel* RBF, número real.

9-Normalização dos arquivos de *features*, recomenda-se sempre utilizar.

10- Realizar Treinamento do classificador multi-classe.

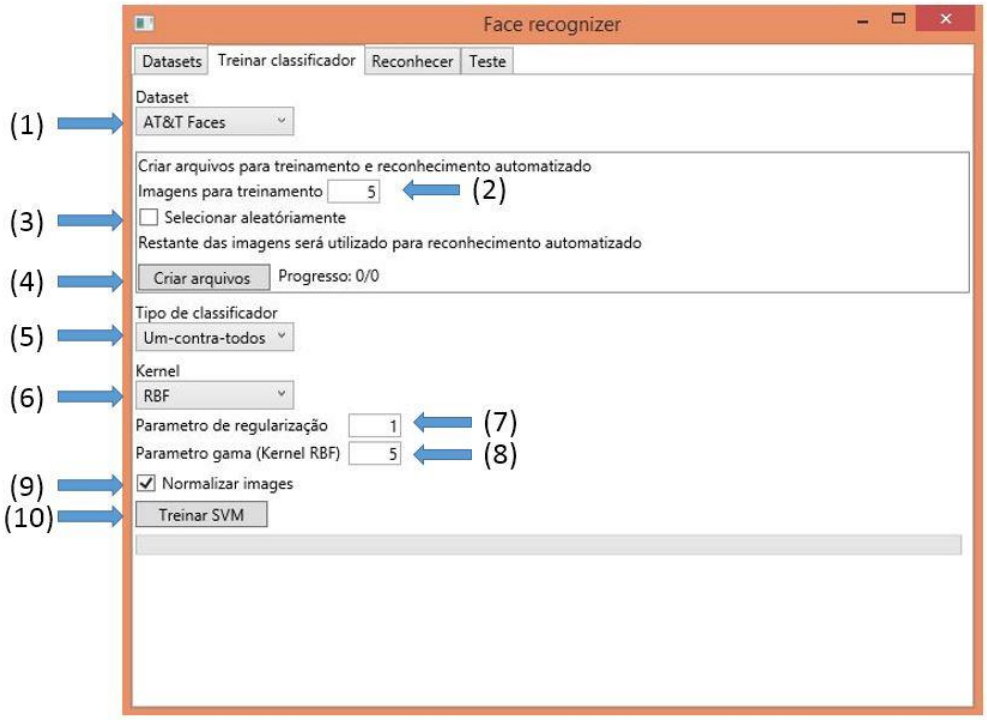


Figura 29 – Tela de treinamento de classificadores com passos para execução. Fonte: do autor.

Por fim é possível realizar o reconhecimento. Os passos a seguir fazem referência a Figura 30, que vem em seguida a estes.

1- Seleciona-se um dos *datasets* processados anteriormente.

2- Seleciona-se o classificador multi-classe.

A seguir, pode-se reconhecer a partir de uma imagem, ou realizar o teste sobre o *dataset.*

3.1.a- Para realizar o reconhecimento por imagem, arrasta-se a imagem para o setor apontado pela legenda (3.1.a).

3.1.b- Caso não detecte automaticamente face e olhos, requer a seleção pelo usuário, como explicado anteriormente em tela de pré-processamento.

3.1.c- Executa-se então o reconhecimento

3.1.d- O resultado do reconhecimento é exibido nesta área, na forma de uma imagem do indivíduo reconhecido.

3.2.a- Outra opção é a execução do teste sobre o *dataset*, onde não são necessários os passos "3.1" anteriores. Executa-se o teste.

3.2.b- Neste setor é gerado o relatório do teste sobre *dataset*.

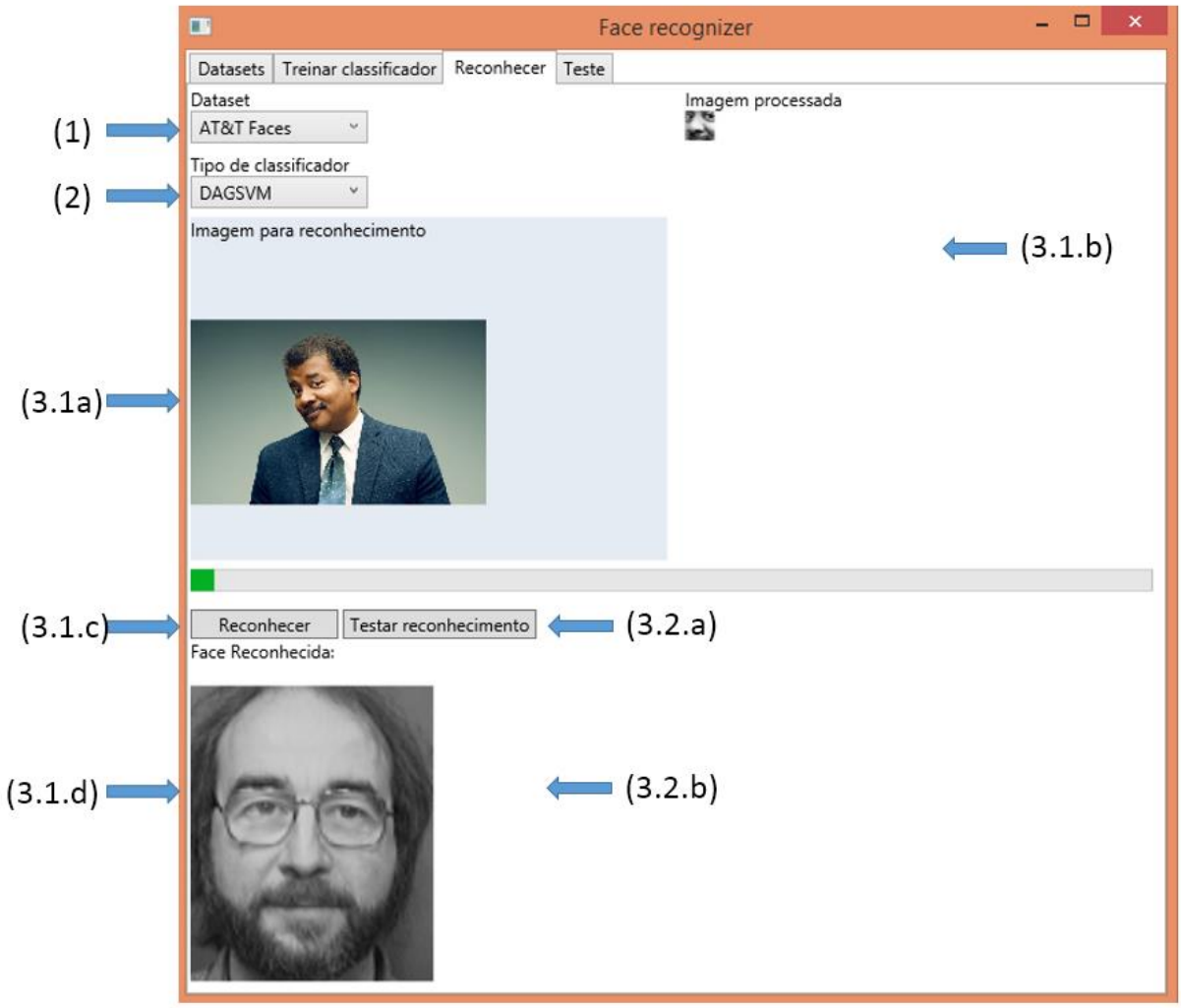


Figura 30 – Tela de reconhecimento, com passos para execução. Fonte: do autor

5.3 Resultados

Nessa seção são apresentados os resultados decorrentes da aplicação desenvolvida. Para tanto, foi empregado um *dataset* internacional que possui 400 imagens distintas, 10 para cada um dos 40 indivíduos. Tal *dataset* é da *The AT&T Database of Faces*. As imagens são frontais e possuem variações de inclinação de face, expressões faciais e algumas possuem a presença de óculos.

Seguem, na Figura 31, exemplos de faces e suas variações:

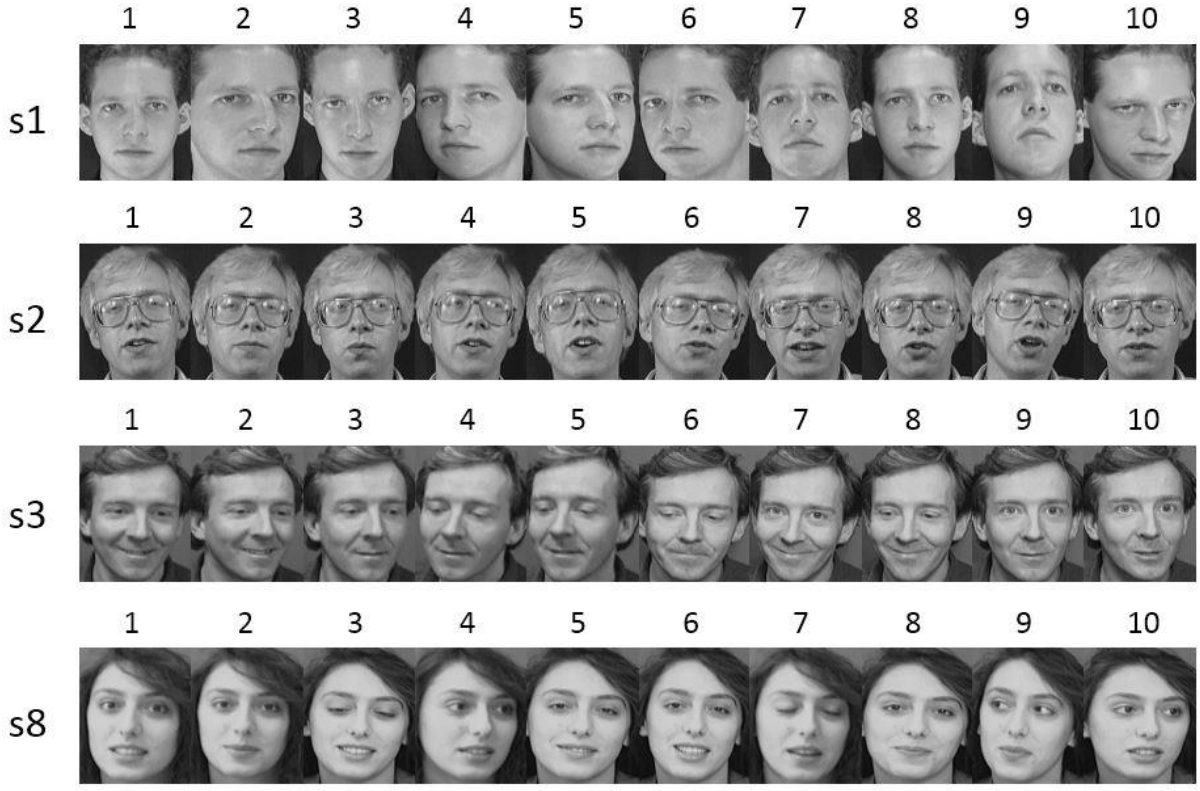


Figura31 – Exemplos de indivíduos *do dataset The AT&T Database of Faces*. A esquerda se encontra a identificação dos mesmos, seguidas das 10 fotos existentes para cada. Fonte: do autor, criada com imagens do *dataset*.

A seguir são descritos os testes de reconhecimento realizados, e a análise do resultado dos mesmos. Para os 4 testes realizados, foi utilizada a mesma configuração de treinamento de classificadores, descrita a seguir.

Parâmetros para o classificador:

Kernel: RBF
Parâmetro de regularização: 1
Normalização: Sim
Parâmetro Gama: 5

Todos os testes utilizaram do pré-processamento transformações geométricas, sendo que este é de uso obrigatório e não selecionável em tela. Para os testes 1 e 2, as etapas de pré-processamento foram mantidas constantes e definidas como:

Tamanho de face pré-processada: 20x20 pixels
Variações de imagens de entrada: Sim
Equalização por histograma: Sim
Filtro Bilateral: Sim
Máscara Elíptica: Sim

Para o teste 1, utilizou-se para treinamento as primeiras 5 imagens de cada sujeito. Já para a etapa de reconhecimento, as 5 restantes de cada, totalizando 200 faces empregadas na validação.

Tabela 1 – Resultados para o teste 1.

	Um-contra-todos	**DAGSVM**	**Um-contra-um**
Acertos	79,500%	91,000%	91,000%
Sem reconhecimento	18,000%	0,000%	0,000%
Falsos positivos	2,500%	9,000%	9,000%
Falsos positivos múltiplos	1,000%	0,000%	0,500%
Falsos positivos múltiplos incluindo o correto	1,000%	0,000%	0,500%
Tempo total (segundos)	2064	471	9487
Tempo total relativo	0,23	0,05	1
Tempo médio por imagem (segundos)	10,32	2,35	47,44

Fonte: do autor.

Para todas as tabelas desta seção, o campo acertos exibe a taxa de reconhecimentos corretos e únicos sobre todas imagens fornecidas para validação. Sem reconhecimento é outra taxa, porém de reconhecimentos que não retornaram resultado. Já falsos positivos são a taxa de reconhecimentos errados, incluindo por retorno de múltiplos resultados para um mesmo indivíduo. Falsos positivos múltiplos são a taxa causada por resultados múltiplos, e "falsos positivos múltiplos incluindo o correto" é a taxa de erros causados apenas pela presença de resultados múltiplos.

O valor tempo total nas tabelas de resultado é o somatório do tempo requerido para o reconhecimento de todas as imagens fornecidas para validação. O tempo total relativo é a comparação entre os tempos dos três classificadores multi-classe. Divide-se cada tempo total pelo maior tempo total dentre os três. Isso é realizado para mostrar a comparação de desempenho, de forma independente do hardware em que foi realizado o teste. O tempo médio por imagem é a divisão do tempo total de cada classificador pela quantidade de imagens para validação.

Após a análise comparativa entre os resultados de DAGSVM e um-contra-um, percebe-se que são os mesmos, à exceção de 0,5% de falsos positivos com mais de um resultado para o classificador um-contra-um. A diferença real está no tempo total relativo. DAGSVM leva 5% do tempo gasto pela classificação um-contra-um, logo sendo 20 vezes mais rápido. Um-contra-todos apresentou taxa 11,5% menor de acerto que os outros, porém a diferença se encontra em obter menos falsos positivos, 2,5% contra 9% dos outros métodos. Seu tempo total relativo foi de 23% do tempo total gasto para realização da classificação um-contra-um, 4 vezes mais rápido.

O teste 2 foi realizado com os mesmos parâmetros de pré-processamento e treinamento de classificadores do Teste 1, porém com a utilização de seleção aleatória de faces para treinamento, na etapa de mesmo nome. Realizou-se o teste 4 vezes, cada um com diferente seleção de faces, e a média dos resultados obtidos é descrita na tabela 2.

Tabela 2 – Resultados para teste 2.

	Um-contra-todos	**DAGSVM**	**Um-contra-um**
Acertos	87,875%	94,250%	94,125%
Sem reconhecimento	11,875%	0,000%	0,000%
Falsos positivos	0,250%	5,750%	5,875%
Falsos positivos múltiplos	0,000%	0,000%	0,500%
Falsos positivos múltiplos incluindo o correto	0,000%	0,000%	0,250%
Tempo total (segundos)	1853	532	9475,75
Tempo total relativo	0,196	0,056	1
Tempo médio por imagem (segundos)	9,265	2,66	47,379

Fonte: do autor.

O teste 2 com uso de aleatoriedade confirmou a comparação anterior de índices de acerto e diferenças de tempo sobre DAGSVM e um-contra-um. Houve ganho significativo de 8,375% na taxa de acertos para um-contra-um, e cerca de 3 % para os outros classificadores. Essa melhora possivelmente deve-se a inserção de faces peculiares para treinamento, que no teste 1 forneceram problemas de reconhecimento ao serem utilizadas para testes de validação.

Os próximos testes realizados, 3 e 4, tiveram por objetivo explorar e comparar o efeito da utilização de pré-processamentos na capacidade de reconhecimento. Para tanto, o teste 3 será realizado com nenhum pré-processamento selecionável, e o teste 4 apenas com a geração de variante da imagem de entrada. Devido a semelhança de resultados ente DAGSVM e um-contra-um, optou-se por não utilizar-se do último, devido à demora excessiva pra conclusão dos testes de reconhecimento.

A seguir, seguem as tabelas 3 e 4, e a explicação dos resultados das mesmas.

Tabela 3 – Resultados para teste 3.

	Um-contra-todos	**DAGSVM**
Acertos	0,750%	72,000%
Sem reconhecimento	99,250%	0,000%
Falsos positivos	0,250%	28,000%
Falsos positivos múltiplos	0,000%	0,000%
Falsos positivos múltiplos incluindo o correto	0,000%	0,000%
Tempo total (segundos)	136	42,25
Tempo total relativo	1	0,31
Tempo médio por imagem (segundos)	0,68	0,21

Fonte: do autor.

Pela tabela 3 pode-se observar que houve perda quase total de funcionalidade do classificador um-contra-todos, com apenas 0,75% de faces corretamente reconhecidas. Porém DAGSVM sofreu perda de apenas 20% da taxa anterior de acerto, mantendo-se viável, e com taxa de 28% de falsos positivos. Ambos classificadores sofreram grande redução de tempo de classificação, 13,6 vezes para um-contra-todos e 12,6 vezes para DAGSVM.

Tabela 4 – Resultados para teste 4.

	Um-contra-todos	**DAGSVM**
Acertos	75,500%	93,625%
Sem reconhecimento	24,250%	0,000%
Falsos positivos	0,250%	6,375%
Falsos positivos múltiplos	0,125%	0,000%
Falsos positivos múltiplos incluindo o correto	0,125%	0,000%
Tempo total (segundos)	1055,5	270,75
Tempo total relativo	1	0,26
Tempo médio por imagem (segundos)	5,2775	1,35

Fonte: do autor.

A tabela 4 mostra a alta taxa de acertos encontrada em média pelo DAGSVM, apenas com a adição da criação de variantes, o que mostrou que pré-processamentos selecionáveis, à exceção de geração de variantes na imagem, possuem pouca relevância para o reconhecimento. O mesmo pode ser dito sobre o classificador um-contra-todos, que de quase 0% de acertos passou para 75,5%. Testes futuros precisam ser realizados para apurar se esta é uma situação gerada pela baixa quantidade de imagens originais para treinamento e testes (5 de cada), ou se é possível generalizar que tais pré-processamentos não são relevantes para classificadores SVM.

6. CONCLUSÃO

Neste trabalho foram vistas algumas técnicas existentes para reconhecimento de faces, e métodos de classificação baseados em SVMs.

O fato da área de reconhecimento de faces já ser madura, implicou na existência de diversos pacotes de acesso livre com métodos prontos para uso. A biblioteca OpenCV conta com algumas implementações relacionadas, incluindo métodos para detecção de faces, o que permitiu remover o foco desta questão e trabalhar apenas com o reconhecimento. Semelhantemente, para a questão de classificação por SVMs, encontrou-se a biblioteca LIBSVM, com implementações diversas de classificadores baseados em SVM, requisitando o desenvolvimento das estruturas multi-classe.

No caso da aplicação específica de SVMs como método de aprendizado para reconhecimento de faces, foram encontrados poucos trabalhos recentes, sendo a maioria até 2004.

O sistema foi implementado, permitindo a utilização de diferentes bases de imagens de faces, tratamentos de imagens e parâmetros para os classificadores SVM, de forma a permitir a exploração desse tipo de aprendizado de máquina aplicado ao reconhecimento de faces.

Das implementações de classificação multi-classe testadas, DAGSVM se mostrou superior, tanto em taxas de reconhecimento (até 94,25%), quanto velocidade, inclusive levando 5% do tempo de execução de um-contra-um, comparado também à um-contra-todos, que levou 20% do tempo de execução de um-contra-um.

Um-contra-um possui taxas de acerto semelhantes a DAGSVM, diferindo apenas na possibilidade de retorno de mais de um indivíduo por reconhecimento. Esta característica pode ser removida se for utilizado algum critério arbitrário de desempate nos resultados da votação. Um-contra-todos apresentou menor taxa de falsos positivos, pois permite o retorno de nenhum resultado. Ainda para este, a quantidade de dados para treinamento de cada classificador individual é 20 vezes superior à dos outros dois classificadores. Isto leva à necessidade de maior exploração dos diferentes *Kernels* disponíveis e valores para o parâmetro de regularização C.

Os testes 3 e 4, na seção resultados demonstraram que o pré-processamento geração de variantes da face original possui grande impacto na qualidade do reconhecimento. Já a equalização por histogramas, filtro bilateral e aplicação de máscara elíptica possuem efeito ínfimo, para a escala de quantidade de imagens utilizadas e sem utilização de métodos de

redução de dimensionalidade, como PCA. É necessário realizar testes para descobrir se existe algum momento em que passam a possuir relevância.

Dos testes realizados, confirmou-se a importância da posição da face e das expressões faciais. Para todos os classificadores SVM, alguns testes de classificação para o indivíduo s3, disponível na Figura 19, não conseguiram identificá-lo em imagens em que o indivíduo aparece com olhos abertos, quando todas as imagens de treinamento foram com olhos fechados.

Quanto ao tamanho das imagens pré-processadas utilizadas pra treinamento, foram testados tamanhos de 10x10, 20x20, 30x30 e 40x40 pixels. O tamanho que forneceu melhores resultados foi de 20x20. Os tamanhos de 30x30 e 40x40, apresentaram perda de acertos e maior demora nos processos de treinamento e reconhecimento. O tamanho de 10x10, apesar de mais rápido, possuiu grande diminuição nas taxas de acerto.

Para trabalhos futuros, existem itens a serem explorados tanto na detecção quanto no treinamento de classificadores. A detecção pode ser melhorada para permitir a obtenção de componentes da face, cujo treinamento sobre esses pode melhorar a tolerância do reconhecedor à rotações horizontais ou verticais de uma face. Pode-se também explorar o uso de detectores de corpo, para aumentar a certeza de que objeto detectado de fato é uma face, caso detecte-se corpo abaixo e próximo da face.

Testes preliminares utilizando *Kernel* polinomial mostraram-se mais rápidos que o *Kernel* RBF, apesar de perda nas taxas de acerto. A utilização de *Kernels* diferentes e busca automatizada de valores ótimos para o parâmetro de regularização C (e.g. *grid-search*) são também uma excelente opção pra trabalhos futuros.

Por fim, pode-se estudar a utilização de redução de dimensionalidade/extração de características principais, como PCA e *Wavelets*, de modo a tornar o reconhecimento possivelmente mais rápido e correto.

REFERÊNCIAS

1. CHANG, C.-C; LIN, C.-J; *LIBSVM:* A Library for Support Vector Machines, 2013. Disponível em <http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf>. Acesso em 30 de mai. 2015.

2. CORTES, C; VAPNIK, V. *Support-Vector Networks*, Machine Learning, Vol. 20, p. 273-297, (1995). Disponível em <http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf>

3. DEGTYAREV, N; SEREDIN, O. *Comparative Testing of Face Detection Algorithms*, Tula State University. Disponível em <http://lda.tsu.tula.ru/papers/degtyarev-2010-icisp-ctfd.pdf>. Acesso em 30 de abr. 2015.

4. FRIEDMAN, J. *Another Approach to Polychotomous Classification*, Dept. Statist., Stanford Univ., Stanford, CA., 1996 disponível em <http://www-stat.stanford.edu/reports/friedman/poly.ps.Z>. Acesso em 7 de jun. 2015.

5. GUO, G; LI, S. Z; CHAN, K. *Face Recognition by Support Vector Machines,* School of Electrical and Electronic Engineering Nanyang Technological University, 2000. Disponível em < http://pages.cs.wisc.edu/~gdguo/myPapersOnWeb/FG2000Guo.pdf>. Acesso em 26 de abr. 2015.

6. HEISELE, B; HO, P; POGGIO, P. *Face Recognition with Support Vector Machines:* Global versus Component-based Approach, Massachusetts Institute of Technology Center for Biological and Computational Learning, 2001. Disponível em < http://cbcl.mit.edu/cbcl/publications/ps/iccv2001.pdf>. Acesso em 26 de abr. 2015.

7. HUA, G; LEARNED-MILLER, E; TURK, M; HUANG T. S. *Introduction to the Special Section on Real-World Face-Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, Vol.33, No. 10. . Disponível em <http://ieeexplore.ieee.org/iel5/34/5989906/05989909.pdf?arnumber=5989909>. Acesso em 24 de mar. 2015.

8. HSU, C.-W; LIN, C.-J. *A Comparison of Methods for Multiclass Support Vector Machines*, IEEE Transactions on Neural Networks, Vol. 13, No. 2, 2002. Disponível em <http://cs.ecs.baylor.edu/~hamerly/courses/5325_11s/papers/svm/hsu2001multiclass.pdf> . Acesso em 30 de mai. 2015.

9. KABEER, V; *Computer recognition of human face image using nonlinear dynamical system and wavelet based artificial light receptor models*, Department of Information Technology. Kannur University, 2010. Disponível em <http://hdl.handle.net/10603/2538>. Acesso em 20 de abr. 2015.

10. LIENHART, R; MAYDT, J. *An extended set of Haar-like features for rapid object detection*, Proc. of ICIP, 2002. Disponível em : <http://www.multimedia-computing.de/mediawiki//images/5/52/MRL-TR-May02-revised-Dec02.pdf> Acesso em 7 de jun. 2015.

11. PLATT, J. C; CRISTIANI, N; SHAWE-TAYLOR, J. *Large margin DAG's for multiclass classification*, Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2000, Vol. 12, p. 547–553. Disponível em <http://research.microsoft.com/pubs/68541/dagsvm.pdf>. Acesso em 7 de jun. 2015.

12. SENIOR, A. W; BOLLE, R. M. *Face recognition and its applications*, Biometric Solutions for Authentication in an E-World, Cap. 4, 2002. Disponível em <http://andrewsenior.com/papers/SeniorB02FaceChap.pdf>. Acesso em 30 abr. 2015.

13. VIOLA, P; JONES, M. *Rapid Object Detection using a Boosted Cascade of Simple Features*, Conference on Computer Vision and Pattern Recognition 2001. Disponível em <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>. Acesso em 23 de mai. 2015.

14. WOLF, L; HASSNER, T; MAOZ, I. *Face Recognition in Unconstrained Videos with Matched Background*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pg. 529-534. Disponível em <http://www.openu.ac.il/home/hassner/projects/ytfaces/ytfaces.pdf>. Acesso em 26 de mar. 2015.

15. YANG, M.-H; KRIEGMAN, D. J; AHUJA, N. *Detecting faces in images:* A survey. IEEE Trans. on PAMI, Vol. 24(1), p.34–58, 2002. Disponível em <http://vision.ucsd.edu/sites/default/files/pami02.pdf>. Acesso em 16 mai. 2015.

16. ZHANG, C; ZHANG, C. *A Survey of Recent Advances in Face Detection*, Microsoft Research, Microsoft Corporation, 2010. Disponível em <http://research.microsoft.com/pubs/132077/facedetsurvey.pdf>. Acesso em 15 mai. 2015.

17. ZHANG, N; PALURI, M., TAIGMAN, Y; FERGUS, R; BOURDEV L. *Beyond Frontal Faces:* Improving Person Recognition Using Multiple Cues Accessing outcomes and processes for deep approaches to learning. Disponível em <http://arxiv.org/abs/1501.05703>. Acesso em 16 de mar. 2015.

18. ZHAO, W; CHELLAPA, R; ROSENFELD, A; PHILLIPS P. J. *Face Recognition:* A Literature Survey, ACM Computing Surveys, 2003, pg. 399-458. Disponível em <http://mplab.ucsd.edu/~marni/Igert/Zhao_2003.pdf>. Acesso em 18 de mar. 2015.

19. BAGGIO, D. L; ESCRIVÁ, D. M; MAHMOOD, N; SHILKROT, R; EMAMI, S; IEVGEN, K; SARAGIH, J. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing, 2012. ISBN 978-1-84951-782-9 Chapter 8. Face Recognition using Eigenfaces or Fisherfaces.