colloquium exactarum

RECONHECIMENTO FACIAL EM IMAGENS CAPTURADAS POR CÂMERAS DIGITAIS DE REDE

FACIAL RECOGNITION IN IMAGES CAPTURED BY DIGITAL CAMERAS NETWORK

Rogerio Kazuhiro Okabe[1;] Silvio Antonio Carro[2]

Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática – FIPP, Presidente Prudente, SP. e-mail: silvio@unoeste.br

RESUMO - Este trabalho apresenta um estudo e um protótipo para o reconhecimento facial humano utilizando imagens capturadas por câmeras de segurança IP. A metodologia do trabalho faz uso dos recursos fornecidos pela biblioteca OpenCV em conjunto com a biblioteca JavaCV para a detecção e reconhecimento facial, onde foram explorados os algoritmos de HaarCascade, Eigenface, Fisherface e LBPH. Para o desenvolvimento do trabalho foi implementado um protótipo com o objetivo de identificar indivíduos em meio a um grupo de pessoas. Os testes foram realizados com alunos da Faculdade de Informática de Presidente Prudente/Unoeste e os resultados da precisão de reconhecimento foram relatados e apresentados nesse estudo.
Palavras-chave: câmera de rede; reconhecimento facial; opencv; javacv.

ABSTRACT - This paper aims to present a study and a prototype for human face recognition using captured by security cameras IP images. The methodology for the study makes use of the resources provided by the OpenCV library in conjunction with JavaCV library for detection and face recognition, where HaarCascade, Eigenface, Fisherface and LBPH algorithms were explored. For the development of a prototype work in order to identify individuals in the midst of a group of people was implemented. The tests were conducted with students of the Faculty of Informatics of Presidente Prudente / Unoeste and the results of recognition accuracy were reported and presented in this study.
Keywords: network camera; face recognition; opencv; javacv.

Recebido em: 02/10/2014
Revisado em: 25/10/2014
Aprovado em: 27/11/2014

1 INTRODUÇÃO

O reconhecimento facial é uma técnica de biometria baseada em traços do rosto humano. O processo de reconhecimento é realizado a partir de pontos de medida do rosto, que fazem uma ligação algorítmica de traços e tamanhos, como exemplo pode-se citar a distância exata entre nariz e orelhas, tamanho do crânio, arcada dentária, entre outros detalhes.

Um dos fatores que diferenciam o reconhecimento facial dos outros sistemas biométricos reside no fato desta tecnologia poder ser utilizada para fins de vigilância. Uma das maiores vantagens desta tecnologia está na capacidade de capturar faces em locais públicos à distância, possibilitando assim minimizar as limitações jurídicas.

O presente trabalho propõe utilizar imagens capturadas de câmeras de rede, que são imagens populares da área de vigilância e geralmente com resolução mediana e assim processá-las com recursos fornecidos pela biblioteca OpenCV/JavaCV, avaliando a eficiência dos algoritmos em relação ao reconhecimento facial de indivíduos em uma multidão.

O OpenCV é gratuito tanto para uso acadêmico e comercial. Possui interfaces de C, C ++, Python e Java, projetada para a eficiência computacional e com foco em aplicações em tempo real (OPENCV, 2013a). O JavaCV é um *wrapper* que permite acessar a biblioteca OpenCV diretamente de dentro do Java e Android. JavaCV envolve C API, e C ++ API, quando necessário.

Para validação dos testes, foi desenvolvido um protótipo na linguagem Java com vários módulos, dentre eles pode-se destacar o módulo de acesso e captura de frames por endereço IP; o módulo de adequação de imagens para o reconhecimento, onde foram testados vários filtros e o módulo de detecção para então realizar o reconhecimento facial por meio dos métodos disponíveis no OpenCV Eigenface, Fisherface e LBPH. Os módulos foram organizados na forma de classes e podem ser facilmente reutilizados em outros produtos.

2 TRABALHOS RELACIONADOS

Encontra-se no mercado alguns produtos que exploram diretamente o reconhecimento facial de indivíduos em multidões. Dentre eles pode-se destacar o BrFace, o FaceIt e o Checkin. A maioria deles utilizam tecnologia proprietária, ao contrário da presente proposta que utiliza solução *open source* e gratuita.

2.1 Brface

De acordo com BrScan (2013), são desenvolvidos aplicativos de publicidade

direcionada até segurança pública e privada, em conjunto com a Cognitec do Brasil.

A tecnologia de reconhecimento facial pode ser aplicada em diversas ocasiões distintas, sendo o principal mercado hoje relacionado à segurança, no entanto há uma variedade de aplicativos relacionados ao uso pessoal, aumento de produtividade, entre outros (COGNIEX, 2013).

O aplicativo detecta as faces das pessoas no vídeo em tempo real contra as bases de dados das imagens para encontrar a pessoa conhecida. Inclui-se um kit integrado para o sistema de vídeo incluindo C++ e API de serviços web (COGNIEX, 2013).

O FaceVACS-SDK é uma ferramenta de desenvolvimento para provêr o algoritmo nativo do reconhecimento facial para integradores do sistemas. A tecnologia patenteada do FaceVACS da Cognitec Systems reconhece os rostos com precisão independente da variações do rosto. Esta tecnologia manipula o posicionamento, variação de idade, estilos e alterações de luminosidade (COGNIEX, 2013).

2.2 Faceit

A empresa Visionics da Nova Jersey desenvolveu a tecnologia de reconhecimento facial a FaceIt extraindo o rosto de uma pessoa da multidão e comparando com um banco de dados de faces. São baseados em aspectos definidos como pontos nodais podendo se a distâncias entre os olhos, comprimento do nariz, cavidade orbital, ossos laterais da face, linha da mandíbula, queixo, entre outros (BONSOR, 2013).

O software baseia-se no reconhecimento facial e avalia as técnicas distintas de cada rosto necessitando de 14 a 22 pontos nodais para completar o processo do reconhecimento facial. É obtido uma assinatura facial através dos pontos medidos para criar um código numérico representando o rosto no banco de dados (BONSOR, 2013).

2.3 Check-in

A agência Redpepper utilizou a tecnologia de código aberto OpenCV e Facebook Graph API incluindo o Raspberry Pi, Arduino, e câmeras com acesso a uma conexão Wi-Fi utilizam aplicativos check-in, seja pelo FourSquare ou diretamente pelo Facebook, expondo a localização atual dos usuários com o objetivo de apontar onde a pessoa está (2ND DESIGN DIGITAL, 2013).

As câmeras são instaladas nos locais físicos e após a identificação do consumidor em seu local, seu check-in é realizado a partir do app instalado em seu Smartphone, e com base em seu histórico de curtidas no Facebook ele ainda pode receber promoções ou descontos no estabelecimento (2ND DESIGN DIGITAL, 2013).

Tornando a novidade comportamento dos internautas e as empresas passaram a aproveitar desses check-in para promover promoções e descontos fora do ambiente (2ND DESIGN DIGITAL, 2013).

3 DESENVOLVIMENTO

Como forma de testar e validar as técnicas de reconhecimento facial disponíveis na biblioteca OpenCV foi desenvolvido um protótipo na linguagem Java, organizado em módulos, os quais serão discutidos nessa Seção.

3.1 Visão geral

O protótipo foi desenvolvido utilizando a linguagem Java em conjunto com as bibliotecas OpenCV 2.4.5 e JavaCV 0.5. As imagens foram capturadas por meio de uma câmera IP modelo M1031 fabricada pela Axis Comunnications. A câmera foi posicionada em um corredor de acesso aos laboratórios da Faculdade de Informática de Presidente Prudente da Universidade do Oeste Paulista. As imagens capturadas possuem resolução de 640x480 pixels.

O diagrama de classes do protótipo pode ser visualizado na Figura 1 e pode-se identificar a implementação de dois módulos distintos: módulo de acesso à câmera (classe AxisCamera); módulo para detectar as faces (classe FaceDetection) para então executar os métodos do OpenCV de reconhecimento. A classe Tools representa um módulo de apoio com funcionalidades para preparação da imagem e a classe VideoDetectarFace representa uma thread de captura dos frames da câmera.

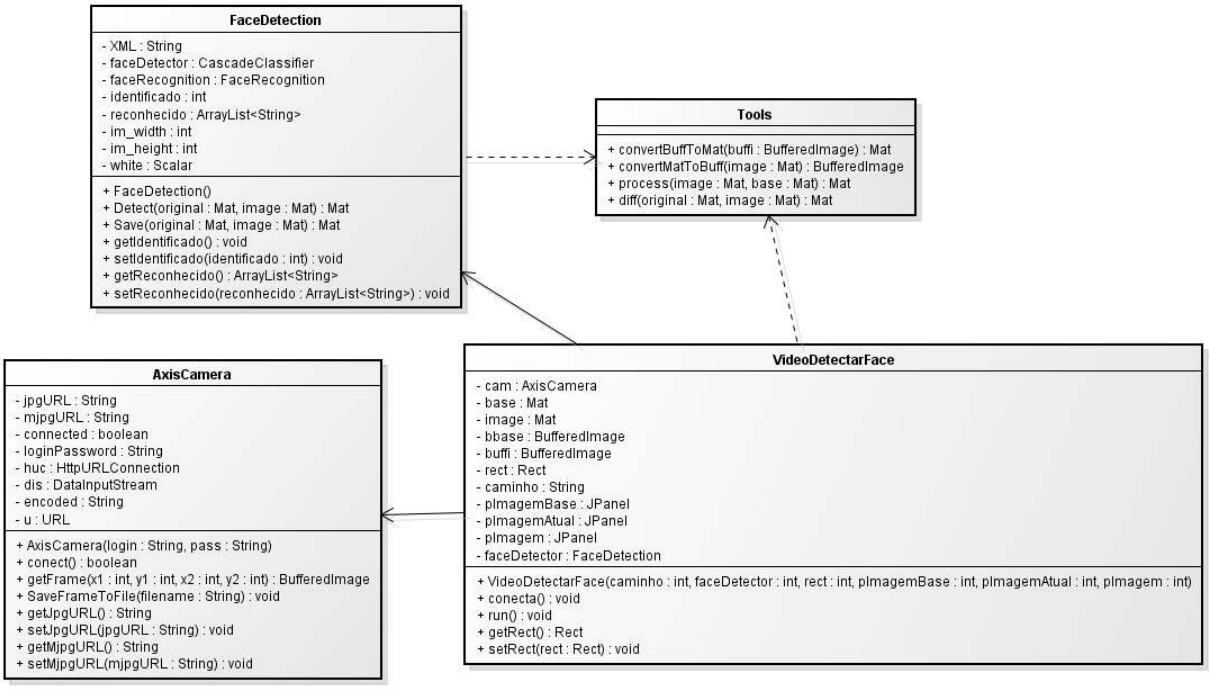

Figura 1. Diagrama de classes.

3.2 Classe axiscamera: módulo de acesso a câmera

O acesso à câmera é realizado por meio dos objetos da classe AxisCamera. A classe acessa a câmera através de um endereço IP, capturando e retornando os frames atuais.

A Figura 2 apresenta os passos que envolvem a criação do objeto, sua configuração e a aquisição de um frame.

```
cam=new AxisCamera("usuario","senha");
cam.setJpgURL("http://177.131.34.241/axis-cgi/jpg/image.cgi");
cam.conect();
...
BufferedImage buffi=cam.getFrame(0,0,0,0);
```

Figura 2. Exemplo do algoritmo para acesso à câmera IP.

3.3 Classe tools: módulo de pré-processamento de imagem

Os frames capturados são submetidos às técnicas de pré-processamento de imagem e assim diminuir o tempo de processamento e melhorar a precisão durante a identificação das faces.

Para a diminuição de tempo foi implementado delimitações de área, eliminando assim processos em locais no ambiente sem acesso humano.

A subtração de fundos calcula a máscara do plano realizando uma subtração entre o quadro atual e um modelo de fundo, contendo a parte estática da cena ou, mais em geral, tudo o que pode ser considerado como fundo dadas as características da cena observada.

A função suaviza uma imagem usando o filtro de mediana e luminância para diminuir aspectos que possam dificultar a detecção da face. O threshold é utilizado para transformar uma escala de cor cinza em preto e branco, removendo áreas indesejadas.

O Fechamento é composto pela dilatação seguida da erosão para suavizar o contorno, quebra istmos estreitos e elimina protrusões finas. A Erosão remove picos de bordas, aumenta buracos internos e reduz a região enquanto a dilatação preenche buracos e aumenta a região.

Por fim foi realizada a diferença entre a imagem pré-processada e a original. Obtendo assim apenas as diferenças entre o fundo e as pessoas da imagem. Para uma melhor compreensão os passos são apresentados na Figura 3.

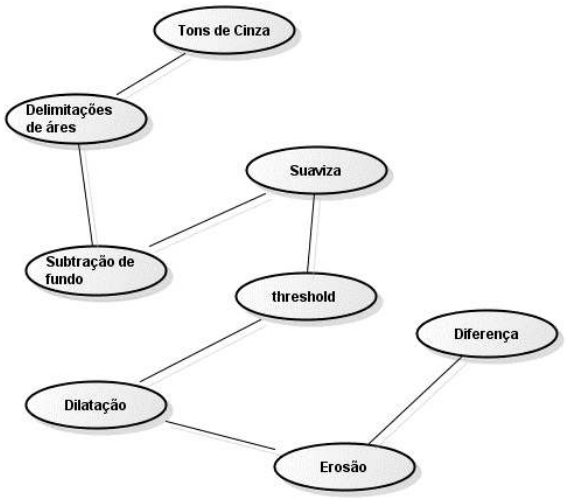

Figura 3. Passo a passo do pré-processamento.

3.4 Classe facedetection: módulo de detecção de faces

Com base em recursos Haar Cascade classifier para detecção de objetos, foi proposto por Paul Viola e melhorado pelo Rainer Lienhart. É uma abordagem baseada em aprendizado de máquina onde uma função é treinada a partir de um conjunto de imagens positivas e negativas para então extrair características.

O OpenCV já contém vários classificadores pré-qualificados para a detecção de rosto, olhos, mão, entre outros, em arquivos XML armazenadas na pasta opencv/data/haarcascade. O arquivo XML utilizado no projeto foi o haarcascade_frontalface_alt2.xml contendo as características do rosto. Esse arquivo XML é então carregado para detecção da face frontal através de uma imagem. Quando há a detecção da face na imagem é retornando retângulos contendo as posições dos rostos detectados na imagem, para então posteriormente serem reconhecidas.

A Figura 4 apresenta os passos que envolvem de detecção a partir de uma imagem salvando os resultados na variável faceDetections, contendo a quantidade de faces detectadas e o posicionamento dos mesmos na imagem.

```
//carrega o arquivo para detecção da face
String XML="C:/opencv/data/haarcascades/haarcascade_frontalface_alt2.xml";
CascadeClassifier faceDetector = new CascadeClassifier(XML);

//detecta várias faces
MatOfRect faceDetections = new MatOfRect();
faceDetector.detectMultiScale(image, faceDetections);

//quantidade de faces detectadas
int face=faceDetections.toArray().length;

//quadros das faces detectadas
Rect rect[] = faceDetections.toArray();
```

Figura 4. Exemplo do algoritmo para detecção facial.

3.5 Reconhecimento facial

O sistema permite o treinamento, onde são necessárias para as preparações de reconhecimento facial com o agrupamento das amostras que desejam ser reconhecida, a identificação da face classifica as faces das imagens para o reconhecimento e a classificação contempla a busca das imagens de treino além da classificação e rotulação das faces.

O treino é realizado a partir de um arquivo texto, contendo o índice da pessoa, e o endereço da imagem Portable Network Graphics (PNG). As imagens do conjunto de treinamento consistem em imagens de 200 x 200 pixels em tons-de-cinza no formato PNG que são apresentadas na Figura 5.

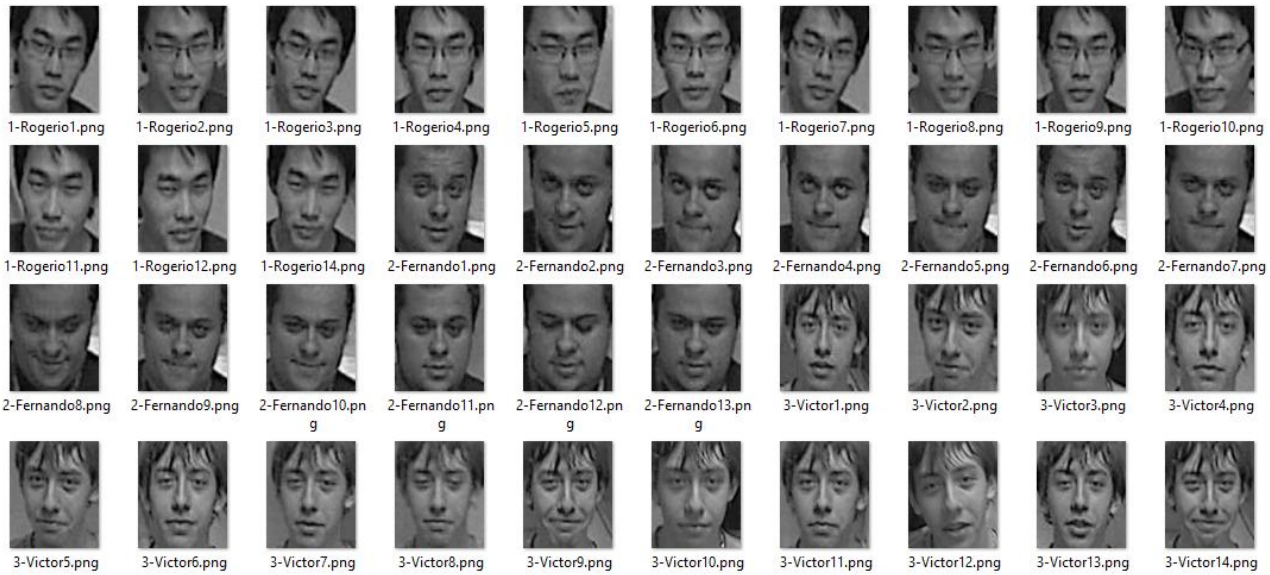

Figura 4. Conjunto de uma determinada face utilizado para o treinamento.

A partir do OpenCV 2.4 possibilita o uso do FaceRecognizer, uma classe para o reconhecimento facial. Os algoritmos atualmente disponíveis são o Eigenfaces, Fisherfaces, e Padrões binários locais de histogramas (OPENCV, 2013b), que serão abordados nos próximos tópicos.

A partir dos cálculos gerados pela fase de treinamento foi-se gerado um arquivo Extensible Markup Language (XML) contendo todas as características da face. Esse mesmo arquivo é então carregado para o reconhecimento da face, eliminando todo o processo de treinamento sempre que o programa for utilizado.

A face é classificada como pertencente quando o valor mínimo for abaixo de um valor de corte, apresentado uma rotulação da pessoa que acessou o ambiente, caso contrário é classificada como desconhecida e não apresenta a rotulação.

3.5.1 Eigenface

O Eigenfaces foi desenvolvido por Sirovich e Kirby (1987) e utilizado por Matthew Turk e Alex Pentland para classificação face. A técnica considera o reconhecimento facial como um problema de reconhecimento em 2-D, com imagens das faces projetadas em um espaço de características melhorando a representação da variação entre as faces já conhecidas (ALBERGARIA; SANTOS; ALVIM JÚNIOR, 2013), esse espaço é definido como Eigenfaces, que são os autovetores do conjunto das faces, para então ser comparando a posição obtida com a posição de indivíduos já conhecidos (GUIMARÃES, 2010).

O reconhecimento ocorre a partir do conjunto de treino calculando as eigenfaces. No momento em que uma nova face é encontrada é calculado um conjunto de pesos baseados nas entradas e eigenfaces através da projeção da imagem de entrada em cada eigenfaces. É verificado se a imagem está suficientemente próxima do espaço de faces para determinar se é uma face ou não. Se a imagem for uma face é classificada o padrão de pesos como uma pessoa conhecida ou não (ALBERGARIA; SANTOS; ALVIM JÚNIOR, 2013).

Também conhecido como Principal Component Analysis (PCA) é um dos métodos mais clássicos com ótimos resultados (DANTAS; BATISTA, 2013).

3.5.2 Fisherface

O Eigenface encontra uma combinação linear de funções que maximiza a variância total nos dados. É uma forma poderosa de representar os dados, mas não considera as classes e várias informações de discriminação podem ser perdidas quando se descartam esses componentes, em situação onde a variação vem de uma fonte externa como a luz. Os componentes identificados não necessariamente contêm nenhuma informação discriminativa, então as amostras projetadas, são manchadas em conjunto e uma classificação torna-se impossível (OPENCV, 2013b).

O método Fisherfaces ou Linear Discriminant Analysis (LDA) executa uma redução de dimensionalidade de classe específica inventado pelo estatístico Sir RA Fisher, usado com sucesso para a classificação de flores em seu artigo “O uso de múltiplas medições em problemas taxonômicos” em 1936 e aplicada pelo Belhumeur, Hespanha e Kriegman uma análise discriminante para o reconhecimento facial (OPENCV, 2013b).

Fisherface é um método utilizado de extração de características e redução de dimensionalidade em reconhecimento de padrões. Tenta encontrar a melhor direção

da projeção em que amostras de treinamento pertencentes a diferentes classes são mais bem separados (SHAN et al., 2014), maximizando a diferença entre as classes e minimizando a diferença dentro das classes (SINFIC, 2013).

O método aprende uma matriz de transformação de classe específica, de forma a não captarem a iluminação. O desempenho do Fisherface depende muito dos dados de entrada. Se o treinamento for realizado apenas fotografias com boa iluminação e os experimentos em faces com má iluminação, o método se torna susceptível de encontrar os componentes errados (OPENCV, 2013b).

3.5.3 Padrões binários locais

Eigenface e Fisherface possuem uma abordagem holística, tratando os dados como um vetor em um espaço de alta dimensão. Como a alta dimensionalidade é ruim, um subespaço de menor dimensão é identificado, preservando somente as informações úteis. A abordagem Eigenface maximiza a dispersão total, podendo conduzir a problemas se a variância for gerada por uma fonte externa, por causa de componentes com um desvio máximo de mais de todas as classes não são necessariamente útil para a classificação. Então, para preservar algumas informações discriminativas foi aplicado a LDA, conforme a Seção 3.5.2 (OPENCV, 2013b).

Os Padrões binários locais descrever apenas as características locais de um objeto, reduzindo a dimensionalidade, resumindo a estrutura local em uma imagem comparando os pixels com a sua vizinhança. Se a intensidade do pixel central é maior, igual ao seu vizinho, denota-se como 1 caso contrário 0 (WAGNER, 2014).

A representação proposta por Ahonen é dividir a imagem LBP em regiões locais e extrair um histograma cada. O vetor de características espacialmente reforçadas é obtido concatenando os histogramas locais. Estes histogramas são chamados Local Binary Patterns Histograms (LBPH) (OPENCV, 2013b).

3.6 Protótipo Facedetect

O Para testar as soluções desenvolvidas, foi criado o protótipo FaceDetect (Figura 5). O protótipo além de proporcionar a experimentação das soluções, também propiciou a criação de estatísticas sobre a precisão no reconhecimento facial, descritas na Seção 4.

Algumas funcionalidades do protótipo:

- Conexão com qualquer câmera IP, dados o endereço (url).
- Captura de fames “ao vivo”
- Captura de frames de um vídeo pré gravado.

- Treinamento a partir de coleções de imagens do tipo PNG.
- Apresentação da imagem base, imagem atual e a imagem resultante do pré-processamento, ou seja, a imagem a ser utilizada na detecção das faces.

O protótipo ainda mostra as faces detectadas e se reconhecida, o nome do indivíduo.



Figura 5. Interface com o usuário.

4 EXPERIMENTOS E RESULTADOS

Os experimentos foram conduzidos no sentido de testar a capacidade de detecção e reconhecimento das funções embutidas no OpenCV. Outro parâmetro de teste foi a adoção ou não de filtros utilizados para preparar as imagens.

De acordo com OpenCV (2013b) os testes realizados com 10, 30, ..., 310, Eigenfaces, mostrou-se que 10 autovetores não foram suficientes para uma boa reconstrução da imagem, 50 autovetores já pode ser suficiente para codificar características faciais importantes. Uma boa reconstrução poder ser feita com cerca de 300 autovetores. Dependendo muito das entradas dos dados.

Foram utilizados os métodos Eigenface, Fisherface e LBPH e um banco de

faces composto por três pessoas, e cada qual representada por 300 características faciais.

Três conjuntos de frames foram gerados a partir da captura direta da câmera IP. Os vídeos foram denominados de vídeo1, video2 e video3. O video1 com duração de 22 segundos representa um cenário com onze pessoas com faces frontais dentre elas, três previamente reconhecidas. O video2 com duração de 19 segundos possui quatorze pessoas acessando o ambiente dentre elas nove com faces frontais e somente duas cadastradas. O video3 com duração maior __ 1 minuto e 02 segundos __ conta com dezoito pessoas acessando ambiente das quais treze apresentando faces frontais e nenhuma delas cadastrada.

Os filtros utilizados são os mesmos apresentados na Seção 3.3.

A utilização dos filtros obteve grande diferença nos resultados, limitando quantidades de identificações falsas como reflexos e áreas sem acesso humano. E consequentemente melhorando o tempo de execução, desconsiderando os processos de reconhecimento em locais onde não havia faces.

O arquivo de treino conforme descrito na Seção 3.5 geram arquivos XML, cada método gera diferentes tamanhos de arquivos dependendo da quantidade de arquivos de treino. Influenciando no tempo de carregamento do mesmo.

As Tabelas 1, 2 e 3 mostram os resultados dos vídeos quanto as faces identificadas corretamente, cadastradas e reconhecidas, cadastradas e reconhecidas erroneamente e faces que não foram cadastradas porem foram reconhecidas. Aplicando as técnicas Eigenfaces na Tabela 1, Fisherfaces na Tabela 2 e LBPH na Tabela 3.

Tabela 1. Resultados utilizando o Eigenface.

Vídeo	Pessoas	Faces cadastradas	Identificadas corretamente	Cadastradas e reconhecidas	Cadastradas mas reconhecidas erroneamente	Não cadastradas mas reconhecidas
1	11	3	11	3	2	8
2	9	2	9	2	1	5
3	13	0	13	0	0	9

Tabela 2. Resultados utilizando o Fisherface.

Vídeo	Pessoas	Faces cadastradas	Identificadas corretamente	Cadastradas e reconhecidas	Cadastradas mas reconhecidas erroneamente	Não cadastradas mas reconhecidas
1	11	3	11	3	2	7
2	9	2	9	2	0	6
3	13	0	13	0	0	12

Tabela 3. Resultados utilizando o LBPH.

Vídeo	Pessoas	Faces cadastradas	Identificadas corretamente	Cadastradas e reconhecidas	Cadastradas mas reconhecidas erroneamente	Não cadastradas mas reconhecidas
1	11	3	11	3	3	8
2	9	2	9	2	1	5
3	13	0	13	0	0	10

Com os resultados obtidos uma análise dos erros a partir dos testes realizados nos vídeos foi possível gerar um gráfico comparando o grau de erro para cada método no ambiente testado, onde é apresentada na Figura 6. Para esse cenário o método Eigenface se sobressaiu melhor, comparados aos métodos Fisherface e LBPH. Embora seu tempo de processamento para treinamento e carregamento do arquivo de treino demostrou ser mais lenta que os outros métodos.

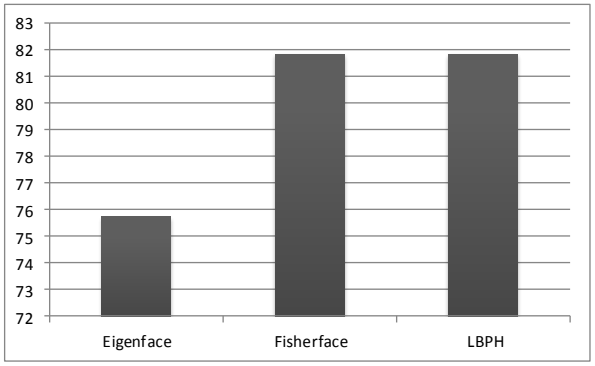

Figura 6. Erros dos métodos testados.

6 CONSIDERAÇÕES FINAIS

Baseado no sistema proposto utilizando as bibliotecas OpenCV, JavaCV e rotinas de controle da câmera Axis, o aplicativo desenvolvido bem como os algoritmos utilizados, mostrou-se com um bom desempenho e resultados satisfatórios. Dessa forma a proposta se torna uma

realidade para o reconhecimento facial nas cenas com multidões.

Apesar de o método Eigenface apresentar os melhores resultados baseados no teste realizados, seu desempenho em relação ao tempo de execução e carregamento do arquivo de treino despende muito tempo de processo, o que pode comprometer o uso em hardware de baixo desempenho e pouca memória.

Como trabalhos futuros espera-se um melhor desenvolvimento para expressões faciais, reconhecer e detectar rostos em diferentes ângulos e envelhecimento. Generalizar a classe AxisCamera e criar especializações para câmeras de outras marcas e modelos.

REFERÊNCIAS

2nd DESIGN DIGITAL. **O futuro da tecnologia do check-in – 2nd**. Disponível em: <http://www.2nd.com.br/o-futuro-da-tecnologia-do-check-in/ >. Acesso em: 09 abr. 2013.

ALBERGARIA, E. T.; SANTOS, K. C. L.; ALVIM JÚNIOR, M. S. F. Reconhecimento de faces utilizando programação genética. In: SEMINARIO DE PROJETO E ANÁLISE DE ALGORITMOS, 26 jun. 2006. Disponível em: < http://homepages.dcc.ufmg.br/~nivio/cursos/pa06/seminarios/seminario12/seminario12.pdf>. Acesso em: 24 nov. 2013.

BONSOR, Kevin. **HowStuffWorks**: o rosto. Desenvolvido por HSW International. Disponível em: < http://pessoas.hsw.uol.com.br/sistema-de-reconhecimento-facial1.htm >. Acesso em: 09 abr. 2013.

BRSCAN. **BrScan Tecnologia**. Disponível em: <http://www.brscan.com.br/servicos/brface> . Acesso em: 01 abr. 2013.

COGNIEX. **Cogniex:** excelência em reconhecimento facial. Desenvolvido por Conceptiva. Disponível em: <http://www.cogniex.com.br/index.php>. Acesso em: 01 abr. 2013.

DANTAS, C. A.; BATISTA, M. A. **Detecção, caracterização e recuperação de faces**. EnAComp. 2013. Disponível em: < http://www.aptor.com.br/enacomp2013/pdf/43.pdf>. Acesso em: 24 nov. 2013.

GUIMARÃES, T. S. **Reconhecimento de faces utilizando transformada discreta do cosseno bidimensional, análise de componentes principais bidimensional e mapas auto-organizáveis concorrentes**. 2010. 136 f. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia elétrica, Universidade Federal de Uberlândia.

OPENCV. **OPENCV - Open Source Computer Vision**. Desenvolvido por Itseez. Disponível em: <http://opencv.org/>. Acesso em: 18 mar. 2013a.

OPENCV. **Face recognition with OpenCV**. Disponível em: < http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html>. Acesso em: 18 mar. 2013b.

SHAN, S. et al. **Extend fisherface for face reocognition from a single example image per person**. Disponível em: < http://www.jdl.ac.cn/user/sgshan/pub/shan-iscas.pdf> Acesso em: 18 jun. 2014.

SINFIC. **Sinfic AS**. Disponível em: <http://www.sinfic.pt/Sinfic Web/displayconteudo.do2?numero=24923> Acesso em: 05 mar. 2013.

WAGNER, P. **Local binary patterns**. Disponível em: < http://www.bytefish.de/blog/local_ binary_patterns/> Acesso em: 20 jun. 2014.