IMPLEMENTAÇÃO DE UM ALGORITMO DE RECONHECIMENTO FACIAL USANDO EIGENFACE

Bruno Bechler Machado[1]
Magali Maria de Araújo Barroso, Miriam Lourenço Maia[2] **– Orientadora**
Gabriel Peixoto Guimarães Ubirajara e Silva[3] **– Orientador**
[1]brunoasdfgh@gmail.com; [2]mbarroso@unibh.br; mlmaia@unibh.br [3]gabriel@neocontrol.com.br
Curso de Ciência da Computação – Uni-BH (www.unibh.br)

Resumo *–Propõe-se neste artigo realizar um estudo no campo da visão computacional através da implementação do algoritmo de Eigenfaces. O objetivo dessa implementação é realizar o reconhecimento facial, desde o cadastramento de imagens até o reconhecimento e treinamento das informações na base de dados. Essa base de dados será formada por imagens capturadas por uma webcam. Será utilizada uma biblioteca open source capaz de prover ferramentas de cálculos matemáticos e filtros de imagens que formam a base do algoritmo.*
Um estudo de caso é realizado ao final utilizando o reconhecimento facial para cadastrar e identificar um usuário.

Palavras-chave *– Eigenface. PCA. OpenCV. FaceVACS. Face template. Face trackers .Algoritmo. C++. Open source. Reconhecimento facial. Visão computacional.*

Abstract *– This article intends to research about computer vision through the implementation of Eigenfaces algorithm. The purpose of this implementation is to accomplish face recognition through the process of registering images, recognizing and training the database information captured by a webcam. An open source library will be used for the provision of mathematical calculation tools and image filters which will enable the algorithm implementation. By using face recognition to register and identify a user, the article will demonstrate the latest advancements in the field of computer vision and the lessons so far learned on this subject.*

Keywords – Eigenface. PCA. OpenCV. FaceVACS. Face template. Face trackers . Algorithm. C++. Open source. Face Recognition. Computer vision.

1 INTRODUÇÃO

Nos últimos 10 anos, reconhecimento facial se tornou um campo específico dentro do campo da visão computacional [3].

Alguns softwares comerciais já realizam detecções de faces, detecção de movimento, de padrões e inclusive o reconhecimento facial. Estes softwares, possuem métodos para cadastramento, treinamento e otimização dos bancos de dados de imagens em tempo real. Mas as informações detalhadas do funcionamento da maioria deles são confidenciais.

Métodos significativamente sofisticados de detecção de faces foram desenvolvidos por empresas privadas não disponibilizando os algoritmos para a comunidade científica. Os softwares fazem pré-processamentos das imagens, selecionando as melhores para realizar a detecção. A partir da imagens selecionadas são gerados dados para treinamento de algoritmos baseados em redes neurais. Desta forma, são armazenadas informações das diferenças entre as faces cadastradas para auxiliar na detecção de faces [7].

O propósito deste artigo é detalhar o funcionamento do algoritmo de reconhecimento facial Eigenfaces criado por TURK e PENTLAND (1991) do *Massachusetts Institute of Technology*. Adicionalmente, uma análise do algoritmo FaceVACS, fornecido pela empresa Cognitec, será realizada.

Eigenfaces foi o primeiro algoritmo de reconhecimento facial utilizado com sucesso para reconhecer faces na visão computacional [6].

A dificuldade de se desenvolver um algoritmo de reconhecimento facial se dá devido à existência de uma grande variedade de fatores que influenciam na qualidade do algoritmo como, por exemplo, a iluminação, a posição da face na imagem e alterações na composição facial [9]. Barba, cabelos, e até mesmo a

tonalidade da pele são exemplos de alteração na composição da face.

Neste trabalho não foi estudada a qualidade do algoritmo em relação às diferenças de ambiente em que as imagens foram capturadas, nem sua performance computacional.

Utilizando a linguagem de programação C++, foi implementado o algoritmo de Eigenfaces em um software que realiza o cadastramento e verificação de faces. Além do algoritmo, algumas otimizações no processo de captura e cadastramento das imagens foram realizadas.

A implementação utiliza uma biblioteca desenvolvida pela Intel no ano de 2000, OpenCV, que é uma biblioteca *open source* para o desenvolvimento de aplicações na área de visão computacional.

A biblioteca OpenCV possui módulos de processamento de imagens e vídeo, estrutura de dados, álgebra linear e interface gráfica básica que permitem o desenvolvimento de aplicações de visão computacional [8]

2 ESTADO DA ARTE

Nessa seção será detalhado método PCA. Este método é utilizado pelo Eigenface para realização do reconhecimento facial. Um detalhamento do FaceVACS e Eigenfaces também é considerado aqui.

2.1 FACEVACS

FaceVACS é uma ferramenta de desenvolvimento que implementa um método de reconhecimento facial. Estas funções de reconhecimento são voltadas para o processo de cadastramento das faces, verificação em tempo real e otimizações para busca em grandes bases de dados.

A ferramenta está disponível em uma variedade grande de linguagens de programação e suporta o cadastramento dos *face templates*, que são as informações extraídas de uma face para posterior reconhecimento. O FaceVACS pode cadastrar seus *face templates* em diversos banco de dados, também suporta os diferentes sistemas operacionais e arquiteturas de processador 32 bits e 64 bits [5].

O algoritmo incorpora melhorias em imagens antes do reconhecimento, como localização dos olhos de uma face, verificação do número de dimensões da imagem, caso seja apresentado um grupo de imagens. O *face trackers* também é incorporado. Ele busca a mesma face na imagem e utilizam todas as imagens, tanto às utilizadas no reconhecimento quanto as imagens capturadas durante o cadastro, para melhorar a base de dados. Para isso técnicas de inteligência artificial são utilizadas.

Por ser um algoritmo comercial, a Cognitec não fornece informações detalhadas do funcionamento de seus algoritmos, apenas as especificações técnicas para o desenvolvimento de softwares de reconhecimento facial utilizando seu SDK.

Na busca em imagens de duas dimensões, existem alguns requerimentos mínimos para que o algoritmo consiga gerar um *face template* aceitável, são eles [4]:

- boa qualidade de imagem;
- uma face completamente visível, frontal, e com os olhos abertos e visíveis;
- uma distancia entre os dois olhos na imagem de pelo menos 32 *pixels*, sendo que 60 *pixels* são recomendáveis;
- a imagem não precisa ser colorida, mas com pelo menos 64 escalas de cinza na região da face para um contraste adequado.

Uma vez com os requisitos básicos, o algoritmo é capaz de executar a extração do *face template*. O *face template* é então armazenado com mais de 500 pontos relevantes da face. Antes da extração do *face template*, são feitos ajustes na imagem, e informações são disponibilizadas ao projetista do software para relatar a qualidade da imagem que será utilizada. Esses ajustes são [4]:

- detecção exata da posição dos olhos;
- contraste estimado da imagem;
- detecção de óculos;
- determinação de exposição à luz;
- determinação da abertura dos olhos;
- determinação da abertura da boca;
- tamanho da cabeça;
- a imagem da cabeça é rotacionada, e colocada na escala para caber no padrão especificado internamente pelo algoritmo;
- ajuste no vermelho dos olhos;
- determinação do sexo;
- determinação uniforme do cenário de fundo;
- é retirado reflexões de luz no rosto, nas lentes dos óculos e nos olhos;
- determinação étnica da pessoa;
- classificação de idade;
- ajuste de intensidade de cores na imagem.

Esta verificação é feita com bem mais exigência durante a geração do *face template* para cadastramento, garantindo a qualidade desejada da informação que será armazenada.

Para realizar o reconhecimento, o nível de exigência é menor para que o algoritmo seja capaz de realizar o reconhecimento em tempo real.

Durante a fase de reconhecimento, uma vez identificada uma face na imagem, uma porcentagem informando o quanto aquela imagem é semelhante a algum *face templates* cadastrado na base de dados é retornado. Cabe ao software que está sendo desenvolvido limitar um valor de porcentagem mínima aceita para garantir que a pessoa que será reconhecida seja ela mesma, por exemplo, a Fig. 1 foi a imagem usada na geração do *face template* para o usuário X:

Figura 1. Imagem capturada no software de cadastramento.

Posteriormente foi utilizado o software de reconhecimento, que identificou o usuário X na imagem conforme Fig. 2:

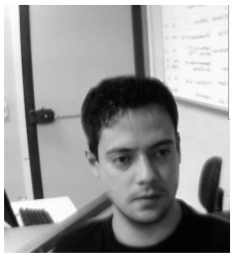

Figura 2. Imagem capturada no software de reconhecimento.

O algoritmo retornou uma similaridade de 86,50% entre a face utilizada no cadastro e a face durante o reconhecimento. O software indicou que a pessoa foi reconhecida, pois foi definido previamente nosso limite mínimo para garantir o reconhecimento é de 80% de similaridade. Qualquer imagem com valor abaixo de 80% não é válida. O software gera então um alerta.

2.2 PCA

Os criadores do Eigenface, TURK e PENTLAND (1991) popularizaram o uso do

PCA (*Principal Component Analysis*) para reconhecimento de faces [2]. Eles usaram o PCA para computar um grupo de subespaços de vetores, chamados de *Eigenfaces*. Estes são representações das imagens do banco de dados dentro de um subespaço multidimensional.

PCA é provavelmente a técnica mais utilizada para se projetar um subespaço para reconhecimento facial [3]. Os vetores básicos do PCA são computados de um grupo de imagens treinadas I. E como primeiro passo, a média da imagem em I é computada e subtraída das imagens de treinamento, gerando um grupo de amostras, conforme Eq. (1):

$$i_1, i_2, \ldots, i_n \in I - \bar{I} \tag{1}$$

Estas amostras são então vetorizadas em uma matriz X com uma coluna por imagem amostrada, conforme Eq. (2).

$$X = \begin{bmatrix} \begin{bmatrix} \vdots \\ i_1 \\ \vdots \end{bmatrix} & \cdots & \begin{bmatrix} \vdots \\ i_n \\ \vdots \end{bmatrix} \end{bmatrix} \tag{2}$$

XX**T** é então a matriz de co-variância das amostras das imagens treinadas, e o componente principal do PCA é calculado resolvendo-se conforme Eq. (3):

$$R^T \left(XX^T \right) R = \Lambda \tag{3}$$

Onde A é a matriz diagonal dos valores dos vetores *eigenvectors* e R é a matriz com os *eigenvectors*. Estes vetores são armazenados no subespaço.

Normalmente, apenas os N vetores de *eigenvectors* associados aos maiores valores definem o subespaço multidimensional.

2.3 EIGENFACE

O algoritmo Eigenface é um algoritmo que gera, como resposta, um conjunto de vetores usados na visão computacional para resolver o problema da detecção de padrões em imagens. O Eigenfaces é considerado o primeiro caso de sucesso para a tecnologia de reconhecimento facial [10] e seu fundamento básico é a utilização de vetores de distribuições probabilísticas para gerar uma informação matemática do rosto de um ser humano para sua futura identificação.

Ao ser computado, o Eigenfaces cria vetores *eigenvectors* através da técnica PCA. O PCA é responsável por treinar uma base de dados de imagens, transformando as informações visuais em vetores, que representam os pontos mais marcantes na imagem. estes vetores podem ser utilizados para detecção de objetos, padrões e faces humanas.

Qualquer aplicação que se propõe a realizar o reconhecimento facial utilizando o Eigenfaces deve utilizar uma base de dados capaz para receber os valores gerados pelo algoritmo.

O primeiro passo para a execução de um sistema utilizando o Eigenfaces é o treinamento dos vetores de reconhecimento descrito acima.

Após uma lista de imagens da mesma pessoa, é possível gerar vetores com as características mais marcantes do rosto e armazená-los em uma base de dados, associando-os a um identificador da pessoa.

Qualquer imagem computada pelo algoritmo é projetada no subespaço multidimensional. Quando duas imagens distintas são computadas, o subespaço possui uma representação como a Fig. 3:

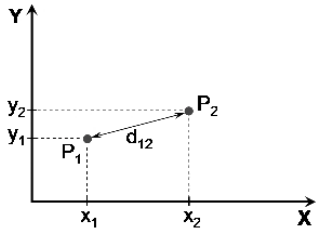

Figura 3. Gráfico representando o subespaço, ele pode possuir várias dimensões dependendo da quantidade de imagens cadastradas.

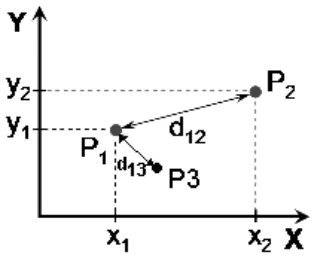

Figura 4. Gráfico representando o subespaço, agora com o ponto *P3* que é a projeção de uma imagem durante o processo de verificação da face.

Onde *P1* é a primeira imagem projetada, *P2* a segunda imagem, e *d12* é a distância no subespaço entre cada uma das imagens.

Ao cadastrar uma nova face o algoritmo projeta no subespaço multidimensional um ponto referenciando esta face. Este ponto é calculado com base nos valores dos vetores gerados pelo PCA na face. Como o Eigenface computa qualquer tipo de imagem, para o uso no reconhecimento facial, outros algoritmos de cadastro devem garantir que as imagens colocadas no subespaço possuam apenas faces.

No processo de verificação da face na base de dados, a imagem colocada para ser computada pelo Eigenfaces é calculada para serem identificados seus vetores de face. A imagem então é projetada no subespaço mas não é cadastrada. O ponto encontrado é projetado e sua distancia para pontos cadastrados calculada, como pode ser visto na Figura 4.

O ponto *P3* é a representação da projeção da nova imagem no subespaço dimensional, e *d13* é a distância entre *P3* e o ponto cadastrado no subespaço mais próximo à *P3*, no caso *P1*. O que significa que a imagem colocada para verificação tem mais chances de ser do usuário cadastrado em *P1* do que qualquer outro da base de dados.

2.4 OpenCv

OpenCV (*Open Source Computer Vision*) é uma biblioteca desenvolvida pela Intel no ano de 2000 voltada para o desenvolvimento de aplicações de tempo real no campo da visão computacional.

Originalmente desenvolvida em C++, ela é portável para todas as plataformas de sistemas operacionais.

O OpenCV possui módulos de Processamento de Imagens e Vídeo I/O, Estrutura de dados, Álgebra Linear, GUI (Interface Gráfica do Usuário), controle de mouse e teclado, além de mais de 350 algoritmos de visão computacional [8].

Os algoritmos do OpenCV, oferecem filtros de imagem, calibração de câmera, reconhecimento de objetos, análise estrutural e

operações matemáticas com matrizes de imagens de blocos de dados. Estas são algumas das funções principais disponíveis pelo OpenCV:

- *Traking* de objetos
- Detecção de borda
- Filtros de cores
- Remoção de ruídos em imagens
- Detecção de faces e padrões
- Algoritmos de treinamentos de padrões.
- Detecção de linhas, quadrados, círculos, triângulos.
- Filtros variados de desenho em imagens

OpenCV é utilizado hoje em muitos projetos *open sources* e algumas aplicações comerciais devido a sua boa performance e grande variedade de filtros e algoritmos, sendo a biblioteca *open sources* mais completa no campo da visão computacional [1].

3 DESENVOLVIMENTO

Na implementação do algoritmo, foi feito um estudo das formas mais importantes de se tratar uma imagem antes da mesma ser colocada para a execução do Eigenface, a eficiência do algoritmo está diretamente relacionada às imagens que estiverem na base de dados.

As imagens que foram capturadas possuem a resolução 320x240 *pixels*.

Assim sendo, o software deve ser capaz de capturar imagens de uma *webcam*, que já vem no formato digitalizado, e então realizar o algoritmo de detecção da face na imagem, com base na detecção de padrões do OpenCV.

Como as imagens capturadas podem possuir características muito variadas, é detectada a posição x, y da face na imagem e o tamanho da face em *pixels*.

É importante que a face mantenha uma boa simetria, e o único ponto de referência que estamos levando em consideração para esta simetria, é a posição dos olhos.

Sem um segundo momento é utilizado outro algoritmo de detecção de padrões, para identificar a posição x, y dos olhos. Com a posição dos olhos, deve ser calculado o ângulo de diferença da altura dos dois olhos, como demonstra a Fig. 5:

O ponto *C* que representa a coordenada do meio do olho esquerdo do usuário, que deveria estar no lugar do ponto *B*, para garantir uma simetria na imagem, desta forma, é calculado um ângulo alfa, através dos vértices *A*, *B* e *C* e da altura *H*, e então a imagem é rotacionada no ângulo alfa, alinhando a posição dos olhos.

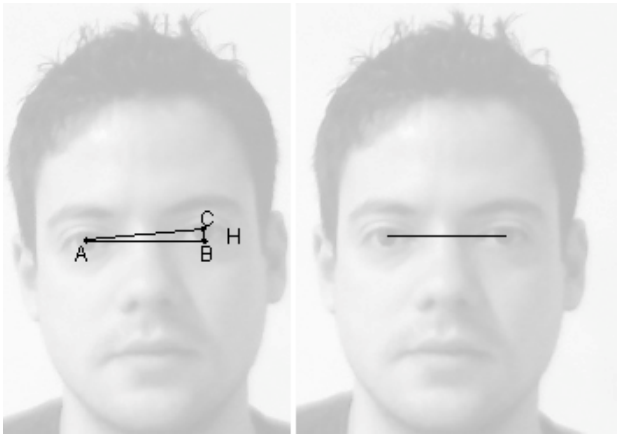

Figura 5. Imagem da face retirada pelo algoritmo e os vértices *A* e *C*, que representam o meio dos olhos identificados pelo padrão na face da esquerda. Na face da direita, foi feita uma rotação usando um ângulo alfa para alinhar a altura dos olhos.

Retira-se então a imagem da face do resto da imagem capturada. Se o tamanho da face na imagem for diferente do tamanho especificado

previamente, é feito um redimensionamento na imagem para que ela corresponda ao padrão 100x100 *pixels*.

Assim que gerada uma imagem do tamanho necessário contendo apenas a face, deve-se verificar a simetria da imagem É necessário equilibrar os tons de cinza. Desta forma, evita-se que pequenas diferenças de iluminação interfiram no processo de reconhecimento.

A próxima etapa consiste no cadastramento da imagem na base de dados. Um identificador do usuário foi criado automaticamente, Uma função que calcula o PCA foi utilizada para gerar os vetores *eigenvectors*, chamada *doPCA*, como mostra o algoritmo 1 a seguir:

Algoritmo 1 – gera os vetores e cadastra no subespaço a imagem

```
void EigenFace::doPCA(){
int i;
CvTermCriteria calcLimit;
CvSize faceImgSize;
/* seta o numero de eigenfaces que já estão cadastrados
na base */
nEigens = nTrainFaces-1;
// aloca espaço de memória para os eigenvectors
faceImgSize.width  = faceImgArr[0]->width;
faceImgSize.height = faceImgArr[0]->height;
eigenVectArr = (IplImage**)cvAlloc(sizeof(IplImage*) *
nEigens);
for(i=0; i<nEigens; i++)
eigenVectArr[i] = cvCreateImage(faceImgSize,
IPL_DEPTH_32F, 1);
// aloca o vetor de eigenvalue
eigenValMat = cvCreateMat( 1, nEigens,
CV_32FC1 );
// aloca a média da imagem
pAvgTrainImg = cvCreateImage(faceImgSize,
IPL_DEPTH_32F, 1);
// configura o algoritmo PCA
calcLimit = cvTermCriteria( CV_TERMCRIT_ITER,
nEigens, 1);
/* computa a media da imagem, eigenvalues, e
eigenvectors */
cvCalcEigenObjects(nTrainFaces, (void*)faceImgArr,
(void*)eigenVectArr,
CV_EIGOBJ_NO_CALLBACK,0,0,&calcLimit,
pAvgTrainImg,eigenValMat->data.fl);

cvNormalize(eigenValMat, eigenValMat, 1, 0, CV_L1,
0);
}
```

Este processo deverá ser executado para todas as imagens que se deseja cadastrar no banco de dados.

A imagem gerada pela função *doPCA* com os *eigenvectors*, pode ser vistas na Fig 6:

Figura 6. Imagem da face antes e depois o processo de calculo do PCA pela função *doPCA*.

Após a implementação da função *doPCA*, deve-se implementar a função que reconhece as faces chamada de *doRecognize*. Ela executa a verificação do reconhecimento facial, como o algoritmo 2 a seguir:

Algoritmo 2 – Algoritmo que identifica a imagem projetada e sua distância em relação a um dos pontos já cadastrados.

```
void EigenFace::doRecognize()     {
int i, nTestFaces  = 0;
CvMat * trainPersonNumMat = 0;
float * projectedTestFace = 0;
projectedTestFace = (float *)cvAlloc(
nEigens*sizeof(float) );
for(i=0; i<nTestFaces; i++)          {
```

```
 int iNearest, nearest, truth;
/* projeta a imagem no subespaço calculado previamente
pelo PCA */
 cvEigenDecomposite(
  faceImgArr[i],  nEigens,  eigenVectArr,  0, 0,
  pAvgTrainImg,  projectedTestFace);
/* identificador do vetor de eigenvectors da face
identificada */
  iNearest =    findNearestNeighbor(projectedTestFace);
// distância no subespaço
  truth    = personNumTruthMat->data.i[i];
// identificador cadastrado da pessoa identificada
  nearest  = trainPersonNumMat->data.i[iNearest];
}}
```

O Eigenface retorna apenas três informações, o identificador do vetor de *eigenvectors* da face reconhecida no subespaço, a distância da imagem de teste do ponto cadastrado e o identificador da face.

A função *doRecognize* é executada dez vezes, com dez imagens capturadas durante o processo. Desta forma, vamos ter dez valores de identificadores encontrados pelo algoritmo.

Esta técnica é utilizada para tentar gerar alguma porcentagem de semelhança entre a imagem capturada e as imagens cadastradas na base. Caso, por exemplo, forem testadas dez imagens e o valor dos identificadores retornados for: 1, 2, 1, 1, 1, 2, 1, 1, 4, 1, podemos dizer que no final do reconhecimento a imagem reconhecida tem 70% de chance de ser do usuário com identificador um.

4 CONCLUSÃO

A utilização do Eigenfaces para realizar o reconhecimento facial se mostrou bem eficiente na comparação de pessoas cadastradas na base de dados.

O Eigenfaces, por comparar apenas valores probabilísticos em um subespaço multidimensional, sempre retorna algum valor para seus *eigenvectors*. Isto é um problema, pois gera confusão e falsos positivos no reconhecimento de faces.

É comum o reconhecimento facial com Eigenfaces retornar positivo para faces não cadastradas na base de dados. Os softwares comerciais lidam com este problema e não geram falsos positivos com tanta freqüência.

Por ser o primeiro algoritmo de reconhecimento, as técnicas de pré-processamento das imagens evoluíram muito tentando minimizar os problemas de falsos positivos gerados pelo Eigenfaces.

Se apresentadas imagens de faces com boa resolução, iluminação, face visível e frontal, no cadastramento e na verificação, o algoritmo é executado sem apresentar erros no reconhecimento.

O OpenCV disponibiliza todas as ferramentas necessárias para o desenvolvimento do algoritmo, auxiliando na implementação do algoritmo Eigenfaces e nas técnicas de pré-processamento das imagens e detecção de padrões.

O Eigenfaces não é o melhor algoritmo de reconhecimento facial e não é capaz de competir com os algoritmos existentes hoje, como o FaceVACS. Acredito, entretanto, que com uma metodologia adequada ele possa ser evoluído a um algoritmo mais completo.

No campo da detecção de padrões, o Eigenfaces tem um melhor desempenho. Se armazenados *eigenvectors* extremamente diferentes, como por exemplo, uma face humana e uma árvore, dificilmente o Eigenfaces irá apresentar um falso positivo.

O Eigenfaces mostrou-se um bom algoritmo para ser estudado, sua implementação não é complexa e existem um grande número de publicações elucidando seu funcionamento.

5 REFERÊNCIAS

[1] BRADSKY, G. R.; PISAREVSKY, V.; BOUGUET, J. Learning OpenCV: *Computer Vision with the OpenCV Library*. Springer, Estados Unidos, 2006, [12 p.].

[2] COTTRELL, Garrison; FLEMING, Megan, *Face recognition using unsupervised feature extraction*, Dordrecht, Holanda : apresentado na Conferência Internacional de Redes Neurais, 1990.

[3] DRAPER, Bruce; BAEK, Kyungim; STEWART, Marian, BEVERIDGEL, Ross, *Recognizing Faces with PCA and ICA*, Colorado, Estados Unidos: *Computer Vision and Image Understanding*, Vol. 91, Itens 1-2 [22 p.], Julho e Agosto 2003.

[4] FaceVACS - SDK Version 8.0 Technical Specification: Manual de especificações técnicas, Dresden: Cognitec, 2009. [3 p.]. Disponível em: http://www.cognitec-systems.de/FaceVACS-SDK.19.0.html. Acesso em 20 março 2009.

[5] FaceVACS Technology - B4T8 Algorithm Performance: Performance do algoritmo FaceVACS, Dresden: Cognitec, 2009. [2 p.]. Disponível em: http://www.cognitec-systems.de/FaceVACS-SDK.19.0.html. Acesso em 20 março 2009.

[6] HEWITT, Robin, *Face Recognition With Eigenface*, SERVO Magazine, Abril 2007. Disponível em: www.servomagazine.com.

[7] MOGHADDAM, Baback; PENTLAND, Alex, *Beyond Eigenfaces: Probabilistic Matching for Face Recognition,* Nara, Japão: apresentado na conferência internacional de Reconhecimento Automático e Reconhecimento de Gestos, 1998.

[8] OPENCV REFERENCE MANUAL - *Open Source Computer Vision Library*: biblioteca de visão computacional open source. Estados Unidos, Intel Corporation, 2001 [436 p.]. Disponível em: http://developer.intel.com Acesso em: 1 abril 2009.

[9] RUIZ-DEL-SOLAR, Javier; QUINTEROS Julio, *Illumination Compensation and Normalization in Eigenspace-based Face Recognition: A comparative study of different pre-processing approaches.* 2008. Artigo científico, Universidad de Chile, Santiago, Chile.

[10] TURK, Mattew; PENTLAND, Alex, *Eigenfaces for Recognition*. Massachusetts, Estados Unidos: *Journal of Cognitive Neuroscience*, vol. 3, [6 p.]. 1991.