Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

OESLLE ALEXANDRE SOARES DE LUCENA

Detector de Faces Baseado no Pós-Processamento da Detecção de Pele

Campina Grande, Paraíba

Março de 2016

OESLLE ALEXANDRE SOARES DE LUCENA

Detector de Faces Baseado no Pós-Processamento da Detecção de Pele

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Área de Concentração: Processamento Digital de Imagens

Orientador: Prof. Dra. Luciana Ribeiro Veloso

Campina Grande, Paraíba

Março de 2016

OESLLE ALEXANDRE SOARES DE LUCENA

Detector de Faces Baseado no Pós-Processamento da Detecção de Pele

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovado em _____ /_____ /_____

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Prof. Dra. Luciana Ribeiro Veloso
Universidade Federal de Campina Grande
Orientador

Campina Grande, Paraíba
Março de 2016

Dedico este trabalho aos meus pais e todos que encontram na ciência um refúgio contra a ignorância.

Agradecimentos

Aos meus pais, Oscar e Izabel e minha irmã, Izabelle, pelo apoio incondicional para que eu sempre pudesse continuar estudando e jamais desistisse dos meus sonhos.

Aos meus amigos de graduação, Luciana Joviniano, Lucas Henrique, Lucas José, Érico Castro, Geraldo Landim, Felipe Pontes e Renata Garcia, pelo incentivo e todo apoio durante o período da graduação.

Aos meus amigos do Laboratório de Visão Computacional, Ítalo de Pontes e Tarciso Oliveira que mais que companheiros de trabalho sempre me deram suporte neste trabalho seja dando dicas ou tirando minhas dúvidas.

Ao meu grande amigo Wesley Cavalcante, por sempre me escutar e me apoiar em todas as horas.

Ao Professor Waslon Lopes, pelo apoio, amizade, compreensão e conselhos que sempre atentamente escutei.

À Professora Luciana Veloso, pela orientação deste Trabalho de Conclusão de Curso.

A todos que anonimamente fazem diferença em minha vida.

For me, it is far better to grasp the Universe
as it really is than to persist in delusion,
however satisfying and reassuring.
(Carl Sagan)

Resumo

Identificar pessoas em uma imagem tem inúmeras aplicações nas áreas de visão computacional e processamento digital de imagens, sendo os algoritmos de reconhecimento de faces humanas a técnica recorrente para esta atividade. A detecção de faces tem como objetivo determinar em uma imagem a região na qual os *pixels* representam faces humanas. Para uma maior eficácia de detecção de faces, características de informações de pele humana podem ser combinadas ao detector de faces de modo a reduzir o número de falsas detecções. Nesse contexto, o presente trabalho apresenta um Detector de Faces Baseado no Pós-processamento da Detecção de Pele(*Face Detector Based on Skin Detection - FDBSD*) que proporcionou um aumento da precisão *(Positive Predict Value - PPV)* na detecção de faces, reduzindo, assim, o números de falsas detecções para um classificador. Para isto, utilizou-se de um detector de pele baseado em histogramas de cores de *Jones and Rehg* e os detectores de faces *Haar Cascade* e *PICO*.

Palavras-chave: Detecção de Faces. Detecção de Pele. *FDBSD*. *PPV*. *Haar Cascade*. *PICO*.

Abstract

Detecting people in an image is one application of the Vision Computer and Digital Image Processing, and face detection is one of the most used algorithm to do this task. Face detection's goal is to find in an image an area where the pixels corresponds to a human face. In order to achieve a better efficiency in face detection field, human skin features can be combined with a face detector to reduce the number of false detections of the classifier. In this work, it is presented an Face Detector Based on Skin Detection (FDBSD) algorithm that increases the precison (PPV) of the face detector and reduces the number of false detections. To do that, it was used a skin detector based on coloured histograms of Jones and Regh and the face detectors of Haar Cascade and PICO.

Keywords: Face Detection. Skin Detection. *FDBSD. PPV. Haar Cascade. PICO.*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TCC	Trabalho de Conclusão de Curso
RGB	Vermelho, Verde e Azul (*Red, Green, Blue*)
ROC	*Receiver Operating Characteristic*
TP	Verdadeiro Positivo (*True Positive*)
FP	Falso Positivo (*False Positive*)
TN	Verdadeiro Negativo (*True Negative*)
FN	Falso Negativo (*False Negative*)
TPR	Taxa de Verdadeiro Positivo (*True Positive Rate*)
TNR	Taxa de Verdadeiro Negativo (*True Negative Rate*)
FPR	Taxa de Falso Positivo (*False Positive Rate*)
FNR	Taxa de Falso Negativo (*False Negative Rate*)
ACC	Acurácia (*Accuracy*)
PPV	Precisão (*Precision or Positive Predict Value*)
AUC	Área embaixo da curva (*Area Under the Curve*)
OpenCV	*Open Source Computer Vision*
gif	*Graphics Interchange Format*
jpg	*Joint Photographic Experts Group*
PICO	*Pixel Intensity Comparisons Organized in Decision Trees*
ROI	Região de Interesse (*Region of Interest*)
FC	Confiança de Face (*Face Confidence*)
FDBSD	Detector de Faces Baseado no Pós-processamento da Detecção de Pele (*Face Detector Based on Skin Detection*)
SP	Percentual de Pele (*Skin Percentage*)
SPT	Limiar para Percentual de Pele (*Skin Percentage Threshold*)

RAM	*Random-Access Memory*
IDE	*Integrated Development Environment*

Lista de símbolos

Θ	Limiar para detecção de pele

Sumário

1 Introdução

O uso de imagens digitais vem se expandindo bastante no dia a dia das pessoas. As áreas de biometria, vigilância residencial e comercial são alguns dos exemplos da aplicabilidade das imagens digitais. Para atender essa crescente demanda, muitos algoritmos de processamento digital de imagens estão sendo estudados e desenvolvidos.

Identificar pessoas em uma imagem tem inúmeras aplicações nas áreas de visão computacional e processamento digital de imagens, sendo os algoritmos de reconhecimento de faces humanas a técnica recorrente para esta atividade. Um exemplo notável é o detector de faces *Haar Cascade* desenvolvido por (VIOLA; JONES, 2004) que atualmente é o mais utilizado em diversas atividades (LIU; CHANG, 2015).

Vários métodos para detecção de faces foram desenvolvidos de modo a se obter uma detecção ótima em aplicações especifícas. Assim, a necessidade de uma comparação entre os classificadores torna-se indispensável. Em (JAIN; LEARNED-MILLER, 2010) os autores compararam os resultados de avaliação de desempenho de diversos classificadores utilizado o conjunto de imagens *FDDB* (JAIN; LEARNED-MILLER, 2010). Esses resultados podem ser vistos em (JAIN; LEARNED-MILLER, 2015).

A identificação de pele humana é outra área de estudo, que pode ser utilizada na detecção de pessoas e vários algoritmos têm sido estudados e desenvolvidos para este fim, dentre eles pode se destacar o clássico *Jones and Regh* desenvolvido por (JONES; REHG, 2002), entre outros algoritmos apresentados em (PHUNG; CHAI; BOUZERDOUM, 2003), (KAWULOK; KAWULOK; NALEPA, 2014).

Com objetivo de aumentar a eficácia da detecção de faces, características de identificação de pele podem ser combinadas com as de identificação de faces. A avaliação do uso de informações de pele humana via detector de pele é apresentada em (ANCHIT; MATHUR, 2014) promovendo bons resultados para detecção de faces.

Sob esse contexto, busca-se, num pós-processamento, analisar as informações fornecidas pelo detector de pele para aumentar a eficácia do sistema de detecção de faces. Assim, tem-se como objetivo diminuir as falsas detecções de face, aumentando a precisão do detector de faces.

Este trabalho encontra-se dividido em 5 capítulos. No Capítulo 2 encontra-se a fundamentação teórica para detecção de faces, especificando os detectores utilizados no algoritmo proposto e suas implementações em código. No Capítulo 3 é apresentado a fundamentação teórica para detecção de pele, especificando as características do detector utilizado, a fundamentação teórica da avaliação de desempenho e os experimentos de

validação para o detector de pele. Na sequência, no Capítulo 4 é descrito o algoritmo de detecção proposto no trabalho, com detalhes de sua implementação e exemplos de sua aplicabilidade. Os resultados e procedimentos da aplicação do detector de faces baseado no pós-processamento da detecção de pele (*Face Detector Based on Skin Detection (FDBSD)*) encontram-se no Capítulo 5, o qual detalha as avaliações de desempenho do detector de faces proposto e os tempos de processamento referentes a diferentes testes com o algoritmo. Além disso, há uma discussão dos resultados para os testes realizados, ressaltando o que era esperado e suas limitações. Por último, o Capítulo 6 traz as conclusões seguido das referências bibliográficas e os apêndices.

2 Detecção de Faces

A detecção de faces tem como objetivo determinar em um vídeo ou em uma imagem a região na qual os *pixels* representam faces humanas. Diferentemente da técnica usada na detecção de pele neste trabalho, a detecção de faces, em boa parte dos detectores, usa extração de caraterísticas em vez de análise *pixel-a-pixel*. Ou seja, uma janela deslizante (*sliding window*) de tamanho pré-definido passa em toda a imagem identificando as características desejáveis.

Neste trabalho foram utilizados os detectores *Haar Cascade* proposto por (VIOLA; JONES, 2004) e o *Pixel Intensity Comparisons Organized in Decision Trees (PICO)* proposto por (MARKUS et al., 2013). Esses detectores foram escolhidos devido a viabilidade de obtenção dos códigos, no caso do *PICO* os autores fornecem o código fonte, já o *Haar Cascade* encontra-se incorporado na biblioteca do *Open Source Computer Vision (OpenCV)* que foi utilizada neste trabalho. Maiores detalhes sobre a biblioteca *OpenCV* são encontrados no Apêndice A.

2.1 Detector de Faces *Haar Cascade*

Um dos métodos mais utilizados para detecção de faces é o *Haar Cascade*. Neste método, são utilizadas diferenças entre regiões retangulares vizinhas com intuito de extrair Cascatas de Características de Haar (*Haar Features*) utilizando-se de imagens integrais. E, por meio do método de treinamento *AdaBoost*, cascatas de características são construídas para formar um detector bastante robusto (PEREIRA, 2012).

2.1.1 Características de Haar

As Características de Haar são basicamente retângulos que determinam padrões de caraterísticas a serem identificadas em uma imagem, tais como: borda, linha, vizinhanças, entre outros (PAPAGEORGIOU; OREN; POGGIO, 1998). Essas características variam em sua orientação podendo identificar linhas verticais, horizontais e diagonais por exemplo.

A Figura 1 ilustra as características de Haar utilizadas no detector de faces *Haar Cascade*, que combinadas determinam uma face. Basicamente, essas características têm um tamanho pré-definido, por exemplo 24×24 *pixels* e são janelas que deslizam na imagem consistindo em uma subtração de somas de valores de *pixels* das regiões em preto e em branco. Caso esse resultado seja maior que um determinado limiar, a característica proposta por esse retângulo condiz com a região aplicada.

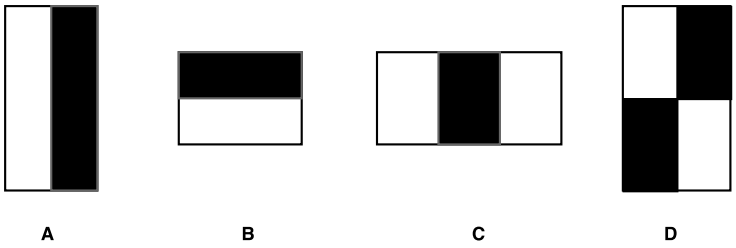


Figura 1 – Características de Haar. A e B representam regiões de borda, C de linhas verticais e D representados linhas diagonais.

2.1.2 Imagem Integral

A extração de características por janela deslizante requer um tempo de processamento maior devido ao número de operações realizadas a cada passo. De forma a otimizar o processo e evitar o cálculo repetitivo das áreas das regiões analisadas, surge a ideia da imagem integral.

A representação integral de imagens pré-computa todas as possíveis somas de valores de *pixels* antes da passagem da janela deslizante, como apresentado na Equação 2.1. Sendo $i(x, y)$ o valor do *pixel* na imagem na posição x e y e $s(x, y)$ a soma de todas os *pixels* à esquerda e acima do *pixel* na devida posição mais a soma do valor do pixel.

A Figura 2 ilustra o processo de cálculo da imagem integral.

$$s(x, y) = i(x, y) + s(x - 1, y) + s(x, y - 1) - s(x - 1, y - 1). \tag{2.1}$$

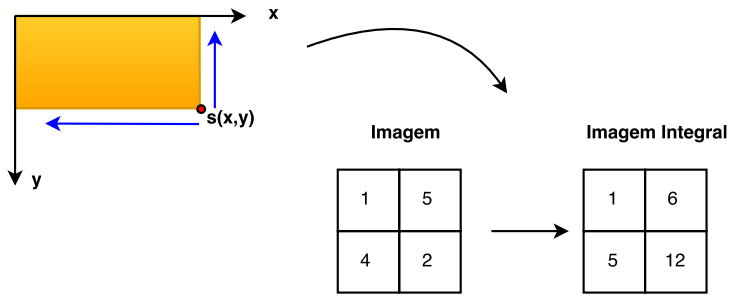


Figura 2 – Representação do Cálculo da Imagem Integral.

2.1.3 Cascata de Características de *Haar*

Um classificador único baseado em apenas uma das Características de Haar não é muito eficaz, sendo considerado como um classificador "fraco" (VIOLA; JONES, 2004). Logo, a ideia é montar uma Cascata de Características de *Haar* baseada no uso de vários classificadores "fracos", de modo que ele só passará para o próxima característica se ele obter êxito com a característica anterior, caso contrário ele é identificado como não face. A região só é considerada uma face no final da Cascata de Características de *Haar* quando passar por todos os classificadores "fracos".

A Figura 3 ilustra uma Cascata de Classificadores de Características de *Haar*.

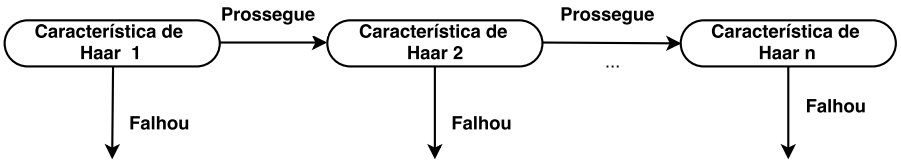


Figura 3 – Classificadores em Cascata.

2.1.4 Algoritmo *AdaBoost*

De forma a selecionar uma pequena quantidade de Características de *Haar* que identifique as faces dentre uma gama de posssíveis características, é utilizado o algoritmo AdaBoost, proposto por (FREUND; SCHAPIRE, 1997), que combina os classificadores cujas características falharam de modo a construir um bom classificador.

A cada iteração o algoritmo dedica mais esforço computacional para melhorar os resultados dos classificadores que detectaram incorretamente a maior quantidade de padrões (PEREIRA, 2012). Esse algoritmo pondera os classificadores que falharam e redimensionam o tamanho da janela para uma próxima iteração.

2.2 Detector de Faces *PICO*

O detector de faces *PICO* é uma modificação do detector de faces *Haar Cascade*. Nesse caso, a imagem é submetida a uma cascata de classificadores binários para diferentes posições e tamanhos que consistem em um conjunto de árvores de decisão com comparações de intensidades de *pixel* em seus nós internos. Assim, como no *Haar Cascade*, a região é considerada uma face quando passa por todos os estágios.

Um novo algoritmo de *boosting* é utilizado, o *GentleBoost* (FRIEDMAN; HASTIE; TIBSHIRANI, 2000), uma modificação do *AdaBoost* (MARKUS et al., 2013).

As comparações de intensidades de *pixel* nos nós internos são feitas de acordo com a Equação 2.2, em que $I(I_i)$ representa a intensidade do pixel na coordenada I_i e *bintext* representa os nós internos binários da árvore de decisão.

$$bintest(I, I_1, I_2) = \begin{cases} 0, & I(I_1) \leq I(I_2). \\ 1, & c.c. \end{cases} \tag{2.2}$$

2.2.1 Detecção e Agrupamento de Regiões de Interesse

Cada estágio do processo de detecção é representado por um conjunto de árvores de decisão, em que os primeiros níveis possuem menos árvores e a cada nível a complexidade é aumentada para refinar as detecções dos classificadores fracos assim como no *Haar Cascade*.

Várias detecções podem ser feitas na mesma região de interesse e para definir onde a face se encontra é necessário agrupar todas as detecções. Desse modo, para duas regiões que se intercedem é realizado um agrupamento entre elas caso a sobreposição for maior que 30% (MARKUS et al., 2013), ou seja:

$$\frac{D1 \cap D2}{D1 \cup D2} > 0,3. \tag{2.3}$$

2.3 Detecção de Faces no *OpenCV*

O *OpenCV* possui funções para detecção de Faces utilizando o *Haar Cascade* como algoritmo (OPENCV, 2015). Para o caso do detector de faces *PICO*, os autores disponibilizaram seu código na *internet*, o *link* para acesso está disponível em (MARKUS et al., 2013). Este último utiliza-se também das funções da biblioteca do *OpenCV*.

2.3.1 *Haar Cascade Face Detector* no *OpenCV*

O procedimento para detecção de faces deste algoritmo é bastante simples. Dado uma imagem, ele precisa inicialmente carregar um aquivo *.xml* contendo as características que definem uma face pré-processadas. Para o caso deste trabalho, utilizou-se do arquivos de características para faces frontais *lbpcascade_frontalface.xml*.

Uma vez feito isto, utiliza-se da função *CascadeClassifier::detectMultiScale* que após a inserção de alguns parâmetros retorna entre outros parâmetros um vetor de retângulos (*Rect*). Cada *Rect* é uma estrutura que contém informações do ponto de origem (x, y) onde a face começa, largura e altura da mesma. Outro fator importante a ser mencionado é a confiança de face (*Face Confidence - FC*) para a detecção daquela região de interesse (*ROI*), que é calculada por meio dos parâmetros *rejectlevels* e *levelweights*,

detalhados no próximo capítulo. O Código 2.1 detalha o processo de detecção de faces usando o *Haar Cascade*, os detalhes sobre os parâmetros utilizados são apresentados no Apêndice B.

Código 2.1 – Exemplo de Implementação do *Haar Cascade*.

```
CascadeClassifier cc;
cc.load("lbpcascade_frontalface.xml");
imageinput = imread("\people.jpg");

double scalefactor= 4.3;
int minneighbors = 3;
int flag = 0;
Size minsize = Size(25,25);
Size maxsize = Size();

cc.detectMultiScale(imageinput, objects, rejectlevels,
          levelweights, scalefactor, minneighbors,
          flag, minsize, maxsize, outputrejectlevels);

//draw circles on the detected faces
for( int i = 0; i < vrect.size(); i++ )
{
    Point center(vrect[i].x + vrect[i].width*0.5,
                 vrect[i].y + vrect[i].height*0.5);
    ellipse(imageinput, center, Size( vrect[i].width*0.5,
            vrect[i].height*0.5),0, 0, 360,
            Scalar( 255, 0, 0), 4, 8, 0);
}

imshow("detected face", imageinput);
```

As Figuras 4 e 5 ilustram o resultado de uma detecção de faces realizado utilizando o algoritmo do *Haar Cascade*. A Figura 4 representa a imagem original e a Figura 5 mostra os círculos que demarcam as faces identificadas utilizado esse algoritmo.

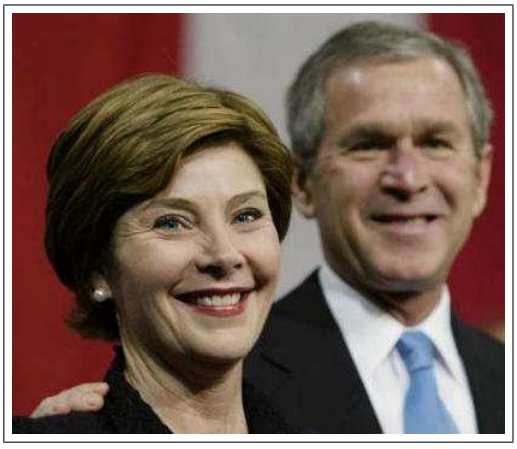


Figura 4 – Imagem Original.

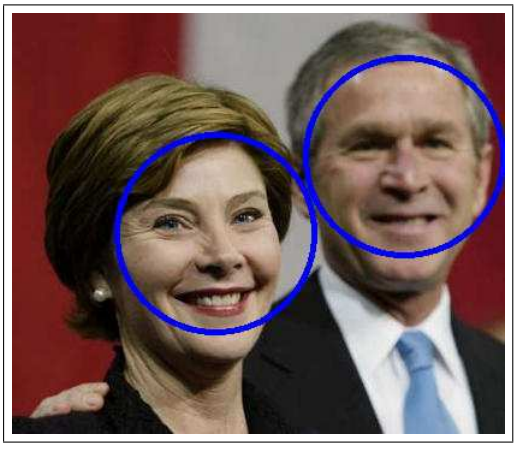


Figura 5 – Círculos demarcando as faces identificadas com o uso do detector de faces*Haar Cascade*

2.3.2 *PICO Face Detector* no *OpenCV*

Para esse algoritmo, os códigos disponíveis pelo autor em (MARKUS et al., 2013) foram reorganizados em algumas funções e classes de modo a facilitar seu uso. Basicamente, foi criada uma classe chamada de *faceDetector* e duas funções da mesma são utilizadas no processo de detecção de faces, *faceDetector::detectFaces* e um dos possíveis construtores *faceDetector::faceDetector* que inserem os parâmetros para detecção. A classe responsável por esse algoritmo encontra-se com maiores detalhes no Apêndice C.

Dada uma imagem, a função *faceDetector* retorna um parâmetro de vetor de retângulos da mesma forma que o *Haar Cascade* e, diferentemente do algoritmo anterior, este calcula as confianças de face e as retorna em um vetor junto com as informações dos retângulos. Assim, tem-se um vetor de vetores em que cada vetor interno tem as informações contidas da *ROI* no *rect* e a confiança de face para aquela região.

O Código 2.2 detalha o processo de detecção de faces usando o *PICO*.

As Figuras 6 e 7 ilustram o resultado de uma detecção de faces realizado utilizando o algoritmo do *PICO*. A Figura 6 representa a imagem original e a Figura 7 mostra os círculos que demarcam as faces identificadas utilizado esse algoritmo.

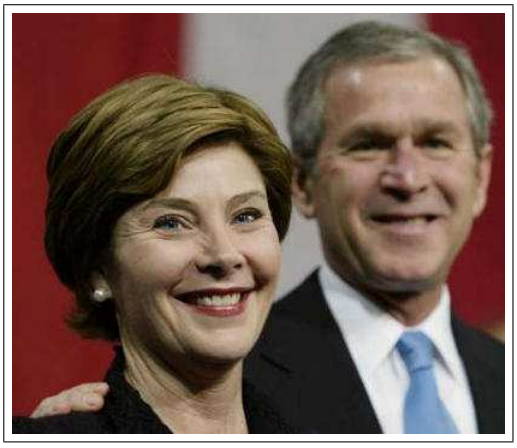


Figura 6 – Exemplo de Imagem para Aplicação de Detecção de Faces.

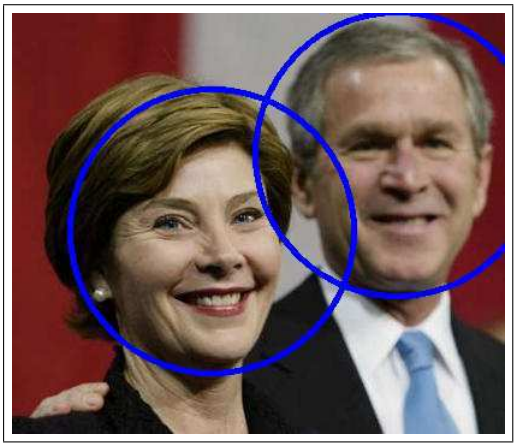


Figura 7 – Círculos demarcando as faces identificadas com o uso do detector de faces *PICO*.

Código 2.2 – Exemplo de Implementação do detector de faces *PICO*

```
Mat inputImage, outputImage, rectFrame;
vector<Rect> vRect;
vector<float>vConfidenceParam;
faceDetector fd = faceDetector();
fd.setMinSize(minSize);
inputImage = imread("img_627.jpg");
vRect = fd.detectFaces(inputImage,vConfidenceParam);

//draw circles on the detected faces
for( int i = 0; i < vRect.size(); i++ )
{
    Point center( vRect[i].x + vRect[i].width*0.5,
         vRect[i].y + vRect[i].height*0.5 );
        ellipse(inputImage, center, Size(
        vRect[i].width*0.8,
         vRect[i].height*0.8),
         0, 0, 360, Scalar( 255, 0, 0), 4, 8, 0 );
}

imshow("detected face", inputImage);
imwrite("img_627_dfp.jpg");
```

3 Detecção de Pele

A detecção de pele tem como objetivo determinar em um vídeo ou em uma imagem a região nas quais os *pixels* representam pele humana. Diversas aplicações com o seu uso podem ser encontradas, por exemplo: reconhecimento de gestos e interfaces homem máquinas (KAWULOK; KAWULOK; NALEPA, 2014).

Sob esse conceito, um detector de pele baseado no modelo descrito em (JONES; REHG, 2002) foi implementado no presente trabalho. O algoritmo se utiliza de modelos de cor de pele e não-pele e análise *pixel-a-pixel* utilizando-se de um limiar, que mede a proporção entre cor de pele e não pele para um dado pixel, de modo a identificar pele humana nas imagens. O modelo abordado é explicado com maiores detalhes nas próximas seções.

3.1 Modelos de Histograma Baseados em Cor

Neste trabalho, os modelos de histogramas utilizados são baseados em cores, obtidos por meio da criação de histogramas de cores RGB (*Red, Green, Blue*). Estes, por sua vez, possuem 256 *bins* por canal, em que o número de *bins* representa o número de divisões de intensidades de cores do *pixel* no histograma.

Para o modelo tratado, há 3 canais que representam a intensidade de cor do *pixel*. Dessa forma, o modelo em *3D* para representação do histograma baseado em cor tem 16,7 milhões de graus de liberdade (256^3) e as intensidades de cores de pixel variam entre preto ($RGB = (0, 0, 0)$) e branco ($RGB = (255, 255, 255)$).

De modo a criar um modelo probabilístico que represente a probabilidade de cada cor no histograma, é feita uma conversão para uma distribuição discreta de probabilidade usando a Equação 3.1:

$$P(cor) = \frac{H_c(cor)}{N_c} \tag{3.1}$$

em que $H_c(cor)$ representa o número de *pixels* no *bin* da devida cor e N_c representa o número total de *pixels* somados em cada *bin* do histograma. Essa representação é utilizada para caracterização de cores de pele e não pele em uma imagem e os detalhes de seu uso serão melhor descritos nas próximas seções.

3.1.1 Modelos de Histograma para Cor de Pele e Não Pele

Os modelos de histograma para cor de pele e não pele são modelos probabilísticos que se utilizam dos conceitos apresentados na Seção 3.1 e representam a probabilidade de cada cor de pele e não pele em seus respectivos histogramas.

Esses modelos são obtidos com o uso de um conjunto de imagens (*dataset*) e suas respectivas máscaras, as quais se caracterizam por rótulos manuais indicando quais *pixels* na imagem representam ou não intensidades de cor de pele.

Para construção dos modelos de histogramas de cor pele e não pele, utiliza-se do mesmo princípio abordado na Seção 3.1, com o uso das Equações 3.2 e 3.3,

$$P(cor|H_p) = \frac{H_p(cor)}{N_p} \tag{3.2}$$

e

$$P(cor|H_{np}) = \frac{H_{np}(cor)}{N_{np}} \tag{3.3}$$

em que $H_p(cor)$ representa o número de *pixels* para uma dada cor de pele no histograma de pele e $H_{np}(cor)$ representa o número de *pixels* para uma dada cor de não pele no histograma de não pele. N_p representa o número total de *pixels* do histograma de cor de pele e N_{np} representa o mesmo para o histograma de cor de não pele. Desse modo, $P(cor|H_p)$ representa a probabilidade de uma cor ser cor de pele e $P(cor|H_{np})$ representa a probabilidade de uma cor ser cor de não pele.

Segundo (KAWULOK; KAWULOK; NALEPA, 2014), uma redução no número de *bins* na construção de histogramas é eficaz e reduz o tempo computacional e o uso de histogramas com 64 *bins* promovem bons resultados. Desta forma, neste trabalho os modelos de histogramas de cor utilizados possuem 64 *bins*.

Dado histogramas de tamanho pré-definidos para de cor de pele e cor de não pele e todas suas posições inicialmente zeradas, analisa-se cada *pixel* da imagem e compara-se com seu correspondente na máscara, que é de forma prévia manualmente rotulada para cor de pele e não pele. Dessa forma, verifica-se com base no rótulo se a cor em análise é cor de pele ou de não pele. Uma vez que o *pixel* identificado representa uma cor de pele, incrementa-se o valor da cor no histograma de cor de pele, caso contrário a cor é incrementada no histograma de cor de não pele. Este processo é repetido até que todos os *pixels* de todas as imagens do *dataset* sejam analisadas. Uma representação desse processo de criação dos modelos de histogram de cor de pele e não pele para uma imagem em análise encontra-se na Figura 8.

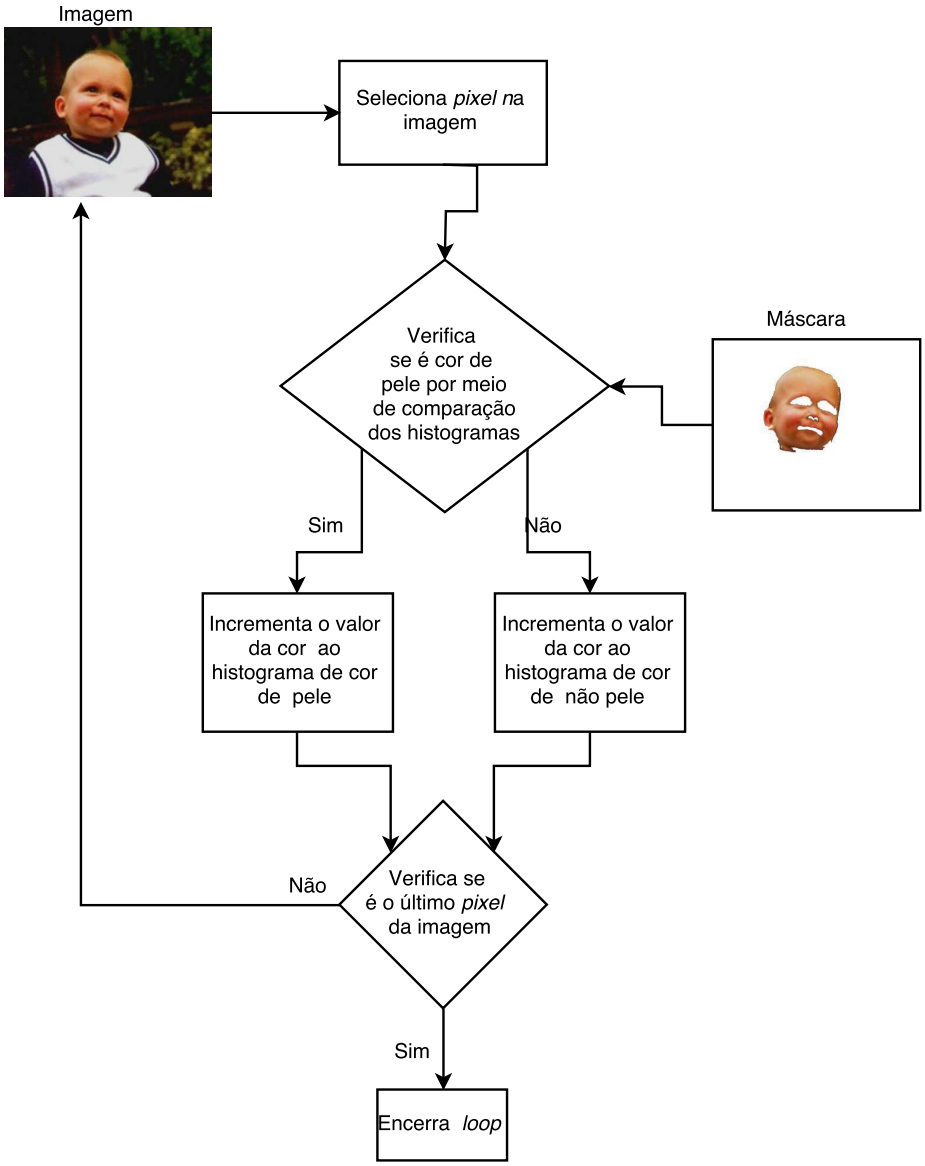


Figura 8 – Ilustração do Processo de Criação dos Histogramas de Cor de Pele e Cor de Não Pele.

3.1.2 Detecção de Pele Usando Histogramas Baseados em Cor de Pele e Não Pele

Os modelos de histograma descritos na Seção 3.1.1 são descritores que possuem informações sobre as prováveis cores que representam ou não a cor de pele. Um detector de pele pode utilizar os modelos de histogramas de pele e não pele para analisar se o *pixel* em questão é ou não pele utilizando a razão entre as probabilidade condicionais de sua cor de ser pele ou não ser maior que um determinado limiar (*threshold*).

Sob esse raciocínio, para um certo *pixel* de imagem e dado um limiar, caso a razão entre as probabilidades condicionais (pele e não pele) seja maior que uma dado limiar, este *pixel* é identificado como pele, caso contrário considera-se como não pele. A Expressão 3.4 define a análise supracitada,

$$\frac{P(cor|H_p)}{P(cor|H_{np})} \geq \Theta, \tag{3.4}$$

em que Θ é o limiar obtido da análise das detecções corretas e das falsas detecções da curva *Receiver Operating Characteristic (ROC)*.

Antes de apresentar os resultados da aplicação desse tipo de detecção faz-se necessário introduzir o conceito de análise por curva *ROC*.

3.2 Análise pela Curva *ROC Curve*

A curva ROC é uma maneira de visualização do desempenho de classificadores binários quando esses variam um limiar (*threshold*) (FAWCETT, 2005) (ERKEL; PATTYNAMA, 1998). A curva leva em consideração as informações da matriz de confusão tornando possível uma avaliação gráfica do desempenho dos classificadores.

3.2.1 Matriz de Confusão

Dado um conjunto de experimentos e um classificador binário, há quatro formas de avaliar os resultados de classificação (FAWCETT, 2005). Por exemplo, suponha uma doença rara em um grupo de pessoas, dentre as quais uma parte possui a doença e outra parte são indivíduos saudáveis. E, baseados em um diagnóstico, as pessoas podem ser classificadas por estarem doentes ou não. Associando ter a doença como positivo e não ter a doença como negativo, pode se fazer a seguinte análise: Se o diagnóstico do individuo for positivo (individuo possui a doença) e este possui a doença incrementa-se o contador de Verdadeiro Positivos (*True Positive - TP*). Caso o seu diagnóstico indique negativo (o indivíduo não possui a doença) e o indivíduo possua a doença (positivo), incrementa-se o contador de Falso Negativo (*False Negative - FP*), pois considerou-se negativo um evento que era positivo. Seguindo este raciocínio, incrementa-se o contador de Verdadeiro Negativo (*True Negative - TN*) para diagnóstico quando o indivíduo não possui a doença e seu diagnóstico é de indivíduo saudável, por outro lado incrementa-se o contador de Falso Negativo (*False Negative - FN*) para um diagnóstico de indivíduo saudável quando este está doente.

Essas avaliações de detecções são descritas em uma tabela chamada de Matriz de Confusão ilustrada na Figura 9. A Figura 10 ilustra a detecção de experimentos dado um

classificador binário os quais são avaliados de acordo com um limiar arbitrário. Este, ao ser alterado, pode variar o desempenho do detector via taxas de Sensibilidade e Especificidade.

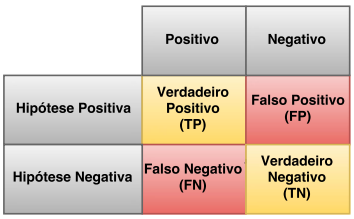


Figura 9 – Matriz de Confusão.

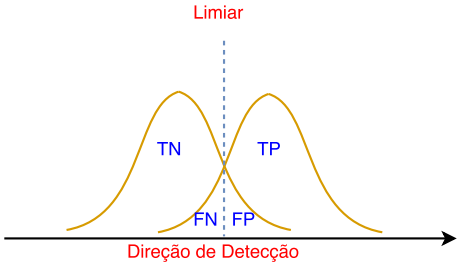


Figura 10 – Ilustração da Avaliação de Resultados para um Limiar Arbitrário.

A Sensibilidade (*Sensitivity*) ou Taxa de Verdadeiro Positivos (*True Positive Rate - TPR*), é a variação da quantidade de acertos do classificador para os que foram rotulados positivos (Verdadeiro Positivos - TP) em função do número de classificações negativas quando o rótulo era positivo (Falso Negativos - FN), ou seja quanto menor o FN maior é a sensibilidade do classificador.

A Especificidade (*Specificity*) ou Taxa de Verdadeiro Negativos (*True Negative Rate - TNR*)é a variação da quantidade de acertos do classificador para as classificações negativas (Verdadeiro Negativos - TN) em função do número de falsas detecções para as classificações positivas (Falso Positivos - FP), ou seja quanto menor o FP maior será a especificidade do classificador. A Sensibilidade e a Especificidade são descritas nas Equações 3.5 e 3.6 respectivamente.

$$TPR = \frac{TP}{TP + FN} \tag{3.5}$$

e,

$$TNR = \frac{TN}{FP + TN} \tag{3.6}$$

Existem mais duas taxas de análise que são calculadas a partir da matriz de confusão, a Taxa de Falso Negativos (*False Negative Rate - FNR*) e a Taxa de Falso Positivos (*False Positive Rate - FPR*). Estas são os complementos da Sensibilidade e da Especificidade, ou seja, $1 - Sensisiblidade$ e $1 - Especificidade$, respectivamente, e avaliam também a Especificidade e Sensibilidade, dando ênfase à análise das falsas detecções tanto positivas quanto negativas. A descrição destas taxas encontra-se nas Equações 3.7 e 3.8

respectivamente.

$$FNR = \frac{FN}{TP + FN} \tag{3.7}$$

e,

$$FPR = \frac{FP}{FP + TN} \tag{3.8}$$

3.2.1.1 Métricas de Avaliação

De forma a avaliar a qualidade de um classificador, duas métricas se destacam entre as mais usadas: Acurácia (*Accuracy* - *ACC*) e Precisão (*Precision* - *PPV*). A Acurácia é simplesmente a avaliação do número total de acertos, tanto positivos quanto negativos, sobre o número total de avaliações do classificador (ERKEL; PATTYNAMA, 1998). Ou seja, quanto mais acertos o classificador tiver maior será sua Acurácia. Por outro lado, a Precisão avalia o número de acertos positivos que o classificador teve em relação ao número de falsos acertos, logo, o classificador tem uma boa precisão quando a quantidade que ele acertou é consideravelmente maior que o número de falsas detecções para o rótulo positivo, ou seja, o numero de falso positivos.

As duas métricas são dadas por:

$$ACC = \frac{TP + TN}{TP + FN + FP + TN} \qquad PPV = \frac{TP}{TP + FP}. \tag{3.9}$$

3.2.2 Espaço *ROC*

A Curva *ROC* é uma representação gráfica obtida quando se varia os valores de um dado limiar e calcula-se as taxas de *TPR* e *FPR* para cada ponto de limiar. A curva representa uma avaliação da Sensibilidade em função de $1 - Especifidade$.

A Figura 11 exemplifica espaço *ROC*, no qual são marcados alguns pontos, de forma a se fazer uma análise pontual se seguir. Visualmente, percebe-se que um ponto é melhor que outro quando este se encontra mais acima e mais a esquerda, ou seja, possui um alto valor de *TPR* e baixo valor de *FPR*. Assim, intuitivamente pode ser afirmado que o melhor ponto encontra-se no canto superior esquerdo (coordenadas $x, y = (1, 0)$ ou TPR=1 e FPR=0) apresentando a taxa máxima de acertos e a mínima de erros, como também o pior encontra-se no canto inferior direito $x, y = (0, 1)$, apresentando o oposto. Vale salientar que qualquer ponto na curva tracejada da Figura 11 tem as mesmas taxas de detecção corretas e incorretas, ou seja $TPR = FPR$. Assim, percebe-se neste exemplo que o ponto B é melhor que os pontos A e C.

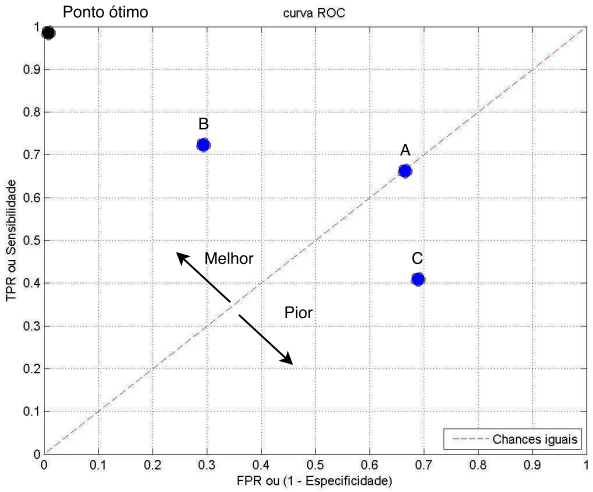


Figura 11 – Espaço *ROC*.

3.2.2.1 Ponto de Operação e Área Debaixo da Curva (*Area Under the Curve - AUC*)

A análise pontual é facilmente realizável, tendo em vista que o julgamento parte somente em cima de suas coordenadas e é possível utilizar de um classificador para identificar seu melhor ponto de operação. Entretanto, quando há mais de uma curva, ou seja, mais de um classificador, a avaliação pontual torna-se um pouco complicada, pois alguns pontos de uma curva podem ficar acima ou abaixo da outra e identificar o melhor classificador torna-se difícil.

A única forma de avaliar pontualmente, seria calcular o ponto de operação ótimo do classificador. Este é dado pelo ponto que possui menor distância Euclidiana com relação ao ponto de operação ideal que possui coordenadas $(x, y) = (1, 0)$. O ponto com menor distância Euclidiana do ponto de operação ideal é uma sugestão que pode não ser aplicável em todos os casos. Como exemplo, este pode ter taxas de TPR muito baixas para uma determinada aplicação sendo necessário a escolha de outro ponto de operação.

Contudo, a identificação do melhor classificador não é resolvida sob esta análise, pois mesmo que os pontos de operação estejam em coordenadas diferentes, o classificador considerado ruim via análise pontual pode ter melhor desempenho.

O conceito de área abaixo da curva (AUC) surge para permitir avaliar qual classificador é melhor com base na área de abaixo sua curva ROC (FAWCETT, 2005). Dado o ponto de operação ideal, a curva ótima para um classificador deve conter este ponto e manter a taxa de TPR igual a $1,0$ para todos os limiares, totalizando uma área unitária, ou seja uma taxa de 100% de acerto em toda a curva. Logo, pode-se perceber que a curva que tiver o valor mais próximo do máximo ($AUC = 1,0$) é considerada a de melhor desempenho para um classificador. Na Figura 12 são apresentadas duas curvas

ROC cujas *AUC* são 0,64 e 0,62.

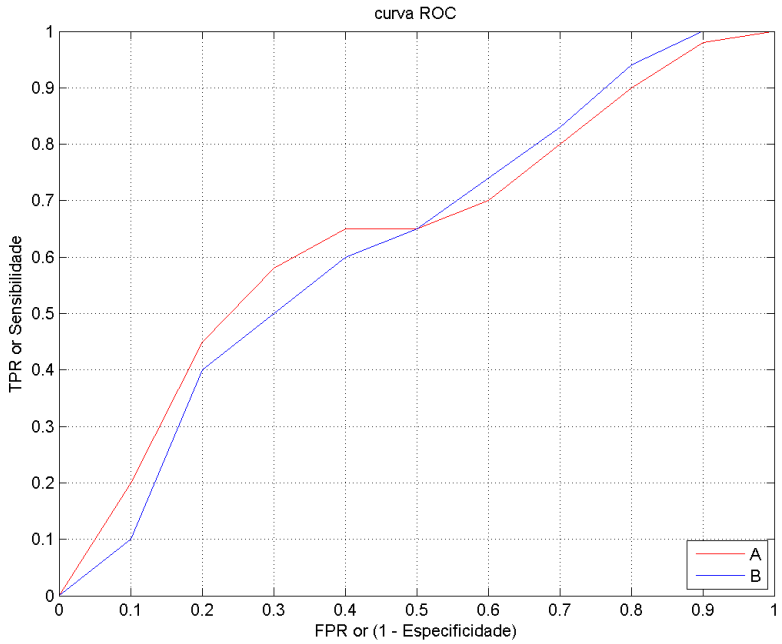


Figura 12 – Curvas *ROC* para Classificadores Diferentes.

Visualmente, é muito difícil certificar qual curva é melhor pois há pontos em que uma está acima da outra e há pontos em que está abaixo. Dessa forma, calculando-se a *AUC* para ambas, tem-se como resultado que a curva *A* possui uma $AUC = 0{,}64$ e a curva *B* possui uma $AUC = 0{,}62$. Sob este parâmetro pode-se concluir que a curva *A* representa um melhor classificador que a curva *B*.

3.3 Avaliação de Desempenho do Detector de Pele

O experimento para avaliar o detector de pele deste trabalho consistiu nas fases de treinamento e validação. A fase de treinamento consiste na criação dos modelos de histogramas, descritos na Seção 3.1.1, utilizando as imagens contidas no conjunto de treinamento. Por outro lado, a fase de teste consiste em gerar a curva *ROC* descrita na Seção 3.2, de modo a avaliar o desempenho variando o limiar Θ.

Foram realizados quatro experimentos de validação utilizando conjuntos de dados (*datasets*) diferentes, as informações dos experimentos são descritas na Tabela 1.

***dataset* utilizados**	**Nº de imagens de teste**	**Nº de imagens de treino**
Jones and Rehg[1]	6839	6839
ECU[2]	2000	2000
Mix-JonesTr-ECUTs[3]	6839	2000
Mix-ECUTr-JonesTs[4]	2000	6839

Tabela 1 – Dados dos *datasets* usados nos Experimentos de Validação

Devido a problemas de compatibilidade com a biblioteca *Open Source Computer Vision (OpenCV)*, que não lê imagens de tipo *.gif*, o conjunto de *Jones and Rehg* foi alterado devido a algumas imagens corrompidas após a conversão de *.gif* para *.jpg*.

Para geração da curva *ROC* o Θ da Equação 3.4 foi variado no intervalo $[0, 10]$ com passo de $0,1$, totalizando 101 iterações, com intuito de gerar um gráfico com maior representatividade visando extrair o Θ ótimo. A Figura 13 ilustra os resultados para os quatro experimentos da Tabela 1.

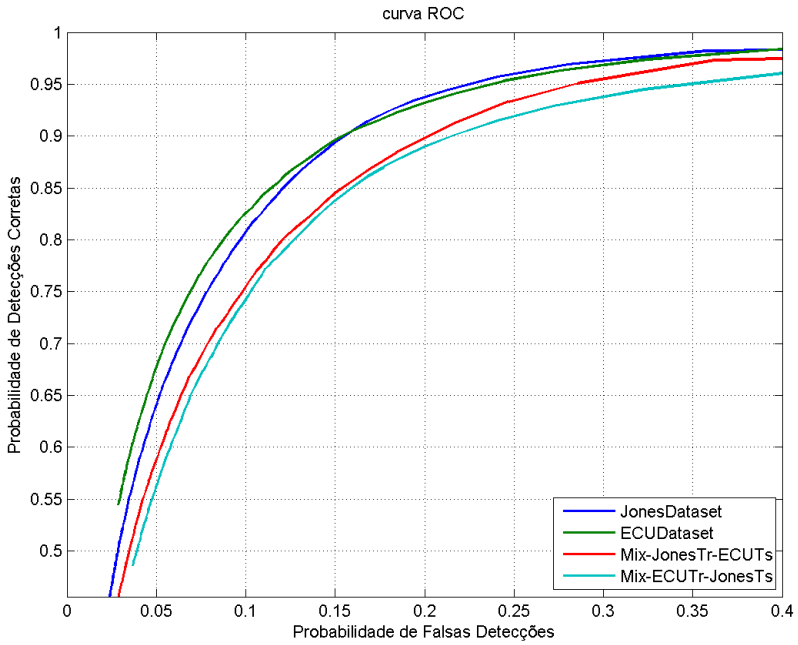


Figura 13 – Curvas *ROC* Usando Diferentes *datasets*.

A Tabela 2 mostra os valores obtidos pelas curvas para sua *AUC*, os valores *TPR*, *FPR* e Θ para o ponto de operação ótimo da curva.

[1] 4511 imagens que não contêm pele humana e 2328 imagens que contêm pele humana
[2] 4000 imagens e 4000 máscaras contendo faces humanas dividido em 2 pastas
[3] *Jones and Rehg* para fase de treino e *ECU* para fase de teste
[4] *ECU* para fase de treino e *Jones and Rehg* para fase de teste

Nome do *dataset*	**AUC**	**TPR**	**FPR**	**Θ**
Jones and Rehg	0,9287	89,4%	15%	0,9
ECU	0,9289	88,1%	13,7%	1,3
Mix-JonesTr-ECUTs	0,9165	85,6%	15,9%	0,8
Mix-ECUTr-JonesTs	0,8978	86,1%	16,8%	0,9

Tabela 2 – Resultados Obtidos das curvas *ROC* nos Experimentos Para o Ponto Ótimo de Operação.

Como dito na Seção 3.2, o ponto ótimo de operação calculado pela menor distância do ponto ótimo teórico, em que $(x, y) = (1, 0)$, é uma sugestão de uso. Ao se variar o Θ ao longo da cuva *ROC*, pontos de operação com melhores taxas podem ser encontrados a depender da aplicabilidade do detector. Ou seja, o ponto ótimo calculado pela distância euclidiana pode fornecer uma taxa de *TPR* baixa, por exemplo. Sob este raciocínio, avaliou-se outros valores para o limiar extraídos das curvas *ROC* dos experimentos com o conjunto de imagens *Jones and Rehg* e *ECU*, com intuito de selecionar taxa de acerto maiores que 90%. A combinação dos *datasets* promoveu resultados inferiores aos testes realizados sem combinação, como visto na Tabela 2. Desse modo extraiu-se das curvas *ROC* apenas os valores de limiar para os experimentos sem combinação de *datasets*. Estes resultados encontram-se na Tabela 3.

Nome do *dataset*	**TPR**	**FPR**	**Θ**
Jones and Rehg	94,4%	21,2%	0,4
ECU	95,3%	24,4%	0,4

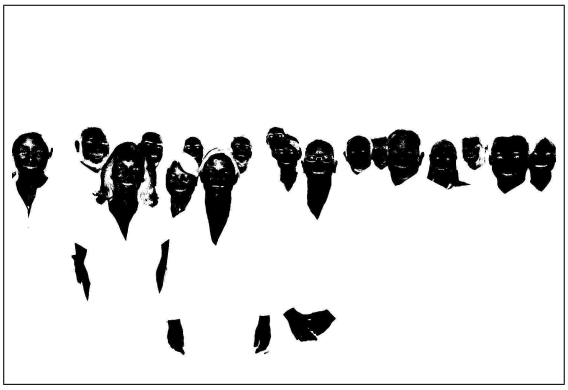
Tabela 3 – Resultados Obtidos das curvas *ROC* Escolhidos para taxa de acerto maiores que 90%

O resultados apresentados nas Tabelas 2 e 3 mostram taxas de acertos (TPR) e taxas de erro (FPR) ótimas e resultam da validação do detector de pele implementado. Visualmente fica mais fácil perceber os bons resultados do classificador. Logo, a Figura 14 mostra exemplos da aplicação do detector de pele em uma imagem com várias pessoas.

Ao variar o valor de Θ e do *dataset* para as detecções de pele a seguir, percebe-se que há variações nas áreas identificadas. Por exemplo, o cabelo da segunda mulher na primeira fila posicionada a direita da imagem original é identificado como pele nas Figuras 14b e 14d. O detector de pele, por ser baseado apenas em cores, verifica que a probabilidade da cor do cabelo de ser pele é alta, inferindo, assim, em sua detecção como pele. Todavia, as taxas de acertos permanecem altas e as falsas detecções desse tipo geram erros aceitáveis para validação do algoritmo.

(a) Imagem Original.

(b) *Jones and Rehg dataset* $\Theta = 0,4$.



(c) *Jones and Rehg dataset* $\Theta = 0,9$.

(d) *ECU dataset* $\Theta = 0,4$.

(e) *ECU dataset* $\Theta = 1,3$.

Figura 14 – Exemplo de Aplicação do Detector de Pele para o *Jones and Rehg dataset* e o *ECU dataset* para Diferentes Valores Θ.

Como apresentado neste capítulo, o detector de pele foi validado utilizando-se de conjunto de imagens diferentes e para diferentes valores de limiares Θ. Este algoritmo será utilizado em conjunto com os detectores de face, apresentados no capítulo 2, para a construção do algoritmo de detecção de faces baseado no pós processamento da detecção de pele. Os detalhes dessa combinação são apresentados nos próximos capítulos.

4 Detector de Faces Baseado no Pós-Processamento da Detecção de Pele - FDBSD

Buscando uma maior eficácia na detecção de face, vários métodos foram estudados, desde o clássico *Haar Cascade* a variações do mesmo, como o *PICO* apresentados no capítulo anterior entre tantos outros descritos em (PEREIRA, 2012). Diversas caracteristicas podem ser analisadas visando melhorar a detecção de faces, como por exemplo, textura descrito em (PEREIRA, 2012) e informações sobre pele como feito em (ANCHIT; MATHUR, 2014) em que a detecção de face é feita após a segmentação de pele.

Sob essa perspectiva, é proposto neste trabalho um algoritmo de pós-processamento da detecção de pele após a detecção de faces realizada no intuito de aumentar a precisão (PPV) do classificador. Desse modo, foram usados os detectores de face *PICO* e *Haar Cascade* apresentados no Capítulo 2 e o detector de pele baseado em histogramas de cor apresentado no Capítulo 3.

4.1 Algoritmo Proposto

O Detector de Faces baseado no Pós-Processamento da Detecção de Pele (*Face Detector Based in Skin Detection* - FDBSD) consiste em eliminar os quadros detectados como face que apresentam um percentual de pele (*Skin Percentage* - SP) menor que um determinado limiar (*Skin Percentage Threshold* - SPT). Isto é feito de modo a diminuir a taxa de Falso Positivos (FP), ou seja, eliminar o número de falsas detecções. Como consequência, há um aumento na taxa de precisão (PPV) do detector deixando mais preciso para uma confiança de face (*FC*) utilizada.

Dada uma imagem contendo faces humanas e com intensidades de pixel representadas no formato *RGB* e uma vez carregados os histogramas de cor de pele e não pele no programa, o detector de faces, seja o *PICO* ou o *Haar Cascade*, é aplicado e um vetor de retângulos (*Rect*), como também um vetor de confiança das respectivas faces são retornados. Em seguida, para cada *ROI* identifica-se as faces detectadas via as informações dos retângulos, em seguida realiza-se a detecção de pele. Feito isto, é calculado a porcentagem de pele para cada imagem segmentada via Equação 4.1.

$$SP = \frac{número\ de\ pixels\ identificados\ como\ pele}{número\ total\ de\ pixels}. \tag{4.1}$$

Por fim, compara-se o SP com um SPT pré-definido mantendo as imagens $SP \geq SPT$ e eliminando as que não satisfazem a condição. A Figura 15 ilustra o fluxograma de funcionamento do algoritmo.

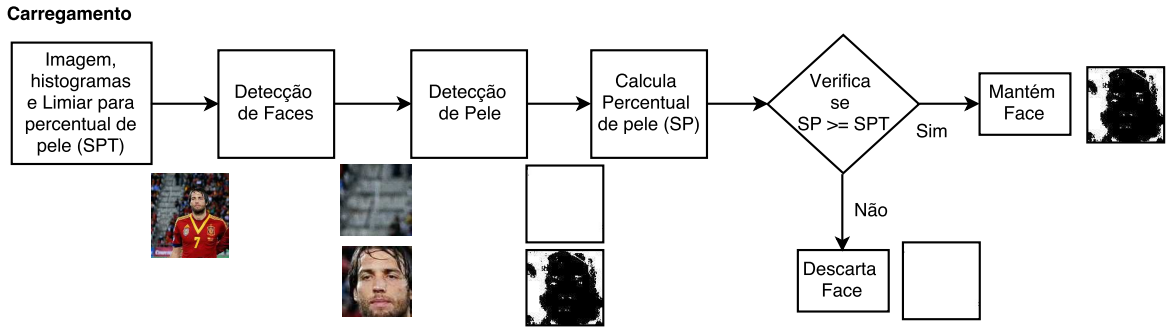


Figura 15 – Fluxograma de Funcionamento do Detector de Faces Baseado no Pós-Processamento da Detecção de Pele.

O Algoritmo 1 detalha o funcionamento do *FDBSC* reiterando a descrição supracitada.

Algoritmo 1 Algoritmo *FDBSD* proposto

Entradas:

- Histogramas de cor de pele e não pele
- Uma imagem de tamanho qualquer representada em cores RGB
- Valor para o limiar de detecção de pele SPT

Saídas:

- Conjunto de vetores de imagens de tamanho N contendo as possíveis faces que foram aprovados no teste de restrição de percentual de pele
- Conjunto de vetores com as confianças de pele FC de tamanho N das possíveis faces aprovadas pós restrição

Inicialização:

- Carregamento dos Histogramas
- Inserção de um valor para SPT desejado
- Aplica-se detector de faces
- Inicializa-se contador $i = 0$

Iteração: Para $i < N$ **faça**

- Aplica-se detecção de pele na imagem i
- Calcula-se percentual de pele SP
- **Se** $SP \geq SPT$
  - Mantém imagem i no vetor
- **Senão**
  - Remove imagem i do vetor
- Atualiza i

A Figura 16 ilustra o processo de detecção para uma imagem e os valores de SP para um $SPT = 20\%$ na implementação encontram-se na Tabela 4. Percebe-se que as imagens a e h não satisfazem o critério imposto pelo limiar de percentual de pele, que é de 20%, e são removidas do vetor de faces identificadas. As demais são iguais e superam esse limiar imposto permanecendo no vetor de faces identificadas.

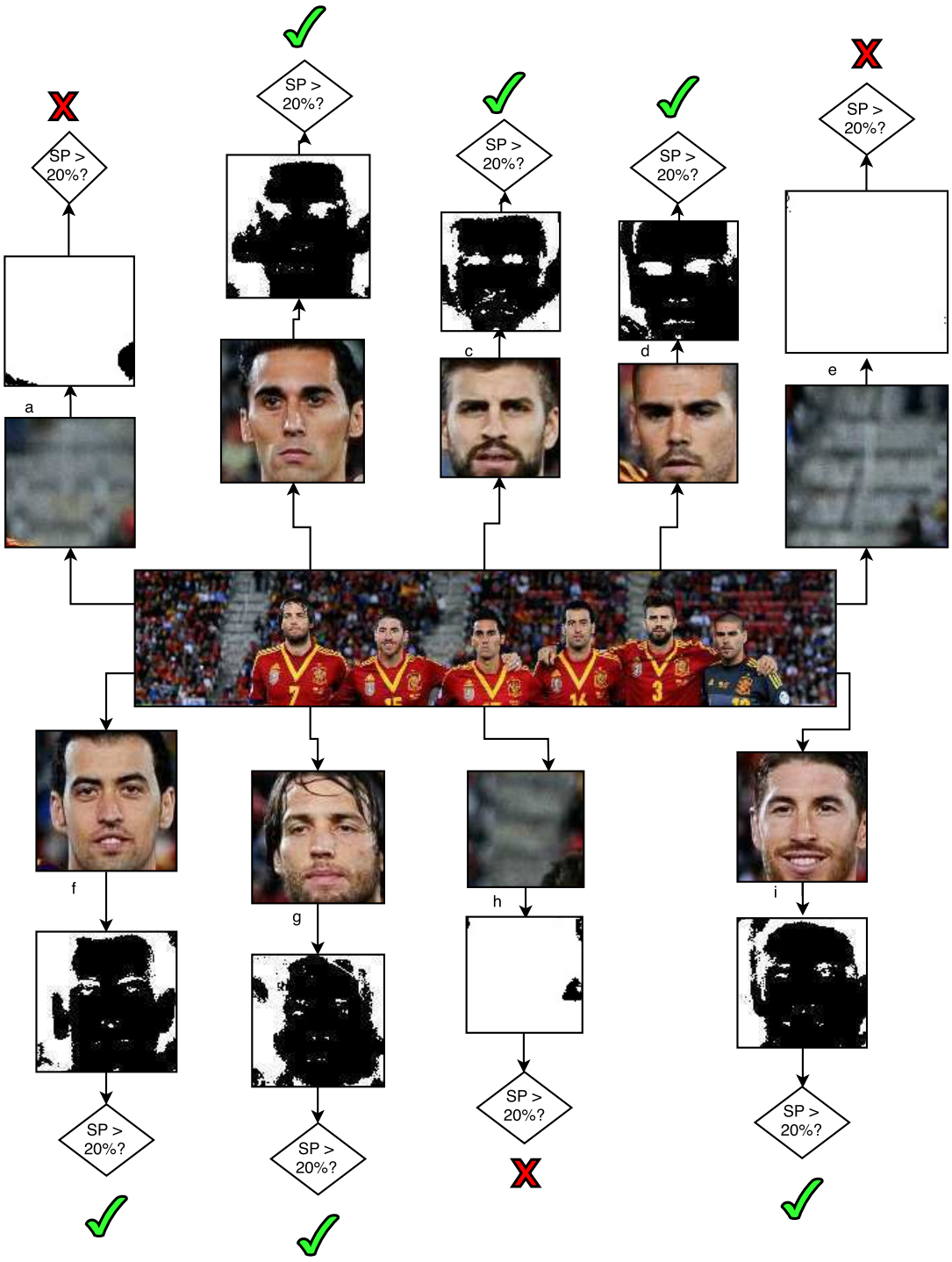


Figura 16 – Funcionamento do Detector de Faces Baseado no Pós-Processamento da Detecção de Pele.

Imagem	**Percentual de Pele - SP**	**Limiar de pele - SPT**	**Resultado**
a	3,38%	20%	Removida
b	60,75%	20%	Mantida
c	68,14%	20%	Mantida
d	83,55%	20%	Mantida
e	1,53%	20%	Removida
f	64,15%	20%	Mantida
g	65,20%	20%	Mantida
h	3,54%	20%	Removida
i	70,50%	20%	Mantida

Tabela 4 – Resultados do Algoritmo *FDBSD* Obtidos Para Imagem da Figura 16 Usando o *PICO* como Detector de Faces e o *Jones dataset* para Detecção de pele

5 Procedimentos e Resultados

De modo a avaliar o desempenho do *FDBSD*, é necessário uma validação de seu algoritmo. Isto é feito, submetendo o detector a um conjunto de imagens que possui faces previamente reconhecidas, ou seja, avaliando seu desempenho de classificação via curva *ROC* e comparando algumas de suas taxas, como por exemplo a taxa de *PPV*, antes e depois da restrição de informação de pele (JAIN; LEARNED-MILLER, 2010).

Sob esta consideração, rotinas foram criadas para a avaliação do desempenho do algoritmo *FDBSD* utilizando os detectores de face do Capítulo 2 no conjunto de imagens *FDDB* (*Face Detection Data Set and Benchmark*) descrito em (JAIN; LEARNED-MILLER, 2010).

Esse conjunto de imagens se caracteriza por imagens retiradas da *internet* entre os anos de 2002 e 2003, contendo fotos de pessoas em diferentes posições e resoluções. O *dataset* contém 2845 imagens em cores *RGB* e em tons de cinza com 5171 faces reconhecidas. A especificação das faces encontra-se em arquivos denominados de *annotations.txt*, contendo especificações para regiões de interesse do tipo retângulo ou elipse, sendo em formato de 5-tupla ou 6-tupla respectivamente. Para o caso do formato 5-tupla encontram-se especificados as coordenadas (x, y) onde a face começa, a largura e altura do retângulo e a confiança de face (FC) associada. Para o formato 6-tupla tem-se os raios maior e menor da elipse, o ângulo que o raio maior faz com a horizontal, as coordenadas (x, y) da origem da elipse e a confiança de face (FC) associada.

Além das imagens e seus respectivos aquivos de *annotations*, o *FDDB dataset* fornece os códigos para gerar as curvas *ROC* para comparação de resultados com outros algoritmos que estão disponíveis em (JAIN; LEARNED-MILLER, 2015). A rotina utilizada para gerar a curva *ROC* é baseada na variação da confiança de face em um intervalo de zero até um número infinitivamente grande de confiança de modo a praticamente zerar as taxas de acerto e de falso positivos. Isso acontece, porque usualmente para detecção de faces a curva é representada como $TPR \times FP$, e este formato é utilizado para avaliação dos detectores de face (JAIN; LEARNED-MILLER, 2010).

5.1 Metodologia Utilizada

Antes de detalhar a abordagem utilizada, é importante salientar que o detector de pele usado no *FDBSD* é baseado em histogramas de cor e para imagens representadas em tons de cinza, existentes no conjunto *FDDB*, a detecção de pele não pode ser aplicada. De fato, não é possível aplicar o algoritmo em imagens em tons de cinza, devido ao fato

de que o detector de pele é baseado em cor de pele e não pele. Ou seja, seu treinamento leva em consideração os tons de vermelho, verde e azul que compõem a intensidade de cor de pele, logo este algoritmo falharia para imagens em tons de cinza. Assim, antes de aplicar a detecção de faces proposta ao conjunto foi necessário retirar todas as imagens representadas em tons de cinza. Logo, para os procedimentos e resultados adotados neste trabalho utilizou-se o conjunto de Imagens *FDDB modificado* contendo agora 2793 imagens com 5060 faces identificadas.

A abordagem utilizada para validação baseou-se na aplicação do algoritmo *FDBSD* no conjunto de imagens *FDDB* de modo a gerar a curva *ROC* e avaliar seu desempenho. Os experimentos foram realizados variando o uso do detector de faces entre o *Haar Cascade* e *PICO* para o algoritmo proposto.

Para obter um resultado mais representativo com relação à variabilidade de cores nos histogramas, os parâmetros do detector de pele foram alterados. Inicialmente utilizou-se o *Jones and Rehg dataset* para treinar o detector de pele e realizou-se experimentos utilizando diferentes valores para o limiar Θ em busca de uma detecção ótima para os pontos de operação do Capítulo 3, que são $0,4$ e $0,9$. Feito isto, realizou-se experimentos com o *ECU dataset* para treinar o detector de pele e repetiu-se o procedimento, sendo desta vez os valores de Θ iguais à $0,4$ e $1,3$.

Uma vez realizada a detecção, faz-se necessário salvar as informações para cada face detectada em arquivos tipo texto de modo a comparar com as informações das localizações de faces cedidas pelo conjunto de imagens. Adotou-se nas rotinas o formato de retângulo, como explicado no Capítulo 2 e portanto os resultados obtidos foram comparados com os dos arquivos *annotations.txt* em formato de 5-tupla.

Nos testes realizados, foram utilizados vários valores para o limiar de percentual de pele *SPT*, os quais foram adotados 0%, 5%, 10%, 15%, 20% e 40%, sendo o de 0% apenas o detector de faces puro sem informação de pele, ou seja, sem o *FDBSD*. Porcentagens maiores não foram utilizadas, pois experimentalmente viu-se que a quantidade de acertos para taxas superiores a 40% diminui muito excluindo muitas faces identificadas pelo detector. Isso deve-se ao fato que para o conjunto de imagens adotado, 40% consegue identificar bem as faces nela contidas e o uso de taxas superiores acarreta em diminuição redução da precisão ao invés de aumento dela.

Para execução das rotinas e realização dos testes utilizou-se um *laptop* da marca *Acer* com processador *Intel i7–3632QM* de 2.2 GHz, 8 GB de memória *RAM* com sistema operacional de 64 *bits* e *Windows 10 Home*. Todas as rotinas foram implementadas na Linguagem *C++*, com exceção do *PICO* que teve parte do código implementado em linguagem *C*, utilizando-se das funções da biblioteca *OpenCV*, particularmente a versão 2.4.9. Além disso, a *IDE (Integrated Development Environment)* utilizada foi o *Visual Studio 2012*.

5.2 Resultados

Nesta Seção, são apresentadas as curvas *ROC* para os testes realizados, assim como as tabelas comparativas com resultados obtidos e os tempos de processamento dos algoritmos implementados.

As curvas *ROC* para os testes realizados encontram-se na Seção 5.2.1. Para uma comparação numérica das taxas provenientes da curva *ROC*, a Seção 5.2.2 traz os resultados expostos em tabelas para todos os testes realizados. Por último, a Seção 5.2.3 traz os resultados para o tempo de processamento da aplicação do *FDBSD* em comparação com a aplicação dos detectores de faces isoladamente.

5.2.1 Curvas *ROC*

As curvas *ROC* representam o comportamento do classificador ao longo de vários valores de confiança de face representando as taxas de *TPR* em função do número de *FP*, alterando o detector de faces utilizado, o valor do *SPT* e o valor de Θ.

As Figuras 17 e 18 ilustram as curvas *ROC* obtidas para o *FDBSD* utilizando o detector de faces *Haar Cascade* sendo o primeiro resultado com *Jones and Rehg dataset* e o segundo com o *ECU dataset*, utilizando seus devidos valores para Θ. Por outro lado, as Figuras 19 e 20 ilustram as curvas obtidas para o *FDBSD* utilizando o detector de faces *PICO* sendo respectivamente o primeiro resultado com *Jones and Rehg dataset* e o segundo com o *ECU dataset*, utilizando seus devidos valores para Θ.

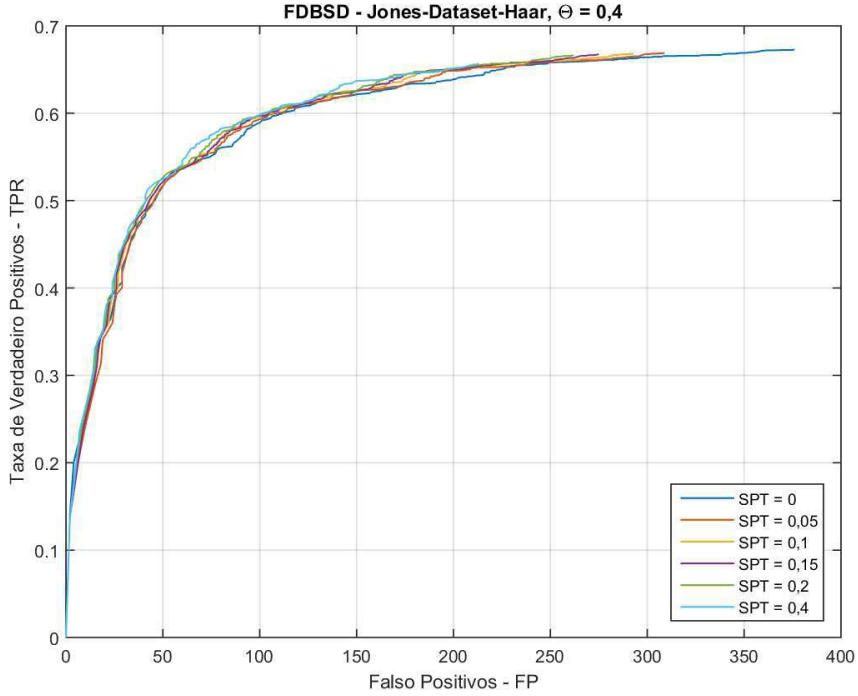
As Figuras 17a e 17b ilustram as curvas *ROC* para os experimentos utilizando o detector de faces *Haar Cascade*, o conjunto de imagens *Jones e Regh* para treino dos modelos de histogramas e valores Θ iguais as a $0,4$ e $0,9$, respectivamente. Percebe-se que as curvas apresentam um comportamento similar ao detector de faces aplicado isoladamente, sem utilização do *FDBSD*, e, quando comparadas entre si, variam pouco o comportamento quanto a mudança do Θ. Vê-se que o número de *FP* é reduzido com o aumento *SPT*, todavia a visualização desse fato é penalizada devido a proximidade entre as curvas.

As Figuras 18a e 18b ilustram as curvas *ROC* para os experimentos utilizando o detector de faces *Haar Cascade*, o conjunto de imagens *ECU* para treino dos modelos de histogramas e valores Θ iguais as a $0,4$ e $1,3$, respectivamente. Percebe-se, mias uma vez, que as curvas apresentam um comportamento similar ao detector de faces aplicado isoladamente, sem utilização do *FDBSD*, e também variam o comportamento quanto a mudança do *dataset* e Θ. Novamente, o número de *FP* é reduzido com o aumento *SPT* e a visualização ainda é penalizada devido a proximidade entre as curvas.

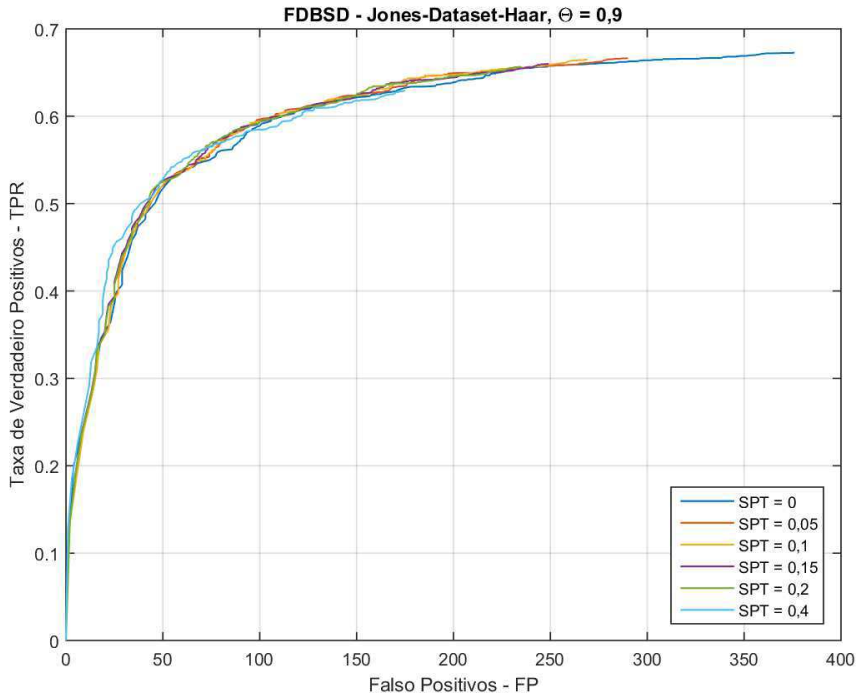
As Figuras 19a e 19b ilustram as curvas *ROC* para os experimentos utilizando o detector de faces *PICO*, o conjunto de imagens *Jones e Regh* para treino dos modelos de

histogramas e valores Θ iguais as a $0,4$ e $0,9$, respectivamente. Novamente, percebe-se que as curvas apresentam um comportamento similar ao detector de faces aplicado isoladamente, sem utilização do *FDBSD*, e variam o comportamento quanto a mudança do detector de faces, ao se utilizar dos mesmos parâmetros das Figura 17a e 17b, respectivamente. Mais uma vez, o número de *FP* é reduzido com o aumento *SPT* e a visualização é penalizada devido a proximidade entre as curvas.

As Figuras 20a e 20b ilustram as curvas *ROC* para os experimentos utilizando o detector de faces *PICO*, o conjunto de imagens *ECU* para treino dos modelos de histogramas e valores Θ iguais as a $0,4$ e $1,3$, respectivamente. Novamente, percebe-se que as curvas apresentam um comportamento similar ao detector de faces aplicado isoladamente, sem utilização do *FDBSD*, e variam o comportamento quanto a mudança do detector de faces, ao se utilizar dos mesmos parâmetros das Figura 18a e 18b, respectivamente. Novamente, o número de *FP* é reduzido com o aumento *SPT* e a visualização é penalizada devido a proximidade entre as curvas.

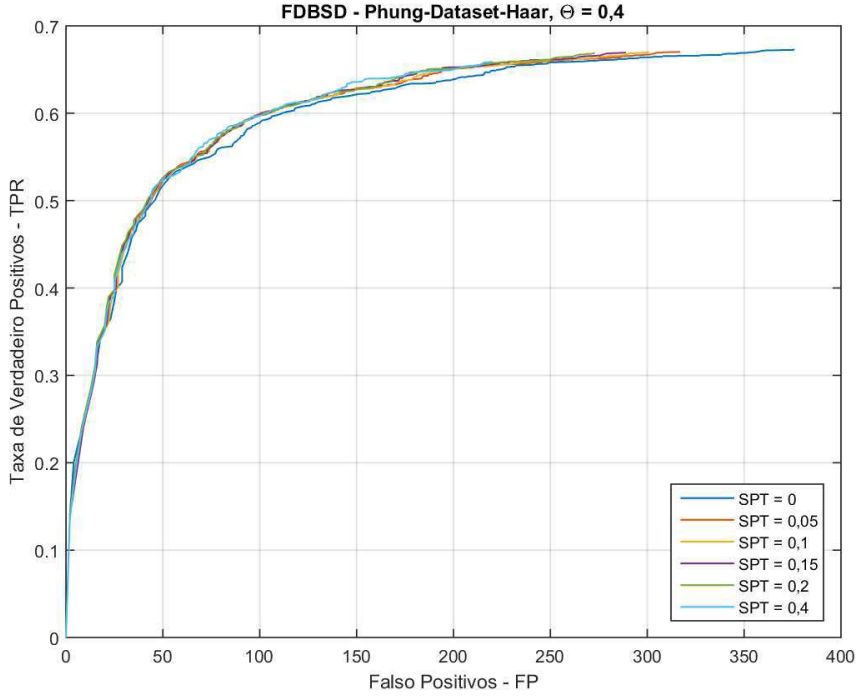


(a) *FDBSD* Usando *Jones and Rehg dataset* para Detecção de Pele com $\Theta = 0,4$ e *Haar Cascade* para Detecção de Faces.

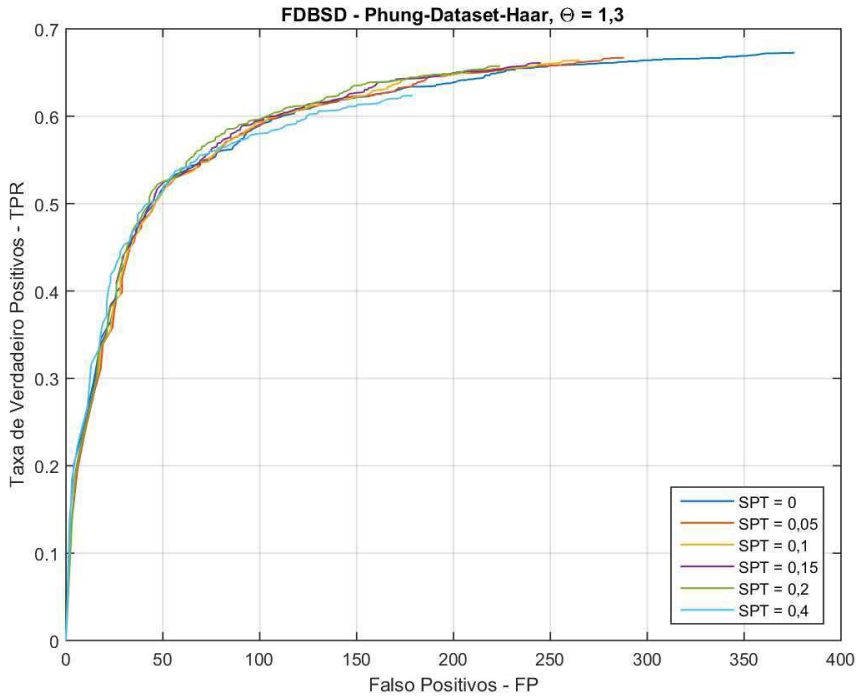


(b) *FDBSD* Usando *Jones and Rehg dataset* para Detecção de Pele com $\Theta = 0,9$ e *Haar Cascade* para Detecção de Faces.

Figura 17 – Resultados do *FDBSD* Utilizando-se do *Haar Cascade* e *Jones and Rehg dataset*

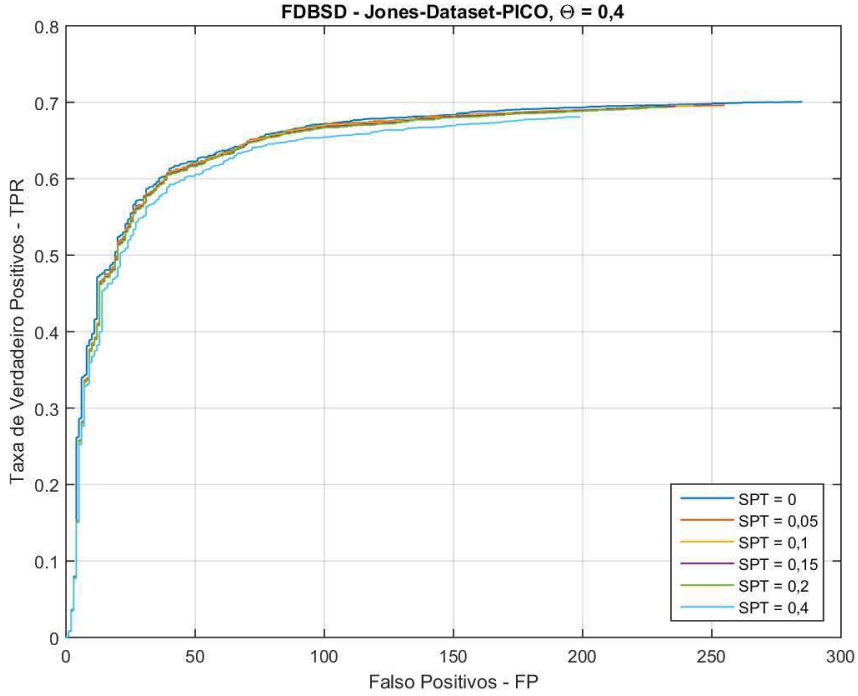


(a) *FDBSD* Usando *ECU dataset* para Detecção de Pele com $\Theta = 0,4$ e *Haar Cascade* para Detecção de Faces.

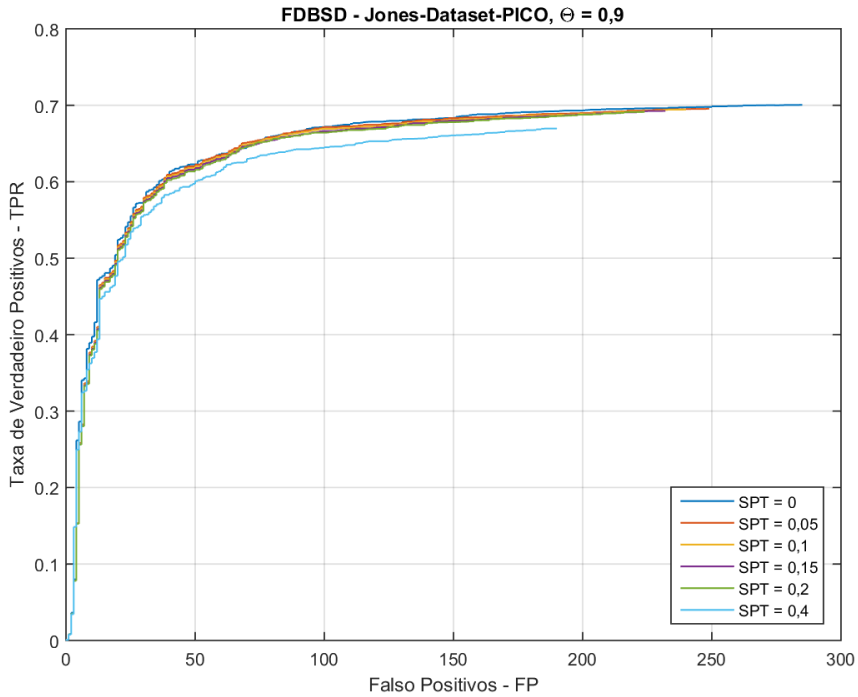


(b) *FDBSD* Usando *ECU dataset* para Detecção de Pele com $\Theta = 1,3$ e *Haar Cascade* para Detecção de Faces.

Figura 18 – Resultados do *FDBSD* Utilizando-se do *Haar Cascade* e *ECU dataset*.

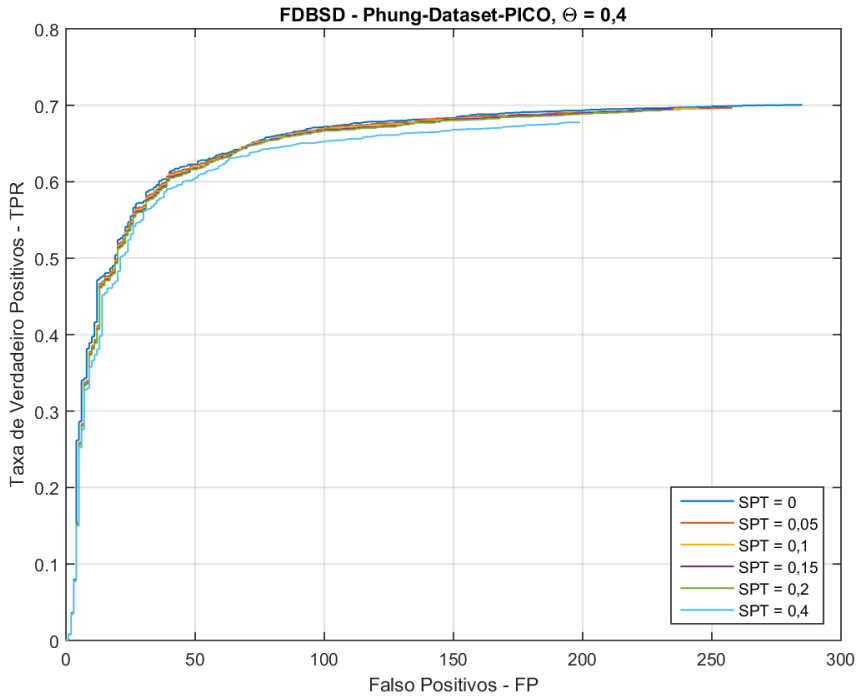


(a) *FDBSD* Usando *Jones and Rehg dataset* para Detecção de Pele com $\Theta = 0,4$ e *PICO* para Detecção de Faces.

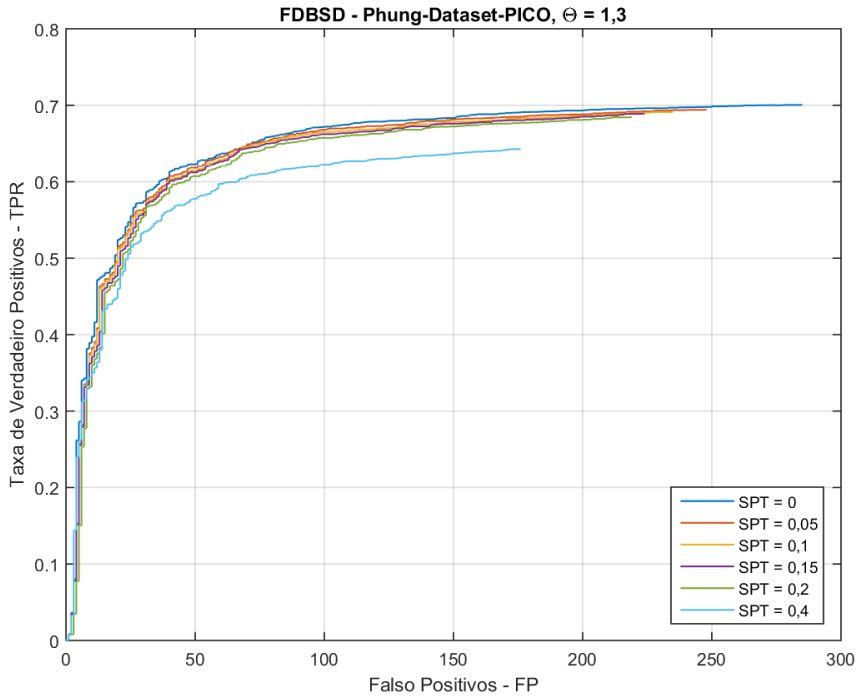


(b) *FDBSD* Usando *Jones and Rehg dataset* para Detecção de Pele com $\Theta = 0,9$ e *PICO* para Detecção de Faces.

Figura 19 – Resultados do *FDBSD* Utilizando-se do *PICO* e *Jones and Rehg dataset*.



(a) *FDBSD* Usando *ECU dataset* para Detecção de Pele com $\Theta = 0,4$ e *PICO* para Detecção de Faces.



(b) *FDBSD* Usando *ECU dataset* para Detecção de Pele com $\Theta = 1,3$ e *PICO* para Detecção de Faces.

Figura 20 – Resultados do *FDBSD* Utilizando-se do *PICO* e *ECU dataset*.

Percebe-se que as curvas alteram seus formatos ao variar o detector de faces, o *SPT* e Θ. Quando há a utilização do *FDBSD* o número de *FP* é reduzido, como também,

há uma diminuição na taxa de *TPR*, contudo os valores de *TPR* ficam muito próximos da curva sem a utilização do *FDBSD* e visualmente fica difícil qual o melhor classificador em cada teste. A discussão destes resultados é vista com maiores detalhes na Seção 5.3.

5.2.2 Tabelas

Os parâmetros avaliados são para o menor valor de confiança da curva, ou seja, o ponto com maior *TPR* do gráfico, os quais são: a taxa de acerto (*TPR*) do detector, o número de faces detectadas (*TP*), o número de faces detectadas incorretamente (*FP*), o valor de confiança de face (*FC*) associado, o limiar de pele utilizado (Θ) e o limiar de percentual de pele (*SPT*) utilizado no teste.

A variação do número de *FP* e dos valores de *PPV* encontram-se em negrito em todas as tabelas.

5.2.2.1 Resultados Comparativos do *FDBSD* Utilizando o *Haar Cascade*

As Tabelas 5 a 8 apresentam os resultados da aplicação do *FDBSD* utilizando o *Haar Cascade* como detector de faces, variando o *dataset* de treino utilizado e os valores para Θ.

As Tabelas 5, 6, 7 e 8 apresentam os resultados retirados das curvas *ROC* usando o conjunto de imagens *Jones e Regh* e $\Theta = 0,4$, o conjunto de imagens *Jones e Regh* e $\Theta = 0,9$, o conjunto de imagens *ECU* e $\Theta = 0,4$ e o conjunto de imagens *ECU* e $\Theta = 1,3$, respectivamente.

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	Θ	**SPT**
Haar	67,25%	3403	**376**	**90,05%**	51,46	0,4	-
FDBSD	66,88%	3384	**309**	**91,63%**	51,46	0,4	5%
FDBSD	66,80%	3380	**293**	**92,02%**	51,46	0,4	10%
FDBSD	66,72%	3376	**275**	**92,47%**	51,46	0,4	15%
FDBSD	66,62%	3371	**262**	**92,79%**	51,46	0,4	20%
FDBSD	65,67%	3323	**213**	**93,97%**	109,175	0,4	40%

Tabela 5 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *Haar Cascade* e o *Jones dataset* para $\Theta = 0,4$

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	**Θ**	**SPT**
Haar	67,25%	3403	**376**	**90,05%**	51,46	0,9	-
FDBSD	66,88%	3372	**290**	**92,08%**	51,46	0,9	5%
FDBSD	66,50%	3365	**269**	**93,06%**	109,175	0,9	10%
FDBSD	66,03%	3341	**249**	**93,06%**	109,175	0,9	15%
FDBSD	65,69%	3324	**235**	**93,40%**	109,175	0,9	20%
FDBSD	62,92%	3184	**175**	**94,79%**	109,175	0,9	40%

Tabela 6 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *Haar Cascade* e o *Jones dataset* para $\Theta = 0,9$

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	**Θ**	**SPT**
Haar	67,25%	3403	**376**	**90,05%**	51,46	0,4	-
FDBSD	67,02%	3391	**317**	**91,45%**	51,46	0,4	5%
FDBSD	66,95%	3388	**301**	**91,84%**	51,46	0,4	10%
FDBSD	66,94%	3387	**289**	**92,14%**	51,46	0,4	15%
FDBSD	66,86%	3383	**273**	**92,53%**	51,46	0,4	20%
FDBSD	65,91%	3335	**220**	**93,81%**	109,175	0,4	40%

Tabela 7 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *Haar Cascade* e o *ECU dataset* para $\Theta = 0,4$

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	**Θ**	**SPT**
Haar	67,25%	3403	**376**	**90,05%**	51,46	1,3	-
FDBSD	66,70%	3375	**288**	**92,14%**	51,46	1,3	5%
FDBSD	66,44%	3362	**265**	**92,69%**	109,175	1,3	10%
FDBSD	66,13%	3346	**245**	**93,18%**	109,175	1,3	15%
FDBSD	65,77%	3328	**224**	**93,69%**	109,175	1,3	20%
FDBSD	62,37%	3156	**179**	**94,63%**	109,175	1,3	40%

Tabela 8 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *Haar Cascade* e o *ECU dataset* para $\Theta = 1,3$

5.2.2.2 Resultados Comparativos do *FDBSD* Utilizando o *PICO*

As Tabelas 9 a 12 apresentam os resultados da aplicação do *FDBSD* utilizando o *PICO* como detector de faces, variando o *dataset* de treino utilizado e os valores para Θ.

As Tabelas 9, 10, 11 e 12 apresentam os resultados retirados das curvas *ROC* usando o conjunto de imagens *Jones e Regh* e $\Theta = 0,4$, o conjunto de imagens *Jones e Regh* e $\Theta = 0,9$, o conjunto de imagens *ECU* e $\Theta = 0,4$ e o conjunto de imagens *ECU* e $\Theta = 1,3$, respectivamente.

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	**Θ**	**SPT**
PICO	70%	3545	**285**	**92,6%**	5,00211	0,4	-
FDBSD	69,6%	3522	**255**	**93,2%**	5,00211	0,4	5%
FDBSD	69,6%	3521	**243**	**93,5%**	5,00211	0,4	10%
FDBSD	69,5%	3515	**236**	**93,7%**	5,00211	0,4	15%
FDBSD	69,3%	3509	**233**	**93,8%**	5,00211	0,4	20%
FDBSD	68,1%	3447	**199**	**94,5%**	5,00211	0,4	40%

Tabela 9 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *PICO* e o *Jones dataset* para $\Theta = 0,4$

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	**Θ**	**SPT**
PICO	70%	3545	**285**	**92,6%**	5,00211	0,9	-
FDBSD	69,5%	3519	**249**	**93,4%**	5,00211	0,9	5%
FDBSD	69,4%	3514	**240**	**93,6%**	5,00211	0,9	10%
FDBSD	69,3%	3505	**232**	**93,8%**	5,00211	0.9	15%
FDBSD	69,1%	3498	**224**	**94,0%**	5,00211	0,9	20%
FDBSD	66,9%	3389	**190**	**94,7%**	5,00211	0,9	40%

Tabela 10 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *PICO* e o *Jones dataset* para $\Theta = 0,9$

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	**Θ**	**SPT**
PICO	70%	3545	**285**	**92,6%**	5,00211	0,4	-
FDBSD	69,7%	3525	**258**	**93,2%**	5,00211	0,4	5%
FDBSD	69,6%	3522	**245**	**93,5%**	5,00211	0,4	10%
FDBSD	69,5%	3516	**235**	**93,7%**	5,00211	0,4	15%
FDBSD	69,3%	3508	**231**	**93,8%**	5,00211	0,4	20%
FDBSD	67,8%	3431	**199**	**94,5%**	5,00211	0,4	40%

Tabela 11 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *PICO* e o *ECU dataset* para $\Theta = 0,4$

Nome do Detector	**TPR**	**TP**	**FP**	**PPV**	**FC**	**Θ**	**SPT**
PICO	70%	3545	**285**	**92,6%**	5,00211	1,3	-
FDBSD	69,4%	3513	**248**	**93,4%**	5,00211	1,3	5%
FDBSD	69,1%	3498	**235**	**93,7%**	5,00211	1,3	10%
FDBSD	68,9%	3486	**224**	**94,0%**	5,00211	1,3	15%
FDBSD	68,4%	3463	**219**	**94,1%**	5,00211	1,3	20%
FDBSD	64,3%	3253	**176**	**94,9%**	5,00211	1,3	40%

Tabela 12 – Resultados Obtidos Para a menor Confiança de Face a qual há Detecção Usando o *PICO* e o *ECU dataset* para $\Theta = 1,3$

5.2.3 Tempos de Processamento

De forma a avaliar o tempo de processamento, ao aplicar o algoritmo proposto do *FDBSD* comparando com os detectores de face isoladamente, utilizou-se as imagens da Figura 21 que possuem tamanhos diferentes e quantidade de faces variadas.

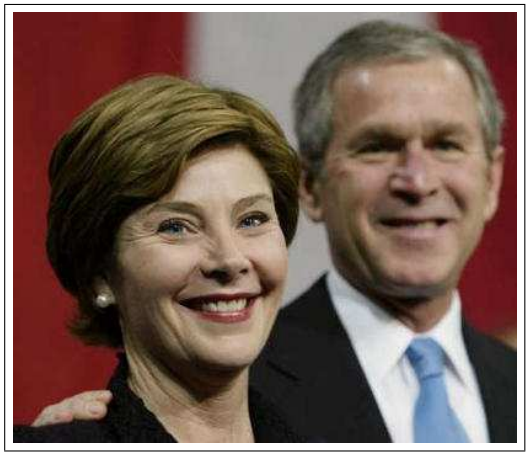
Para isto, realizou-se testes aplicando os detectores de face *Haar Cascade* e *PICO* medindo o tempo de processamento para cada imagem. Em seguida, aplicou-se o *FDBSD* utilizando os detectores de face já mencionados e mediu-se seu tempo de processamento também. Os resultados obtidos encontram-se na Tabela 13.

Imagem Usada	**Tamanho da Imagem** (*pixels*)	**Tempo dos algoritmos(s)**			
		Haar Cascade	***PICO***	***FDBSD - Haar***	***FDBSD - PICO***
img_627.jpg	410×350	0,004507	0,030097	0,011413	0,035743
phelps.jpg	594×396	0,0074427	0,05331	0,011123	0,063487
people.jpg	1698×1131	0,066657	0,3659	0,13737	0,40667

Tabela 13 – Comparação de tempo de processamento para detecção de paces usando o algoritmo *FDBSD* e usando apenas o *Haar Cascade* e o *PICO* isoladamente para as imagens da Figura 21

(a) Imagem *people.jpg*.

(b) Imagem *phelps.jpg*.



(c) Imagem *img_627.jpg*.

Figura 21 – Imagens utilizadas para avaliação do tempo de processamento na detecção de faces.

5.3 Discussão dos Resultados

Ao analisar as curvas *ROC* da Seção 5.2.1, das Figuras 17 a 20, percebe-se que a aplicação do *FDBSD* reduziu o número de *FP* dos classificadores. E, ao aumentar a restrição do *SPT*, o número de falso positivos fica cada vez menor. Todavia, fica bem difícil identificar qual o melhor classificador só olhando para as curvas *ROC*, como também analisar suas taxas de *TPR* devido a sobreposição das curvas dos classificadores ao longo dos pontos de operação, os quais são as confianças de face (*FC*).

Com auxílio das Tabelas da Seção 5.2.2, consegue-se perceber de forma mais evidente a variação do *PPV* e do *FP*. Por exemplo, a Tabela 5 mostra que o número de *FP* reduziu de 376 com $SPT = 0\%$ para 213 com $SPT = 40\%$ e a precisão aumentou de $90,05\%$ para $93,75\%$, para as mesmas taxas, com *FDBSD* usando o *Haar Cascade* como detector de faces e $\Theta = 0.4$.

Percebe-se também que as curvas são sensíveis a mudança do *dataset* utilizado para detecção de pele e seus valores para Θ adotados em ambos detectores de faces utilizados. Isto se dá pelo fato que na fase de treino dos modelos de histogramas do detector de pele o uso de *datasets* diferentes produzirem histogramas de cor de pele e não pele com representatividades de cores distintas, ocasionando, assim, em diferentes pontos de operação para o detector de pele, consequentemente alterando os percentuais de pele calculados para uma dada imagem. Ou seja, o limiar Θ adotado pode excluir um quadro contendo uma face para um modelo de histogramas adotado ao se aplicar o *FDBSD* e, ao variar o *dataset* de treino gerando outro modelo de histogramas, o quadro permanecer inalterado.

Os próprios detetores de faces também influem nos resultados. O *Haar Cascade* é um classificador menos robusto que o *PICO* e os resultados da aplicação do algoritmo tornam-se mais perceptíveis nele. Por exemplo, ele é o que possui maior redução absoluta dos números de falsos positivos nos teste apresentados na Tabela 6 quando comparado com qualquer outro teste usando o *PICO*. Isto se dá poqur que este detector erra mais por não ser tão robusto, logo o algoritmo apresentado é mais eficaz para a redução de falso positivos nele do que no *PICO*.

As Tabelas da Seção 5.2.2 trazem os resultados numéricos provenientes dos testes realizados. Para isto, adotou-se o menor valor de confiança de face, pois este produz a maior taxa de acertos facilitando assim a comparação entre classificadores. Devido a sensibilidade comentada anteriormente e também devido ao aumento do *SPT*, os testes com o *Haar Cascade* acabaram não tendo o mesmo valor de confiança mínima para todos os classificadores por conta da exclusão de algumas possíveis faces durante a detecção. Dessa forma, a menor *FC* para alguns classificadores acabou sendo diferente, todavia isto não muda o resultado da avaliação, visto que estes foram os que possuíram as maiores

taxas de *TPR*.

Os resultados numéricos reiteram o que foi constatado visualmente. Há a redução do número de falso positivos dos classificadores ao utilizar-se do *FDBSD* e ficando mais evidente ao variar os valores de *SPT*. Estes ainda vão além, mostrando que a precisão *PPV* do classificador é melhorada com a aplicação do algoritmo proposto ocorrendo independentemente do detector de faces utilizado.

Quanto mais preciso o classificador fica, ou seja, quanto mais ele garante que os quadros detectados são faces, este acaba deixando de acertar. Pois, a medida que o *TPR* diminui, o *PPV* aumenta para diferentes valores da restrição de *SPT* como apresentados nas tabelas. Logo, para algumas aplicações as quais se deseja taxa de acertos elevadas isto seria uma limitação para o uso do *FDBSD*.

Algumas vezes o quadro detectado contém face(s) porém há muito mais informação de não pele do que de pele o que provoca o descarte desta *ROI* pelo *FDBSD*. Uma vez ocorrido o descarte a taxa de acerto do classificador diminui, pois considerou-se não-face(s) a região que continha face(s). E, devido a representatividade das faces do conjunto *FDDB*, a situação descrita foi frequente, ocorrendo que ao mesmo tempo em que a precisão aumentou a taxa de acertos do detector foi penalizada.

O *FDBSD* não implica em um grande esforço computacional. Prova disto é que para a imagem *people.jpg* com tamanho de $1698x1131$ *pixels* e um número relativamente grande de faces como mostrado na Figura 21a, seu tempo de processamento variou de $0,04077$ segundos com relação ao detector de faces *PICO* e de $0,070713$ segundos com relação ao uso do *Haar Cascade*. O que mesmo sendo considerável para aplicação do *Haar Cascade* isoladamente ainda é pequeno para o processamento de uma imagem. Como também, o uso do *FDBSD* com o *Haar Cascade* foi de aproximadamente quatro vezes menor que seu uso com o *PICO* garantindo seu desempenho mais rápido.

6 Conclusão

O algoritmo proposto *FDBSD* teve como objetivo aumentar a eficácia na detecção de faces utilizando-se de características de identificação de pele. A ideia consistiu em aumentar a precisão do classificador com o uso do pós-processamento da detecção de pele. Nos testes realizados, as Tabelas 5 a 12 confirmam o aumento do PPV e a redução da taxa de falso positivos. Todavia, o custo de uma precisão mais elevada penalizou a taxa de acertos limitando assim seus uso para aplicações que requerem altas taxas de acerto.

Durante os testes, percebeu-se a sensibilidade do algoritmo a modificação de alguns parâmetros. O conjunto de imagens utilizadas para treinamento na detecção de pele e o valor de Θ alteraram os resultados, para um mesmo detector de face utilizado, elevando ou reduzindo as taxas de TPR e PPV quando comparadas com o detector de faces sem a informação de pele. Entretanto, a precisão dos classificadores aumentou, foi incrementada com a aplicação do algoritmo.

A base de imagens do *FDDB* contribuiu para redução da TPR dos classificadores em teste, uma vez que imagens contendo faces foram excluídas por baixo percentual de pele. A aplicação dos teste apenas neste conjunto de imagens limitou o possibilidade de resultados mais expressivos.

Para trabalhos futuros, sugere-se um estudo para este algoritmo com conjuntos de imagens diferentes para que se possa avaliar de forma mais abrangente o comportamento da TPR. Como também, a aplicação do algoritmo em outros detectores de faces e uma possível implementação de um detector de pele mais robusto que seja aplicável a imagens em representação de tons de cinza.

Por fim, vale ressaltar que vários conteúdos da disciplina Processamento Digital de Sinais, experiências extra classe adquiridas no Laboratório de Visão Computacional (*LVC*) foram de fundamental importância para realização deste TCC. Assim, este contribuiu para consolidação e aplicação de conceitos nas aéreas de Processamento Digital de Imagens *PDI* e Visão Computacional.

Referências

ANCHIT, A.; MATHUR, S. Comparative analysis of haar and skin color method for face detection. In: IEEE. *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*. [S.l.], 2014. p. 1–5. Citado 2 vezes nas páginas 15 e 35.

ERKEL, A. R. v.; PATTYNAMA, P. M. T. Receiver operating characteristic (roc) analysis: Basic principles and applications in radiology. *European Journal of Radiology*, v. 27, p. 88—94, May 1998. Citado 2 vezes nas páginas 27 e 29.

FAWCETT, T. An introduction to roc analysis. *In: Pattern Recognition Letters*, v. 27, p. 861—874, December 2005. Citado 2 vezes nas páginas 27 e 30.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *European Journal of Radiology*, v. 55, p. 119—-139, August 1997. Citado na página 19.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Additive logistic regression: a statistical view of boosting. annals of statistics. *Annals of Statistics*, v. 28, n. 2, p. 337–407, 2000. Citado na página 19.

JAIN, V.; LEARNED-MILLER, E. *FDDB: A Benchmark for Face Detection in Unconstrained Settings*. [S.l.], 2010. Citado 2 vezes nas páginas 15 e 40.

JAIN, V.; LEARNED-MILLER, E. *Face Detection Data Set and Benchmark Home (FDDB : Main)*. 2015. <http://vis-www.cs.umass.edu/fddb/>. Acesso 22 de Dezembro de 2015. Citado 2 vezes nas páginas 15 e 40.

JONES, M. J.; REHG, J. M. Statistical color models with application to skin detection. *International Journal of Computer Vision*, v. 46, p. 81–96, January 2002. Citado 2 vezes nas páginas 15 e 24.

KAWULOK, M.; KAWULOK, J.; NALEPA, J. Spatial-based skin setection using discriminative skin-presence features. *Pattern Recognition Letters*, v. 41, p. 3–13, May 2014. Citado 3 vezes nas páginas 15, 24 e 25.

LIU, C.; CHANG, F. Cascaded split-level colour haar-like features for object detection. *Electronics Letters*, v. 51, p. 2106–2107, December 2015. Citado na página 15.

MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, v. 16, n. 1, p. 125–160, 2009. Citado na página 58.

MARKUS, N. et al. A method for object detection based on pixel intensity comparisons. *Computing Research Repository*, abs/1305.4537, march 2013. Citado 4 vezes nas páginas 17, 19, 20 e 22.

OPENCV, D. *Cascade Classification — OpenCV*. 2015. <http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html>. Acesso 15 de Dezembro de 2015. Citado na página 20.

PAPAGEORGIOU, C.; OREN, M.; POGGIO, T. A general framework for object detection. *International Conference on Computer Vision*, p. 555—562, January 1998. Citado na página 17.

PEREIRA, E. T. *Investigação do Problema de Detecção de Faces com Variações de Orientação*. Tese — Universidade Federal de Campina Grande, Maio 2012. Citado 3 vezes nas páginas 17, 19 e 35.

PHUNG, S.; CHAI, D.; BOUZERDOUM, A. Adaptive skin segmentation in color images. *In: Proceedings of IEEE ICASSP*, v. 3, p. 353–356, April 2003. Citado na página 15.

VIOLA, P.; JONES, M. Robust real-time face detection. *International Journal of Computer Vision*, v. 57, p. 137–154, May 2004. Citado 3 vezes nas páginas 15, 17 e 19.

Apêndices

APÊNDICE A – *OpenCV*

OpenCV (Open Source Computer Vision) é uma biblioteca de programação, de código aberto, desenvolvida inicialmente pela *Intel Corporation*. O *OpenCV* implementa uma variedade de ferramentas de interpretação de imagens, indo desde operações simples como um filtro de ruído, até operações complexas, tais como a análise de movimentos, reconhecimento de padrões e reconstrução em *3D*. O pacote *OpenCV* está disponível gratuitamente na Internet bem como o manual de referência (MARENGONI; STRINGHINI, 2009).

A biblioteca *OpenCV* foi desenvolvida pela Intel e possui mais de 500 funções. Foi idealizada com o objetivo de tornar a visão computacional acessível a usuários e programadores em áreas tais como a interação humano-computador em tempo real e a robótica. A biblioteca está disponível com o código fonte e os executáveis (binários) otimizados para os processadores Intel. Um programa *OpenCV*, ao ser executado, invoca automaticamente uma *DLL (Dynamic Linked Library)* que detecta o tipo de processador e carrega, por sua vez, a *DLL* otimizada para este. Juntamente com o pacote *OpenCV* é oferecida a biblioteca *IPL (Image Processing Library)*, da qual a *OpenCV* depende parcialmente, além de documentação e um conjunto de códigos exemplos (MARENGONI; STRINGHINI, 2009).

A biblioteca está dividida em cinco grupos de funções: Processamento de imagens; Análise estrutural; Análise de movimento e rastreamento de objetos; Reconhecimento de padrões e Calibração de câmera e reconstrução *3D* (MARENGONI; STRINGHINI, 2009).

APÊNDICE B – *CascadeClassifier::detectMultiScale*

- **cascade** – Haar classifier cascade (OpenCV 1.x API only). It can be loaded from XML or YAML file using **Load()**. When the cascade is not needed anymore, release it using cvReleaseHaarClassifierCascade(&cascade).
- **image** – Matrix of the type CV_8U containing an image where objects are detected.
- **objects** – Vector of rectangles where each rectangle contains the detected object.
- **scaleFactor** – Parameter specifying how much the image size is reduced at each image scale.
- **minNeighbors** – Parameter specifying how many neighbors each candidate rectangle should have to retain it.
- **flags** – Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.
- **minSize** – Minimum possible object size. Objects smaller than that are ignored.
- **maxSize** – Maximum possible object size. Objects larger than that are ignored.

APÊNDICE C – classe *faceDetector* para o PICO

Código C.1 – Detalhes da classe *faceDetector* para Implementação do *PICO*

```
#ifndef _SAMPLE_H
#define _SAMPLE_H

using namespace std;
using namespace cv;

class faceDetector {
public:
    faceDetector();
    faceDetector(int minS, int maxS);
    faceDetector(float qThreshold, float scaleFactor,
    float strideFactor,
            int usePyr, int minS, int maxS);
    float getticks();
    vector<Rect> detectFaces(Mat frameMat,
    vector<float> &vConfidenceParam);
    Mat drawFaces(Mat frame, vector<Rect> vRect);
    //void processFromWebcam();

    int getMaxSize();
    void setMaxSize(int maxSize);
    int getMinSize();
    void setMinSize(int minSize);
    float getQthreshold();
    void setQthreshold(float qthreshold);
    float getScalefactor();
    void setScalefactor(float scalefactor);
    float getStridefactor();
    void setStridefactor(float stridefactor);
    int getUsepyr();
    void setUsepyr(int usepyr);

private:
    // * detection quality threshold (must be >= 0.0f)
    // * you can vary the TPR and FPR with this value
    // * if you're experiencing too many false positives,
    try a larger number here (for example, 7.5f)
    float qthreshold; // Default is 5.0f

    // * how much to rescale the window
    during the multiscale detection process
    // * increasing this value leads to lower
     number of detections and higher processing speed
    // * for example, set to 1.2f if you're
    using pico on a mobile device
    float scalefactor;

    // * how much to move the window between
    neighboring detections
```

```
    // * increasing this value leads to lower number of
    detections and higher processing speed
    // * for example, set to 0.05f if you want
    really high recall
    float stridefactor;

    // * coarse image pyramid support
    // * can improve noise and aliasing problems in
    some applications
    // * set to 1 if pico fails to detect large objects
    int usepyr;

    int minSize, maxSize;
};

#endif
```