Trabalho Final PDI
Utilização de técnicas de detecção de faces em imagens de baixa resolução

Departamento de Computação e Automação - DCA/UFRN
Engenharia de Computação e Automação - DCA/UFRN
Aluno/Discente: Lukas Maximo Grilo Abreu Jardim
Professor/Docente: Agostinho Brito de Medeiros Jr.

June 27, 2018

Figure 1: Múltiplas faces em uma imagem.

1 Introdução

O Objetivo desse trabalho era demonstrar e adaptar um sistema de reconhecimento de faces para imagens em baixa resolução, com a utilização da técnica de Haar Cascade, uma método que consiste em aplicar máscaras de filtros em uma imagem afim de se obter uma correspondência com um banco de dados definido previamente, definida a partir da combinação de vários destes filtros em um sistema de detecção que envolve técnicas de aprendizado de maquina.

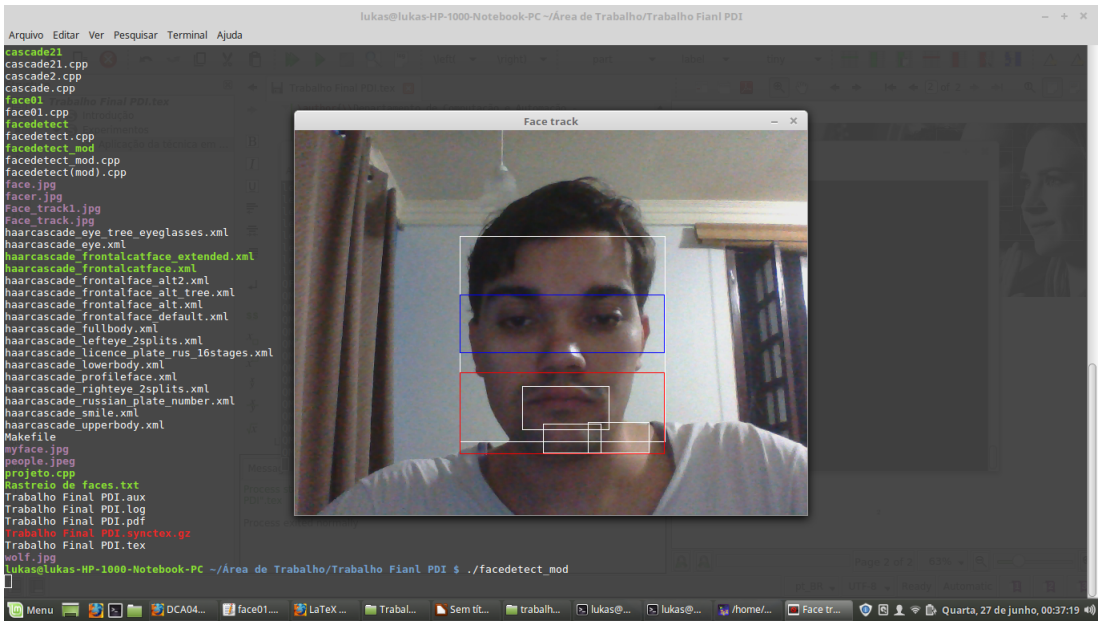

Figure 2: vários rostos detectados em uma mesma imagem.

2 Experimentos

2.1 Aplicação da técnica em uma imagem de vídeo

Inicialmente, a técnica de reconhecimento foi aplicada em uma imagem de vídeo obtida através de uma câmera de computador onde ela procura catalogar e reconhecer vários elementos além do rosto da pessoa (no caso, os dois olhos e a boca), para melhor representação do método.

Figure 3: Uma Imagem em perfeitas condições.

2.2 detecção de faces em uma imagem

No segundo teste, o algoritmo de detecção foi aplicado em duas imagem em tons de cinza bem definidas, e da mesma maneira que foi executada na imagem de vídeo, a recognição foi realizada normalmente.
Nas imagens acima, que estão em boas condições, foram processadas por um algorítimo não muito diferentes do algorítmo de vídeo, mas que também processou a imagem para encontrar todas as faces possíveis, tantoque ele não detectou todas as faces direito na imagem superior.

Figure 4: Imagem definitiva.

3 Teste Definitivo: Detecção de faces em uma imagem em baixa qualidade

Por último, e mais importante, o algoritmo foi utilizado em uma imagem que continha várias faces, e ao mesmo tempo, possuía diferentes graus de péssima qualidade de visualização. Mesmo assim, com a baixa qualidade da imagem, a partir de um certo limite, o algorítimo permite que ela ainda possa ter faces detectadas, como demonstrado na imagem acima.

Para realizar multiplas detecções, foi utilizado um vetor que armazena cada face ou elemento de face detectadas pelo algoritmo (cada elemento de cada vez), onde uma vez detectado uma face, a mesma face não vai ser detectada novamente.

4 Conclusão

Os códigos utilizados nesse trabalho, vide Apêndice, foram melhorados ou adaptados com o intuito de atender as exigências requeridas no mesmo. Os algorítmos utilizados para detectar as faces nas imagens foram adaptados para serem exibidos com maior visibilidade na detecção.

5 Apêndice

5.1 Códigos do Trabalho

5.1.1 facedetect modificado

```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>
#include <stdio.h>

using namespace std;
using namespace cv;

void detectAndDraw( Mat& img);

CascadeClassifier cascadeFace, cascadeMouth, cascadeEye;

String cascadeFaceName  = "haarcascade_frontalface_alt.xml";
String cascadeMouthName = "haarcascade_smile.xml";
String cascadeEyeName = "haarcascade_eye_tree_eyeglasses.xml";

int main( int argc, const char** argv ){
CvCapture* capture = 0;
Mat frame, frameCopy, image;
VideoCapture cap(0);
int key;

if( !cascadeFace.load( cascadeFaceName )) {
cerr << "ERRO: Nao carregou filtro em cascata facefrontal" << endl;
return -1;
}
if( !cascadeMouth.load( cascadeMouthName )) {
cerr << "ERRO: Nao carregou filtro em cascata mouth" << endl;
return -1;
}
if( !cascadeEye.load( cascadeEyeName )) {
cerr << "ERRO: Nao carregou filtro em cascata olho" << endl;
return -1;
}

//  cap.set(CV_CAP_PROP_FRAME_WIDTH, 320);
//  cap.set(CV_CAP_PROP_FRAME_HEIGHT, 240);

for(;;){
cap >> frame;
flip(frame, frameCopy, 1); // inverte a imagem horizontalmente
///imshow("image", frameCopy);
//cout << "foi\n";
```

```
46          detectAndDraw(frameCopy); // detecta
47
48          key = (char) waitKey(10);
49          if( key == 27 ) break;
50          }
51          return 0;
52          }
53
54          void detectAndDraw( Mat& img){
55          int i = 0;
56          double t = 0;
57          vector<Rect> faces;
58
59          Mat gray;
60          cascadeFace.detectMultiScale(img, // imagem para deteccao
61          faces,
62          1.1,
63          3,
64          0 | CV_HAAR_FIND_BIGGEST_OBJECT, // (normalmente nao usados
                )
65          Size(30, 30) ); // minimo tamanho para deteccao de um
                objeto
66
67          for( vector<Rect>::const_iterator r = faces.begin(); r !=
                faces.end(); r++ ){
68          Mat imgROI, imgROI2;
69          vector<Rect> nestedObjects, nestedObjects2;
70
71          rectangle(img,
72          Point(r->x, r->y),
73          Point(r->x + r->width, r->y + r->height),
74          CV_RGB(255, 255, 255), 1, 8, 0);
75
76          if( cascadeMouth.empty() )
77          continue;
78
79          Rect mouthROI = Rect(r->x, r->y + (r->height/1.5),
80          r->width, r->height/2.5);
81
82
83          rectangle(img, mouthROI, CV_RGB(255, 0, 0), 1, 8, 0);
84
85          imgROI = img(mouthROI);
86
87          cascadeMouth.detectMultiScale(
88          imgROI,
89          nestedObjects,
90          1.1,
91          2,
92          0 | CV_HAAR_FIND_BIGGEST_OBJECT,
93          Size(30, 30) );
94
95          // busca os olhos e as bocas encontradas e desenha os
                retangulos
96          for( vector<Rect>::const_iterator nr = nestedObjects.begin
                (); nr != nestedObjects.end(); nr++ ){
97          rectangle(img,
```

```
98          Point(r->x + nr->x, r->y + (r->height/1.5) + nr->y  ),
99          Point(r->x + nr->x + nr->width, r->y + (r->height/1.5) + nr
                ->y + nr->height),
100         CV_RGB(255, 255, 255), 1, 8, 0);
101         }
102         if( cascadeEye.empty() )
103         continue;
104         Rect eyeROI = Rect(r->x, r->y + (r->height/3.5),
105         r->width, r->height/3.5);
106         rectangle(img, eyeROI, CV_RGB(0, 0, 255), 1, 8, 0);
107
108         imgROI2 = img(eyeROI);
109         cascadeEye.detectMultiScale(
110         imgROI,
111         nestedObjects,
112         1.1,
113         2,
114         0 | CV_HAAR_FIND_BIGGEST_OBJECT,
115         Size(30, 30) );
116         for( vector<Rect>::const_iterator nr = nestedObjects.begin
                (); nr != nestedObjects.end(); nr++ ){
117         rectangle(img,
118         Point(r->x + nr->x, r->y + (r->height/3.5) + nr->y  ),
119         Point(r->x + nr->x + nr->width, r->y + (r->height/3.5) + nr
                ->y + nr->height),
120         CV_RGB(255, 255, 255), 1, 8, 0);
121         }
122         }
123         imshow("Face_track", img );
124         }
```

5.1.2 face01

```
1  #include "opencv2/objdetect/objdetect.hpp"
2  #include "opencv2/highgui/highgui.hpp"
3  #include "opencv2/imgproc/imgproc.hpp"
4
5  #include <iostream>
6  #include <stdio.h>
7
8  using namespace std;
9  using namespace cv;
10
11 void detectAndDraw( Mat& img);
12
13 CascadeClassifier cascadeFace, cascadeMouth, cascadeEye;
14
15 String cascadeFaceName  = "haarcascade_frontalface_alt.xml"
       ;
16 String cascadeMouthName = "haarcascade_smile.xml";
17 String cascadeEyeName = "haarcascade_eye_tree_eyeglasses.
       xml";
18
19 int main( int argc, const char** argv ){
20 Mat image;
21 int key;
22
23 if( !cascadeFace.load( cascadeFaceName )) {
24 cerr << "ERRO: Nao carregou filtro em cascata facefrontal"
       << endl;
25 return -1;
26 }
27 if( !cascadeMouth.load( cascadeMouthName )) {
28 cerr << "ERRO: Nao carregou filtro em cascata mouth" <<
       endl;
29 return -1;
30 }
31 if( !cascadeEye.load( cascadeEyeName )) {
32 cerr << "ERRO: Nao carregou filtro em cascata olho" << endl
       ;
33 return -1;
34 }
35
36
37 for(;;){
38 image = imread(argv[1],CV_LOAD_IMAGE_GRAYSCALE);
39 detectAndDraw(image); // detecta
40
41 key = (char) waitKey(10);
42 if( key == 27 ) break;
43 }
44 return 0;
45 }
46
47 void detectAndDraw( Mat& img){
48 int i = 0;
49 double t = 0;
50 vector<Rect> faces;
```

```
51
52          Mat gray;
53          cascadeFace.detectMultiScale(img,
54          faces,
55          1.1,
56          3,
57          0 | CV_HAAR_FIND_BIGGEST_OBJECT,  (normalmente nao usados)
58          Size(30, 30) );
59
60          for( vector<Rect>::const_iterator r = faces.begin(); r !=
                faces.end(); r++ ){
61          Mat imgROI, imgROI2;
62          vector<Rect> nestedObjects, nestedObjects2;
63
64          rectangle(img,
65          Point(r->x, r->y),
66          Point(r->x + r->width, r->y + r->height),
67          CV_RGB(255, 255, 255), 1, 8, 0);
68
69          if( cascadeMouth.empty() )
70          continue;
71
72          // posicao aproximada da boca em relacao a face...
73          Rect mouthROI = Rect(r->x, r->y + (r->height/1.5),
74          r->width, r->height/2.5);
75
76
77          rectangle(img, mouthROI, CV_RGB(255, 255, 255), 1, 8, 0);
78
79          imgROI = img(mouthROI);
80
81          cascadeMouth.detectMultiScale(
82          imgROI,
83          nestedObjects,
84          1.1,
85          2,
86          0 | CV_HAAR_FIND_BIGGEST_OBJECT,
87          Size(30, 30) );
88
89          // busca os olhos e as bocas encontradas e desenha os
                retangulos
90          for( vector<Rect>::const_iterator nr = nestedObjects.begin
                (); nr != nestedObjects.end(); nr++ ){
91          rectangle(img,
92          Point(r->x + nr->x, r->y + (r->height/1.5) + nr->y  ),
93          Point(r->x + nr->x + nr->width, r->y + (r->height/1.5) + nr
                ->y + nr->height),
94          CV_RGB(255, 255, 255), 1, 8, 0);
95          }
96          if( cascadeEye.empty() )
97          continue;
98          Rect eyeROI = Rect(r->x, r->y + (r->height/3.5),
99          r->width, r->height/3.5);
100         rectangle(img, eyeROI, CV_RGB(255, 255, 255), 1, 8, 0);
101
102         imgROI2 = img(eyeROI);
103         cascadeEye.detectMultiScale(
```

```
104             imgROI,
105             nestedObjects,
106             1.1,
107             2,
108             0 | CV_HAAR_FIND_BIGGEST_OBJECT,
109             Size(30, 30) );
110             for( vector<Rect>::const_iterator nr = nestedObjects.begin
                    (); nr != nestedObjects.end(); nr++ ){
111             rectangle(img,
112             Point(r->x + nr->x, r->y + (r->height/3.5) + nr->y  ),
113             Point(r->x + nr->x + nr->width, r->y + (r->height/3.5) + nr
                    ->y + nr->height),
114             CV_RGB(255, 255, 255), 1, 8, 0);
115             }
116             }
117             imwrite("Face_track.jpg", img );
118             imshow("Face_track", img );
119             }
```