```
 1 /*
 2  * face_detection_CPU_final.cpp
 3  *
 4  *  Created on: May 23, 2017
 5  *  Author: Patricia Navarro Martín
 6
 7  By downloading, copying, installing or using the software you agree to
   this license.
 8 If you do not agree to this license, do not download, install,
 9 copy or use the software.
10
11
12                           License Agreement
13                For Open Source Computer Vision Library
14                        (3-clause BSD License)
15
16 Copyright (C) 2000-2015, Intel Corporation, all rights reserved.
17 Copyright (C) 2009-2011, Willow Garage Inc., all rights reserved.
18 Copyright (C) 2009-2015, NVIDIA Corporation, all rights reserved.
19 Copyright (C) 2010-2013, Advanced Micro Devices, Inc., all rights
   reserved.
20 Copyright (C) 2015, OpenCV Foundation, all rights reserved.
21 Copyright (C) 2015, Itseez Inc., all rights reserved.
22 Third party copyrights are property of their respective owners.
23
24 Redistribution and use in source and binary forms, with or without
   modification,
25 are permitted provided that the following conditions are met:
26
27   * Redistributions of source code must retain the above copyright notice,
28     this list of conditions and the following disclaimer.
29
30   * Redistributions in binary form must reproduce the above copyright
   notice,
31     this list of conditions and the following disclaimer in the
   documentation
32     and/or other materials provided with the distribution.
33
34   * Neither the names of the copyright holders nor the names of the
   contributors
35     may be used to endorse or promote products derived from this software
36     without specific prior written permission.
37
38 This software is provided by the copyright holders and contributors "as
   is" and
39 any express or implied warranties, including, but not limited to, the
   implied
40 warranties of merchantability and fitness for a particular purpose are
   disclaimed.
41 In no event shall copyright holders or contributors be liable for any
   direct,
42 indirect, incidental, special, exemplary, or consequential damages
43 (including, but not limited to, procurement of substitute goods or
```

```
  services;
44 loss of use, data, or profits; or business interruption) however caused
45 and on any theory of liability, whether in contract, strict liability,
46 or tort (including negligence or otherwise) arising in any way out of
47 the use of this software, even if advised of the possibility of such
  damage.
48  */
49
50 #include "opencv2/core/core.hpp"
51 #include "opencv2/highgui/highgui.hpp"
52 #include "opencv2/imgproc/imgproc.hpp"
53 #include "opencv2/objdetect/objdetect.hpp"
54 #include <iostream>
55 #include <fstream>
56 #include <sys/time.h>
57 #include  <iomanip>
58
59  using namespace std;
60  using namespace cv;
61
62 struct timeval crono_on, crono_off;
63 struct timeval empieza, acaba;
64
65 //Función para calcular los FPS
66  double calc_fps(void)
67  {
68      gettimeofday(&crono_off,NULL);
69      double us = (crono_off.tv_usec-crono_on.tv_usec);
70      double s = (crono_off.tv_sec-crono_on.tv_sec);
71      double total = s + (us/1000000);
72      double fps = 1/total;
73      cout << setw(3) << fixed << fps << " FPS "<<endl;
74      gettimeofday(&crono_on,NULL);
75      return fps;
76  }
77
78  int main( int argc, const char** argv )
79 {
80      //Inicio del temporizador del programa global y definición de display
  en la Jetson TK1
81      gettimeofday(&empieza,NULL);
82      setenv("DISPLAY", ":0",0);
83
84      //Declaración de las variables iniciales
85      CascadeClassifier cascade_cpu;
86      string cascadeName;
87      VideoCapture capture;
88
89      //Control del número de argumentos de entrada del programa
90      cout<<argc<<endl;
91      if(argc != 3)
92      {
93             cerr<<"(!) Argumento no válido. Debe ser: video_demo/webcam
```

```
   <dirección_del_classificador>"<<endl;
 94             return -1;
 95     }
 96     else
 97     {
 98             if(string(argv[1])=="webcam")
 99             {
100                 capture.open(0);
101             }
102             else if(string(argv[1])=="video_demo")
103             {
104                 capture.open("/home/ubuntu/Desktop/video_demo.mp4");
105             }
106             else
107             {
108                 cerr<<"(!) Argumento no válido. Debe ser: video_demo/
   webcam <dirección_del_classificador>"<<endl;
109                 return -1;
110             }
111
112             cascadeName = string(argv[2]);
113     }
114
115
116     //DEFINICIÓN DE VARIABLES
117     Mat frame;
118     int frame_num=1;
119     string log_path="./RESULTADOS/"+string(argv[1])+".csv";
120
121     //COMPROBACIÓN DE RECURSOS
122     if(!capture.isOpened())
123     {
124         cerr << "(!) No se pudo abrir:" << string(argv[1])<<endl;
125         return -1;
126     }
127     if(!cascade_cpu.load(cascadeName))
128     {
129         cerr << "(!) No se pudo cargar el clasificador" << string(argv
   [2])<<endl;
130         return -1;
131     }
132
133     //APERTURA DEL LOG
134     ofstream log;
135     log.open(log_path.c_str());
136
137     //COMPROBACIONES PREVIAS AL VIDEOWRITER
138     capture >> frame;
139     // check if we succeeded
140     if (frame.empty()) {
141         cerr << "(!) Frame vacío\n";
142         return -1;
143     }
```

```
144     bool isColor = (frame.type() == CV_8UC3);
145     int height = capture.get(CV_CAP_PROP_FRAME_HEIGHT);
146 
147     //DECLARACIÓN DEL VIDEOWRITER
148     VideoWriter writer;
149     int codec = CV_FOURCC('M', 'J', 'P', 'G');
150     double fps_write = 10.0;
151     string filename = "./RESULTADOS/"+string(argv[1])+".avi";
152     writer.open(filename, codec, fps_write, frame.size(), isColor);
153     //Control de errores en la creación
154     if (!writer.isOpened()) {
155        cerr << "((!) No se pudo abrir el archivo de video para escribir.
    \n";
156        return -1;
157     }
158     cout << "Archivo de video: " << filename << endl;
159 
160 
161 
162    //_____________INICIO DE LECTURA-ANÁLISIS-GRABACIÓN_____________
163     //Iniciamos el cronómetro para el cálculo del fps
164     gettimeofday(&crono_on,NULL);
165 
166    for(;;)
167    {
168     //Captura del frame
169     capture >> frame;
170     //Comprobar que no se ha terminado el video
171     if (!capture.read(frame)) {
172        cout << "Video de lectura finalizado\n";
173        break;
174     }
175 
176     //Comienza la detección
177     std::vector<Rect> faces;
178     Mat frame_gray;
179     cvtColor( frame, frame_gray, CV_BGR2GRAY );
180 
181     cascade_cpu.detectMultiScale( frame_gray, faces, 1.25, 4);
182     //Detección finalizada
183 
184     //Localización de los rostros
185     for( size_t i = 0; i < faces.size(); i++ )
186     {
187              Point pt1 = faces[i].tl();
188              Size sz = faces[i].size();
189              Point pt2(pt1.x+sz.width, pt1.y+sz.height);
190              rectangle(frame, pt1, pt2, Scalar(255,255,0),3,8);
191     }
192 
193     //Cálculo de los frames por segundo y almacenamiento del valor en el
    log
194     double fps = calc_fps();
```

```
195     log<<fps<<"\n";
196 
197     //Escritura de texto informativo en frame
198     ostringstream ss;
199     ss<<"FPS = "<<fixed<<fps<< "  con CPU";
200     putText(frame,ss.str(),Point(40,
    (height-25)),CV_FONT_HERSHEY_DUPLEX,0.8,Scalar(255,255,0),1,8,false);
201 
202     writer.write(frame);
203 
204     //Mostrar el frame resultante por pantalla y grabarlo en el archivo de
    video de escritura.
205     imshow("Detección facial - Versión CPU", frame);
206 
207     //Mantiene el frame durante 1ms y el programa se puede interrumpir si
    se pulsa la tecla Esc (en ASCII 27)
208     int c = waitKey(1);
209     if( (char)c == 27 )break;
210 
211     frame_num++;
212    }
213    //_____________FIN DE LECTURA-ANÁLISIS-GRABACIÓN_____________
214 
215     //Cálculo de duración del programa total
216      gettimeofday(&acaba,NULL);
217      double us = (acaba.tv_usec-empieza.tv_usec);
218      double s = (acaba.tv_sec-empieza.tv_sec);
219      int min = (int)s/60;
220      int seg = (int)s%60;
221      double total = s + us/1000000;
222 
223      //Mostrar resultado por consola y registrar en log
224      cout << "Tiempo de ejecución total (segundos):  "<<total<<endl;
225      cout<< min <<" minutos "<<seg<<" segundos"<<endl;
226      log<<total<<"segundos\n";
227 
228       //Cerrar el log
229       log.close();
230 }
231 
232 
233 
234 
235 
236 
```