**使用OpenCV与Dlib提取人脸地标点**

由于OpenCV没有提取人脸地标点的相关函数，而Dlib的人脸检测使用Hog特征，速度很慢，因此结合OpenCV与Dlib来进行人有地标点的提取。

    Dlib的配置在这里不再介绍，可参考之前写的“DLIB编译与使用”文档。代码很简单，在这里直接贴出代码：

#include "stdafx.h"

#include <dlib/opencv.h>

#include <opencv2/opencv.hpp>

#include <dlib/image\_processing/frontal\_face\_detector.h>

#include <dlib/image\_processing/render\_face\_detections.h>

#include <dlib/image\_processing.h>

#include <dlib/gui\_widgets.h>

using namespace dlib;

using namespace std;

int *main*()

{

    try

    {

        cv::*VideoCapture* cap(1);

        if (!cap.*isOpened*())

        {

*cerr* << "Unable to connect to camera" << *endl*;

            return 1;

        }

        image\_window win;

        shape\_predictor pose\_model;

        cv::*CascadeClassifier* cascader;

        cascader.*load*("F:\\ImageFiles\\lbpcascade\_frontalface.xml");

        deserialize("F:\\ImageFiles\\shape\_predictor\_68\_face\_landmarks.dat") >> pose\_model;

        // Grab and process frames until the main window is closed by the user.

        while (!win.is\_closed())

        {

            // Grab a frame

            cv::*Mat* temp;

            cap >> temp;

            cv::*resize*(temp, temp, cv::*Size*(320, 240));

            std::vector<cv::*Rect*> faceRects;

            cascader.*detectMultiScale*(temp, faceRects);

            cv::*Rect* singleFace;

            if (faceRects.*size*() > 0)

            {

                for (int i = 0; i < faceRects.*size*(); i++)

                {

                    if (singleFace.*area*() < faceRects[i].area())

                    {

                        singleFace = faceRects[i];

                    }

                }

*cv\_image*<bgr\_pixel> cimg(temp);

                std::vector<full\_object\_detection> shapes;

                rectangle dlbRect(singleFace.*x*, singleFace.*y*, singleFace.*x* + singleFace.*width*,

                    singleFace.*y* + singleFace.*height*);

                shapes.*push\_back*(pose\_model(cimg, dlbRect));

                // Display it all on the screen

                win.clear\_overlay();

                win.set\_image(cimg);

                win.add\_overlay(render\_face\_detections(shapes));

            }

        }

    }

    catch (serialization\_error& e)

    {

*cout* << "You need dlib's default face landmarking model file." << *endl*;

*cout* << *endl* << e.what() << *endl*;

    }

    catch (*exception*& e)

    {

*cout* << e.what() << *endl*;

    }

}

