# SeetaFace的使用

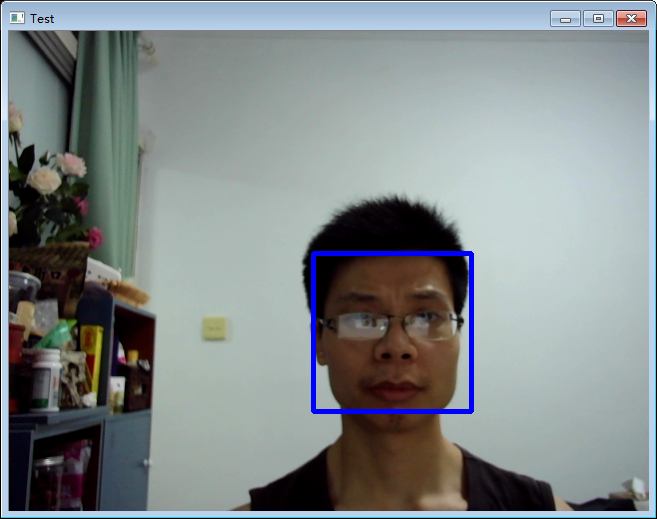
SeetaFace包括人脸检测模块、人脸对齐模块和人脸识别模块，项目地址：

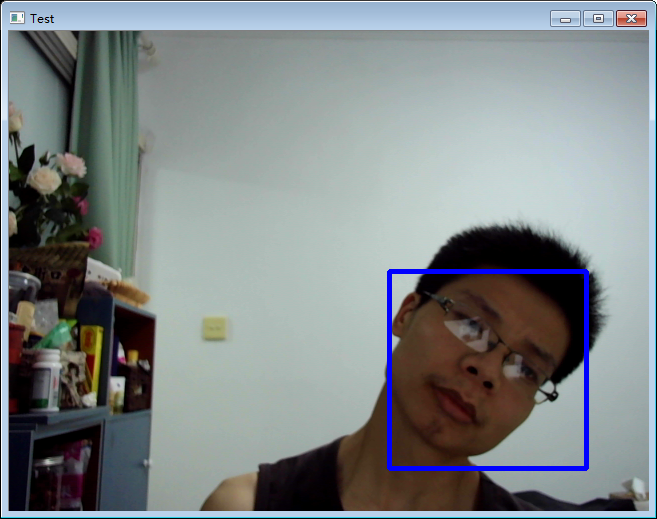
<https://github.com/seetaface/SeetaFaceEngine>，项目中包括详细的配置文档，参考该文档即可完成工程配置，所以配置过程在这里省略。

## 1人脸检测模块

配置好工程后，我仿照官方的例子，编写了以下代码进行测试：

|  |
| --- |
| #include <cstdint>  #include <fstream>  #include <iostream>  #include <string>  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include "face\_detection.h"  using namespace std;  using namespace cv;  int main(int argc, char\*\* argv) {  const char\* img\_path = "data/test\_01.jpg";  seeta::FaceDetection detector("data/seeta\_fd\_frontal\_v1.0.bin");  detector.SetMinFaceSize(40);  detector.SetScoreThresh(2.f);  detector.SetImagePyramidScaleFactor(0.8f);  detector.SetWindowStep(4, 4);  VideoCapture cap(0);  Mat img;  Mat img\_gray;  cap >> img;  while (img.data)  {  if (img.channels() != 1)  cv::cvtColor(img, img\_gray, cv::COLOR\_BGR2GRAY);  else  img\_gray = img;  seeta::ImageData img\_data;  img\_data.data = img\_gray.data;  img\_data.width = img\_gray.cols;  img\_data.height = img\_gray.rows;  img\_data.num\_channels = 1;  long t0 = cv::getTickCount();  std::vector<seeta::FaceInfo> faces = detector.Detect(img\_data);  long t1 = cv::getTickCount();  double secs = (t1 - t0) / cv::getTickFrequency();  cout << "Detections takes " << secs << " seconds " << endl;  cout << "Image size (wxh): " << img\_data.width << "x"  << img\_data.height << endl;  cv::Rect face\_rect;  int32\_t num\_face = static\_cast<int32\_t>(faces.size());  for (int32\_t i = 0; i < num\_face; i++) {  face\_rect.x = faces[i].bbox.x;  face\_rect.y = faces[i].bbox.y;  face\_rect.width = faces[i].bbox.width;  face\_rect.height = faces[i].bbox.height;  cv::rectangle(img, face\_rect, CV\_RGB(0, 0, 255), 4, 8, 0);  }  cv::namedWindow("Test", cv::WINDOW\_AUTOSIZE);  cv::imshow("Test", img);  cv::waitKey(10);  cap >> img;  }  } |





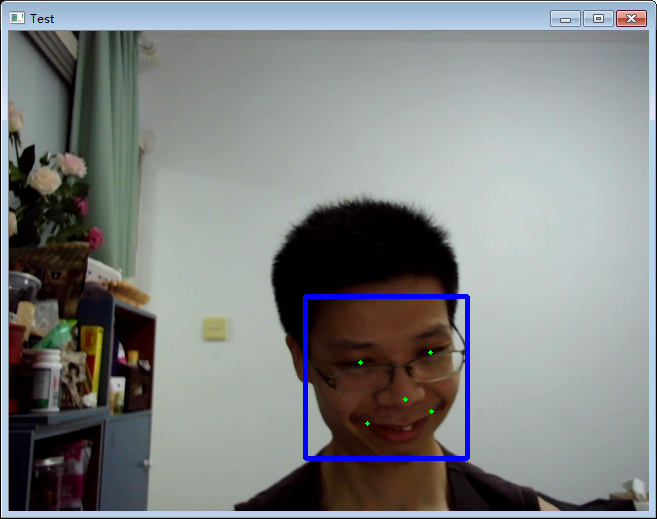
SeetaFace的人脸检测效果较好，对倾斜的人脸也能检测出来，速度也较快，在我的机器上平均160ms左右能够完成640\*480大小图像的处理。

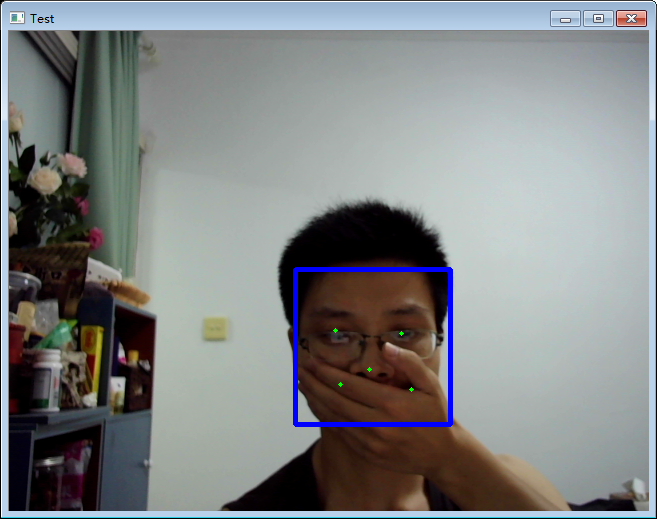
## 2人脸对齐模块

配置好工程后，我仿照官方的例子，编写了以下代码进行测试：

|  |
| --- |
| #include <cstdint>  #include <fstream>  #include <iostream>  #include <string>  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include "face\_detection.h"  #include "face\_alignment.h"  using namespace std;  using namespace cv;  int main(int argc, char\*\* argv) {  //人脸检测模型  seeta::FaceDetection detector("data/seeta\_fd\_frontal\_v1.0.bin");  detector.SetMinFaceSize(40);  detector.SetScoreThresh(2.f);  detector.SetImagePyramidScaleFactor(0.8f);  detector.SetWindowStep(4, 4);  //人脸对齐模型  seeta::FaceAlignment point\_detector("data/seeta\_fa\_v1.1.bin");  VideoCapture cap(0);  Mat img;  Mat img\_gray;  cap >> img;  while (img.data)  {  if (img.channels() != 1)  cv::cvtColor(img, img\_gray, cv::COLOR\_BGR2GRAY);  else  img\_gray = img;  seeta::ImageData img\_data;  img\_data.data = img\_gray.data;  img\_data.width = img\_gray.cols;  img\_data.height = img\_gray.rows;  img\_data.num\_channels = 1;  long t0 = cv::getTickCount();  //检测人脸  std::vector<seeta::FaceInfo> faces = detector.Detect(img\_data);  cv::Rect face\_rect;  int32\_t num\_face = static\_cast<int32\_t>(faces.size());  for (int32\_t i = 0; i < num\_face; i++) {  face\_rect.x = faces[i].bbox.x;  face\_rect.y = faces[i].bbox.y;  face\_rect.width = faces[i].bbox.width;  face\_rect.height = faces[i].bbox.height;  cv::rectangle(img, face\_rect, CV\_RGB(0, 0, 255), 4, 8, 0);  //检测人脸特征点  seeta::FacialLandmark points[5];  point\_detector.PointDetectLandmarks(img\_data, faces[0], points);  for (int i = 0; i<5; i++)  {  circle(img, Point(points[i].x, points[i].y), 2, CV\_RGB(0, 255, 0), -1);  }  }  long t1 = cv::getTickCount();  double secs = (t1 - t0) / cv::getTickFrequency();  cout << "Detections takes " << secs << " seconds " << endl;  cv::namedWindow("Test", cv::WINDOW\_AUTOSIZE);  cv::imshow("Test", img);  cv::waitKey(10);  cap >> img;  }  } |

从程序效果来看，SeetaFace能够准确检测出5个人脸特征点，并且当人脸局部被遮挡后，仍能够较为准确的给出人脸特征点的位置。





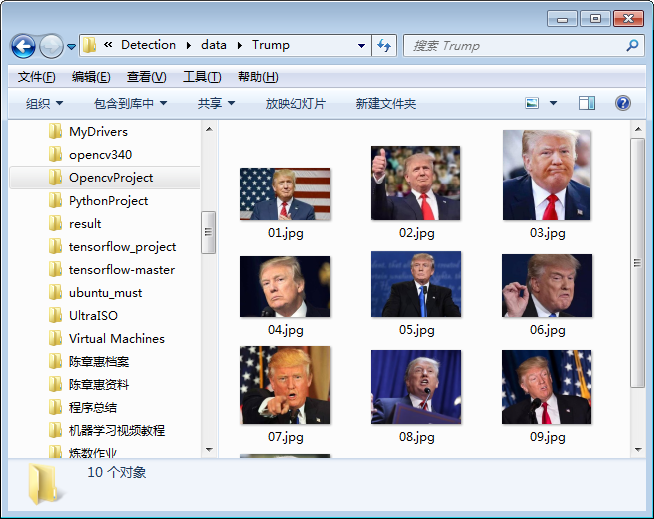
## 3人脸识别模块

为方便使用，我将人脸检测和特征点检测模块封装成两个函数模块，并参考例程编写了以下测试程序：

|  |
| --- |
| #include <cstdint>  #include <fstream>  #include <iostream>  #include <string>  #include "opencv2/highgui/highgui.hpp"  #include "opencv2/imgproc/imgproc.hpp"  #include "face\_detection.h"  #include "face\_alignment.h"  #include "face\_identification.h"  using namespace std;  using namespace cv;  #define SAMPLE\_NUM 10  struct FACE\_SHAPE  {  Rect rect;  vector<seeta::FacialLandmark> landmark;  };  vector<FACE\_SHAPE> detectFace(seeta::FaceDetection& detector, seeta::FaceAlignment& point\_detector, Mat image)  {  vector<FACE\_SHAPE> face\_shapes;  Mat img\_gray;  cvtColor(image, img\_gray, COLOR\_RGB2GRAY);  seeta::ImageData img\_data;  img\_data.data = img\_gray.data;  img\_data.width = img\_gray.cols;  img\_data.height = img\_gray.rows;  img\_data.num\_channels = 1;  //检测人脸  std::vector<seeta::FaceInfo> faces = detector.Detect(img\_data);  cv::Rect face\_rect;  int32\_t num\_face = static\_cast<int32\_t>(faces.size());  for (int32\_t i = 0; i < num\_face; i++) {  FACE\_SHAPE face\_shape;  face\_shape.rect.x = faces[i].bbox.x;  face\_shape.rect.y = faces[i].bbox.y;  face\_shape.rect.width = faces[i].bbox.width;  face\_shape.rect.height = faces[i].bbox.height;  //检测人脸特征点  seeta::FacialLandmark points[5];  point\_detector.PointDetectLandmarks(img\_data, faces[i], points);  for (int j = 0; j<5; j++)  {  face\_shape.landmark.push\_back(points[j]);  }  face\_shapes.push\_back(face\_shape);  }  return face\_shapes;  }  void FeatureExtract(seeta::FaceIdentification& face\_recognizer, FACE\_SHAPE face\_shape, Mat image, float\* feat)  {  seeta::FacialLandmark\* pt5 = &(face\_shape.landmark[0]);  seeta::ImageData src\_img\_data(image.cols, image.rows, image.channels());  src\_img\_data.data = image.data;  face\_recognizer.ExtractFeatureWithCrop(src\_img\_data, pt5, feat);  }  int main(int argc, char\*\* argv) {  //人脸检测模型  seeta::FaceDetection detector("data/seeta\_fd\_frontal\_v1.0.bin");  detector.SetMinFaceSize(40);  detector.SetScoreThresh(2.f);  detector.SetImagePyramidScaleFactor(0.8f);  detector.SetWindowStep(4, 4);  //人脸对齐模型  seeta::FaceAlignment point\_detector("data/seeta\_fa\_v1.1.bin");  //人脸特征提取  seeta::FaceIdentification face\_recognizer("data/seeta\_fr\_v1.0.bin");  int feat\_size = face\_recognizer.feature\_size();  assert(feat\_size == 2048);  int n = 50, c = 2048, h = 1, w = 1;  Mat img;  Mat img\_gray;  string Obama\_images[SAMPLE\_NUM] = {  "data/Obama/01.jpg",  "data/Obama/02.jpg",  "data/Obama/03.jpg",  "data/Obama/04.jpg",  "data/Obama/05.jpg",  "data/Obama/06.jpg",  "data/Obama/07.jpg",  "data/Obama/08.jpg",  "data/Obama/09.jpg",  "data/Obama/10.jpg",  };  string Trump\_images[SAMPLE\_NUM] = {  "data/Trump/01.jpg",  "data/Trump/02.jpg",  "data/Trump/03.jpg",  "data/Trump/04.jpg",  "data/Trump/05.jpg",  "data/Trump/06.jpg",  "data/Trump/07.jpg",  "data/Trump/08.jpg",  "data/Trump/09.jpg",  "data/Trump/10.jpg",  };  float \*obama\_feats = new float[n \* c \* w \* h \* SAMPLE\_NUM];  float \*trump\_feats = new float[n \* c \* w \* h \* SAMPLE\_NUM];  memset(obama\_feats, 0, (n \* c \* w \* h \* SAMPLE\_NUM) \* sizeof(float));  memset(trump\_feats, 0, (n \* c \* w \* h \* SAMPLE\_NUM) \* sizeof(float));  for (int i = 0;i < SAMPLE\_NUM;i++)  {  Mat img = imread(Obama\_images[i]);  vector<FACE\_SHAPE> face\_shapes = detectFace(detector, point\_detector, img);  for (int t = 0;t < face\_shapes.size();t++)  {  FeatureExtract(face\_recognizer, face\_shapes[t], img, obama\_feats + i \* feat\_size);  }  img = imread(Trump\_images[i]);  face\_shapes = detectFace(detector, point\_detector, img);  for (int t = 0;t < face\_shapes.size();t++)  {  FeatureExtract(face\_recognizer, face\_shapes[t], img, trump\_feats + i \* feat\_size);  }  }  //人脸检测及识别  Mat image = imread("data/test.jpg");  vector<FACE\_SHAPE> face\_shapes = detectFace(detector, point\_detector, image);  for (int i = 0;i < face\_shapes.size();i++)  {  float\* feat = new float[n \* c \* w \* h];  FeatureExtract(face\_recognizer, face\_shapes[i], image, feat);  //特征比对  float sim\_obama = 0, sim\_trump = 0;  for (int j = 0;j < SAMPLE\_NUM;j++)  {  float sim = face\_recognizer.CalcSimilarity(feat, trump\_feats + j \* feat\_size);  sim\_trump += sim;  cout << " ,trump\_sim:" << sim << endl;  sim = face\_recognizer.CalcSimilarity(feat, obama\_feats + j \* feat\_size);  sim\_obama += sim;  cout << "obama\_sim:" << sim;  }  sim\_obama /= SAMPLE\_NUM;  sim\_trump /= SAMPLE\_NUM;  cout << sim\_obama << "," << sim\_trump << endl;  if (max(sim\_obama, sim\_trump) > 0.5)  {  if (sim\_obama > sim\_trump)  {  rectangle(image, face\_shapes[i].rect, Scalar(0, 255, 0), 2);  putText(image, "Obama", face\_shapes[i].rect.tl(), 1, 2, Scalar(0, 255, 0), 2);  }  else  {  rectangle(image, face\_shapes[i].rect, Scalar(255, 255, 0), 2);  putText(image, "Trump", face\_shapes[i].rect.tl(), 1, 2, Scalar(255, 255, 0), 2);  }  }  }  delete[] obama\_feats;  delete[] trump\_feats;  cv::imshow("Test", image);  cv::waitKey(0);  } |

我准备了一些人物照片：





测试结果：

