



我爱计算机视觉

52CV.NET

[首页](#) > [人脸检测](#) > OPENCV VS DLIB 人脸检测比较分析

OpenCV vs Dlib 人脸检测比较分析

52cv.net 2018年10月23日

0

人脸检测是计算机视觉最典型的应用之一，早期OpenCV的logo就是Haar人脸检测的示意图。

很多人的第一个OpenCV学习目标就是跑通Haar级联人脸检测，Dlib库在业内开始流行很大程度上是因为其HOG-SVM人脸检测比OpenCV Haar的好，而近年来OpenCV和Dlib均已包含基于深度学习的人脸检测算法实现。

Haar-Cascade,HOG-SVM,深度学习正是代表着人脸检测乃至目标检测的三个时代。

昨天Learn OpenCV网站博主Vikas Gupta博士发表文章，对OpenCV与Dlib中四种人脸检测算法实现进行了比较分析，包含C++/Python的代码示例，且对精度和速度都进行了量化。

先来看看作者发布的视频：



Adobe Flash Player 已被屏蔽

1. OpenCV Haar Cascade人脸检测

算法无需赘言。

代码示例：

Python

```
1 faceCascade = cv2.CascadeClassifier('./haarcascade_frontalface_default.xml')
2 faces = faceCascade.detectMultiScale(frameGray)
3 for face in faces:
4     x1, y1, w, h = face
5     x2 = x1 + w
6     y2 = y1 + h
```

C++

```
1 faceCascadePath = "./haarcascade_frontalface_default.xml";
2 faceCascade.load( faceCascadePath )
3 std::vector<Rect> faces;
4 faceCascade.detectMultiScale(frameGray, faces);
5
6 for ( size_t i = 0; i < faces.size(); i++ )
7 {
8     int x1 = faces[i].x;
9     int y1 = faces[i].y;
10    int x2 = faces[i].x + faces[i].width;
11    int y2 = faces[i].y + faces[i].height;
12 }
```

优点

- 1) 几乎可以在CPU上实时工作；
- 2) 简单的架构；
- 3) 可以检测不同比例的人脸。

缺点

- 1) 会出现大量的把非人脸预测为人脸的情况；

2) 不适用于非正面人脸图像；

3) 不抗遮挡。

2. OpenCV DNN 人脸检测

从OpenCV3.3版本后开始引入，算法出自论文《SSD: Single Shot MultiBox Detector》(<https://arxiv.org/abs/1512.02325>)。使用ResNet-10作为骨干网。

OpenCV提供了两个模型：

1) 原始Caffe实现的16位浮点型版本 (5.4MB)；

2) TensorFlow实现的8位量化版本 (2.7MB)。

Vikas Gupta的代码包含了这两种模型。

模型加载代码示例：

Python

```
1 DNN = "TF"
2 if DNN == "CAFFE":
3     modelFile = "res10_300x300_ssd_iter_140000_fp16.caffemodel"
4     configFile = "deploy.prototxt"
5     net = cv2.dnn.readNetFromCaffe(configFile, modelFile)
6 else:
7     modelFile = "opencv_face_detector_uint8.pb"
8     configFile = "opencv_face_detector.pbtxt"
9     net = cv2.dnn.readNetFromTensorflow(modelFile, configFile)
```

C++

```
const std::string caffeConfigFile = "./deploy.prototxt";
const std::string caffeWeightFile = "./res10_300x300_ssd_iter_140000_fp16.caffemodel";

const std::string tensorflowConfigFile = "./opencv_face_detector.pbtxt";
const std::string tensorflowWeightFile = "./opencv_face_detector_uint8.pb";

#ifdef CAFFE
    Net net = cv::dnn::readNetFromCaffe(caffeConfigFile, caffeWeightFile);
#else
    Net net = cv::dnn::readNetFromTensorflow(tensorflowWeightFile, tensorflowConfigFile);
#endif
```

检测测试代码示例：

Python

```
1 blob = cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], False
2
3 net.setInput(blob)
4 detections = net.forward()
5 bboxes = []
6 for i in range(detections.shape[2]):
7     confidence = detections[0, 0, i, 2]
8     if confidence > conf_threshold:
9         x1 = int(detections[0, 0, i, 3] * frameWidth)
10        y1 = int(detections[0, 0, i, 4] * frameHeight)
11        x2 = int(detections[0, 0, i, 5] * frameWidth)
12        y2 = int(detections[0, 0, i, 6] * frameHeight)
13
```

C++

```
1 #ifdef CAFFE
2     cv::Mat inputBlob = cv::dnn::blobFromImage(frameOpenCVDNN, inScaleFactor, cv::Si
3 #else
4     cv::Mat inputBlob = cv::dnn::blobFromImage(frameOpenCVDNN, inScaleFactor, cv::Si
5 #endif
6
7 net.setInput(inputBlob, "data");
8 cv::Mat detection = net.forward("detection_out");
9
10 cv::Mat detectionMat(detection.size[2], detection.size[3], CV_32F, detection.ptr<flo
11
12 for(int i = 0; i < detectionMat.rows; i++)
13 {
14     float confidence = detectionMat.at<float>(i, 2);
15
16     if(confidence > confidenceThreshold)
17     {
18         int x1 = static_cast<int>(detectionMat.at<float>(i, 3) * frameWidth);
19         int y1 = static_cast<int>(detectionMat.at<float>(i, 4) * frameHeight);
20         int x2 = static_cast<int>(detectionMat.at<float>(i, 5) * frameWidth);
21         int y2 = static_cast<int>(detectionMat.at<float>(i, 6) * frameHeight);
22
23         cv::rectangle(frameOpenCVDNN, cv::Point(x1, y1), cv::Point(x2, y2), cv::Scal
24     }
25 }
```

优点

- 1) 在这四种方法中是最准确的；
- 2) 在CPU上能够实时运行；
- 3) 适用于不同的人脸方向：上，下，左，右，侧面等。
- 4) 甚至在严重遮挡下仍能工作；
- 5) 可以检测各种尺度的人脸。

缺点

作者认为没有什么大的缺点^_^

(52CV君不敢妄提缺点，但认为不能使用NVIDIA GPU绝对是个遗憾)

3. Dlib HoG人脸检测

代码示例：

Python

```
1 hogFaceDetector = dlib.get_frontal_face_detector()
2 faceRects = hogFaceDetector(frameDlibHogSmall, 0)
3 for faceRect in faceRects:
4     x1 = faceRect.left()
5     y1 = faceRect.top()
6     x2 = faceRect.right()
7     y2 = faceRect.bottom()
```

C++

```
1 frontal_face_detector hogFaceDetector = get_frontal_face_detector();
2
3 // Convert OpenCV image format to Dlib's image format
4 cv_image<bgr_pixel> dlibIm(frameDlibHogSmall);
5
6 // Detect faces in the image
7 std::vector<dlib::rectangle> faceRects = hogFaceDetector(dlibIm);
8
9 for ( size_t i = 0; i < faceRects.size(); i++ )
10 {
11     int x1 = faceRects[i].left();
12     int y1 = faceRects[i].top();
13     int x2 = faceRects[i].right();
14     int y2 = faceRects[i].bottom();
15     cv::rectangle(frameDlibHog, Point(x1, y1), Point(x2, y2), Scalar(0,255,0), (int)(f
16 }
```

优点

- 1) CPU上最快的方法；
- 2) 适用于正面和略微非正面的人脸；
- 3) 与其他三个相比模型很小；
- 4) 在小的遮挡下仍可工作。

缺点

- 1) 不能检测小脸，因为它训练数据的最小人脸尺寸为80×80，但是用户可以用较小尺寸的人脸数据自己训练检测器；
- 2) 边界框通常排除前额的一部分甚至下巴的一部分；
- 3) 在严重遮挡下不能很好地工作；
- 4) 不适用于侧面和极端非正面，如俯视或仰视。

4. Dlib CNN人脸检测

算法来自论文《Max-Margin Object Detection》

(<https://arxiv.org/abs/1502.00046>) 。

代码示例：

Python

```
1 dnnFaceDetector = dlib.cnn_face_detection_model_v1("./mmod_human_face_detector.dat")
2 faceRects = dnnFaceDetector(frameDlibHogSmall, 0)
3 for faceRect in faceRects:
4     x1 = faceRect.rect.left()
5     y1 = faceRect.rect.top()
6     x2 = faceRect.rect.right()
7     y2 = faceRect.rect.bottom()
```

C++

```
1 String mmodModelPath = "./mmod_human_face_detector.dat";
2 net_type mmodFaceDetector;
3 deserialize(mmodModelPath) >> mmodFaceDetector;
4
5 // Convert OpenCV image format to Dlib's image format
6 cv_image<bgr_pixel> dlibIm(frameDlibMmodSmall);
7 matrix<rgb_pixel> dlibMatrix;
8 assign_image(dlibMatrix, dlibIm);
9
10 // Detect faces in the image
11 std::vector<dlib::mmod_rect> faceRects = mmodFaceDetector(dlibMatrix);
12
13 for ( size_t i = 0; i < faceRects.size(); i++ )
14 {
15     int x1 = faceRects[i].rect.left();
16     int y1 = faceRects[i].rect.top();
17     int x2 = faceRects[i].rect.right();
18     int y2 = faceRects[i].rect.bottom();
19     cv::rectangle(frameDlibMmod, Point(x1, y1), Point(x2, y2), Scalar(0,255,0), (int)(
20 }
```

优点

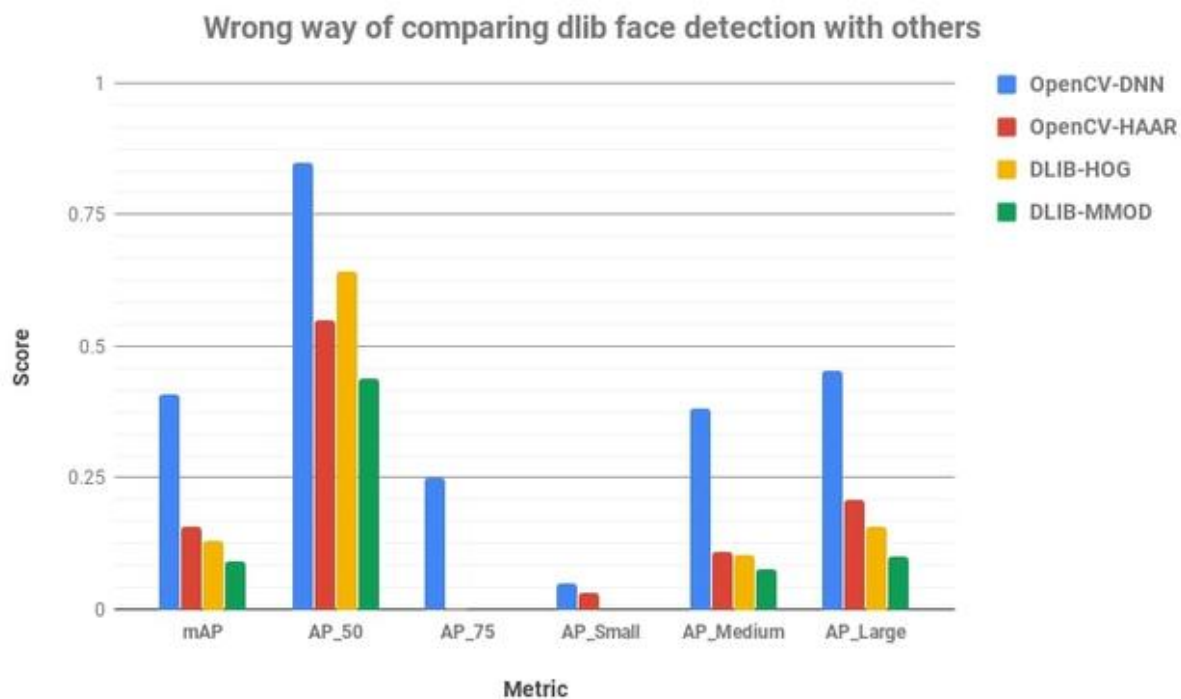
- 1) 适用于不同的人脸方向；
- 2) 对遮挡鲁棒；
- 3) 在GPU上工作得非常快；
- 4) 非常简单的训练过程。

缺点

- 1) CPU速度很慢；
- 2) 不能检测小脸，因为它训练数据的最小人脸尺寸为80×80，但是用户可以用较小尺寸的人脸数据自己训练检测器；
- 3) 人脸包围框甚至小于DLib HoG人脸检测器。

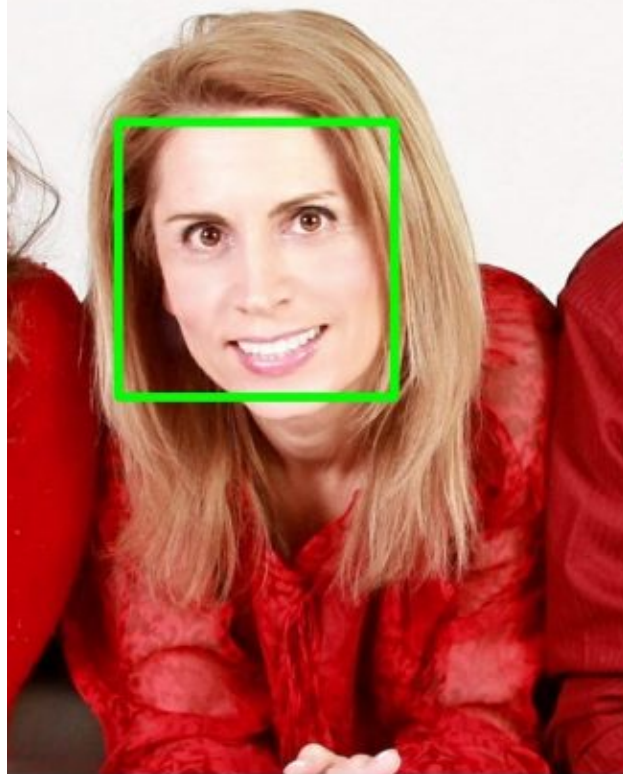
5. 四种方法精度比较

作者在FDDB数据库中测评了四种人脸检测算法实现的精度，结果如下：

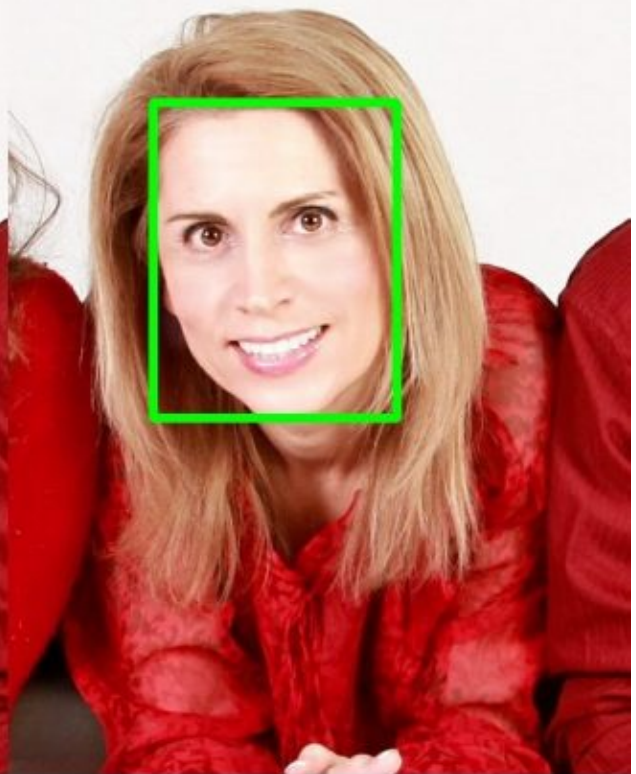


可以看到Dlib的两种方法效果都不怎么好，作者发现原来Dlib训练使用的数据集的人脸包围框较小，导致按照FDDB的评价标准不公平。

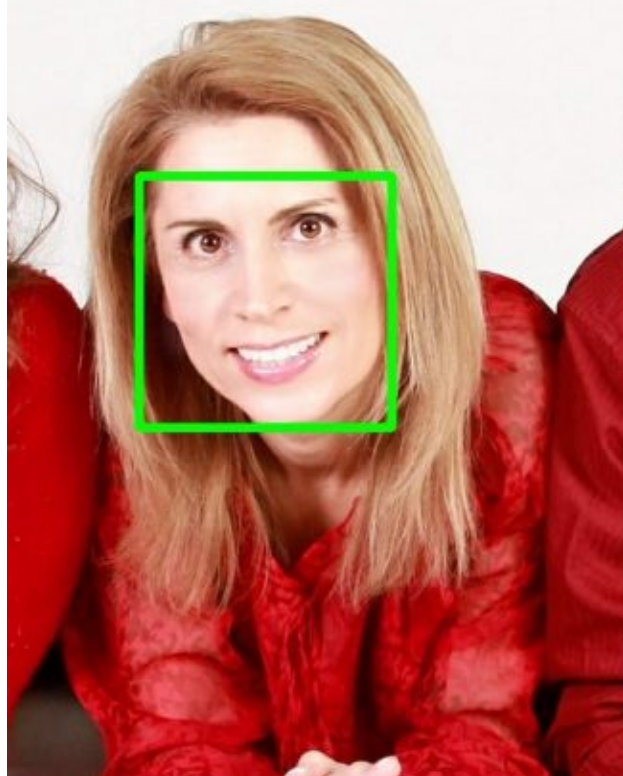
OpenCV Haar



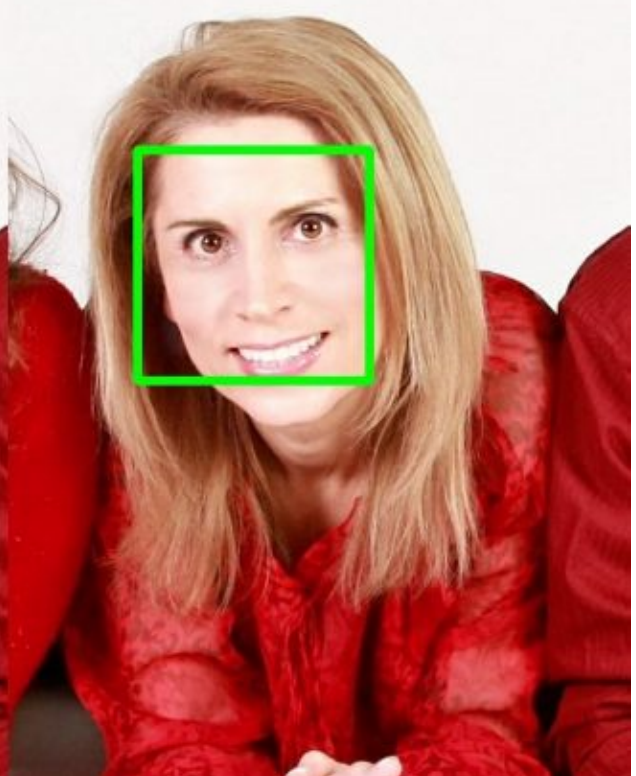
OpenCV DNN



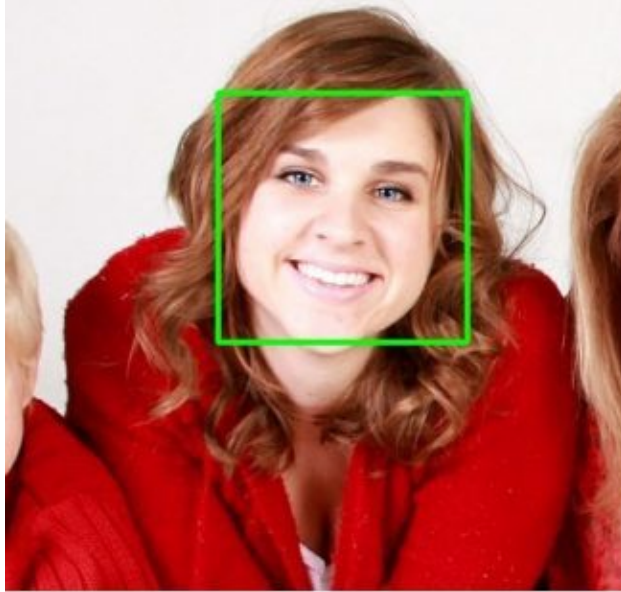
DLIB HoG



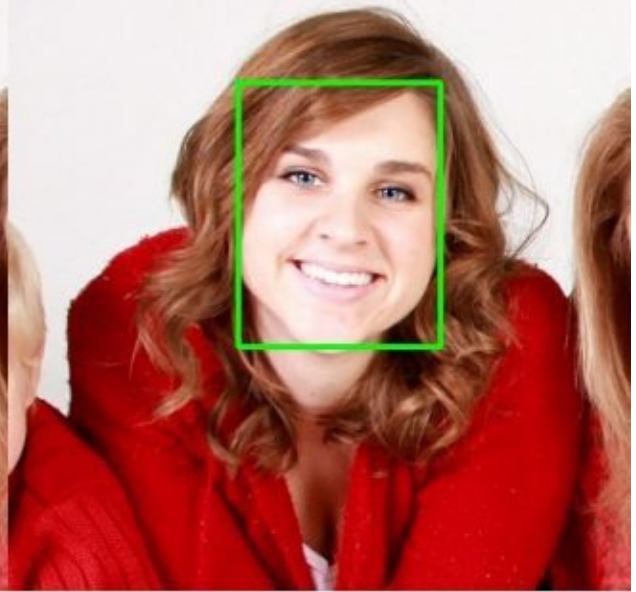
DLIB MMOD



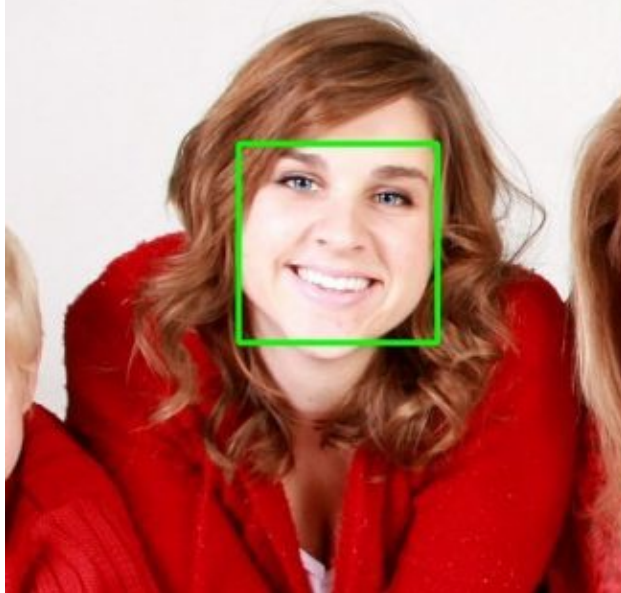
OpenCV Haar



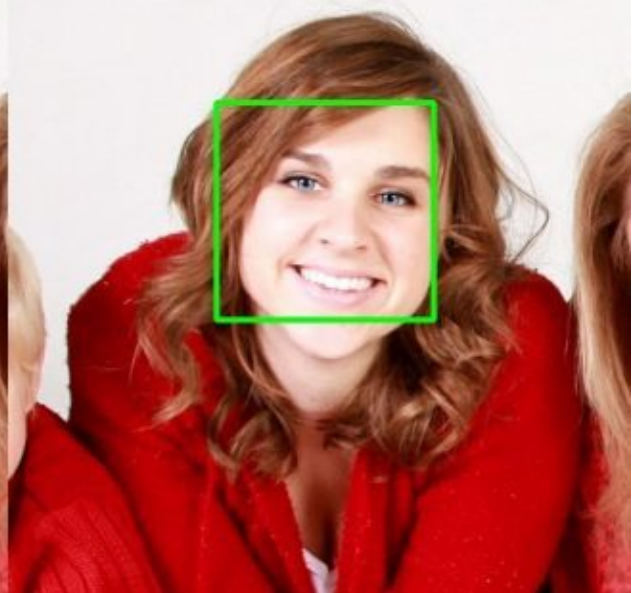
OpenCV DNN

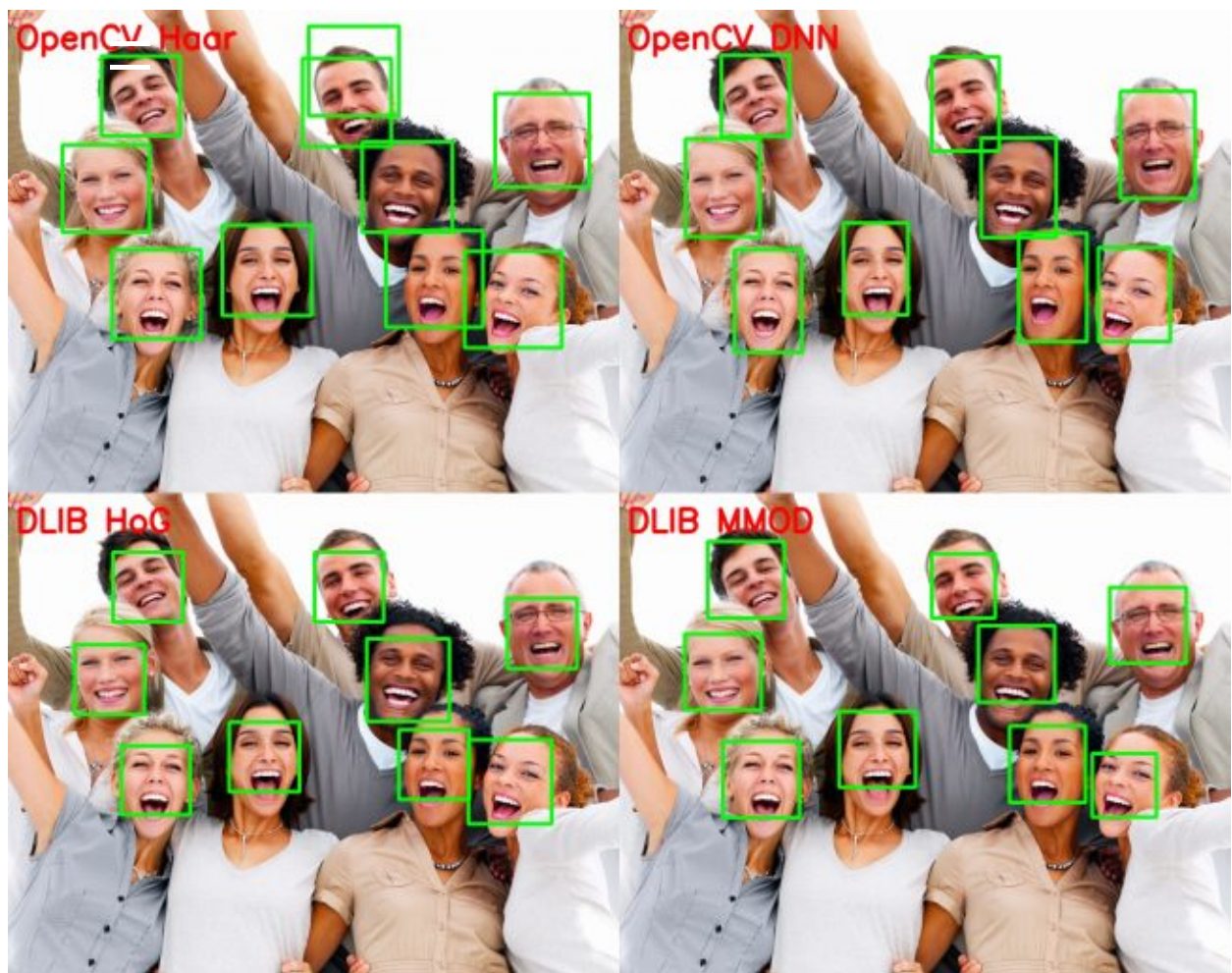


DLIB HoG



DLIB MMOD





另外，Dlib无法检测小脸也拉低了分数。

6. 速度比较

软硬件环境：

Processor : Intel Core i7 6850K – 6 Core

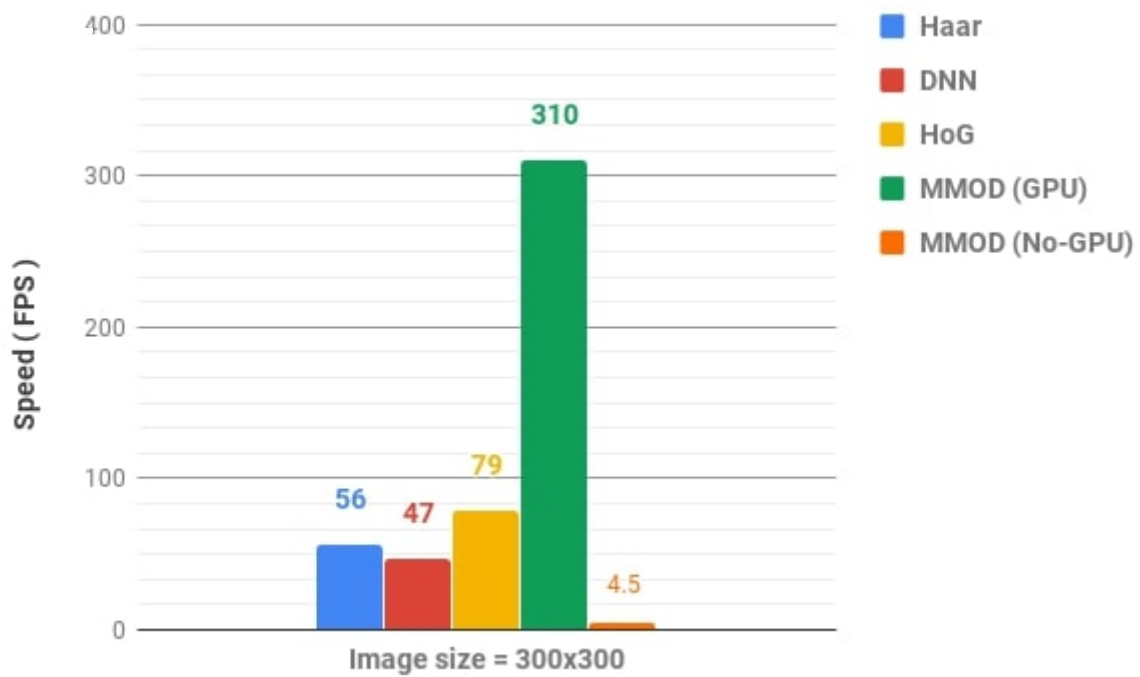
RAM : 32 GB

GPU : NVIDIA GTX 1080 Ti with 11 GB RAM

OS : Linux 16.04 LTS

Programming Language : Python

图像大小300*300，测试结果如下：



可以看到除了MMOD 其他方法都达到实时，而MMOD方法的GPU计算是最快的。

7. 分情况检测结果示例

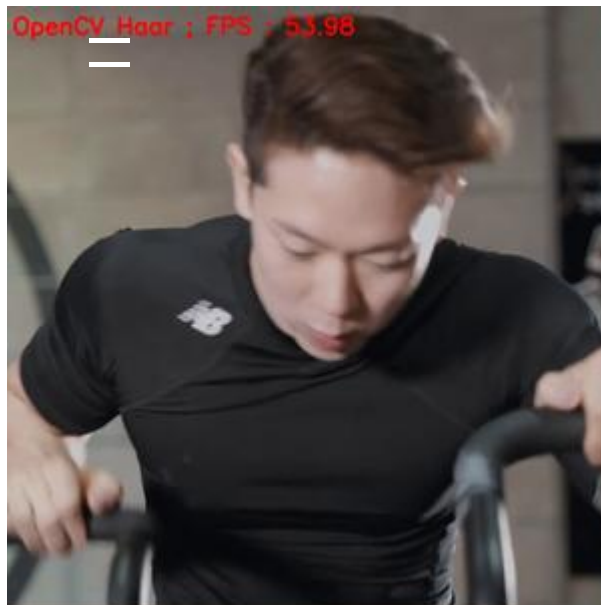
7.1跨尺度检测



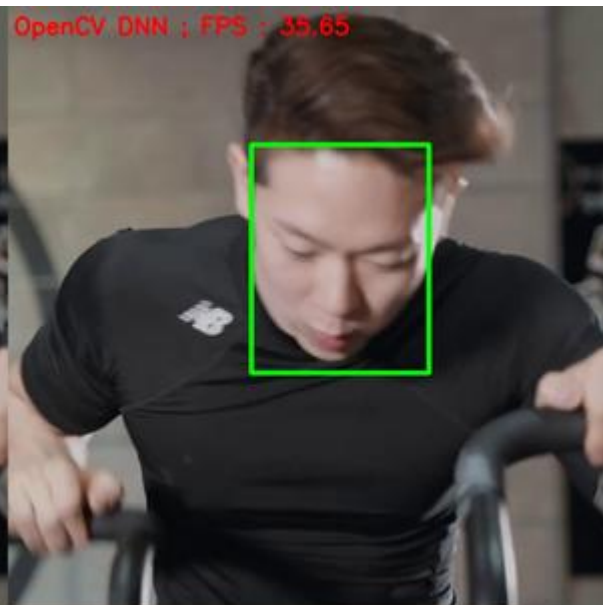
此图片来自微信公众平台
未经允许不可引用

7.2 非正面人脸

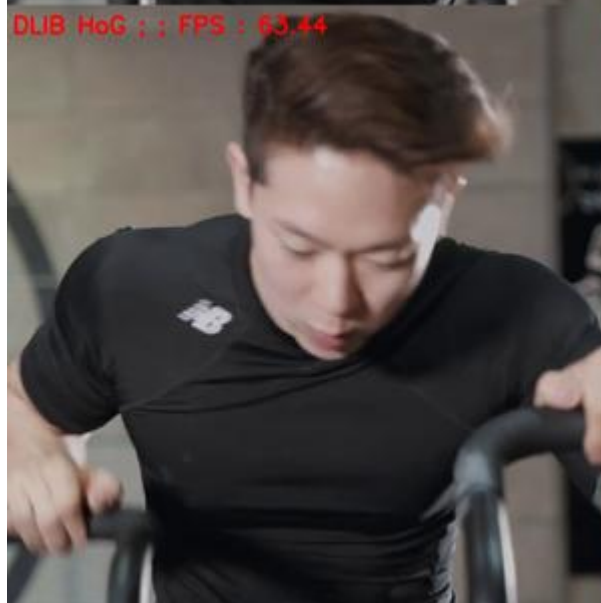
OpenCV Haar ; FPS : 53.98



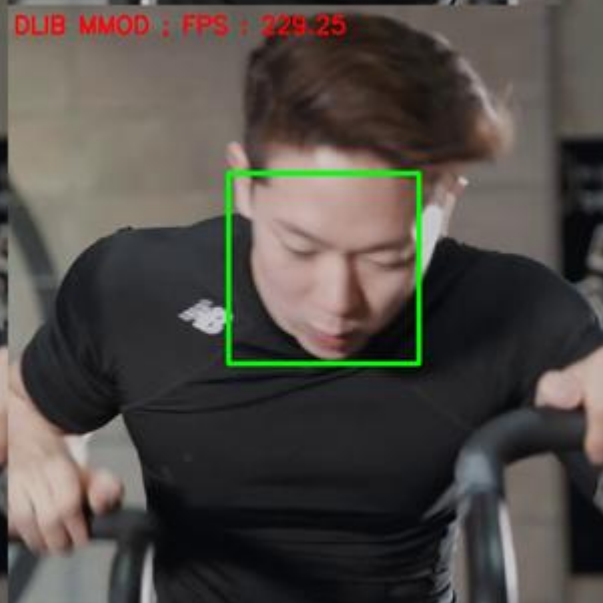
OpenCV DNN ; FPS : 35.65



DLIB HoG ; FPS : 63.44



DLIB MMOD ; FPS : 329.25



OpenCV Haar ; FPS : 48.74



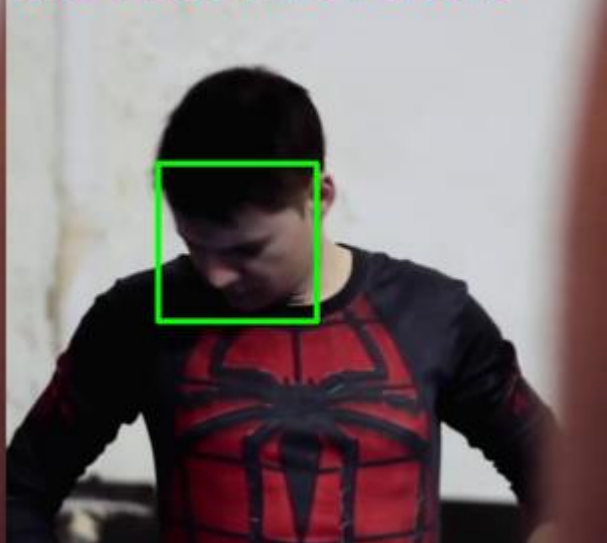
OpenCV DNN ; FPS : 34.72



DLIB HoG ; ; FPS : 68.26



DLIB MMOD ; FPS : 266.45



OpenCV Haar ; FPS : 53.23



OpenCV DNN ; FPS : 36.04



DLIB HoG ; ; FPS : 68.12



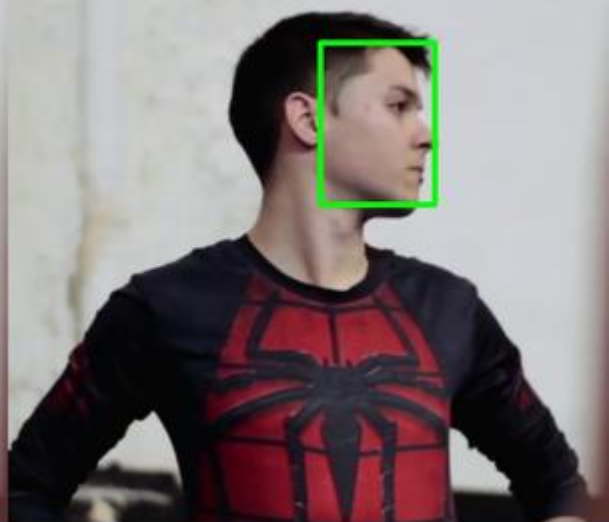
DLIB MMOD ; FPS : 257.50



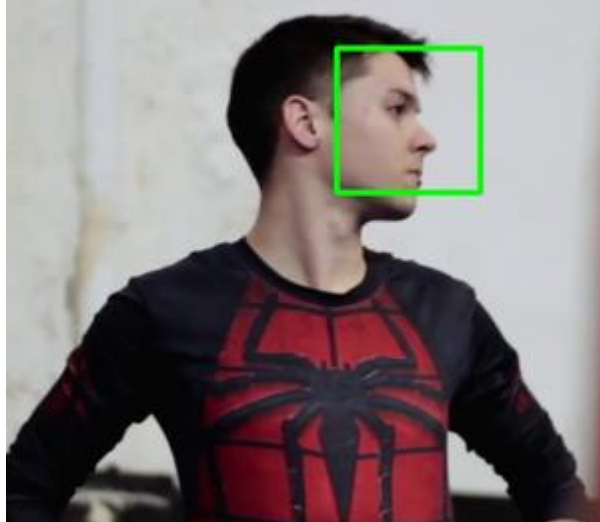
OpenCV Haar ; FPS : 53.49



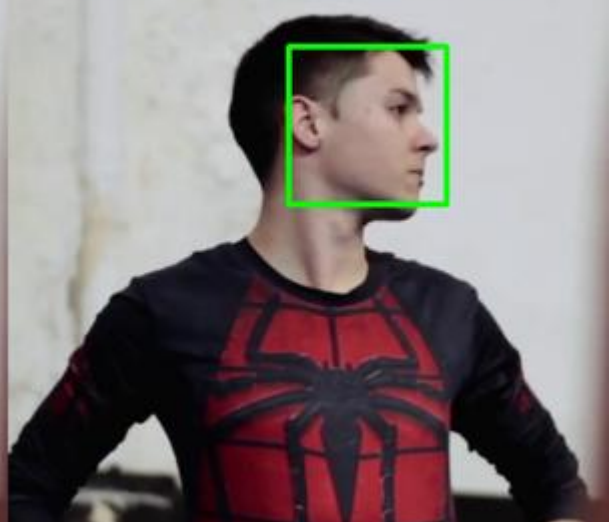
OpenCV DNN ; FPS : 37.30



DLIB HoG ; FPS : 69.15



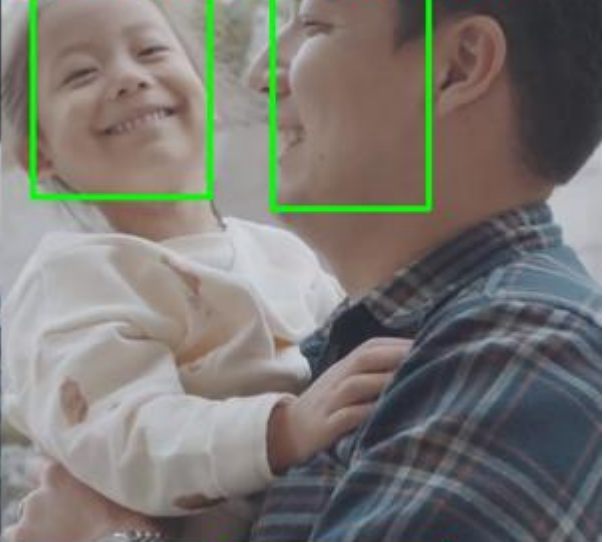
DLIB MMOD ; FPS : 255.49



OpenCV Haar ; FPS : 54.54



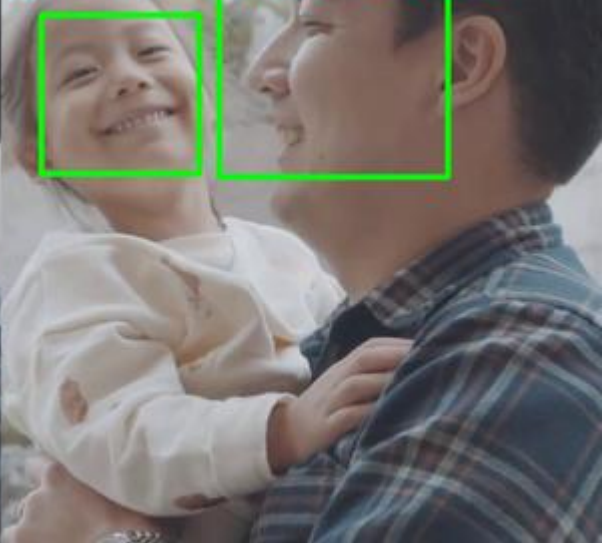
OpenCV DNN ; FPS : 37.99

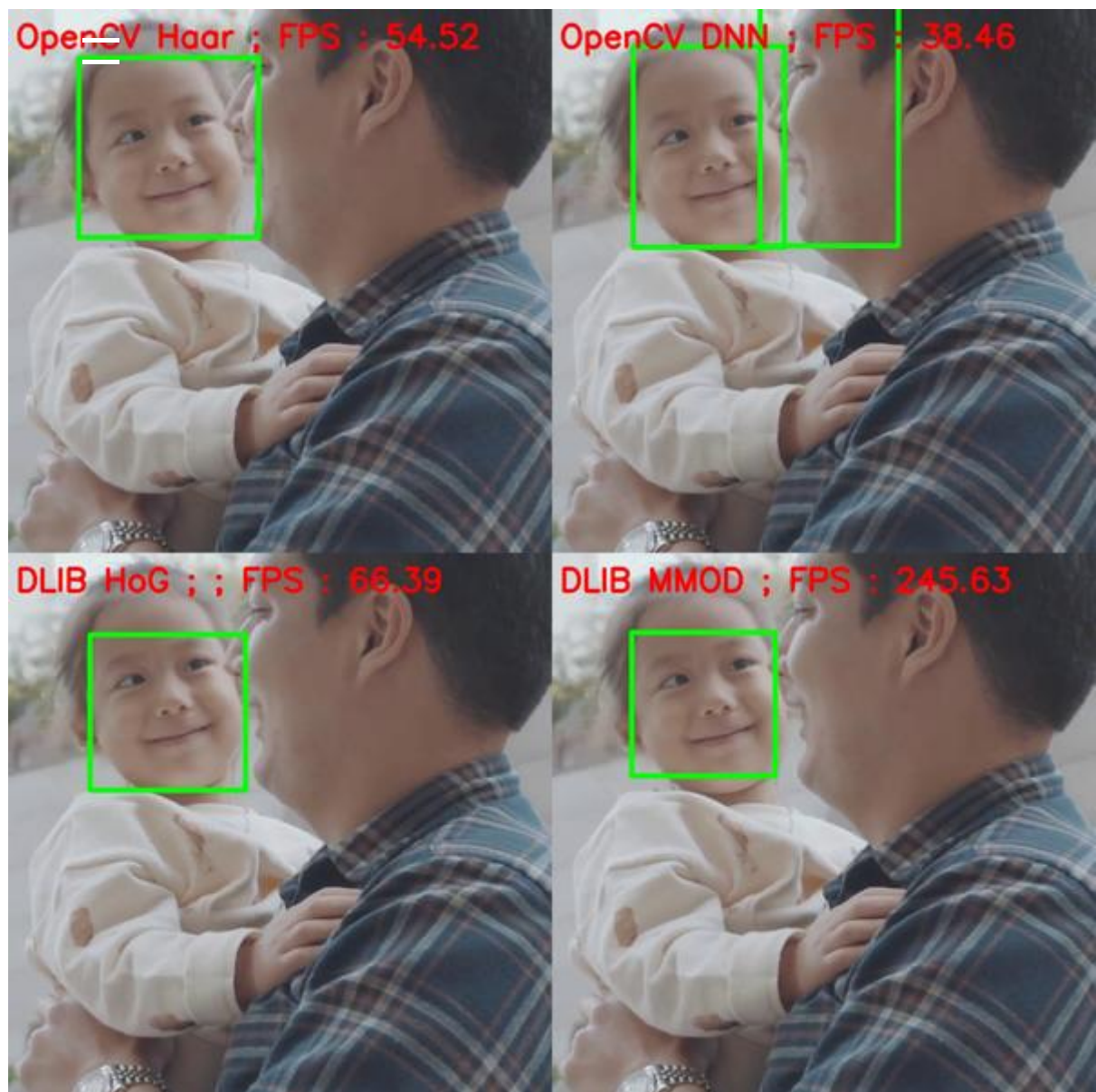


DLIB HoG ; ; FPS : 66.35



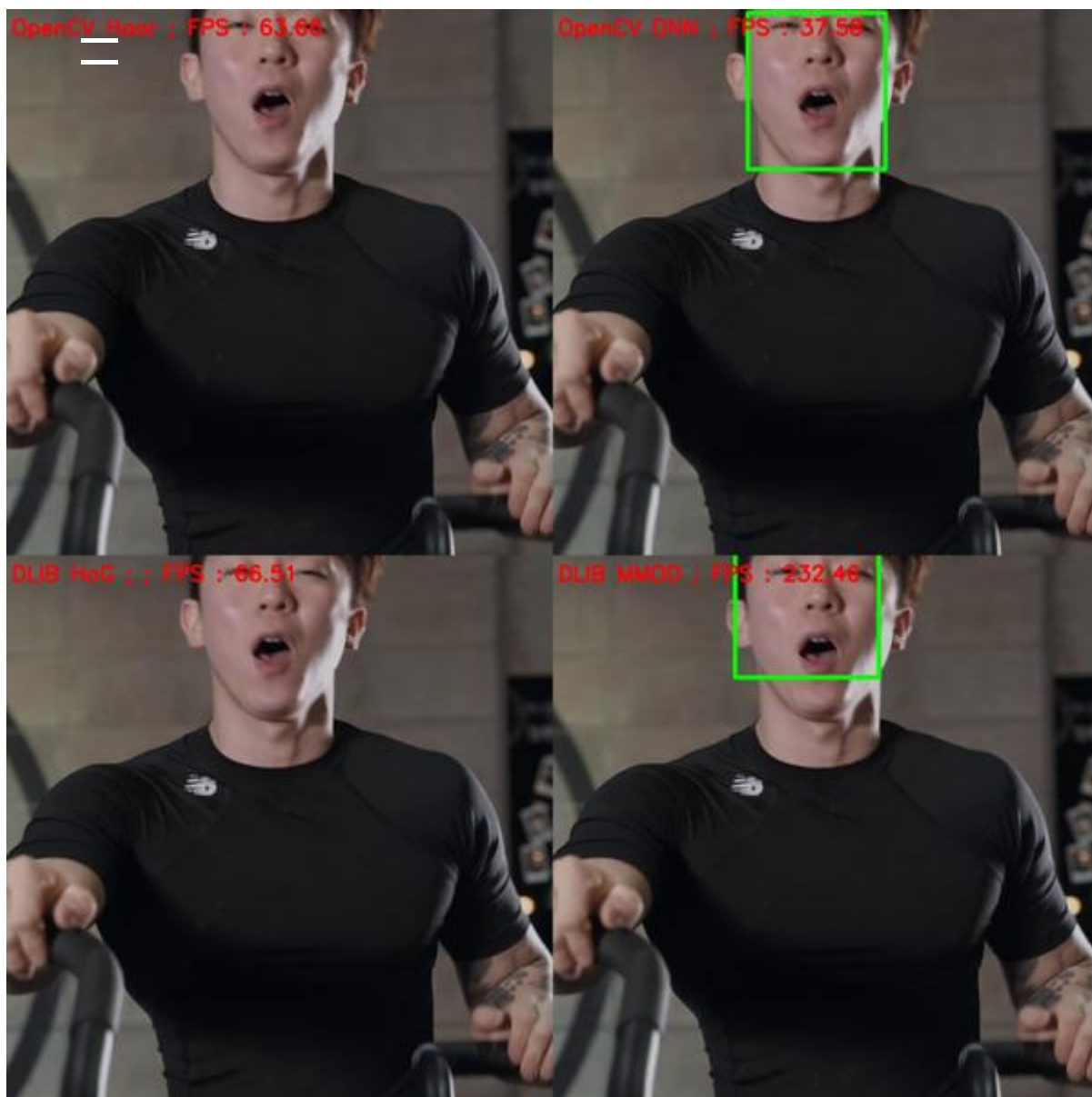
DLIB MMOD ; FPS : 245.39





7.3 遮挡





8 总结推荐

如何在应用中选择人脸检测算法呢？作者认为应该首先尝试OpenCV DNN方法与Dlib HOG方法，然后再做决定。

一般情况

在大多数应用程序中，我们无法知道图像中人脸尺寸的大小。因此，最好使用OpenCV-DNN方法，因为它非常快速且非常准确，即使对于小尺寸的人脸也是如此。它还可以检测各种角度的人脸。所以OpenCV-DNN是首选。

中到大尺寸的图像

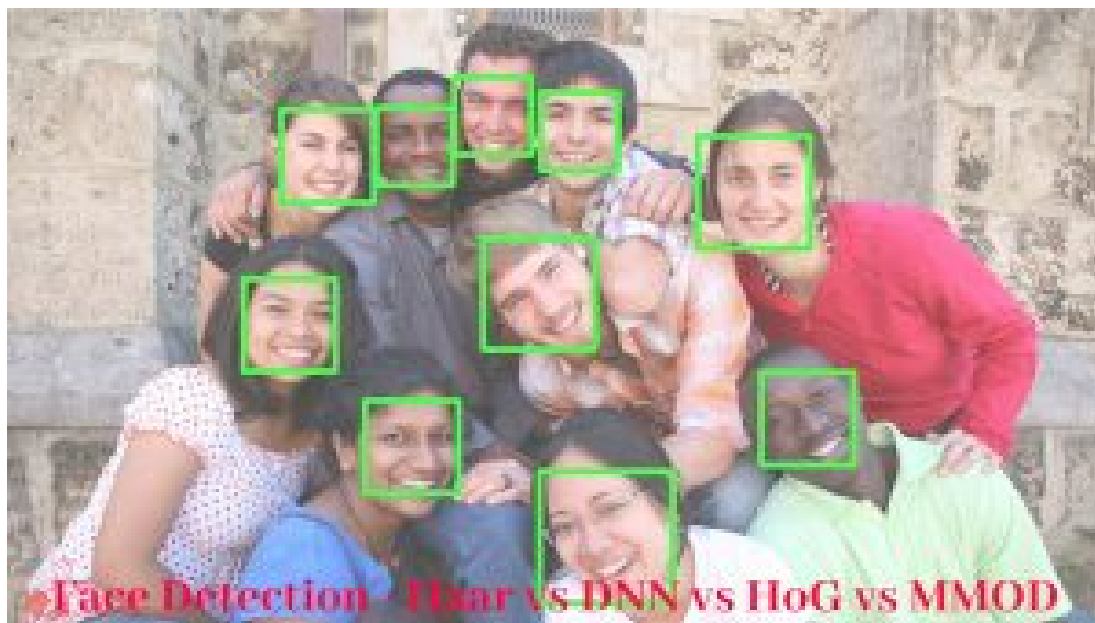
Dlib HOG是CPU上最快的方法。但它不能检测到小脸（ $<70 \times 70$ ）。因此，如果知道程序不会处理非常小的人脸（例如自拍照），那么基于HOG的人脸检测器是更好的选择。

此外，如果你可以使用GPU（NVIDIA家的），那么MMOD人脸检测器是最好的选择，因为它在GPU上非常快，并且还提供各种角度的检测。

高分辨率图像

由于在高分辨率图像中，这些算法的速度都会很慢，而如果缩小图像尺寸，HOG/MMOD可能会失败，同时OpenCV-DNN却可以检测小脸，所以对于高分辨率图像推荐缩小图像再使用OpenCV-DNN的方法。

原文链接：



Face Detection – OpenCV, Dlib and Deep Learning (C++ / Python)

In this tutorial, we will discuss the various Face Detection methods in OpenCV and Dlib and compare the methods quantitatively. We will share code in C++ and Python for the following Face Detectors : Haar Cascade Face Detector in OpenCV Deep Learning based Face Detector in OpenCV HoG Face Detector in Dlib Deep Learning based ... Continue reading



Learn OpenCV

23

代码数据下载：

在“我爱计算机视觉”微信公众号对话界面回复“人脸检测比较”，即可收到该文代码、模型与使用数据百度云下载地址。

人脸检测开源技术众多，除了OpenCV和Dlib,你还有什么推荐吗？欢迎留言~

(欢迎关注“我爱计算机视觉”公众号，一个有价值有深度的公众号~)



转载请注明：《[OpenCV vs Dlib 人脸检测比较分析](#)》

分类

人脸检测

开源软件

计算机视觉教程

上一篇文章

英特尔重磅开源OpenVINO™！附送的预训练模型是最大亮点

下一篇文章

ACM MM2018 Best Paper 被华人包揽

或许您还喜欢这些文章

人脸检测



Fig. 3. On the left, the original image of WIDER-FACE dataset. On the center and on the right, modified crops in which the face size matches our targets.



惊呆了！速度高达1500ofps的人脸检测算法！

人脸检测



深度学习 目标检测 算法大全列表 awesome object detection

人脸检测



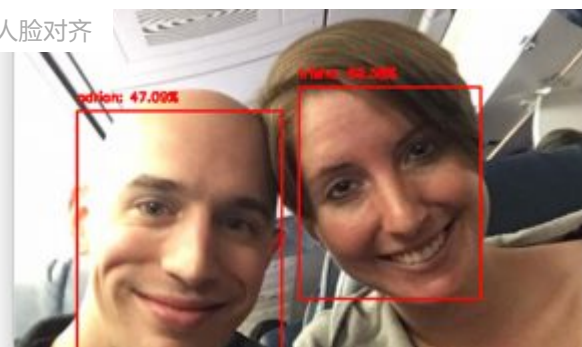
人脸检测-SSH: Single Stage Headless Face Detector

交通标志检测



英特尔重磅开源OpenVINO™！附送的预训练模型是最大亮点

人脸对齐



OpenCV深度学习人脸识别示例——看大佬如何秀恩爱

发表评论

评论

姓名 *

电子邮件 *

站点

发表评论

全站搜索

搜索...

搜索



📌 群号：928997753

欢迎关注微信公众号



最新文章列表

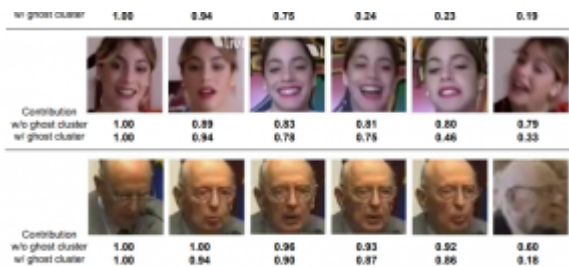
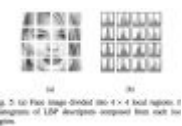


Fig. 4: Effect of ghost clusters. Each row shows shows 6 images from a template in the IJB-B dataset. The contribution (relative to the max) of each image

DeepMind&VGG提出基于集合的人脸识别算法GhostVLAD,精度远超IJB-B数据集

52cv.net 2018年11月9日 0

(欢迎关注“我爱计算机视觉”公众号，一个有价值有深度的公众号~) 在人脸识别应用中，很多场景能够获取某一个体的...



人脸识别的前世今生：从人工特征的百花齐放到深度学习的一统江湖

2018年11月5日



英特尔重磅开源OpenVINO™！附送的预训练模型是最大亮点

2018年10月20日



重磅！2018人脸识别研究报告（全文下载）

2018年10月13日

分类导航

- › 3D视觉
- › CV招聘
- › Dehazing
- › Domain Adaptation
- › image captioning
- › Image Inpaint
- › KDD CUP
- › Paper Daily
- › SLAM
- › VQA
- › 三维重建
- › 交通标志检测
- › 人群计数
- › 人脸对齐
- › 人脸检测
- › 人脸识别
- › 光流法
- › 关于本站
- › 医学图像处理与分析
- › 去雨derain
- › 图像匹配
- › 图像处理
- › 图像检索
- › 图像融合
- › 姿态估计
- › 开源软件
- › 掌纹识别

- › 数据集
- › 文字识别
- › 文本检测与识别
- › 显著性检测
- › 期刊会议资讯
- › 机器学习
- › 标注工具
- › 深度估计
- › 深度学习
- › 深度神经网络压缩
- › 点云处理
- › 生成对抗网络
- › 目标检测
- › 目标跟踪
- › 算法详解
- › 行人计数
- › 行人重识别
- › 视线估计
- › 视频分析
- › 视频预测
- › 计算机视觉教程
- › 计算机视觉竞赛
- › 语义分割
- › 语义抠图
- › 超分辨率重建
- › 车牌识别
- › 迁移学习
- › 难分样本挖掘
- › 集成学习

