OpenCV vs Dlib 人脸检测比较分析

原创： 52CV君

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：

1. OpenCV Haar Cascade人脸检测

算法无需赘言。

代码示例：

Python

```python
faceCascade = cv2.CascadeClassifier('./haarcascade_frontalface_default.xml')
faces = faceCascade.detectMultiScale(frameGray)
for face in faces:
    x1, y1, w, h = face
    x2 = x1 + w
    y2 = y1 + h
```

C++

```cpp
faceCascadePath = "./haarcascade_frontalface_default.xml";
faceCascade.load( faceCascadePath )
std::vector<Rect> faces;
faceCascade.detectMultiScale(frameGray, faces);

for ( size_t i = 0; i < faces.size(); i++ )
{
  int x1 = faces[i].x;
  int y1 = faces[i].y;
  int x2 = faces[i].x + faces[i].width;
  int y2 = faces[i].y + faces[i].height;
}
```

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

```python
DNN = "TF"
if DNN == "CAFFE":
    modelFile = "res10_300x300_ssd_iter_140000_fp16.caffemodel"
    configFile = "deploy.prototxt"
    net = cv2.dnn.readNetFromCaffe(configFile, modelFile)
else:
    modelFile = "opencv_face_detector_uint8.pb"
    configFile = "opencv_face_detector.pbtxt"
    net = cv2.dnn.readNetFromTensorflow(modelFile, configFile)
```

C++

```cpp
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

```python
blob = cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], False

net.setInput(blob)
detections = net.forward()
bboxes = []
for i in range(detections.shape[2]):
    confidence = detections[0, 0, i, 2]
    if confidence > conf_threshold:
        x1 = int(detections[0, 0, i, 3] * frameWidth)
        y1 = int(detections[0, 0, i, 4] * frameHeight)
        x2 = int(detections[0, 0, i, 5] * frameWidth)
        y2 = int(detections[0, 0, i, 6] * frameHeight)

```

C++

```cpp
#ifdef CAFFE
    cv::Mat inputBlob = cv::dnn::blobFromImage(frameOpenCVDNN, inScaleFactor, cv::Si
#else
    cv::Mat inputBlob = cv::dnn::blobFromImage(frameOpenCVDNN, inScaleFactor, cv::Si
#endif

net.setInput(inputBlob, "data");
cv::Mat detection = net.forward("detection_out");

cv::Mat detectionMat(detection.size[2], detection.size[3], CV_32F, detection.ptr<flo

for(int i = 0; i < detectionMat.rows; i++)
{
    float confidence = detectionMat.at<float>(i, 2);

    if(confidence > confidenceThreshold)
    {
        int x1 = static_cast<int>(detectionMat.at<float>(i, 3) * frameWidth);
        int y1 = static_cast<int>(detectionMat.at<float>(i, 4) * frameHeight);
        int x2 = static_cast<int>(detectionMat.at<float>(i, 5) * frameWidth);
        int y2 = static_cast<int>(detectionMat.at<float>(i, 6) * frameHeight);

        cv::rectangle(frameOpenCVDNN, cv::Point(x1, y1), cv::Point(x2, y2), cv::Scal
    }
}
```


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

```python
hogFaceDetector = dlib.get_frontal_face_detector()
faceRects = hogFaceDetector(frameDlibHogSmall, 0)
for faceRect in faceRects:
    x1 = faceRect.left()
    y1 = faceRect.top()
    x2 = faceRect.right()
    y2 = faceRect.bottom()
```

C++

```cpp
frontal_face_detector hogFaceDetector = get_frontal_face_detector();

// Convert OpenCV image format to Dlib's image format
cv_image<bgr_pixel> dlibIm(frameDlibHogSmall);

// Detect faces in the image
std::vector<dlib::rectangle> faceRects = hogFaceDetector(dlibIm);

for ( size_t i = 0; i < faceRects.size(); i++ )
{
  int x1 = faceRects[i].left();
  int y1 = faceRects[i].top();
  int x2 = faceRects[i].right();
  int y2 = faceRects[i].bottom();
  cv::rectangle(frameDlibHog, Point(x1, y1), Point(x2, y2), Scalar(0,255,0), (int)(f
}
```


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

```python
dnnFaceDetector = dlib.cnn_face_detection_model_v1("./mmod_human_face_detector.dat")
faceRects = dnnFaceDetector(frameDlibHogSmall, 0)
for faceRect in faceRects:
    x1 = faceRect.rect.left()
    y1 = faceRect.rect.top()
    x2 = faceRect.rect.right()
    y2 = faceRect.rect.bottom()
```

C++

```cpp
String mmodModelPath = "./mmod_human_face_detector.dat";
net_type mmodFaceDetector;
deserialize(mmodModelPath) >> mmodFaceDetector;

// Convert OpenCV image format to Dlib's image format
cv_image<bgr_pixel> dlibIm(frameDlibMmodSmall);
matrix<rgb_pixel> dlibMatrix;
assign_image(dlibMatrix, dlibIm);

// Detect faces in the image
std::vector<dlib::mmod_rect> faceRects = mmodFaceDetector(dlibMatrix);

for ( size_t i = 0; i < faceRects.size(); i++ )
{
  int x1 = faceRects[i].rect.left();
  int y1 = faceRects[i].rect.top();
  int x2 = faceRects[i].rect.right();
  int y2 = faceRects[i].rect.bottom();
  cv::rectangle(frameDlibMmod, Point(x1, y1), Point(x2, y2), Scalar(0,255,0), (int)(
}
```


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

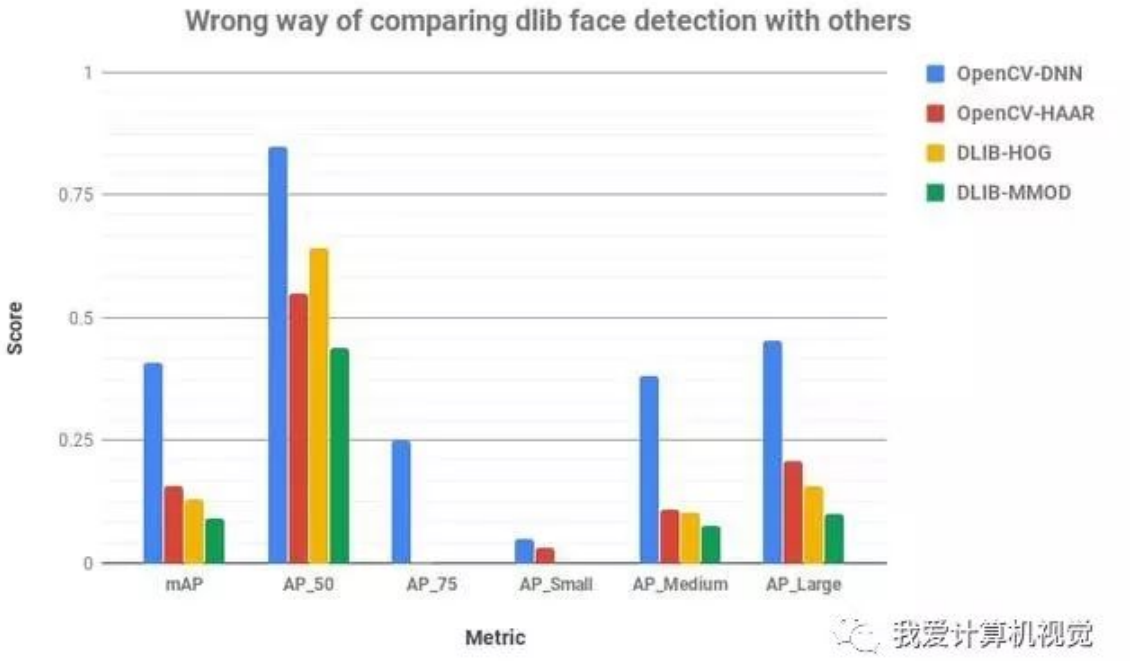
1）CPU速度很慢；

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；

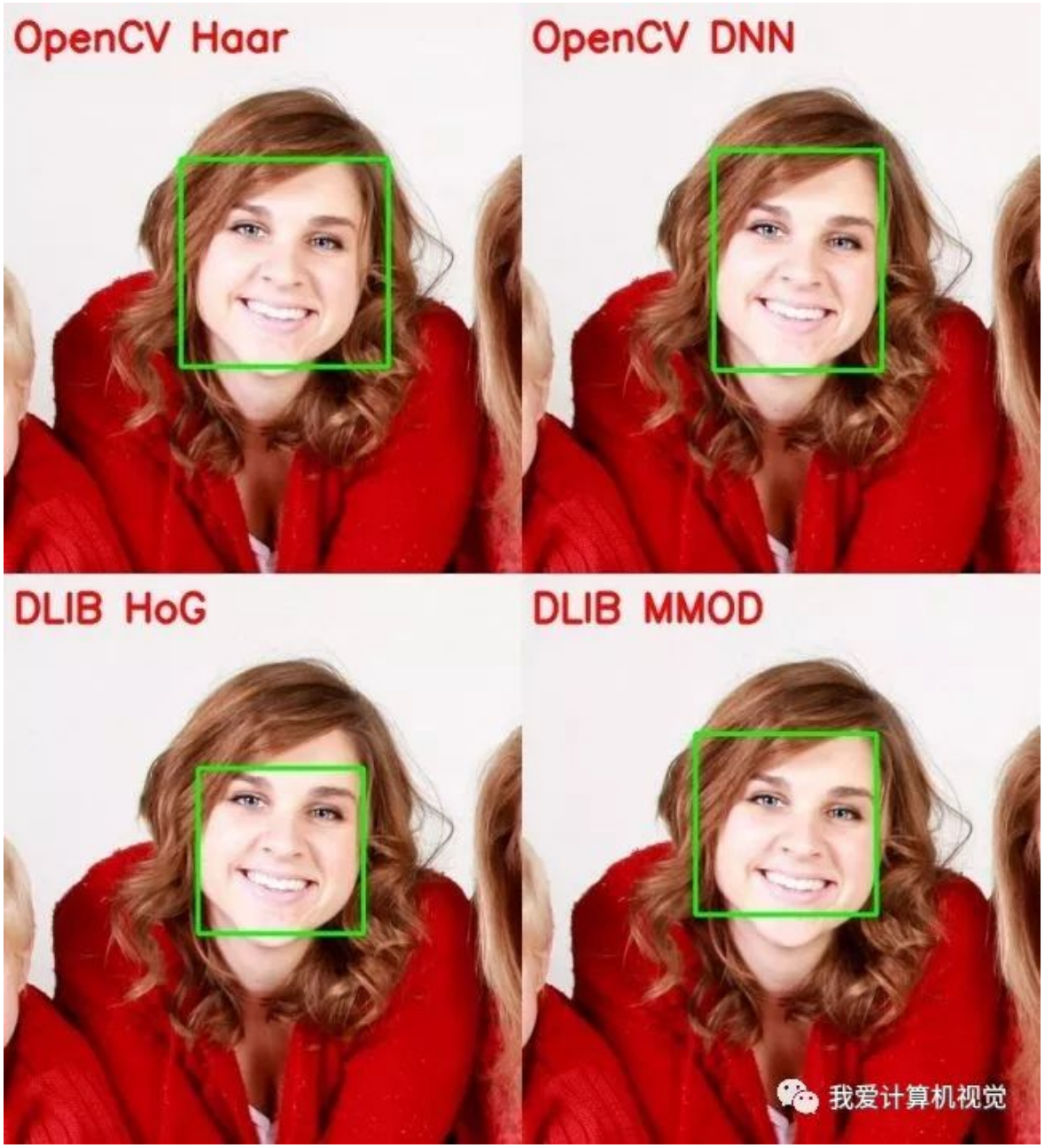
3）人脸包围框甚至小于DLib HoG人脸检测器。

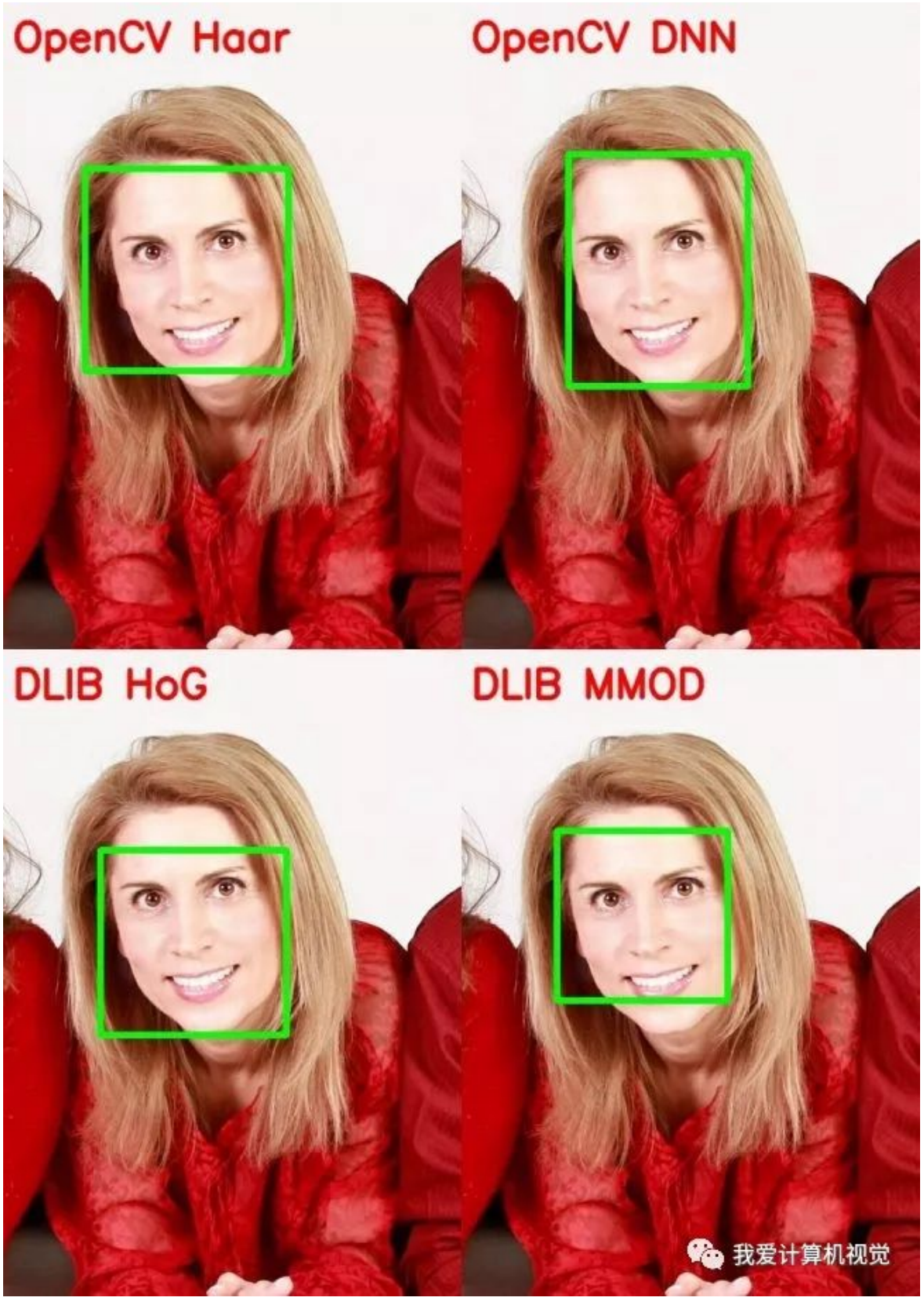
5. 四种方法精度比较

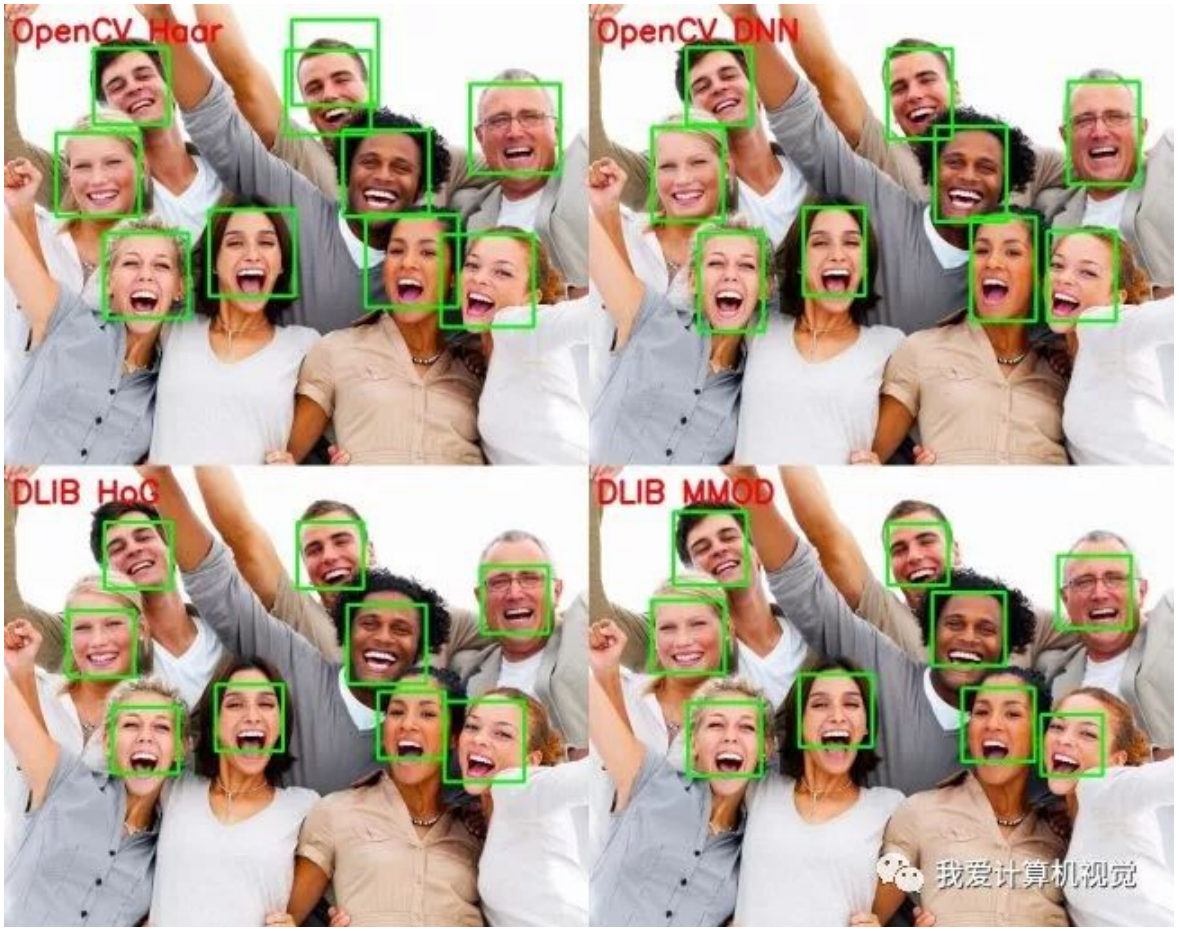
作者在FDDB数据库中测评了四种人脸检测算法实现的精度，结果如下：



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
我爱计算机视觉


OpenCV Haar
OpenCV DNN
DLIB HoG
DLIB MMOD
我爱计算机视觉



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

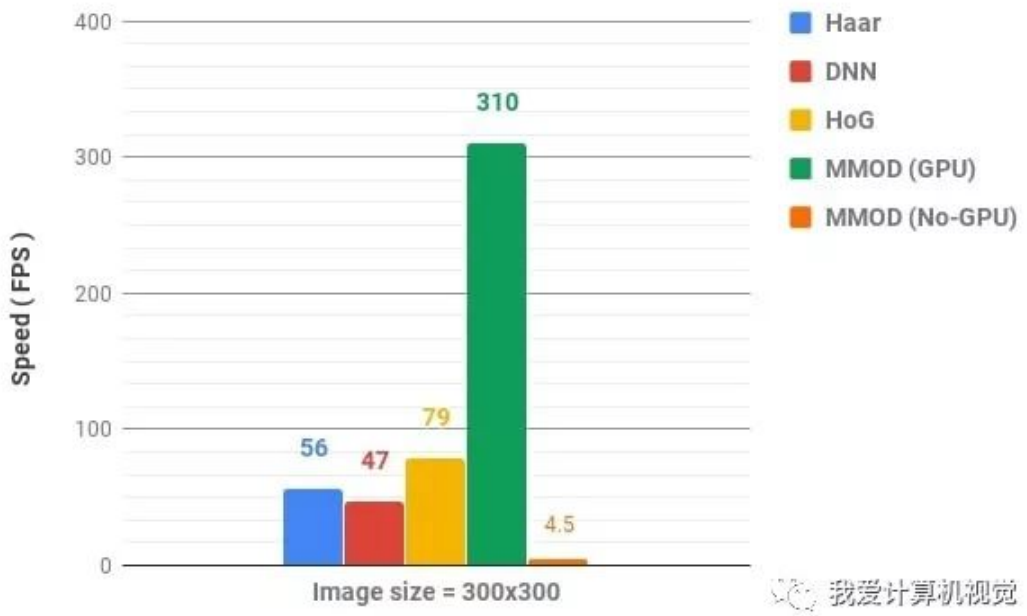
RAM : 32 GB

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：



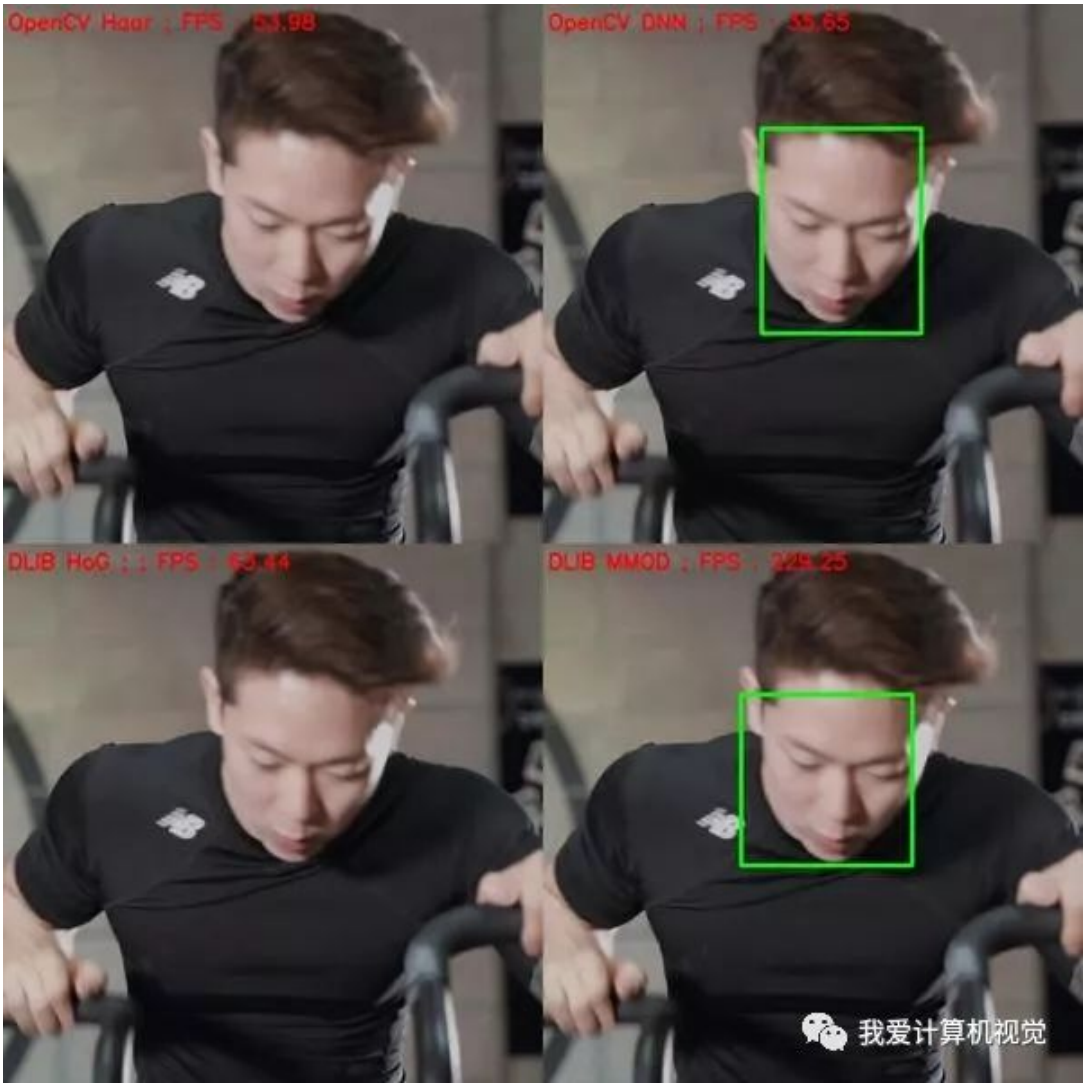
可以看到除了MMOD 其他方法都达到实时，而MMOD方法的GPU计算是最快的。

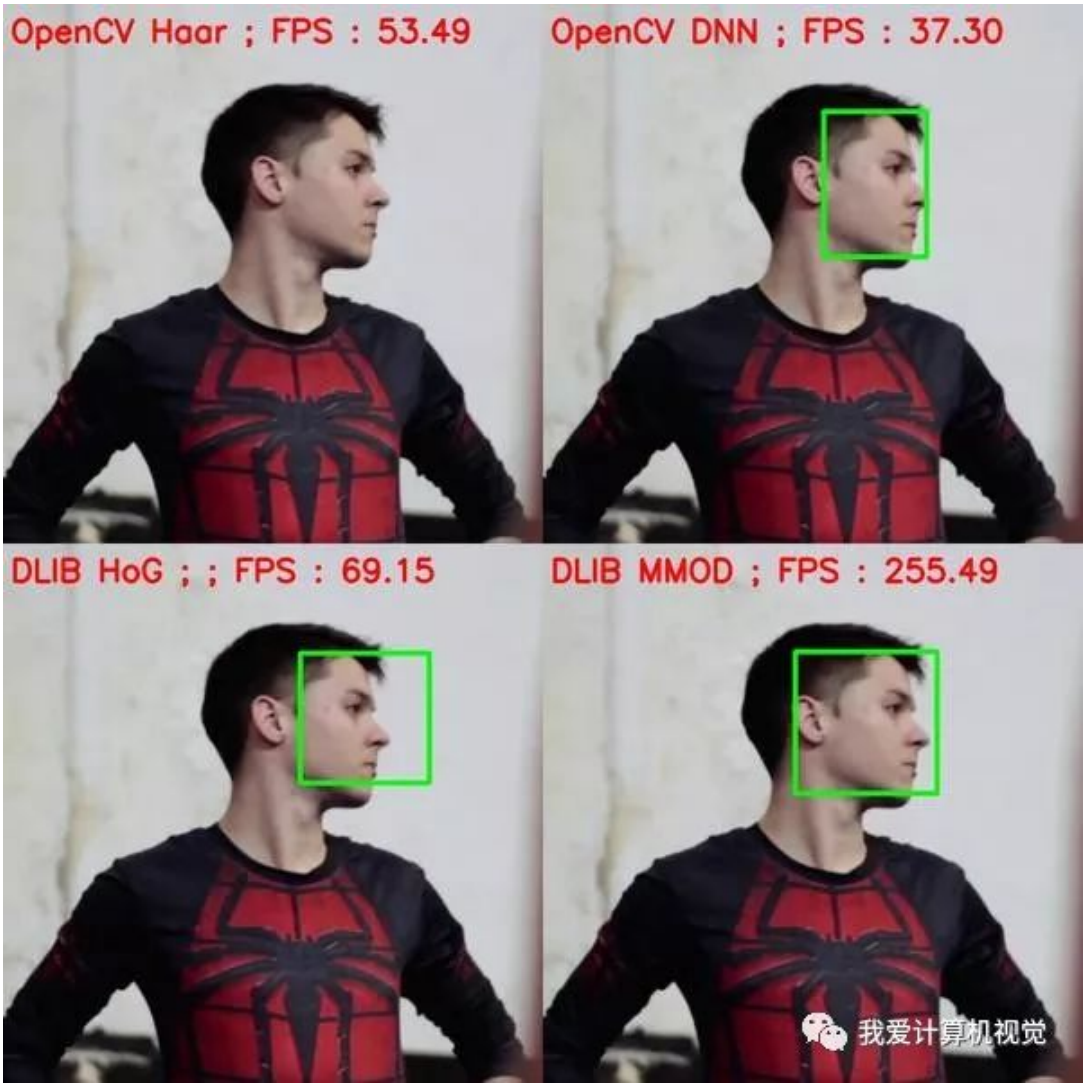
7. 分情况检测结果示例

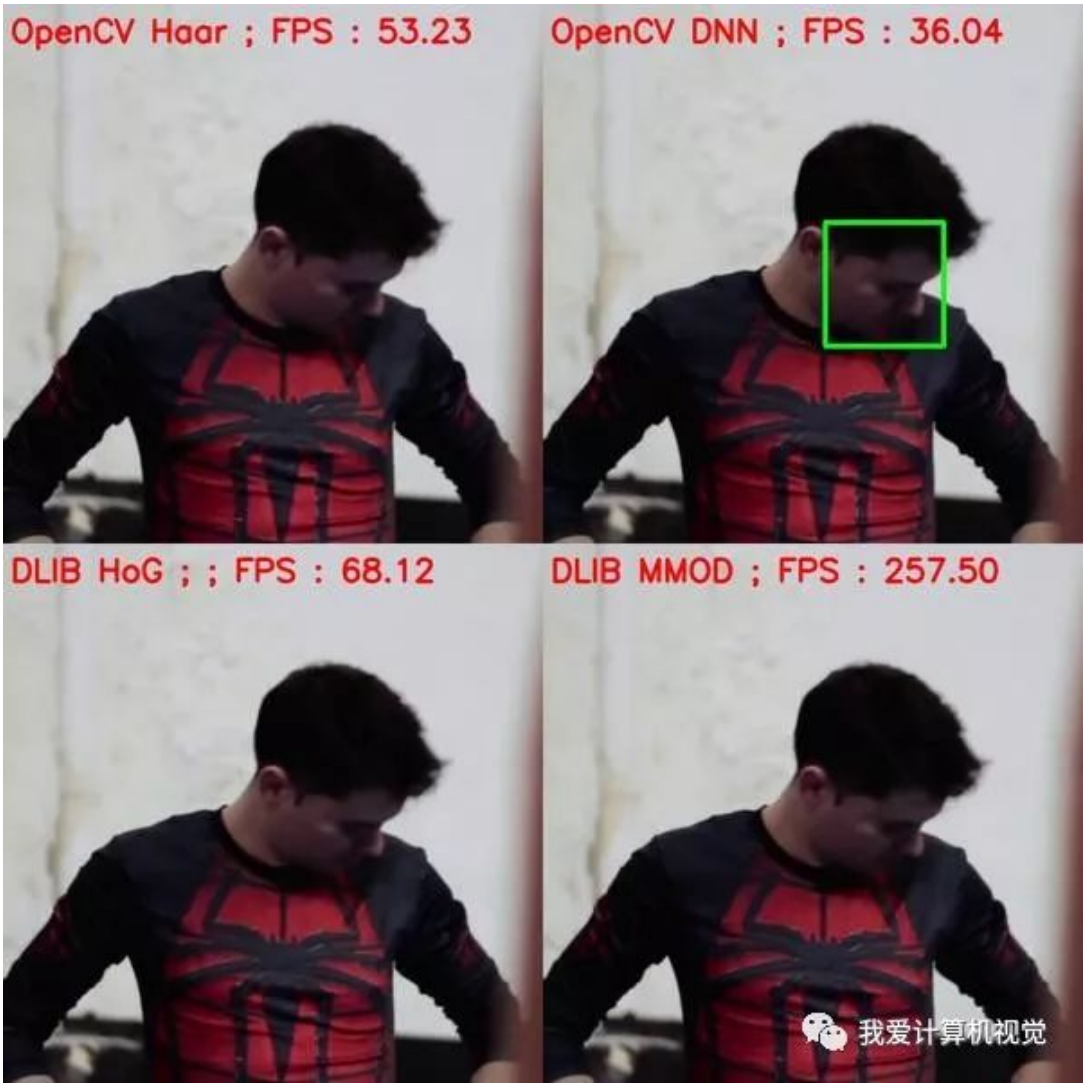
7.1跨尺度检测

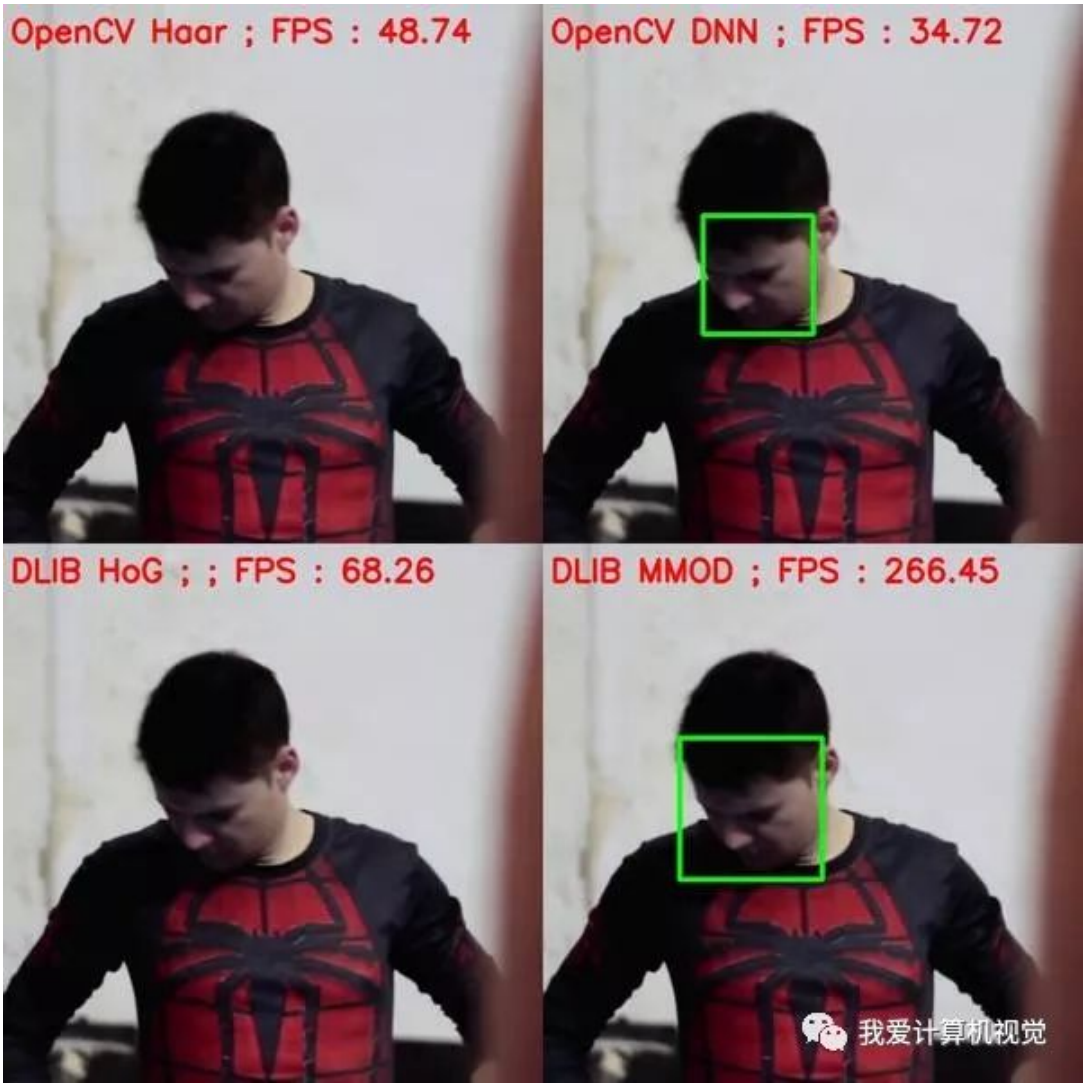

OpenCV Haar ; FPS : 5.19
53x53
OpenCV DNN ; FPS : 13.15
60x42
DLIB HoG ; ; FPS : 71.08
DLIB MMOD ; FPS : 2.11

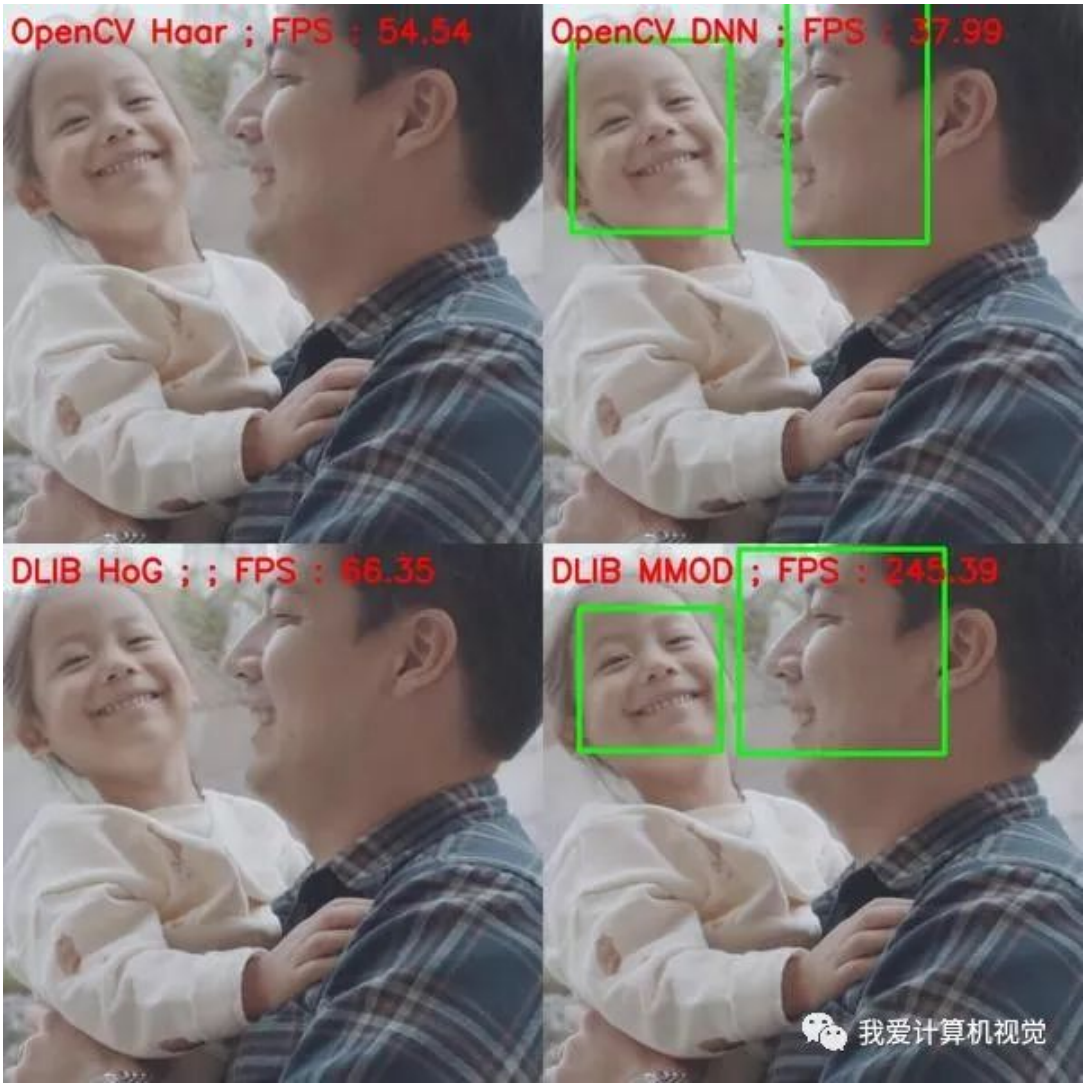
7.2 非正面人脸

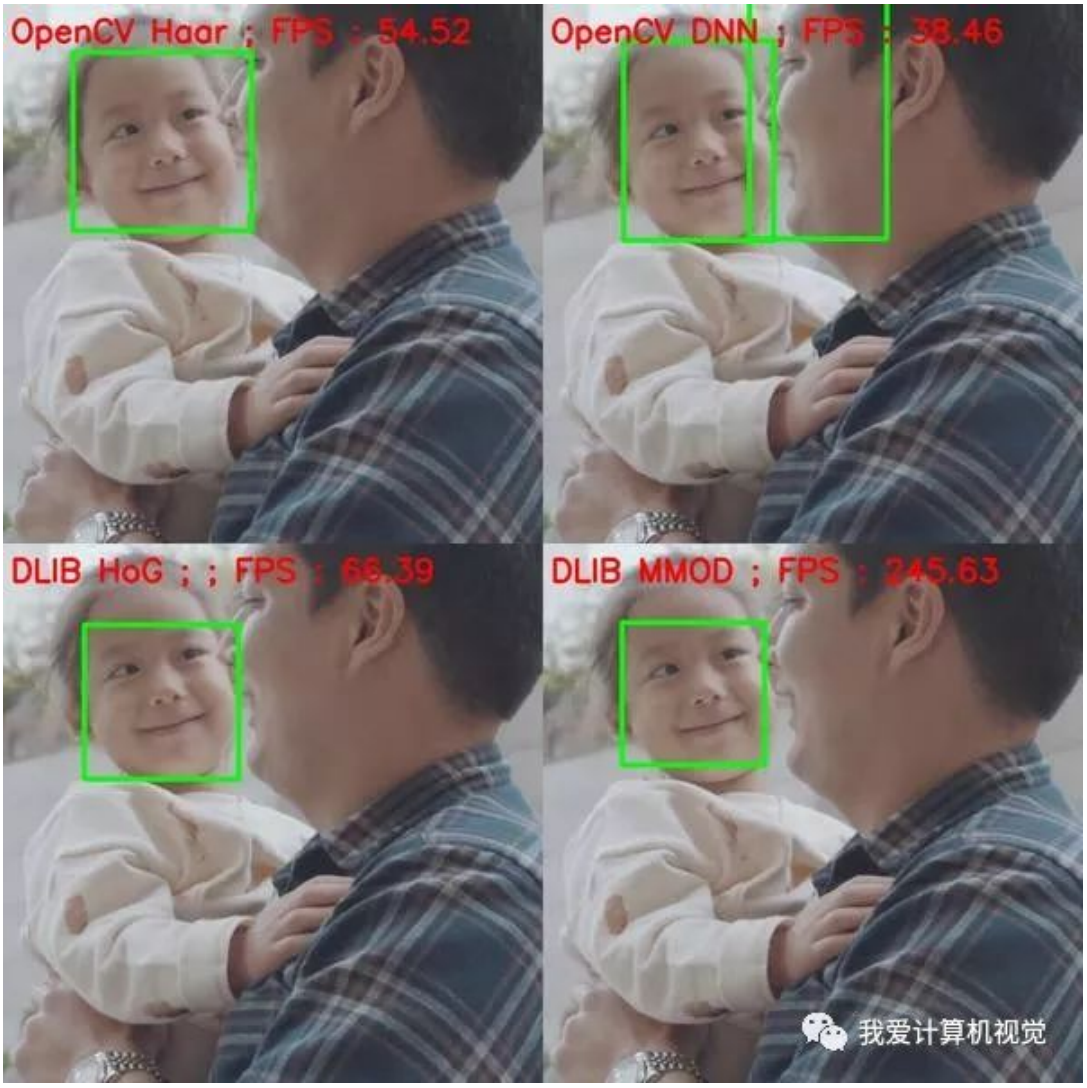

OpenCV Haar ; FPS : 53.98
OpenCV DNN ; FPS : 55.65
DLIB HoG ; ; FPS : 63.44
DLIB MMOD ; FPS : 24.25
我爱计算机视觉


OpenCV Haar ; FPS : 53.49
OpenCV DNN ; FPS : 37.30
DLIB HoG ; ; FPS : 69.15
DLIB MMOD ; FPS : 255.49
我爱计算机视觉


OpenCV Haar ; FPS : 53.23
OpenCV DNN ; FPS : 36.04
DLIB HoG ; ; FPS : 68.12
DLIB MMOD ; FPS : 257.50
我爱计算机视觉

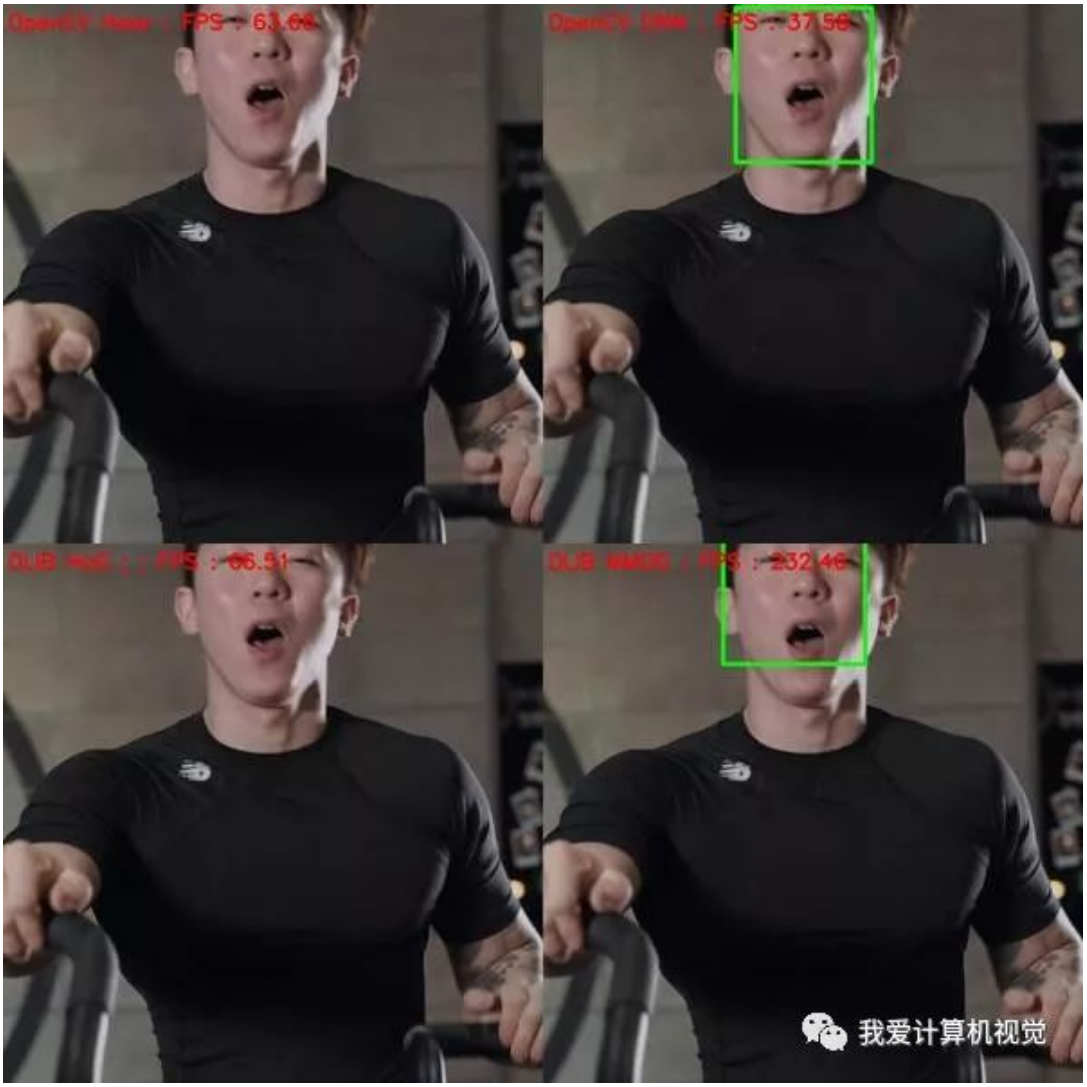

OpenCV Haar ; FPS : 48.74
OpenCV DNN ; FPS : 34.72
DLIB HoG ; ; FPS : 68.26
DLIB MMOD ; FPS : 266.45
我爱计算机视觉

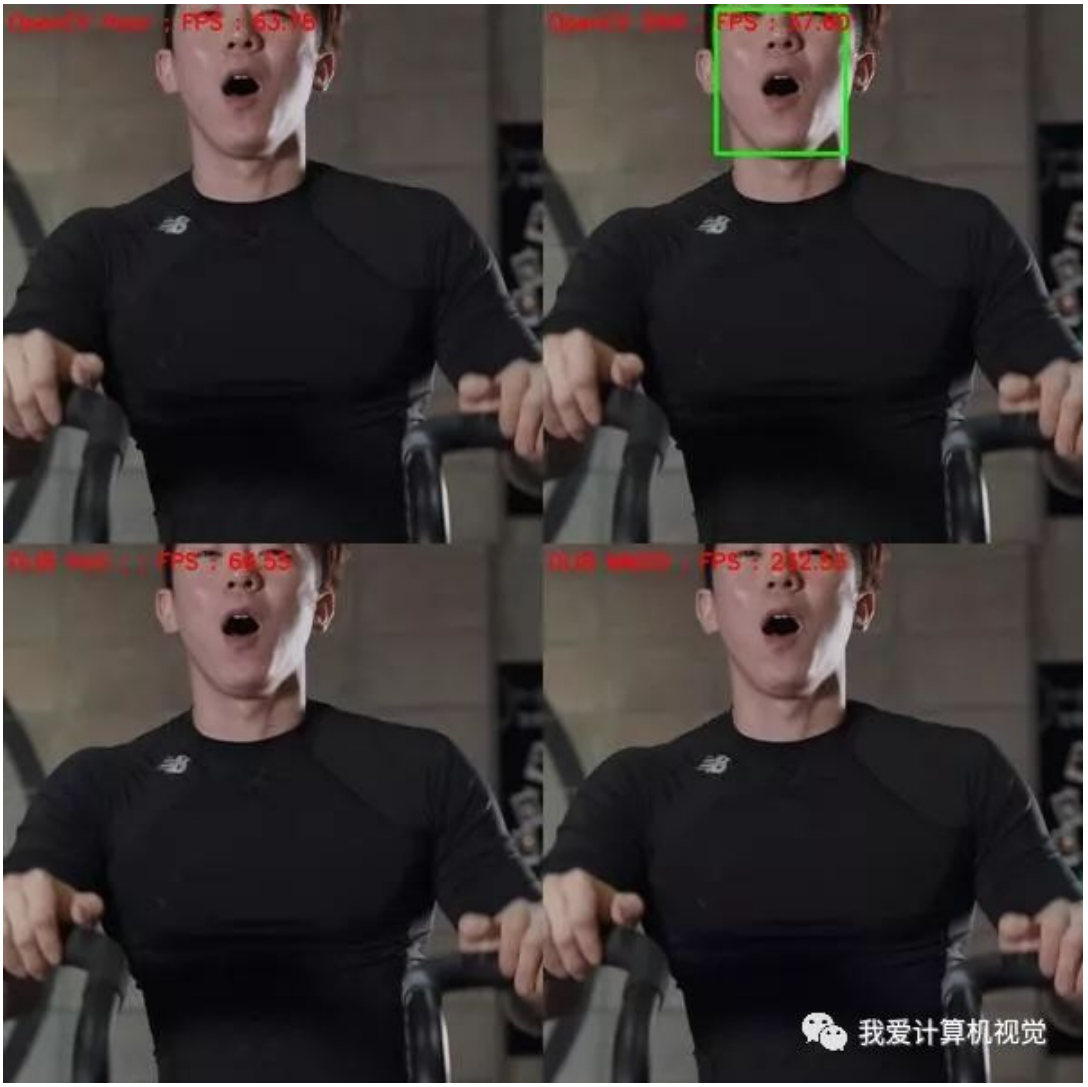

OpenCV Haar ; FPS : 54.54
OpenCV DNN ; FPS : 37.99
DLIB HoG ; ; FPS : 66.35
DLIB MMOD ; FPS : 245.39
我爱计算机视觉


OpenCV Haar ; FPS : 54.52
OpenCV DNN ; FPS : 38.46
DLIB HoG ; ; FPS : 66.39
DLIB MMOD ; FPS : 245.63
我爱计算机视觉

7.3 遮挡


OpenCV Haar ; FPS : 39.64
OpenCV DNN ; FPS : 36.31
DLIB HoG ; ; FPS : 39.13
DLIB MMOD ; FPS : 152.36
我爱计算机视觉


我爱计算机视觉



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（<70x70）。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：

https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/

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~

文章已于 修改

阅读原文