# Face Detection

## 1. Basic Theory

Face detection algorithms can be divided into two categories according to the method, **feature-based algorithms and image-based algorithms**.

- **Feature-based algorithms**

Feature-based algorithms are to extract features from images and match them with facial features. If they match, it means it is a face, otherwise it is not. The extracted features are artificially designed features, such as Haar, FHOG. After the features are extracted, the classifier is used to make a judgment. In layman's terms, template matching is used, that is, the template image of the face is matched with each position in the image to be detected. The matching content is the extracted features, and then the classifier is used to judge whether there is a face.

- **Image-based algorithms**

**Image-based algorithms**, **divide the image into many small windows, and then judge whether each small window has a face**. Usually, image-based methods rely on statistical analysis and machine learning, and find the statistical relationship between faces and non-faces through statistical analysis or learning processes to perform face detection. The most representative one is CNN. CNN is also the best and fastest for face detection.

- Haar feature

We use machine learning to complete face detection. First, we need a large number of positive samples (face images) and negative samples (images without faces) to train the classifier. We need to extract features from them. The Haar features in the figure below will be used, just like our convolution kernel. Each feature is a value, which is equal to the sum of the pixel values in the black rectangle minus the sum of the pixel values in the white rectangle.

This tutorial uses it. Haar cascade classifier This is the most common and efficient method for target detection. This machine learning method is based on training cascade functions based on a large number of positive and negative images, and then used to detect objects in other images. If you don't want to create your own classifier, OpenCV also contains many pre-trained classifiers that can be used for face, eye, smile, etc. detection.

## 2. Main code

Code path:

```
~/dofbot_pro/src/dofbot_basic_visual/scripts/04.Face_Detection.ipynb
```

Import Haar cascade classifier xml file

```
self.faceDetect = cv.CascadeClassifier("haarcascade_frontalface_default.xml")
```

Filter faces

```
def face_filter(self, faces):
        '''
        Filter the face
```

```
        对人脸进行一个过滤
        '''
        if len(faces) == 0: return None
        # At present, we are looking for the face with the largest area in the
pictur
        # 目前找的是画面中面积最大的人脸
        max_face = max(faces, key=lambda face: face[2] * face[3])
        (x, y, w, h) = max_face
        # Set the minimum threshold of face detection
        # 设置人脸检测最小阈值
        if w < 10 or h < 10: return None
        return max_face
```

Detect and select faces

```
def find_face(image):
    # Copy the original image to avoid interference during processing
    # 复制原始图像,避免处理过程中干扰
    img = image.copy()
    # Convert image to grayscale
    # 将图像转为灰度图
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    # Face detection
    # 检测人脸
    faces = faceDetect.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
    pos = None
    if len(faces) != 0:
        # Face filtering
        # 人脸过滤
        face = face_filter(faces)
        if face is not None:
            (x, y, w, h) = face
            # Draw a rectangle on the original color map
            # 在原彩图上绘制矩形
            cv.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 4)
    return image
```

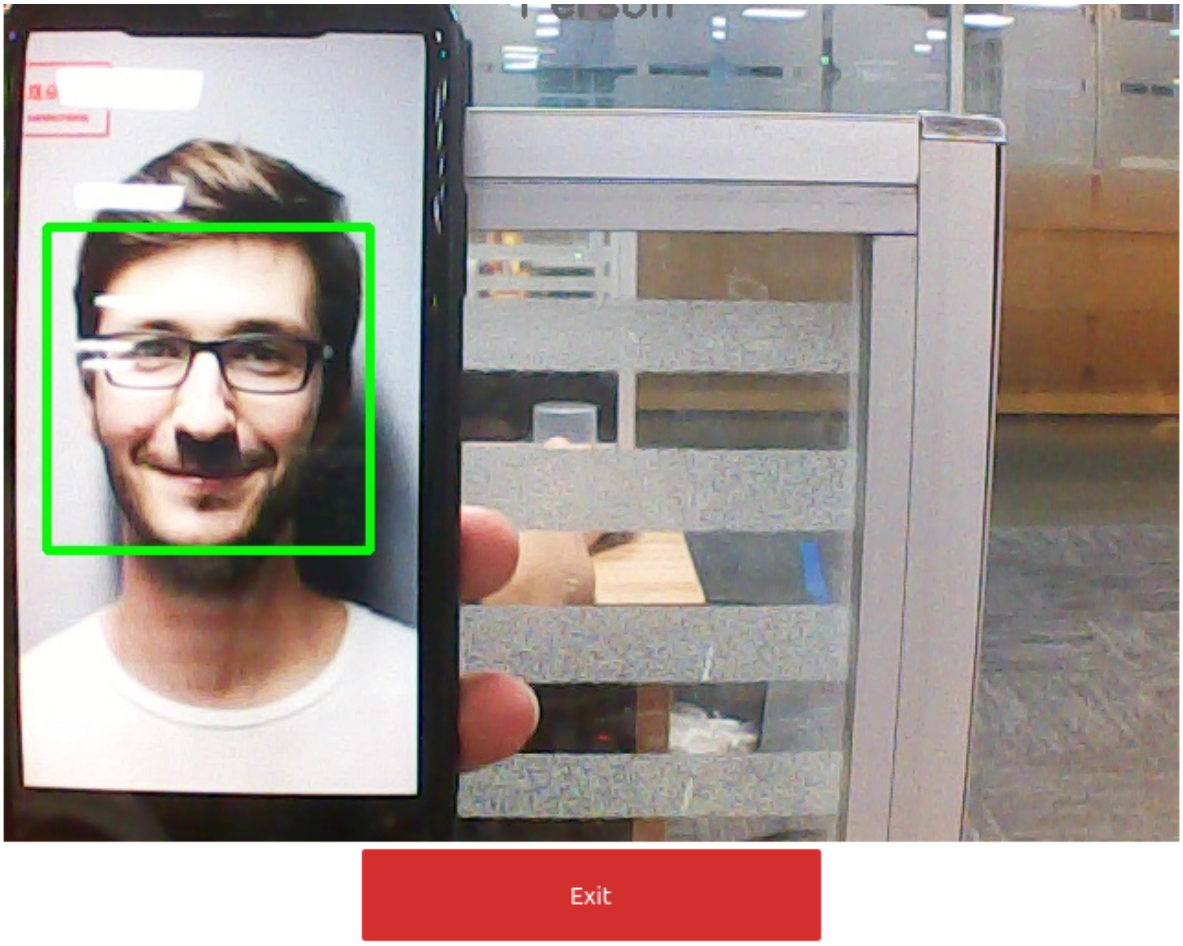
Main thread:

```
def camera():
    global model
    # 打开摄像头 Open camera
    capture = cv.VideoCapture(0)
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
    while capture.isOpened():
        try:
            if model == 'Exit':
                capture.release()
                break
            _, img = capture.read()
            fps.update_fps()
            find_face(img)
            fps.show_fps(img)
```

```
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except Exception as e:
            capture.release()
            print(e)
            break
```

Program Click the Run Entire Program button on the Jupyterlab toolbar, then scroll to the bottom to see the camera component display.


If a face appears on the camera screen at this time, the screen will detect and frame the face.




If you need to exit the program, please click the [Exit] button.