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 judgments. In layman's terms, it is to use template matching, that is, to match the template image of the face 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 features**

We use **machine learning** to complete **face detection**. First, **we need a large number of positive sample images (face images) and negative sample images (images without faces) to train the classifier**. We need to **extract features** from them. **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. Haar cascade classifier This is the most common and efficient method for object 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. About code

Code path：

```
~/jetcobot_ws/src/jetcobot_ai_basic/scripts/4.Face_detections.ipynb
```

The following code needs to be executed according to each actual step. You cannot run all at once. Running the last unit will directly exit the thread.

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
```

识别框选人脸

```
 def follow_function(self, img):
        img = cv.resize(img, (640, 480))
        # 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 = img.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 = self.faceDetect.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=5)

        if len(faces) != 0:
            face = self.face_filter(faces)
            # Face filtering
            # 人脸过滤
            (x, y, w, h) = face
            # Draw a rectangle on the original color map
            # 在原彩图上绘制矩形
            cv.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 4)
            cv.putText(img, 'Person', (280, 30), cv.FONT_HERSHEY_SIMPLEX, 0.8,
(105, 105, 105), 2)
```

主线程:

```
def camera():
    global model
    # 打开摄像头 Open camera
    capture = cv.VideoCapture(0)
    while capture.isOpened():
        try:
            _, img = capture.read()
            img = cv.resize(img, (640, 480))
```

```
            img = follow.follow_function(img)
            if model == 'Exit':
                cv.destroyAllWindows()
                capture.release()
                break
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except KeyboardInterrupt:capture.release()
```

After running the program, the system can recognize the face and select it.