Lesson 5 Advanced Image Processing Technology

1. Facial Detection and Recognition

Extracting the details of the image is very useful for generating stable classification results and tracking results. These extracted results are called features. For a given image, the features may be different due to the size of the area which can be called the window size. Even if the window sizes are different, two images that differ only in scale should have similar characteristics. Therefore, generating features that can be adapted to windows of different sizes is a very critical point in image processing.

The collection of the same feature in these different size windows is called cascade. Haar feature in OpenCV is a feature for real-time face tracking. The Haar cascade is scale-invariant, which means that its characteristics can be applied to images with different window sizes.

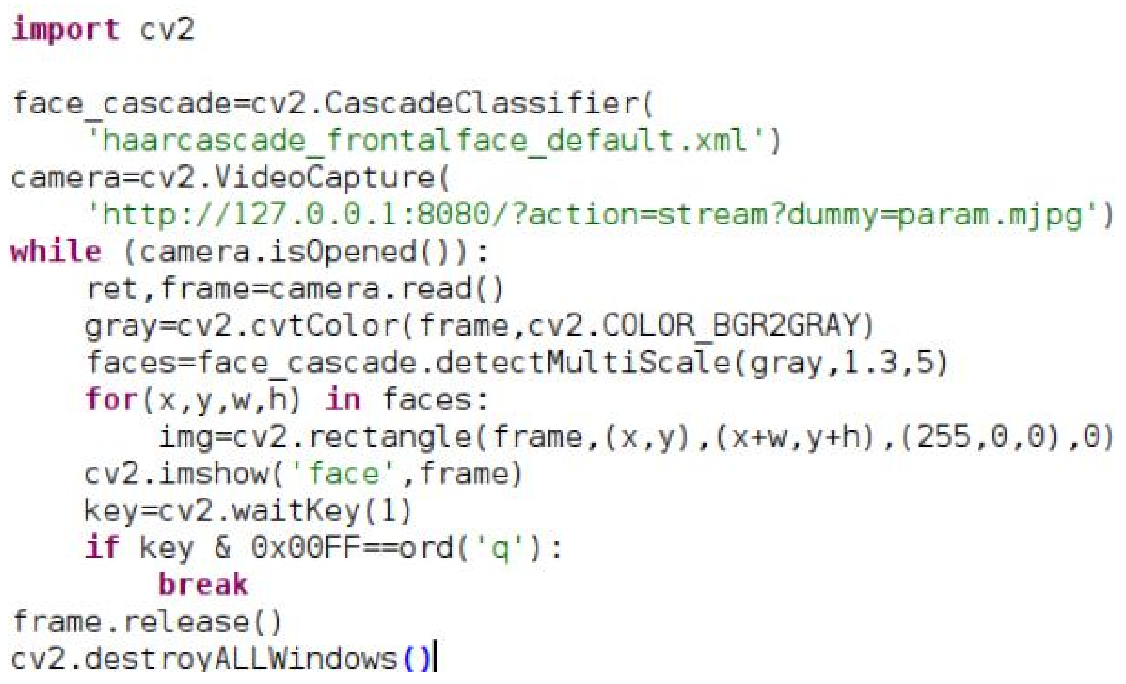
OpenCV provides Haar cascaded classifiers and trackers with constant scale, and can be saved as a specified file format. However, it should be noted that OpenCV's Haar cascade does not have rotation invariance. For example, Haar cascade does not consider the inverted face image to be the same as the upright face image.

The operation of detecting faces in still images or videos is very similar. Video facial recognition just reads each frame of image from the camera, and then uses the facial recognition method in the static images to detect.

Before programming the facial recognition, copy the facial recognition XML file in the haarcascades folder in the OpenCV source code (address in Raspberry

Pi: /usr/share/opencv/haarcascades) to the saved facial recognition Python code Folder.

Example: Detect a face through the camera and circle the face in a rectangle.



```
import cv2

face_cascade=cv2.CascadeClassifier(
    'haarcascade_frontalface_default.xml')
camera=cv2.VideoCapture(
    'http://127.0.0.1:8080/?action=stream?dummy=param.mjpg')
while (camera.isOpened()):
    ret,frame=camera.read()
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces=face_cascade.detectMultiScale(gray,1.3,5)
    for(x,y,w,h) in faces:
        img=cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),0)
    cv2.imshow('face',frame)
    key=cv2.waitKey(1)
    if key & 0x00FF==ord('q'):
        break
frame.release()
cv2.destroyALLWindows()
```

Face recognition generally uses scripts to generate original sampled images at first. Detects human faces, crop the gray frame area, adjust it to a fixed size, and finally save it in a folder. This face data will be used to train the face recognition model later.

After the sampling is completed, next step is face recognition. OpenCV 3 supports three face recognition methods, which are Eigenfaces, Fisherfaces and Local Binary Pattern Histogram (LBPH). After one of algorithm is selected, training and identification can be carried out.

2. Image Recognition and Retrieval

OpenCV can detect the main features of an image, and then extract these features to make it an image descriptor. These image features can be used as a database for image search, which can help you find the elements you are looking for in the image library. For example, if you want to find out who has a specific tattoo in tens of thousands of pictures, you can use the tattoo as an image descriptor to search for matching in each picture.

Image feature detection includes corner points and spots. A small change of a pixel at a certain point in an image in any direction will cause a large change in the gray value, then we call this point a corner point or key point. Spots refer to areas in a two-dimensional image that have color differences and gray-scale differences with the surrounding colors.

The most common feature detection and extraction algorithms in OpenCV are as follows:

Harris：detect corner point

SIFT：detect spot（blob）

SURF：detect spot

FAST：detect corner point

BRIEF：detect spot

ORB：This algorithm represents the FAST algorithm with direction and the BRIEF algorithm with rotation invariance.

Feature matching after detection has Brute-Force, which is based on FLANN.

The Brute-Force matching method compares two descriptors and list a matching results. It is called brute force because the algorithm basically does not involve optimization. All the features of the first descriptor are compared with the features of the second descriptor, and all possible combinations are listed.

FLANN is short for Fast_library_for_Approximate_Nearest_Neighbors. It is a collection of algorithms for searching for the nearest similar point on large data sets and high-dimensional features. These algorithms have been optimized and have high processing efficiency.

3. Deep Learning

In addition to the aforementioned, OpenCV can also detect a moving target in the lens and track and predict the trajectory of the target (using Kalman filter).

With ANN (artificial neural network) part in the ml module of OpenCV, we can complete some machine learning projects, such as handwriting recognition and human-computer interaction.

OpenCV belongs to the field of artificial intelligence. The above courses are just an introductory guide for beginners to lead everyone to understand the application principle and learning direction of OpenCV. However, it is far from enough to rely on this tutorial to master OpenCV technology. If you have strongly interests, please check the related books.