Face detection

In OpenCV, face detection in video only reads each frame of image from the camera, and then uses the static image detection method for detection. Face detection requires the classifier:

@ Face detector(default): haarcascade_frontalface_default.xml
@ Face detector(fast Harr): haarcascade_frontalface_alt2.xml
@ Face detector(Side view): haarcascade_profileface.xml
@ Eye detector(left eye): haarcascade_lefteye_2splits.xml
@ Eye detector(right eye): haarcascade_righteye_2splits.xml
@ Mouth detector: haarcascade_mcs_mouth.xml
@ Nose detector: haarcascade_mcs_nose.xml
@ Body detector: haarcascade_fullbody.xml
@ Face detector(fast LBP): lbpcascade_frontalface.xml
@ Only open eyes can be detected: haarcascade_eye.xml
@ Only person with glasses can be detected: haarcascade_eye_tree_eyeglasses.xml
@ https://github.com/opencv/opencv/tree/master/data Download classifier file link

haarcascade_profileface.xml is the cascading data of Haar. This xml can be obtained from this link
https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_profileface.xml

Next, we can start face detection by face_cascade.detectMultiScale (). We need to convert the image into a grayscale, then, transfer each frame of the image obtained by the camera into .detectMultiScale ().

(Note: we need to ensure enter the correct location of haarcascade_profileface.xml correctly.)

OpenCV API function:

detectMultiScale(const Mat& image, vector& objects, double scaleFactor=1.1, int minNeighbors, int flag, cvSize)

Parameter analysis:

image --- Input grayscale image
objects --- The rectangular box vector set of the detected object
scaleFactor --- Each scale parameter in the image scale, the default value is 1.1.
minNeighbors --- Default is 3.
minNeighbors --- The default value of 3 indicates that there are at least 3 overlap detections, so we think that the face is exist.
minSize --- Target minimum size
maxSize --- Target maximum size

Code path:

/home/pi/Yahboom_Project/Raspbot/3.AI Vision course/08.Face detection/Face detection.ipynb

```
#bgr8 to jpeg format
import enum
import cv2

def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```
# Camera display components
import cv2
import ipywidgets.widgets as widgets
import threading
import time
import sys

image_widget = widgets.Image(format='jpeg', width=320, height=240)
display(image_widget)
```

```
image = cv2.VideoCapture(0)
image.set(3,320)
image.set(4,240)
image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
image.set(cv2.CAP_PROP_BRIGHTNESS, 40) #set brightness -64 - 64    0.0
image.set(cv2.CAP_PROP_CONTRAST, 50)     #set contrast -64 - 64    2.0
image.set(cv2.CAP_PROP_EXPOSURE, 156)    #set exposure 1.0 - 5000    156.0
ret, frame = image.read()
image_widget.value = bgr8_to_jpeg(frame)
```

```
# Thread function operation library
import inspect
import ctypes
def _async_raise(tid, exctype):
    """raises the exception, performs cleanup if needed"""
    tid = ctypes.c_long(tid)
    if not inspect.isclass(exctype):
        exctype = type(exctype)
    res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid,
ctypes.py_object(exctype))
    if res == 0:
        raise ValueError("invalid thread id")
    elif res != 1:
        # """if it returns a number greater than one, you're in trouble,
        # and you should call it again with exc=NULL to revert the effect"""
        ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)

def stop_thread(thread):
    _async_raise(thread.ident, SystemExit)
```

```
# body_haar = cv2.CascadeClassifier("haarcascade_upperbody.xml")
  face_haar = cv2.CascadeClassifier("haarcascade_profileface.xml")
# face_haar = cv2.CascadeClassifier("haarcascade_fullbody.xml")
#eye_haar = cv2.CascadeClassifier("haarcascade_eye.xml")
# eye_haar = cv2.CascadeClassifier("haarcascade_eye_tree_eyeglasses.xml")
def Camera_display():
        while 1:
                ret, frame = image.read()
                # Convert the image to black-white image
                gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
#                   # Detect all pedestrians in the video
#               bodies = body_haar.detectMultiScale(gray_img, 1.3, 5)
#               for body_x,body_y,body_w,body_h in bodies:
#                cv2.rectangle(frame, (body_x, body_y), (body_x+body_w, body_y+body_h),
(0,255,0), 2)

                faces = face_haar.detectMultiScale(gray_img, 1.1, 3)
                for face_x,face_y,face_w,face_h in faces:
                        cv2.rectangle(frame, (face_x, face_y), (face_x+face_w, face_y+face_h),
(0,255,0), 2)

#                 eyes = eye_haar.detectMultiScale(gray_img, 1.1, 3)
#                 for eye_x,eye_y,eye_w,eye_h in eyes:
#                         cv2.rectangle(frame, (eye_x,eye_y), (eye_x+eye_w, eye_y+eye_h),
(255,0,0), 2)

#                   eyes = eye_haar.detectMultiScale(gray_img, 1.3, 5)
#                   for eye_x,eye_y,eye_w,eye_h in eyes:
#                           cv2.rectangle(frame, (eye_x,eye_y), (eye_x+eye_w, eye_y+eye_h),
(255,0,0), 2)

                image_widget.value = bgr8_to_jpeg(frame)
                time.sleep(0.010)
```

```
# Start thread
thread1 = threading.Thread(target=Camera_display)
thread1.setDaemon(True)
thread1.start()
```

```
# End the process, release the camera, need to be executed at the end
stop_thread(thread1)
image.release()
```

Because we chose the model of the side of the face, after running the above program, we can recognize the side of the face.