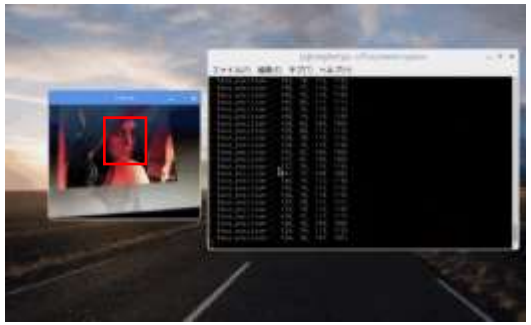
カメラを使った顔認識 (1/2)



- 概要
  - ラズパイ専用カメラで、動画を撮影し、画面上に表示する。
  - 随時、顔認識を行い、顔を認識できればその部分を赤枠で囲む。
  - コンソールには、顔の位置、高さ、幅を表示する。

- 準備
  - ラズパイをインターネットに接続した状態で、下記の手順にてOpen CVをインストールする

```
$ sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev
  libswscale-dev python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev
  libjasper-dev libdc1394-22-dev
$ sudo pip3 install opencv-python
$ sudo pip3 install opencv-contrib-python
$ sudo apt-get install libcblas-dev libatlas3-base libilmbase12 libopenexr22 libgstreamer1.0-0
  libqtgui4 libqttest4-perl
```

  - 実は、環境の構築が結構大変。インストールやビルド時にエラーやワーニングが発生し、その解決にはある程度のLinuxやOpen CVの知識が必要となる。

  - 顔認識するために必要なカスケードファイル(学習したモデル)は、以下からダウンロードする。
    https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

カメラを使った顔認識 (2/2)

- スクリプト

```
import picamera
import picamera.array
import cv2

cascade_file = "haarcascade_frontalface_default.xml"

with picamera.PiCamera() as camera:
  with picamera.array.PiRGBArray(camera) as stream:
    camera.resolution = (320, 240)

    while True:
      # stream.arrayに映像データを格納
      camera.capture(stream, 'bgr', use_video_port=True)

      # グレースケールに変換
      gray = cv2.cvtColor(stream.array, cv2.COLOR_BGR2GRAY)

      # カスケードファイルを利用して顔の位置を見つける
      cascade = cv2.CascadeClassifier(cascade_file)
      face_list = cascade.detectMultiScale(gray, minSize=(100, 100))

      for (x, y, w, h) in face_list:
        print("face_position:",x, y, w, h)
        color = (0, 0, 255)
        pen_w = 5
        cv2.rectangle(stream.array, (x, y), (x+w, y+h), color, thickness = pen_w)

      # stream.arrayをウィンドウに表示
      cv2.imshow('frame', stream.array)

      # "q"でウィンドウを閉じる
      if cv2.waitKey(1) & 0xFF == ord("q"):
        break

      # streamをリセット
      stream.seek(0)
      stream.truncate()

  cv2.destroyAllWindows()
```


【出典】 https://qiita.com/K_M95/items/41a24d7639cffc629227

カメラを使った物体認識 (1/3)

- スクリプト

- yolov2-tinyを使用する。
- ラズパイ専用カメラで撮影した動画に対して物体認識を行い、認識した物体には枠とその名称を表示する。
- 物体としては、class_namesで定義している20種類を識別することができる。
- 動画の解像度が高いため、認識には数秒程度の時間が掛かる。

```
from darkflow.net.build import TFNet
import cv2
import numpy as np
import picamera
import io

options = {"model": "cfg/yolov2-tiny-voc.cfg", "load": "bin/yolov2-tiny-voc.weights", "threshold": 0.1}
tfnet = TFNet(options)
class_names = [ 'aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable',
          'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor' ]

# カメラの起動
stream = io.BytesIO()
CAMERA_WIDTH = 1024
CAMERA_HEIGHT = 768
camera = picamera.PiCamera()
camera.resolution = (CAMERA_WIDTH, CAMERA_HEIGHT)
num_classes = len(class_names)
class_colors = []
for i in range(0, num_classes):
    hue = 255*i/num_classes
    col = np.zeros((1,1,3)).astype("uint8")
    col[0][0][0] = hue
    col[0][0][1] = 128
    col[0][0][2] = 255
    cvcol = cv2.cvtColor(col, cv2.COLOR_HSV2BGR)
    col = (int(cvcol[0][0][0]), int(cvcol[0][0][1]), int(cvcol[0][0][2]))
    class_colors.append(col)
```


【出典】 https://qiita.com/daiarg/items/2ef986732632af5a5133

カメラを使った物体認識 (2/3)

- スクリプト

```
def main():
    while(True):
        #カメラ撮影
        camera.capture(stream, format='jpeg')
        #numpy型に変換
        data = np.fromstring(stream.getvalue(), dtype=np.uint8)
        #opencv型に変換
        frame = cv2.imdecode(data, 1)
        # 動画ストリームからフレームを取得
        result = tfnet.return_predict(frame)
        for item in result:
            tlx = item['topleft']['x']
            tly = item['topleft']['y']
            brx = item['bottomright']['x']
            bry = item['bottomright']['y']
            label = item['label']
            conf = item['confidence']
            if conf > 0.6:
                for i in class_names:
                    if label == i:
                        class_num = class_names.index(i)
                        break
                #枠の作成
                cv2.rectangle(frame, (tlx, tly), (brx, bry), class_colors[class_num], 2)
                #ラベルの作成
                text = label + " " + ('%.2f' % conf)
                cv2.rectangle(frame, (tlx, tly - 15), (tlx + 100, tly + 5), class_colors[class_num], -1)
                cv2.putText(frame, text, (tlx, tly), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1)
```

カメラを使った物体認識 (3/3)

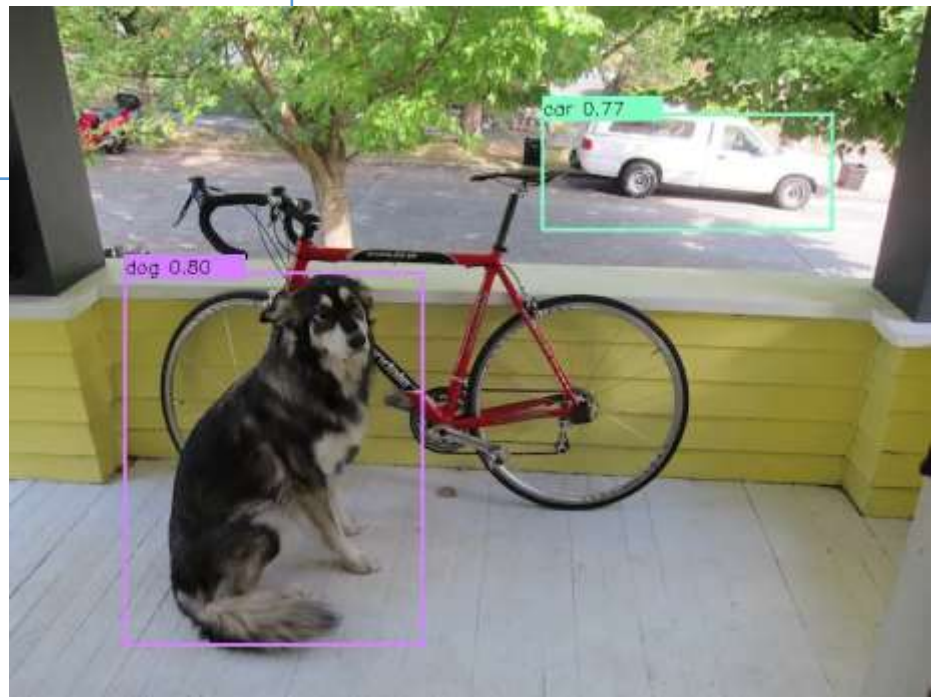
- スクリプト

```
        # 表示
        cv2.imshow("Show FLAME Image", frame)
        #カメラの状態をリセット
        stream.seek(0)

        # escを押したら終了
        k = cv2.waitKey(10);
        if k == ord('q'):  break;

    cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```



物体認識の例