SHIP Workshop Deck

Raspberry Piを使って画像認識に挑戦しよう(2)

#ラズパイ

#SHIP静岡

SBS情報システム
アリフ・マウラナ

SBS情報システム

自己紹介

アリフ　マウラナ

2017年にSBS情報システムに入社。医療事業本部にてシステム開発に従事。

2018年に次世代ソリューション開発本部へ異動後は、

2020年コロナ禍における行動履歴を管理するアプリでIBM DXチャレンジ参加。

2021年には無人販売所の運営を支援するシステムでNAGOYA BOOSTに参加。

2022年には静岡市清沢地区にてリアルタイム商品情報公開システム実証実験を
実施するなど、Arduino, Raspberry Pi, and Jetson Nanoといったシングルボード
コンピュータと3Dプリンタを駆使して農業を中心とした課題の解決に取り組んでいる。

アジェンダ

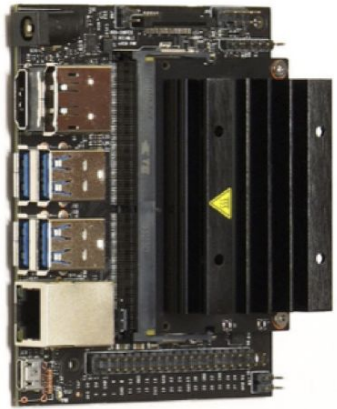
- Raspberry Pi と 画像認識について
- Python 開発環境の準備
- Visual Studio Code のインストール
- OpenCV を使ってみる
- 顔認識に挑戦！

Raspberry Pi と画像認識

Raspberry Pi (ラズパイ) とは？

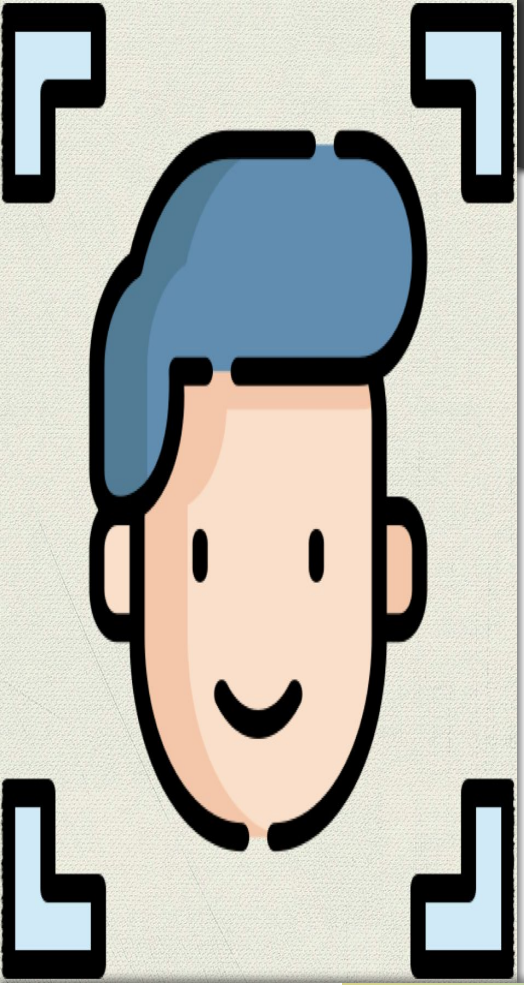
Arduino
2,000～7,000円
CPU 8~32bit

Raspberry Pi
10,000～17,000円
CPU 32~64bit



Jetson Nano
24,000～30,000円
CPU 64bit & GPU

- **クレジットカードサイズのコンピューター**
  動画再生やインターネット接続など、
  普通のPCにできることはほとんどできる
- 似たようなデバイスには他に、
  Arduino や Jetson Nano がある
- 普通のPCとの大きな違いは、**GPIOポート**
  これにより、外部のセンサーからデータを
  取得したり、モーターを動かしたりできる



Raspberry Pi で顔認識

今回は Raspberry Pi で出来ることの一例として、
ライブラリを使用した顔認識プログラムを紹介します

顔認識とは、画像の中に誰がいるかをコンピュータが
認識できるようにするための仕組みです

Raspberry Piに用意した環境と同様の環境を
みなさんのPCに用意して実行できるようにしましょう

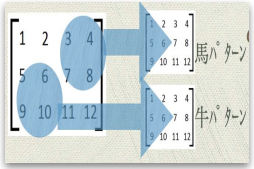
コンピュータが画像から物体を認識する仕組み

コンピュータにとって、画像は数字の羅列でしかありません
「牛や馬が写っている」という情報は持っていません

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

その代わり正確で、人間のように忘れたりしません

画像に牛や馬が写っていることをコンピュータに認識させるためには、牛や馬が写っている画像のパターンを学習させる必要があります



Python 開発環境の準備

Python 3.7.9 のインストール (1/3)

- 公式サイトからPythonのインストーラをダウンロード
  https://www.python.org/downloads/release/python-379/

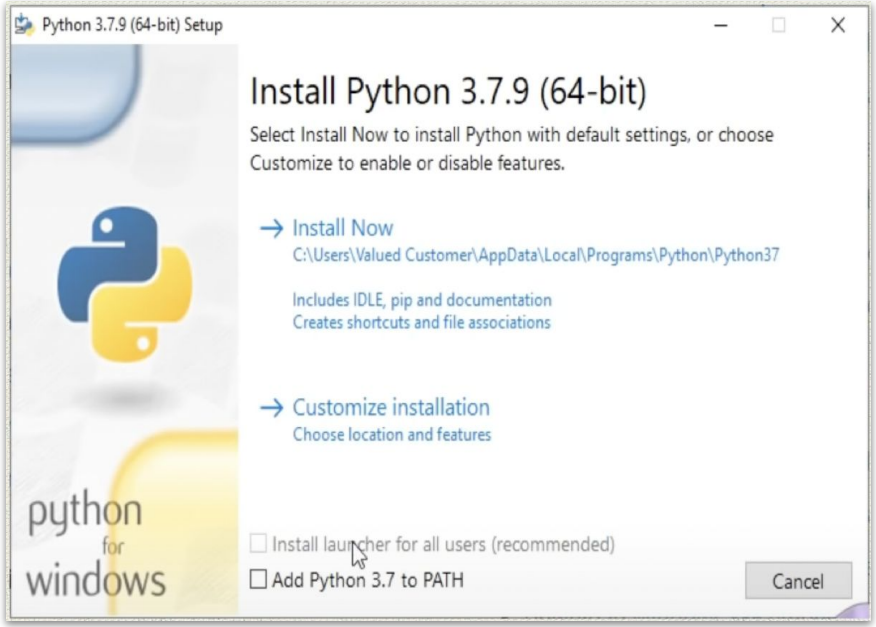
Files

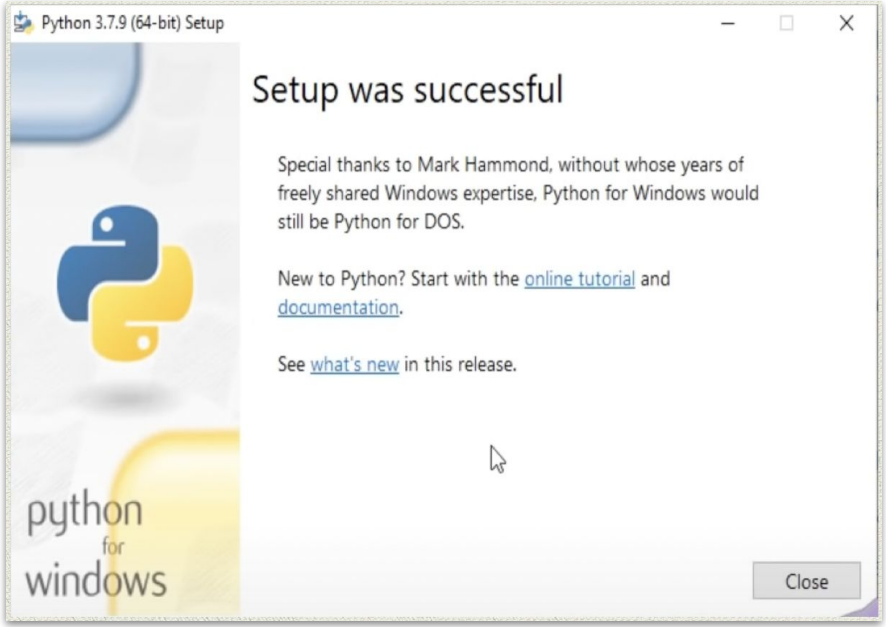
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		bcd9f22cf531efc6f06ca6b9b2919bd4	23277790	SIG
XZ compressed source tarball	Source release		389d3ed26b4d97c741d9e5423da1f43b	17389636	SIG
macOS 64-bit installer	macOS	for OS X 10.9 and later	4b544fc0ac8c3cffdb67dede23ddb79e	29305353	SIG
Windows help file	Windows		1094c8d9438ad1adc263ca57ceb3b927	8186795	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	60f77740b30030b22699dbd14883a4a3	7502379	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	7083fed513c3c9a4ea655211df9ade27	26940592	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	da0b17ae84d6579f8df3eb24927fd825	1348904	SIG
Windows x86 embeddable zip file	Windows		97c6558d479dc53bf448580b66ad7c1e	6659999	SIG
Windows x86 executable installer	Windows		[illegible]	25875560	SIG
Windows x86 web-based installer	Windows		22f68f09e533c4940fc006e035f08aa2	1319904	SIG

Windowsの方は基本コチラ
(Windows x86-64 executable installer)

Python 3.7.9 のインストール (2/3)

- インストーラーを実行
- 「 ☑ **Add Python 3.7 to PATH** 」にチェックをいれて **Install Now**





Python 3.7.9 のインストール (3/3)

- Windows の方は PowerShell、Mac の方はターミナルを起動
- python コマンドを実行

PowerShell (ターミナル)

```
PS C:\Users\...> python
```

※ コマンドが見つからない等のエラーが出た方は教えてください

- Python 3.7.9 … と表示されたら quit() を実行して終了する

PowerShell (ターミナル)

```
>>> quit()
```

※ここではコマンドプロンプトは使わないでください

Python Virtual Environment の準備 (1/2)

- デスクトップに作業フォルダを作成
- PowerShell (ターミナル)

```
PS C:\Users\...> cd .\Desktop\
PS C:\Users\...\Desktop> mkdir workshop
PS C:\Users\...\Desktop> cd .\workshop\
PS C:\Users\...\Desktop\workshop> ls
```

lsを実行した後、何も出てこなければOK

- PowerShell (ターミナル)

```
PS C:\Users\...\Desktop\workshop> python -m venv pyrasppi
```

Python Virtual Environment の準備 (2/2)

- Windows の場合

PowerShell

```
PS C:\...\workshop> set-executionpolicy remotesigned -Scope CurrentUser
PS C:\...\workshop> pyrasppi/Scripts/activate
```

- Mac の場合

ターミナル

```
C:\...\workshop> source pyrasppi/Scripts/activate
```

- activate の成功が確認できた方は deactivate していただいて構いません

PowerShell (ターミナル)

```
(pyrasppi) PS C:\...\workshop> deactivate
```

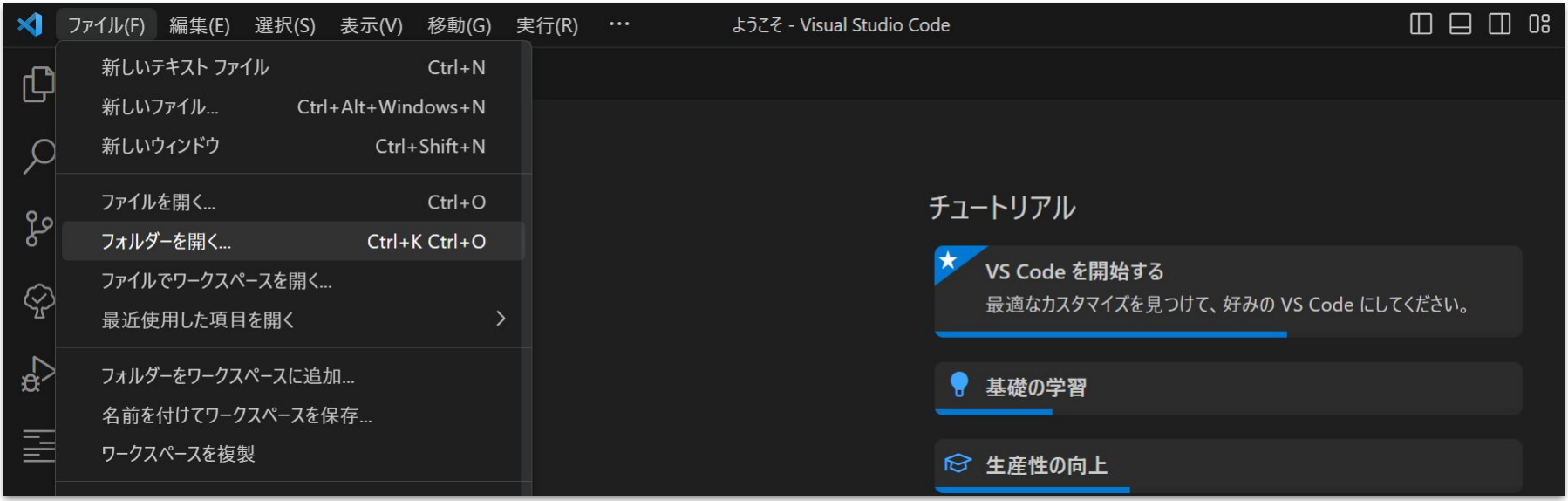
Visual Studio Code

Visual Studio Code のインストール

- Visual Studio Code：プログラムを書くためのアプリケーション
- 公式サイトからインストーラーをダウンロードして実行
  https://code.visualstudio.com/download

Visual Studio Code で作業フォルダを開く

- 「ファイル (File)」を開き、「フォルダーを開く (Open Folder)」をクリック
  デスクトップにある「workshop」を開く



OpenCV を使ってみる

OpenCV のインストール

- 「ターミナル (Terminal)」の「新しいターミナル」をクリック

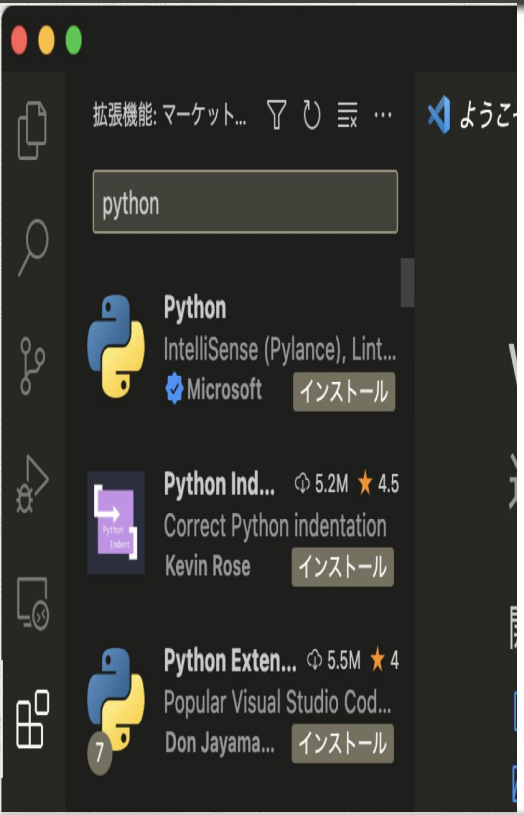
- 現れたターミナルでコマンドを実行

※Terminalの種類は不問

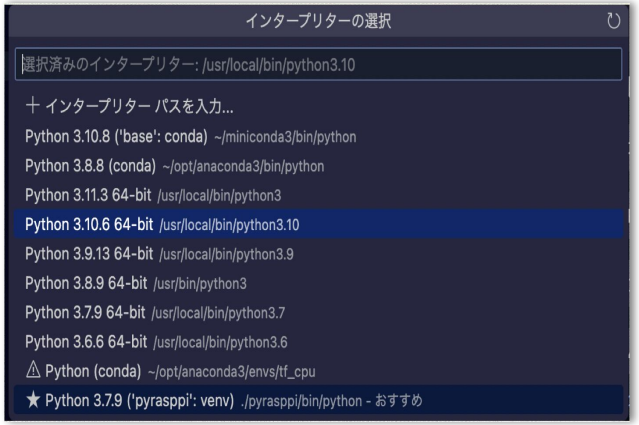
ターミナル

```
C:\...\workshop> pip install opencv-python
C:\...\workshop> python
>>> import cv2
>>> print(cv2.__version__)
>>> quit()
```

拡張機能の追加



- Python の拡張機能を検索してインストール
- [Ctrl] と [Shift] と [P] を同時に押して「Python: Select Interpreter」をクリック
- 「★ Python 3.7.9 ('pyrasppi': venv)」を選択



Visual Studio Code で Python を実行する

- エクスプローラーを開き、新規ファイル (openCV-1.py) を作成
- openCV-1.py を開き、以下のプログラムを記述

openCV-1.py

```
import cv2
print(cv2.__version__)
```

- 右上の ▷ を押して実行

カメラを起動してみよう

- 以下のプログラムを記述して実行

openCV-1.py

```
import cv2
print(cv2.__version__)
cam=cv2.VideoCapture(0)
while True:
    ignore, frame = cam.read()
    cv2.imshow('my WebCam', frame)
    if(cv2.waitKey(1) & 0xff == ord('q')):
        break
cam.release()
```

追加

顔認識に挑戦！

顔認識ロジックをどこに書くか？

```
import cv2
print(cv2.__version__)
cam=cv2.VideoCapture(0)
while True:
    ignore, frame = cam.read()

    cv2.imshow('my WebCam', frame)
    if(cv2.waitKey(1) & 0xff == ord('q')):
        break
cam.release()
```

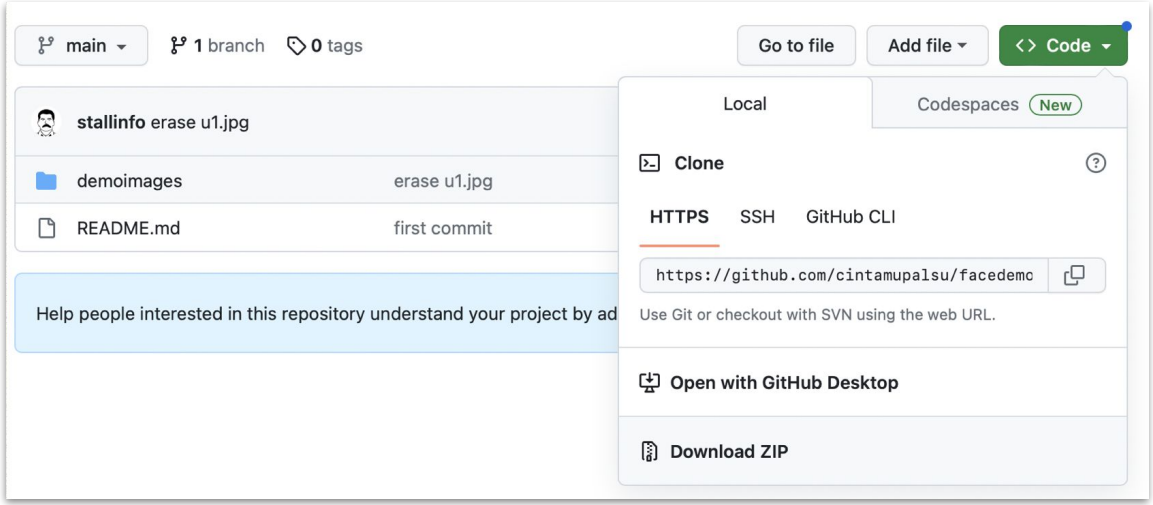
映像を読み込む

ここに顔認識を実装します

画像を表示する

デモ画像のダウンロード

- こちらのURLから.zipファイルをダウンロード
  https://github.com/cintamupalsu/facedemoimages



- demoimages フォルダを workshop フォルダの中へ

Python Virtual Environment を有効化

- Windows の場合

ターミナル

```
C:\...\workshop> pyrasppi/Scripts/activate
(pyrasppi) C:\...\workshop>
```

- Mac の場合

ターミナル

```
C:\...\workshop> pyrasppi/bin/activate
(pyrasppi) C:\...\workshop>
```

顔認識ライブラリをインストール

- Cmake をインストール

ターミナル

```
(pyrasppi) C:\...\workshop> pip install Cmake
```

※ 権限エラーになった場合、頭に sudo を付けてみてください

- face_recognition をインストール

ターミナル

```
(pyrasppi) C:\...\workshop> pip install face_recognition==1.2.3
```

Windows で上記コマンドが成功しなかった場合、以下を試してください

```
(pyrasppi) C:\...\workshop> pip install dlib
(pyrasppi) C:\...\workshop> pip install wheel
```

Windows でまだ問題が解決しない場合は、以下をダウンロードしてください
https://visualstudio.microsoft.com/ja/visual-cpp-build-tools/

インストール時に「C++ によるデスクトップ開発」にチェックを入れて
インストールしてください

顔認識ライブラリをテスト

- 新規ファイル (openCV-2.py) を作成
- 以下のプログラムを記述して実行

openCV-2.py

```
import cv2
import face_recognition as FR
font=cv2.FONT_HERSHEY_SIMPLEX
```

- エラーが出なければ、ライブラリが正常にインストールされています

顔の場所を認識する

- 今回はSBSの岡村アナウンサーの画像を使います。

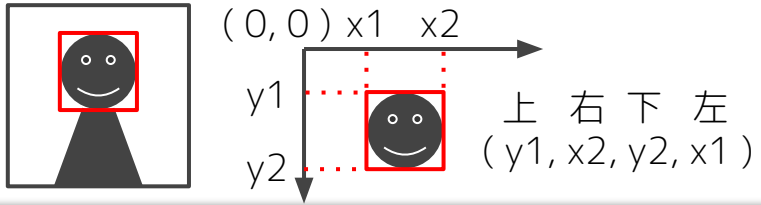
openCV-2.py

```
import cv2
import face_recognition as FR
font=cv2.FONT_HERSHEY_SIMPLEX
```

追加

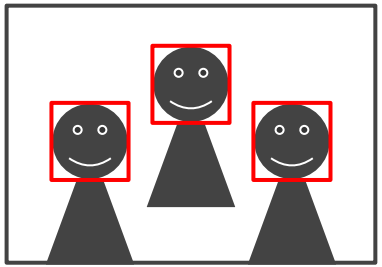
```
okamuraKao=FR.load_image_file('okamura_hisanori.jpgのファイルパス')
faceLoc=FR.face_locations(okamuraKao)
print(faceLoc)
```

- 実行すると、認識に成功した顔の座標（上、右、下、左）が出力されます



結果が配列型になる理由

- 1枚の画像の中に複数の顔がある可能性があるため、
  複数のデータを表すことのできる **配列形式** で出力される



[(a1, a2, a3, a4), (b1, b2, b3, b4), (c1, c2, c3, c4)]

1人目　　2人目　　3人目

- プログラミングでは配列の1個目を取得する時には[0]、2個目は[1]、…
  というように選択する

```
faceLoc    = [ (a1, a2, a3, a4), (b1, b2, b3, b4), (c1, c2, c3, c4) ]
faceLoc[0]= (a1, a2, a3, a4)
faceLoc[1]= (b1, b2, b3, b4)
```

画像を出力する

- 顔の位置を正しく認識できているかを画像で確認するために、
  まずは読み込んだ岡村アナの画像を出力してみましょう

openCV-2.py

```
import cv2
import face_recognition as FR
font=cv2.FONT_HERSHEY_SIMPLEX
okamuraKao=FR.load_image_file('okamura_hisanori.jpgのファイルパス')
faceLoc=FR.face_locations(okamuraKao)[0]
print(faceLoc)
cv2.imShow('my Window', okamuraKao)
cv2.waitKey(5000)
```

追加

色が…変…？

- 人間じゃない色をしていますね。これは何故でしょうか？
- 一般的に、色コードは Red, Green, Blue の順番で表されます (RGB形式)
  OpenCVでは Blue, Green, Red の順番で表されるので (BGR形式)
  赤と青が入れ替わってしまい、顔色が悪いような画像になってしまいました
- okamuraKaoの赤と青を入れ替えた、okamuraKaoBGR を作りましょう！

openCV-2.py

```
font=cv2.FONT_HERSHEY_SIMPLEX
okamuraKao=FR.load_image_file('okamura_hisanori.jpgのファイルパス')
追加 okamuraKaoBGR=cv2.cvtColor(okamuraKao,cv2.COLOR_RGB2BGR)
faceLoc=FR.face_locations(okamuraKao)[0]
print(faceLoc)
```

画像で座標を確認する

- 顔の位置を正しく認識できているかを画像で確認するために、
  座標を元に四角形を描画しましょう

openCV-2.py

```
okamuraKao=FR.load_image_file('okamura_hisanori.jpgのファイルパス')
okamuraKaoBGR=cv2.cvtColor(okamuraKao,cv2.COLOR_RGB2BGR)
faceLoc=FR.face_locations(okamuraKao)[0]
```

追加

```
top,right,bottom,left=faceLoc
cv2.rectangle(okamuraKaoBGR,(left,top),(right,bottom),(255,0,0),2)
cv2.imshow('my Window',okamuraKaoBGR)
```

```
cv2.waitKey(5000)
```

特定の顔だけを認識する

- face_locations では、誰の顔でも認識していましたが、ここからは
  特定の人の顔だけを認識するようにしてみましょう
  これにはまず、face_encodings を使用します

openCV-2.py

```
import cv2
import face_recognition as FR
font=cv2.FONT_HERSHEY_SIMPLEX
okamuraKao=FR.load_image_file('okamura_hisanori.jpgのファイルパス')
okamuraKaoEncode=FR.face_encodings(okamuraKao)[0]
```

複数の人を区別して認識する

- さらに、岡村アナと重長アナをそれぞれ認識できるようにしてみましょう
  重長アナの画像も同様にエンコードします

openCV-2.py

```
import cv2
import face_recognition as FR
font=cv2.FONT_HERSHEY_SIMPLEX
okamuraKao=FR.load_image_file('okamura_hisanori.jpgのファイルパス')
okamuraKaoEncode=FR.face_encodings(okamuraKao)[0]
shigenagaKao=FR.load_image_file('shigenaga_tomoko.jpgのファイルパス')
shigenagaKaoEncode=FR.face_encodings(shigenagaKao)[0]
```

エンコードに名前を付ける

- 「これは○○さんです」という判定ができるようにするために、
  エンコードしたデータに名前を付けます

openCV-2.py

```
shigenagaKao=FR.load_image_file('shigenaga_tomoko.jpgのファイルパス')
shigenagaKaoEncode=FR.face_encodings(shigenagaKao)[0]
knownEncodings=[okamuraKaoEncode,shigenagaKaoEncode]
names=['Okamura Hisanori','Shigenaga Tomoko']
```

追加

顔を識別する (1/2)

- 画像を入力すると、中に写っている顔を認識し、知っている人の顔であれば その名前を出力できるようにします

openCV-2.py

```
shigenagaKao=FR.load_image_file('shigenaga_tomoko.jpgのファイルパス')
shigenagaKaoEncode=FR.face_encodings(shigenagaKao)[0]
knownEncodings=[okamuraKaoEncode,shigenagaKaoEncode]
names=['Okamura Hisanori','Shigenaga Tomoko']
```

追加

```
unknownKao=FR.load_image_file('u2.jpgのファイルパス')
unknownKaoBGR=cv2.cvtColor(unknownKao,cv2.COLOR_RGB2BGR)
faceLocations=FR.face_locations(unknownKao)
unknownEncodings=FR.face_encodings(unknownKao,faceLocations)
```

顔を識別する (2/2)

- knownEncodings と unknownEncodings について順番に比較して
  同じ人かどうかを比べます

openCV-2.py

```
faceLocations=FR.face_locations(unknownKao)
unknownEncodings=FR.face_encodings(unknownKao,faceLocations)
for faceLocation, unknownEncodings in zip(faceLocations, unknownEncodings):
    top,right,bottom,left=faceLocation
    cv2.rectangle(okamuraKaoBGR,(left,top),(right,bottom),(255,0,0),2)
    name='Unknown Person'
    matches=FR.compare_faces(knownEncodings,unknownEncodings)
    print(matches)
```

追加

- 実行して **True** が出た部分が同じ人と判定されています。何個ありますか？

結果を可視化する

- 知っている人と認識された部分には、その人の名前を付けましょう

openCV-2.py

```
unknownEncodings=FR.face_encodings(unknownKao,faceLocations)
for faceLocation, unknownEncodings in zip(faceLocations, unknownEncodings):
    top,right,bottom,left=faceLocation
    cv2.rectangle(okamuraKaoBGR,(left,top),(right,bottom),(255,0,0),2)
    name='Unknown Person'
    matches=FR.compare_faces(knownEncodings,unknownEncodings)
    if True in matches:
        matchIndex=matches.index(True)
        name=names[matchIndex]
    cv2.putText(unknownKaoBGR,name,(left,top),font,.5,(0,0,255),2)
cv2.imshow('my Window',unknownKaoBGR)
```

追加

顔認識プログラム、完成！

エンコード済みの顔を別の画像から検出する
ことができました！

エンコードしていない人の顔が載っている写真
でも試してみましょう。
精度は100%ではありません。

更に次のステップに進める方は、画像ファイルから
ではなく、Webカメラで撮った画像から顔を認識
するコードを書いてみましょう。

まずは自分でやってみて、分からなかったら
homeworkフォルダにある答えを参考にしてください

SHIP Workshop Deck

Raspberry Piを使って画像認識に挑戦しよう(2)

ご清聴ありがとうございました

社会のいわ！ をここから

⇦アンケートに
ご協力をお願い
いたします。

SBS情報システム