ラズベリー・パイで画像認識

目次

1　ウェブカメラの接続

Raspberry Pi にウェブカメラを接続します。

1.1　FSWEBCAM のインストール

一般的な USB の Web カメラを使ってラズベリーパイ上で写真や動画を撮るために、fswebcam パッケージをインストールします。SSH で接続した端末から以下のコマンドを実行します。

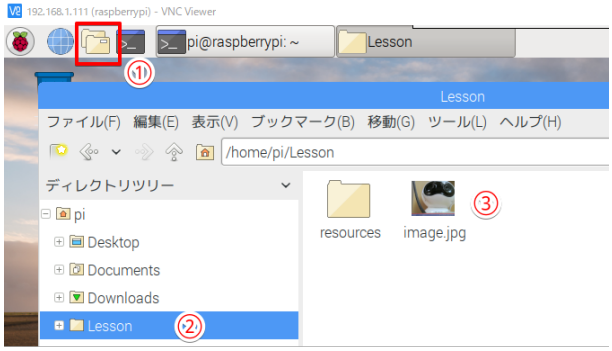
```
sudo apt-get install fswebcam
```

1.2　ウェブカメラのテスト

ウェブカメラで画像が取得できるか確認します。以下のコマンドでウェブカメラから画像が取得できます。

```
cd ~/Lesson
fswebcam image.jpg
```

取得した画像（ /Lesson/image.jpg）を VNC ビューワで確認し、画像が取得できているかどうかを確認します。



①クリックすると、ファイルマネジャーが開きます。
②「Lesson」フォルダを指定します。
③クリック（又は右クリックで「イメージビューワ」を選択）すると、画像を確認できます。
※アイコンはイメージの画像でない場合もありますので、イメージビューワで開いて確認します。

（参考）オプションの指定により、画像の解像度の指定や、画像下部のバナーの消去ができます。

```
fswebcam -r 1280x720 --no-banner image2.jpg
```

2　音声出力先の設定

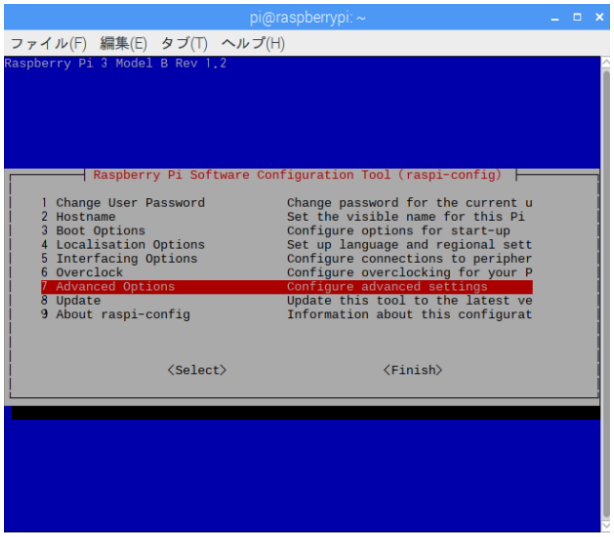
Raspberry Pi にスピーカーを接続します。電源が必要なタイプのスピーカーの場合には、スピーカーに電源を供給します。

2.1　音声出力先の設定

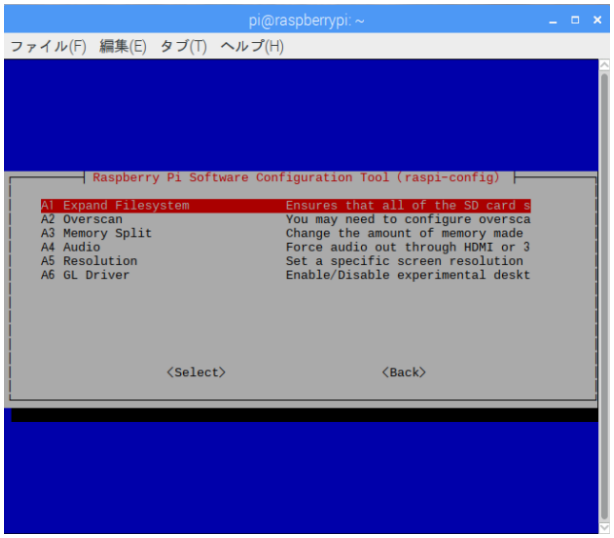
ターミナルを起動して、

```
sudo raspi-config
```

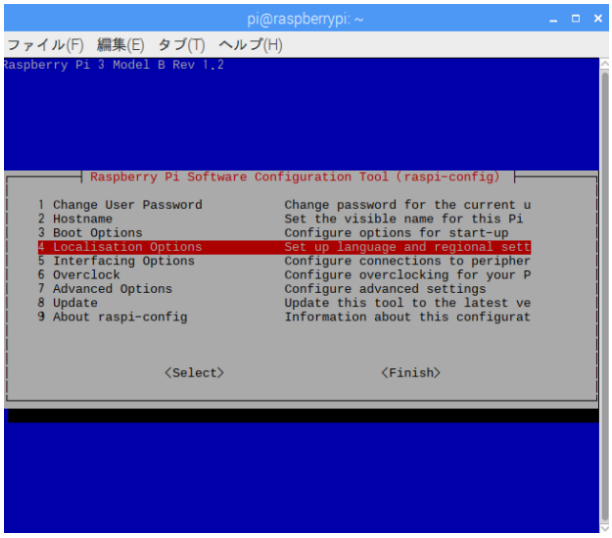
と入力します。



「Advanced Options」を選択して改行します。



「Audio」を選択して改行します。



「Auto」の場合には、HDMI → HeadPhone の順で接続を確認し出力を行います。HeadPhone に外部スピーカーをつなげている場合には、必要に応じて「Force 3.5mm ('headphone') jack」を選択します。

2.2 音声出力先のテスト

```
speaker-test -t sine -f 1000
```

1kHz の正弦波を出力されます。指定したデバイスから音が鳴るかを確認します。外部スピーカーにボリュームが付いている場合には、ボリュームを調整して音がでるようにします。

内蔵の WAV ファイルを用いてテストすることもできます。

```
speaker-test -t wav
```

3 OPEN CV のインストール

この授業で通り行ってきた通り、anaconda を使って OpenCV をインストールできればよいのですが、anaconda は、ARMv7 版は対応していません。そのため、OpenCV のソースをダウンロードしコンパイルを行います。かなり時間がかかります（RaspberryPi3 で 2 時間程度)。

3.1 依存関係のあるパッケージのインストール

インストールされているパッケージを更新します。

```
sudo apt-get update
(sudo apt-get upgrade)
```

OpenCV のビルドに必要な cmake 等のツールをインストールします。

```
sudo apt-get install build-essential cmake pkg-config
```

画像の入出力に必要なパッケージをインストールします。

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-
   dev
```

ビデオの入出力に必要なパッケージをインストールします。

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
   libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
```

OpenCV は、highgui というサブモジュールを利用して画像の表示などを行っています。このモジュールをコンパイルするのに、libgtk2.0-dev　が必要となります。

```
sudo apt-get install libgtk2.0-dev
```

OpenCV 内での行列演算などの最適化のために、以下のモジュールをインストールします。

```
sudo apt-get install libatlas-base-dev gfortran
```

OpenCV のコンパイルのために、python2.7 と python3 のヘッダーファイルをインストールします。

```
sudo apt-get install python2.7-dev python3-dev
```

3.2　OpenCV のソースコードのダウンロード

OpenCV のリポジトリから、OpenCV3.1.0 のソースコードをダウンロードして解凍します。

```
cd ~/Lesson
wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
unzip opencv.zip
```

OpenCV のフルインストールを行うためには、opencv_contrib も必要となります。opencv_contrib は OpenCV の「extra」モジュールを集めたものです。これらの module が成熟してポピュラーになると、OpenCV リポジトリに移されます。OpenCV と同じバージョンのソースコードをダウンロードして解凍します。

```
wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/
   archive/3.1.0.zip
unzip opencv_contrib.zip
```

3.3　OpenCV をコンパイルするための環境の設定

3.3.1　pip のインストール

python のパッケージ管理ツールである「pip」をインストールします。pip を用いて python のパッケージのイントール等の管理を行います。

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
```

3.3.2 virtualenv のインストール

プロジェクト毎に、python を実行する環境を作成し、管理するために、virtualenv と virtualenvwrapper をインストールします。

```
sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/.cache/pip
```

エディタで /.profile を開いて、WORKON_HOME という環境変数を設定します。※ nano を使っていますが、vim や emacs でもかまいません。

```
nano ~/.profile
```

nano の画面での移動は、上下左右のカーソルキーで行えます。
文字を入力すると、ファイルの編集を行うことができます。
「ctrl+g」でヘルプを表示します。
「ctrl+o」でファイルを保存します。
「ctrl+x」で編集を終了します。

/.profile の最後の行に以下を追加します。

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

変更を反映させます。

```
source ~/.profile
```

3.3.3 仮想環境の作成

「cv」という python の仮想環境を作成します。

```
mkvirtualenv cv -p python2
```

仮想環境の作成は、最初の 1 回のみです。一度環境を作成したら、作業は、その作成した環境に入って行います

3.4 OpenCV のインストール

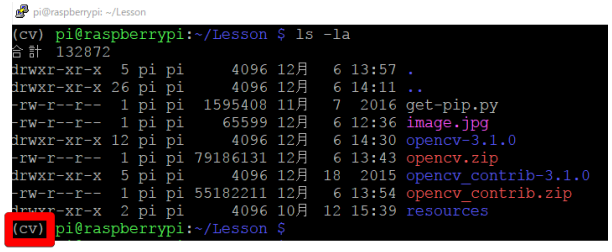
3.4.1 仮想環境に入る

作成した仮想環境に入ります。。新しいターミナルを開いたり、コンピュータを再起動した場合にも、作成した環境に入ってから作業を行います。同じです。再度、環境を作成する必要はありません。

```
source ~/.profile
workon cv
```

仮想環境に入ると、コマンドラインの先頭に括弧つきで仮想環境の名前が表示されます。

仮想環境 cv 内でインストールされた python パッケージは、cv 内でのみ有効です。またグローバル環境でインストールされた python パッケージは、仮想環境からはアクセスできません。



次のように表示されていることを確認します。「cv」という表示がない場合には、仮想環境 cv に入っていません。仮想環境に入って操作します。

3.4.2 numpy のインストール

仮想環境 cv 内で、Numpy をインストールします。Numpy は、Python の数値計算用パッケージです。インストールには多少時間（１２〜１３分程度）がかかります。

```
pip install numpy
```

3.4.3 OpenCV のコンパイルとインストール

仮想環境 cv 内で、OpenCV のコンパイルとインストールを行います。

```
cd ~/Lesson/opencv-3.1.0/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=OFF \
    -D OPENCV_EXTRA_MODULES_PATH=~/Lesson/opencv_contrib-3.1.0/modules
       \
    -D ENABLE_PRECOMPILED_HEADERS=OFF \
    -D BUILD_EXAMPLES=OFF ..
```

OpenCV のコンパイルを行います。

```
make -j4
```

-j は、コンパイルに利用するコアの数です。４コアでのコンパイルでエラーがでる場合には、先のコンパイルでできた不要なファイルを削除してから、再度、１コアでコンパイルします。

```
make clean
```

```
make
```

OpenCV のインストールを行い、共有ライブラリの依存関係情報を更新します。

```
sudo make install
sudo ldconfig
```

3.4.4 仮想環境内からのリンクの作成

(python2) OpenCV のインストールにより、/usr/local/lib/python2.7/site-packages にパッケージがインストールされます。

```
ls -l /usr/local/lib/python2.7/site-packages/
total 1852
-rw-r--r-- 1 root staff 1895772 Mar 20 20:00 cv2.so
```

仮想環境 cv 内からもパッケージを参照できるように、リンクを張ります。

```
cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

3.5 インストールの確認

新しいターミナルを開きます。

```
source ~/.profile
workon cv
python
>>> import cv2
>>> cv2.__version__
'3.1.0'
>>>quit()
```

import をしてエラーが出なければ、無事にインストールが出来ています。

インストールできでいることが確認できたら、ディスクスペースを確保するために opencv-3.1.0 と、opencv_contrib-3.1.0 を削除することができます。

```
cd ~/Lesson
rm -rf opencv-3.1.0 opencv_contrib-3.1.0
```

ただし、削除すると、再度コンパイルするのに非常に時間がかかるので、正しくインストールされていることを確認してから削除するようにしましょう。

4 ディープラーニングのフレームワーク、その他のモジュールのインストール

4.1 TensorFlow のインストール

```
wget https://github.com/samjabrahams/tensorflow-on-raspberry-pi/
   releases/download/v1.1.0/tensorflow-1.1.0-cp27-none-linux_armv7l.whl
pip install tensorflow-1.1.0-cp27-none-linux_armv7l.whl
```

4.2 Keras のインストール

Keras は、TensorFlow などをバックエンドとして利用する、Deep Learning のフレームワークです。

```
pip install keras
```

4.3 h5py のインストール

```
sudo apt-get install libhdf5-dev
pip install h5py
```

4.4 PIL のインストール

```
pip install Pillow
```

5 InceptionV3 のインストール

inception_v3 のモデルをインストールします。

```
git clone https://github.com/fchollet/deep-learning-models.git
```

inception_v3 のモデルを使って画像認識を行います。

```
cd deep-learning-models
wget https://upload.wikimedia.org/wikipedia/commons/6/63/
   African_elephant_warning_raised_trunk.jpg -O elephant.jpg
python inception_v3.py
Using TensorFlow backend.
Predicted: [[('n02504013', 'Indian_elephant', 0.87686646), ('n01871265
   ', 'tusker', 0.044712357), ('n02504458', 'African_elephant',
```

```
   0.02874627), ('n02398521', 'hippopotamus', 0.0072720782), ('
   n02092339', 'Weimaraner', 0.0020943223)]]
```

実行すると、学習済みの InceptionV3 のモデルをダウンロードします。デフォルトでは、象の画像の認識を行います。「elephant.jpg」というファイルを用意して、認識させます。画像を解析すると、「ラベル番号・ラベル名・予測値」が多次元配列で返ってきます。

```
cd deep-learning-models
wget ＜画像のURL＞ -O elephant.jpg
```

で画像を保存できますので、他の画像を探して、試してみましょう。

6 OpenCV 画像の解析

OpenCV で読み込んだ画像を認識するように、プログラムを修正します。if __name__ == '__main__以降を以下のように書き換えます。

```
cp inception_v3.py answerthings.py
nano answerthings.py
```

answerthings.py

```
398 import cv2
399 if __name__ == '__main__':
400     model = InceptionV3(include_top=True, weights='imagenet')
401     cam = cv2.VideoCapture(0)
402     print("Start")
403 
404     while(True):
405         ret, frame = cam.read()
406         cv2.imshow("Show FLAME Image", frame)
407 
408         k = cv2.waitKey(1)
409         if k == ord('s'):
410             cv2.imwrite("output.png", frame)
411             # img_path = 'elephant.jpg'
412             img_path = "output.png"
413             img = image.load_img(img_path, target_size=(299, 299))
414             x = image.img_to_array(img)
415             x = np.expand_dims(x, axis=0)
416 
417             x = preprocess_input(x)
418             preds = model.predict(x)
```

```
419             recognize = decode_predictions(preds)
420             #print('Predicted:', decode_predictions(preds))
421             print('Label:', recognize[0][0][1])
422
423         elif k == ord('q'):
424             break
425     cam.release()
426     cv2.destroyAllWindows()
```

```
python answerthings.py
```

これで、「s」キーを押すと、画像を「output.png」に保存し、そのファイルの解析を行います。「q」キーを押すと、プログラムを終了します。

7 英語ラベルの発話

画像を認識すると、英語のラベルが返ってきますので、それを発声します。英語の発声には、espeak を利用します。

```
sudo apt-get install espeak
jack_control start
pulseaudio --start
espaek “Hello”
```

subprocess を使って、python で上記の espaek コマンドを実行します。上記のプログラムを以下のように書き換えます。

```
cp answerthings.py speakthings.py
nano speakthings.py
```

speakthings.py

```
1 # linuxコマンドを使うためにsubprocessをインポート
2 import subprocess
3
4 # 422行目以降に以下を追記
5 speak = "This is a " + recognize[0][0][1]
6 subprocess.check_output(["espeak", "-k5", "-s150", speak])
```

```
python speakthings.py
```

8 日本語の発話

①英語ラベルを日本語に変換②日本語を発声というプロセスで、日本語の発声をするようにします。

8.1 英語ラベルを日本語に変換

英語ラベルの数が限られているので、変換用の辞書を使って変換します。

```
git clone https://gist.github.com/PonDad/4
    dcb4b242b9358e524b4ddecbee385e9
cd 4dcb4b242b9358e524b4ddecbee385e9
```

imagenet_class_index.json

```
[
    {
        "num": "n01440764",
        "en": "tench",
        "ja": "テンチ"
    },
    {
        "num": "n01443537",
        "en": "goldfish",
        "ja": "金魚"
    },
・・・
]
```

という構造になっています。

app.py

```
1 import sys
2 import json
3
4 with open('imagenet_class_index.json', 'r') as f:
5   obj = json.load(f)
6   for i in obj:
7     if i['en'] == sys.argv[1]:
8         print(i['ja'])
```

```
python app.py goldfish
金魚
```

8.2 日本語を発声

open-jtalk を使って発声します。open-jtalk をインストールします。

```
sudo apt-get install open-jtalk open-jtalk-mecab-naist-jdic hts-voice-
   nitech-jp-atr503-m001
```

発声に使う音声ファイル（.htsvoice）をダウンロードし、抽出します。

```
wget https://sourceforge.net/projects/mmdagent/files/MMDAgent_Example/
   MMDAgent_Example-1.6/MMDAgent_Example-1.6.zip/download -O
   MMDAgent_Example-1.6.zip
unzip MMDAgent_Example-1.6.zip MMDAgent_Example-1.6/Voice/*
```

抽出した音声ファイルを設置します。

```
sudo cp -r MMDAgent_Example-1.6/Voice/mei/ /usr/share/hts-voice
```

jtalk.py

```
1  #coding: utf-8
2  import subprocess
3  from datetime import datetime
4
5  def jtalk(t):
6      open_jtalk=['open_jtalk']
7      mech=['-x','/var/lib/mecab/dic/open-jtalk/naist-jdic']
8      htsvoice=['-m','/usr/share/hts-voice/mei/mei_normal.htsvoice']
9      speed=['-r','1.0']
10     outwav=['-ow','open_jtalk.wav']
11     cmd=open_jtalk+mech+htsvoice+speed+outwav
12     c = subprocess.Popen(cmd,stdin=subprocess.PIPE)
13     c.stdin.write(t)
14     c.stdin.close()
15     c.wait()
16     aplay = ['aplay','-q','open_jtalk.wav']
17     wr = subprocess.Popen(aplay)
18
19 def say_datetime():
20     d = datetime.now()
21     text = '%s月%s日、%s時%s分%s秒' % (d.month, d.day, d.hour, d.minute
           , d.second)
22     jtalk(text)
```

```
23
24 if __name__ == '__main__':
25     say_datetime()
```

```
python jtalk.py
```

python モジュールとして利用する場合

```
import jtalk
jtalk.jtalk(u'日本語を話します')
```

8.3 2つをまとめる

上記の２つを合わせて、見たものを日本語で発声します。

英語のラベルを日本語に変換する関数

```
1 def en_to_ja(en_text):
2     with open('imagenet_class_index_with_ja.json', 'r') as f:
3         obj = json.load(f)
4         for i in obj:
5             if i['en'] == en_text:
6                 return i['ja']
7     return ""
```

日本語の発声

```
1 recognize_rabel_ja = en_to_ja(recognize_rabel)
2 ja_text_to_speak = u'これは' + recognize_rabel_ja + u'だよ'
3 jtalk.jtalk(ja_text_to_speak.encode('utf-8'))
```

の２つを追加します。上記の２つをまとめて以下の通りに修正します。

```
cp speakthings.py speakzou.py
nano speakzou.py
```

speakzou.py

```
1 import subprocess
2 import json
3 import sys
4 import jtalk
5
6 def en_to_ja(en_text):
7     with open('imagenet_class_index_with_ja.json', 'r') as f:
```

```
8           obj = json.load(f)
9           for i in obj:
10              if i['en'] == en_text:
11                  return i['ja']
12      return ""
13
14  if __name__ == '__main__':
15      model = InceptionV3(include_top=True, weights='imagenet')
16      cam = cv2.VideoCapture(0)
17      print("Start")
18
19      while(True):
20          ret, frame = cam.read()
21          cv2.imshow("Show FLAME Image", frame)
22
23          k = cv2.waitKey(1)
24          if k == ord('s'):
25              cv2.imwrite("output.png", frame)
26              # img_path = 'elephant.jpg'
27              img_path = "output.png"
28              img = image.load_img(img_path, target_size=(299, 299))
29              x = image.img_to_array(img)
30              x = np.expand_dims(x, axis=0)
31
32              x = preprocess_input(x)
33              preds = model.predict(x)
34              recognize = decode_predictions(preds)
35              recognize_rabel = recognize[0][0][1]
36              recognize_rabel_ja = en_to_ja(recognize_rabel)
37              ja_text_to_speak = u'これは' + recognize_rabel_ja + u'だよ'
38              jtalk.jtalk(ja_text_to_speak.encode('utf-8'))
39              # print('Predicted:', decode_predictions(preds))
40              # print('Label:', recognize[0][0][1])
41
42              #speak = "This is a " + recognize[0][0][1]
43              #subprocess.check_output(["espeak", "-k5", "-s150", speak])
44              #print(speak)
45
46          elif k == ord('q'):
47              break
```

```
48      cam.release()
49      cv2.destroyAllWindows()
```

```
python speakzu.py
```

として、実行してみましょう。