AITCシニア勉強会

# OpenCV入門
## ～ラズパイ３でOpenCVとTensorFlowを動かしてみよう～

2019年4月13日

先端IT活用推進コンソーシアム
クラウド・テクノロジー活用部会
リーダー　荒本道隆

# この資料の目的

- 目的
  - カメラ映像の扱い方を知る
  - OpenCVで顔認識
  - TensorFlowも動かしてみる
- 今回のゴール
  - OpenCVによる顔画像自動収集
  - TensorFlowによる画像認識
- 準備するもの
  - Raspberry PI 3
  - USB接続Webカメラ（電気屋のワゴンセールで2,000円以下）
  - PC（Windows/Mac）
  - Internet接続

# Pythonを使用する上での注意事項

- Python2か、Python3か、どちらがデフォルト？
  - Python2：歴史が長く、ライブラリも豊富
  - Python3：新しいものは「Python3のみ」も多い

- 何が問題なのか？
  - 「python」と打った場合、2か3かは、環境によって異なる
  - 「pip」も同様
  - 2専用ライブラリ、3専用ライブラリがある
  - プログラム中に「python」と書かれていると超困る

- 今回のRasbianには、最初からPython2が入っている
  - この資料では、python = Python2という前提です

# OpenCVとは

- **概要**

> 画像処理・画像解析および機械学習等の機能を持つC/C++、Java、Python、MATLAB用ライブラリ[1] [2]。プラットフォームとしてMac OS XやFreeBSD等全てのPOSIXに準拠したUnix系OS、Linux、Windows、Android、iOS等をサポートしている。
>
> Wikipediaより

- **できること**
  - **たくさんあるのでWikipedia参照**
    - http://ja.wikipedia.org/wiki/OpenCV
  - **Windows/Macで開発して、ラズパイで実行**
- **できないこと**
  - **音、赤外線、より高いCPUパワーを使った処理**
  - **参照：Microsoft Kinect, Intel Realsense**

# ラズパイでOpenCVを動かすために

- X-Window を使用
  - ラズパイにモニタを接続して、GUIを起動
  - パソコンにX-Windowサーバを導入
    - Windows：Xming を導入
      - 接続方法：TeraTermの『X Forwarding』を有効にする
      - もしくは：「export DISPLAY=WindowsのIPアドレス:0 」を設定
    - Mac：標準装備
      - 接続方法：ターミナルから「ssh -Y pi@IPアドレス」
- ブラウザのみを使用　← 今回はこれをメインで説明
  - 仮想X-Windowを画像化し、PCのブラウザで表示
  - 注意：操作はできません
  - 注意：SDカードとネットワークが高負荷になります
  - 画面が見えなくても、ゴールはクリアできます

# まずは、必要なものを導入

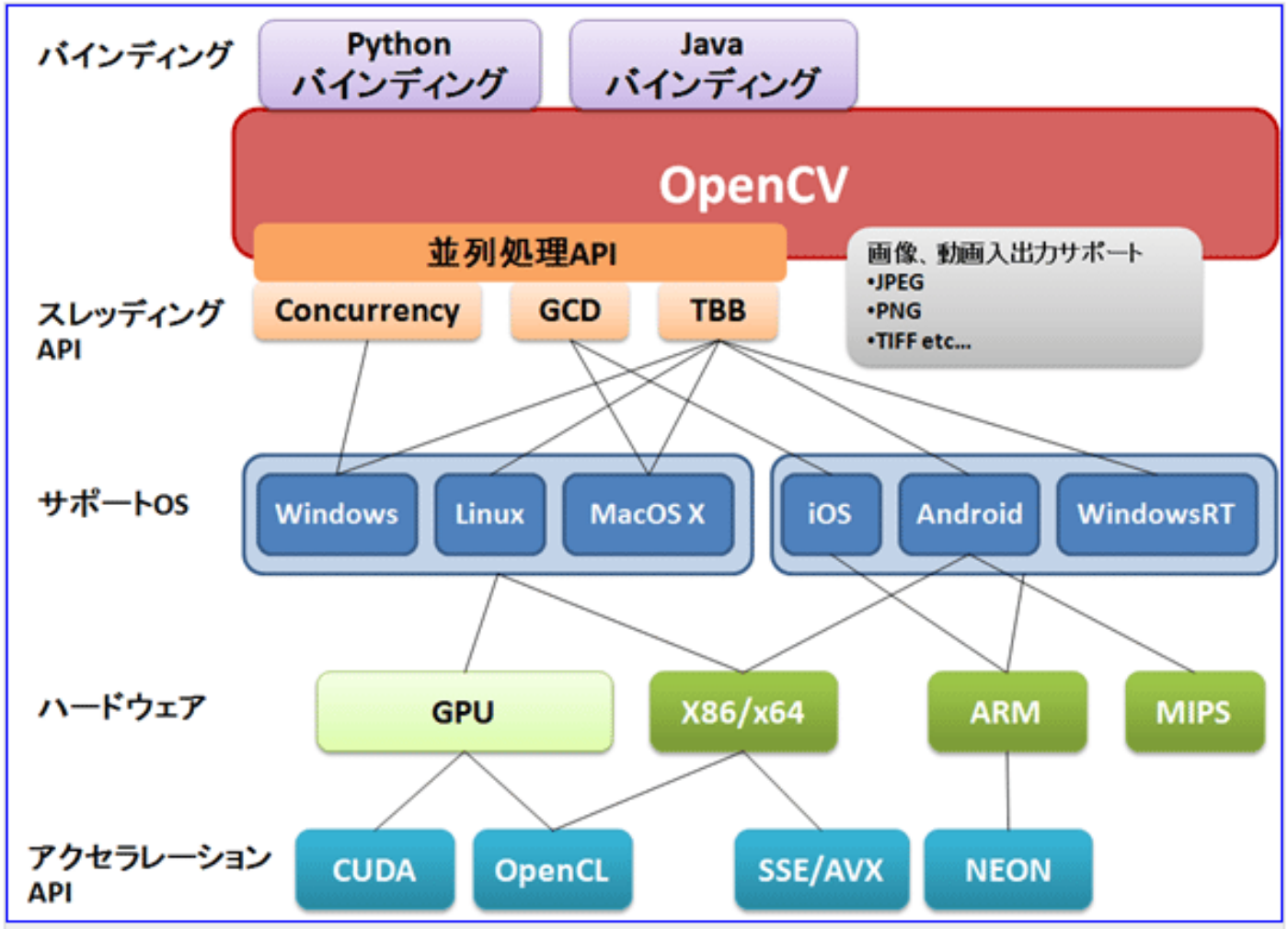
- 今回、使用するパッケージ一覧
  - OpenCV:libopencv-dev  python-opencv　　# 538MB
  - 仮想X-Window：xvfb  imagemagick　　# 80MB
  - Webサーバ：apache2  php　　# 17MB
- 今回使用するものをまとめてインストール

```
sudo apt update
sudo apt install libopencv-dev python-opencv xvfb imagemagick apache2 php
sudo apt install git python-pip
```

  - テキストファイルから１行づつコピペしてください
- 参照するサンプルをダウンロード＆解凍

```
cd /home/pi
wget http://aramoto.sakura.ne.jp/20180421/opencv-2.4.13.zip
unzip opencv-2.4.13.zip
wget http://aramoto.sakura.ne.jp/20180421/html.tar
tar xvf html.tar
```

# OpenCVを構成する技術




『図6. OpenCVを構成する技術』より
http://www.buildinsider.net/small/opencv/01

このページを読んでみる

# Python環境の確認方法：全OS共通

- ライブラリが正しく参照できているか確認する方法

```
[aramoto@localhost ~]$ python
Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import  numpy
>>> print  numpy.__version__
1.7.1
>>> import  cv2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named cv2
```

バージョンを確認

インストールに失敗している場合

# OpenCVを起動してみよう

・カメラが無いと、固定の画像が使われます

　　固定の画像を強制的に使用：引数に「1」を追加

# X-Windowが使える環境（上級者用）

- Windows
  - Xming というツールを導入して、起動
  - TereTerm で接続

```
export  DISPLAY=WindowsのIPアドレス:0
```

- Mac
  - ターミナルから接続

```
ssh  -Y  pi@ラズパイのIPアドレス
```

- 以下のコマンドでOpenCVのサンプルを起動
  - カメラ映像をそのまま表示

```
cd  /home/pi/opencv-2.4.13/samples/python2
python  video.py
```

**可能な人は、こっちの方法でやってください**

# ブラウザが使える環境（お手軽）

- ラズパイでWebサーバを起動（OSを起動するたび）

```
sudo  service  apache2  start
```

- Webコンテンツを準備

```
sudo  cp  -rp  /home/pi/html/viewx  /var/www/html/
```

- PCのブラウザから参照
  - http://ラズパイのIPアドレス/viewx/
- 仮想X-Window起動＆画像生成（OSを起動するたび）
  - 最後の「&」は、裏で実行し続ける、という意味

```
cd
wget  http://cloud.aitc.jp/20190413_RaspberryPi2/capture.sh
bash  capture.sh  &
```

  - コンソールに文字が出続けたら、ラズバイ３を再起動

注意：ブラウザからは操作不可

# ブラウザが使える環境（お手軽）－２

- 以下のコマンドでOpenCVのサンプルを起動
  - カメラ映像をそのまま表示

```
cd  /home/pi/opencv-2.4.13/samples/python2
export  DISPLAY=:1
python  video.py
```

終了は[Ctrl]+[C]

# 顔認証をしてみる

- 以下のコマンドでOpenCVのサンプルを起動
  - 顔認識

X-Windowの人は不要

```
export  DISPLAY=:1
python  facedetect.py
```

終了は[Ctrl]+[C]

うまく動作しない人は言ってください。
うまく動作する人
・ソースコードを見てみる
・他のサンプルを実行してみる

# X-Windowが使えない場合は

- 画面描画をコメントアウト
  - 赤字を追加
  - /home/pi/opencv-2.4.13/samples/python2/facedetect.py

```
56:         draw_str(vis, (20, 20), 'time: %.1f ms' % (dt*1000))
57:         # cv2.imshow('facedetect', vis)
```

- 処理自体は動作します
  - ただし、実行状況が目視できません

# カメラがない場合は

- 使用したい画像をラズパイにコピーする
  - 人物の正面顔が入っているもの
  - デフォルトでは ”lena.jpg” が入っています
- 画像ファイル名を変更
  - 赤字を実際のファイル名に合わせて変更
  - /home/pi/opencv-2.4.13/samples/python2/facedetect.py

```
38:     cam = create_capture(video_src, fallback='synth:bg=/home/pi/kao.jpg:noise=0.05')
```

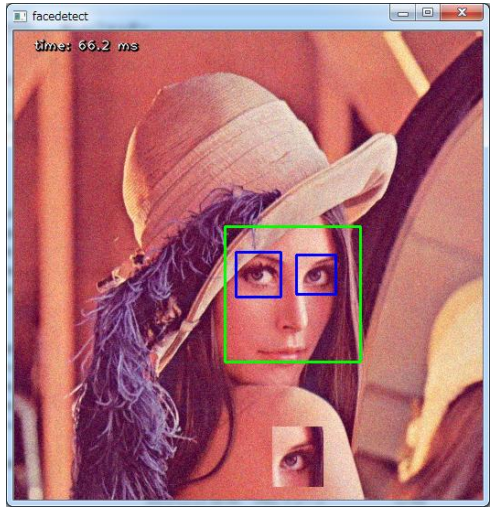
操作するものは、
X-Windowが必要

# OpenCV付属のデモの紹介

- カメラが無いと、固定の画像が使われます
  固定の画像を強制的に使用：引数に「1」を追加

# facedetect.py

- 顔と目を認識
  - 解説ページ
  - 事前に「顔」「顔じゃない」を学習済み
    - 学習結果が、XML形式で source/data に入っている
    - 他の物も、学習させれば認識できるようになる
      - ねこと画像処理： http://rest-term.com/archives/3131/




sample/cpp/lena.jpg

顔の外に目があっても、
認識しない作りになっている

目だけを肩に貼ってみた

# peopledetect.py

- 人っぽいものを検出
  - 検出精度はイマイチ
- 操作方法
  - 起動時に、引数で画像ファイルを指定

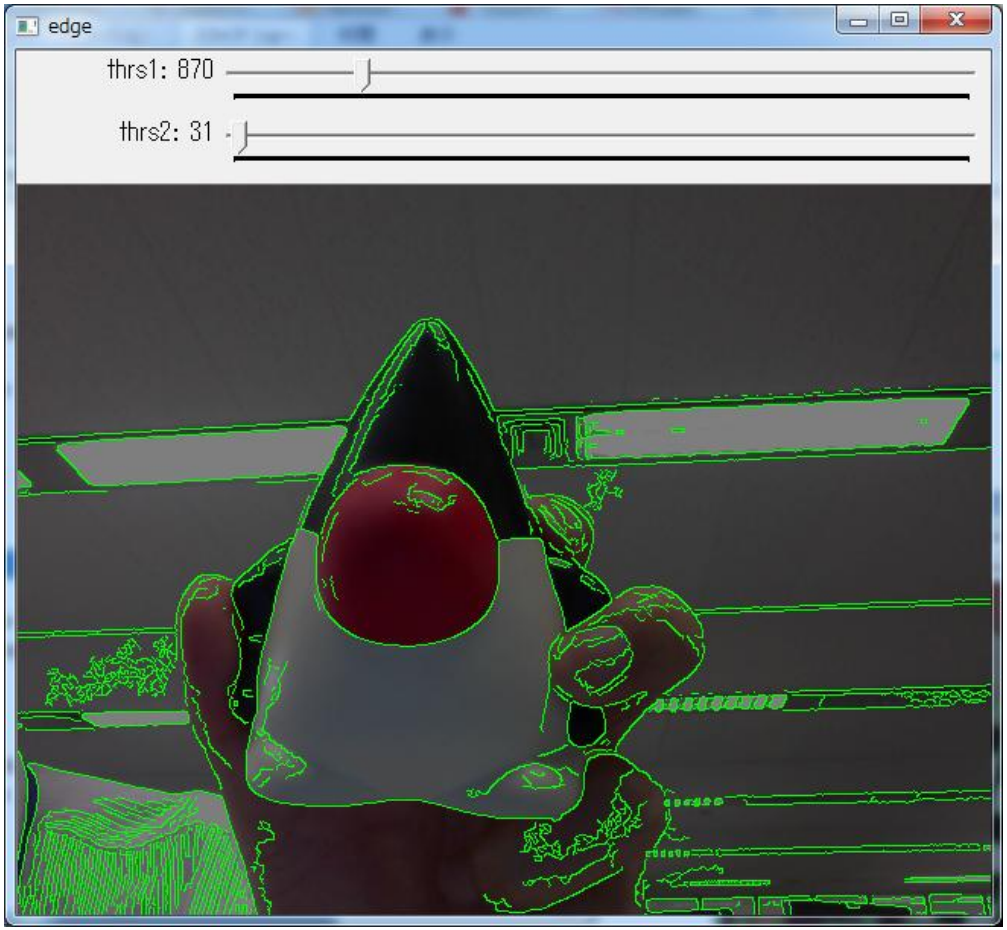
AITCニュースレター第3号の画像

# digits.py

- 機械学習による文字認識
  - 解説ページ
  - KNearest
  - SVM
  - digits_video.py で動画中の数字を探す

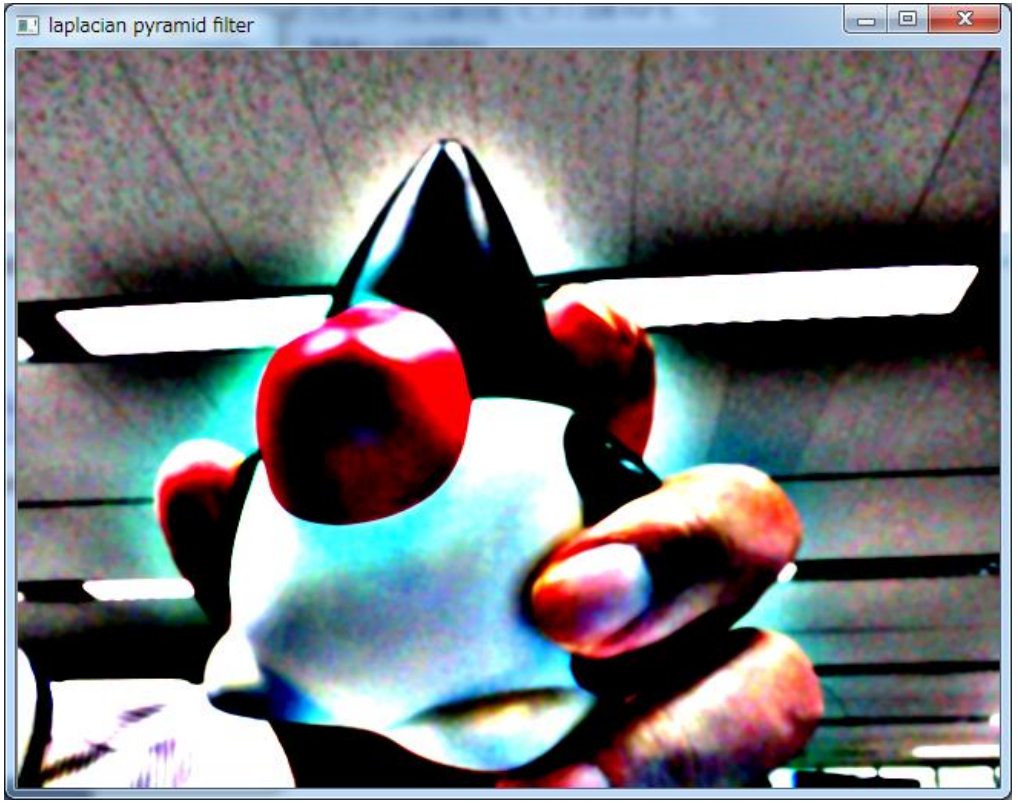
# edge.py

- 境界抽出
  - 『漫画メーカー』が作れそう



# lappyr.py

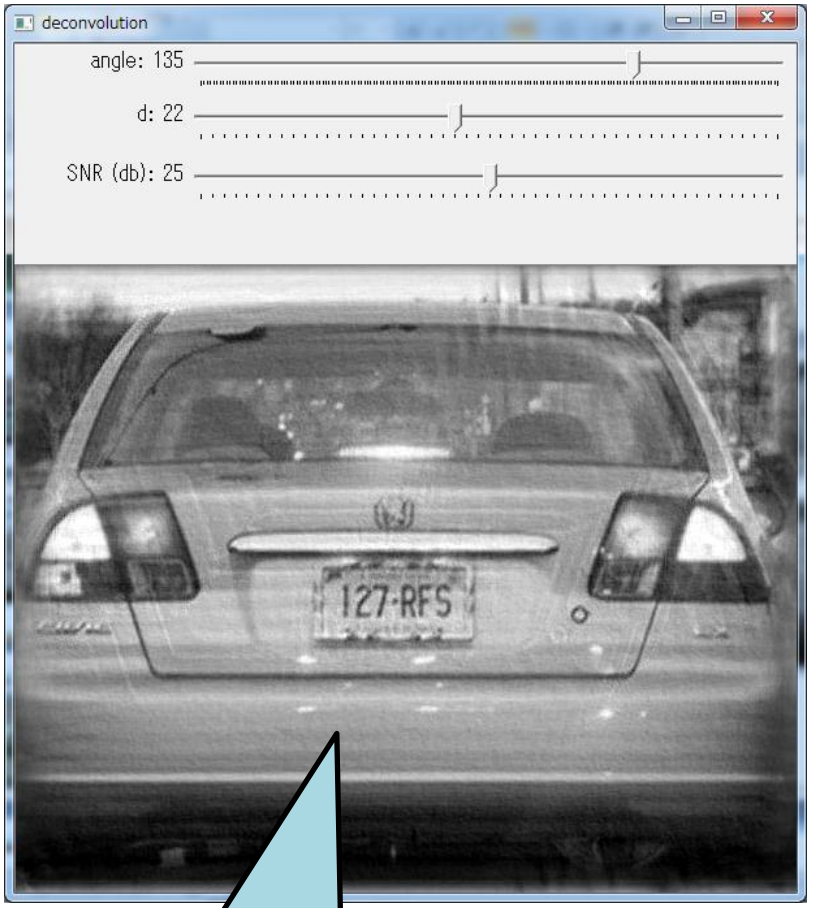
- 動画にエフェクトをかける



# deconvolution.py

- 「Convolution（畳みこみ）＝ブレ」を解除する

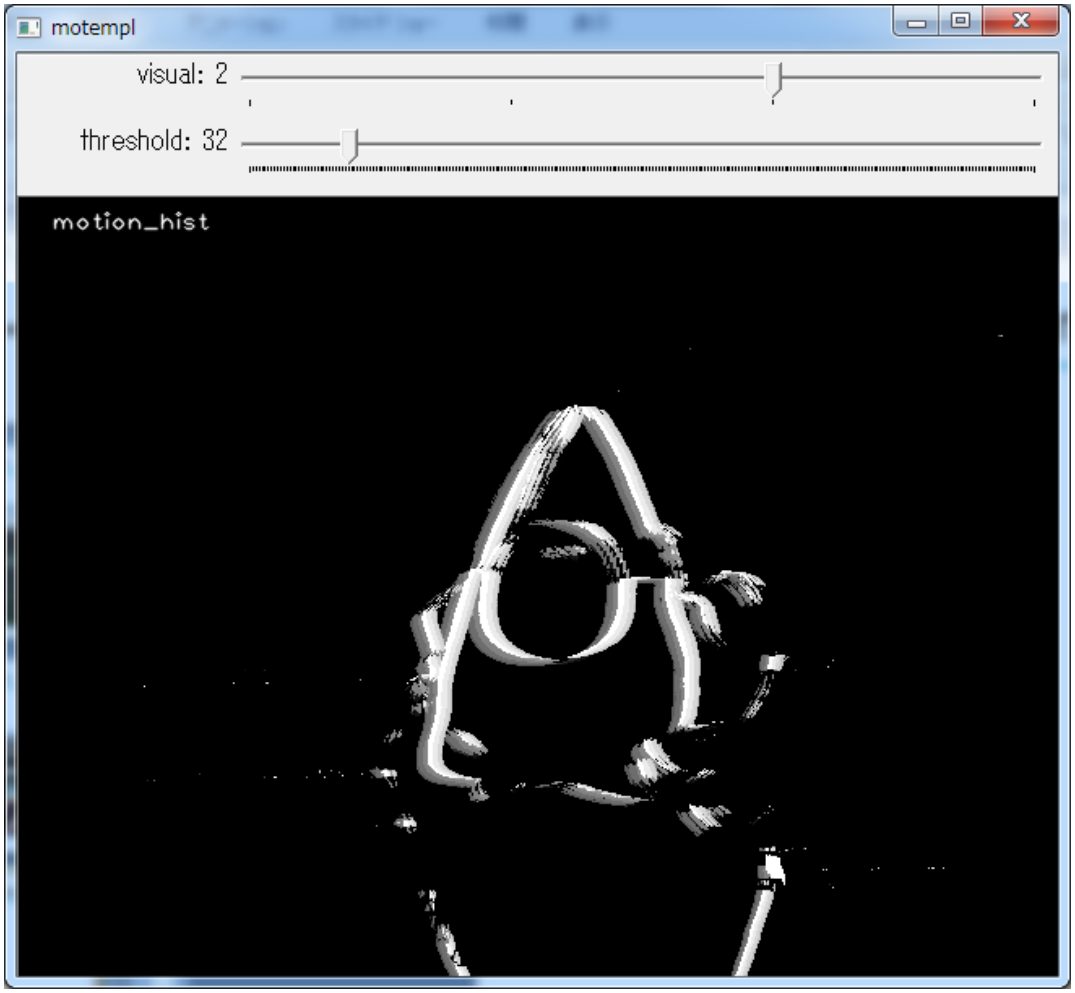

ナンバーが全く読めない




補正したら読めた

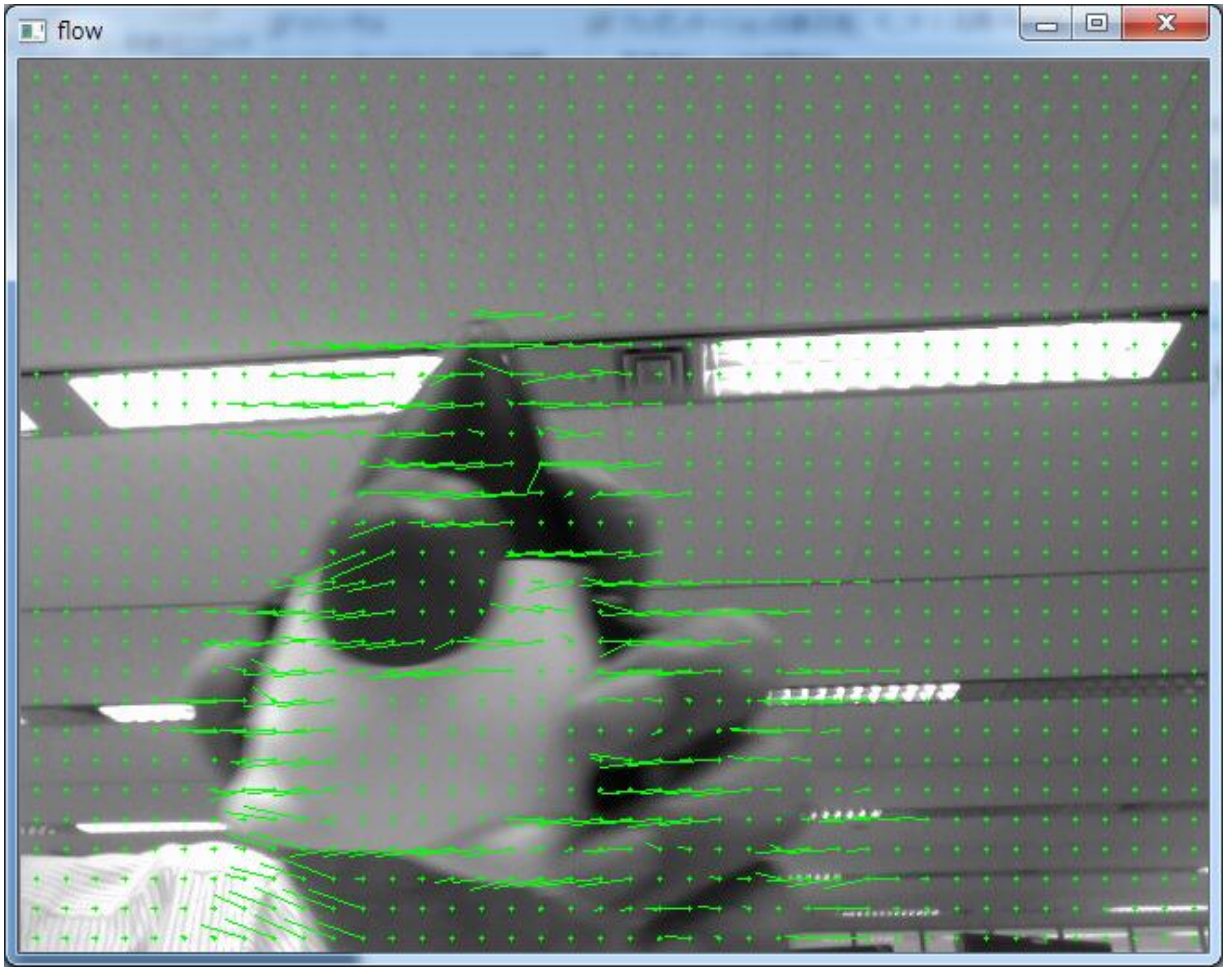
# motempl.py

- 動画の変化部分を検出



# opt_flow.py

- 動画中の物体が、どっちに動いているか



# camshift.py

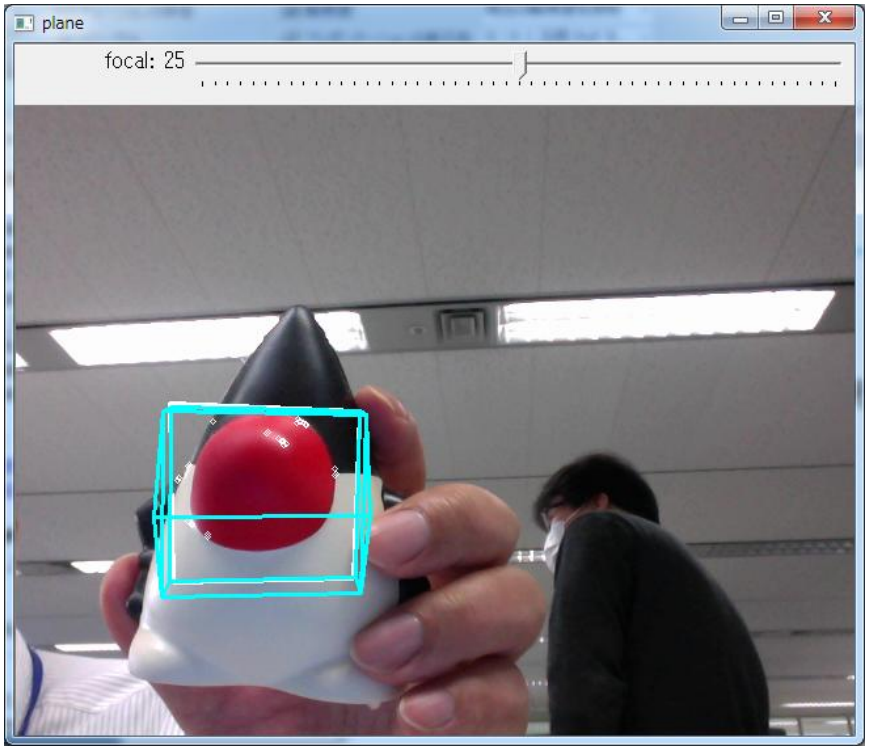
- 特定の色の領域を追いかける
- 操作方法
  - 追いかけたい色をマウスで範囲指定する

Dukeの赤い鼻を追いかけている

# plane_ar.py

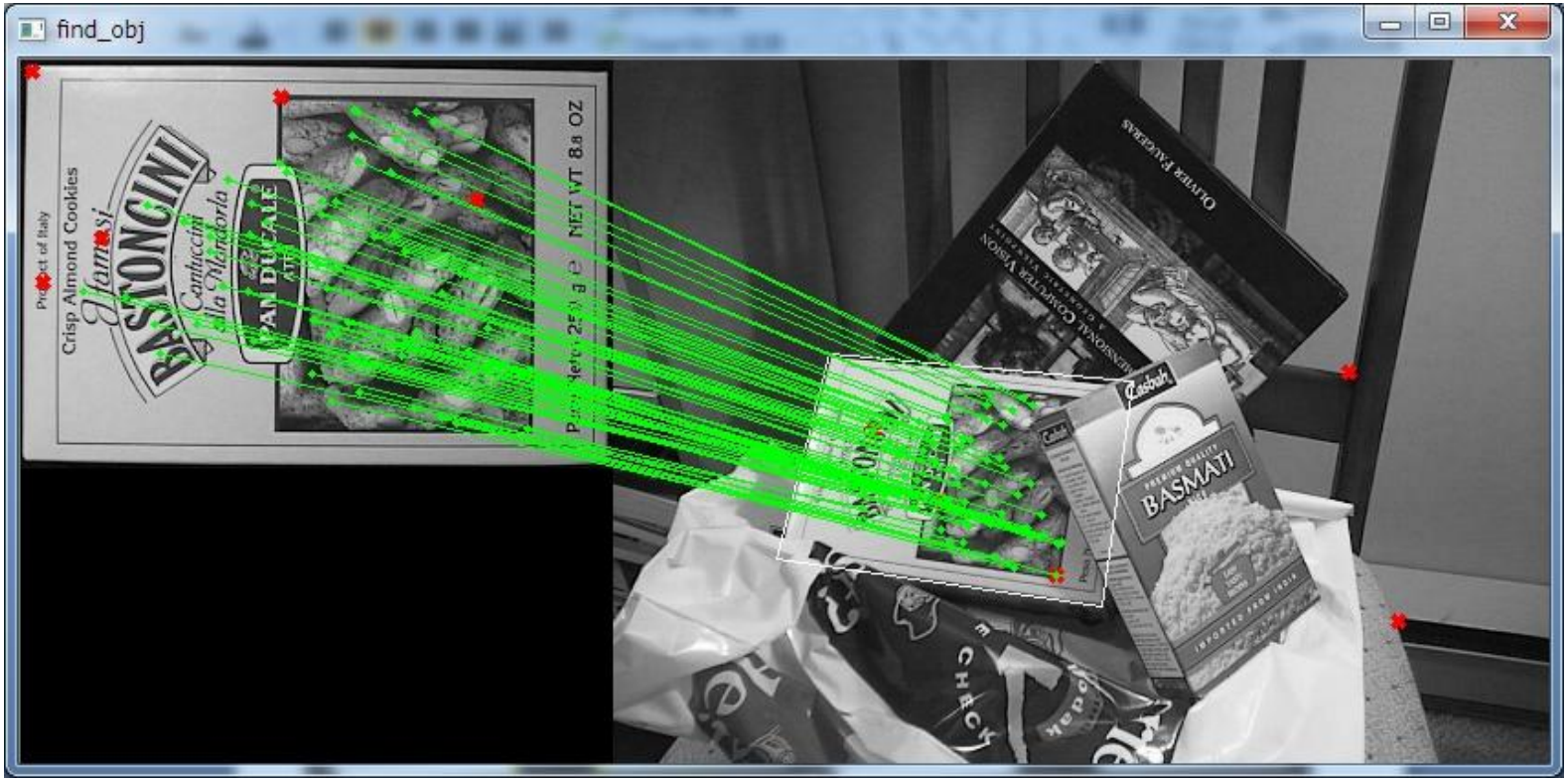
- ARマーカー無しのAR
- 操作方法
  - ARマーカーとして使いたい領域をマウスで選択

Dukeの鼻を
ARマーカーとして指定



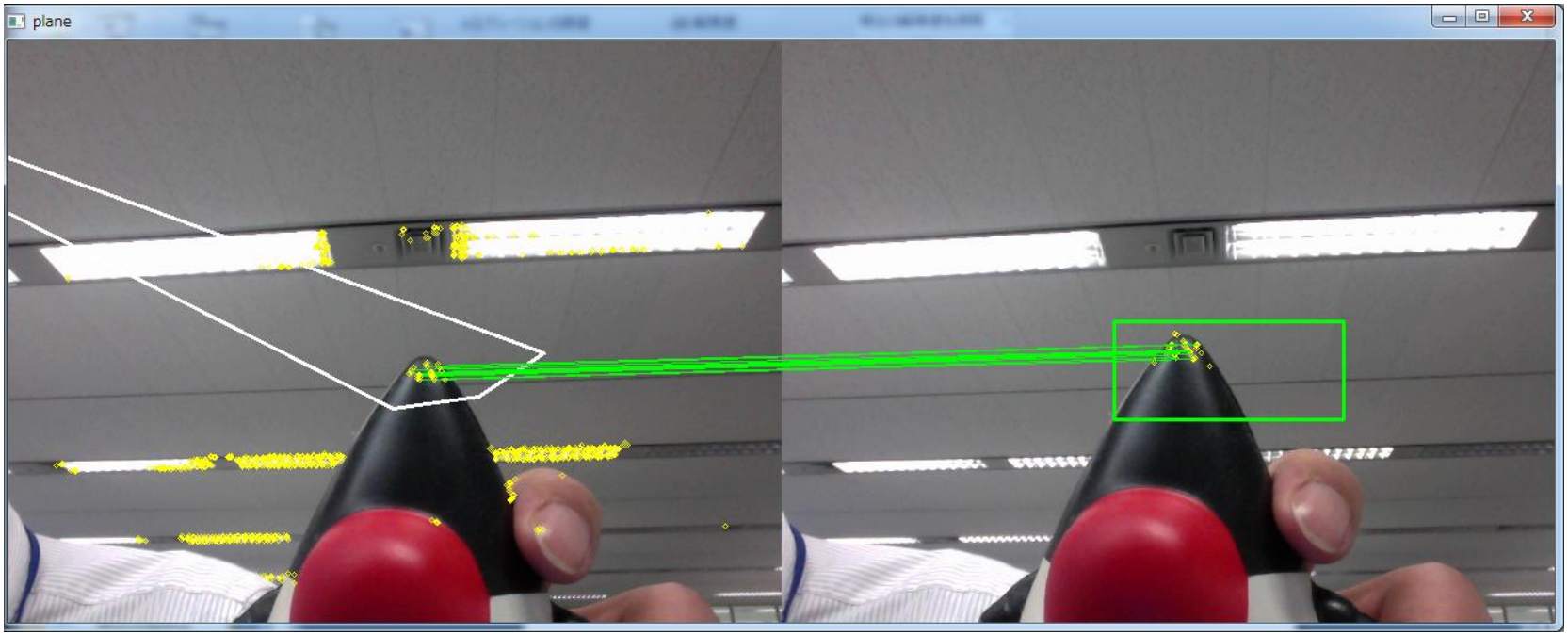
# find_obj.py

- 特徴点抽出を使って、同じ物体を探す
- 操作方法
  - 起動時に、引数で２つの画像ファイルを指定



# feature_homography.py

- 特徴点抽出を使って、同じ物体を探す
- 操作方法
  - 探したい領域をマウスで選択
  - 注意：特徴点が無いところを選択すると、終了する



# その他のトラッキング

- lk_track.py
- mosse.py
- plane_tracker.py

# 顔認識サンプルの拡張

- カメラが無いと、固定の画像が使われます
  固定の画像を強制的に使用：引数に「1」を追加

# 顔画像をひらすら収集－１

- 元ファイルをコピー

```
cd  /home/pi/opencv-2.4.13/samples/python2
cp  facedetect.py  facesave.py
```

- 顔画像を保存する機能を追加
  - ファイル名は、年月日_時分秒.jpg
  - 赤字の部分を追加

viコマンドに不慣れな人は、ダウンロードも可

```
rects = detect(gray, cascade)
        vis = img.copy()
        draw_rects(vis, rects, (0, 255, 0))
        for x1, y1, x2, y2 in rects:
            import datetime
            now = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
            cv2.imwrite("/var/www/html/faces/img/" + now +".jpg", vis[y1:y2, x1:x2])
            print (now + ".jpg saved")
```

```
cd  /home/pi/opencv-2.4.13/samples/python2
wget  http://cloud.aitc.jp/20190413_RaspberryPi2/facesave.py
```

# 顔画像をひらすら収集－２

- Webコンテンツを準備

```
sudo  cp  -rp  /home/pi/html/faces  /var/www/html/
```

- Apacheを起動

```
sudo  service  apache2  start
```

- 実行

```
export  DISPLAY=:1
python  facesave.py
```

- 収集結果をブラウザで参照
  - http://ラズパイのIPアドレス/faces/

# ラズパイ3にTensorFlowを導入

# TensorFlowとは

- **概要**

> 機械学習や数値解析、ニューラルネットワーク（ディープラーニング）に対応しており、GoogleとDeepMindの各種サービスなどでも広く活用されている。
> ……省略……
> 開発された目的は、人間が用いる学習や論理的思考と似たように、パターンや相関を検出し解釈するニューラルネットワークを構築、訓練することができるシステムのための要求を満たすためである[6]。

Wikipediaより

- **目的**
  - **巷で流行っているAIをラズパイ上で動かす**
- **注意事項**
  - **ラズパイ３のGPUに対応していないので、遅い**

# 目的その２

- すぐにAIが動作するボードが沢山出てくる
  - AIに限れば、ラズパイ３の100倍ほど早いはず？
- 今から、その準備をしておく


Intel Neural Compute Stick2
¥13,500-
購入可

Google Edge TPU
$149.99
予約不可？

nVidia Jetson Nano
¥12,312-
予約可

# ラズパイ３へのTensorFlowの導入

- 正当な方法

```
sudo pip install tensorflow
```

  - MemoryErrorで失敗

- 最新のバージョンを手動で導入

```
wget https://www.piwheels.org/simple/tensorflow/tensorflow-1.13.1-...
time sudo pip install tensorflow-1.13.1-...
```

  - 途中でコンパイルエラーが発生

- エラーが出ないバージョンを手動で導入：約1分

```
cd
wget https://www.piwheels.org/simple/tensorflow/tensorflow-1.1.0-cp27-none-linux_armv7l.whl
time sudo pip install tensorflow-1.1.0-cp27-none-linux_armv7l.whl
```

  - 「sudo pip install tensorflow==1.1.0」は約8分かかる

# ImageNetとは

- **概要**

  画像分類などの学習に使うための画像を集めたものです。
  この画像分類において、2015年2月にディープラーニングを使って
  「人間を超えた」ことで、ディープラーニングが大きく注目されました。

- URL
  - http://image-net.org/
- **今回は、全体に何が映っているかを判定**

# TensorFlowによる画像分類の導入

- 画像分類のサンプルコードをダウンロード

```
cd
wget   https://raw.githubusercontent.com/tensorflow/models/master/
tutorials/image/imagenet/classify_image.py
```

- 実行

```
time  python  classify_image.py
```

  - 初回はデータのダウンロードを行うので、遅い：約１分
  - ２回目はちょっとだけ早い：約40秒

- 画像ファイルを指定して実行

```
time  python  classify_image.py  --image_file="cat.jpg"
```

- 識別可能なもののリストを参照

```
less  /tmp/imagenet/imagenet_synset_to_human_label_map.txt
```

  - このファイルは編集してはならない

# OpenCV + TensorFlowによる画像分類

- せっかくカメラがあるので、カメラ画像を分類
- 遅い画像分類をどこまで高速化できるか？
  - classify_image.py が遅い原因
    - import tensorflow
    - 巨大なtarファイルのオープン
    - グラフのロード
    - TensorFlowでの画像分類

2回目以降は省略できれば、
めちゃめちゃ早くなる？！

- つまり、
  - OpenCVでカメラ画像を保存し、classify_image を呼び出すだけで、カメラに映ったものを分類できるはず
  - 注意：外部コマンド呼び出しだと、毎回40秒かかる

  - video.py と classify_image.py を合体

# OpenCV + TensorFlowによる画像分類

- 合体のための改造ポイント
  - video2.py
    - 表示を邪魔せず、キー入力も受け付ける（意外と難しい）
    - 画像保存を bmp → jpeg に変更 （メモリ渡しは面倒なのでパス）
    - classify_image2 を呼び出す機能を追加（保存したファイル名を渡す）
  - classify_image2.py
    - 外部から呼び出される機能を追加
    - ２回目以降は、重複する処理を省略する
    - ダウンロードしたデータの保存先を変更

# OpenCV + TensorFlowによる画像分類

- 今回は改造済みのものを用意

```
cd  /home/pi/opencv-2.4.13/samples/python2
wget  http://cloud.aitc.jp/20190413_RaspberryPi2/video2.py
wget  http://cloud.aitc.jp/20190413_RaspberryPi2/classify_image2.py
```

- 実行

```
export  DISPLAY=:1
python  video2.py
```

  - [Enter]キーを押すと、カメラに映ったものを保存して分類
  - １回目はちょっと遅いけど、２回目から早くなる
  - 専用ハードウエアを購入すれば、100倍早くなるかも？
  - パソコンで動かすと、どのくらいの速度か？
- 興味のある人は、元ソースとの差分を取ってみてください

```
diff  video.py  video2.py
diff  /home/pi/classify_image.py  classify_image2.py
```