

AITCシニア勉強会

# OpenCV入門

～ラズパイでOpenCVを動かしてみよう～

2018年4月21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で何ができるのか

- 今回のゴール

- 大量の顔画像の収集
- 顔の検出結果をクラウドに送信
- 画像処理

- 準備するもの

- Raspberry PI
- USB接続Webカメラ（電気屋のワゴンセールで2,000円以下）
  - 無ければ、適当な画像ファイル
- PC（Windows/Mac）

# 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。

## ● できること

Wikipediaより

### ○ 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

- 動作状況を確認するために：X-Window
  - 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」
- OSに依存しないやり方
  - X-Windowを画像化し、ブラウザでラズパイのGUIを表示
  - **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

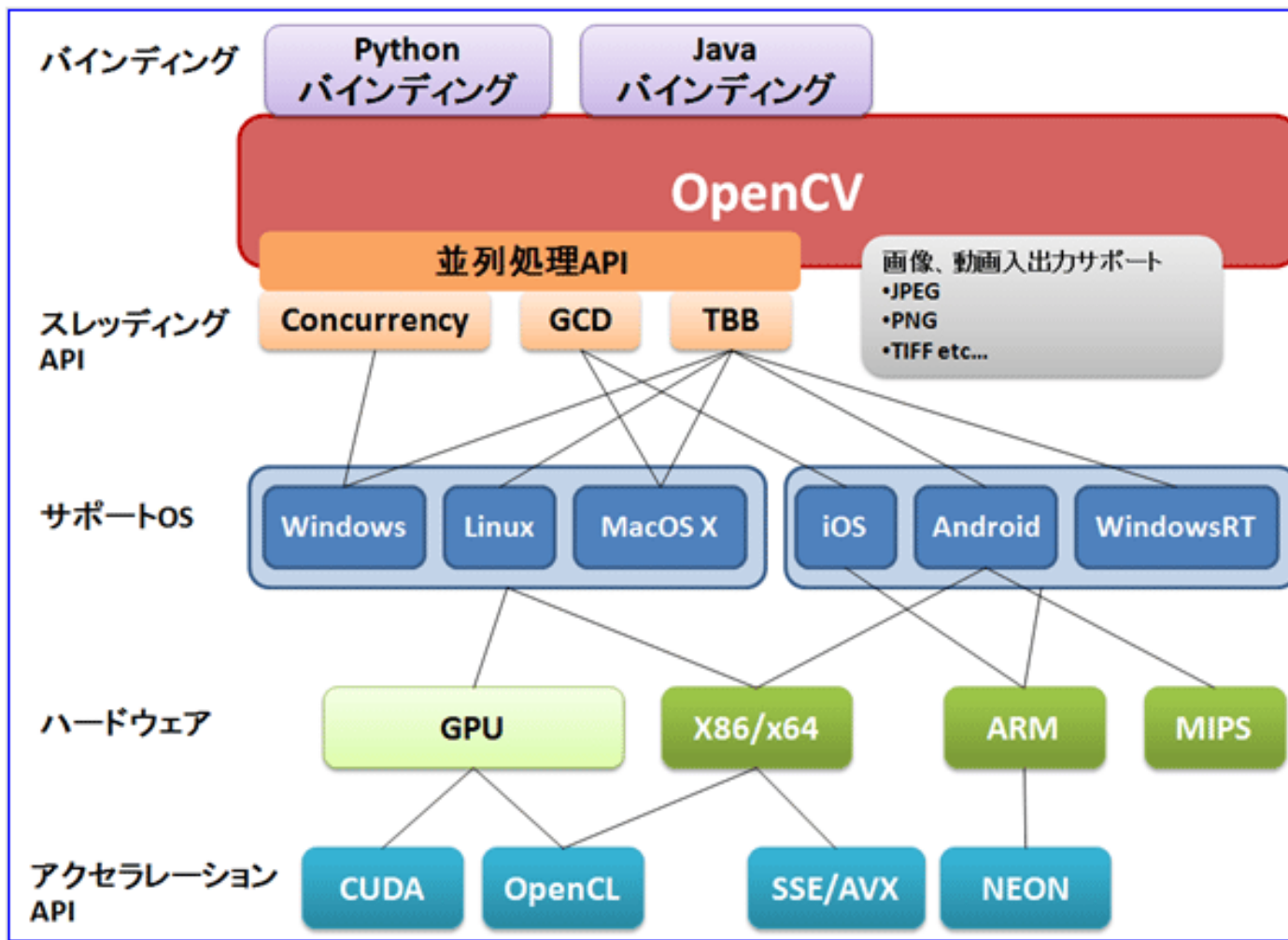
- 今回、使用するパッケージ一覧
  - OpenCV:libopencv-dev python-opencv # 538MB
  - 仮想X-Window : xvfb imagemagick # 80MB
  - Webサーバ : apache2 php # 17MB
- 全てまとめてインストール

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

- テキストファイルから1行ずつコピーしてください
- 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

# その他のOpenCV動作環境

- 動作検証済み環境

- Windows7 Professional
- Mac OS X
- Linux CentOS7, Ubuntu 16.04

- インストールするソフトウェア

- Python 2.7.13
- Python用ライブラリ numpy
- OpenCV 2.4.11

導入が簡単な2.4系を使用  
最新は3.2

# 導入手順：Windows7

- Python 2.7.14
  - <https://www.python.org/downloads/windows/>
    - 「Python 2.7.14」をダウンロードして、インストール
      - **注意**：32bit版（X86 MSIの方）を選択。Numpyが32bit のみのため
    - 「Install just for me」を選択
    - インストール先は、デフォルトの「C:¥Python27」
- numpy
  - <http://sourceforge.net/projects/numpy/files/NumPy/1.9.2/>
    - 「numpy-1.9.2-win32-superpack-python2.7.exe」をダウンロードして、インストール
- OpenCV
  - <http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.11/>
  - Windows専用版。ダウンロードして、実行＝解凍
    - 解凍先は、c:¥temp を指定



# 実行手順：Windows7

- コマンドプロンプトから、以下のコマンドを実行

```
set PATH=C:¥Python27;%PATH%
```

```
set PYTHONPATH=C:¥temp¥opencv¥build¥python¥2.7¥x86
```

```
cd C:¥temp¥opencv¥sources¥samples¥python2
```

```
python facedetect.py
```

[ESC]キーで終了

解凍先を変えた場合は、パスを修正

# 導入手順：Mac OS X

- Xcodeの導入

- AppStoreからXcodeをインストール
- 1度Xcodeを起動して、ライセンス情報を承認

- Homebrewの導入

- [http://brew.sh/index\\_ja.html](http://brew.sh/index_ja.html)
- ターミナルから、以下のコマンドを実行

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

プロキシが必要な場合

```
export http_proxy=http://IPアドレス:ポート番号/  
export https_proxy=http://IPアドレス:ポート番号/
```

- OpenCVの導入

- ターミナルから、以下のコマンドを実行

```
brew install opencv
```

- 「Error: No available formula for opencv」と出たら、

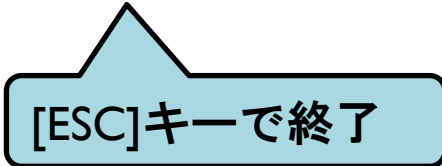
```
brew tap homebrew/science
```

1時間くらいかかる

# 実行手順：Mac OS X

- サンプルの実行環境を構築
  - <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.11/>
  - 「opencv-2.4.11.zip」をダウンロード（勝手に解凍される）
- ターミナルから、以下のコマンドを実行

```
cd Downloads/opencv-2.4.11/samples/python2/
python facedetect.py
```



[ESC]キーで終了

# 導入手順簡易版 : Linux CentOS7

- Python
  - 標準で入っている
- Numpy
  - 以下のコマンドを実行

```
yum install numpy
```

- OpenCV
  - 以下のコマンドを実行

```
yum install opencv*
```

- ・ 2.4.5と開発ライブラリがインストールされる
- ・ そのため、最新2.4.11のサンプルの一部は正常に動かない

プロキシが必要な場合  
/etc/yum.conf に追加  
proxy=http://IPアドレス:ポート番号/

# 導入手順最新版 : Linux CentOS7

- OpenCVの最新版をインストールする方法

- [公式手順書](#)

- 以下のコマンドを実行

```
sudo yum install cmake python-devel numpy gcc gcc-c++
```

```
sudo yum install gtk2-devel libdc1394-devel libv4l-devel ffmpeg-devel gstreamer-plugins-base-devel
```

- <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.11/>

- 「opencv-2.4.11.zip」をダウンロードして、ホームディレクトリに解凍

- 以下のコマンドを実行

```
cd $HOME/opencv-2.4.11
```

```
mkdir build
```

```
cd build
```

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local..
```

```
make
```

```
# sudo make install
```

30分くらいかかる

環境をできるだけ変更しないよう、実行しない

# 導入手順簡易版 : Linux Ubuntu 16.04

- Python + Numpy + OpenCV

```
sudo apt-get install libopencv-dev python-opencv
```

- 2.4.9がインストールされる
- そのため、最新2.4.11のサンプルの一部は正常に動かない
- 参考URL : <http://milq.github.io/install-opencv-ubuntu-debian/>

プロキシが必要な場合

```
export http_proxy=http://IPアドレス:ポート番号/  
export https_proxy=http://IPアドレス:ポート番号/
```

# 導入手順最新版 : Linux Ubuntu 16.04

- OpenCVの最新版をインストールする方法

- 公式手順書

- 以下のコマンドを実行

```
sudo apt-get install build-essential
```

```
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev  
libswscale-dev
```

```
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-  
dev libjasper-dev libdc1394-22-dev
```

- <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.11/>

- 「opencv-2.4.11.zip」をダウンロードして、ホームディレクトリに解凍

- 以下のコマンドを実行

```
cd $HOME/opencv-2.4.11
```

```
mkdir build
```

```
cd build
```

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D WITH_FFMPEG=OFF -D  
CMAKE_INSTALL_PREFIX=/usr/local ..
```

```
make
```

エラー無くビルドできるのは2.4.9

動画の扱いに制限が

これを付けないとmakeが失敗する

30分くらいかかる

# 実行手順：Linux CentOS & Ubuntu

- サンプルの実行環境を構築

- <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.11/>

- 「opencv-2.4.11.zip」をダウンロードして、ホームディレクトリに解凍

```
sudo yum install unzip
```

```
unzip opencv-2.4.11.zip
```

CentOSのみ

- 最新版を使いたい場合だけ、以下のコマンドを実行

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/opencv-2.4.11/build/lib
```

```
export PYTHONPATH=$PYTHONPATH:$HOME/opencv-2.4.11/build/lib
```

- Windows/MacからLinuxにssh接続している場合

- X-Window Serverが必要

```
export DISPLAY=Windows/Mac側のIPアドレス:0.0
```

- オススメツール：Xming（無償）

- <http://itcweb.cc.affrc.go.jp/affrit/documents/guide/x-window/x-win-xming>

- 以下のコマンドを実行

```
cd $HOME/opencv-2.4.11/samples/python2/
```

```
python facedetect.py
```

[ESC]キーで終了



# 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（操作不可）－ 1

- 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を起動（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

```
Xvfb :1 -screen 0 800x600x24 &
```

- 先頭のXは大文字

# ブラウザが使える環境（操作不可）－ 2

- 画面保存の動作テスト

```
export DISPLAY=:1
import -window root /var/www/html/viewx/img/capture.jpeg
```

- ブラウザに真っ黒な画像が表示されたら、成功

- ずっと画像を作り続ける（OSを起動するたび）

```
export DISPLAY=:1
while true; do import -window root /tmp/capture.jpeg ; mv /tmp/capture.jpeg /var/www/html/viewx/img; done &
```

- おかしくなったら `ps -x` で見て kill する
- もしくはラズパイを再起動

- これでX-Windowがブラウザで参照できます

- **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**
- **注意：Apacheのログが大量に貯まります**

# ブラウザが使える環境（操作不可） — 3

- 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]

- 顔認識

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

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

- ・ソースコードを見してみる
- ・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

# facedetector.py

- 顔と目を認識

- 解説ページ

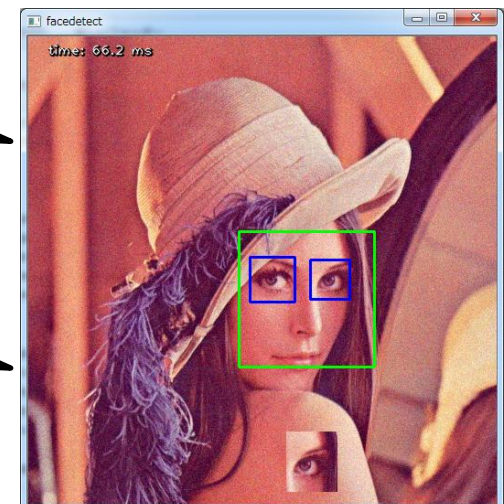
- 事前に「顔」「顔じゃない」を学習済み

- 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/>

顔の外に目があっても、認識しない作りになっている

sample/cpp/lena.jpg

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ニューズレター第3号  
の画像



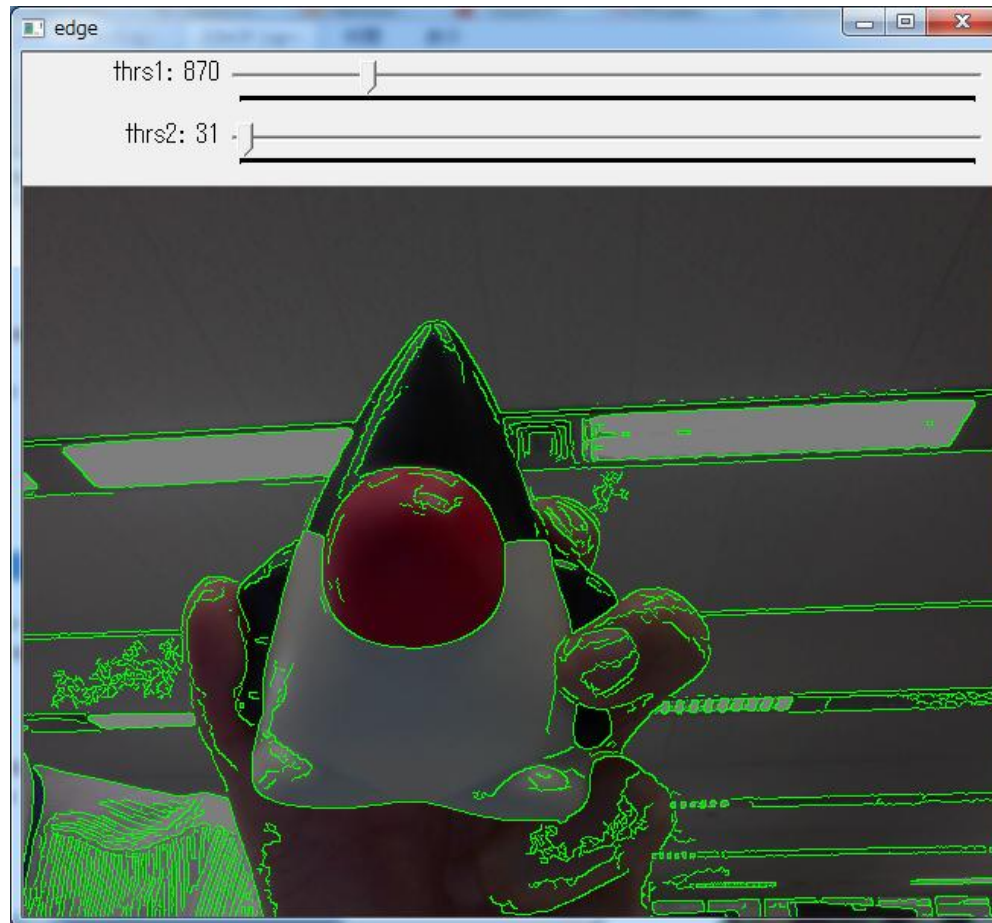
# digits.py

- **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

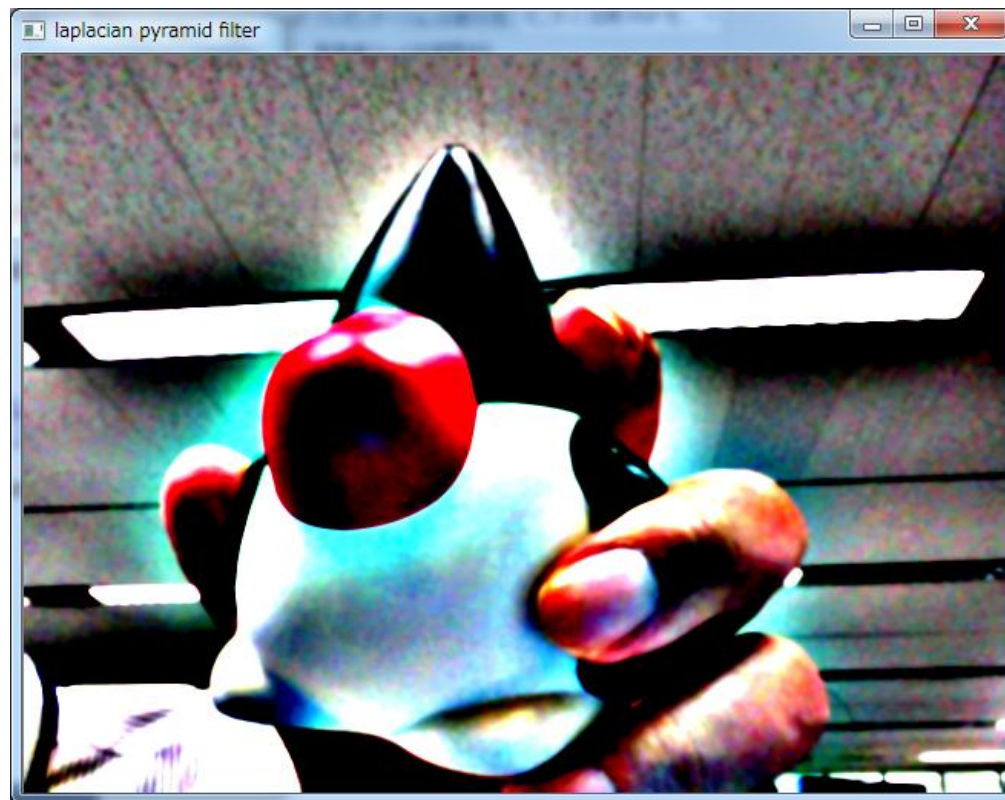
- 境界抽出

- 『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

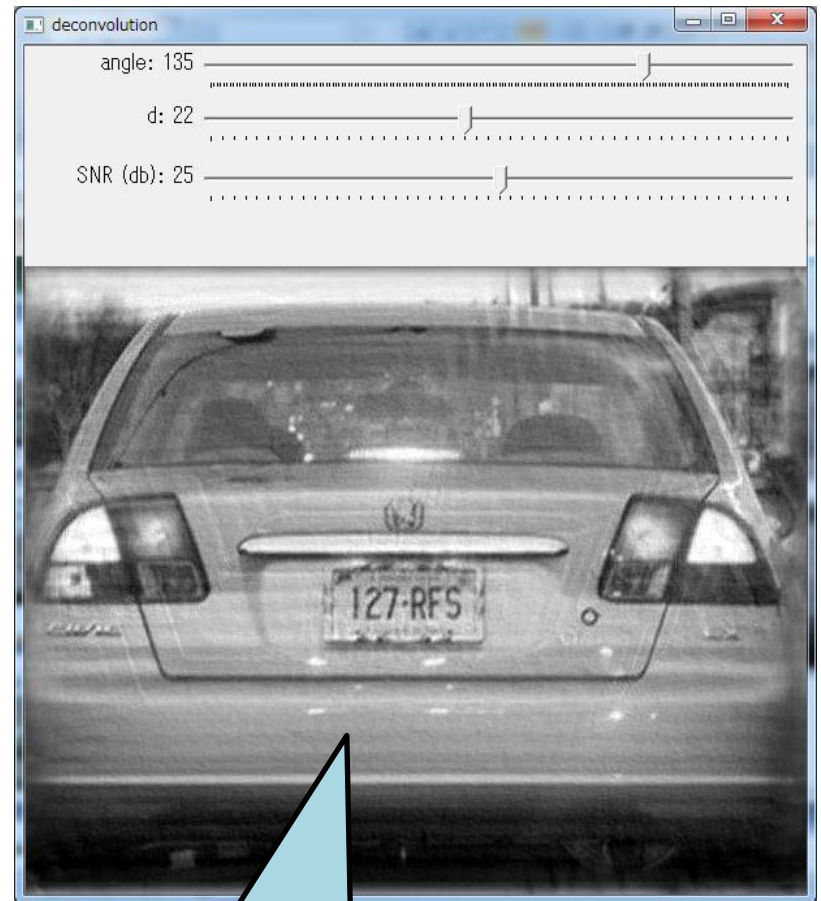


# deconvolution.py

- 「Convolution (畳みこみ) = ブレ」を解除する



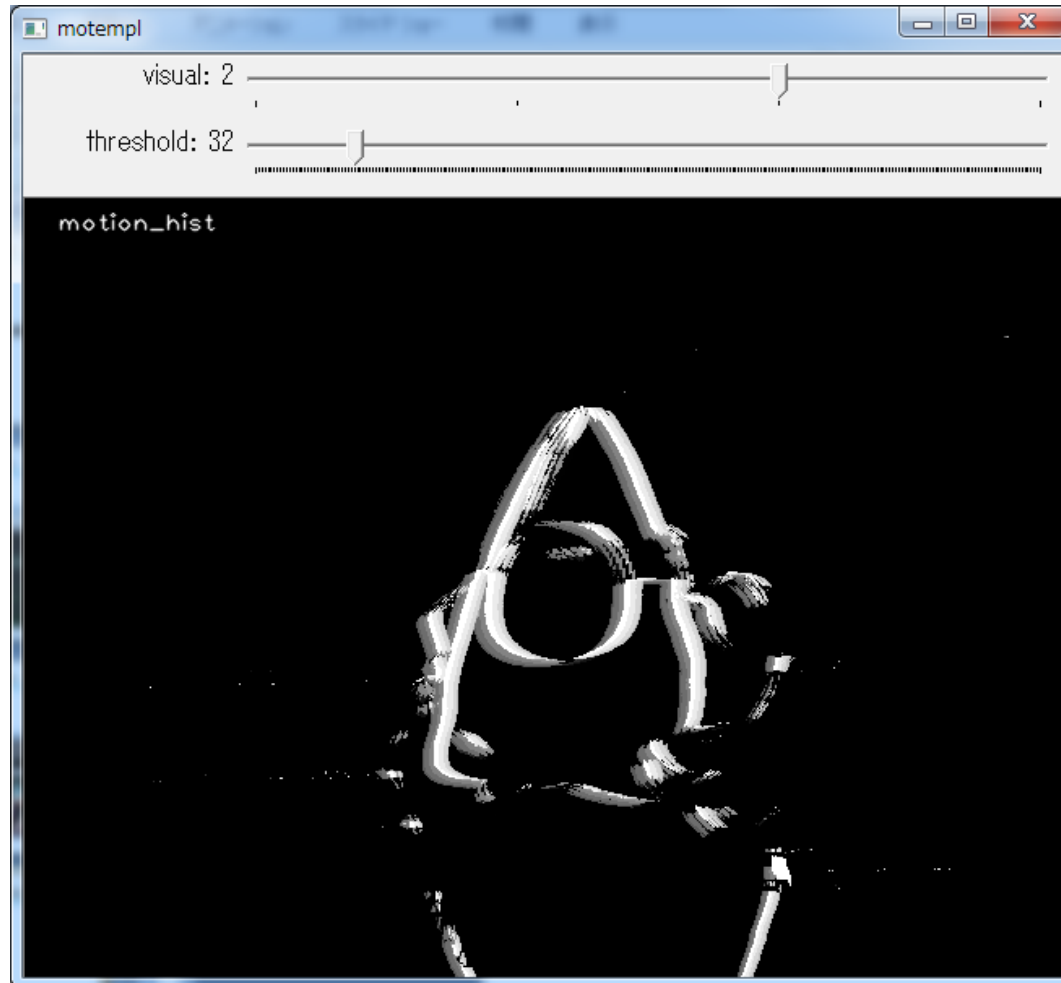
ナンバーが全く読めない



補正したら読めた

# motempl.py

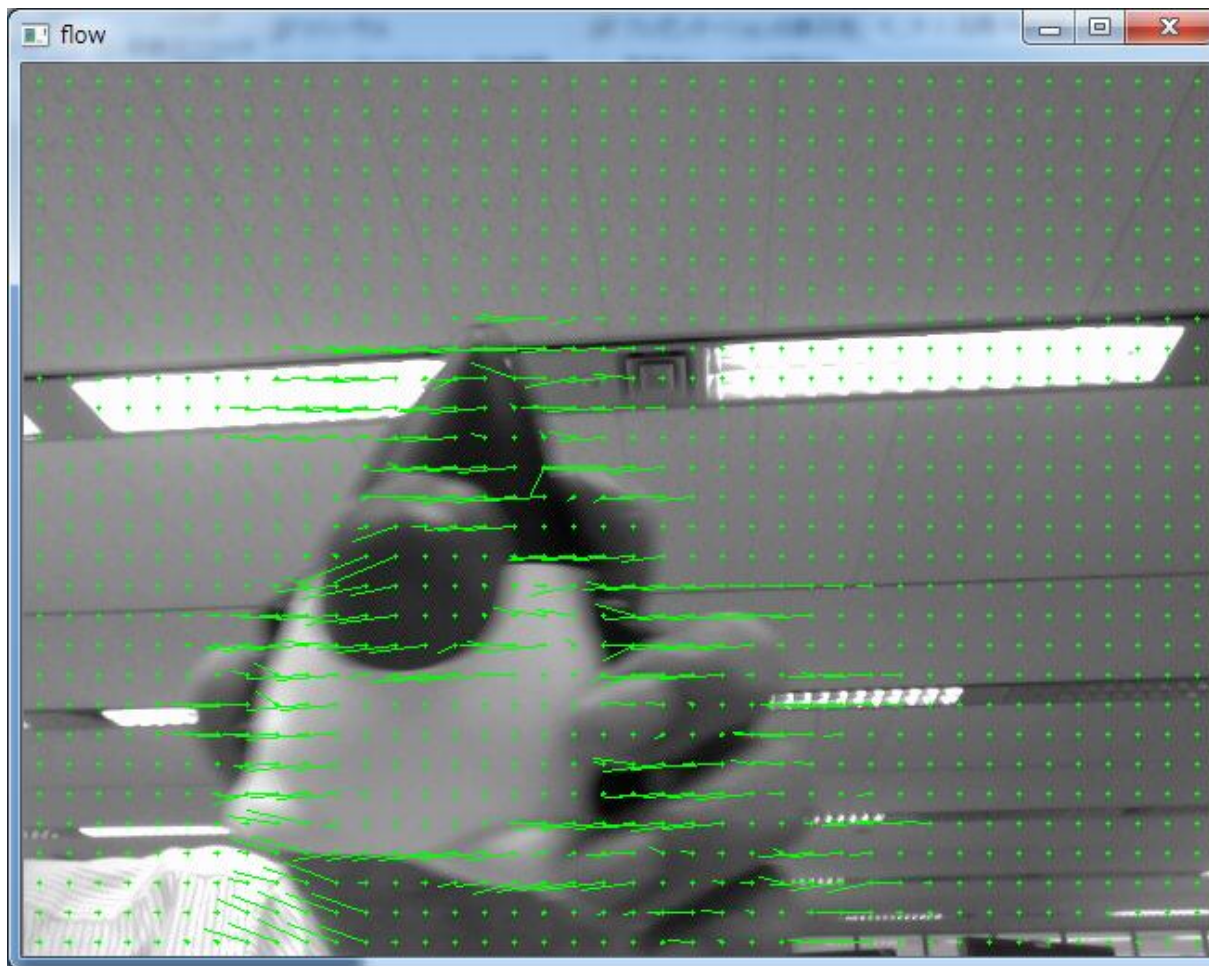
- 動画の変化部分を検出





# 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
  - 追いかけていたい色をマウスで範囲指定する

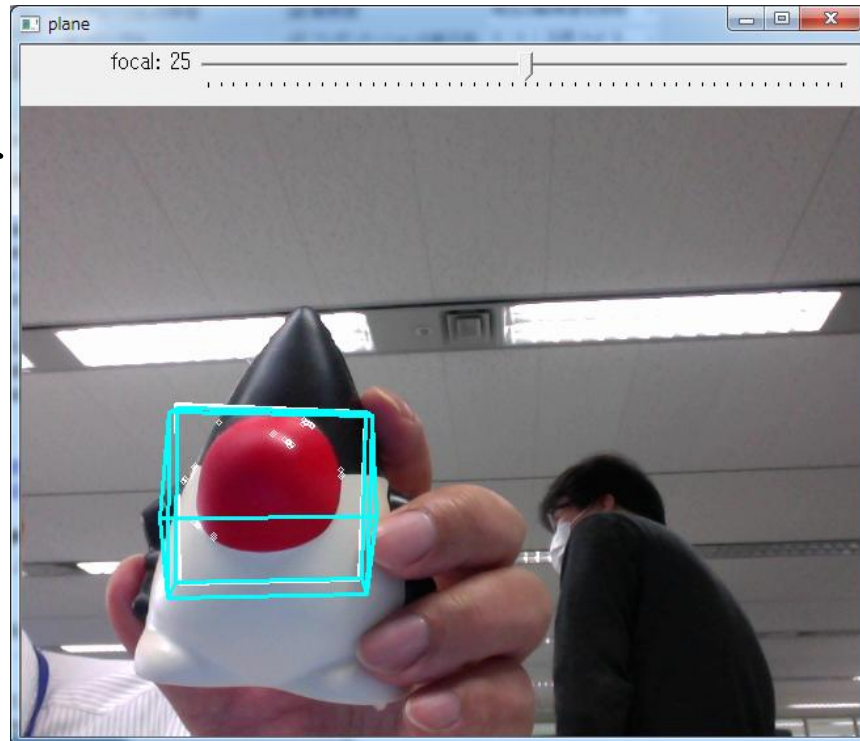
Dukeの赤い鼻を追いかけている



# plane\_ar.py

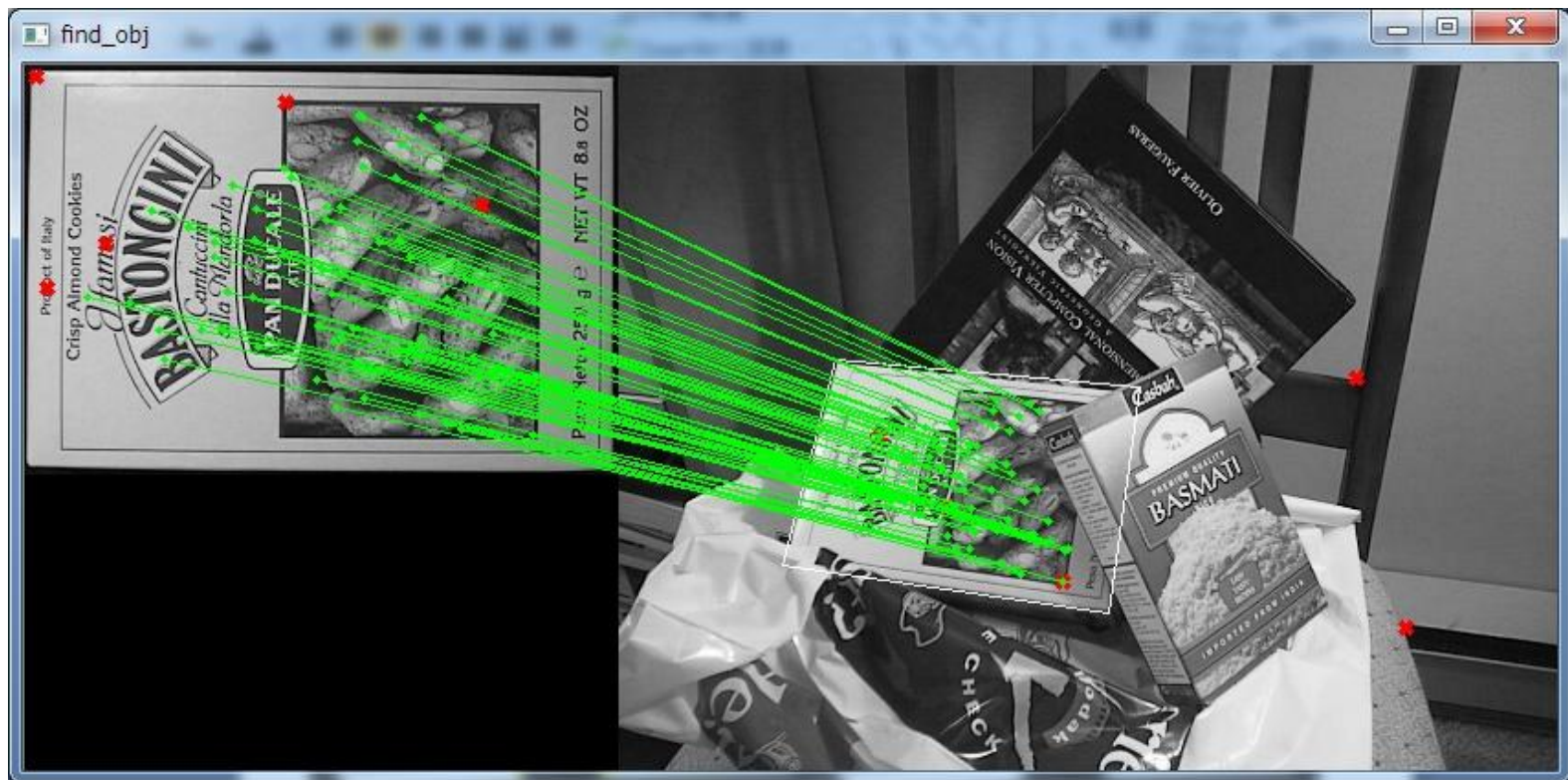
- ARマーカー無しでのAR
- 操作方法
  - ARマーカーとして使いたい領域をマウスで選択

Dukeの鼻を  
ARマーカーとして指定



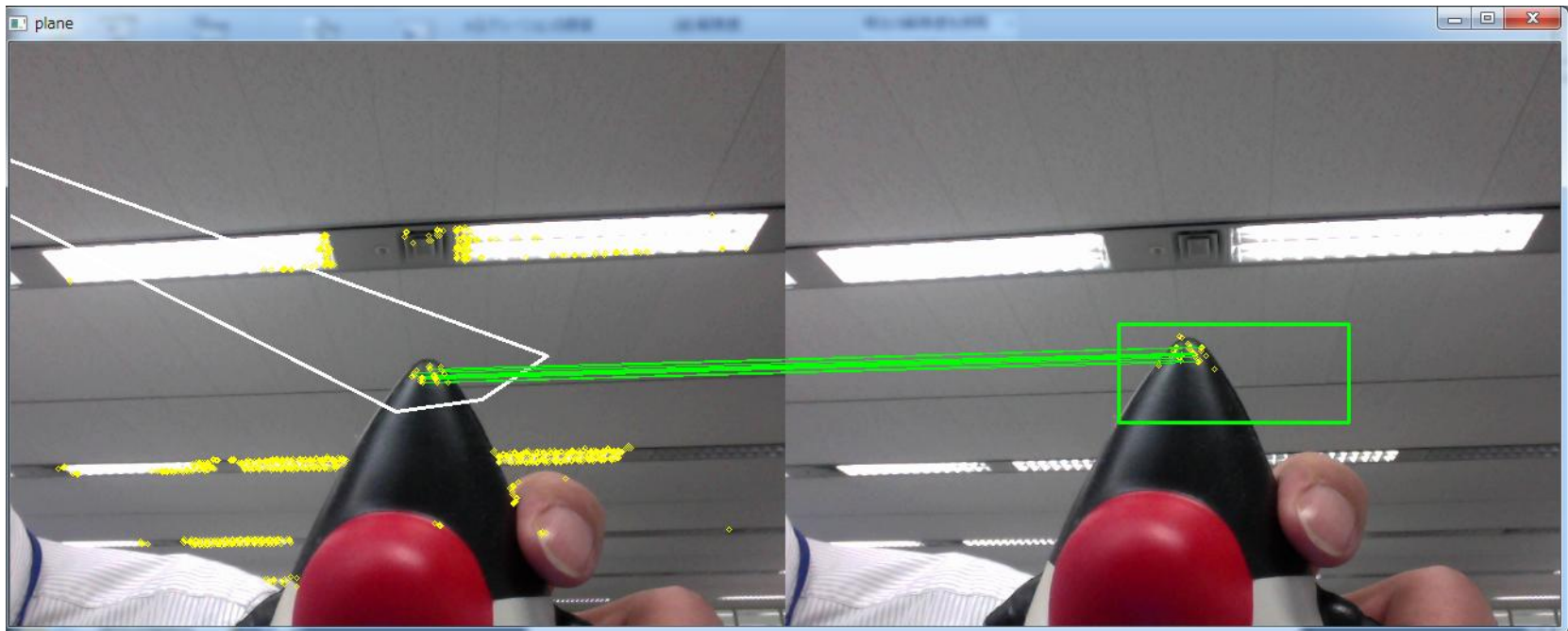
# find\_obj.py

- 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

# 顔認識サンプルを拡張していきます

- ・ 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ー1

- 元ファイルをコピー

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

- Webコンテンツを準備

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

- 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

```
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]);
```



# 顔画像をひらすら収集ー2

- Apacheを起動

```
sudo service apache2 start
```

- 実行

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

- ブラウザで参照

- <http://ラズパイのIPアドレス/faces/>

# 顔認識した数をクラウドに通知ー1

- 元ファイルをコピー

```
cp facedetect.py facecount.py
```

- 顔の数をWebAPIで通知する機能を追加

```
draw_rects(vis, rects, (0, 255, 0))
count = 0
for x1, y1, x2, y2 in rects:
    count += 1
    roi = gray[y1:y2, x1:x2]
    vis_roi = vis[y1:y2, x1:x2]
    subrects = detect(roi.copy(), nested)
    draw_rects(vis_roi, subrects, (255, 0, 0))
print count
url = "http://aramoto.sakura.ne.jp/aitc/?id=aramoto&val=" + str(count)
print url
import urllib
result = urllib.urlopen( url ).read()
dt = clock() - t
```

# 顔認識した数をクラウドに通知ー2

- 実行

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

- 以下のURLから結果を確認

- <http://aramoto.sakura.ne.jp/aitc/view2.php>

- 自分が指定した id を選択
- 前回の温度センサの値を送付も可

- 画像をクラウド上にアップする方法

- python から scp コマンドを実行

- sshpass でパスワード入力を省略

例： “sshpass -p パスワード scp ...../img/” + now + “.jpg  
aramoto@aramoto.sakura.ne.jp:./www/faces/img/” 43

# カメラ画像を漫画風に変換ー1

- 以下のサイトを参考にする

- <https://algorithm.joho.info/programming/python/opencv-manga-camera-py/>

- 元ファイルをコピー

```
cp facedetect.py manga.py
```

- スクリプトをダウンロード

```
wget https://algorithm.joho.info/wp-content/sample/Python/opencv/work/manga-camera/screen.jpg
```

- 漫画化フィルタを上記サイトからコピーする

```
# 漫画化フィルタ  
def manga_filter(src, screen, th1=60, th2=150):
```

```
    # グレースケール変換
```

```
    :
```

```
    # 三値画像と輪郭画像を合成
```

```
    return cv2.bitwise_and(gray, edge)
```

} この関数全体

# カメラ画像を漫画風に変換ー2

- 日本語のコメントがエラーにならないように

```
#!/usr/bin/env python  
# -*- coding: utf-8 -*-
```

- 漫画化フィルタを呼び出す

```
while True:  
    ret, img = cam.read()  
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
    gray = cv2.equalizeHist(gray)  
  
    screen = cv2.imread("screen.jpg")  
    img = manga_filter(img, screen, 60, 150)  
  
    t = clock()
```

- 実行

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

# カメラ画像を漫画風に変換ー3

- 漫画化した画像全体を保存する
  - 撮った顔と混ぜたくない人は、パスを変更

```
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);

roi = gray[y1:y2, x1:x2]
vis_roi = vis[y1:y2, x1:x2]
```

- 実行

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

- ブラウザで参照
  - <http://ラズパイのIPアドレス/faces/>