

MATLAB+Pythonで簡単機械学習

一般社団法人二足歩行ロボット協会
西村輝一

機械学習

OpenCVで物体検出器を作成する

https://www.pro-s.co.jp/engineerblog/opencv/post_6202.html

- 学習
 - 物体を検出する際、あらかじめ画像から、特徴量を抽出する。
- 推論/検証
 - 学習済の特徴に合う部分がないかを画像上を探索する。

カスケード分類器

haartraining (ハールトレーニング)

Haar-Like特徴による機械学習

trainscascade (トレインカスケード)

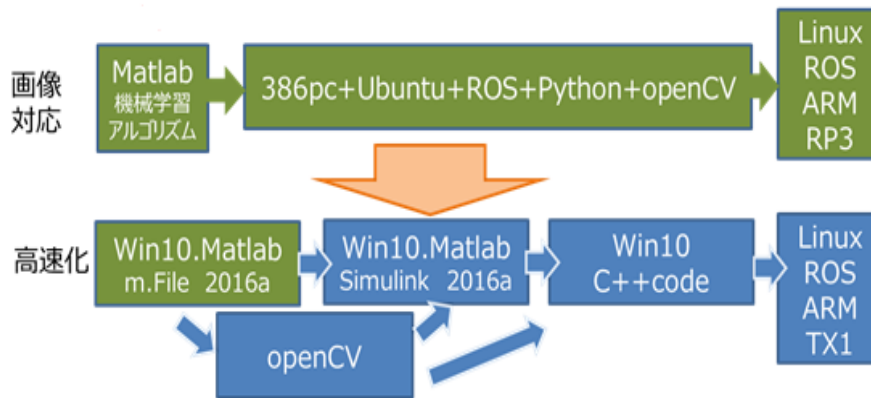
haartrainingのあとに作られたもの

Haar-Like特徴・LBP特徴・HOG特徴から1つを選択

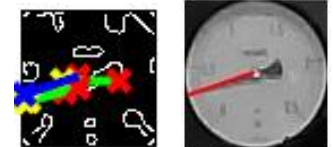
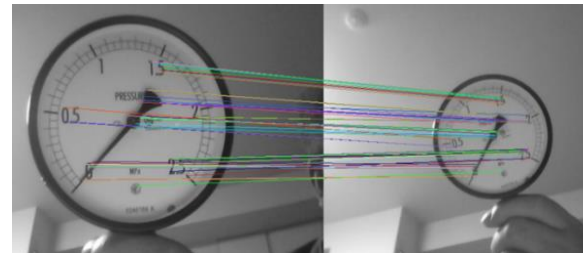
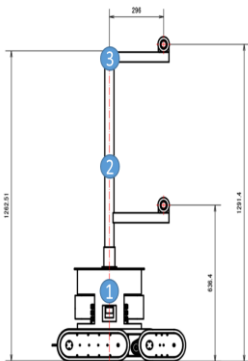
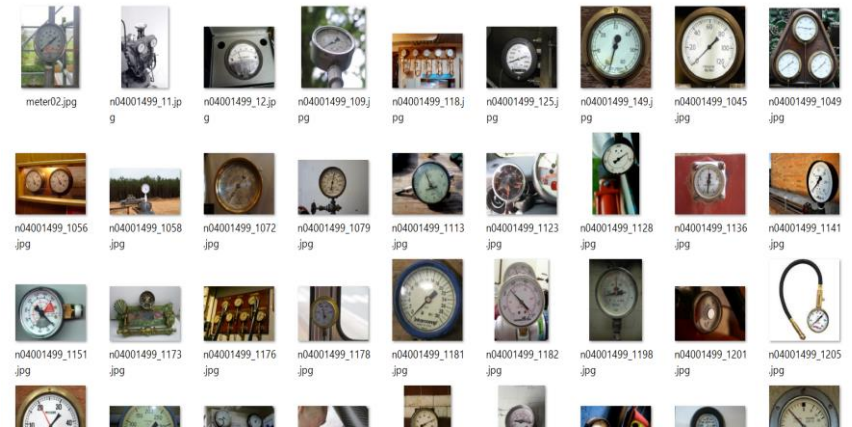
- 正解画像は7000枚以上、不正解画像は3000枚以上

ARGOS Challenge

ROS, OpenCV, Pythonの活用



機械学習(カスケード分類器)



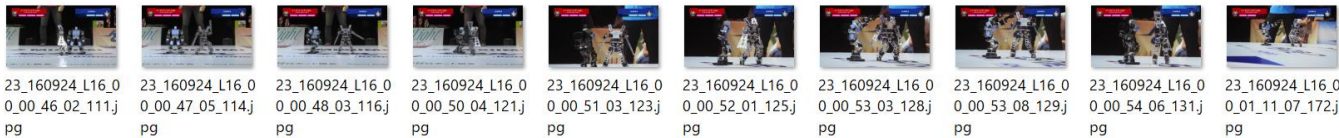
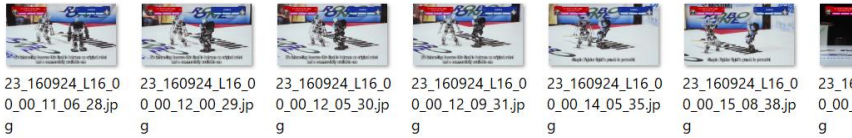
具体的にはどうすれば良いのか？

- MATLABをインストール
 - 仕様
 - 2017a
 - Windows 64bit
- 画像を集める。
 - 映像から切り出し。
- Labelerでラベリング
 - 正解画像と不正解画像を準備
- MATLABで学習
 - 学習と検証
- 推論/検出を行う。
 - MATLAB
 - Python+OpenCV

画像を集める。

正解画像 (ポジティブ画像)

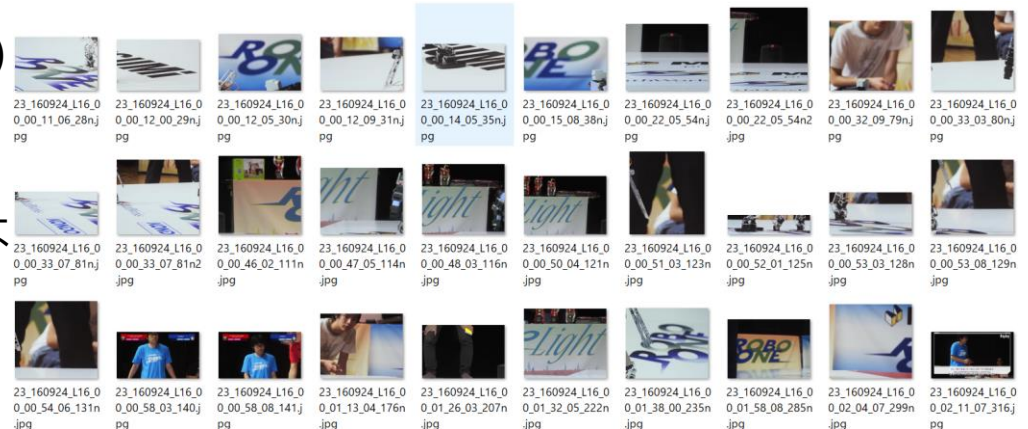
└ 検出させたい物体が映っている画像



不正解画像 (ネガティブ画像、背景画像)

└ 検出させたい物体が映っていない画像

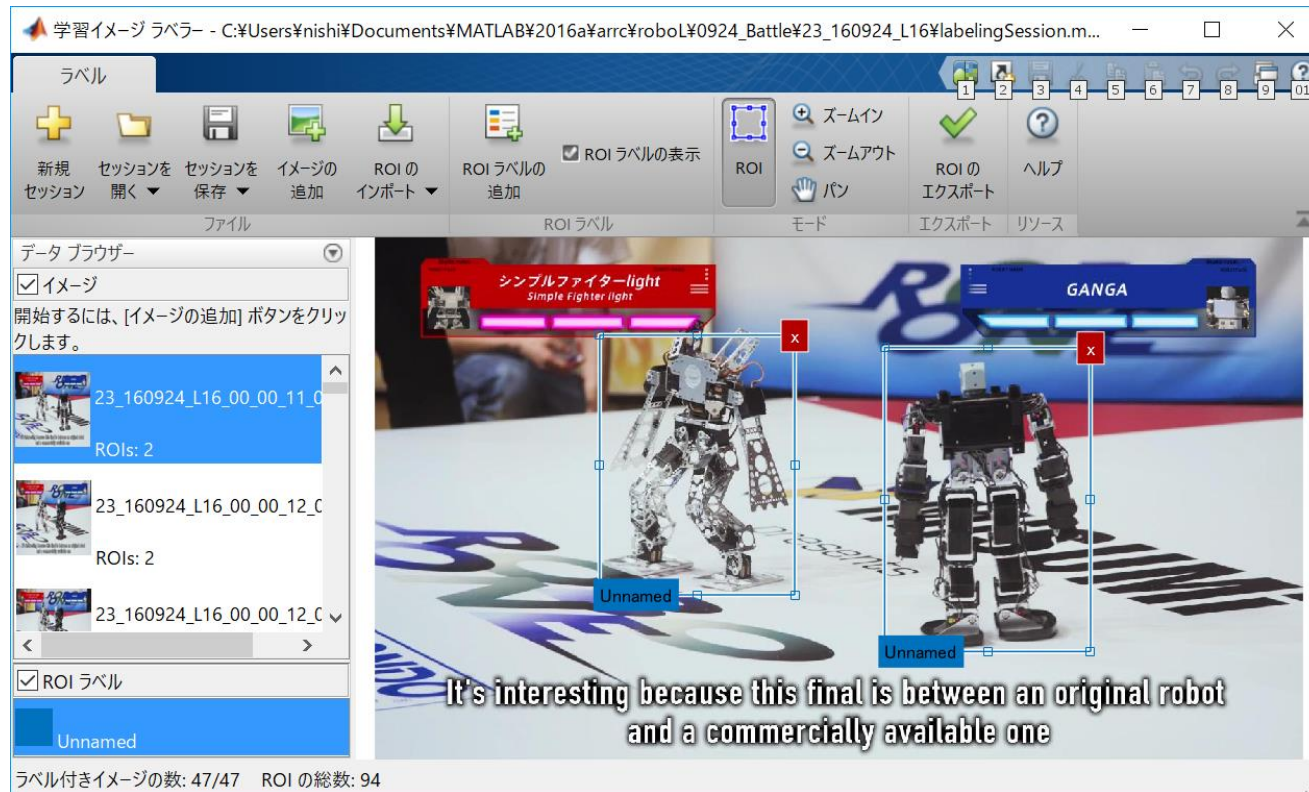
正解画像は7000枚以上、不正解画像は3000枚以上あると良いようですが今回は1/100程度で行った。



MATLAB Labelerでラベリング

アプリケーションの学習イメージラベラーを使う。

実演



全ての画像にROIを指定し、ROIのエクスポートを構造体で行う。

トレーニング

実演

Negativeフォルダーを指定して学習を開始するプログラム

```
%% Neg画像のフォルダの指定
negFolder = 'C:\Users\%niishi%\Documents\MATLAB\2016a\%arc\%roboL\%0924_Battle\%23_160924_L16\%Negative'
winopen(negFolder); % フォルダを開いて画像を確認（実際は数百枚以上の画像を使用）

%% 学習の実行（学習結果が、XML形式のファイルで生成される）
% データベースファイル： roboj.xml が生成される
% 実際はもっと多くの画像で学習

trainCascadeObjectDetector('roboj.xml', positiveInstances, negFolder, 'FalseAlarmRate', 0.1, 'NumCascadeStages', 10);
```

trainCascadeObjectDetectorの説明

```
%% 学習の実行
% ObjectTrainingSize : トレーニングの前に、pos/negサンプルは、このサイズへリサイズされる。(デフォルト : Auto)
% このサイズが、検出時の最小サイズ。小さくすると学習速度上がる。
% 各ステージで使われるNegサンプル数 = NegativeSamplesFactor x ステージで使われたPosサンプル数
% NumCascadeStages : カスケードステージの数
% FalseAlarmRate : 各ステージでのacceptable(max) false positive rate (デフォルト : 0.5) : このステージ数乗=>十分小さなfalse positive rate
% TruePositiveRate : 各ステージでの最小 true positive rate (デフォルト : 0.995) : このステージ数乗が最終rate

% false positive: NegサンプルがPosと検出 => 以降のStageで正しくNegと検出できればよい
% false negative: PosサンプルをNegと検出 -> 低くなるように (TruePositiveRateをあげると低くなる)

% トータルのfalse positive rate =  $f^{\text{NumCascadeStages}}$  : ステージ数増やすと下がり良くなる
% トータルの true positive rate =  $t^{\text{NumCascadeStages}}$  : ステージ数増やすと下がり悪くなる

% [Posサンプル]
% 各ステージで使われる Posサンプル = 総Posサンプル数 / (1 + (NumCascadeStages-1)*(1-TruePosRate)) で計算される

% [Negサンプル]
% 前までのステージで出来たDetectorで、Neg『画像』を認識
% => 検出したfalse positive => 新しいステージの学習のNegサンプルとして使う
% => 一つも検出(false positive)なし => そのNeg画像は、一つもNegサンプルが含まれないので以降使わない

% 例)
% トレーニングデータ多いとき : false positive rateを増やし、かつ、Stage数を増やす
% トレーニングデータ少ないとき : false positive rateを下げ、かつ、Stage数を減らす

% trainCascadeObjectDetector('stopSignDetector.xml', data, negFolder, 'FalseAlarmRate', 0.2, 'NumCascadeStages', 5);
```

特徴量

MATLAB2016a

```
<?xml version="1.0"?>
<opencv_storage>
<!-- Created using Computer Vision System Toolbox(
<!-- Version 9.0.0.341360 (R2016a) -->
<!-- Compatible with OpenCV 2.4 -->
<cascade>
<stageType>BOOST</stageType>
<featureType>HOG</featureType>
<height>32</height>
<width>32</width>
<stageParams>
<boostType>GAB</boostType>
<minHitRate>9.9500000476837158e-01</minHitRa
<maxFalseAlarm>2.0000000298023224e-01</maxF
<weightTrimRate>9.499999999999996e-01</weig
<maxDepth>1</maxDepth>
<maxWeakCount>100</maxWeakCount></stagePa
<featureParams>
<maxCatCount>0</maxCatCount>
<featSize>36</featSize></featureParams>
<stageNum>5</stageNum>
<stages>
<!-- stage 0 -->
<>
<maxWeakCount>3</maxWeakCount>
<stageThreshold>-9.3406349420547485e-01</st
<weakClassifiers>
<>
<internalNodes>
0 -1 17 4.8831082880496979e-02</internalN
<leafValues>
-8.9519649744033813e-01 8.3636361360549
<>
<internalNodes>
0 -1 23 3.7281963974237442e-02</internalNodes>
```

MATLAB2017a

```
<?xml version="1.0"?>
<opencv_storage>
<!-- Created using Computer Vision System Toolbox(tm) for MATLAB(R) -->
<!-- Version 9.2.0.556344 (R2017a) -->
<!-- Compatible with OpenCV 3.1 -->
<cascade>
<stageType>BOOST</stageType>
<featureType>HOG</featureType>
<height>45</height>
<width>32</width>
<stageParams>
<boostType>GAB</boostType>
<minHitRate>9.9500000476837158e-01</minHitRate>
<maxFalseAlarm>1.0000000149011612e-01</maxFalseAlarm>
<weightTrimRate>9.499999999999996e-01</weightTrimRate>
<maxDepth>1</maxDepth>
<maxWeakCount>100</maxWeakCount></stageParams>
<featureParams>
<maxCatCount>0</maxCatCount>
<featSize>36</featSize></featureParams>
<stageNum>8</stageNum>
<stages>
<!-- stage 0 -->
<>
<maxWeakCount>2</maxWeakCount>
<stageThreshold>-3.1767034530639648e-01</stageThreshold>
<weakClassifiers>
<>
<internalNodes>
0 -1 34 1.4037833549082279e-02</internalNodes>
<leafValues>
-9.7777777910232544e-01 8.0392158031463623e-01</leafValues></>
<>
<internalNodes>
0 -1 38 1.0892081074416637e-02</internalNodes>
<leafValues>
-9.7785842418670654e-01 6.6010743379592896e-01</leafValues></>
</weakClassifiers></>
```


検証

%% 未知の画像の読み込み

```
I = imread('test.jpg');
```

```
figure; imshow(I);
```

%% 物体認識オブジェクトの定義、実行 [2行のMATLABコード]

```
detector = vision.CascadeObjectDetector('roboj.xml');
```

```
robos = step(detector, I)
```

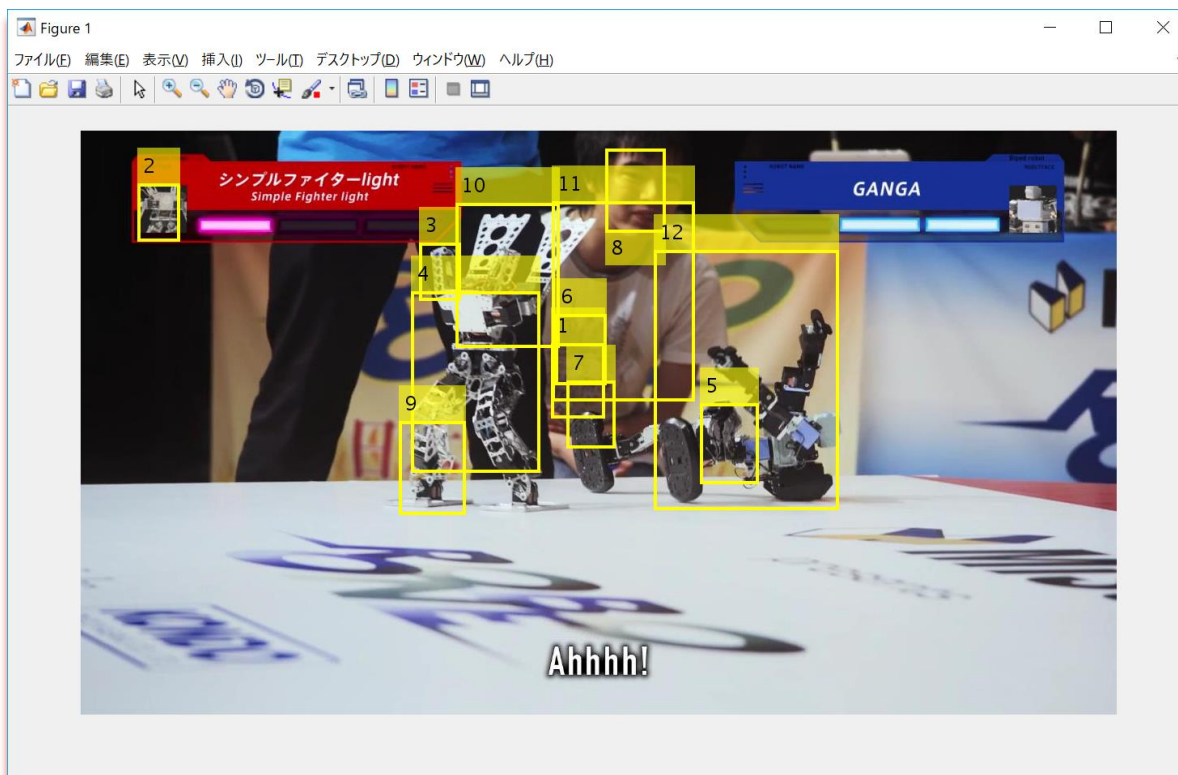
%% 検出された車の位置に、四角い枠とテキストを追加

```
I2 = insertObjectAnnotation(I, 'rectangle', robos, [1:size(robos,1)], 'FontSize', 24,
```

```
'LineWidth', 4);
```

```
imshow(I2); shg;
```

```
release(detector);
```



黄色の枠で囲まれたところがロボットと判断された。

Pythonを使ってみる

実演

Anacondaをインストール

<https://www.continuum.io/>からダウンロード。

Windowsでも使えます。

Anaconda 4.3.1

For Windows

Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

Changelog

1. Download the installer
2. Optional: Verify data integrity with MD5 or SHA-256 [More info](#)
3. Double-click the .exe file to install Anaconda and follow the instructions on the screen

Behind a firewall? Use these [zipped Windows installers](#)

Python 3.6 version

64-BIT INSTALLER (422M)

32-BIT INSTALLER (348M)

Python 2.7 version

64-BIT INSTALLER (414M)

Spyderを使うとMATLABな感じ。
NumPyとmatplotlibなどの
ライブラリーが有難い。

The screenshot shows the Spyder Python IDE interface. The main window displays a Python script with the following code:

```
3 import matplotlib.pyplot as plt
4 from numpy import*
5 import cv2
6 import matplotlib.pyplot as plt
7 import matplotlib.image as mpimg
8 import numpy as np
9
10 def distance_1(a,b,c):
11
12     u = np.array([b[0]-a[0],b[1]-a[1]])
13     v = np.array([c[0]-a[0],c[1]-a[1]])
14     L = abs(cross(u,v)/linalg.norm(u))
15
16     return L
17
18 if __name__ == '__main__':
19     # img = cv2.imread('checkpoint3.png')
20     # cv2.imshow('Source Image',img)
21     # cv2.imwrite('Image4.jpg',img)
22     #img=mpimg.imread('c:/Users/nishi/Documents/Python Scripts/imageData/cpimg/Lev
23     #img=mpimg.imread('stinkbug.png')
24     #imgplot = plt.imshow(img)
25     # 点a,b,cの定義
26     a = (0,0)
27     b = (2,2)
28     c = (0.5,1.5)
29
30     # 点a,bを結ぶ直線と点cとの最短距離
31     print "H=",distance_1(a,b,c)
32
33     # グラフ表示
34     plt.plot([a[0],b[0]],[a[1],b[1]],"r-o")
35     plt.plot([c[0],[c[1]],"ro")
36     plt.grid()
37     plt.show()
38
```

The right-hand side of the IDE shows a file explorer with a list of folders and files, and a Python console at the bottom displaying the output of the script:

```
In [2]: runfile('C:/Users/nishi/Documents/Python Scripts/kyori.py', wdir='C:/Users/nishi/Documents/Python Scripts')
H= 0.707106781187
```

Below the console, a small plot is visible, showing a red line segment connecting two points (a and b) and a red circle representing point c. The plot is a 2D coordinate system with x and y axes ranging from 0 to 100.

カメラ画像から認識

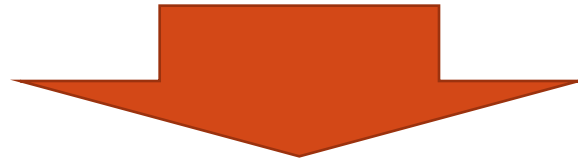
図はSpyder画面

PythonからOpenCVライブラリーを呼び、PCのカメラでROBO-ONEポスターのロボットをリアルタイムで認識させてみた。



まとめ

- MATLABのM-fileとPythonのコードは似ている。
- MATLABの画像処理関数と同じものがOpenCVにもある。
- MATLABのHMI環境は整っている。
- Python, OpenCVは不安、英文での情報が豊富だが・・・



- 企業の方にもこれから就職する方にもMATLAB+Pythonはお勧めです。
- ROBO-ONE用カスケード分類器が増えることを期待。
- さらなる高速で深層学習もROBO-ONE実装に期待。