

従来法による画像・映像処理

今回の内容

- Google ColabによるOpenCV画像処理
- CondaによるPython環境構築
- 実機によるOpenCV画像処理

OpenCV



- 1999年よりIntelが開発している画像・映像処理ライブラリ
- Colabではプリインストール済み。Python環境では追加でインストールが必要
- BSDライセンスによるオープンソース
- 色変換、エッジ抽出、モザイク加工などの基本的なことから、顔や物体の検出、ラベリング、姿勢推定など様々な処理が可能
- チュートリアルも非常に豊富：
https://docs.opencv.org/master/d9/df8/tutorial_root.html

Google ColabによるOpenCV画像処理

演習ファイルのColabフォルダ内をすべてColabにコピー

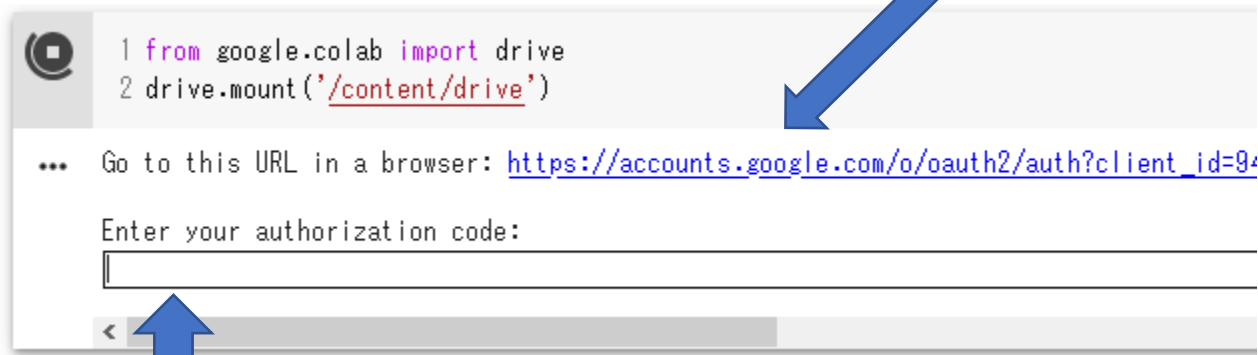
- Google Driveのマウント
- 画像の表示、回転、ぼかし、モザイク
- 顔や人の認識、猫の認識
- Webカメラの利用

ColabでのGoogle Driveのマウント

- 次のコードを実行し、認証コードを取得・入力

```
from google.colab import drive  
drive.mount('/content/drive')
```

クリックし、承認する



The image shows a Colab interface. At the top, there is a code editor with two lines of Python code: `1 from google.colab import drive` and `2 drive.mount('/content/drive')`. Below the code editor, there is a message that says "Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=94...". Below the URL, there is a text input field labeled "Enter your authorization code:". A blue arrow points from the text "クリックし、承認する" to the URL. Another blue arrow points from the text "発行された認証コードをここに入力" to the input field.

発行された認証コードをここに入力

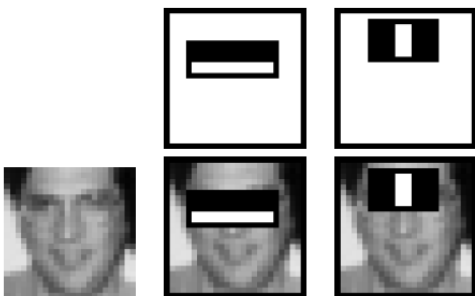
Google Drive と Google Colab の ファイル

- 特にパスを指定しなければColabのVM上にファイルは保存
- パスを指定することで、DriveとColabに分けて保存可能
- もちろんColabに保存すると12時間後には消滅
- Colab: [/content](#)
- Drive: [/content/drive/My drive](#)



OpenCVによる画像識別

- ハール(Haar)特徴量にもとづく **Cascade** 識別器を使用
- P. Viola (2001)によるオブジェクト検出の研究と R. Lienhart (2002)による改良がベースになっている
- 白と黒で構成される矩形を画像に適用し、対象画像の特徴量を作成する
- 例えば顔画像は、目の領域の画素は周辺よりも暗い、口の領域の画素が周辺より明るい、などの特徴が得られる
- 顔、目、正面、上半身、下半身、笑顔、などの識別器が OpenCVで利用できる



Sample face detection from, Paul Viola and Michael J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE CVPR, 2001.

CondaによるPython環境構築

- 別紙参照
- Minicondaをインストールし、Python環境を構築する
- Minicondaは、任意のバージョンのPython環境を構築できる一種の仮想実行環境
- Anaconda > Conda > Miniconda の順番でパッケージが大きいが、最低限の用途ならMinicondaがコンパクトで良い
- Minicondaをインストールし、Python環境を構築、必要なパッケージをpipコマンドでインストールする

PC上でのPythonの利用(Miniconda)

- Anaconda Promptから、`conda activate <環境名>`
- プロンプトが (base) から (環境名) となったことを確認できたら、`python <スクリプト>` で実行
- 遠隔受講でPythonからWebカメラを使用する場合は、必ずZoomのカメラをOFFにすること



PCで実行するファイル

開始前に、Colabフォルダ前のmodelフォルダをそのままPC内にコピーしておく

- 画像表示：[image1.py](#), [image2.py](#), [image3.py](#)
 - [image1.py](#): システムの関連付けで表示
 - [image2.py](#): Matplotlibで表示
 - [image3.py](#): OpenCVで表示
- Webカメラ(モザイク、エッジ)：[webcam.py](#)
- 顔と目の検出(Webcam)：[face_eye.py](#)
- 顔領域の書き出し(Webcam)：[face_file.py](#)
- 人間検出(Webcam)：[person.py](#)
- 動体検出(Webcam)：[detect.py](#)

OpenCVによる人間の検出

- HoG特徴量 + SVMによる識別器
- 顔はだいたい誰も明暗差が変わらないが、人間全体だと服装によって明暗差が異なるため、輪郭情報を主に使用する
- HoG特徴量は、画像の輝度の勾配を主にパラメータとして用いる
- 複数人の検出も可能だが、精度はそれほどよくない & かなり処理が重い
- `getDaimlerPeopleDetector`と`getDefaultPeopleDetector`の2つの処理方法があり、ソースによってどちらが向いているか若干異なる

OpenCVによる動体検出

- Webカメラからの入力のうち、動いている部分を検出
- フレーム間差分を計算し、差が大きいところが動いている部分になる
- 参考：
<https://ensekitt.hatenablog.com/entry/2018/06/11/200000>

参考：自分で識別器を作成

- 一種の教師あり学習
- 正解画像と不正解画像を用意し、モデルを構築

例：

- <https://premium.aidemy.net/magazine/entry/2018/05/21/210100>
- <https://qiita.com/penstood/items/97421a91d3f4075d39a4>